HALO (Highly Addictive, sociaLly Optimized) Software Engineering

Swapneel Sheth, Jonathan Bell, Gail Kaiser
Department of Computer Science, Columbia University, New York, NY 10027
{swapneel, jbell, kaiser}@cs.columbia.edu

ABSTRACT

In recent years, computer games have become increasingly social and collaborative in nature. Massively multiplayer online games, in which a large number of players collaborate with each other to achieve common goals in the game, have become extremely pervasive. By working together towards a common goal, players become more engrossed in the game. In everyday work environments, this sort of engagement would be beneficial, and is often sought out. We propose an approach to software engineering called HALO that builds upon the properties found in popular games, by turning work into a game environment. Our proposed approach can be viewed as a model for a family of prospective games that would support the software development process. Utilizing operant conditioning and flow theory, we create an immersive software development environment conducive to increased productivity. We describe the mechanics of HALO and how it could fit into typical software engineering processes.

Categories and Subject Descriptors

K.6.3 [**Management Of Computing And Information Systems**]: Software Management—*Software Process*; D.2.6 [**Software Engineering**]: Programming Environments—*Interactive environments*; D.2.9 [**Software Engineering**]: Management—*Productivity*

General Terms

Human Factors, Management

Keywords

Web 2.0, Games, MMORPG, Quests, Flow, Operant Conditioning, Social Rewards

1. INTRODUCTION

Social games have changed the nature of the gaming industry. Traditionally, computer games have been viewed as competitive (me versus the rest of the world; me versus the computer AI). However, with the ubiquitousness of Internet access, social games - games where players collaborate with each other for a common goal - have become increasingly popular. The first examples of social games were Multi-User Dungeon (MUD) games in the 1970s such as MUD1 [20] and Scepter of Goth [12]. In the early 2000s, massively multiplayer online games ("MMOGs") such as World of Warcraft (an "MMORPG," or MMO role playing game) [2] and Second Life [13] (an "MMOSG," or MMO social game) have helped to make social games pervasive. Social games such as FarmVille and CityVille on Facebook and the Apple App Store have over 80 million users [3]. Even though these games still maintain a notion of competition, they emphasize the collaborative nature of gaming. These games mirror the competitive-collaborative nature of software engineering (SE). Even though there is competition for promotions and recognition in the workplace, SE is inherently collaborative and large software development projects require teams of programmers, testers, and designers working towards a common goal [18].

In this paper, we propose an approach to SE, called HALO[1], that builds upon the features of popular MMORPGs, adapting them to a work environment. We use the benefits of gaming to aid the software development process and do not address software processes (or architectures, testing, middleware, etc.) to aid in the development of games.

Jim Whitehead, in his roadmap for collaboration in SE, conjectured: "It might also be possible for projects to graft the narrative and reward structure of MMO roleplaying games onto traditional engineering project work. In a game like WoW, in order to advance, players go on quests, a goal-oriented activity in the game world (go to a dungeon, kill all monsters, retrieve valuable artifact, return for reward). As players perform quests, their abilities increase, which is reflected in their character 'leveling up.' Many players find the game setting (typically from fantasy or science fiction) combined with the quest narrative and leveling reward structure to be very motivational, and for some addictive. It would be intriguing for a project to map development activities onto this style of gameplay. One could imagine engineer experience and capabilities represented in the form of levels, with project subgoals broken down into quest-like units. If this could tap into the motivational aspects of MMO style

[1]There is no connection between our proposal and any games with similar names. The authors have no affiliations with any companies involved in producing such games except as customers.

gameplay, it might increase team productivity by providing a range of incentive structures in addition to the traditional ones of salary, promotion, and satisfaction at completing a project." [22]

As far as we know, this idea was never pursued further, other than as implied by the organization of this workshop, and our goal now is to revisit this idea and elaborate the concepts.

2. BACKGROUND

2.1 Behavioral Psychology

Csikszentmihalyi originally defined flow in 1975 to be a state where an individual is doing an activity for the sake of the activity - not for any benefit outside of the activity [6]. Primarily, Csikszentmihalyi found that to gain this high level of involvement, participants must have a clear set of goals that are challenging and require skill, have immediate feedback, and be utilizing their entire concentration. Finneran and Zhang describe a Person-Artifact-Task (PAT) model for analyzing flow, which is more directly applicable to flow states that involve human-computer interaction [9]. In their work, they describe a component-based analysis model for flow, separating the tools used from the task being performed. They argue that complex computer-based tools can satisfy the requirements for flow differently and that there must be a fit between the participant, artifact (i.e., tool), and task.

Many games focus on the power of operant conditioning - a model that rewards players for good behavior and thus encourages repeat behavior - and social rewards in retaining players [4, 21]. Multi-user games can provide a player with social rewards through recognition from their peers [21].

Douglas and Hargadon provide a definition for immersion, describing that "The pleasures of immersion stem from our being completely absorbed within the ebb and flow of a familiar narrative schema" [8]. Park and Hwang studied the connection between game immersion, flow, and addiction, finding that the immersive environments provided by online games help players to achieve a flow state, which can lead to addiction [17]. While typically viewed as negative behavior, addiction can be beneficial, when the behavior is productive such as being addicted to SE.

Jane McGonigal recently proposed using the power of games to promote *blissful productivity* and *social fabric* in her recent TED video and book [14, 15]. Blissful productivity refers to an ideal state of mind in which gamers are happier working hard (in the game) than relaxing. McGonigal also suggests that gamers will work hard all the time, when given the right kind of work. When people play games together, they build bonds, trust, and cooperation resulting in stronger social relationships, which Mcgonigal refers to as social fabric.

2.2 MMORPG Mechanics

Jim Whitehead's paper described in Section 1 mentions the basic mechanics of MMORPGs: quests and leveling. Quests are often combined into a "quest chain" - a series of tasks with a theme that build upon each other towards a central goal. In these quest chains, some tasks may be mandatory and others may be optional. Further, some quest chains may require the quests to be done in a specific order whereas for others, the order may not matter. Whether they are directly chained together or not, quests are typically grouped within a geographical region. Early on in the game, quests are usually easy to complete, while they become significantly more difficult and complex as the game progresses.

Players typically receive rewards for completing quests that can include experience points and currency. After completing major objectives, players are often rewarded with titles: prefixes or suffixes to display with their name to indicate their accomplishment. There are also rewards for intermediate quests that are parts of larger quest chains. Players may also elect to organize into a formal group of individuals, all virtually playing together, referred to as "parties." These parties facilitate a social community within the game.

The following section describes the game mechanics and how HALO could be incorporated in a typical daily SE Process and Development Environment.

3. HALO GAME MECHANICS

3.1 Software Engineering Process

HALO represents everyday tasks as quests, which can range from something simple such as closing a bug to something as complex as porting the code to a different operating system or platform. For example, if a new intern joins the company, somebody would need to give him/her basic training and introduce him to artifacts such as code repositories or bug report systems. This task could be represented in HALO by a quest.

In some cases, a quest may be too difficult for a single player to undertake on their own. In this case, they will be forced to create a group of other players - a party. This would highlight the collaborative nature of these quests, and SE in general. For example, a player could create a party to internationalize the code for a new region.

Small quests could also be chained together into series of quests - a simple analog for representing use-cases, complicated bug fixes, or daily SE processes. Quests can be grouped in different ways within HALO, improving productivity - similar to geographical groupings in MMORPGs. When a programmer is working on a certain module, the brain needs to context-switch to work on a different, unrelated module. These context switches can be time consuming. HALO optimizes its players' time by grouping all quests related to the same code module together. This strategy causes the developer to deal with everything related to the same module in its entirety, avoiding costly context switches. These groupings need not be truly serial - some parts could be completed in any order; others might have to be ordered. Similar groupings could also be used by management to give priorities to certain tasks such as critical bug fixes (even though these may be across different modules and necessitate context switching).

Quests are predefined in MMORPGs and similarly, they could be created in HALO during the initial planning stages for SE projects. Additionally, new quests and quest series can be constructed on the fly, as warranted. As the design of HALO is flexible, it doesn't have to follow MMORPG concepts strictly; concepts from other types of games or concepts familiar to conventional SE such as priorities and deadlines could be added.

3.2 Blissful Productivity in HALO

3.2.1 Operant Conditioning

HALO utilizes operant conditioning to hook players by providing simple rewards for completing tasks. These rewards come in the form of experience, currency, and achievement titles. Players receive larger rewards for completing more difficult tasks and collaborating with others. HALO's difficulty is scaled throughout the course of a player's game time, i.e., the time of employment at a company. At first, rewards are easily earned, providing a sort of "instant gratification." As an SE project progresses using HALO, rewards become less common but more valuable - a technique commonly used in game environments [21].

As with MMORPGs, HALO focuses on providing social in-game rewards over material out-of-game rewards. These social rewards provide immediate recognition from a player's peers. Developers who complete large quest chains are rewarded with titles - prefixes or suffixes for players' character names. For example, a player who successfully closes 500 bugs may get the title "The Bugslayer." Players who contribute a lot of code to a particular module could get the title of "The Mayor" of that module. These titles provide for quick recognition of successful players.

Currency points are awarded from completing quests and could be used to purchase small real-world rewards: better parking spaces, a free lunch in the company cafeteria, or a gift card to a nice restaurant. These rewards are tiered such that higher valued rewards require more participation and success within the game. Experience points provide a record of a player's past progress through the different quests.

3.2.2 Flow Theory

HALO is designed to directly address many of Csikszentmihalyi's conditions for flow [6]: (1) Clear goals - By dividing tasks into individual quests, HALO always allows developers to see the next task on the horizon. (2) Concentration - HALO provides an immersive environment, fully capturing developer's attention. (3) Loss of self-consciousness - The immersive environment in HALO helps remove developers from the everyday world, an effect commonly found in games [5]. (4) Direct feedback - HALO provides real time feedback to developer actions: code compiles, quests are completed, bugs are marked as closed, etc and developers receive rewards to reinforce their progress. (5) A balance of ability and challenge - HALO's ability to scale difficulty as players progress through the game allows players to reach a point where they consistently feel challenged, but within their ability level.

3.3 Software Development Environment

We envision that HALO will be a simple plugin to IDEs such as Eclipse or Microsoft Visual Studio. This would be similar to commonly used plugins such as Bug Tracking and Version Control. Our HALO plugin would interact with the user and the other plugins to keep track of achievements and progress and communicate these to a central server periodically. This implementation will be extremely light-weight and provide the operant conditioning and flow state benefits of HALO. In this form, HALO's UI would be similar to TextSL, a command-based virtual world interface for games [10]. HALO would need an automated way for figuring out when quests have been completed. This might be easy for some quests - if the quest was to fix a bug, running unit or regression tests related to the bug will tell HALO when the quest is complete. But for complex quest chains, this might be harder and HALO might need to look at artifacts such as code check-ins, statically analyze the source code, and perhaps also require explicit feedback from the players. Such systems might require some novel SE research. We highlight the research challenges that we foresee in Section 5.

4. RELATED WORK

In the late 1990s, our lab developed the CHIME system, an immersive virtual reality for collaboration and software development [7]. CHIME was modeled on popular games of the time, such as "Quake," allowing users represented as avatars to walk around in a 3D world, interacting with project artifacts such as code, bug reports, and email archives. CHIME focused on providing the framework for a collaborative game-like environment, but did not encompass any other game-like features. Systems like CHIME focused on the artifacts of SE, rather than the tasks; HALO focuses on the latter.

There has been previous work in the field of games for SE education. In 1977, Software Hut [11] was one of the earliest works and was intended as a course project to teach Computer Science to graduate students. More recently, other games such as SimSE [16] and Card Game [1] have been proposed to teach SE to students. These games, however, have some limitations in the context of professional SE. SimSE was designed as a single-player game and doesn't have any multi-player features. Software Hut and the Card Game are primarily competitive games and have limited or no collaborative aspects. Further, all of these games are focused towards teaching SE in a classroom setting; they are not meant for "real-world" professional SE. While these games may be beneficial for teaching SE in a classroom, professional SE is usually done collaboratively in teams and hence, these games are not suitable. While HALO could be used to teach SE in a classroom, its main purpose is for use in a real world SE environment. There have been other collaborative games for science education such as DinoQuest [19], but these games do not focus on SE. One of the major research challenges will be moving from gaming for SE education to gaming for professional software development efforts; other research challenges are highlighted in the next section.

5. RESEARCH AGENDA

As there hasn't been a lot of research in this area, we foresee many different SE challenges and potential for novel research. An interesting area of research would be to look at different SE processes and methodologies and explore how they could be mapped to games. For example, if Company A follows the Agile methodology while Company B follows a more traditional Waterfall model, should the same game mechanics be used for both or should there be customizations? Similarly, if there is Global Software Development, do certain game mechanics translate better across countries and cultures?

Another potential research area is exploring the SE aspects of developing such games. It's very common in the normal gaming community to have "game engines". Can we perhaps build game engines for Software Development? Would

such game engines be similar to the more traditional game engines or is there something inherently different about SE processes? A related research aspect would be to explore software architectures and design patterns for these games.

While HALO provides an approach for a game-based SE environment, the actual gameplay is largely dependent upon the project. Quests must be created so that they are satisfying, engaging, and encourage blissful productivity and social fabric. What kind of quest templates would be most appropriate for SE, and how would these be used to design new quests? Could existing bug tracking systems automatically be integrated with a game like HALO?

An interesting research area would be the intersection of SE and Human Computer Interaction (HCI). It would be important to evaluate such games and see how they benefit the users. Traditional HCI evaluation techniques such as user studies may not be sufficient. Would SE games warrant a new methodology for evaluation, and if so, what would be involved?

6. CONCLUSION

We have outlined a new approach to software engineering called HALO that builds upon the properties of popular online collaborative games. We have described HALO's mechanics and how it could fit into typical Software Engineering processes. Having elaborated on Jim Whitehead's high-level concept, we have highlighted some of the future research challenges that could lead to fruitful avenues of research in games and software engineering and our goal is that this will help foster discussion at the workshop.

7. ACKNOWLEDGEMENTS

The authors are members of the the Programming Systems Laboratory, funded in part by NSF CNS-0717544, CNS-0627473 and CNS-0426623, and NIH 2 U54 CA121852-06.

8. REFERENCES

[1] A. Baker, E. O. Navarro, and A. van der Hoek. An experimental card game for teaching software engineering processes. *Journal of Systems and Software*, 75(1-2):3 – 16, 2005. Software Engineering Education and Training.

[2] Blizzard Entertainment. World of Warcraft. `http://us.battle.net/wow/en`.

[3] E. Caoili. CityVille Has Largest Facebook Audience Ever. `http://www.gamasutra.com/view/news/32231/CityVille_Has_Largest_Facebook_Audience_Ever.php`, January 2011.

[4] J. P. Charlton and I. D. Danforth. Distinguishing addiction and high engagement in the context of online game playing. *Computers in Human Behavior*, 23(3):1531 – 1548, 2007.

[5] B. Cowley, D. Charles, M. Black, and R. Hickey. Toward an understanding of flow in video games. *Comput. Entertain.*, 6:20:1–20:27, July 2008.

[6] M. Csikszentmihalyi. *Beyond boredom and anxiety*. Jossey-Bass Publishers, San Francisco, 1st edition, 1975.

[7] S. E. Dossick and G. E. Kaiser. CHIME: a metadata-based distributed software development environment. In *Proceedings of the 7th European software engineering conference held jointly with the 7th ACM SIGSOFT international symposium on Foundations of software engineering*, ESEC/FSE-7, pages 464–475, London, UK, 1999. Springer-Verlag.

[8] Y. Douglas and A. Hargadon. The pleasure principle: immersion, engagement, flow. In *Proceedings of the eleventh ACM on Hypertext and hypermedia*, HYPERTEXT '00, pages 153–160, New York, NY, USA, 2000. ACM.

[9] C. M. Finneran and P. Zhang. A person-artefact-task (PAT) model of flow antecedents in computer-mediated environments. *Int. J. Hum.-Comput. Stud.*, 59:475–496, October 2003.

[10] E. Folmer, B. Yuan, D. Carr, and M. Sapre. TextSL: a command-based virtual world interface for the visually impaired. In *Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility*, Assets '09, pages 59–66, New York, NY, USA, 2009. ACM.

[11] J. Horning and D. Wortman. Software Hut: A Computer Program Engineering Project in the Form of a Game. *Software Engineering, IEEE Transactions on*, SE-3(4):325 – 330, July 1977.

[12] M. Keegan. A Classification of MUDs. `http://mk.ucant.org/info/classification_muds.html#13`.

[13] Linden Lab. Second Life. `http://secondlife.com`.

[14] J. McGonigal. Gaming can make a better world. `http://www.ted.com/talks/lang/eng/jane_mcgonigal_gaming_can_make_a_better_world.html`.

[15] J. McGonigal. *Reality Is Broken: Why Games Make Us Better and How They Can Change the World*. The Penguin Press HC, 2011.

[16] E. O. Navarro and A. van der Hoek. SimSE: an educational simulation game for teaching the software engineering process. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, ITiCSE '04, pages 233–233, New York, NY, USA, 2004. ACM.

[17] S. Park and H. Hwang. Understanding online game addiction: Connection between presence and flow. In *Human-Computer Interaction. Interacting in Various Application Domains*, volume 5613 of *Lecture Notes in Computer Science*, pages 378–386. Springer Berlin / Heidelberg, 2009.

[18] S. Sawyer. Software development teams. *Commun. ACM*, 47:95–99, December 2004.

[19] W. Scacchi, R. Nideffer, and J. Adams. Collaborative game environments for informal science education: DinoQuest and DinoQuest Online. In *International Symposium on Collaborative Technologies and Systems*, pages 229 –236, May 2008.

[20] R. Trubshaw. Welcome to the Home of MUD1. `http://www.british-legends.com`.

[21] P. Wallace. *The Psychology of the Internet*. Cambridge University Press, March 2001.

[22] J. Whitehead. Collaboration in Software Engineering: A Roadmap. In *2007 Future of Software Engineering*, FOSE '07, pages 214–225, Washington, DC, USA, 2007. IEEE Computer Society.