Towards Democratizing Computer Science Education through Social Game Design

Navid Ahmadi
Faculty of Informatics
University of Lugano
Lugano, Switzerland
navid.ahmadi@usi.ch

Mehdi Jazayeri
Faculty of Informatics
University of Lugano
Lugano, Switzerland
mehdi.jazayeri@usi.ch

Alexander Repenning
Computer Science
Department
University of Colorado
ralex@cs.colorado.edu

ABSTRACT

Computer science and software engineering education are limited to formal courses that are being taught in the school. Those who do not have access to the educational courses miss the learning context, even if educational tools are accessible for free. Computer game design has been employed as an engaging medium for practicing software engineering and computer programming skills. However, collaborative work is not supported by educational game design environments and peer learning is limited to face-to-face communication in the classroom. In this paper, we suggest democratizing computer science education by incorporating social learning into the educational game design using existing Web 2.0 mechanisms. Consequently, online users will benefit from situated learning in the game design activities that take place in their social networking space. We present AgentWeb, a Web-based game design environment as the steppingstone to enable social game design activities, and explore the challenges in fostering social learning in online game design practices.

Categories and Subject Descriptors

K.3.1 [**Computers and Education**]: Computer Uses in Education—*collaborative learning, distance learning*; K.3.2 [**Computers and Education**]: Computer and Information Science Education—*computer science education*

General Terms

Design, Human Factors, Theory

Keywords

Educational game design, peer learning, communities of practice, social learning, social interfaces, Web 2.0

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GAS '11, May 22, 2011, Waikiki, Honolulu, HI, USA
Copyright 2011 ACM 978-1-4503-0578-5/11/05 ...$10.00.

1. INTRODUCTION

Educational games and simulations are becoming increasingly appealing in public schools [8]. Particularly, game design is employed as an engaging medium for teaching computer programming [9] and software engineering [20][4]. Students explore computer science concepts through creating game components, programming the components, and observing their behavior in the game scene. AgentSheets [17], Alice [5], Game Maker [16], and Scratch [19] are instances of such educational game design environments.

Existing game design environments enable individuals to create their own games and simulations during classroom sessions. However, collaborative game design is not supported by these environments.

- The collaborative tools are not situated in the learning context [11]. As a result, peer learning [24] is limited to the classroom session through face-to-face communication or explicit communication tools such as email.
- Those with no access to formal education in a classroom setting miss the educational context, even if they have free access to a game design environment.

Game sharing Web sites have facilitated collecting and sharing the games developed by end users regardless of their learning context. Early on, *Behavior Exchange* [18] let users share AgentSheets games through the Web. Game Maker community share their games through a Web site, although online playing or exploring of the games is not possible. Recently, the Scratch Web site has collected millions of user-created games. Online users can play the games and write comments on them from inside the Web browser. However, the game design process is not supported by collaborative and social modalities. In other words, these Web sites let users share the result of their game design, i.e., "what" they have built, not the design process, i.e., "how" it was built. In order to explore inside the game to learn how a game was created, users have to download the game and open it in the game design environment. Such interface draws a line between playing the game as the entertaining artifact and exploring the game components as the educational artifact. As the game design process takes place on the desktop of individuals, rather than in the game sharing environment, the Web site fails to incorporate the game design knowledge transfer into the social environment.

In this paper, we suggest the integration of the game design process into the social context of users on the Web to address the described problem. Accordingly, we employ the online social networking space of the users as the platform through which computer science knowledge is exchanged. This approach includes several benefits over existing game sharing Web sites:

- Benefiting from being a cloud-based environment, users will have pervasive access to their games and sharing with others.
- The game design process turns into a white box that is explored and modified by multiple users. Consequently, the learning is situated in design practices.
- The access to the game design and its educational context is scaled up to Web-scale. Online users, regardless of their geographical location, will be able to join or form a community of practice according to their background knowledge, to practice software engineering and programming skills.

In order to enable users to design their games inside their online social environment, we have developed AgentWeb[1], a fully Web-based game design and programming environment. Users design and program their games right inside the Web browser. Built upon only open Web technologies, AgentWeb can be easily integrated into existing social networking environments, through which users socialize around the game design process rather than the game itself. This paper briefly describes the design rationale of AgentWeb. Furthermore, we discuss the challenges and underlying design of integrating AgentWeb with the social environment of online users in order to foster social learning through game design.

2. RELATED WORK

Vygotsky's notion of the *zone of proximal development* [21] suggests that people, when engaging in problem solving, benefit substantially from collaborating with more capable peers. It has been observed that software engineering is a social activity in different settings. The importance of collaborative and social work in professional software engineering settings have been identified and supported by collaborative software. Booch et al. [3] recognize the emergence of *collaborative development environments* and categorize them according to their application domain. Whitehead [22] distinguishes software engineering collaboration as a model-based collaboration, focused on development of shared meaning around the model and elimination of ambiguities. Ahmadi et al. [1] survey the collaborative software engineering practices from a social perspective.

Nardi describes the collaboration patterns among end users [14]. Collaborative work, such as pair programming, has improved learning in educational contexts [23]. However, collaborative work is seldom supported by educational computer science tools. Kelleher et al. [9] argue the lack of social support for novice programmers. Web 2.0 [15] has been recognized as a medium that supports social learning [12]. Web 2.0 based tools have been employed in professional software engineering settings [1]. As described in Section 1, several game design communities, commercial and educational, have employed Web 2.0 to form a community around the game design environment. Yet, game design has not made it to the Web as a social activity. Therefore, the social learning is limited to discussion forums and comments on the game, rather than collaborative exploration of the game design process.

[1]AgentWeb is available online at http://agentweb.inf.usi.ch

3. WANTED: WEB-BASED GAME DESIGN

The preliminary step towards enabling social game design among online users is to create a Web-based game design environment. A Web-based environment turns game design to a white box experience which lets users explore the game components and programs right from inside their Web browser. This is in contrast to existing desktop-based game design tools that deliver the games to the Web as a black box that can only be played. The white box approach enables seamless integration of existing Web-based social interfaces such as comments, tags, and recommendations into the game design and exploration process which in turn fosters the social learning in educational game design contexts.

A game design environment provides functionalities to create the game objects, choose or draw their depiction, program them, and put them together in the game scene. When targeted for novices, the core components of such an environment include well-designed graphical user interfaces, support for end-user programming and underlying language execution system, the game engine and corresponding runtime system, and graphics rendering of the games. None of these are typical or even existing components of a Web-based application. Combining all the mentioned components into a Web-based integrated development environment (IDE) has been impossible until recently due to the lack of, or immaturity of, relevant Web technologies:

- Graphics drawing libraries were missing or they had poor performance. Only recently, HTML5 canvas has been able to render graphics natively inside the browser at a speed reasonable to create open Web interactive games.
- Building complex user interfaces using Document Object Model is frustrating. Even basic user interface features such as drag and drop is not natively supported until HTML5. Only recently, existing open Web user interface development libraries such as jQuery and Dojo are becoming mature enough to build sophisticated user interfaces such as a game design IDE.
- Browser-based execution using JavaScript interpreters have been discounted due to insufficient execution speed. Only recently, emerging just-in-time compilers for JavaScript have made the execution of interactive games possible.

Due to these obstacles, to date, most of the interactive content on the Web including the games are developed by professional programmers using Rich Internet Application (RIA) frameworks such as Flash or Java, and embedded into the Web pages to make them accessible to end users. Unlike other artifacts that are being created and shared by end users in Web 2.0, end users have no part in creating and sharing the games on the Web.

With the recent enhancements in Web technologies, primarily HTML5, as described above, creating a Web-based game design environment for novices has become possible. Accordingly, we have designed and implemented a fully Web-based game design environment called AgentWeb. While creating games using agentWeb, online users are involved in

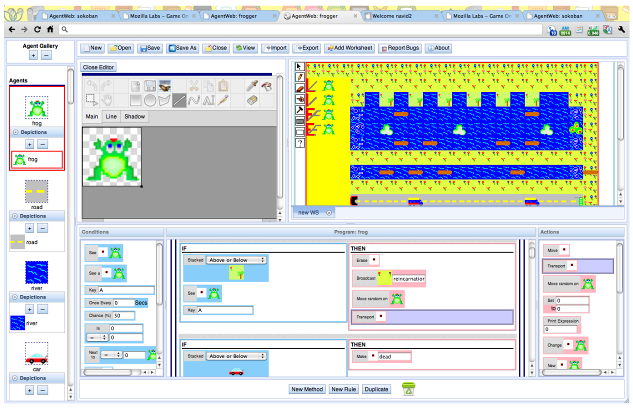

Figure 1: AgentWeb, a fully Web-based game design environment, enables the integration of educational game design practices into the social environment of online users.

educational software engineering activities including identifying the game components, creating the components as *agents* in the environment, programming the agents and the interaction among the agents using provided message passing mechanisms, instantiating the agents in the game scene (as the runtime system) and executing the game, receiving feedback from observing agents' behavior, and modifying the program of agents until they achieve the desired behavior. AgentWeb resembles AgentSheets [17], a desktop-based educational game authoring tool. Figure 1, shows the AgentWeb game design environment used for developing a Frogger game. AgentWeb supports the user in exploring and modifying the game components, without the barrier of downloading and uploading.

AgentWeb provides novices with a visual rule-based programming language. The visual language is dynamically compiled into JavaScript. Benefiting from dynamic compilation, AgentWeb supports *interactive programming* [2] using which users are capable of modifying the agent program while the game is running and the changes are immediately applied into the execution. Interactive programming lets users explore the behavior of agents and receive immediate feedback to their programming activities. For instance, we have observed that middle school students use interactive programming to tune the movement speed of game objects while they play the game.

Our usability evaluations verify that the Web-based game design environment enables novice programmers to develop educational games and simulations in the same amount of time as they spend in desktop-based AgentSheets creating the same game. The performance evaluation of execution and rendering the end-user developed games inside the Web browser reveals a performance comparable to desktop-based equivalent implementations, and in some cases even outperforming them.

4. SOCIAL GAME DESIGN CHALLENGES

Having a Web-based game design environment ready for the masses, we aim at integrating it into the social networking space of online users through which users socialize with their friends and acquaintances. The integration with social networking environments delivers the game design tool to online users on the Web-scale. Although, the social integration goes beyond democratizing and providing instant access to the game design tools on the Web. The goal of such integration is to foster social learning by motivating user involvement in game design scenarios. Integration with the social networking space of online users intrinsically encourages user involvement up to a generic level [7]. However, designing the underlying social interface to foster educational game design activities remains challenging. Some of the challenges and their design implications are discussed below.

Support the communities of practice The social interface has to support contextualizing user participation. In order to situate their learning into the social context, users with different knowledge and background should be able to form their own communities around the game or simulation of interest or join existing communities according to their interests and skills. Such communities are considered as long-lived evolving ecosystems [6]. Respectively, the social interface requires an underlying model that takes the diversity and evolution of the communities into account. Legitimate peripheral participation [11], for instance, can be adapted to the game design communities to define several stages of skills and involvement. Skill levels have been explored in the end-user development field [13].

Cover the collaboration time/space continuum On the space continuum, geographically distributed users can form communities and collaborate in game design activities, inherently benefiting from the Web platform. However, distance collaboration has to be facilitated through communication channels. Not only the social interface supports community members to collaborate formally on a shared project, but also it has to provide informal communication channels such as integrated instant messaging through which the distance collaboration is mediated. On the time continuum, not only the social interface supports asynchronous collaboration by letting users modify the game at different times, but also it has to support synchronous game design which lets multiple users work on the same game at the same time. A real-time awareness interface informs users of the activities of other users.

Support peer learning scenarios Supported by full coverage of collaboration continuum, peer learning scenarios can be supported to increase the learning opportunities in educational contexts. In particular, in communities of practice, peripheral members acquire knowledge from the core members. Peer learning scenarios can be identified in the game design context and be explicitly supported by the social interface. A possible scenario is that the social interface lets peripheral members follow precisely the core member by not only informing peripheral members of the changes made by the core member and updating the game objects, but also updating peripheral members' game design environment (such as opening the menus, showing the dialogs, switching the views) corresponding to the core member's interface. In other words, the core member plays the role of trainer and the peripheral member plays the role of trainee.

Support for measuring the learning outcomes By monitoring user involvement in the community, we can define metrics such as number of comments, time spent on designing a game, and amount/quality of program written by a user, to assess their contribution automatically. To assess the quality of programs written by novice users, approaches based on latent semantic analysis have been employed [10]. Automatic assessment of user contribution can further be used to create social engagement scenarios which in turn lead to user involvement in the community. Such social engagement scenarios include rewarding the user, e.g., promotion in the community and increase of authority, and creating competition scenarios inside the community. In the latter case, competitions should be designed for learning purposes.

5. CONCLUSIONS

Computer science and software engineering education are largely limited to formal courses that are being taught in the school. Those who do not have access to the educational courses miss the learning context, even if educational tools such as game design environments are accessible for free on the Web. In this paper we introduced our approach to leverage Web 2.0 as the social learning medium to compensate for missing educational context through communities of practice. Accordingly, we suggest integrating game design as a Web-based activity with the social networking space of online users. We presented AgentWeb, a fully Web-based game design environment, as the first step towards enabling social game design.

6. ACKNOWLEDGMENTS

This research has been funded by the Swiss National Science Foundation and in part supported by the National Science Foundation (grant No. DLR-0833612). Opinions expressed are those of the authors and not necessarily those of the mentioned organizations.

7. REFERENCES

[1] N. Ahmadi, M. Jazayeri, F. Lelli, and S. Nesic. A survey of social software engineering. *1st International Workshop on Social Software Engineering and Applications*, page 12, 2008.

[2] D. R. Barstow, H. E. Shrobe, and E. Sandewall. *Interactive Programming Environments. McGraw-Hill*, page 609, 1984.

[3] G. Booch and A. Brown. Collaborative development environments. *Advances in Computers*, 59:1–27, 2003.

[4] K. Claypool and M. Claypool. Teaching software engineering through game design. *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, pages 123–127, 2005.

[5] S. Cooper, W. Dann, and R. Pausch. Alice: a 3-d tool for introductory programming concepts. *Journal of Computing Sciences in Colleges*, 15(5):107–116, 2000.

[6] G. Fischer, E. Giaccardi, H. Eden, M. Sugimoto, and Y. Ye. Beyond binary choices: Integrating individual and social creativity. *International Journal of Human-Computer Studies*, 63(4-5):482–512, 2005.

[7] G. Fischer, K. Nakakoji, and Y. Ye. Metadesign: Guidelines for supporting domain experts in software development. *Software, IEEE*, 26(5):37 – 44, 2009.

[8] M. A. Honey and M. Hilton. *Learning Science Through Computer Games and Simulations. The National Academies Press*, page 180, 2011.

[9] C. Kelleher and R. Pausch. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys*, 37(2):83–137, 2005.

[10] K. H. Koh, A. Basawapatna, V. Bennett, and A. Repenning. Towards the automatic recognition of computational thinking for adaptive visual language learning. *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 59 – 66, 2010.

[11] J. Lave and E. Wenger. *Situated Learning: Legitimate Peripheral Participation. Cambridge University Press*, 1991.

[12] C. McLoughlin and M. Lee. Social software and participatory learning: Pedagogical choices with technology affordances in the web 2.0 era. *ICT: Providing choices for learners and learning. Proceedings ascilite Singapore*, 2007.

[13] A. Mørch. Three levels of end-user tailoring: customization, integration, and extension. *Computers and design in context, MIT Press*, 1997.

[14] B. Nardi. *A Small Matter of Programming: Perspectives on End User Computing. MIT Press*, 1993.

[15] T. O'Reilly. *What is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software.* 2005.

[16] M. Overmars. Teaching computer science through game design. *Computer*, 37(4):81–83, 2004.

[17] A. Repenning and J. Ambach. Tactile programming: a unified manipulation paradigm supporting program comprehension, composition and sharing. *Proceedings of the 1996 IEEE Symposium on Visual Languages*, pages 102 – 109, 1996.

[18] A. Repenning, A. Ioannidou, M. Rausch, and J. Phillips. Using agents as a currency of exchange between end-users. *Proceedings of the WebNET 98 World Conference of the WWW, Internet, and Intranet, Orlando, FL*, pages 762–767, 1998.

[19] M. Resnick, B. Silverman, Y. Kafai, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, and J. Silver. Scratch: Programming for all. *Commun. ACM*, 52(11):60–67, 2009.

[20] M. Shaw. Software engineering education: a roadmap. *Proceedings of the conference on The future of Software Engineering (ICSE '00)*, pages 371–380, 2000.

[21] L. S. Vygotsky. *Mind in Society: The Development of Higher Psychological Processes. Harvard University Press*, 1978.

[22] J. Whitehead. Collaboration in software engineering: A roadmap. *Future of Software Engineering (FOSE '07)*, 2007.

[23] L. Williams, E. Wiebe, K. Yang, M. Ferzli, and C. Miller. In support of pair programming in the introductory computer science course. *Computer Science Education*, 12(3):197–212, 2002.

[24] C. Wills, D. Finkel, M. Gennert, and M. Ward. Peer learning in an introductory computer science course. *ACM SIGCSE Bulletin*, 26(1):309–313, 1994.