# Supporting Social Interactions and Awareness in Educational Programming Environments

Christopher D. Hundhausen
Human-centered Environments for Learning and Programming (HELP) Lab
School of Elec. Eng. and Computer Science
Washington State University
hundhaus@wsu.edu

Adam S. Carter
Human-centered Environments for Learning and Programming (HELP) Lab
School of Elec. Eng. and Computer Science
Washington State University
cartera@wsu.edu

## Abstract

Empirical evaluations of programming environments have traditionally focused on human performance measures such as task efficiency, error rates, and learnability. In addition to these effectiveness measures, we believe there is good reason to consider the ability of programming environments to promote *social interactions and awareness* during programming tasks. Indeed, especially in educational contexts, programming success and persistence in the computing discipline have been positively correlated with programmers' sense of community and ability to communicate with others. We introduce *social programming environments* as a new breed of educational programming environment designed to promote social interaction and awareness, and we propose a way to evaluate such environments relative to social learning theory.

***Categories and Subject Descriptors*** D.2.6 [**Programming Environments]** Interactive environments; K.3.1 [**Computers and Education**] Computer Uses in Education - *collaborative learning*; K.3.2

***General Terms*** Design, Experimentation, Human Factors, Languages.

***Keywords*** Social programming environments, learning analytics, social learning theory.

## 1. Introduction

In the software engineering profession, the ability to monitor and communicate with other team members in software development projects is assumed to be essential to promoting programming efficiency and success. To that end, software engineering researchers have long been interested in developing tools to support awareness and communication in software development activities. For example, Microsoft Research has developed tools that enable programming team members to explore where other team members are spending their development time in a code base [3], and to peripherally monitor what other team members are up to [2]. In a similar vein, tools have been developed to support collaborative programming at a distance (e.g., [4]), and off-the-shelf communication tools are commonly used to support asynchronous discussions about shared code bases.

In contrast, promoting collaboration and awareness of others' activities has been of little interest in computing education, where it is often assumed that such collaboration and awareness would constitute "cheating." Especially in early programming courses, learners are often required to complete programming assignments individually, and are often discouraged from talking to anyone except teaching personnel about programming issues they encounter.

In this position paper, we use social learning theory to argue that social interaction and awareness are essential to promoting effective learning experiences in computing education. We then introduce a *social* plug-in to an integrated programming environment (IDE) that leverages trends in social networking to support social awareness and interaction. We conclude by outlining a strategy for evaluating social participation within the environment, and for exploring relationships between such participation and key performance, affective, and demographic variables.

## 2. Social Learning Theory

Bandura's *self-efficacy theory* [1] posits that students develop a positive sense of their own programming abilities (so-called *self-efficacy*) by being able to observe the activities of


*PLATEAU'14,* October 21, 2014, Portland, Oregon, USA.

http://dx.doi.org/10.1145/2688204.2688215

their peers, and by being able to evaluate themselves relative to their peers. Not surprisingly, positive self-efficacy has been strongly correlated with persistence in the computing discipline [6]. Likewise, *situated learning theory* [5] posits that participation in a community of practice, which involves both observation of others and actual participation in community activities, is essential to learning. Both of these social theories of learning suggest that learners may have difficulties making progress if they are forced to program in isolation from a broader community. Thus, while it is, of course, important to be able to evaluate individual student work, computing educators would do well also to provide students with opportunities to observe and interact with a broader learning community.

## 3. Social Programming Environment

How might we rethink the design of IDEs so that they provide greater opportunities for learners to observe each other, ask and answer questions, and build a learning community? Inspired by trends in social networking, we have been developing OSBIDE (Online Studio-Based IDE), a social plug-in to an IDE that supports an *activity stream* (see Figure 1). As a learning community's activities—including compilations, run-time exceptions, and debugging events—unfold, they are injected into the activity stream; a back-end social recommender relates them to other learners' activities (e.g, "You and 3 others have gotten this error"). In addition, using the social plug-in, learners can

- search and filter the activity stream for events that might help them with their issues.
- select programming artifacts—e.g., sections of code, pieces of the call stack—to ask questions about; these artifacts are then injected into the activity stream, making it easy for learners to ask a question about them.
- pose and respond to questions by commenting on any item in the activity stream, just as they can comment on posts on social networking sites like Facebook.

## 4. Evaluation Approach

We have developed a back-end system that logs all programming events that occur in the IDE: edits, build events, execution events, passive social events (e.g., clicks within the activity stream), and active social events (posts and replies within the activity stream). In addition, in the computing courses in which we are deploying OSBIDE, we are collecting demographic data, course grade data, and pre- and post-survey data on student attitudes (e.g., self-efficacy, sense of community). These data provide a foundation for performing correlational analyses involving programming behavior, social behavior, course performance, demographics, and affective measures. They also provide a foundation for building exploratory timeline visualizations of learners' programming and social activities.

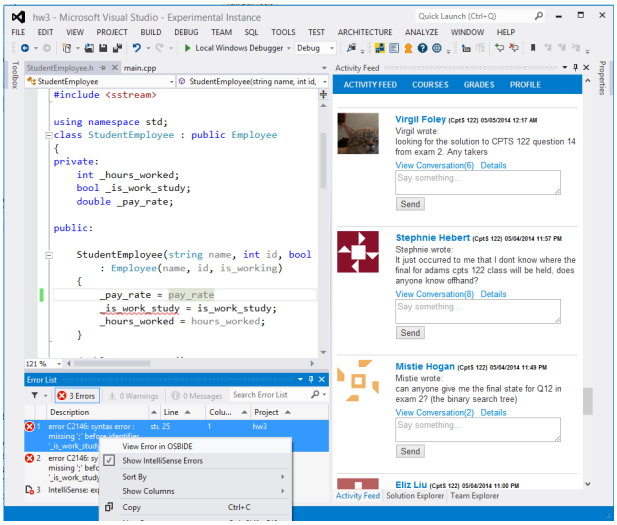


**Figure 1.** Screenshot of OSBIDE Social Plug-In

In order to analyze patterns of student participation relative to social learning theory, we partition students into participation levels based on the amount of active and passive social events they log as they work on a given programming assignment. In later programming assignments, social learning theory posits that students' level of participation should increase. In ongoing analyses, we are refining operational definitions of differing levels of participation, and building probabilistic models of how students transition from level to level over time. The ultimate goal is not only to better understand the ways learners appropriate an activity stream in order to participate more fully in computing courses, but also to gain insight into how best to design social programming environments to promote learning.

## Acknowledgments

This research is supported by the National Science Foundation under grant no. IIS-1321045.

## References

[1] Bandura, A. 1997. *Self-efficacy: the exercise of control*. Worth Publishers.

[2] Biehl, J.T. et al. 2007. FASTDash: a visual dashboard for fostering awareness in software teams. *Proc. SIGCHI Conference.* ACM. 1313–1322.

[3] DeLine, R. et al. 2005. Easing Program Comprehension by Sharing Navigation Data. *Proc.VL/HCC.* IEEE. 241–248.

[4] Goldman, M. and Miller, R.C. 2011. Real-time collaborative coding in a web IDE. *Proc. UIST.* ACM. 155–164.

[5] Lave, J. and Wenger, E. 1991. *Situated learning: Legitimate peripheral participation*. Cambridge Univ. Press.

[6] Rosson, M.B. et al. 2011. Orientation of Undergraduates Toward Careers in the Computer and Information Sciences: Gender, Self-Efficacy and Social Support. *ACM Trans. Comput. Educ.* 11, 3 (2011), 1–23.