An Interactive Web-based IDE Towards Teaching and Learning in Programming Courses

Hai T. Tran, Hai H. Dang, Kha N. Do, Thu D. Tran, Vu Nguyen
University of Science, VNU-HCM
Ho Chi Minh City, Vietnam
{tthai, dhhai, dnkha, tdthu, nvu}@fit.hcmus.edu.vn

***Abstract*—In programming courses, students are often asked to work in groups to write multiple programs. The existing Integrated Development Environments (IDE), however, do not encourage concurrent interactions among student programmers. Students often encounter difficulties in collaborative activities, sharing resources, reviewing code and discussing ideas, especially when the group cannot reach a common arrangement for face-to-face meetings at a convenient time and place. We propose to resolve this problem by transforming the traditional IDE into Software as a Service on the Web, and integrating collaborative features in order to create an interactive and responsive environment, where real-time on-the-job guidance, communication and collaboration can be delivered. In this paper, we introduce a web-based IDE designed for this purpose, and show how it can be applied in programming courses.**

***Keywords*— web-based IDE; collaborative learning.**

I. Introduction

Collaborative learning consists of a variety of educational practices in which interactions among peers is an important characteristic [1]. In many programming courses, besides small individual exercises, students are usually asked to work in groups to complete a larger and more comprehensive software program. By working in group, students practice their teamwork skills, communication skills, and gain experience in collaborating with different people on their projects. Such skills and experience are important for students to become successful software engineers in the future. During these projects, students take part in many group activities, including sharing materials, reviewing code, and discussing solutions and issues. Unfortunately, existing IDEs lack relevant capabilities to help students perform such activities while encouraging them to engage in the project [2]. Students have to use different tools such as desktop-based Eclipse for programming and Skype for chatting and screen sharing. This approach does not give students a whole picture of what is going on during their projects. It also makes students difficult to collaborate on their programming and source-control activities, thus discouraging their participation on the project.

With the power of the Internet, it is of natural incentives for education and software professionals to study, evaluate and introduce new web-based software services for more effective learning. These tools are simply not some Web pages providing course contents, but they should be fully interactive environments that simulate actual software development activities. Students thus can learn technologies together with communication and management skills, and can earn valuable practical experiences for career preparation. Teachers can easily keep tracks of students' activities, provide timely responses, adjustments or suggestions, and evaluate students' performance more completely.

A web-based IDE can offer users the following several benefits:

• It is "zero-install", meaning that any user with a Web browser can use this system to do necessary programming tasks without downloading and installing an IDE and related tools.

• It suggests storing data on the Cloud, which enables easy global sharing and collaboration. Shared documents can be synchronized instantly, and discussions can be delivered through various social media channels.

• It encourages the continuity of work because users can do tasks on many devices, wherever and whenever they want.

We develop a web-based IDE, namely IDEOL, that leverages such advantages while integrating collaborative features of social networks. This system allows programmers to perform traditional IDE tasks in a single window of a Web browser. This application also provides many functions for group activities such as editing collaboratively, reviewing code and discussing details of programs or solutions. With these features, we expect that the IDE helps enhance student's learning experience and overall effectiveness of learning through project assignments.

In this paper, we introduce key features of IDEOL that allow students and teachers to perform programming and collaboration activities in programming language courses. To initially evaluate the effectiveness of the system, we performed two experiments of students using the system in their programming language courses. The results from the experiments show that IDEOL has the potential to encourage students to interact with each other and allow students to manage their groups better than the typical environment that includes Eclipse, Visual Studio, and popular communication tools such as Skype and Yahoo Messenger.

II. Background and Related Works

A. Interactive Learning and Supporting Tools

It is important to understand the definition of interaction before discussing the influence of supporting Web tools in collaborative learning. Moore's classification of interactions [3] includes:

- Learner-content: intellectual interaction with content – text, audio, video clip or computer program.
- Learner-instructor: interaction between the learner and the instructor in order that the learner understands the content or information better.
- Learner-learner: interaction "between one learner and other learners, alone or in group settings, with or without the real-time presence of an instructor".

Many tools have been developed to increase the quality of various interaction types:

- **Course management systems** advance the administration, documentation, tracking, and reporting of training programs, classroom, and online events.
- **Shared repositories** provide capabilities to save and share materials among students and lecturers.
- **Instant messaging tools** facilitate collaborative learning and teamwork activities, and promote different types of communication [2][4].

These tools, however, have many limitations. First, they are separate products rather than an integrated complete environment where students can do all their tasks conveniently. Second, the practical skills and experience that students acquire through their class assignments are less useful because the approaches applied in these assignments quite differ from the software development processes and tools actually used in industry. Finally, they do not take advantage of the current social network trend in which students interact heavily through these networks.

B. Integrated Development Environments

IDEs are the irreplaceable tools for software development. Software projects have become more complex [5]: more diverse customer's requirements, shorter production time for more delicate software processes, millions of lines source code in size with hundreds of team members, different platforms supporting software with a combination of more than one programming languages, etc. This raises the need of a unified working environment with integrated tools, where an IDE comes in to play its role.

Existing desktop-based IDEs such as Microsoft Visual Studio, Eclipse, NetBeans or Xcode not only support traditional tasks but also provide many advanced features like graphic user interface (GUI) builder, class explorer, static code analysis, source code revision control, and code completion. However, due to their complexity, these environments require transfer of heavy setup files, laborious installation and configuration processes, and high computing resources consumption. Moreover, these systems with advanced functions are overwhelming to novice programmers [6].

Recently, these systems are used with other tools to support collaborative programming. Programmers can utilize applications such as *Skype, TeamViewer*, *Google Hangouts*, *VNC* and *Microsoft NetMeeting* to synchronize screens and to chat. However, these tools require high data bandwidth and they do not allow the concurrent use of a single computer by multiple users. Plug-ins such as *RIPPLE* [7] or *Saros* [8] are available for Eclipse. With RIPPLE, users can use an instant messaging program and synchronize views with the low bandwidth. Saros supports more features: a real-time collaborative editor, a concurrent drawing tool, screen sharing and other functions. Unfortunately, these plug-ins are still desktop-based.

In the last decade, web-based IDEs have been developing to utilize power of the cloud to solve these problems: heavy IDE cores are moved to the cloud, allowing users with a Web browser to instantly start up the IDE and work. Other benefits include (but not limit to) work mobility, independence of devices and platforms, modest local resource consumption, easier data management and synchronization, as well as real-time sharing and collaborations. There are a few environments available on the market, for example *CodeRun*[1], *Kodingen*[2], *Cloud9*[3], and *eXo Cloud IDE*[4]. All of them focus on Web development. The latter two provide more features for chatting, reviewing code, collaborative editing, and teaching by showing. Although online environments supporting compiled programming languages such as *Compilr*[5], *ECCO*[6] and *Wwworkspace*[7] were introduced, users cannot debug their projects on these systems. The development of ECCO and Wwworkspace has been stopped. Recently, IBM and Eclipse foundation have started a new project named *Orion*[8] in an effort to bring the Eclipse IDE online, although Orion 2.0 is still a platform for developing Web applications. Other notable web-based IDEs arising from related research works are: *Adinda* [9], on mining of interactive activities; *CEclipse* [10], on service composition and program analysis; *Collabode* [11], on supporting close and synchronous collaboration.

We aim to overcome the shortcomings of simple, one-way interactive environments in programming and software development courses with our online IDE, called *IDEOL*.

III. THE IDEOL

Developing IDEOL is a part of our long-term project to develop an interactive learning environment that includes a course management system, a wiki, a web-based IDE, a testing tool, and a web-based design tool. The initial need of a web-based IDE with interaction features came from our programming courses introducing students to basic programming concepts and practices. In these courses, which teach students how to work with fundamental language elements such as structure, data types, and functions, students are only required to use core IDE functions including writing, compiling, and debugging simple programs. We realized that, although modern IDEs such as Eclipse and Microsoft Visual Studio have

[1] *http://coderun.com*
[2] *https://kodingen.com/*
[3] *https://c9.io/*
[4] *https://codenvy.com/*
[5] *http://compilr.com/*
[6] *http://ecco.sourceforge.net/*
[7] *http://www.willryan.co.uk/WWWorkspace/*
[8] *http://www.eclipse.org/orion/*

all of these functions, they lack interaction capabilities suitable in the class setting.

IDEOL is an online integrated development environment that allows programmers to write, build, run, and debug C/C++ programs while collaborating with each other on a certain source file or parts of a source file.

A. IDEOL's Key Features

The IDEOL system has the following key features:

- A real-time and interactive editor. Team members can edit multiple source files and see changes made by others instantly.
- A real-time discussion board with the tagging mechanism. This feature allows users tag lines of code or source files in a discussion. Students can set simple states for an issue, and rate a comment. They also have the ability to attach files, debug results, compilation errors, and execution inputs/outputs in their messages.
- A summarization feature. This feature summarizes changes in a project. Users can view the changes summary when they log in into IDEOL or through emails.
- A debugger. Our IDE utilizes GNU GDB version 7.2 and allow users to debug C/C++ console programs.
- Build automation tools with GNU C/C++ compilers version 4.4.7 and an interactive execution tool for console applications. Users can obtain and examine the output produced by the compiler and application.

These basic components are provided through a simple and comprehensive user interface that suits beginner programmers (see Figure 1). Advanced features are added for more experienced student programmers and are designed to be less visible in the GUI, so as not to distract beginners from the main functions.

IDEOL was tested smoothly in any HTML5-supported browsers, on various platforms like Windows, Linux. It is partially runnable on tablets – for example Blackberry Playbook and iPad. With HTML5, the client side application is real-time interactive, giving the feel no different from familiar desktop-based environments.

The first version of IDEOL was deployed on a server at our school in Spring 2011. Lecturers and students were invited for alpha trial. It was also experimentally used in introduction courses of programming in Fall 2011. In order to evaluate the potential of the IDE, we conducted surveys for the needs of students and for the satisfaction and expectation about our system.

The feedbacks have shown that the first version IDEOL can provide basic and common operations required by a programmer. Novice students think that IDEOL is simpler to start programming with than desktop IDEs, and that it maintains a universal interface, unlike different interfaces they run into when using installations of desktop IDEs on different computers. More senior students, however, expect that more advanced features be added to support their needs. This reflects the diverse opinions about levels of complexity.

Basing on the feedbacks and survey results, we continuously update the system to remove defects or introduce new features, and more importantly, devise ways of using the system in education. The current version of IDEOL can be accessed freely through the address https://demo.ideol.net. An updated HTML5-supported browser is required in order to fully test out the features of the system.

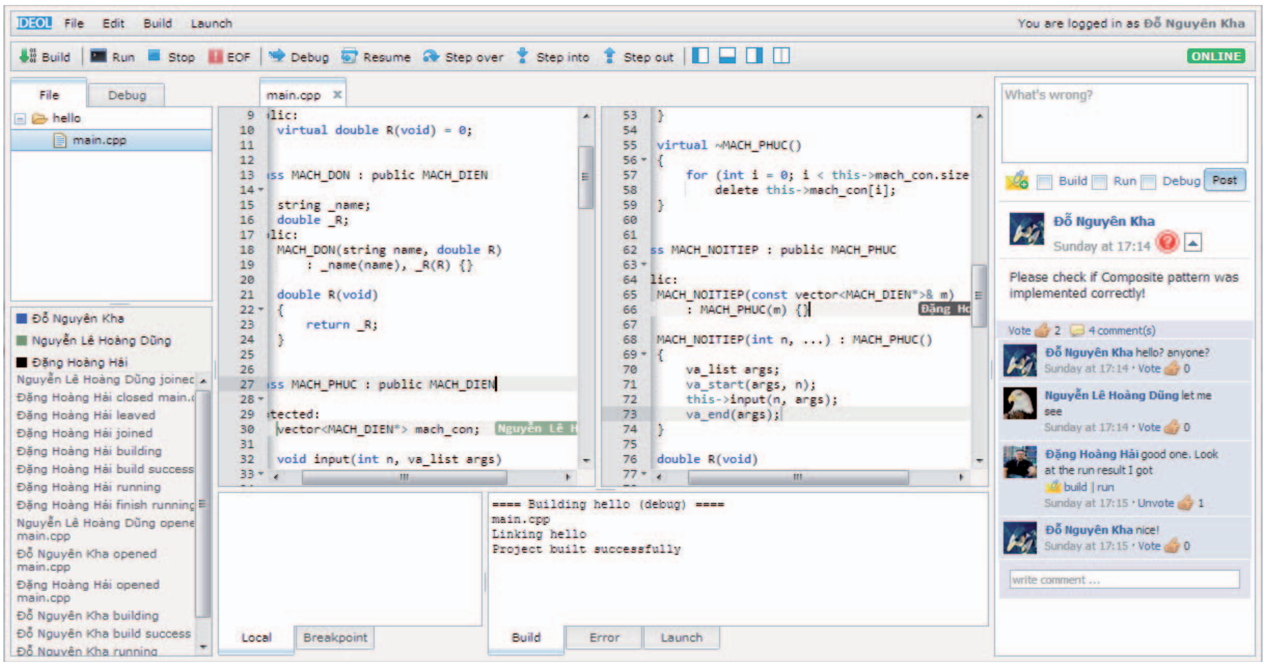
B. IDEOL's Technical Aspects for Concurrent Collaboration

To handle concurrent collaborations among members, we divide user actions into two types, exclusive actions and non-exclusive actions. Exclusive actions include compiling, running, and debugging. At a time, there is at most one exclusive action on a project, held by one member, while many non-exclusive actions can be obtained by anyone to open, view and edit source code, and provide comments. Users are provided with information about these activities' states, especially for exclusive actions, as one cannot make an exclusive action when another exclusive action is being carried out.

IDEOL does not use locking on source code editing, thus many members can edit the same file at the same time. When a a file is opened, a copy of its content and initial revision number are stored on the server main memory (server version). Changes made to the file in clients are tracked by a simple revision system and synced with the server version. The server then broadcasts the new version to all clients. Frequently, the server version will be persisted on the server's secondary storage, e.g. on a hard drive. If the server receives change operations from more than one client on the same revision number, which occurs when students make changes on the same file at the same time, Operational Transformation (OT) algorithms [12] are adopted to merge the changes. OT is a set of technologies for supporting collaborative computing and it has been applied to a wide range of real-time collaborative applications.

Using a server version of source code files can also help resolve conflicts between exclusive and non-exclusive actions. For example, when one member is editing a source code file, another can still compile and execute the program, since the compilation is made on the persisted version on a hard drive, while editing is made on the server version, which is on the main memory. After the execution, another execution with the newly-edited source code can be made as the system frequently updates the persisted version with the server version.

These technical solutions help create a system that allows its users to freely work by themselves even in a collaborative context, with little blocking of other members' activities (only an exclusive action blocks another exclusive action).



```
lic:
 virtual double R(void) = 0;


ss MACH_DON : public MACH_DIEN

 string _name;
 double _R;
lic:
 MACH_DON(string name, double R)
     : _name(name), _R(R) {}

 double R(void)
 {
     return _R;
 }


ss MACH_PHUC : public MACH_DIEN

tected:
 vector<MACH_DIEN*> mach_con;   Nguyễn Lê H

 void input(int n, va_list args)
```

```
}

virtual ~MACH_PHUC()
{
    for (int i = 0; i < this->mach_con.size
        delete this->mach_con[i];
}


ss MACH_NOITIEP : public MACH_PHUC

lic:
 MACH_NOITIEP(const vector<MACH_DIEN*>& m)
     : MACH_PHUC(m) {}                  Đặng Ho

 MACH_NOITIEP(int n, ...) : MACH_PHUC()
 {
     va_list args;
     va_start(args, n);
     this->input(n, args);
     va_end(args);
 }

 double R(void)
```

```
==== Building hello (debug) ====
main.cpp
Linking hello
Project built successfully
```

■ Đỗ Nguyên Kha
■ Nguyễn Lê Hoàng Dũng
■ Đặng Hoàng Hải

```
Nguyễn Lê Hoàng Dũng joinec
Đặng Hoàng Hải closed main.c
Đặng Hoàng Hải leaved
Đặng Hoàng Hải joined
Đặng Hoàng Hải building
Đặng Hoàng Hải build success
Đặng Hoàng Hải running
Đặng Hoàng Hải finish running
Nguyễn Lê Hoàng Dũng opene
main.cpp
Đỗ Nguyên Kha opened
main.cpp
Đặng Hoàng Hải opened
main.cpp
Đỗ Nguyên Kha building
Đỗ Nguyên Kha build success
Đỗ Nguyên Kha running
```

Đỗ Nguyên Kha
Sunday at 17:14

Please check if Composite pattern was implemented correctly!

Vote 2 4 comment(s)

Đỗ Nguyên Kha hello? anyone?
Sunday at 17:14 · Vote 0

Nguyễn Lê Hoàng Dũng let me see
Sunday at 17:14 · Vote 0

Đặng Hoàng Hải good one. Look at the run result I got
build | run
Sunday at 17:15 · Unvote 1

Đỗ Nguyên Kha nice!
Sunday at 17:15 · Vote 0

Figure 1. **IDEOL's main screen**. The left panel features a project explorer (*File*), a debug stack view (*Debug*) and a log reporting other members' activities. In the center, along with the familiar build logs (*Build*), error logs (*Error*), console logs (*Launch*) and a debug variable view (*Local* and *Breakpoint*), is a dual source code editor that supports concurrent editing. The editor also tracks where and what other members are typing.The right panel is a simple issue tracker, allowing members to create and discuss over issues, and to change an issue state to "unsolved" or "resolved". Various build, debug, or execution results can be attached to the discussions, and members can also tag source code files and lines in their posts.

C. *Applying IDEOL in Programming Classes*

IDEOL can be used as a general-purpose IDE with the basic functions as editing, debugging, and executing programs. It is more useful and effective in programming classes where requirements go beyond these basic functions. We can use IDEOL to raise the quality of learning by enhancing interaction among users, better evaluating students' ability, and making programing classes more interesting. For example, the instructor in a programing class can define programming assignments by providing initial code and asking students to complete the rest. The instructor can specify what the students need to do, where to start, and when to complete. While working on the assignment, students can ask the instructor by providing questions directly on the tool and linking them with the specific code. The instructor then provides answers to these questions on the tool. Students may also ask for help from team mates, of course, if they are allowed. Such collaboration is especially useful in group assignments. Finally, assignment submission and grading can be done through the tool.

Moreover, teachers can evaluate students' ability more accurately with the assistance of a simple log on user activities, basing on requests of action. This system can also helpful in programming language courses or programming contests as students do not need to install or configure any software, and the environment is uniform to all users.

IV. Experiment and Evaluation

A. *Research Questions*

Our goal in introducing IDEOL is to improve the effectiveness of learning in programming courses. To aim at this goal, we initially conducted two experiments to evaluate the following research questions.

Research Question 1: Does IDEOL help students to manage their groups better?

Research Question 2: Does IDEOL enhance the interaction among members in a group?

The two experiments were carried out in Introduction to C Programming and Advanced C Programming classes.

In a typical class in our university, students have to do individual exercises and group-based assignments. According to our survey, in addition to 1 to 2 hours meeting face to face to discuss their problems and solutions, students need 1 to 2 hours per week at home to finish their exercises. When working at home, students usually meet troubles: material sharing, idea representation and performance evaluation.

With the collaborative features of IDEOL, we expect this system can enhance the interaction among members, and help students to work together more efficiently.

B. *The first experiment*

The first experiment was conducted in January 2013 in a course of Introduction to C Programming. Fifty first-year students joined this experiment, and were arranged into 18 groups basing on pre-test results. Five out of 18 groups volunteered to use IDEOL for their assignments. We marked these 5 groups with letters A to E, and the remaining groups with letters F to S. The experiment went through two phases.

In the first phase, the students were grouped to complete a series of assignments under a controlled environment in a laboratory session. Each student was located in a separate room from the group members. Students in the F-S group use a traditional desktop IDE - Visual Studio. Students in the A-E groups use IDEOL. All groups can use supporting tools like VNC, Facebook, Skype, Gmail or any other online communication channels. Face-to-face communications are completely restricted, except in a 15-minute break. The session was extended up to 2 hours. An introductory session was delivered to introduce a variation of supporting tools.

The second phase was the same as the first phase but was not controlled by the teaching staff and last up to a week. The groups from the first phase continue to work on another series of assignments, using the same tools. Group members are free to communicate with others by any channels. In addition to the submissions of assignments, a quick post-test and survey was conducted to evaluate the outcomes of the two phases of the experiment.

Only 4 of 5 IDEOL-using groups provided a response after the experiment. Based on the results of pre-tests, we ranked students and selected 4 out of the 13 Visual Studio using groups that had equivalent abilities compared to the 4 IDEOL groups. Out of 12 students from the Visual Studio-using groups, 7 students reported that they had difficulties in tracking the work progress of the whole group, while only 2 out of 12 students using IDEOL said so. For the 10 out of 12 Visual Studio-using students encountered difficulties in supporting other group members who are in need, while the corresponding number for the IDEOL case is 2/12. Also, in the second phase in which students freely did their work at home, according to the feedbacks from the students, the main features such as the tagging mechanism, the collaborative editor, and the capability of debugging and compiling for all group members help them know problems and evaluate the performance of others faster and more exactly. However, 50% of the IDEOL-using students reported that in addition to IDEOL, they had to use Visual Studio due to problems like poor Internet connection.

While our experiments failed to detect any important learning outcomes differences between the two groups, the results obtained from this experiment show potential evidence to confirm our research questions.

The students using IDEOL managed their project quite more easily since they could evaluate and check the performance of others faster and more correctly. With the log information and the collaborative editor, IDEOL helps students answer important questions such as: which of my classmates worked on this piece of code before? how many members are working on the project code at this moment? who is modifying what part of the system? Such information leads to significant increases in students' productivity and project quality [9].

The students using IDEOL may have specific advantages to discover bugs and find solutions more efficiently over the students from the other group. Students in a group do not have to share system configuration information since all of them work on a same version of the same system. IDEOL has the capability to synchronize content of all files and to share debug results in projects.

C. *The second experiment*

The second experiment took place in March 2013 in a course of Advanced C Programming. The experiment was conducted in a different way, with only 31 students from the first experiment. All of the students completed an assignment individually and then were arranged randomly into groups of 2 students each to review each other's source code using IDEOL. A detailed scale of source-code evaluation in various programming aspects was created, and review results were analyzed. We wanted to see how much peer-reviewing through IDEOL could help students identify problems of their source code with the programming aspects listed in the scale. 80% students responded that IDEOL was suitable for that activity. Still several students suggested that a notification system, either through email or Facebook, be added to make it easier to follow changes and discussions. 70% students said that the peer-review activity should be done in conjunction with Facebook. Several students even commented that this kind of activity can be done completely through Facebook. We will discuss this further in the next section.

V. DISCUSSION

At the current stage, IDEOL has limitations. Therefore, efforts are to be invested, so as to extend the model of online IDEs and make complete use of their advantages. For facilitating programming education and training, we propose research questions.

A. *Collaboration*

Social networking sites, especially Facebook, have become nearly ubiquitous at universities. According to our survey in two above experiments, all students used Facebook to discuss their exercises. With lightweight social computing mechanisms such as tagging, new feeds, and notification, social networking sites can enhance the interaction among students in many activities [13]. Moreover, there are researches suggesting that these mechanisms can bridge the gap between technical and social aspects in software development and they have become a significant part of many informal processes [14] [15]. However, it is under consideration that social networking sites may cause disruptions and decrease the performance of students. Is it good to create a strong connection between IDEs and social networking sites to enhance interactions in work?

B. *Progress tracking*

Web-based applications have the unique advantage of tracking users' activities and behaviors. IDEOL can easily collect these data through requests of services sent to servers. Such data also allow teachers to monitor student's working progress and trace changes, so that they can be aware of each individual's strong/weak points and give appropriate adjustments. Furthermore, by data mining, the system may discover useful knowledge about users' habits and interaction patterns, thus can deliver recommendations on work assignment or defect resolution [2]. This aspect is not only invaluable in training, but also in actual industrial software development.

C. Integrated systems for Software Engineering education

To create a thorough learning environment, IDEOL can be used in conjunction with other online solutions, such as course management systems, project management systems, wikis for requirement elicitation and management, software testing and designing tools. These systems should be integrated to create a single environment for all kinds of education, research and software development activities, thus introduce a completely online approach to software engineering with numerous advantages.

VI. CONCLUSION

In the growing trend of online collaborative learning, many Web technologies and tools are being developed to support learners and instructors and to enhance interactions among them. In this paper, we described IDEOL – a Web IDE that does not only offer common functions of a traditional desktop-based IDE but also has unique advantages of the Web environment and social networks. We have represented initial positive results of applying our IDE in programming classes and discussed potential contributions of such systems in collaborative learning. We also have identified research challenges that need to be addressed in order to improve IDEOL's capabilities.

The development of IDEOL is still an ongoing effort. We will implement new features, especially social media mechanisms to satisfy the high demands of users about the software development environment and to fill the gap between learners and instructors. We plan to integrate IDEOL within a broader scale of systems to enhance education, research and software development activities. This will be an efficient environment for its users to interact conveniently, to share practical experiences, and for instructors, to evaluate the students' learning process.

ACKNOWLEDGMENTS

This work was supported by the Vietnam National University – Ho Chi Minh City (VNU-HCMC) under grant B2013-18-01.

REFERENCES

[1] P. Dillenbourg, S. Järvelä, and F. Fischer, "The Evolution of Research on Computer-Supported Collaborative Learning" in *Technology-Enhanced Learning Principles and Products*, 1st ed, N. Balacheff, S. Ludvigsen, T. de Jong, A. Lazonder, and S. Barnes, Netherlands: Springer, 2009, pp. 3-19.

[2] L. T. Cheng, C. R. de Souza, S. Hupfer, J. Patterson, and S. Ross, "Building collaboration into IDEs," *Queue*,vol 1, pp. 40–50, 2003.

[3] M.G. Moore, "Three types of interaction," *The American Journal of Distance Education*, vol. 3(2), pp. 1–6, 1989.

[4] C.H. Lu, G.F. Chiou, M.Y. Day, C.S. Ong, and W.L. Hsu, "Using Instant Messaging to provide an intelligent learning environment," *Proceedings of the 8th International Conference on Intelligent Tutoring Systems* (ITS), pp. 575-583, 2006.

[5] W. Weinberg, "Real Programmers Do Use IDEs - The Case for Integrated Development with Embedded Linux," 2007, [Online]. Available: http://www.linuxpundit.com/cv/docs/IDE_WP.pdf [Accessed April 2012].

[6] F. Mueller. and A. L. Hosking, "Penumbra: An eclipse plugin for introductory programming," *Proceedings of the OOPSLA Workshop on Eclipse Technology eXchange*, pp. 65–68, 2003.

[7] K. E. Boyer, A. A. Dwight,R. T. Fondren, M. A. Vouk, and J. C. Lester, "A development environment for distributed synchronous collaborative programming," *Proceedings of the 13th annual conference on Innovation and technology in computer science education*, pp. 158–162, 2008.

[8] S. Salinger, C. Oezbek, K. Beecher, and J. Schenk, "Saros: an eclipse plug–in for distributed party programming," *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*, pp. 48–55, 2010.

[9] A.V. Deursen, A. Mesbah, B. Cornelissen, A. Zaidman, M. Pinzger, and A. Guzzi, "Adinda: A knowledgeable, browser-based IDE," *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering* (ICSE), vol. 2, pp. 203–206, 2010.

[10] L. Wu, G. Liang, S. Kui, and Q. Wang, "CEclipse: An online IDE for programing in the cloud," *Proceedings of the 2011 IEEE World Congress on Services* (SERVICES), pp. 45–52, 2011.

[11] M. Goldman, G. Little, and R.C. Miller, "Real-Time Collaborative Coding in a Web IDE," *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 155–164, 2011.

[12] C. Sun, "OT FAQ – Operational Transformation Frequently Asked Questions and Answers," 2011, [Online]. Available: http://www3.ntu.edu.sg/home/czsun/projects/otfaq/ [Accessed July 2012].

[13] R. Junco, "Too much face and not enough books: The relationshop between multiple indices of facebook use and academic performance," *Computers in Human Behavior*, vol. 28, pp. 187–198, 2012.

[14] C. Treude, and M. A. Storey, "How tagging helps bridge the gap between social and technical aspects in software development," *Proceedings of the 31st International Conference on Software Engineering*, pp 12–22, 2009.

[15] C. Treude, and M. A. Storey, "Awareness 2.0: staying aware of projects, developers and tasks using dashboards and feeds," *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, vol. 1, pp. 365 –374, 2010.