Towards a Collaborative Integrated Development Environment for Novice Programmers

Trong Khanh Nguyen
Posts and Telecommunications Institute of Technology
35 PUBLICATIONS **303** CITATIONS
SEE PROFILE

Nguyen Ngoc Doanh
Thuyloi University
39 PUBLICATIONS **280** CITATIONS
SEE PROFILE

International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075, Volume-6 Issue-5, October 2016

Towards a Collaborative Integrated Development Environment for Novice Programmers

Khanh Nguyen Trong, Doanh Nguyen Ngoc

Abstract: Integrated Development Environments (IDEs) are one of the most used tools in many programming courses. However, they usually do not support the interaction and collaboration between learners and instructors. The aim of our research is to provide methods and frameworks facilitating the collaboration. The originality of our approach is to place courses, and also code sources at the center of collaboration. From this idea, we have designed and developed a collaborative IDE (CIDE). It is a type of web-based groupware containing common conventional collaborative tools (video-conferencing, instant messaging, and so on) and specific IDE dedicated to the programming practice (write code together, track change, versioning...). In this paper, we will present our collaborative IDE.

Index Terms: Computer Support Collaborative Learning; CSCL; Collaborative and Interactive Programming; CSCW; Collaborative Integrated Development Environment; Collaborative learning.

I. INTRODUCTION

Programming is considered as a complex and difficult intellectual activity, with students struggling through their first programming subject and with educators struggling to teach it [1]. The teaching and the learning of the basic programming language (such as Assembly, Pascal, C, C++, Java...) still have shortcomings. Especially, at many universities of Vietnam, the learners receive the knowledge in a passive way [2]. The practical works in the laboratory are difficult, which a teacher usually guides numerous students simultaneously. The interaction among the learner – teacher, and the learner – learner is still lacking; the observation and examination of the work of a learner is not really accurate [3].

At the Posts and Telecommunications Institute of Technology (PTIT) of Vietnam, the students of the faculty information technology start to learn the basic programming right from the second year. Each institutor teaches about 250 students, which is separated into six classes of theory and ten groups (30 - 40 students per group) of practice.

Actually, the time for each practice is long (4 hours); the management is difficult and depends on the sense of students. Besides, the processes of fixing errors and the demonstration for each student are not optimal. Many students often repeat the same errors; therefore, the instruction for each one takes a lot of time. Especially, the lecturer is almost impossible to track the practicing progress of each student to give appropriate advice or to change lectures accordingly.

Revised Version Manuscript Received on September 25, 2016.

Dr. Khanh Nguyen Trong, Software Engineering, Posts and Telecommunications Institute of Technology, Hanoi, Vietnam.

Dr. Doanh Nguyen Ngoc, Computer Science, Thuyloi University, Hanoi, Vietnam; UMMISCO UMI 209 Lab, IRD/UPMC, France.

Teamwork form is designed to solve this problem. Accordingly, the students shall practice teamwork skills, communication skills and skill of collaboration with other members of the group [4]. These skills play an important role, which helps students becoming a software engineer in the future. In the course of work, students are involved in many group activities; such as collaborate programming, sharing documents, reviewing code and discussing problems and solutions. Learning is no longer an individual academic process, but rather a social one.

Integrated Development Environments (IDEs) are one of the most heavily used tools in many programming courses. IDEs are places where coding activity takes place and the home of many different development tools. Students may interact and collaborate with the instructors and the other students to obtain advices about the codes, to fix each other's bugs, and get a general understanding. However, most of the actual IDEs do not support the collaboration around learners and the teachers, for example, Eclipse, NetBean, Visual Studio and so on.

They use a variety of collaborative tools. These include tools such as: revision control, source control, bug tracking systems, email, and instant messaging. According to [5], the most used technologies are asynchronous, like email, revision control system.

Integrating such collaborative tools into the IDE, and enabling them with awareness of coding processes and artifacts may help to reduce learners' cognitive context switching between tools inside and outside the IDE and make the connection between development and collaboration more seamless [6].

The aim of our research is to provide methods and frameworks facilitating the collaboration among students, and instructors in programming courses. The originality of our approach is to place the courses, and also the code source at the center of collaboration. From this idea, we have designed and developed a collaborative IDE (CIDE). CIDE is a type of web-based groupware containing common collaborative tools (video-conferencing, instant messaging, and so on) and specific IDE dedicated to the programming practice (write code together, track change, versioning...).

The remainder of this paper is organized as follows. In Section 2, relevant research works are reviewed. In Section 3, the designing and development of CIDE are discussed. Some discussions will be presented in section 4. Finally, the conclusion of this work is given in Section 5.

II. RELATED WORKS

Many efforts have been made to implement collaborative

IDE in the past few years. In this section, we briefly discuss recent developments. Generally, there are three categories of collaborative IDE.

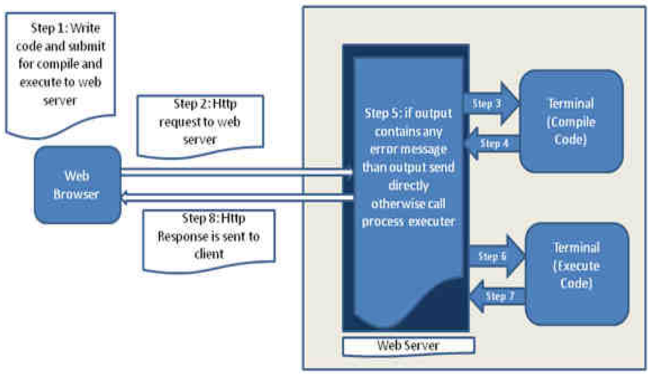
The most popular category is the one that an IDE is used with collaborative synchronous tools, like Skype, TeamViewer, Google Hangout, VNC or Microsoft Meeting [7]. However, these tools demand Internet access, which is usually limited to many laboratories. Moreover, they need high bandwidth.

The second method, which is less common, is the integration of dedicated collaboration tools into IDEs, for example, RIPPLE [8] or Saros [9] for Eclipse. These tools fully support the functionality needed for collaborative work in programming, such as chat, writing real-time collaboration code, screen sharing, etc. However, these tools are often just for some IDEs. In addition, the use of sophisticated IDEs such as, NetBean, Microsoft Visual Studio, etc. for beginners is inappropriate.

Recently, the development of technologies such as web services and cloud computing has opened a new direction for the development of Collaborative IDE: Web-Based IDE [7]. This is the third category, which is very popular nowadays. With this trend, the foundation of IDE (requires a lot of resources) is concentrated in a machine with powerful configuration (server). Users do not have to install any environment, just use any browser to compose and send the code to the computer containing IDE platform to compile, run and return results, as shown in Fig. 1.

In comparison with the methods mentioned above, Web-based IDE has more advantages, such as portability, ability to be independent of devices and operating systems, supporting many different languages, not highly demanding computer resources (at the user side), good management and synchronization. In particular, the ability to collaborate in real-time, Web-based IDE supports a lot better than other methods.

In the literature, there are some IDEs like that, for instance, CodeRun[1], Kodingen[2], Cloud9[3], and eXo Cloud IDE[4]. There are only 2 systems, Cloud9 and eXo Cloud IDE supports collaboration features: chat, code review, writing code simultaneously, and viewing slideshows at the same time. For instance, Dutta *et al.* [10] presented a web-based IDE supporting multi-programming languages. In order to use this



Step 1: Write code and submit for compile and execute to web server
Step 2: Http request to web server
Web Browser
Step 8: Http Response is sent to client
Step 5: if output contains any error message than output send directly otherwise call process executer
Step 3
Step 4
Terminal (Compile Code)
Step 6
Step 7
Terminal (Execute Code)
Web Server

Fig.1. Basic architecture of a Web IDE

IDE, the learners need to have an account. They can use any device with the Internet connection to login, to write code and to execute it. However, the IDE lacks collaborative functionality. Tran *et al.* [3] also proposed an interactive web-based IDE, named IDEOL.

IDEOL supports the basic features of an IDE (write code, compile, run and debug). In addition, this system also offers a number of useful features for collaborative work, such as writing code simultaneously, chat. However, IDEOL still has not become a tool that can be applied in formal teaching. On the pedagogic side, this tool still lacks many features, such as the tool for monitoring learning process, supporting in building the structure for oriented exercises. In terms of the collaboration functionality, IDEOL lacks many features asynchronous work support, such as forums, wiki, etc. In addition, the number of supported languages in IDEOL is limited (C and C ++).

As with IDEOL, Eclipse Che[5] is an IDE that is hosted on a Web server and is run from the user's Web browser. Che is designed to be "a kernel for loading, managing, and running extensions authored in Java that get translated into client-side JavaScript and server-side Java."[6] It supports a variety of Java/J2EE frameworks. This IDE provides unlimited open source community development support and user access. Usability is high, being as simple as using a desktop version of Eclipse or the Netbean IDE. The deployment facility includes a variety of PaaS cloud environments. The integration is seamless and automatic. When a user triggers deployment from an IDE, it connects to the PaaS and pushes the build to the deployment zone. However, this IDE is too complicated to use for novice programmers.

Moreover, a number of tools have been developed to provide real-time awareness of code changes to facilitate coordination and emerging conflicts in collaborative environments. FASTDash [13], ProjectWatcher [14], Palantír [15] and Syde [16] are all examples of this kind of tools. Crystal [17] proactively watches the code and precisely identifies and reports conflicts. There are also growing number of plug-in services being developed for existing IDE's, trying to add more awareness and collaboration features to famiiar tools such as Eclipse JAZZ project [18].

III. CIDE: Collaborative Integrated Development Environment

Based on the experience and also the lack of existing systems, in this part, we will propose a model, and also the technologies to create a Collaborative Integrated Development Environment.

The user that we intend to support is the novice programmer, so that our IDE will be simple but useful enough to use, and particularly to study. Therefore, our IDE will:

- Support basic functions of a compiler/interpreter;
- Allow the collaboration among the users;
- Support multi-language of programming;
- Asily integrate new program languages and collaborative tools.

[1] http://coderun.com
[2] https://kodingen.com/
[3] https://c9.io/
[4] https://codenvy.com/

5 https://www.eclipse.org/che/
6 https://projects.eclipse.org/proposals/flare

For the first purpose, the basic functions of a compiler/interpreter, we just focus on the error checking and the compilation/interpretation process. Because the first issue that a novice concern is whether his/her program is well written, which errors he/she encounter etc.

In the next parts, firstly we will present the meta-model of our collaborative integrated development environment (CIDE). Secondly, we detail the system architecture and the related technologies using in our system.

A. *The meta-model*

The main idea of our meta-model originates from the classical way in which a program is written, compiled/interpreted and executed in a conventional IDE, as shown in the Fig. 2.

This process can be separated into 3 steps. Firstly, students write the code by a code editor, and then the source code is compiled or interpreted into an executable object. There is a little bit different between the compilation and the interpretation: for the compilation, the construction of the executable object need to be completed before the executable, while with the interpretation this one is built and executed simultaneously. But, it does not influents to our model. Secondly, after successfully compiling/interpreting, the students provide input for the executable object in order to run it. Finally, if there are no errors, the output will return to the students. In step 1 and 2, if there are errors, the system will immediately pass to the step 3; and the output is the detail of errors.

From this process, we introduce a meta-model of our CIDE, as shown in the Fig. 3. This meta-model is based on a UML class diagram. This language was chosen because it is widely understood by the scientific community and simple to use. The meta-model can be divided into three parts, as:

- The executable object (*EObject*) and its related parts
- The different tools (*Tool*) and its related parts
- The user, role, and permission

EObject is the main object of the first part. It relates to the machine or binary code that is generated after successfully compiling/interpreting. *EObject* is an aggregation of 4 objects: (i) *SCode* which represents the related source code; (ii) *IBase* which represents the compiler/interpreter of the related language; (iii) *Input* which represents the input data in order to run the *EObject*; and (iv) *Output* which represents the output data after the execution of an instance of *EObject.*

EObject is manipulated via its *Input, SCode* and *Output.* The user, via the different tool, then manages these three objects. The object Tool that associates with *SCode* and *DataController* represents a tool. The later allows managing the *Input* and *Output* of *EObject.* In our model, we plan to integrate easily many tools in the future. Actually, we identify three types of tools: (i) *Editor* represents the editor for the code source; (ii) *CTool* represents the collaborative tools; and (iii) *InOut* relates the tools for the input and output tasks. A user, via his/her role(s), interacts with *EObject* through the tools.

B. *A modular environment*

Such above system, in order to be applied in actual teaching, should not be limited as a collaborative Web-based IDE. Instead, it is an LMS (Learning Management System). Learning management system is the traditional approach to e-learning. Learning in LMS is organized as courses. It usually serves as the online platform for course syllabus releasing, handouts distribution, assignments management, and course discussion for students, teachers, TAs who are the members of the same course.

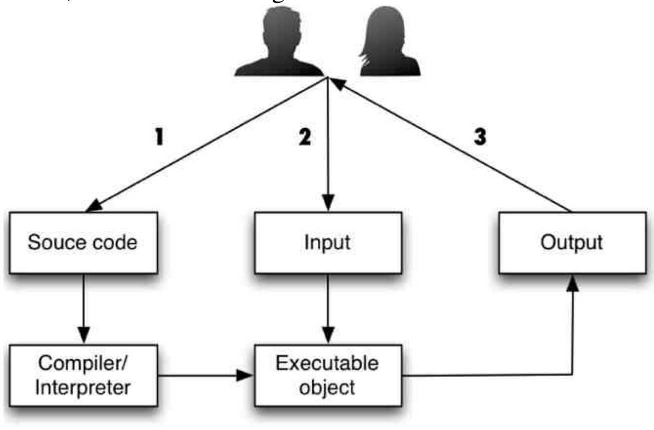


Fig. 2. The compilation/interpretation

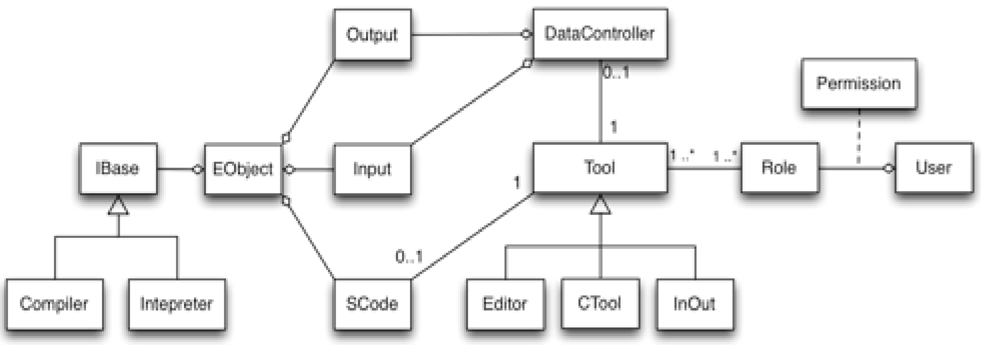


Fig.3. Meta-model of CIDE

Although LMS such as Blackboard, Moodle, and Sakai has been used by numerous universities all over the world to support and improve learning of their students [11]; it is primarily designed for course management purpose and has limited impact on pedagogy. The primary limitations of LMS include lack of personalized control for learners over learning process, limited interaction channel and collaboration manner between learners and educators, restricted interaction and collaboration scope within courses [12].

The integration of CIDE in an LMS allows us to overcome these limitations. Therefore, our CIDE will be surrounded by an LMS, by a numerous of generic collaborative functions.

The core of the system, CIDE, supports the key functions as follows:

- The basic functions of an IDE: compile/interpret, run, debug.
- Real-time collaborative code editor: a group of students can simultaneously write code in a source file and see the changes written by others.
- Integrated discussion tools: enables students, lecturers to discuss together, synchronously and asynchronously.
- Terminal : allows groups of users to run commands on the console.
- Support to compile and run many different languages.
- Save, restart pre-programmed sessions.

Based on that intended purpose, the CIDE has the architecture as the Fig. 5.

The architecture of Web-based IDE will follow the architecture MVC (Model-View-Controller). In which there are 3 layers as following:

- **Presentation layer:** A web application server manages collaborative web interface for writing code to run, showing results and collaboration support functions. This server is built on J2EE platform with Apache Tomcat server. The server to run terminal and collaboration support functions will be developed in Nodejs technology.

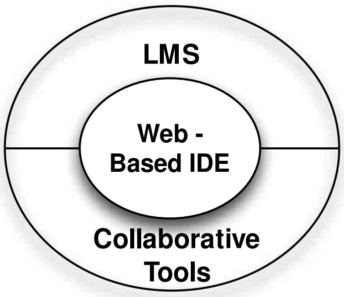


Fig.4. General model

- **Logic layer (Business):** A distributed application server (EJB-Jonas) manages the core of the system (translation machines).
- **Model layer:** The set of source files, and databases (MySQL) to store the information related to the user, the result, the learning process, etc.

C. Technologies Architecture

Our environment is based on a distributed system. It can be viewed as a container in which source codes are loaded, compiled/interpreted, executed by dedicated servers and managed through a collaborative web interface.

We apply technologies of the distributed domain to build CIDE (see the Fig. 6). The front end includes common technologies to represent the content and to support the appearing and interactions of the user, like Jquery, JavaScript, CSS and so on. For the backend, we have:

- **A web application server** based on JSP, Ajax and Servlet (*Tomcat*) to manage the collaborative and adaptable web interface that displays models, experiments, simulators and results in a simple, adaptable and didactic structure of data.
- **An application server** that based on **Enterprise Java bean** and also NodeJS to manage practical courses,

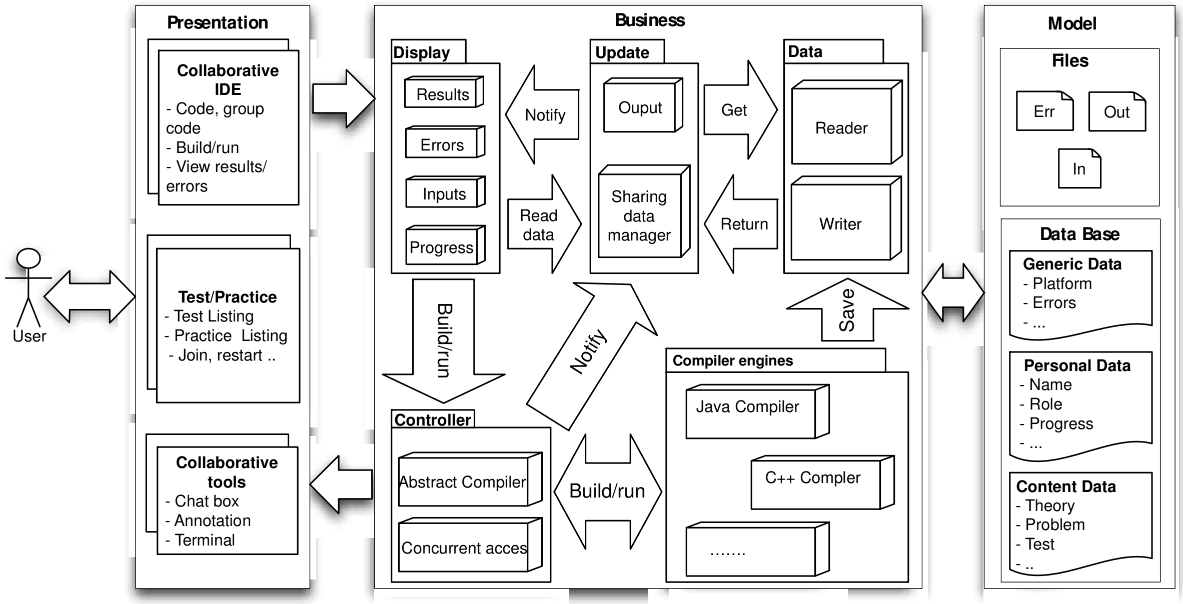


Fig.5. System Architecture

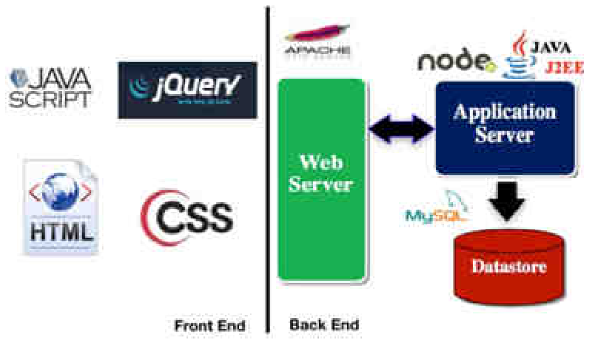


Fig. 6. Technological architecture

compile/interpreter source codes, inputs, outputs and the collaboration.

- **A Database** (MySQL) to store information about users, courses, exercises, and the user's trace that presents the progress of each student.

For LMS and generic collaborative functions, to reduce the developing time and cost, we focus on open source solutions which provide some generic collaborative learning environments and which are possibly extensible with new modules. Because of its feature, we chose Sakai and integrate our CIDE into it.

Sakai is an online Collaboration and Learning Environment. It supports many functions for teaching and learning, communication, collaboration, eportfolios, content and media integration, and administration, for example, Assignments, Calendar, Syllabus, Lessons, Test and Quizzes, Chat, Forum, Wiki and so on. Moreover, it is easy to extend Sakai by adding new features.

The use of Enterprise Java Beans is one of the keys that allow CIDE to be flexible, modular and modifiable. Each module of the framework is composed of several EJBs. Every EJB of the same module is used through a unique interface (determined by the module). To improve the CIDE framework, new EJB could be developed and dynamically deployed without revising old CIDE components. But, these new EJBs must follow predefined interfaces of the CIDE.

Our environment can be deployed on a GRID of computers. Thanks to this distributed architecture, load-balancing rules can be imagined to spread experiment executions over a GRID.

In CIDE, we also use Nodejs to support an interactive terminal. This terminal allows the learner to test your own program by command lines as shown in the Fig. 7.

Another fundamental requirement for supporting collaboration is the awareness that allows one to know what others are doing. The main purpose of an awareness system is to provide information about development activities to help coordinate tasks. In CIDE, the primary responsibility of such a system is to notify students of events relevant to them, such as code changes, comments to discussion threads, user presence, etc.

IV. DISCUSSION

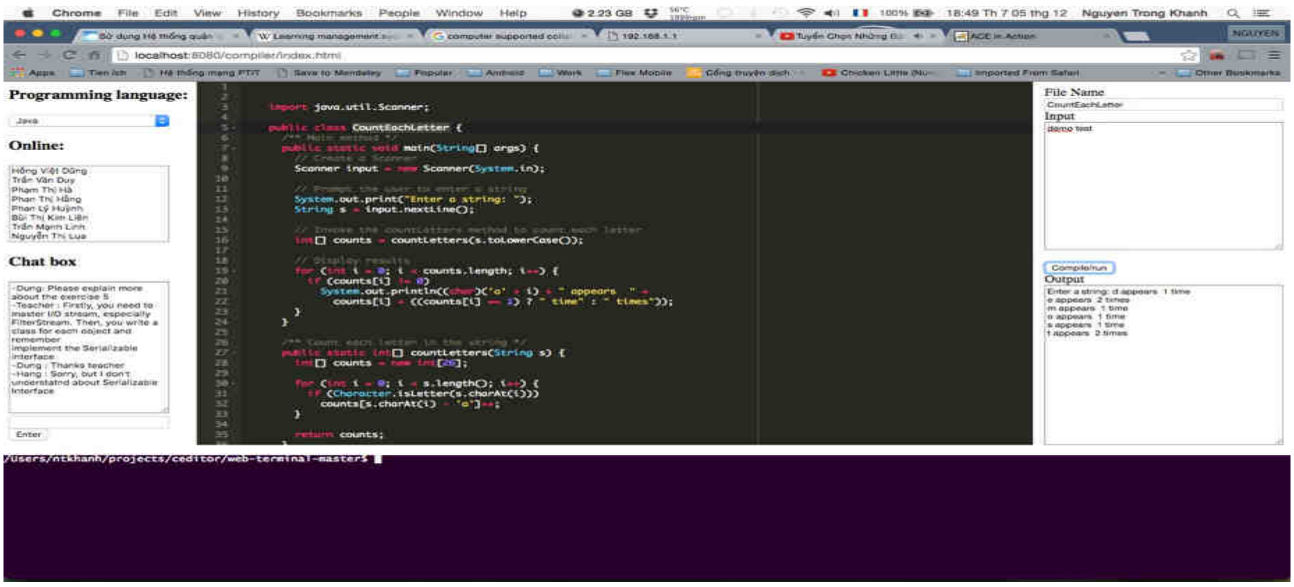
Our CIDE still has limitations. The development of CIDE is still an ongoing effort. Actually, we just implement CIDE with basic functions supporting collaborative programming. Therefore, efforts need to be invested, so as to extend the model of an LMS for basic programming courses and make complete use of their advantages. For facilitating programming education and training, we need to respond to the following issues.

A. *Content Modelling*

The most important issue that we need to deal with is how to model the content of programming courses. We need a standard method to bring the different languages into a unique IDE. It is a complex task; because each course has its own theories and problems; how to represent them, how to re-use them in the similar course ... etc. XML and a specific DSL (Domain Specific Language) is a potential solution in which we can create our own DSL for collaborative languages.

B. *Recommendation Method*

Our environment will track the activities and also the progress of learners. Therefore, based on this information we completely give comments or recommendations to help them improve their knowledge. Actually, CIDE just gives a simple statistic about errors in coding. We need a mechanism more



Hồng Việt Dũng
Trần Văn Duy
Phạm Thị Hà
Phan Thị Hằng
Phan Lý Huỳnh
Bùi Thị Kim Liên
Trần Mạnh Linh
Nguyễn Thị Lụa

-Dung: Please explain more about the exercise 5
-Teacher : Firstly, you need to master I/O stream, especially FilterStream. Then, you write a class for each object and remember
implement the Serializable interface
-Dung : Thanks teacher
-Hang : Sorry, but I don't understatnd about Serializable Interface

```
import java.util.Scanner;

public class CountEachLetter {
  /** Main method */
  public static void main(String[] args) {
    // Create a Scanner
    Scanner input = new Scanner(System.in);

    // Prompt the user to enter a string
    System.out.print("Enter a string: ");
    String s = input.nextLine();

    // Invoke the countLetters method to count each letter
    int[] counts = countLetters(s.toLowerCase());

    // Display results
    for (int i = 0; i < counts.length; i++) {
      if (counts[i] != 0)
        System.out.println((char)('a' + i) + " appears  " +
          counts[i] + ((counts[i] == 1) ? " time" : " times"));
    }
  }

  /** Count each letter in the string */
  public static int[] countLetters(String s) {
    int[] counts = new int[26];

    for (int i = 0; i < s.length(); i++) {
      if (Character.isLetter(s.charAt(i)))
        counts[s.charAt(i) - 'a']++;
    }

    return counts;
```

CountEachLetter

demo test

```
Enter a string: d appears  1 time
e appears  2 times
m appears  1 time
o appears  1 time
s appears  1 time
t appears  2 times
```

```
/Users/ntkhanh/projects/ceditor/web-terminal-master$
```

Fig. 7. CIDE

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

intelligent to efficiently exploit this information.

V. Conclusion

In this paper, we have presented a collaborative IDE that does not only support conventional functions of a traditional IDE but also has unique advantages of collaborative tools. Based on the interaction of an individual with IDEs, the process of compilation/interpretation, we proposed:

- A meta-model of our collaborative IDE that captures the collaboration around learners and instructors and especially allows us to easily integrate new collaborative tools .
- The system architecture of CIDE, with some open source platforms
- The integration of CIDE in an LMS to fully support the teaching and learning

The development of CIDE is still an ongoing effort. Actually, we just implement CIDE with basic functions supporting collaborative programming. In the future, we will implement new features, especially the content modelling to support many different programming courses and also reduce the work of teachers in preparing the course; and the recommendation mechanism.

References

1. Lahtinen E, Ala-Mutka K, et al. (2005). A Study of the Difficulties of Novice Programmers. 10th annual SIGCSE conference on Innovation and technology in computer science education ITiCSE '05.
2. Tran, T.-T., 2013. The Causes of Passiveness in Learning of Vietnamese Students. VNU Journal of Education Research., vol 29., pp 72–84.
3. Hai T. Tran, Hai H. Dang, Kha N. Do, Thu D. Tran, Vu Nguyen. An Interactive Web-based IDE Towards Teaching and Learning in Programming Courses
4. D. Teague and P. Roe, "Collaborative learning - towards a solution for novice programmers," Conf. Res. Pract. Inf. Technol. Ser., vol. 78, pp. 147–153, 2008.
5. Kamrani, A. and Abouel Nasr, E.S. (2008) 'Product design and development framework in collaborative engineering environment', Int. J. of Computer Applications in Technology, Vol. 32, No. 2, pp.85–94.
6. Lewis, S. (2005) Eclipse Communication Framework, Eclipse Foundation, April, Available at http://www.eclipse.org/ecf/, Accessed January 12, 2013.
7. D. McKinney and L.F. Denton, "Developing Collaborative Skills Early in the CS Curriculum in a Laboratory Environment". SIGCSE 2006 Technical Symposium on Computer Science Education. Houston, Texas, USA.
8. H. T. Tran, H. H. Dang, K. N. Do, T. D. Tran, and V. Nguyen, "An interactive Web-based IDE towards teaching and learning in programming courses," Proc. 2013 IEEE Int. Conf. Teaching, Assess. Learn. Eng. TALE 2013, no. August, pp. 439–444, 2013.
9. 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 Inno- vation and technology in computer science education, pp. 158–162, 2008.
10. 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.
11. M. Dutta, K. K. Sethi, and A. Khatri, "Web Based Integrated Development Environment," Int. J. Innov. Technol. Explor. Eng., vol. 3, no. 10, pp. 56–60, 2014.
12. Al-Ajlan A, Zedan H (2008) Why moodle. In: Proceedings of the 12th IEEE international workshop on future trends of distributed computing system (FTDCS'08), 58–64
13. Dalsgarrd C (2006) Social software: e-learning beyond learning management systems. Eur J Open Distance E-Learn
14. Biehl, J.T. Czerwinski, M. Smith, G. and Robertson, G.G. FASTDash: A visual dashboard for fostering awareness in software teams. In CHI, pages 1313–1322, SanJose, CA, USA, Apr. 2007.
15. Schneider, K.A. Gutwin, C. Penner, R. and Paquette, D. Mining a Software Developer's Local Interaction History. MSR 2004, Edinburgh, 2004.
16. Al-Ani,B.Trainer,E.Ripley,R.Sarma,A.Hoek,A. and Redmiles, D. Continuous coordination within the context of cooperative and human aspects of soft- ware engineering. In CHASE, pages 1–4, Leipzig, Germany, May 2008.
17. Hattori, L. and Lanza, M. Syde: A tool for collaborative software development. In ICSE Tool Demo, pages 235– 238, Cape Town, South Africa, May 2010.
18. Brun,Y. Holmes,R.Ernst,M.andNotkin,D.2011. Proactive detection of collaboration conflicts. In Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering (ESEC/FSE '11). ACM, New York, NY, USA, 168-178.
19. Cheng,L.Hupfer,S.Ross,S.andPatterson,J.Jazzing up eclipse with collaborative tools. In 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications / Eclipse Technology Exchange Workshop, pages 102– 103, Anaheim, CA, 2003.

Dr. Khanh Nguyen Trong, was born in 1982. He received his Bachelor degree in Information Technology (IT) at Hanoi University of Science and Technology in 2005, his Master degree in IT at the L'Institut de la Francophonie pour l'Informatique (old IFI) in 2008, and his Ph.D. in IT at the University of Paris VI , France, in 2013. He is currently a lecturer at Posts and Telecommunications Institute of Technology, Hanoi, Vietnam. His domains of interest are Distributed System, Computer Support Collaborative Work, Modelling and simulation of the complex system, Collaborative Simulation and Modelling.

Dr. Doanh Nguyen Ngoc was born in 1981. He received his Bachelor degree in Mathematics at Hanoi National University of Education in 2003, his Master degree in Applied Mathematics at Hanoi National University of Education in 2006, and his PhD in Computer Science at University Pierre and Marie Curie Paris VI, France, in 2010. He is currently a lecturer at Thuyloi University, Hanoi, Vietnam. He is also a member of UMMISCO UMI 209 Lab, IRD/UPMC, France. His domains of interest are Dynamical Systems, Computer Support Collaborative Work, Modelling and simulation of the complex system.