Cinderella, a Prototype of Cloud-Based Integrated Development Environment

Chung Yung*, Yu-Fang Hsieh
Department of Computer Science and Information Engineering, National Dong Hwa University, Taiwan.

* Corresponding author. Tel.: 886-3-8634017; email: yung@mail.ndhu.edu.tw
Manuscript submitted March 1, 2016; accepted May 5, 2016.
doi: 10.17706/jsw.11.11.1132-1138

Abstract: Cloud computing platforms are now popularly used in various domains of computing, and programmers usually use the integrated development environment (IDE) for developing programs. Extended from traditional IDEs with the functionalities of editing, compiling, debugging, and presenting the results, the cloud-based IDEs include two additional features; namely, the interface for connecting with the cloud computing environments, and the intelligent analyses enabled by the cloud computing technology. In the paper, we present a prototype of cloud-based IDE called Cinderella. With Cinderella, the programmers write programs with Android tablets that integrate with servers in cloud computing environment. Cinderella includes the analyses needed by novice Java developers to avoid common programming errors. We evaluate the effectiveness of Cinderella using a survey of questionnaires, and conclude that Cinderella helps the novice Java programmers in reducing the common programming errors. More precisely, there are 77.78% participants that think Cinderella is helpful.

Key words: Cloud computing, integrated development environments, Java programming.

1. Introduction

The increasing popularity of cloud computing is a simple consequence of constant advancement in technology [1]. Migrating to the cloud computing environments, programmers call for the need of new tools for writing programs [2]. In other words, programmers in cloud computing environments write programs using tools and apps on mobile devices, with help from the coordinated servers [3].

A traditional IDE is a development environment that needs to be installed on the computers [4]. One of the disadvantages is that if the environment or hardware changes, reinstallation is required. [5]. And also, it is generally difficult to share and discuss with the other team members using traditional IDEs [6].

In this paper, we analyze the difference between traditional IDEs and cloud-based IDEs, and we propose the criteria for an IDE to be identified as a cloud-based IDE. Based on the criteria, we develop a prototype of cloud-based IDE called Cinderella. This prototype is designed for novice Java programmers to reduce the top errors in developing Java programs.

We investigate on the top programming errors made by novice developers, and we implement the analyses needed for reducing such errors. Through questionnaire, we evaluate the effectiveness of Cinderella when used by computer science students. We conclude that 77.78% of the participants think Cinderella is helpful when they write Java programs.

The rest of this paper is organized as follows. The following section is a brief introduction to the related work. Section 3 discusses the features of cloud-based IDEs. In section 4, we present the design of Cinderella. Section 5 includes the evaluation on the effectiveness of Cinderella. And, at last is a brief conclusion.

2. Related Work

The work described in this paper consists of two major themes: the innovation of cloud-based IDEs, and the study of top errors made by novice Java programmers. This section briefly introduces the related work of both themes.

The innovation of cloud-based IDEs has been a popular topic since cloud computing is pervasive. Bruch et al. introduce the collective intelligence in software development and argue that the IDEs will undergo a revolution in the near future [4]. Especially for embedded systems, Hausladen et al. propose a cloud-based IDE that is entirely based on open-source solutions [5]. With a goal to integrate collaborative development, Mikkonen and Nieminen discuss the elements needed for a cloud-based IDE [6].

Novice programmers make various errors in programming, and the study of top errors helps in both teaching and learning software programming. Jackson et al. identify the top Java errors for novice programmers [7]. McCall and Kolling propose a meaningful categorization of novice programmer errors [8]. And, Mow presents a comprehensive report on analyses of student programming errors [9].

3. Cloud-Based IDE

In this section, we analyze the differences between traditional IDEs and cloud-based IDEs. The general functionalities of traditional IDEs include editing, compiling, debugging, and presenting the results. Extended from these functionalities, we propose the features required for cloud-based IDEs.

An *integrated development environment* (IDE) is a software application that helps programmers developing software systems [4]-[6]. Examples of popularly used IDEs include Eclipse, NetBeans, and Microsoft Visual Studio, among many others. A traditional IDE provides developers with a robust environment for writing, testing, debugging, and deploying code. Compared with the cloud applications [10], [11], we have to install the traditional IDE on computers before using it. Moreover, when software developers want to discuss or share their code, they have to use other applications, such as e-mail and twitter, for this purpose. This motivates the research in extending traditional IDEs for the use in cloud computing environments.

In Fig. 1, we introduce the components needed of an IDE in cloud computing environments. Compared with a traditional IDE, an IDE in cloud computing environments includes three parts: A basic part provides programmers with the functionalities of a traditional IDE, including the tools for editing, testing, debugging and deploying code. An analysis part includes the intelligent analyses enabled by cloud computing technology. With the help from cloud computing environments, there are more and more analyses available for integrating into the IDE. An interface part facilitates the communication with the other services available in cloud computing environments.

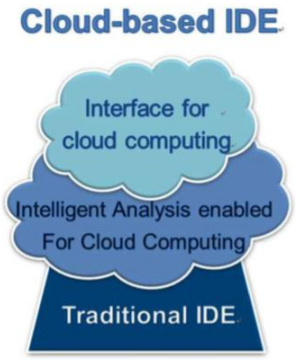


Fig. 1. Components of a cloud-based IDE.

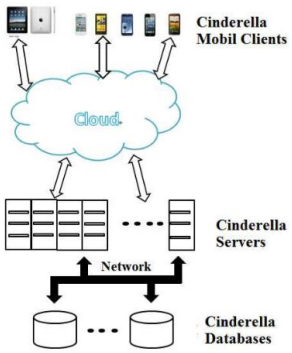


Fig. 2. Overall architecture of Cinderella.

In this paper, we call the IDEs with all the three parts described above as *cloud-based IDEs.*

4. Cinderella

In this section, we present our design for a prototype of cloud-based IDE, called as *Cinderella*. Cinderella allows software developers to write Java programs, to compile and to run the programs in cloud computing environments. We first introduce the overall architecture of Cinderella. Then, we describe the analyses implemented in Cinderella. And, we include a few screenshots of our implementation for Cinderella.

4.1. Overall Architecture

Just like general cloud services, Cinderella is designed for the use with mobile devices, while most of the jobs are done on servers. The overall architecture of Cinderella is shown in Fig. 2. There are three major components in Cinderella:

1) Cinderella mobile clients. Users may use mobile devices with web browsers for connecting to the Cinderella servers. For Android mobile devices, we have a specially designed Cinderella app for Cinderella mobile clients.
2) Cinderella servers. Servers are where most of the jobs are executed. The functionalities of an IDE are provided by a set of services on the Cinderella servers.
3) Cinderella databases. Cinderella requires a few relational databases for storing data.

4.2. Intelligent Analyses

We investigate on the top errors made by the novice Java programmers. McCall and Kolling list the top ten categories of errors that novice Java programmers make [8], shown in Fig. 3(a). In Mow's study of three classes (HCS181, HCS281, and HCS286) [9], the top programming errors by novice Java programmers are listed in Fig. 3(b). And, Fig. 3(c) shows the top ten Java errors listed in the paper by Jackson et al. [7].

In Cinderella, we implement three kinds of analyses for helping to reduce the programming errors: Variable Not Found (VNF) analysis, Semicolon Expected (; expected, or SCE) analysis, and Mismatched Brackets/Parentheses (MBP) analysis.

4.3. Implementation

With an overall architecture shown in Fig. 2, Cinderella consists of two subsystems: the client subsystem and the server subsystem. The detailed design for the implementation of Cinderella is depicted as Fig. 4. The Cinderella client subsystem includes seven utility programs (login, register, option, example, project, code, and result). The Cinderella server subsystem includes of eight services (log, regi, usrfile, newpro, read, cmd, compile, and exe) and three database tables (account, password, and username).

We use six screenshots in Fig. 5 to demonstrate the implementation of Cinderella. When Cinderella starts, it goes to the login screen, shown in Fig. 5(a). Once login succeeds, Cinderella goes to the welcome screen, shown in Fig. 5(b). When Example button is pressed, Cinderella shows the list of examples installed, shown in Fig. 5(c). If Next button is pressed, Cinderella shows the list of programs stored. And then, if the New Project button is pressed, Cinderella goes to a screen shown in Fig. 5(d). When we edit a program, the implemented analyses are running in real time in background. If a potential error is found, a warning message is popped up on screen, as shown in Fig. 5(e). Fig. 5(f) shows the case that Cinderella finds a SCE error, which means that a semicolon is expected.

5. Evaluation

We evaluate the effectiveness of Cinderella in helping novice Java programmers. While there is not many cloud-based IDEs available, JDoodle, an online tool for editing, compiling and executing programs [12], is designed for similar domain to Cinderella, and therefore JDoodle is chosen for comparison. The objectives of our evaluation are university sophomore students with less than one-year experience in Java programming. The following analysis is based on the 33 valid questionnaires collected.

The first part of the questionnaire is about the programming errors usually made. 42.42% of the participants usually make VNF errors; 24.24% of the participants usually make SCE errors; but only 6.06% of the participants

usually make MBP errors.

Category	Frequency
Variable not declared	11.1%
; missing	10.3%
Variable name written incorrectly	8.4%
Invalid Syntax	7.9%
Method name written incorrectly	4.9%
Missing parentheses for constructor call	4.1%
Unhandled exception	3.0%
Class name written incorrectly	2.7%
Method call: parameter type mismatch	2.4%
Type mismatch in assignment	2.4%

(a) Top ten categories of errors listed in [8]

Programming error	HCS181	HCS281	HCS286	Total	% of Total
Variable not found	101	18	41	160	49.8
Identifier expected	23	22	0	45	14.0
Class not found	12	3	1	16	5.0
Mismatched brackets/parentheses	11	6	0	17	5.3
Invalid method Declaration	7	5	0	12	3.7
Illegal start of type	5	6	0	11	3.4
Method not found	4	1	0	5	1.6
Expected	2	4	1	7	2.2

(b) Top programming errors in Meow's study [9]

TOP TEN JAVA ERRORS

Rank	Error	Number of Occurrences	Faculty Identified
1	cannot resolve symbol	81655	Yes
2	; expected	47362	Yes
3	illegal start of expression	32107	No
4	class or interface expected	25650	No
5	<identifier> expected	25223	Yes
6	) expected	21412	Yes
7	incompatible types	15854	No
8	int	14185	No
9	not a statement	13878	No
10	} expected	12808	Yes

(c) Top ten Java errors by Jackson *et al*. [7]

Fig. 3. Top Java programming errors.

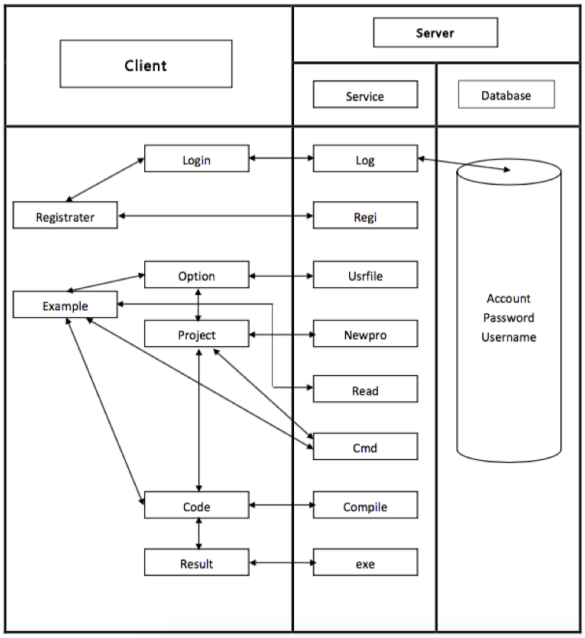
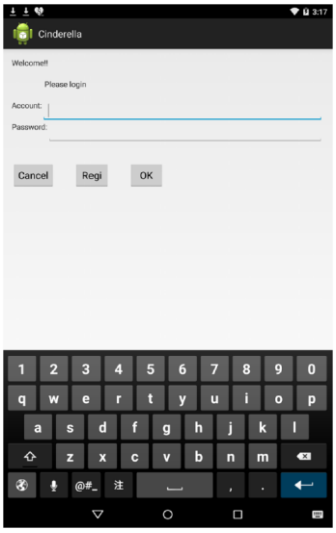
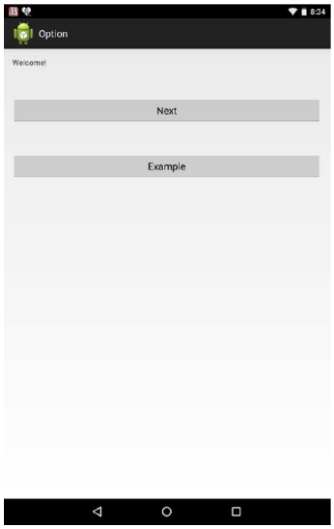


Fig. 4. Implementation of Cinderella.

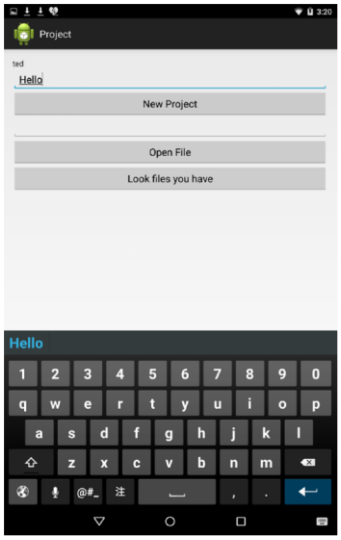


(a) Login

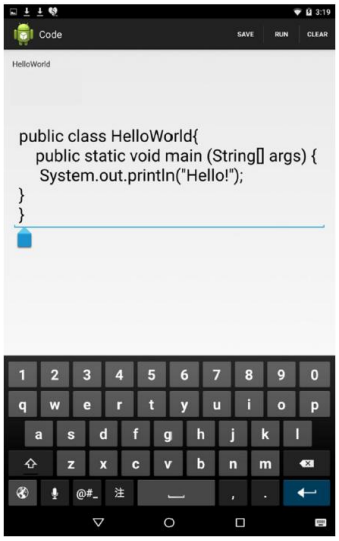


(b) Welcome

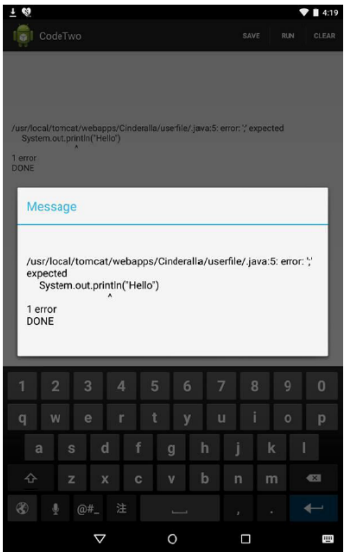
(c) Examples



(d) Stored programs



(e) Editing a program



(f) An error messages

Fig. 5. Cinderella screenshots.

The second part of the questionnaire is about the tools for reducing programming errors. For reducing VNF errors, the percentage of participants that think Cinderella helps is 78.79%, while it is 57.58% for JDoodle, as shown in Fig. 6(a) and (b). For reducing SCE errors, the percentage of participants that think Cinderella helps is 93.94%, while it is 57.58% for JDoodle, as shown in Fig. 6(c) and (d). For reducing MBP errors, the percentage of participants that think Cinderella helps is 60.6%, while it is 39.39% for JDoodle, as shown in Fig. 6(e) and (f). As a summary of reducing programming errors in the three categories, Cinderella has the effectiveness of 77.78% in average, while JDoodle has 51.52% in average.

The final part of the questionnaire is about the effectiveness in promoting the programmers to write programs using mobile devices. None of the participants have the experience of writing programs using mobile devices. After using Cinderella and JDoodle, there are 54.5% of the students that would like to continue writing programs with Cinderella on mobile devices, while none of them would use JDoodle.

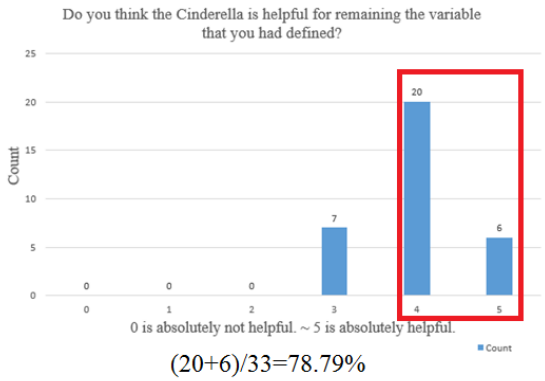
6. Conclusion

In this paper, we propose a prototype of cloud-based integrated development environments for novice Java programmers called as Cinderella. In Cinderella, we implement three analyses for reducing programming errors.

In our evaluation, there are 77.78% of the novice programmers that think Cinderella helps in reducing programming errors, and there are 54.5% of the novice programmers that would like to use mobile devices to develop programs with Cinderella.

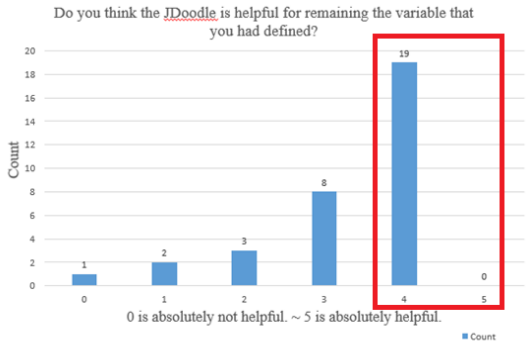
Acknowledgment

This work is partially supported by the Ministry of Science and Technology, R.O.C., under grand number MOST 103-2221-E-259-014-MY3. The authors are grateful to Mr. Meng-Yuan Wu for his help in preparing some of the figures in this paper.



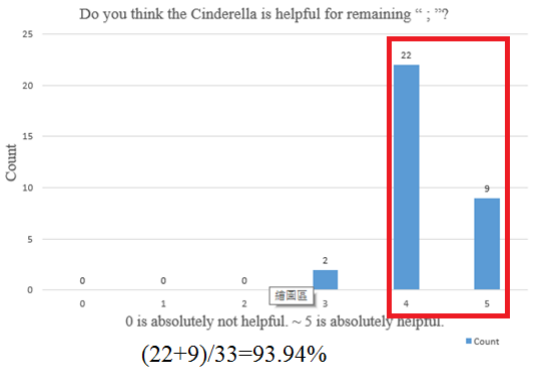
(20+6)/33=78.79%

(a) Cinderella helps in reducing VNF



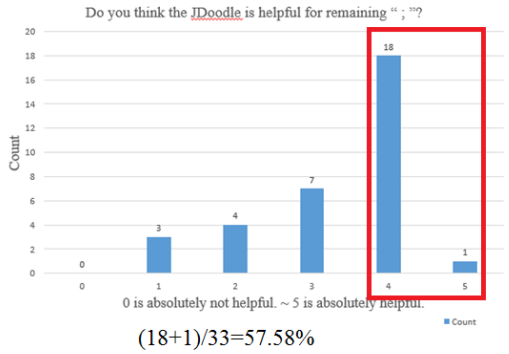
(19+0)/33=57.58%

(b) JDoodle helps in reducing VNF



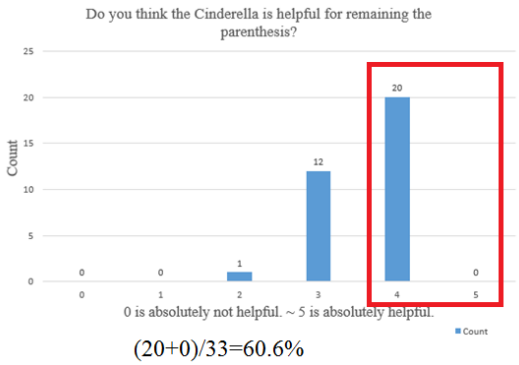
(22+9)/33=93.94%

(c) Cinderella helps in reducing SCE



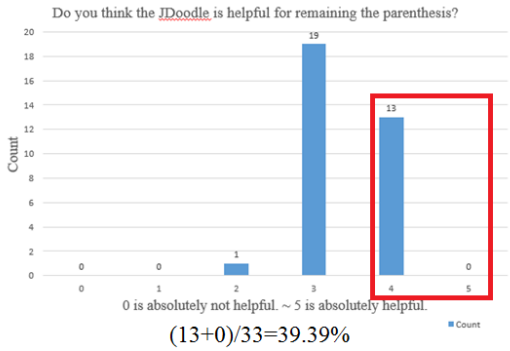
(18+1)/33=57.58%

(d) JDoodle helps in reducing SCE



(20+0)/33=60.6%

(e) Cinderella helps in reducing MBP



(13+0)/33=39.39%

(f) JDoodle helps in reducing MBP

Fig. 6. Comparison on effectiveness of Cinderella analyses vs. JDoodle analyses.

References

[1] Breiter, G., & Naik, V. K. (2013). A framework for controlling and managing hybrid cloud service

integration, *Proceedings of 2013 IEEE International Conference on Cloud Engineering (IC2E)* (pp. 217-224).

[2] Yung, C., & Lin, Y. T. (2015). Implementing TOAST, a tool for agile software project management in cloud computing environments. *Journal of Software (JSW)*, *10(11)*, 1310-1318.

[3] Chang, W. Y., Abu-Amara, H. & Sanford, J. (2014). Transforming enterprise cloud services, *Proceedings of 2014 IEEE Frontiers in Education Conference (FIE)* (pp. 1-8).

[4] Bruch, M., Bodden, E., Monperrus, M. & Mezini, M. (2010) IDE2.0: Collective intelligence in software developament. *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research* (pp. 53-58).

[5] Jausladen, J., Pohn, B., & Horauer, M. (2014). A Cloud-based integrated development environment for embedded system. *Proceedings of 2014 IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Application (MESA)* (pp. 1-5).

[6] Mikkonen, T., & Nieminen, A. (2012) Elements for a cloud-based development environment: Online collaboration revision control and continuous integration. *Proceedings of the Joint 10th Working IEEE/IFIP Conference on Software Architecture and 6th European Conference on Software Architecture (WICSA/ECSA) Companion Volume* (pp. 14-20).

[7] Jackson, J., Cobb, M., & Carver, C. (2005) Identifying top java errors for novice programmers. *Proceedings of the 35th Annual Conference on Frontiers in Education (FIE)* (pp. T4C 24-27).

[8] McCall, D. & Kolling, M. (2014) Meaningful categorisation of novice programmer errors. *Proceedings of 2014 IEEE Frontiers in Education Conference (FIE)* (pp. 1-8).

[9] Mow, I. T. C. (2012). Analyses of student programming errors in java programming courses. *Journal of Emerging Trends in Computing and Information Sciences*, *3(5)*, 739-749.

[10] Wang, Y., Wagstorm, P., Duesterwald, E., & Redmiles, R. (2014) New opportunities for extracting insights from cloud based IDEs. *Proceedings of the 36th International Conference on Software Engineering (ICSE) Companion* (pp. 408-411).

[11] Yanagisawa, H. (2012) Evaluating a web-based programming environment. *Proceedings of the 15th International Conference on Network-based Information Systems (NBiS)* (pp. 633-638).

[12] JDoodle, Retrieved in February 2016, from website: http://www.jdoodle.com/.

Chung Yung received PhD degree in computer science from New York University (USA) in 1999 and BSc degree in computer science and information engineering from National Chiao Tung University (Taiwan) in 1988.

He has been with the Department of Computer Science and Information Engineering of National Dong Hwa University (Taiwan) since 2000. He was also a part-time senior consultant and software project manager between 2003 and 2007. He is currently leading Compiler Technology and Application Laboratory in National Dong Hwa University. His research interests include semantic methods of program analysis, optimizations for cloud software systems, compiler supported software engineering, and programming languages.

Yu-Fang Hsieh received MSc degree in computer science and information engineering from National Dong Hwa University (Taiwan) in 2015 and BSc degree in computer science and information engineering from National Taitung University (Taiwan) in 2013.

She currently works as a program designer in the business operating department of Taihsin International Bank, Taipei, Taiwan. Her major research interests have been in the fields of software engineering in cloud computing environments, software development methodologies, and integrated development environments.