Toward a community enhanced programming education

Ryo Suzuki
University of Tokyo
Tokyo, Japan
1253852881@mail.ecc.u-tokyo.ac.jp

CHI'15, April 18–April 23, 2015, Seoul, Korea.
Copyright © 2015 ACM ISBN/15/04...$15.00.
DOI string from ACM form confirmation

Abstract

As the demand of acquiring programming skills increases, programming education has been an important topic in Human-Computer Interaction. In recent years, several online educational systems have been introduced, and these systems have reduced barriers for novice learners to acquire the fundamental skills of programming. These simple tutorials are beneficial to learn the basics of programming, however, there is still a huge gap between acquiring basic knowledge and putting it to practical use. Therefore, in order to fill this gap, we believe that there needs to be a *community enhanced programming education*, an environment that facilitates users to create a wide variety of tutorials and improve them with the help of community. To address this issue, we propose the two systems: one is a collaborative tutorial creation tool for open source code hosted on GitHub, and the other one is a real-time learning environment that encourages students to interact each other. Our systems aim to facilitate crowds of developers to create a variety of learning resources, as a result, encourage learners to acquire more advanced and practical coding skills. In this paper, we describes the user interface and its implementation, and discuss the future direction of programming education.

Author Keywords

Programming Education, Tutorial Creation, Collaborative Software Development, Integrated Learning Environment

ACM Classification Keywords

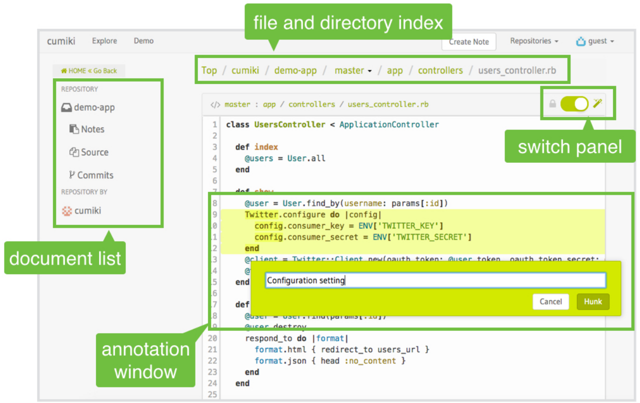
D.2.9 [Software Engineering]: Programming environments.; H.5.2 [Information interfaces and presentation]: User Interfaces

Introduction

As programming skills increases in demand, programming education has become one of the most important topics for a society. In fact, a number of online systems as well as Massive Open Online Courses (MOOCs) have been introduces in recent several years, and millions of people all over the world learn programming with these systemms everyday. For example, Codecademy [1] and Khan Academy [2] provide an interactive learning environment that enables users to learn the basics of programming within the browser interactively.

Since most of these educational systems focus on novel learners, it has been relatively easy to acquire the basics of programming skills. On the other hand, there is still a huge gap between acquiring basic knowledge and putting it to practical use. For example, mastering the basics of JavaScript has become relatively easy; on the other hand, building a practical web application with JavaScript is still difficult to achieve without practical experience. This is due to the lack of interactivity in online resources which would allow users to understand the intricacies of more complex applications of basic coding knowledge.

We believe that one of the key issues in computer science education over next decade would to fill this gap. In order to address this issue, we propose two systems: one is Cumiki, a collaborative tutorial creation tool for open source software code hosted on GitHub, and the other one is a real-time teaching environment for a computer science education in classroom. Cumiki aims to facilitate crowds of developers to create a variety of learning tutorials. Real-time teaching environment focuses on teachers rather than learners, and makes it easier for teachers to improve their tutorials with synchronous feedback from students. In the following sections, we describes the user interface and its implementation, and discuss the future direction of programming education.



```
class UsersController < ApplicationController

  def index
    @users = User.all
  end

  def show
    @user = User.find_by(username: params[:id])
    Twitter.configure do |config|
      config.consumer_key = ENV['TWITTER_KEY']
      config.consumer_secret = ENV['TWITTER_SECRET']
    end
    @client = Twitter::Client.new(oauth_token: @user.token, oauth_token_secret:
    @
  end

  def
    @user = User.find(params[:id])
    @user.destroy
    respond_to do |format|
      format.html { redirect_to users_url }
      format.json { head :no_content }
    end
  end

```

Configuration setting

Figure 1: The interface of Cumiki. Online demo is available at http://cumiki.com/demo/.

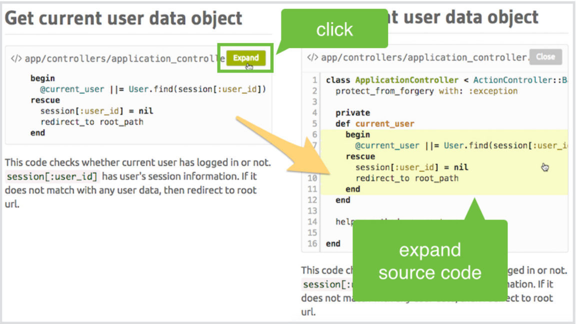
Tutorial Creation Based on Open Source Code

It is said that the software code of open source project is a good textbook to learn how to write practical code. On

[1] http://www.codecademy.com/
[2] https://www.khanacademy.org/computing/

the other hand, it is hard for beginners to read source code of large projects. Software documentation is helpful for programmers to understand how program works, however, it does not provide the recipe about how to create this program.

Cumiki provides an interactive user interface that crowds of developers to create tutorial by annotating the source code. Cumiki is implemented as a web application on Amazon EC2 [3] and its user interface is implemented in HTML and JavaScript and runs entirely within the browser. The server side of Cumiki is implemented in Ruby. The system is closely integrated with a GitHub repository and users can annotate source code of any public repository hosted on GitHub. For extracting information from a git repository via Ruby, we use rugged [4] Ruby library. As a web-based annotation tool, Cumiki has the following features.



Get current user data object

</> app/controllers/application_controlle

```
begin
  @current_user ||= User.find(session[:user_id])
rescue
  session[:user_id] = nil
  redirect_to root_path
end
```

This code checks whether current user has logged in or not. session[:user_id] has user's session information. If it does not match with any user data, then redirect to root url.

t user data object

</> app/controllers/application_controller.

```
class ApplicationController < ActionController::B
  protect_from_forgery with: :exception

  private
  def current_user
    begin
      @current_user ||= User.find(session[:user_i
    rescue
      session[:user_id] = nil
      redirect_to root_path
    end
  end

  help

end
```

This code c   ged in or not. session[:   ation. If it does not m   ct to root url.

Figure 2: Traceable link between code and documentation.

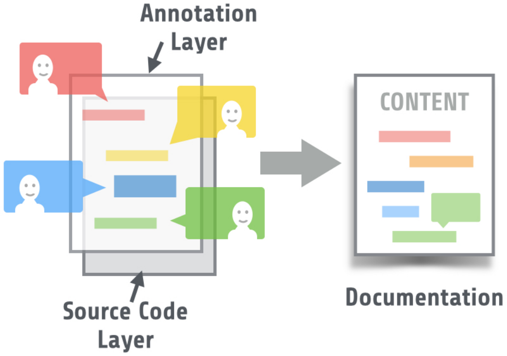


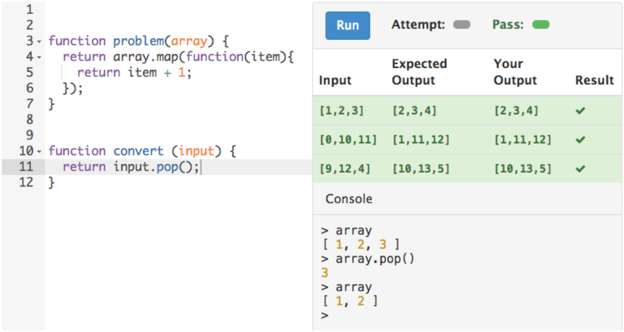
Figure 3: Collaborative annotation among crowds of developers.

Interactive and collaborative annotation: JavaScript-based user interface enables users to create code-embed tutorial by dragging the mouse. After annotating the source code, Cumiki automatically generate a tutorial with code snippets. The user can add a contextual information such as additional code snippets, images, videos, and mathematical equations, that makes it easier to understand the essence of code snippet and how the code works. Unlike the embedded documentation generator, we take an approach to separate code and documentation. Therefore, different developers can annotate the same source file, as a result, crowds of developers can share and accumulate their knowledge about the source code.

[3] http://aws.amazon.com/ec2/

[4] http://rubygems.org/gems/rugged/

Traceability and automated updating:
We implement a one-click link that associates between source code and documentation. As Figure.3 shows, when a button is clicked, Cumiki shows the entire code and highlights a certain piece of code. In addition, our system is able to automatically update the code snippet by analyzing how the code has changed with the information of git versioning system. This feature frees from worry about updating documentation continuously. The mechanism behind this feature is that Cumiki extracts meta data such as line numbers, file name, and commit id from Git repository, and calculate based on the diff data of commit history. Moreover, we propose another approach that separates the layer of annotation from the source code. Our approach makes it possible for crowds of developers to annotate collaboratively and accumulate the knowledge on the source code.



```


function problem(array) {
  return array.map(function(item){
    return item + 1;
  });
}


function convert (input) {
  return input.pop();
}
```

Input	Expected Output	Your Output	Result
[1,2,3]	[2,3,4]	[2,3,4]	✓
[0,10,11]	[1,11,12]	[1,11,12]	✓
[9,12,4]	[10,13,5]	[10,13,5]	✓

Console

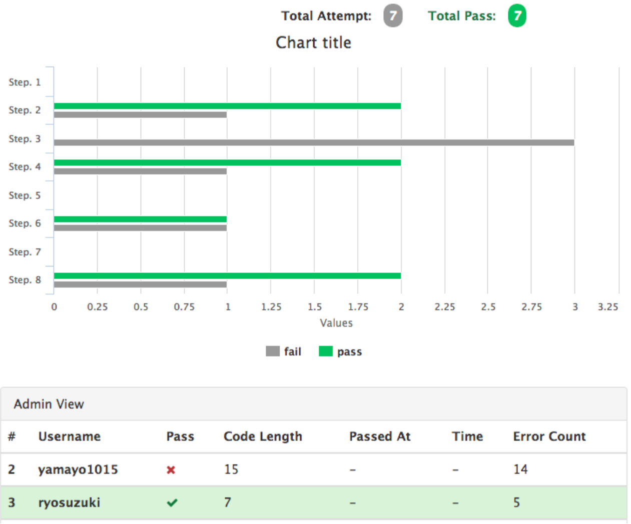
```
> array
[ 1, 2, 3 ]
> array.pop()
3
> array
[ 1, 2 ]
>
```

Figure 4: The user interface of learning enviornment.

Real-time Teaching and Learning Environment

The second system proposed in this paper is an integrated teaching and learning environment for classroom computer science education. The problem in classroom learning is that it is hard for a teacher to grasp students' progresses adaptively. In addition, the interaction among students depends on a teacher's facilitation skills. If the teacher can get continuous feedback from students and facilitate the students' interaction easily, teacher can provide the better learning experience. Therefore, to address this problem, we introduce a cloud-based real-tim educational environment.

The system supports the both learning and teaching aspects of classroom education. First, the learning environment consists from lecture part and problem part. Students learn the fundamental knowledge in the lecture part, and after that they solve the related coding problem. The system encourages the interaction among students. For example, students can see how the best practice or best performance code among the class. It encourage students to learn how to improve their code, as well as motivates them to write good code.



Admin View

#	Username	Pass	Code Length	Passed At	Time	Error Count
2	yamayo1015	✗	15	–	–	14
3	ryosuzuki	✓	7	–	–	5

Figure 5: Real-time information visualization from student low-level output log for adaptive learning.

Second, our system makes it easier for teachers to understand how students work on the problem during the lecture. While students are coding, the system automatically gathers the low-level output data, including how students write code, iterate the debugging processes, handle the error, and interact with the other students. Administration view displays these information in real-time and synchronously, thus it enables for teachers to give a lecture more adaptively, and to improve their tutorials continuously. For these features, the system helps users to improve the lecture and tutorial with the help of community of students.

Related Work

Programming learning is one of the growing fields of research in HCI and there are a number of proposed systesm. In related work, Bret Victor envisions the system for understanding programming with his essay. [12]. Code Hunt [11] offers educational gaming platform. Online Python Tutor [4] focuses on a program visualization. Moreover, there are several related work in MOOCs, for example, RIMES [6] enables teachers to see how students solve the problem in MOOC system.

Our research is also related to crowdsourcing in software engineering. For example, CrowdCode [8] presents web-based IDE that enables crowds of developers to write, test, and debug code. For another example, Collabode [1, 2], browser-based collaborative programming editor, takes different approach by making real-time peer programming easier. Stack Overflow [5] is one of the examples of successful crowdsourcing in software engineering. Prior study shows that over 92% of questions are answered in a median time of 11 minutes [9].

In the literature of Integrated Development Environment (IDE), several systems have been proposed to integrate rich contextual information into IDE. Codelets [10] is an online code editor that has an interactive helper widget to assist the user in understanding and integrating examples on the web. Codetrail [3] and HyperSource [5] embrace an idea that connects source code and online resources such as documentation, examples, error descriptions, and code snippets. In related work of tutorial creation, Community Enhanced Tutorials [7] introduces a novel architecture that create and improve tutorials with the help of community.

Future Impact

Acquiring programming skills have become more and more important not only for software developers but also for end-user programmers. As the demand of programming increases, We believe that learning environment needs to be more adaptive and collaborative, and its tutorials needs to be more practical and diversified. We envision that the learning environment combined with crowdsourced knowledge will enhance the learnability of novice programmer and accelerate software development. The future vision of our system is that learning material and resources are continuously improved with the help of community of teachers, students as well as developers. Programming learning environment in the future will be able to teach a programmer various things, for instance, how to write more effective code, how to design the architecture, what is the necessary software library in the context, and how to use the library. We envision that in the future, more and more people can learn and use programming to create innovative software.

[5] http://stackoverflow.com/

References

[1] Goldman, M., Little, G., and Miller, R. C. Collabode: collaborative coding in the browser. In *Proc. of CHASE'11*, ACM (2011), 65–68.

[2] Goldman, M., Little, G., and Miller, R. C. Real-time collaborative coding in a web ide. In *Proc. of UIST'11*, ACM (2011), 155–164.

[3] Goldman, M., and Miller, R. C. Codetrail: Connecting source code and web resources. *Journal of Visual Languages & Computing 20*, 4 (2009), 223–235.

[4] Guo, P. J. Online python tutor: embeddable web-based program visualization for cs education. In *Proc. of SIGCSE'13*, ACM (2013), 579–584.

[5] Hartmann, B., Dhillon, M., and Chan, M. K. Hypersource: bridging the gap between source and code-related web sites. In *Proc. of CHI'11*, ACM (2011), 2207–2210.

[6] Kim, J., Glassman, E. L., Monroy-Hernández, A., and Morris, M. R. Rimes: Embedding interactive multimedia exercises in lecture videos. *Proc. of CHI'15* (2015).

[7] Lafreniere, B., Grossman, T., and Fitzmaurice, G. Community enhanced tutorials: improving tutorials with multiple demonstrations. In *Proc. of CHI'13*, ACM (2013), 1779–1788.

[8] LaToza, T. D., Towne, W. B., Adriano, C. M., and van der Hoek, A. Microtask programming: Building software with a crowd. In *Proc. of UIST'14*, ACM (2014), 43–54.

[9] Mamykina, L., Manoim, B., Mittal, M., Hripcsak, G., and Hartmann, B. Design lessons from the fastest q&a site in the west. In *Proc. of CHI'11*, ACM (2011), 2857–2866.

[10] Oney, S., and Brandt, J. Codelets: linking interactive documentation and example code in the editor. In *Proc. of CHI'12*, ACM (2012), 2697–2706.

[11] Tillmann, N., Bishop, J., Horspool, N., Perelman, D., and Xie, T. Code hunt: searching for secret code for fun. In *Proc. of SBST*, ACM (2014), 23–26.

[12] Victor, B. Learnable programming. *Worrydream.com* (2012).