Collaborative annotation: Sharing the knowledge between crowds of developers

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

Abstract

The way to share the knowledge behind source code between distributed programmers has a large impact on collaborative software development. Software documentation is a usual way to share and accumulate the knowledge, and developers will benefit from an automated tool that simplifies keeping documentation up-to-date and facilitates collaborative editing. In this paper, we introduce Cumiki, a web-based collaborative annotation tool. Our system is closely integrated with Git version control system, thus helps to maintain traceability between source code and documentation. Cumiki makes it easier for crowds of developers to annotate code collaboratively and accumulate the knowledge behind of the source code. This paper describes the user interface and its implementation, and discuss the future direction of crowdsourcing in software development.

Author Keywords

Crowdsourcing, Collaborative Software Development

ACM Classification Keywords

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

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

Introduction

Software documentation is helpful for programmers to understand how features of software are implemented. To make a good and easy-to-read documentation, developers should not only keep it up-to-date but also add traceable link be- tween code snippets and source code. Several ideas has been proposed to assist programmers to create documentation effectively. One solution is an embedded documentation generator. Embedded documentation generator such as RDoc [1], JSDoc [2] makes it easier to make up-to-date documentation, but developers cannot edit it without the owner ’s permission, thus it could hinder collaboration. 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.

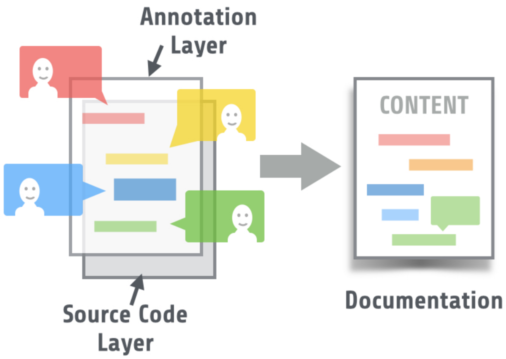


Figure 1: Insert a caption below each figure.

Related Work

Crowdsourcing is one of the growing fields of research in HCI. For example, Soylent [1] introduces a word processing interface that enables writers to shorten, proofread, and otherwise edit parts of their documents on demand with the help of crowdsourcing. The key issue in the future of crowd work is to support more complex and creative work [6], and software development is one of the most challenging areas in crowdsourcing.

As example of creative work, several research . CrowdCode [7] presents web-based IDE that enables crowds of developers to write, test, and debug code. Collabode [2, 3], browser-based collaborative programming editor, takes a different approach in crowdsourcing. Collabode makes an emphasis on synchronous and real-time collaboration. Stack Overflow [3] 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 [8]. In the literature of IDE, several systems have been proposed to integrate rich contextual information into IDE. Codelets [9] is an online code editor that has an interactive helper widget to assist the user in understanding and integrating examples on the web. Codetrail [4] and HyperSource [5] embrace an idea that connects source code and online resources such as documentation, examples, error descriptions, and code snippets.

Implementation

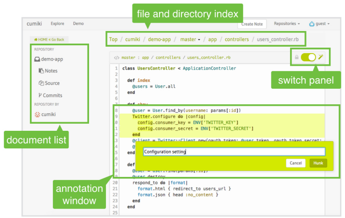
Cumiki is implemented as a web application on Amazon EC2 [4], providing an infrastructure for a scaling

[1] http://rdoc.sourceforge.net/
[2] http://usejsdoc.org/
[3] http://stackoverflow.com/
[4] http://aws.amazon.com/ec2/

application. The 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 [5] Ruby library. As a web-based annotation tool, Cumiki has the following features.

Interactive and collaborative annotation:
JavaScript-based user interface enables users to annotate the code by simply dragging the mouse. After annotating the source code, Cumiki automatically generate documentation with code snippets. The user can edit the annotation with a rich content such as code blocks, images, videos, and even mathematical equations. These rich contextual information makes it easier to understand 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.

Traceability and automated updating:
We implement a one-click link that associates between source code and documentation. As Figure.2 shows, when a button is clicked, Cumiki shows the entire code and highlights a certain piece of code. Moreover, our system is able to automatically up- date 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.



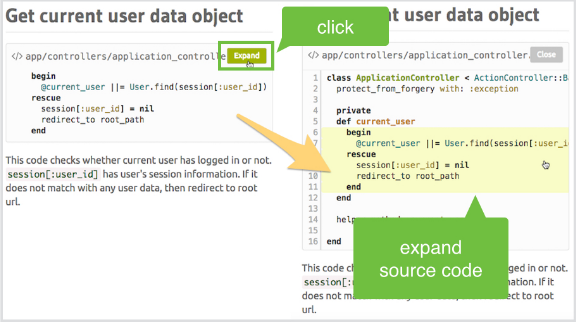
```
class UsersController < ApplicationController

  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
```

Figure 2: One good use of the narrow margin column: callouts that annotate a figure, either with text or a more detailed image.



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 ch ged in or not. `session[:` ation. If it does not ma ct to root url.

Figure 3: Insert a caption below each figure.

Future Impact

Rafa

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

Acknowledgements

This work has been supported by the JSPS KAKENHI grant numbers 25-10499.

References

[1] Bernstein, M. S., Little, G., Miller, R. C., Hartmann, B., Ackerman, M. S., Karger, D. R., Crowell, D., and Panovich, K. Soylent: a word processor with a crowd inside. In *Proc. of UIST'10*, ACM (2010), 313–322.

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

[3] 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.

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

[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] Kittur, A., Nickerson, J. V., Bernstein, M., Gerber, E., Shaw, A., Zimmerman, J., Lease, M., and Horton, J. The future of crowd work. In *Proc. of CSCW'13*, ACM (2013), 1301–1318.

[7] 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.

[8] 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.

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