Design and Implementation of Java Sniper
- A Community-Based Software Code Review Web Solution

Xuesong (Sonya) Zhang
California State University, Fresno
xzhang@csufresno.edu

Bradley Dorn
California State University, Fresno
bradg440@gmail.com

William Jester
California State University, Fresno
williamjester1@gmail.com

Jason Van Pelt
California State University, Fresno
jas.vanpelt@gmail.com

Guillermo Gaeta
California State University, Fresno
ggaeta@csufresno.edu

Daniel Firpo
Claremont Graduate University
Daniel.Firpo@cgu.edu

Abstract
With the advent of the web, more and more information systems are being built web-centric, and more recently, these systems are leveraging the advantages of online community practice as well. This paper describes a lightweight community-based software code review web solution that allows a user to easily upload source code, review, and vote on peer reviews.

1. Introduction

1.1. Software Code Review

A software code review (SCR) refers to the process in which a work product (technical document, code, database design, etc) is examined by the developer and peers. It is typically performed after a software class (or module) has been written and before time is spent on debugging and integrating the software. The process of the peer review involves one or more knowledgeable peers reading the code and submitting comments on areas of the code that might contain errors, have logic problems, not follow standard coding guidelines, or can be improved.

Both industry documentations and research studies indicate that there are many benefits of peer reviews [2,7]. Peer reviews are an important part of software verification and quality assurance, and are a proven mechanism for effective defect removal. It typically takes much less time per defect to identify defects during peer reviews than it does using any of the other defect detection techniques such as using a test case. Thorough peer reviews of source code can result in significant savings over the lifetime of a software development project. In addition, peer reviews provide a learning opportunity for both the person that submits the review as well as the person(s) conducting the review. Less-experienced practitioners can benefit from seeing what a high-quality work product looks like, such as when they help peer review the work of more experienced practitioners. More-experienced practitioners can provide engineering analysis and improvement suggestions that help transition knowledge when they review the work of less-experienced practitioners.

However, it’s also well-known that some traditional code review techniques are inefficient and ineffective [4]. Traditional code review processes are usually constrained by time and location. This can be difficult even for centrally-located employees or project teams, and nearly impossible for many geographically-distributed teams or developers who work independently on a project. Then the developers have to spend lots of time on preparation, trying to remember what code was changed and why, marking the changes, correlating the new version and the old version, and so on. While formal meeting based code review processes can decrease productivity, informal processes (e.g. by email, phone call, or casual conversation) can cause communication issues, such as when no one knows whether the reviews are effective or even happening. Web-based or client-server based code review systems provide a solution to this problem. Developers can upload, access, and review code simultaneously (or asynchronously) via either a secure Internet connection or a virtual private network.
1.2. Open Source and Commercial Peer Review Tools

There are numerous open source and commercial tools that support the peer review process. For example, the author discovered the following systems by searching relevant keywords with Google:

a) http://codestriker.sourceforge.net/ (Code Striker - Open Source Tool)
b) http://www.reviewboard.org/ (Reviewboard - Open Source Tool)
c) http://smartbear.com/codecollab.php (Code Collaborator - Commercial Tool)

Most open source and commercial peer review tools support much more than just the task of code review, as they are often heavily loaded with other features such as chat, forum, wiki, blog, version control integration, reports, workflow, etc. While these features appear to be good selling points, they are not prior purpose of the system or even necessary, and they can consume extra system resources (e.g., memory, bandwidth), cause unexpected errors, and affect system performance. With all the extra features, codes and libraries behind to support them, these tools can also be very complicated to install, manage, troubleshoot, or customize. Small business owners or teachers who want to use such tools in a business or classroom, but lack technical skills to set up and maintain it, immediately find that such a good idea becomes too challenging to implement. This study aims to develop a simple, lightweight, and open source web solution, with a focus on the task of code review only.

Furthermore, most open source and commercial peer review tools are intended for internal use by a company. The intent of this study is to greatly expand the pool of potential reviewers by creating a community around the practice of software peer reviews. The benefits of supporting a community of practice in the software code review process are discussed in the section below.

1.3. Community of Practice

The primary objective of the code review system in this study is to promote computer-based and learning-oriented communities of practice in software quality management. A community of practice (CoP) is a group that works together towards common goals, collaborating on common problems, sharing best practices, supporting one another, and sharing a common identity [3]. In a CoP, knowledge sharing activities involve individuals using the CoP as an effective mechanism for conveying what they know.

A code review system can benefit from online CoP from the following CoP characteristics defined by [1]:

a) Shared sets of interests – membership is organized around the domain issue of code review and software quality assurance.
b) A common identity – members develop shared understanding and common identity as software developers.
c) Shared information and knowledge – members are willing to share information and knowledge in coding issues.
d) Voluntary participation - members voluntarily participate in the community activities, such as responding to requests for help.
e) Autonomy in setting the goals of the community – The community sets its own agenda based on the needs of the members (e.g., creating a code review when there is a need), and these needs can change over time (e.g., closing the code review when the submitter wants to).
f) Awareness of the social protocols and goals of the community – members are aware of the acceptable social protocols, such as how to write a review professionally or respond others with respect, and that the goals of the community are to learn and share.
g) Awareness of membership – members have a reasonable knowledge of themselves and other developers, and their roles and actions within the community.
h) Effective means of communications – a web-based code review system allows users to communicate both synchronously and asynchronously on code reviewing issues.

2. Java Sniper – A community-based software code review web solution

2.1. System Features

The purpose of this project is to develop a community oriented web site that allows registered users to submit source code for review by other registered users. The high level feature set and workflow for the web site is illustrated in Figure 1.
2.1.1. New User Registration. The site allows the public to view a listing of the 10 most recent submissions, however it requires that users be registered in order to submit or participate in a code review. The registration process requests the following registration information from users:
   a) The user's “Real” name
   b) E-mail address (it is used as the unique username)
   c) A password
   d) A password hint

The registration process also requests the following additional information (optional) from registered users.
   a) Date of birth
   b) The URL of the user's web site
   c) A picture that can be uploaded by the user

Users are permitted to update their registration information at any time after they have successfully logged in.

2.1.2. User Login. Each time a user wishes to enter the site, the user is required to provide their registered username (i.e., email address) and a password. The “forgot password” link sends the user to a form where the user provides the username, and the system displays the password hint provided by the user upon registration. After logging into the site, controls are provided to allow the user to access the following information:
   a) Users profile information can be accessed and/or edited
   b) A list of the peer reviews that the user has created
   c) A list of the open peer reviews that the user has joined
   d) A list of all open peer reviews
   e) A list of all peer reviews

2.1.3. Displaying Registered User Information on Dashboard. Any time that a user's name appears on the site, the name is treated as an active link. Clicking on a user's name displays the users profile information. The Registered User Information contains the following required information:
   a) User's “Real” Name
   b) User's picture
   c) User's age
   d) User's website URL
   e) Membership duration (number of days/months/years that the user has been a member of the site)
   f) User's Ranking Score
   g) The reviews created/submitted by that user
   h) The reviews joined by that user

2.1.4. Creating a Peer Review. Any registered user is permitted to create a review. In order to create a review, the site allows the user to do the following:
   a) Upload one or more source files to the site. Note: The current release focus on supporting Java source files, hence the project is named “Java Sniper”, however, it is easy to be customized to accept various types of source files.
   b) Provide a description of the submitted source code. The description is intended to provide reviewers with context as to what the code is intended to do.
   c) Once the files have been uploaded, the source files and associated information is stored in the database. At this point, the review is considered “open” and is available for other users to add comments.

2.1.5. Viewing the List of Peer Reviews. As described in section 2.1.2 above, registered users have the ability to view lists of peer reviews that are filtered as follows:
   a) A list of the peer reviews that the user has created
   b) A list of the open peer reviews that the user has joined
   c) A list of all open peer reviews
   d) A list of all peer reviews

If the user makes a selection from the list of peer reviews that he/she has already joined, then the page can be transitioned to the Source Code Viewing page as described below (section 2.1.6.1). If the user makes a selection from the list of open reviews, then he/she is given the opportunity to “join” the review as described in the following section. If the user selects a review that has been closed, then the user is permitted to view the review source code and all of the review comments, but is not be permitted to add or change any comments.

2.1.6. Conducting a Peer Review. Any registered user has the ability to “join” a peer review. By joining a peer review, users are given the ability to view the source files that were submitted. Users that have joined the review are also given the ability to provide review comments on specific sections of the code. Specific requirements for viewing the source code and providing review comments are provided in the sections that follow.

2.1.6.1 Source Code Viewing. Any registered user is permitted to view the source code associated with any
open code review. Since it is likely that many users will provide comments to the same area of code, the following filter options are supported:

a) Filter by user – Only the comments entered by a specified user are displayed
b) Filter off – All comments entered by all users are displayed

2.1.6.2 Reviewer Comments. While reviewing source code on the code-viewing page (as described in the previous section), users have an ability to select one or more lines of code and provide a comment pertaining to those lines. Each comment entered by a reviewer contains the following information:

a) The name of the reviewer as an active link. Clicking on the reviewers name transitioned to the user’s information page as described in section 2.1.3 above.
b) The review comment text.

2.1.7. Ranking Reviews and Reviewers. The user that created/opened the review is allowed to “up-vote” or “down-vote” the submitted review comments. The “up-vote” and “down-vote” features of the site behave as follows:

a) When a user “joins” a review, his/her ranking is automatically increased by 5.
b) If a reviewer’s comments are “up-voted” the ranking for the reviewer is increased by 5 for each comment entered (e.g. 10 comments yields an increase in ranking of 50). Note: Up-voting indicates that the user felt that the reviewer provided good and thoughtful review comments.
c) If a reviewer’s comments are “down-voted” the ranking for the reviewer is decreased by 5 regardless of the number of comments that the reviewer entered. Note: Down-voting indicates that the user felt that the reviewer poor or disparaging comments.
d) Under no circumstances will a user’s ranking decrease below zero.

2.1.8. Closing a Peer Review. The user that submitted the code review has the ability to close the code review as well. A code review that is closed has the following behavior:

a) No further comments for the code review will be accepted.
b) The review will be displayed in the list of “Closed” code reviews.

2.2. System Design

An online software code review system like Java Sniper needs to support complicated community-wide activities effectively and efficiently, for example, multiple users simultaneously reviewing the same line of source code, good user interface design can help avoid many unexpected problems. Since the registration is open to public, the system needs to be simple, intuitive, consistent, error-tolerant, and easy to use without much training or technical support. Therefore, Java Sniper was carefully and thoroughly designed using a user-centered approach following Human-computer Interaction principles [5].

2.2.1. User Interface Design. The style guide, screen layouts and navigation strategies used are consistent throughout the design of the system. The logo is displayed on the top left hand corner of all web pages, and serves as a link that can redirect the user back to the home page. A traditional navigation bar displaying the main options is consistently available to logged-in users on top of the site. The options are: Dashboard, Account, Submit Code, and Logout.

2.2.2. User Dashboard. When users signs into the system, they are taken to the user dashboard (Figure 3) where they can view their personal information as well as four lists of reviews – the reviews submitted, the reviews joined, all open reviews and all reviews on the system. The personal information is displayed on the left side, and the four lists of reviews are displayed in a content box on the right side which can be toggled based upon the user’s selection of which type of review he/she would prefer to see.

2.2.3. Submitting Code to Create a Peer Review. In order to create a peer review, a user needs to provide a title of the submission, and a textarea to describe the submission (Figure 4). As for submitting the code to Java Sniper, a user can choose to upload a Java file from local computer, or directly paste a segment of code. A user can also easily add another source code file or segment of code, or remove it from the submission. This is enabled by using divs, each with an incremented div name. Because the contents of the preceding div are never modified when code is appended due to the fact that there is already an empty container ready, all user inputted data is preserved throughout the process of adding and removing segments. A hidden textarea keeps track of which sections are active so there is no guesswork to directly
access the contents of a post for inserting into the database.

2.2.4. Source Code Viewing. When the source code files are submitted for review, each file is processed by an open source tool called GeSHi (Generic Syntax Highlighter) and then saved in the database. These source code files are then displayed on the Source Code Viewing page in different colors and fonts according to the programming syntax, providing reviewers with a more familiar view of the code helps users better recognize problem points in the code. The source code viewing page provides consistent code formatting and/or syntax highlighting. Since source code files sometime contain many long lines, the code review viewing page permits scrolling up and down as well as left to right in order to allow all of the source code to be viewed. The code viewing page also resizes with the window, so that users can extend the viewing space if desired.

2.2.5. Conducting a Peer Review. Since the system allows multiple source code files to be submitted for review, also multiple comments to be made on the same line of code, JavaScript is used to choose which file to display, which inline comments to display, and which users to display comments from. All primary information is set to display as block, and any other information that may be accessed through JavaScript prompts is stored in hidden divs. Clicking the name of a file in the code section of the page will allow a viewer to download the code.

Comments can be made on a per-line basis, as well as responses to the segments of code as a whole. The advantage is a simple and straightforward interface. Lines that have inline comments are displayed with yellow toggle buttons. Those without inline comments are displayed with black toggle buttons. When an inline comment toggle display button is clicked, PHP is used to display all inline comments made at the selected line. If the user is logged in, and has joined the review, a textarea allows the user to add an additional inline comment is shown. Beneath this area, all general comments made are shown.

On the left side of the page is the control system for displaying specific segments of code, and comments from specific users. Only one segment of code will be shown at a time; this prevents the user from having to perform excessive scrolling, and gives the owner of the submission greater control over the view frame of various sections of code. Clicking the name of a segment of code will activate it in the center frame. A viewer of a submission has the option to minimize all comments made by all users, minimize no comments made by any user, hide all comments of a specific user, show those comments (by default), and show only the comments of a specific user. Clicking the name of a user in the users bar will take a viewer to their user profile information.

2.3. System Development and Implementation

Java Sniper was developed using open source LAMP (Linux, Apache, MySQL, and PHP) Technologies. The base user interface, dynamic page contents, and reusable functions of Java Sniper were created using HTML, CSS, Javascript, AJAX, and PHP with Adobe Dreamweaver CS5. The data for Java Sniper is stored in a MySQL database and accessed via embedded SQL within PHP pages. Java Sniper is cross-platform – it can be hosted on various operating systems including Linux and windows, it can also be implemented in various web servers including Apache and Microsoft IIS. Users can access the web site via various web browsers, including IE, Firefox, Safari. System architecture is illustrated in Figure 2.

Java Sniper began with a goal of becoming Java source code peer review system. The user requirements were collected and a feature-list was then built. The first thing to be developed was the database structure. The database was structured to work with code segments, various modes of discussion, and users. After the database was structured, the user interface was prototyped and designed to be understandable, intuitive, and simple. PHP scripts supporting main features and functionalities were then developed. The open source utility, “GeSHi” was implemented in a way that stored the code, post-processed as well as pre-processed, in the database. Then, limited web-layer, or JavaScript, code was created and implemented to add features that would allow the site to remain secure, but be more responsive. Throughout the development process, each step was documented, and anything that required any input from the server-environment was noted and adjusted for in a PHP installer. The PHP installer was created and maintained so that Java Sniper could be quickly set up on any server.
3. Discussion

3.1. Comparison to Alternatives

Compared to traditional group meetings within organizations, Java Sniper allows users to upload, access and review source code anytime and anywhere, as long as there is Internet access. This offers great flexibility and permits tremendous savings on time, travel, and other resources. And because the registration is open to public (by default), it is not limited to internal use by a company; it can also help create an online community to share knowledge and learning experience.

Compared to other open source and commercial tools, Java Sniper is very simple and lightweight, with a focus on the task of code reviewing only, therefore it is very easy to install, manage and use. Java Sniper is open source, meaning that it is:

a) at no cost to set up and use;
b) easy to customize (for example, registration can be easily closed or customized to be limited to certain users if necessary);
c) easy to add community features such as chat room, discussion forum, wiki, and blog;
d) easy to add other advanced features such as private messaging function, report abuse function, and review request process;
e) easy to create other plug-ins for organizational use, such as plug-in for Integrated Development Environment (IDE) such as Eclipse or Visual Studio, Revision Control software such as CVS or Subversion, Plug-in for Project Management tools such as Microsoft Project.

3.2. A Scientific Design to be Efficient

Java Sniper requires only PHP, a SQL database, and apache web server to implement. These technologies are popular industry standard for modern web application implementations. Java Sniper can be installed through the web interface, and can be easily implemented by a developer or a knowledgeable web user. Java Sniper is designed to save both user-end and server-side resources. Java Sniper does its server-side processing using PHP. PHP is industry standard, secure, and widely understood. The client-side technology Java Sniper uses is JavaScript, which can be used on a very wide range of devices, from computers, to cell phones. No additional modules or libraries need to be installed or configured to accommodate the functioning of Java Sniper.

Java Sniper stores all settings and information that are not specific to its core functions, such as segments of code or user preferences, in the database. Anything that might require a customization is adapted for because Java Sniper is designed to accept all variables from a single source. Doing this requires minimal server permission and interaction, but gives the program maximum control over how the administrator and users want to implement and use it.

Compared to text, images are typically larger in file size. Adding many images to the database drastically increases the size of the database and decreases performance. Java Sniper accesses the database very frequently - every page is generated from the database. A boost in performance of the database makes every user’s interaction less taxing on the server, and faster from the user perspective. Therefore, in Java Sniper, images are stored in a directory rather than database.

Java Sniper was designed to install and run in any type of web server environment, without having to gain complete access or control over the server. For example, our Java Sniper was developed and implemented in a shared web and database space hosted by public hosting company GoDaddy, Inc. We didn’t need to request for specific PHP, MySQL, or Apache configuration for implementing Java Sniper. This is the benefit of being lightweight software with a focused design purpose compared to many alternative code review software, which may not suffice without web server admin privileges. Installation of a new software or application often imposes a very large security risk to web servers, especially those co-hosting multiple software and applications.

The pre-processing of code coloring in Java Sniper saves an average of two seconds (as compared to the same page, without the process) of both server processor time, as well as end-user response time per code page view. Without this, the web server could risk crashing or responding at a much slower speed, especially with a large number of simultaneous user requests. All other server-side processing is for information structure and managing the permissions and interaction of users.

The in-line commenting system ran primarily on JavaScript. Because of this, no matter how many users
are reviewing the code at once, the inline commenting capabilities do not slow down the server. In addition, because the comments are by default hidden, but can be intuitively and selectively displayed and hidden, the code can be reviewed much faster. The code submitter can also filter through and learn from the contributions much quicker. JavaScript runs inside of a web browser. Because of this, the response time does not have to wait for a user’s Internet connection, and the interaction appears instant. JavaScript also has the advantage that it can be run on just about all computers out-of-the-box, with no software to be installed. Most mobile web browsers, such as that on the iPhone, support JavaScript.

Java Sniper leaves out version tracking, as it is not designed for group- or company-wide code development. Java Sniper is designed for community-wide code review, which involves many random, independent users, who interact and brainstorm in short segmented bursts.

3.3. Efficiency Testing

In order to compare the efficiency of Java Sniper with other popular code review systems on the market (e.g., CodeStriker and Reviewboard) we conducted three different types of testing – installation, code submission, and code review tests.

3.3.1. Installation Test. The installation test involved creating database, installing and configuring scripts, modules and libraries, and getting the system up and running. Reviewboard Review Board is written in Python, using the Django web framework. The installation required the use of sudo commands via the terminal. For the most part, however, all commands were documented, and pretty straightforward. A knowledgeable user should be able to install it in about 20 minutes. The CodeStriker is written in Perl, and it is much more complicated due to the installation and configuration of the supporting modules and libraries. For a complete implementation, it should take about 45 minutes. The following are the results:

a) CodeStriker: 45 minutes
b) Reviewboard: 20 minutes
c) Java Sniper: 15 minutes

3.3.2. Code Submission Test. The code submission test involved navigating to the submission page, loading the page, filling in the information required, and submitting four files containing Java source code designed to output the text "Hello World!" for review. The following are the results:

a) CodeStriker: 12 minutes 20 seconds
b) Reviewboard: 2 minutes 58 seconds
c) Java Sniper: 47 seconds

3.3.3. Code Review Test. The code review test involved accessing the submitted files, reading through it (the files were all very small and reading time was almost negligible), writing and submitting the review (time for writing review was recorded separately because writing time should be independent of all three systems). The following are the results:

a) CodeStriker: 5 minutes 32 seconds + writing time
b) Reviewboard: 2 minutes 4 seconds + writing time
c) Java Sniper: 35 seconds + writing time

The testing results show that Java Sniper is the most efficient of all three systems in terms of installation, code submission, and code review.

In addition, we tried to submit multiple files for review as a whole project. However, we found that neither CodeStriker or Reviewboard provide an avenue for multiple files per submission as Java Sniper does. So we had to submit and review the files separately. Furthermore, the syntax highlighter and in-line commenting in Java Sniper make code reviewing effective and efficient. The feature that allows users to vote on reviews and rank reviewers helps moderate community-wide activities and build trust among community members. These features were designed weakly or not provided in the other systems we tested.

3.4. Alternative Usage

Java Sniper can also be easily customized and used in education for collaboration and knowledge sharing purposes. For example, group project collaboration, and assignments that allow students to practice debugging or developing new functions on existing code, or knowledge sharing across multiple sections, class year, or even schools.

4. Conclusion

This paper describes a community-based software code review web solution that supports the process of code sharing and reviewing. With the community-based nature, a number of state of the art features and a user-centered interface, Java Sniper provides an
effective and efficient web solution for software code review, and contributes to both software development practice and relevant academia research.

5. References


Figure 1. High-level Feature Set and Workflow
Figure 2. System Architecture

Figure 3. User Dashboard
Figure 4. Submitting Code to Create a Peer Review

Figure 5. Viewing and Conducting a Peer Review