# Visual Code Review

Group 5

Daniel Andersson Tenninge

Gustaf Carleson

Johan Björk

Patrik McKiernan

# Project Overview Document

## 1. The users of the system.

This project aims at making the code review process easier and more intuitive as well as more transparent. We believe that currently the software review process at many companies is tedious and waste the developers' time. We also believe that if a company has many different policies for different product releases, it's hard to keep track on all of them.

Our system will make the reviewing of code in large software projects smoother and will allow restrictions on who gets to review what type of code. It will also allow the setting of different review rules for selected parts of the project. In the end, our project will reduce the development time of code in larger software projects as well as it will allow an easier quality assurance of the software.

The users of our system will be experienced software developers, in larger projects, which already work with version control and are familiar with the code review process. The main users will therefore be developers in large software projects, and even the project managers who need to set different policies for different parts of the project.

## 2. Main use of the System.

David, an experienced software developer, is currently working on a software project for a corporation. He has checked out a part of the software from the version control system. After implementing a new feature he then commits his new code back to the version control system. Since this is an important part of the system the company policy states that it needs to be reviewed by two other developers with the required skills. The system then notifies David that his code needs to be reviewed by other developers and therefore puts the actual commit on hold. He is also told that he will be notified whether or not his code passes the reviews and is committed. If it does not pass the review, the other developers will post a note saying what he needs to change. This system allows David to instantly start to work on a new feature or correcting other bugs, instead trying to find someone that will review his code.

Paul, David's coworker, comes back from lunch and logs into the company's code review web page. Once logged in he sees that David has submitted a piece of code that needs reviewing. Since Paul has the qualifications necessary for reviewing this he is allowed to see the committed code and can review it. After looking through David's code and deciding that it is not up to standard he then chooses to reject the code. At the same time as rejecting he uses the feature of the system to let David know what was insufficient with the code. David will now receive a mail that informs him that his code was not submitted and supplies the note from Paul. David will now have to redo his work and submit again for a new review.

The manager of the software project is excited to see how his new system for code review is working out, so he anxiously log in to the web page after lunch and gets a listing of all the pending submits for his project and also gets a listing of all activity from the last two days. There are a few pending commits and since he used to be a programmer on the same project, he sits down and reviews them. Once he is done, the "pending for review" list is empty and he quickly goes through the list of all activity for the past days. He notices that David's new feature is actually in the code base, and immediately goes out and requests a demonstration from David.

## 3. The environment in which the system is used.

The system is to be used in an organization involved in software development. It will most likely be used in the larger scale where code reviews is very important for quality control of the product being developed. It also allows different policies regarding the code reviewing for different parts of the project. Even though we are not targeting smaller projects, we certainly can see a good use for it even for those projects, as the budget for QA is smaller, it's very important to get it right "the first time".
The system will be mostly cross platform. The "backend" will run on the same system as the SCM, but the front end (web server and database) can run on any system appropriate for the given organization. The webbased front end should support most common web browsers and will therefore also be platform independent.

## 4. The scope of the system.

The system will consist of a program running on the server running the version control where it will control all of the commits. When a developer commits code to the repository the backend program will take that piece of code and check the policy that applies to that part. If that part needs to be reviewed the code will be added to the database that holds source code that needs reviewing.

Every user has an account to a webpage that displays the content in the database. Every developer's account has a different authority level bound to it that reflects his or her own area of expertise. The webpage can then check the account's level of authority and only display code up for review that the owner is allowed to review.
The reviewer will have the option to either accept or reject the code directly from the web browser and it will then react accordingly.

The webpage will display the source code in an easily readable way that incorporates syntax highlighting. It will not however be any kind of editor where the reviewers can make changes and then continue the committing process. It will also not implement any kind of reviewing software to automatically check committed code.

| Topic: | In: | Out: |
|---|---|---|
| Version Control System. | | X |
| Web Interface to list all and display code up for review. | X | |
| Plug-in for the version control system in order to control committed code. | X | |
| Database for handling the code up for review and also hold the policies. | X | |
| Web Interface to review and accept code. | X | |

| | | |
|---|---|---|
| Web Interface to manage policies for different parts of the project. | X | |
| Web Interface for editing and committing code that is up for review. | | X |
| The system is responsible for making sure that code gets reviewed. | | X |
| The system shall automatically review code. | | X |
| The systems purpose is to give the manager an overview on how much his employees are committing. | | X |

## 5. The main factors of the system.

Software projects contain millions of lines of code, at certain stages of a project, say a pending release, changes must be monitored rigorously and only the ones meeting high quality standards can be allowed to be included for release.
Normally software engineers work on different releases and different product lines, all which can have different review policies. Different managers are in charge of different releases and product lines. Some developers may have certain skills that make them more suitable to review certain code. If a rejected piece of code is committed again with only minor changes, then the person that rejected it is notified and has the possibility of reviewing it again. This way time spent reviewing will be kept to a minimum.

Factors, which have to be taken into account when developing a system for automatic handling of code review is:

- The manager must be able to set a policy at an appropriate granularity of control.
- Every commit that is handled by a policy must be intercepted correctly
- A commit policy cannot have higher requirements than the users on the system can satisfy.
- The system must be able to handle conflicts of policies.
- The system must only allow users that have the required skills to review changes.
- The system should be able to keep track of the history of a proposed commit, if it gets rejected.
- A commit that has been approved to satisfy the policy must be checked into the desired repository.
- The system must be able to save and record messages from the reviewers.
- Notification when changes occur on a proposed commit to the correct people.
- Since changes can be approved in different order, conflict can arise.

## 6. Technologies and risks.

Components needed are:
(Following each bullet is a list of examples that may be used.)

- Web browser to let a user interact with the website
  - Firefox
  - Internet Explorer
- Web server to present the web site
  - Apache
  - MS IIS
- Dynamic web site which retrieves information from database
  - PHP
  - ASP
  - Ruby on rails
- Version control system
  - SVN
  - CVS
  - Perforce
- Backend to intercept and handle commits to the version control system and database
  - Python
  - Perl
  - PHP
- Database, which contains information about code, policies, commits, reviews, users etc.
  - Postgresql
  - MySql
  - MS SQL
- API for display of syntax highlighted source code and differences from the previous version on the web site
  - SyntaxHighlighter (available at Google Code)

## Potential risks

| Risk type: | Possible risk: |
|---|---|
| Technology | The technologies we've chosen do not interact in the way we expected.<br>We cannot interact with the SCM-system as expected.<br>Conflict handling of source code in the version control system |
| Estimation | We include too many features and are unable to implement them all in time for the project deadline.<br>The solution we've chosen is inefficient and needs unreasonable amounts of CPU. |
| Tool | All, or parts of, our developed software are lost in some way.<br>We're unable to find third party technology needed. |
| People | Developers of our project don't master and/or are unable to learn the technologies used. |
| Requirements | Our software fail to meet the requirements of our intended target group.<br>We may impact the developers' workflow and the users will refuse to use the system. |

## Glossary

**Code Review** – The process where other programmers review code before committing it to the repository.

**Conflict** – A state that occurs when the same file is committed, at the same time, by two users with changes in the same lines.

**Commit** – The act of submitting a proposed change to a repository.

**Branch** – A copy of parts of a repository (source code) that is used to keep track of releases and updates for releases.

**Policy** – The rules specifying how and by whom a certain part of the source code shall be reviewed for approval.

**Repository** – A term related to Source Code Management (SCM) systems, which refers to the place where the current and historical changes to the source tree are stored.

**Syntax highlighting** – Technique that facilitates reading of source code by highlighting text, through different colors and fonts, according to the syntactic category of terms.