Visual Code Review

Group 5

Daniel Andersson Tenninge

Gustaf Carleson

Johan Björk

Patrik McKiernan

1. Preface

Document	Requirements Document
System	Visual Code Review
Intended readership	Customers, stakeholders, project members
Date	2007-12-14

Version history

Version	Rationale and changes	Date	Authors
1.0	First version	2008-01-04	Johan Björk Daniel Tenninge Gustaf Carleson Patrik McKiernan

2. Table of Contents

2.1. The users of the system.

The code review process today is a very time consuming task where developers, instead of continuing to work on some other problem, have to spend time in finding an available reviewer for their code. There is also nothing to actually enforce the code reviewing process, so a developer could in theory commit code to the Source Control Management (SCM) system without the code actually having been reviewed. This poses a serious threat to the code quality of the project.

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 the persons responsible for setting policies that need to set different policies for different parts of the project.

2.2. 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 next day, David comes back to his office and upon reading his e-mail he notices that his code that he committed yesterday did not pass the review process. He reads the supplied note of information from Paul and understands why the code did not pass the review. He goes through his code again and fixes the problems based on Paul's suggestions and when he is done, commits the code again. The code now goes through the same review process as before.

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.

2.3. The environment in which the system is used.

The system is to be used in an organization involved in software development. It will be used in both large and small software development projects 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.

2.4. The scope of the system.

Topic:	***In:***	***Out:***
Version Control System.		X
Web Interface to list all and display code up for review.	X	
Enforce code review policies.	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
Responsibility in making sure that code gets reviewed.		X
The system shall automatically review code.		X

2.5. 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.
- 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.

2.6. Technologies and risks.

Components needed are:

- Web browser to let a user interact with the website.
    - Firefox
- Web server to present the web site.
    - Apache
- Dynamic web site which retrieves information from database.
    - PHP
- Version control system
    - SVN
- Backend to intercept and handle commits to the version control system and database.
    - Python
- Database, which contains information about code, policies, commits, reviews, users etc.
    - MySql
- API for display of syntax highlighted source code and differences from the previous version on the web site.
    - Pygments

2.7. Potential risks

Risk type:	***Possible risk:***
Technology	The technologies we've chosen do not interact in the way we expected. We cannot interact with the SCM-system as expected.
Estimation	We include too many features and are unable to implement them all in time for the project deadline. The solution we've chosen is inefficient and needs unreasonable amounts of CPU.
Tool	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 fails to meet the requirements of our intended target group. We may impact the developers' workflow and the users will refuse to use the system.

3. 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.

Source Code Management – Management of multiple revisions of source code for a project.

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

4. User requirements definition

Nr	***Functional Requirements***
F.1	The system shall intercept commits to the Source Control Management (SCM) system.
F.2	The system shall put intercepted code up for review if a policy states it needs reviewing.
F.3	The system shall store all commits that have been up for review.
F.4	The user shall be able to view code up for review.
F.5	The users shall be able to review code where a policy states they are authorized to do so.
F.6	The user shall be able to accept or reject a pending commit and supply a note to the developer explaining their decision.
F.7	The system shall log the actions of the reviewer. That includes the name of the reviewer, the action taken, the date and the commit.
F.8	The system shall commit the reviewed and approved code to the SCM repository.
F.9	The system shall notify the developer of whether or not his/her code needs reviewing.
F.10	The system shall notify the developer of whether or not his/her code got accepted or not.
F.11	The system shall authenticate users of the system.
F.12	The user, authorized to do so, shall be able to manage user accounts.
F.13	The system shall display all branches and users of the system available for policy setting.
F.14	The user, authorized to do so, shall be able to set policies for users and branches in the SCM system.

Nr	*Non-functional requirements*
F.15	All licensed third party code shall allow unlimited re-distribution of the software without additional cost.
F.16	The system shall be available at least 99 out of 100 times when the SCM system is available (hardware failure excluded).
F.17	The rate of failure occurrence in the software system (hardware failure excluded). • The system shall not loose or corrupt a commit more than one in a thousand. • The system shall not allow reviews by developers with insufficient review rights more than one in a thousand.
F.18	The users of the system shall be able to use the web page of the system after two hours of training. With this training, users of the system will make no more than two errors per day, on average. (An error here implies that the user makes the system do something he did not intend).
F.19	The system shall be capable of handling at least as many users as the underlying SCM system.

5. System architecture

Our system consists of two major parts, the backend, which intercepts and processes SCM communication, and the frontend, which displays the information collected by the backend to the users of the system.

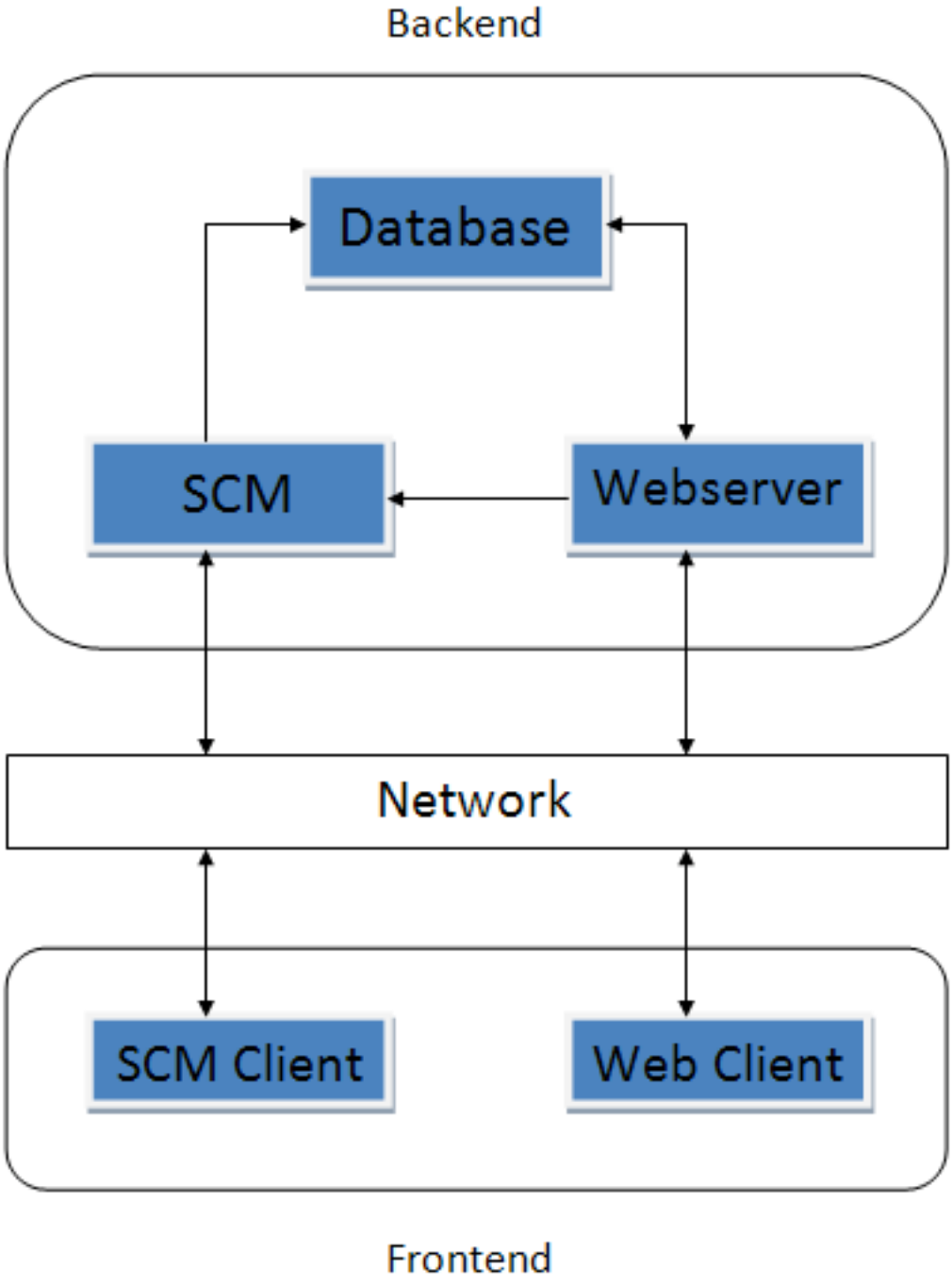

Figure 1: System Architecture

6. System requirements specification

Nr	***Functional Requirements***
#	*User requirement*
	System requirement
F.1	*The system shall intercept commits to the Source Control Management (SCM) system.*
	The system shall implement hooks in the SCM system to intercept commits
F.2	*The system shall put intercepted code up for review if a policy states it needs reviewing.*
	The system shall check the code against the policy database, if stated; the system shall put the code up for review.
F.3	*The system shall store all commits that have been up for review.*
	The system shall store all commits that have been up for review in a database.
F.4	*The user shall be able to view code up for review.*
	The user shall be able to view code up for review using the web interface.
F.5	*The users shall be able to review code where a policy states they are authorized to do so.*
	The user shall be able to review code through the web interface if they are authorized to do so.
F.6	*The user shall be able to accept or reject a pending commit and supply a note to the developer explaining their decision.*
	The user shall be able to accept or reject a pending commit though the web interface and supple a note to the developer explaining the decision
F.7	*The system shall log the actions of the reviewer. That includes the name of the reviewer, the action taken, the date and the commit.*
	The system shall log the actions of the reviewer to a database. The log shall include name of the reviewer, the action taken, the date and the commit.
F.8	*The system shall commit the reviewed and approved code to the SCM repository.*
	The system shall commit the code to the SCM repository using shell scripts when the code is reviewed and approved.
F.9	*The system shall notify the developer of whether or not his/her code needs reviewing.*
	The system shall notify the developer with a text message of whether or not his/her code need reviewing after the commit.
F.10	*The system shall notify the developer of whether or not his/her code got accepted or not.*
	The system shall notify by email the developer of whether or not his/her code got accepted or not, if the not, the reasons will be stated in the included note.
F.11	*The system shall authenticate users of the system.*

	The system shall authenticate users of the system by requesting username and password when necessary.
F.12	*The user, authorized to do so, shall be able to manage user accounts.*
	The user, authorized to do so, shall be able to manage user accounts through the web interface.
F.13	*The system shall display all branches and users of the system available for policy setting.*
	The system shall display all braches and users of the system available for policy setting in the web interface.
F.14	*The user, authorized to do so, shall be able to set policies for users and branches in the SCM system.*
	The user, authorized to do so, shall be able to set review policies for users and branches in the SCM system by using the web interface.

7. Use Cases

Use Case UC1: Submit code to the repository.

Primary Actor: Developer
Stakeholder and Interests:

- Developer: Wants to submit code to the repository using the code review process with as little extra work as possible. Wants to be notified whether or not his code is accepted or rejected.
- Software company: Quality control the code in the system by enforcing review policies.
- Project leader: Wants to be able to check on how the system development is progressing.
- The person responsible for setting policies: Wants the system to enforce the review policies.

Preconditions: Developer is identified and authenticated and has checked out source code from the Source Code Management (SCM) repository. Source code review policies are set for parts of the system that needs policies.

Success guarantee (post conditions): The code review policies are upheld. Code is stored by the system awaiting review if necessary otherwise committed to the repository. The developer is notified of whether or not his code was committed or is now up for review.

Minimal success guarantee (post condition): The change will be recorded by the system and the user will be notified of the result. The code review policies are upheld.

Main Success Scenario (or Basic flow):

1. Developer submits code to the repository.
2. The system applies policies for that code and determines that the code needs reviewing.
3. The system stores the code awaiting review.
4. The system notifies the developer that the code needs reviewing.

Extensions (or Alternate flows)

2.
    a. The system applies the policies for that code and determines that the code does not need reviewing.
        i. The system submits the code, notifies the developer that the code did not need reviewing and the use case exits.
    b. The system determines that there is no policy for the code.
        i. The system submits the code, notifies the developer that the code did not need reviewing and the use case exits.

Special requirements:

- No special requirements.

Technology and Data variation list:
- No variations in technology and data.

Trigger: The developer submits code to the SCM repository.

Frequency of Occurrence: Could be nearly continuous.

Open Issues:
- There are no open issues.

Use Case UC2: Reviewing code.

Primary Actor: Reviewer
Stakeholder and Interests:
- Reviewer: Wants to review submitted code and either accept or reject it based on the code quality. When rejecting code he wants to send the developer a note letting him/her know what was insufficient with the code.
- Software company: Quality control the code in the system by enforcing review policies.
- Project leader: Wants to be able to check on how the system development is progressing.
- The person responsible for setting policies: Wants the system to enforce the review policies.

Preconditions: Reviewer is identified and authenticated and has logged in to the code review web page.

Success guarantee (post conditions): The code review policies are upheld. Code is either committed or rejected based on the actions taken by the developer after reviewing.

Minimal success guarantee (post conditions): The code review policies are upheld.

Main Success Scenario (or Basic flow):
1. The system displays code up for review based on the reviewers' authorization level.
2. The reviewer selects a submit of code.
3. The system displays the code for the reviewer.
4. The reviewer checks the quality of the code and sees that it is up to standard and accepts the code.
5. The system records the action taken by the reviewer.

Extensions (or Alternate flows)
4.

a. The reviewer checks the quality of the code and is not satisfied with the content and rejects the code.
    i. The system records the action taken by the reviewer and the use case exits.

Special requirements:
- No special requirements.

Technology and Data variation list:
- No variations in technology and data.

Trigger: The reviewer is logged in to the code review web page.

Frequency of Occurrence: Could be nearly continuous.

Open Issues:
- There are no open issues.

Use Case UC3: Review action response.

Primary Actor: The system under design
Stakeholder and Interests:
- System under design: Responds to the actions taken by the reviewer by either committing the code or discarding it. Notifying the developer of the actions taken with a note from the reviewer.
- Software company: Quality control the code in the system by enforcing review policies.
- Project leader: Wants to be able to check on how the system development is progressing.
- The person responsible for setting policies: Wants the system to enforce the review policies.
- Developer submitting code: Wants to be notified whether or not his/her code got committed with a note from the reviewer.

Preconditions: Reviewer has reviewed a piece of code and either accepted or rejected it.

Success guarantee (post conditions): The code review policies are upheld. Code is committed to the repository, discarded or kept for further reviewing.

Minimal success guarantee (post condition): The code review policies are upheld.

Main Success Scenario (or Basic flow):
1. The system checks the action from the reviewer and finds that the reviewer has rejected the code.
2. The system removes the code from further reviewing.
3. The system logs the reviewers' action.
4. The system notifies the developer with the supplied note from the reviewer.

Extensions (or Alternate flows)

1.
    a. The system checks the action from the reviewer and finds that the reviewer has accepted the code.
        i. The system compares the code review state with the code policy and finds that the policy is met.
            1. The system commits the code to the repository.
            2. The system removes the code from further reviewing.
            3. The system logs the reviewers' action.
            4. The system notifies the developer with the supplied note from the reviewer and the use case exits.
        ii. The system compares the code review state with the code policy and finds that the policy is not yet met.
            1. The system updates the code review state.
            2. The system keeps the code for further reviewing.
            3. The system logs the reviewers' action and the use case exits.

Special requirements:

- No special requirements.

Technology and Data variation list:

- No variations in technology and data.

Trigger: When a reviewer has taken an action in the review process.

Frequency of Occurrence: When a reviewer has taken an action in the review process.

Open Issues:

- There are no open issues.

Use Case UC4: Setting up policies.

Primary Actor: Person responsible for setting policies (PRP).
Stakeholder and Interests:

- The PRP: Wants the system to enforce the review policies.
- Software company: Quality control the code in the system by enforcing review policies.
- Project leader: Wants to be able to check on how the system development is progressing.

Preconditions: PRP is logged in and authenticated with permissions to set policies.

Success guarantee (post conditions): Eventual changes are saved.

Minimal success guarantee (post conditions): The code policies are not corrupted.

Main Success Scenario (or Basic flow):

1. The PRP selects to set a policy for a branch.
2. The PRP is presented with a list of all the branches in the system for which policies can be set.
3. The PRP selects a branch to set policies on, that currently has no policies set for it.
4. The PRP sets a policy for the branch.
5. The PRP saves or discards the changes.

Step 2 - 5 repeats until the PRP is content.

Extensions (or Alternate flows)

1.
    a. The PRP selects to set a policy for developers.
        1. The PRP is presented with a list of all the developers in the system on which policies can be set.
        2. The PRP selects a developer, which has no policies already set.
            a. The PRP selects a developer, which already has policies set.
                1. The previous policy for the developer is shown to the PRP.
        3. The PRP sets a policy.
            a. The PRP does not set a policy.
        4. The PRP saves the changes.
            a. The PRP does not save the changes.
        5. Steps 2 – 5 continues until the PRP is content and then the use case ends.
3.
    a. The PRP selects a branch for which a policy is already set.
        i. The previous policy is shown to the PRP.
4.
    a. The PRP does not set a policy.

Special requirements:

- No special requirements.

Technology and Data variation list:

- No variations in technology and data.

Trigger: The PRP has logged in to the administration web page.

Frequency of Occurrence: When the PRP needs to set a new policy.

Open Issues:

- There are no open issues.

Use Case UC5: User management.

Primary Actor: System administrator
Stakeholder and Interests:

- Employee: That his user, if he needs one, has the permissions he requires.
- Software company: That each employee that needs a user, has one, and that the user has permissions according to the company policy.

Preconditions: The system administrator is logged in and authenticated.

Success guarantee (post conditions): Eventual changes are saved.

Minimal success guarantee (post conditions): Existing user accounts are not corrupted.

Main Success Scenario (or Basic flow):

1. The administrator is presented with the users of the system.
2. The administrator adds a new user to the system.
3. The administrator sets the properties on the user.
4. The administrator saves the changes.

   Step 2 repeats until the administrator is content.

Extensions (or Alternate flows)

2.
   a. The administrator removes a user from the system.
      1. The user is removed from the system.
      2. The use case continues at step 4.
   b. The administrator edits a user.
      1. The administrator edits the properties for the user.
      2. The use case continues at step 4.
3.
   a. The administrator does not set the properties of the user.
      1. The use case continues at step 4.
4.
   a. The administrator discards the changes.

Special requirements:

- No special requirements.

Technology and Data variation list:

- No variations in technology and data.

Trigger: The system administrator has logged in to the administration web page.

Frequency of Occurrence: When the administrator needs to edit users.

Open Issues:

- There are no open issues.

8. System evolution

8.1. Assumptions on which the system is based

- The users of the system use SVN as their source control system.
- The developers that submit changes to the repository are familiar with SVN.
- All users of the system have a web browser (Firefox).

8.2. Anticipated future changes to the system

We expect our user requirements to change, as to include demands on inter operability with other systems. Potential candidates for these systems could be other SCM engines, bug tracking systems and other company specific in-house software. Another area where we expect changes is the web-based front end. Here we expect to see demands on different ways to visualize the data collected by the system.

Looking at the backend part of the system, we don't anticipate many changes, the only thing is that in the future the software the system will handle will be bigger, and more developers, so making the backend faster may be requested.

9. Appendices

9.1. Database description

Entity / Relationship diagram

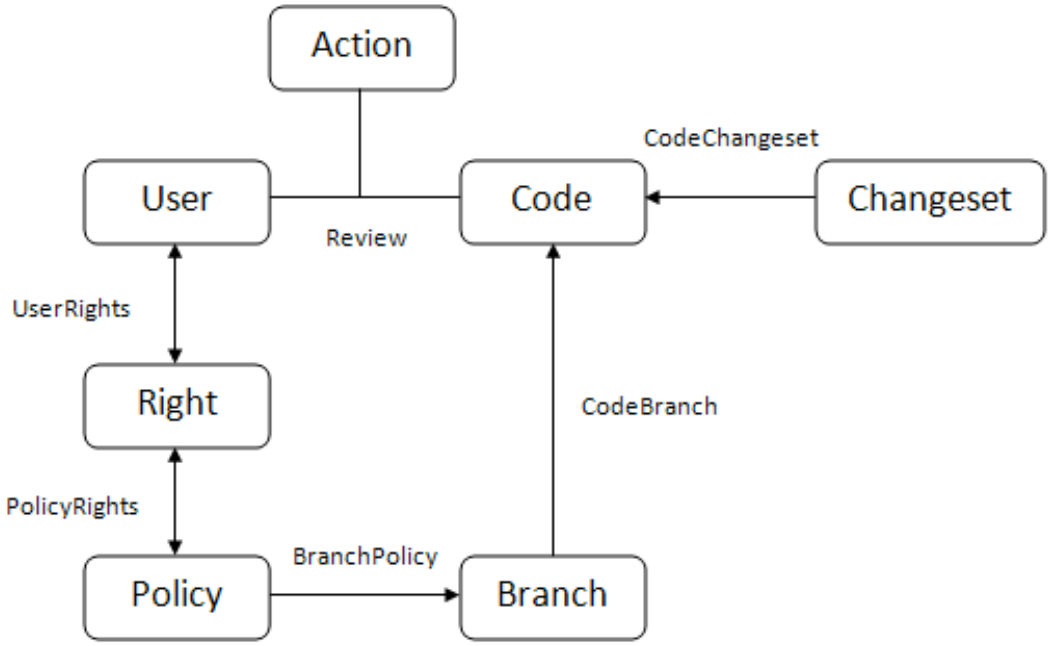

Figure 2: Entity/Relationship diagram

Entity / Relationship	***Description***
User	Contains user information such as login, password and e-mail.
Right	Describes the review rights a user can posess (i.e. C++, SQL, req. engineering).
Action	The actions that a reviewer can take (i.e. accept, reject).
Policy	Consists of a set of rights to be applied to a branch.
Code	Contains the committed code and its relevant information.
Branch	Represents a branch in the Source Control Management (SCM) development.
Changeset	Every set of code committed belongs to a changeset.
Review	Connects objects which are part of a review.
User Rights	Connects users with their rights.

10. Indexes

10.1. Figure Index

10.2. Alphabetic Index