The Codestriker Guide

Version 1.9.5

David Sitsky

The Codestriker GuideVersion 1.9.5

by David Sitsky

Table of Contents

List of Figures

Chapter 1. Introduction

1.1. What is Codestriker?

Codestriker is a web application which facilitates collaborative code reviewing. Authors create code review topics, where the nominated reviewers will be automatically notified by email. Reviewers then submit comments against the code on a per-line basis, and can also view comments submitted by the other reviewers as they are created. Emails are sent to the appropriate parties when comments are created, as an alert mechanism. The author is also free to submit comments against the review comments. Once all reviewers have finished, the author has all review comments available in a structured fashion. All information is stored in a relational database, which Codestriker can search over. All text is stored internally as UTF-8, which supports reviews in all major languages.

Special support is provided for integration with CVS (http://cvshome.org), Subversion (http://subversion.tigris.org), Perforce (http://www.perforce.com), ClearCase (http://www.ibm.com/software/awdtools/clearcase) and Visual SourceSafe (http://msdn.microsoft.com/ssafe). source control management systems, for the display of coloured diffs and for the ability to view original and new files in their entirety to assist in the review process. Codestriker can be optionally linked with a bug tracking system, such as Bugzilla. Other SCM systems will be added in future releases, given demand and help from the developer community. Codestriker can also be linked with LXR (http://lxr.sourceforge.net) to allow for fast lookup of code entities, to assist in the review process.

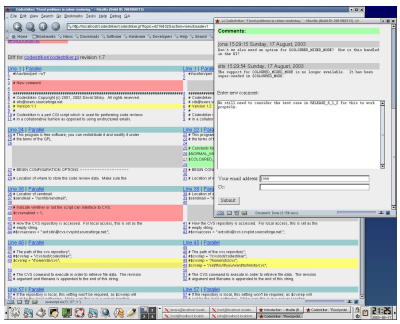
Codestriker is written in cross-platform Perl. It can run on any variant of UNIX (Linux, *BSD) and has been deployed on Win32 systems (Windows 98 and above). It can be used in conjunction with a number of relational databases, including MySQL, PostgreSQL, Oracle, SQL Server, and other ODBC-compliant databases. The Codestriker home page is located at: http://codestriker.sourceforge.net. The Codestriker SourceForge project page, which contains information on file releases, defect lists, mailing lists and information on the CVS repository is located at http://www.sourceforge.net/projects/codestriker.

An example Codestriker topic can be seen at http://codestriker.sourceforge.net/cgi-bin/codestriker.pl?topic=7063366&action=view. To view the current list of open topics, go to http://codestriker.sourceforge.net/cgi-bin/codestriker.pl. From here, there are links to create new topics.

This document is available as a PDF: codestriker.pdf.

A screenshot of Codestriker is action can be seen in Figure 1-1.

Figure 1-1. Codestriker Screenshot



1.2. Document Structure

The remainder of this document is in three parts. The next part is concerned with installing Codestriker and customising it. Following that is the user's guide. Lastly, details on Codestriker's code structure and database schema is presented.

1.3. We Want to Hear from You!

Please send feedback, good or bad, to us. We are continually striving to improve Codestriker, and all feedback is useful. The best place to send feedback is either to the Codestriker mailing list <codestriker-user@lists.sourceforge.net> or to me personally at <sits@users.sourceforge.net>.

Chapter 2. Installation

This chapter is concerned with installing Codestriker on your system. This requires the following steps:

- creating the Codestriker database;
- configuring the webserver; and
- unpacking and configuring Codestriker.

2.1. Codestriker database creation

Codestriker stores all code review topics and comments into a relational database. Currently, MySQL (http://www.mysql.com), PostgreSQL (http://www.postgresql.org), Oracle and SQL Server are supported, but any database system can be used, provided it has an implementation of Perl's DBI interface, which is the case for all major database implementations.

2.1.1. Using MySQL

MySQL can be used on either UNIX or Window platforms. It is available for download from http://www.mysql.com. Make sure you use at least version 4.1 or above, as this supports UTF8 databases. *Note under Solaris, it seems at the time of writing that only the 32-bit version of Perl and MySQL (and DBD::mysql) can be used, the 64-bit versions don't work.* For Linux RedHat distributions, the necessary packages required are mysql, mysql-devel, mysql-server, which may or may not already be present on your system. Depending on your UNIX operating system, starting mysql will be something like the following:

```
/etc/rc.d/init.d/mysql start
```

For Windows, there will typically be a desktop icon or menu shortcut for starting mysqld.

Once mysql is running, it is necessary to create the Codestriker database and create the codestriker user. Under UNIX, a command like the following will be required:

```
% mysql -u root mysql
```

For Windows, there should be a shortcut available for getting a mysql prompt.

At the mysql prompt, issue the following command, but substitute a suitable database password instead of cspasswd as shown here.

```
CREATE DATABASE codestrikerdb CHARACTER SET utf8;

GRANT SELECT,INSERT,UPDATE,DELETE,INDEX,ALTER,CREATE,DROP,REFERENCES
 ON codestrikerdb.* TO codestriker@localhost IDENTIFIED BY 'cspasswd';

FLUSH PRIVILEGES;

QUIT
```

You can check the Codestriker database at any time from the command line, by issuing the following command, and entering the database password:

```
mysql -u codestriker -D codestrikerdb -p
```

If required, the Codestriker database can be dropped, by entering in the following command at the mysql command prompt as user root:

```
DROP DATABASE codestrikerdb;
```

2.1.2. Using PostgreSQL

At the time of writing PostgreSQL was only available for UNIX platforms, however it is quite possible by the time you are reading this that a version for Windows will be available. The project page for PostgreSQL is: http://www.postgresql.org. For Linux RedHat distributions, the postgresql, postgresql-server and postgresql-devel packages are required. Make sure you are using at least version 7.1, as prior versions had restrictions on the size of "text" fields, making it impractical for use with Codestriker. PostgreSQL can be started with a command like:

```
/etc/rc.d/init.d/postgresql start
```

To create the Codestriker database and user, enter the following commands:

```
% createuser --username=postgres -d -A codestriker
% createdb -E UTF8 --username=codestriker codestrikerdb
```

If the last command claims UTF8 is an unknown encoding, try the value UNICODE. If that still fails, you have a distribution which wasn't configured with --enable-multibyte. Try downloading the latest version of PostgreSQL.

Make sure your `pg_hba.conf` file is suitable configured, in particular for authentication. This file is often located in `/etc/postgres`.

You can check the Codestriker database interactively at any time with the following command:

```
% psql -U codestriker codestrikerdb
```

If required, the Codestriker database can be dropped, by entering in the following command:

```
% dropdb --username=codestriker codestrikerdb
```

2.1.3. Using Oracle

Codestriker has been deployed under Linux 2.4 using Oracle 8i. If you have Oracle installed, you most likely will have somebody that can easily create a new Codestriker database for you. Install the DBD::Oracle Perl module, and modify the `$db` variable in the `codestriker.conf` file appropriately so that Codestriker knows how to connect to its database. An example is given in the configuration file. For more advanced connection strings, please consult the DBD::Oracle man page.

2.1.4. Using SQL Server

Codestriker has been deployed using SQL Server on Win32, via the ODBC interface. The first step is to create a "Codestriker" system data source, by going to "Control Panel" -> "Administrative Tools" -> "Data Sources (ODBC)". From here, select the "System DSN" tab, and click "Add". Select the driver named "SQL Server", then click "Finish". Enter in "Codestriker" for the name textfield, "Codestriker database" for the description textfield, and select the appropriate SQL Server from the server dropdown box, then click "Next". Choose the authentication appropriate for your site, and work your way through the final configuration options. Modify the `$db` variable in the `codestriker.conf` file appropriately (see the example) so that Codestriker knows how to connect to this datasource. For more advanced connection settings, please consult the DBD::ODBC man page or online manual.

2.1.5. Other Databases

Codestriker uses the Perl DBI package, a portable API that supports all the major database systems. The database creation code is abstracted so that it can work on a diverse range of database systems. Support for a new database system involves writing a single Perl module in the `Codestriker::DB` package and registering it in the `Codestriker::DB::Database` module. There are a number of example database modules there.

2.2. Configuration

This section is concerned with unpacking the Codestriker distribution into a suitable location, and then configuring it. For UNIX distribution, the following commands may be appropriate on your system:

```
% mkdir /var/www/codestriker
% cd /var/www/codestriker
% tar zxvf /from/installed/location/codestriker-X.Y.Z.tar.gz
% chown -R apache.apache /var/www/codestriker/codestriker-X.Y.Z
```

Here "apache" is the user which runs the Apache server. It could be "nobody" under different systems. Check with the **ps auxww** command, or check your Apache configuration files. Under Windows, the Codestriker distribution could be unzipped into a suitable location under `c:\program files`, or just `c:\codestriker`.

The next task is to edit the `codestriker.conf` configuration file to reflect the settings on your site. The file is documented with examples to assist in setting appropriate values. The file is in Perl syntax, so lines starting with a '#' indicate a comment.

2.2.1. Codestriker Database

The `$db` variable should be set to a DBI URL representing the Codestriker database that was created, as specified in Section 2.1. Basically, if you are using PostgreSQL, this should be:

```
$db = 'DBI:Pg:dbname=codestrikerdb';
```

For MySQL, this would be:

```
$db = 'DBI:mysql:dbname=codestrikerdb';
```

If your database is situated on a different host, for example "dbhost", this could be modified to:

```
$db = 'DBI:mysql:dbname=codestrikerdb;host=dbhost';
```

In this situation, you need to ensure that the webserver host has permission to connect to the database on dbhost. Check the MySQL documentation for further details. The database user and password also need to be specified. If your username was "codestriker", and the password was "cspasswd", the settings would be just:

```
# Database user.
$dbuser = 'codestriker';

# Database password.
$dbpasswd = 'cspasswd';
```

Other examples for other database systems are present in the configuration file.

2.2.2. Email

When a code review topic is created, or a comment against a review is made, an email is sent out as a notification mechanism. Codestriker needs to know what mail host it can use for sending email messages. The configuration file default is "localhost":

```
# Location of the mailing host.  This is used when sending out codestriker
# comments.
$mailhost = 'localhost';
```

If your mail server requires SMTP authentication for sending emails, the username and password can be set via the `$mailuser` and `$mailpasswd` parameters.

```
# Set the user and password parameters if $mailhost requires SMTP
# authentication.  If commented out, it is assumed authentication is
# not required.
$mailuser = 'smtpuser';
$mailpasswd = 'smtppasswd';
```

If these values are commented out, it is assumed SMTP authentication is not required.

2.2.3. Compression

Some of the HTML pages generated by Codestriker can be quite large, depending on the review size. If your deployment is operating to users outside an intranet, it may be worth enabling this option to enable compression. Note, IE doesn't support receiving compressed HTML, so setting this option will have no effect. Initially, it is

best to leave this option turned off (the default), and only to enable it if there is a significant performance problem.

```
# Indicate whether to try and compress output if the client browser
# supports it.  This can make a tremendous difference in bandwidth,
# especially over slow links.
$use_compression = 0;
```

2.2.4. Source Code Management Systems

This part of the configuration deals with informing Codestriker what source code control systems you use. By doing this, Codestriker can then display reviews with revision information, and then allow the reviewer to view the entire contents of a file before a change, and with a change applied. When a review is created, the user specifies which source control system it is applied against (there may not be any, if it is just a simple patch or text file). For many Codestriker deployments, there may only be a single SCM system. There is currently support for CVS, Subversion, Perforce, Visual Source Safe, and ClearCase. Here are examples from the codestriker.conf file:

```
# Valid repositories which may be selected at the create topic screen.
# The order shown here is the order presented in the option list.  Most
# deployments will only require a single repository to be specified.
# Comment out / modify entries appropriate for your deployment.
#
# If this list is empty it won't be possible to view the entire contents of a
# file before the proposed change and/or after. All of the places
# in the application that ask for, or display repository information will
# be hidden and lastly, it will be impossible to make a diff review topic
# on files that already checked in.
#
# You also need to make sure that the user running your webserver has
# permission to run the client SCM program (eg, cvs, p4, svn), and to
# connect to the repository.
@valid_repositories =
    (
     # Example CVSROOT of a CVS repository on the same machine as the
     # codestriker server.
     '/home/sits/cvs',

     # Example of a CVS repository which contains the URL to a viewcvs
     # installation (CVS web is also supported), followed by the
     # CVSROOT of the repository.
     'http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi /cvsroot',

     # The next example is the syntax used for specifying a Subversion
     # repository, which is simply the subversion repository URL
     # prefixed # by svn:
     'svn:http://svn.collab.net/repos/svn/trunk',

     # Subversion server with authentication.  The user name and
     # password should be added to the end and separated by
     # semicolons.
     'svn:http://svn.collab.net/repos/svn/trunk;username;password',
```

```
# Subversion server that uses the Subversion protocol.
'svn://my.subversion.server/repos/product/trunk',

# Example CVS pserver config with username and password
# specified.
':pserver:sits:password@cvs.sourceforge.net:/cvsroot',

# Example CVS pserver config with proxy options.
':pserver;proxy=abc.com;proxyport=8080:sits:pwd@cvs.dev.net',

# Example CVS pserver with empty password.
':pserver:anonymous:@cvs.sourceforge.net:/cvsroot',

# Example CVS server which will be connected to with SSH.  This
# assumes the appropriate ssh keys have been created so that the
# process running the Codestriker application can connect to the
# CVS server without requiring a password to be entered.
':ext:sits@localhost:/home/sits/cvs',

# Visual SourceSafe repository on same machine at default
# location.  Username "admin", password "password".
'vss:admin;password',

# Visual SourceSafe repository on same machine, but with specific
# repository location specified.
'vss:c:\\Program Files\\Microsoft Visual Studio\\VSS;admin;password',

# Visual SourceSafe repository located on a network fileshare.
'vss:\\\\VisualSourceSafeMachineName\\SharedRepositoryPath;admin;password',
# Example Win32 CVS repository on the same machine.
':local:c:\\cvsrep',

# Another Win32 CVS repository on the same machine.
'c:/cvsrep2',

# The next example is for a Perforce repository.  After the
# leading :perforce identifier, the next two components are the
# Perforce user and password parameters.  The last two parameters
# after the '@' symbol represent the host and port number of
# the Perforce server.
'perforce:sits:password@localhost:1666',

# The next example is a ClearCase repository, where the path is
# the location of a shared snapshot view.  From this view, it
# should be  possible to a file of any version can be
# retrieved from the vob using the "cleartool get" command.  It
# is important that this snapshot view is accessible with the
# same path specification for all developers.  This is because
# a diff file created by a developer will refer to the snapshot
# view, and will allow Codestriker to retrieve specific files
# and versions mentioned in the review text, when necessary.
# It is also important that the user account running the
# webserver process has permission to access to the snapshot
# view.
'clearcase:c:\\stuff\\view_name\\vob_name',

# The next example is a repository based off a ClearCase dynamic view.
```

```
     # The clearcase identifier is followed by the dyn indicator,
     # followed by the view name, followed by the location where the
     # view is loaded.
     # 'clearcase:dyn:viewname:/vobs'
    );
```

As explained by the comment in the codestriker.conf file, sometimes there is a benefit for displaying a name instead of a repository URL in the Codestriker UI.

```
# A mapping of repository URLs to names.  In any screen where a
# repository is displayed, if there is a mapping for the repository
# defined here, then the symbolic name will be displayed instead of
# its raw URL.  This is useful when the URL contains sensitive
# username/password information, or the symbolic name is more
# meaningful to the end-user.  If there is no mapping defined for a
# specific repository, its URL will be displayed.
$repository_name_map =
{
    '/home/sits/cvs' => 'Local CVS',
    ':pserver:sits:password@cvs.sourceforge.net:/cvsroot' => 'SF CVS'
};
```

During a review, it is sometimes beneficial to see the revision history of a file, such as information provided by CVSweb or ViewCVS. You can specify a mapping of repository names to URLs for this purpose, for example:

```
$file_viewer =
{
    '/home/sits/cvs' => 'http://localhost/cgi-bin/cvsweb.cgi'
};
```

This indicates that for any review made against the `/home/sits/cvs` repository, file revision information can be found using the URL specified.

If you are using CVS, make sure the `$cvs` setting is set to the correct path. For UNIX, this is likely to be something like:

```
# Location of the cvs binary.
$cvs = '/usr/bin/cvs';
```

For Windows, this could be something like:

```
# Location of the cvs binary.
$cvs = 'c:/gnu/bin/cvs';
```

Note the use of '/' (forward-slash) characters, rather than '\' (back-slash). Even under Windows, when setting filenames, you should always use forward slashes.

If you are using Subversion, make sure the `$svn` setting is set to the correct path.

If you are using Perforce, make sure the `$p4` setting is set to the correct path.

If you are using Visual SourceSafe, make sure the `$vss` setting is set to the correct path.

2.2.5. Bug-Tracking Integration

It is often useful to link the creation of code review topics with the associated bug records that the code is fixing. That way, it is possible to read a bug record, and apart from reading the textual description as to how it has been resolved, Codestriker can add in a link to the code review topic, which shows the actual code which fixed the bug (and any important decisions made in the Codestriker comments). Currently, there is support for Bugzilla, Flyspray and TestDirector, but it is not difficult to add in support for other systems.

If you don't use a bugtracker you can skip this section, as by default, there is no linking to a bug tracking system. An example configuration could be as follows:

```
# Bug tracking type.
$bug_db = 'bugzilla';

# Bug database connection details.
$bug_db_host = 'localhost';
$bug_db_name = 'bugs';
$bug_db_password = 'bugs_password';
$bug_db_dbname = 'bugs';

# Bugzilla codestriker user id.
$bug_db_user_id = '2';
```

The `$bug_db` setting indicates to use Bugzilla. If this value is set to '', then no linkage to a bug tracking system is performed (the default).

The `$bug_db_host` setting indicates the hostname that holds the bugzilla database, while `$bug_db_name` and `$bug_db_password` contain the database username and password to connect to the Bugzilla database. The `$bug_db_dbname` setting contains the Bugzilla database name, which by default is "bugs". You can verify these settings by using **mysql** to connect to the Bugzilla database interactively.

Codestriker adds "comments" to the appropriate bug record whenever a code review topic has been created against it, or the review's state has changed. To do this, a special Bugzilla user needs to be created which the comments will be created against. Create the user using the Bugzilla interface, and call it "codestriker@yourhost.yourdomain". Then connect to the Bugzilla database using **mysql**, and execute the following command to determine the userid of the user just created:

```
SELECT userid FROM profiles WHERE
login_name = 'codestriker@yourhost.yourdomain';
```

This value should be set into the `$bug_db_user_id` setting.

```
# Bugzilla codestriker user id.
$bug_db_user_id = '2';
```

2.2.6. LXR Integration

Codestriker has been integrated with LXR, so that when performing a review, those identifiers known to LXR will be automatically hyperlinked. This is an enormous aid to the code reviewing process, as you can quickly determine where a variable/function/method/class is defined and where it is used quickly, while studying the proposed changes.

The LXR home page is located at http://lxr.sourceforge.net. The integration has been tested with LXR version 0.9.2.

Its possible that each source code repository is associated with a different LXR deployment. The default configuration file shows an example:

```
$lxr_map =
{
    '/home/sits/cvs' => { db => 'DBI:Pg:dbname=lxr',
     user => 'lxr',
     password => '',
     url => 'http://localhost.localdomain/lxr/ident?i='
   },

     'svn:http://svn.collab.net/repos/svn/trunk' =>
                       { db => 'DBI:Pg:dbname=lxr2',
     user => 'lxr',
     password => '',
     url => 'http://localhost.localdomain/lxr2/ident?i='
   }

};
```

This is basically fancy Perl syntax for a map. In this instance, there are two keys, `/home/sits/cvs` and `'svn:http://svn.collab.net/repos/svn/trunk`, which represent the source control repositories specified above within the `@valid_repositories` setting.

Each key is mapped to an object containing four attributes. The `db` attribute is a DBI URL of the LXR database, `user` is the database username, `password` is the database password, and `url` is the URL of the LXR deployment for the identifier search page.

If you don't have LXR installed, you can simply set this variable as follows:

```
$lxr_map =
{
};
```

Also note, LXR at present doesn't seem to work with Taint checking. To avoid warning messages, you'll need to comment out the "PerlTaintCheck On" line in your Apache `httpd.conf` file, and remove the -T switch at the top of the `bin/codestriker.pl.base` file.

2.2.7. Topic Text Encoding

Codestriker stores the topic text, description and comments as UTF-8. When creating a topic, Codestriker needs to be told what encoding your files are stored in. By default, Codestriker assumes it is UTF-8 (compatible with

ASCII). If your source code files are stored in another encoding (for example, gb2312 for a Chinese team), this needs to be specified in the `$topic_text_encoding` variable.

```
# Character encoding to use when reading topic text.  Default is utf8
# (compatible with ASCII) if not set, but this can be over-ridden here.
# List of example encoding names can be retrieved from the following
# URL: http://perldoc.perl.org/Encode/Supported.html.
#$topic_text_encoding = 'utf8';
#$topic_text_encoding = 'gb2312';
```

2.2.8. Deployment Options

There are a number of other options which affect how Codestriker runs. The most important ones are shown below. Unless you have specific reasons to, most intranet deployments of Codestriker can leave these options as is.

```
# Exclude these file types from review topics.
# You will generally want to exclude any non-human-readable files.
@exclude_file_types = ('rtf', 'doc', 'gif', 'bmp', 'jpeg', 'jpg', 'mdb',
         'ppt', 'vsd', 'xls', 'zip', 'tgz', 'tar', 'gz',
         'opt', 'aps', 'ncb', 'a', 'so', 'dll', 'lib',
         'exe', 'png', 'pdf', 'bin', 'out', 'ld', 'fm',
         'indd', 'wav', 'o', 'obj', 'mpp', 'vsw', 'jfif',
         'tif', 'tiff', 'xbm', 'fnt', 'ttf', 'pfm', 'pfb',
         'eps', 'wpj', 'sxi');

# Indicate if topics can be listed/searched.  Turning this to false can be
# useful for "anonymous" installations of Codestriker.
$allow_searchlist = 1;

# Indicate if the repository attribute can be set to a topic.  If this
# is disabled, it won't be possible to view the entire contents of a
# file before the proposed change and/or after.  On some servers (such
# as sourceforge), the firewall doesn't allow CGI scripts to make
# remote connections.
$allow_repositories = 1;

# The following controls project configuration.  Each Codestriker topic is
# a member of a specific project.  Uncomment the option you want
# below.  Note the textual state names below cannot be changed.

# Default option, projects are enabled, but they have no state
# changing operations (ie, projects are always in state 'Open').
@project_states = ('Open');

# Don't use projects at all.  Effectively, an implicit "default
# project" is created and associated with all topics behind the scenes.
# @project_states = ();
#
# Allow for projects to be closed.  Closing a project will
# not allow new topics to be created in that project.
# @project_states = ('Open', 'Closed');
#
```

```
# Allow for projects to be deleted.  This is potentially a dangerous
# option to allow, as deleting a project will delete all of its member
# topics as well.  Use with caution.
# @project_states = ('Open', 'Deleted');
#
# Allow for projects to be closed and deleted.  Use with caution.
# @project_states = ('Open', 'Closed', 'Deleted');

# If true, don't display any email addresses in their true form, but
# truncate them, to beat SPAM harvesters.
$antispam_email = 0;
```

2.2.9. Topic Length Restrictions

As explained by the comments in the configuration file, it is possible to limit the size of code review topics that will be accepted by the system:

```
# The number of problems found per line drops if the size of the
# topic is too large. A common inspection pitfall is for authors to
# attempt to review too much material and then miss problems.
# These two options allow the Codestriker administrator to limit
# the length of the topics. Topics that have more lines than
# $maximum_topic_size_lines are rejected when they are created.
# Topics that are larger than $suggested_topic_size_lines generate
# a warning displayed in the topic page, but are accepted into the
# system. Codestriker measures that length of the topic by counting
# the number of lines in the topic text.
#
# The Codestriker default of not enforcing any limits is specified by
# settings either option to an empty string. If you are not sure
# what a reasonable limit would be, start with a suggested_topic_size_lines
# set to 350, and adjust with experience.
$maximum_topic_size_lines = '';
$suggested_topic_size_lines = '';
```

2.2.10. Comment Email Configuration

By default, whenever a comment it submitted, an email will be sent to the author of the comment, the author of the review, and anyone else who has submitted a comment on the line of code in question. This may not be appropriate for some team processes, and can be changed by setting `$allow_comment_email` to 0.

```
# If true, Codestriker will send out emails to the topic owner and
# comment submitter when a comment is added. If this option is false,
# no email will be sent to either the topic owner or the comment
# submitter. Emails about each comment may not be needed if a meeting
# is planned to discuss the topic. If the comment submitter specifies
# a cc user, an email is always sent out, regardless of this setting.
$allow_comment_email = 1;
```

2.2.11. Source Code Hihglighting

Source code highlighting will be performed if the Highlight program is installed.

```
# Location of the highlight binary, which is used for highlighting source code.
# Available from http://www.andre-simon.de/.  If this is not set, no syntax
# highlighting will be performed.
$highlighter = ”;
#$highlighter = ’/usr/bin/highlight’;
#$highlighter = ’C:/Program Files (x86)/WinHighlight/highlight.exe’;
```

2.2.12. Default View Topic File View Mode

As explained by the comments in the configuration file, it it possible to specify by default, whether topics display the deltas for all files in the review, or just a single file at a time by default. The viewing mode can be changed dynamically on the view topic screen.

```
# When displaying a topic, if this value is -1, then all files in the
# topic are displayed in the one page (default old Codestriker
# behaviour).  If the value is 0, then only the first file is shown,
# with links to display the other files.  This is useful for those
# deployments that review a large amount of code.
$default_file_to_view = -1;
```

2.2.13. Comment Thread Metrics

As explained by the comments in the configuration file, it is possible to defined a number of metrics associated with each comment thread (issue) created in the review.

```
# Each comment thread (or issue) that is created against a specific
# code line in Codestriker can have a configurable number of
# user-defined metrics recorded against it.
#
# Every site has their own requirements, below are a number of example
# configurations.  The "name" attribute refers to the name of the
# metric being recorded.  The "values" attribute is a list of values
# that this metric can be assigned to.  The "default_value" attribute
# is optional, and indicates what the default value of the metric is
# assigned to.  If this attribute is not specified, then the user will
# be required to specify a value for a metric when creating a new
# comment thread.  This is recommended, so that users think about what
# these values should be, rather than blindly accepting default
# values.  For the "Status" metric below however, it is recording the
```

```
# "state" of the thread, so an initial state of "Submitted" is reasonable.
# For the other metrics below, an initial value makes no sense.
# Metric items can have an optional show_on_main_page list that will
# force the numbers of comments with the metric settings to be reported
# on the main page of codestriker.
$comment_state_metrics =
    [
      { name             => 'Status',
 values           => ['Submitted', 'Invalid', 'Completed'],
 default_value    => 'Submitted',
 show_on_mainpage => ['Submitted' ]
      }
    ];

# Two metrics defined: Status and Type.
#$comment_state_metrics =
#    [
#      { name=>'Status', values=>['Submitted', 'Invalid', 'Completed'],
#                default_value=>'Submitted' },
#      { name=>'Type', values=>['Style', 'Minor', 'Major', 'Severe'] }
#    ];
#
# Four metrics defined: Status, Level, Mode and Type.
#$comment_state_metrics =
#    [
#      { name=>'Status', values=>['Submitted', 'Invalid', 'Completed'],
#                default_value=>'Submitted' },
#      { name=>'Level', values=>['Major', 'Minor'] },
#      { name=>'Mode', values=>['Missing', 'Wrong', 'Unclear', 'Suggestion'] },
#      { name=>'Type', values=>['Logic', 'Data Handling', 'Interface',
#          'Error Handling', 'Performance', 'Comments',
#          'Standards'] }
#    ];
#
# Case where no comment thread metrics are to be used.
#$comment_state_metrics = [];
```

2.2.14. Metrics Support

As explained by the comments in the configuration file, it is possible to maintain software metrics obtained from the code reviewing process. There is also scope for customising Codestriker to track your own software metrics.

```
# This options configures the metric support in codestriker. You have
# the following options:
#
# $metric_config = "none", "basic", "all", "metric name, metric name, etc"
#
# "none" - turns off all extra metric support in the application. The
# metric page will only display and manage data that is strictly
# required to perform the review.  Codestriker will not require any
# addition data input from the reviewers and authors.  This is the
# default. However, you still get basic data like how many topics are
# being created and how problems are being found.
```

```
#
# "basic" - Turns on the metrics that are considered to be essential
# for a metric program. It will require that reviewers and authors
# enter the time spent reviewing the topic, the time spent in the
# review meeting, and the time spent preparing for the review. The
# metric selection assumes that you are following a formal review
# process with a preparation meeting, and a defect review meeting.
#
#   kickoff time - time spent preparing for the review
#   checking time - time spent actually reviewing the topic.
#   logging meeting duration - the time spent in the logging meeting.
#
# "all" - Turns on all of the metrics that one could possibly want to
# track. The list of metrics is from the book "Software Inspection" by
# Gilb and Graham.  You should probably not use this unless you are
# using a formal process that is well established. You may want to
# enable this temporally to get a idea of the types of metrics that
# are supported.
#
# "name,name" - Lastly, you can pick and chose what metrics you would
# like to enable.  just list the metric names in a comma separated
# list. You can see all of the build in metrics in the
# lib/Codestriker.pm file. For example, if you don't hold a kick off
# meeting, and but do hold a logging meeting, the basic option will not
# quit fit. You should set the $metric_config as:
# $metric_config = "checking time,logging meeting duration".
#
# If you don't like our choices of metrics, the names, descriptions,
# etc feel free to edit the lib/Codestriker.pm. It contains
# documentations on how to add your own metrics into codestriker. It
# is easy to do, and does not require any coding.

$metric_config = "none";
```

2.2.15. RSS Support

If you install the `XML::RSS` module, and re-run `install.pl`, Codestriker will display an RSS link on the topic list page, which can be used as a URL into your RSS reader, to keep track of new topics being added to the system.

2.2.16. Scmbug Integration

It is possible for Codestriker to integrate with ScmBug (http://www.mkgnu.net/?q=scmbug). This allows users to generate a topic based on the changes done under a given bug ID (or list of bug IDs). An example configuration is:

```
$scmbug_hostname = 'localhost';
$scmbug_port = 3872;
$scmbug_lib_dir = 'C:/Program Files/Scmbug/share/scmbug/lib';
```

This would match the default settings used by Scmbug on Windows. Where `$scmbug_hostname` and `$scmbug_port` are the host and port of the machine where Scmbug is running. The `$scmbug_lib_dir` points to the lib directory under the Scmbug installation. If Scmbug is running on a separate machine a copy of the Scmbug lib directory needs to be staged on the same machine as codestriker and the `$scmbug_lib_dir` variable made to point at this.

2.3. Running install.pl

The `install.pl` script located in the `bin` directory *must* be run before Codestriker can be used, or after each upgrade. There is no harm in running it as many times as you like. The purpose of the script is to ensure all of the necessary pieces on your system are installed, and tells you how to install anything that is missing. The other purpose of the script is to create the Codestriker database, and perform any necessary data migrations.

Perl is available by default for all UNIX systems. For Windows, you can download it for free from http://www.activestate.com/Products/Download/Download.plex?id=ActivePerl. Note if you are using IIS as your web-server, please see Section 2.5 on appropriate ActivePerl releases.

To run `install.pl` under UNIX, do the following:

```
% cd /codestriker/install/location/bin
% ./install.pl
```

For Windows, you need to do something like the following from the command prompt:

```
% cd c:\codestriker\install\location\bin
% install.pl
```

This will tell you what missing Perl modules you have, and how to install them. It will then attempt to try and initialise the Codestriker database. Make sure your Codestriker database has been created and the database system is running, before running this script.

2.4. Apache webserver configuration

This section deals with deploying Codestriker using Apache, on either a UN*X machine or Win32. Codestriker can run as an ordinary CGI script, which allows it to run under any CGI-complaint webserver. The following configuration details are specific for the Apache webserver (http://httpd.apache.org), which is available for download for both UNIX and Window platforms. Apache should be already available for most UNIX distributions. Apache 1.X or 2.X can be used for Win32 systems deploying Codestriker as a CGI script. For Win32 mod_perl installations, Apache 1.X is recommended.

Note any Codestriker or Apache configuration changes require the Apache server to be restarted. For UNIX, a command like the following is required:

```
/etc/init.d/httpd restart
```

or

```
/etc/init.d/apache restart
```

For Windows, the Apache shell window can be terminated by pressing ^C, and started from the Windows menu. There are also shortcuts from the menu for editing the Apache configuration file.

2.4.1. CGI Script

If you installed Codestriker into `/var/www/codestriker/codestriker-X.Y.Z`, the configuration you need in your `apache.conf` file (normally located in either `/etc/httpd.conf`, `/etc/apache.conf` or `/etc/httpd/conf/httpd.conf`) is the following:

```
ScriptAlias /codestriker/  /var/www/codestriker/codestriker-X.Y.Z/cgi-bin/
Alias /codestrikerhtml/  /var/www/codestriker/codestriker-X.Y.Z/html/

<Directory "/var/www/codestriker/codestriker-X.Y.Z/cgi-bin/">
   AllowOverride None
   Options ExecCGI
   Order allow,deny
   Allow from all
   SetHandler cgi-script
</Directory>

<Directory "/var/www/codestriker/codestriker-X.Y.Z/html/">
   AllowOverride None
   Allow from all
</Directory>
```

For Windows, the configuration is the same, but filename paths should use '/' rather than '\'. An example configuration could be the following, if Codestriker was installed in `c:\codestriker\codestriker-X.Y.Z`.

```
ScriptAlias /codestriker/ "C:/codestriker/codestriker-X.Y.Z/cgi-bin/"
Alias /codestrikerhtml/ "C:/codestriker/codestriker-X.Y.Z/html/"

<Directory "C:/codestriker/codestriker-X.Y.Z/cgi-bin/">
   AllowOverride None
   Options ExecCGI
   Order allow,deny
   Allow from all
   SetHandler cgi-script
</Directory>

<Directory "C:/codestriker/codestriker-X.Y.Z/html/">
   AllowOverride None
   Allow from all
</Directory>
```

2.4.2. Apache 1.X mod_perl

Using **mod_perl** provides performance benefits for Perl-based web applications. For CGI deployments, as described in the previous section, each HTTP request will create a *new* Perl interpreter, which needs to initialise, and then parse the Codestriker source code. This may add significant latency for each HTTP request.

Using **mod_perl**, a pool of Perl interpreters which have already parsed the Codestriker source code is maintained, so that whenever an HTTP request is issued, the time spent creating a new Perl interpreter and the parsing of the Codestriker code is removed.

Mod_perl is available for download from http://perl.apache.org. For most UNIX distributions, it is available by default with Apache. For installing **mod_perl** under Windows, http://www.webmatrix.net/log/modperl-win32 contains installation information. You should be able to install it by typing:

```
C:\> ppm
PPM> rep add theory http://theoryx5.uwinnipeg.ca/cgi-bin/ppmserver?urn:/PPMServer58
PPM> install mod_perl
```

Note mod_perl has known to be a little flaky under Windows. Make sure you get Codestriker working deployed as a CGI script before trying to use mod_perl.

The following shows the configuration settings for an Apache 1.X server with **mod_perl** enabled, for a Codestriker distribution installed in `/var/www/codestriker/codestriker-X.Y.Z`.

```
Alias /codestriker/  /var/www/codestriker/codestriker-X.Y.Z/cgi-bin/
Alias /codestrikerhtml/  /var/www/codestriker/codestriker-X.Y.Z/html/

<Directory "/var/www/codestriker/codestriker-X.Y.Z/cgi-bin/">
   SetHandler perl-script
   PerlHandler Apache::Registry
   Options +ExecCGI
</Directory>

<Directory "/var/www/codestriker/codestriker-X.Y.Z/html/">
   AllowOverride None
   Allow from all
</Directory>
```

The settings for Windows are the same, only the pathnames will be different, as per the CGI configuration in the previous section.

For extra security, Codestriker supports Perl taint-mode, so it is advisable to also have the following option in your Apache config:

```
PerlTaintCheck On
```

Note if you are using LXR on the same webserver, this option cannot be used. You'll also need to remove the `-T` argument from `bin/codestriker.pl.base`, and re-run **install.pl** again (see Section 2.3) if you want to use Codestriker and LXR.

Also note there is a strange issue with Perl 5.8 on Win32, the open3() call and taint mode. For Win32 users, don't enable tainted mode.

2.4.3. Apache 2.X mod_perl

For Apache 2.X, make sure the **mod_perl** module is loaded when Apache starts. Near the top of the Apache config file, you should see commands like the following:

```
LoadModule perl_module modules/mod_perl.so
PerlModule Apache2
```

If these commands aren't present, and a Perl startup file is not being used, make sure these are added in.

The Codestriker configuration for Apache 2.X is very similar, the only change is the name of the PerlHandler.

```
Alias /codestriker/  /var/www/codestriker/codestriker-X.Y.Z/cgi-bin/
Alias /codestrikerhtml/  /var/www/codestriker/codestriker-X.Y.Z/html/
```

```
<Directory "/var/www/codestriker/codestriker-X.Y.Z/cgi-bin/">
   SetHandler perl-script
   PerlHandler ModPerl::Registry
   Options +ExecCGI
</Directory>

<Directory "/var/www/codestriker/codestriker-X.Y.Z/html/">
   AllowOverride None
   Allow from all
</Directory>
```

To enable Perl taint mode checking, using the following option:

```
PerlSwitches -T
```

As mentioned in the previous section, Win32 users should not enable this mode.

2.5. IIS configuration

This section deals with deploying Codestriker under IIS, as a CGI script. *At the time of writing, the current version of ActivePerl (5.8.8.817) contains a version of CGI.pm which is broken for IIS deployments. Use http://downloads.activestate.com/ActivePerl/Windows/5.8/ActivePerl-5.8.7.813-MSWin32-x86-148120.msi instead.* These instructions were performed on a Windows 2000 machine, so hopefully this provides an indication as to what needs to be performed for other Win32 configurations. Startup the IIS configuration application by going to "Control Panel" -> "Administrative Tools" -> "Internet Services Manager". Like the Apache configuration, the webserver needs to be told where to find the Codestriker CGI script, and where to find the associated CSS and html help files.

Assuming you are deploying Codestriker under the "Default Web site", right-click this entry on the left hand frame of the window, and select "New" -> "Virtual Directory" -> "Next". Enter in `codestriker` into the Alias textfield, then click "Next". Then enter the `cgi-bin` directory of your unpacked Codestriker distribution into the "Directory" textfield, then click "Next". Make sure the "Execute" access permission checkbox is selected, then click "Next", click then "Finished".

It is important that the `codestriker` virtual directory is setup to be able to execute Perl scripts. Right-click the `codestriker` virtual directory and select "Properties". Click on the "Configuration..." button. Check that there is an entry for `.pl` files, and that the "Executable Path" entry looks like (substitute the path to your local Perl installation):

```
C:\Perl\bin\perl.exe "%s" %s
```

If there is no mapping for `.pl` files (which would be odd, since the ActiveState install does this for you automatically), add a new entry in with the above setting, limited to "GET,HEAD,POST". Make sure you enter the quotes, as shown above.

Follow a similar process for the `codestrikerhtml` directory. Right-click "Default Web site" and select "New" -> "Virtual Directory" -> "Next". Enter in `codestrikerhtml` into the Alias textfield, then click "Next". Then enter the `html` directory of your unpacked Codestriker distribution into the "Directory" textfield, then click

"Next". Make sure the "Read" and "Browse" access permission checkboxes are selected, then "Next", then "Finished".

Ensure the security for the two virtual directories `codestriker` and `codestrikerhtml` are appropriate for your site. Right-click on each directory, and select "Properties", then the "Directory Security" tab. Click the "Edit..." button and make the appropriate changes for your deployment.

Right-click "Default Web site" and select "Start" if the website is not currently running, and test it.

Further information on setting up IIS with Perl CGI can be found from http://support.microsoft.com/kb/q245225.

2.6. Upgrading Codestriker

To upgrade, extract the distribution into another directory, check the `CHANGELOG` file to see if any configuration changes are required, since the last version you used, copy/modify your existing `codestriker.conf` file, and *most importantly, run* `install.pl`. There is no harm in running this script as many times as you like.

Chapter 3. User’s Guide

3.1. Introduction

The goal of this chapter is to present how to create/manage Codestriker projects, how to create code review topics, and then how to review these topics. Assuming Codestriker has been configured and is running, you can get started by going to the following URL: `http://yourhost.yourdomain/codestriker/codestriker.pl`. This should take you to the topic list screen, which is talked about in the following section. From a fresh install, there will obviously be no code review topics in the list. However there is a link on this page to "Administer Projects", which will allow you to create some new projects to match your development process. Once you have defined all of your initial projects, you can create a new code review topic by clicking on "Create new topic". These screens are covered in the following sections.

3.2. Topic List Screen

The topic list screen is used for displaying a list of code review topics, which contain the topic’s title, author, reviewers, cc, creation date, links to the bugs it fixes, and its state. The email address for the author, reviewers and cc fields are shortened for brevity reasons. The links from the bug ids will take you to the bug tracking system (currently, only Bugzilla is supported) for more information on that specific bug. A topic can be in one of the following states:

Open

: This is the initial state of the topic, and indicates it is still open for reviewing.

Closed

: Closed indicates that the review is no longer relevant, or invalid. A reviewer might decide that a review posted is fixing a problem in the wrong way, and after adding a comment, closes the review off.

Committed

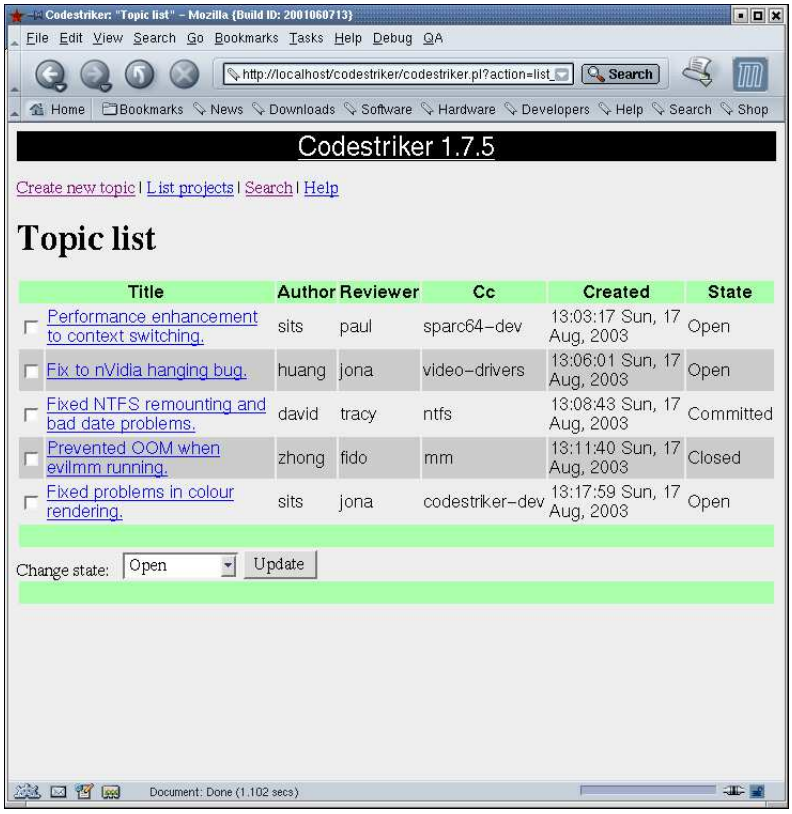
: Committed indicates that all the comments posted to the review have been addressed, and that the source code has been committed into the SCM system.

Obsoleted

: Obsoleted indicates that the topic has been superseeded by another topic.

An example screen can be seen in Figure 3-1.

Figure 3-1. Topic List Screenshot



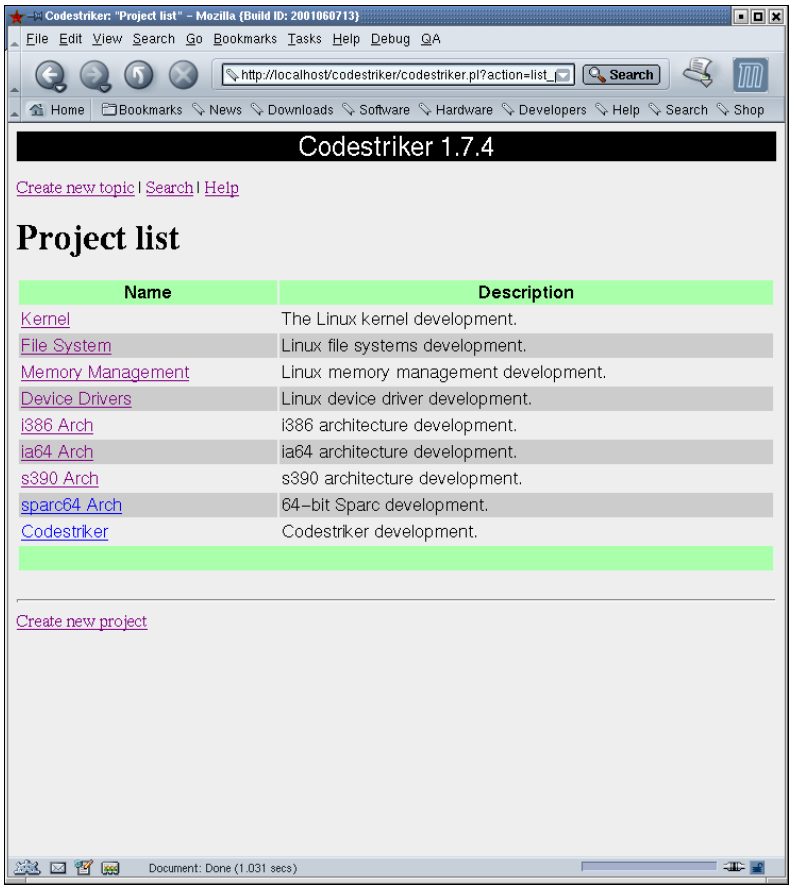
To view a specific topic, click on its title. This will take you to the view topic page, which is covered in Section 3.5.1. From the "topic list" screen, it is possible to change the state of many topics in one action, by selected the checkboxes next to the topics of interest, and pressing the "Update" button with the state dropdown selected to the desired state. If topic deletion functionality has been enabled, topics can be deleted by clicking on the individual topic, and then deleting it. Generally speaking, topics are only deleted if the wrong content was posted by mistake. It is useful to maintain all historical topics in the Codestriker database, as they can be searched over in the future, which is covered in Section 3.9.

3.3. Creating a new Project

When a new code review topic is created, it is necessary to specify what project the review is related to. Projects can be used as a filtering mechanism when displaying lists of topics (see Section 3.9). This can be a great benefit, in the situation where a single Codestriker deployment is serving a large number of developer groups. Typically, a member of a developer group only wants to see the list of open code review topics in their project.

An example project list screen can be seen in Figure 3-2.

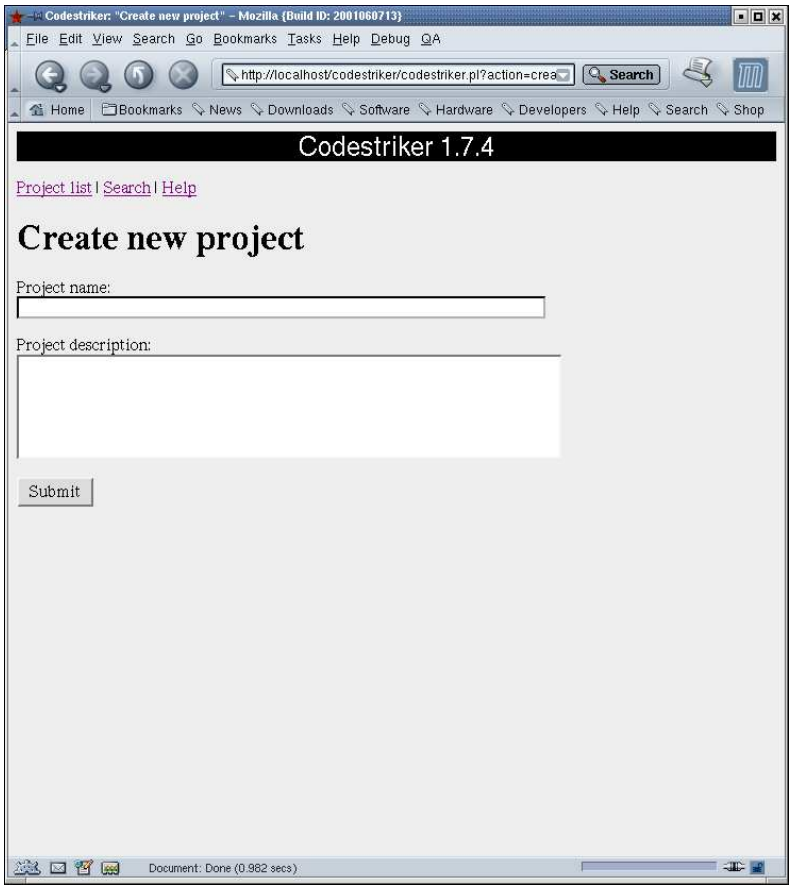
Figure 3-2. Project List Screenshot



This figure shows an example Codestriker deployment, which shows the list of projects defined, where each row contains the project's name and description. To edit a project's name and/or description, click on the project's name. This will take you to a screen for modifying the project's attributes.

At the bottom of the "project list" page, is a link to "Create new project". This will take you a screen, as shown in Figure 3-3.

Figure 3-3. Create Project Screenshot



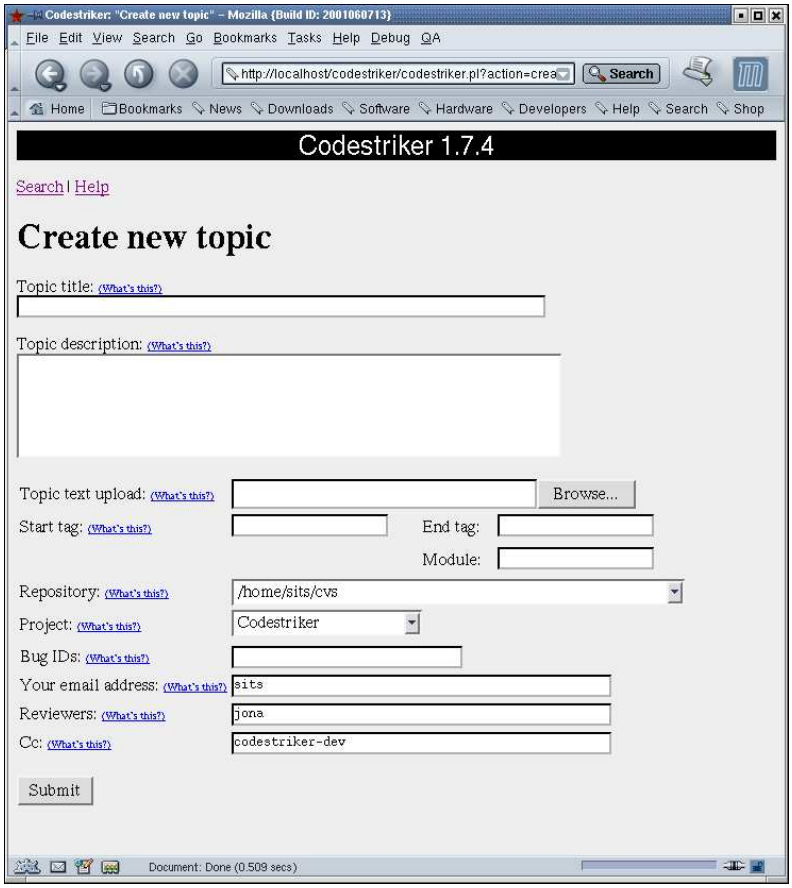
To create the project, simple fill in the details for the project's name and description, and press the "Submit" button. This will then display an updated project list.

3.4. Creating a new Topic

The "Create topic" screen can be accessed from a number of other screens, and requires the submission of a number of details in order to create a topic. A screenshot can be seen in Figure 3-4. It is also possible to create a new topic which obsoletes a number of other topics via the "Obsolete Topics" button on the topic list screen (see Section 3.2). This is normally done when a review has many comments, and a new review (which obsoletes the old review) needs to be created which has addressed these comments. It is also possible to generate new topics from the command line. Check out the `SubmitCodeReview.pl` script in the `bin` directory in the Codestriker

distribution.

Figure 3-4. Create Topic Screenshot



The topic title is a mandatory field which contains a one-line summary of the topic. This title is used when the topic is displayed in the "Topic list" screen (see Section 3.2). The topic description contains a more detailed description of the review itself, and should be enough for a reviewer to understand the purpose of the review. If Codestriker is linked to a bug tracking system, further down in the form, the "Bug IDs" field gives the opportunity to enter in multiple bug ids (either space or comma separated), which will also give the reviewer an opportunity to see what bugs this review fixes.

The "Project" dropdown list should be selected onto the project this review belongs to. The last three fields contain comma-separated email addresses. The "Your email address" field contains the email address which will receive review comments. This is typically, the author's email address. The "Reviewers" field contains the list of reviewer email address which will be notified once the review is created. The "Cc" field is typically used for notifying a group email list that a new review has been created. These field values are stored into your browser's cookie, so that the next time a review is created, these fields will be pre-filled, as they don't typically change much between different reviews.

The most complex part of this form are the values required for the "Topic text upload", "Start tag", "End tag", "Module" and "Repository" fields. These fields dictate what the actual review text is. The first step is to select what source control management system the review was made against, and updating the "Repository" field accordingly. The following sub-sections detail the different review inputs that can be handled. Once all the details have been entered, and the "Submit" button is pressed, a confirmation screen will come up indicating that

the review has been created. An email, with a URL to view the topic will be sent to the author, reviewers and any email addresses entered in the "Cc" field.

3.4.1. Creating CVS Diff Topics

For CVS projects, a common situation is you have a checkout of your project's module, and you have fixed a bug which needs to be reviewed. To get this reviewed in Codestriker, you need to generate the review text, by going to the top-level directory of your checkout, and issuing a command like:

```
% cvs diff -uN > /tmp/review.txt
```

This command creates a diff file, which consists of the code changes you have made since you checked out the project from CVS. The `-u` option indicates to create a "unidiff" formatted file, which is what Codestriker can parse. The `-N` option means to show added and removed files. Depending on your development process, you may want to issue different cvs diff commands with different flags, or even using "cvs rdiff" instead. Codestriker can accept any input, as long as `-u` is specified, to ensure the diff is in unidiff format. Note for reviews which include C/C++ code, you can also include the `--show-function-line` and/or `--show-c-function` options, which will indicate for each diff delta in the display, what function is being modified.

This filename can either be entered directly into the "Topic text upload" field, or selected from a file dialog which is created when the "Browse" button is pressed. The "Start tag", "End tag" and "Module" fields can be left blank. Once all the other fields have been entered, clicking the "Submit" button will create the review, and show a confirmation screen.

In some development processes, CVS tags are used to mark units of work that have been applied to the repository. Rather than generating the review text via a CVS command, as specified in the previously, the review text can be specified by entering in the "Start tag", "End tag" and "Module" name into their respective fields. Codestriker will then fetch the review text from the nominated repository automatically, by executing the appropriate CVS command itself.

Alternatively, Codestriker can also accept CVS diffs of already committed code from commands such as:

```
% cvs rdiff -uN -r START_TAG -r END_TAG MODULE
```

Where START_TAG, END_TAG and MODULE are substituted with appropriate values. Note this is no different to what Codestriker does when values are entered into the "Start tag", "End Tag" and "Module" fields when creating a topic.

If you want to review code in its full form as it exists in the repository for a specific tag value, you only need to specify this tag value in either the START_TAG *or* END_TAG field, and an appropriate value in the MODULE field. Codestriker will fetch all of the source files for this module and tag into the topic, and treat all of the files as "new" (ie, no diffs). This is no different to executing the following command to generate a diff:

```
% cvs rdiff -uN -r 1.0 -r TAG MODULE
```

3.4.2. Creating CVS Diff Topics Automatically

Depending on your development process, it may be more convenient for CVS to automatically create a Codestriker topic *automatically* whenever a commit occurs. The benefit with this approach is developers don't

need to explicitly create code reviews. A "code reviewer" can then monitor all commits to the repository, and has the opportunity to send comments to the authors if they spot any issues that need addressing.

The standard `commit_prep.pl` and `log_accum.pl` scripts from CVS have been modified so that the email which is sent after each commit which contains the diff of the commit, also includes the URL of the associated Codestriker topic that was created. *Unfortunately, these scripts can only be executed on the UNIX platform.*

The `commit_prep.pl` and `log_accum.pl` files are located in the Codestriker `bin` directory, and need to be copied into your `CVSROOT` directory, with a line like the following in your `commitinfo` file (read the comment in this file to set this correctly for your site):

```
DEFAULT $CVSROOT/CVSROOT/commit_prep.pl -r
```

and a line like the following in your `loginfo` file (read the comment in this file to set this correctly for your site):

```
DEFAULT $CVSROOT/CVSROOT/log_accum.pl %s
```

The `log_accum.pl` file needs to have some configuration variables set at the start of the script to reflect your site. You will also need to make sure you create a `$CVSROOT/CVSROOT/commitlogs` directory with the appropriate user-writable permissions, as commit information is written here as a result of these scripts running.

Note the first line of the log message will be used as the topic title. The entire log message is used as the topic description. Any strings of the form "Bug nnn" in the log message will be taken as a reference to a bug id, and will be used in the topic's bugid field.

3.4.3. Creating Diff Topics

In some deployments, there may not even be a source control management system, and it might be the case that only diffs are reviewed. An example command to generate the review text could be something like:

```
% diff -urN ../old-version/ . > /tmp/review.txt
```

Like the CVS command shown above, the `-u` and `-N` options are specified to output a unidiff file, which contains new and old files. The `-r` option indicates to recursively check the source and target directories specified (in this case `../old-version` and `.` to find differences. There are likely to be other options that will be specified, run **man diff** for more options. Note for reviews which include C/C++ code, you can also include the `--show-function-line` and/or `--show-c-function` options, which will indicate for each diff delta in the display, what function is being modified.

The "Start tag", "End tag" and "Module" fields have no relevance in this situation.

3.4.4. Creating Plain Text Topics

It is also possible to use Codestriker to simply review text that is not in the form as a diff. In this situation, Codestriker will simply treat the text as a single new file. Reviewers will still be able to view the text and make comments on a per-line basis as before. Simply put the filename to be reviewed in the "Topic text upload" field. The "Start tag", "End tag" and "Module" fields have no relevance in this situation.

3.4.5. Creating Subversion Diff Topics

Similar to CVS, Subversion diffs are created by again, going to the top of your project directory, and issuing the following command:

```
% svn diff -N > /tmp/review.txt
```

This file should then be selected for the "Topic text upload" field. The "Start tag", "End tag" and "Module" fields have no relevance in this situation. Because of an issue in Subversion, running the **svn diff** command *outside* of the repository root will not allow Codestriker to download and review the full file (although the patch segments can still be reviewed). If you commit your code first, then let Codestriker make the diff, as explained in the next section, then you can avoid this limitation. If you really don't want to commit your code, you will need to modify the diff file to include the paths on the Index: +++, and --- lines generated by **svn diff**.

In some development processes, branches are used as the logical unit of work that should be reviewed. Rather than generating the review text via the **svn diff** command, as specified in the previous section, the review text can be created by setting the appropriate values into the "Start tag", "End tag" and "Module" fields. Codestriker will then fetch the review text from the nominated repository automatically, by executing the following command:

```
% svn diff --non-interactive -r START_TAG:END_TAG --old repository_url MODULE
```

In Subversion the tags can be a repository revision number or HEAD, The HEAD shortcut indicates the latest revision number in the repository. If you would like to review a specific single check in, for example, check in 544, then you would use 543 for START_TAG and 544 for END_TAG.

If you would like to review a branch before merging then enter the start tag as the version number of the branch creation. The end tag should be HEAD, and the module path should be the path into the branch directory relative to the repository location. All topics created by Codestriker have a builtin workaround for the Subversion diff problem.

3.4.6. Creating Perforce Diff Topics

Similar to CVS, Perforce diffs are created by going to the top of your checkout (or client view in Perforce terminology) and issuing a command like:

```
% p4 diff -du > /tmp/review.txt
```

This file should then be selected for the "Topic text upload" field. The "Start tag", "End tag" and "Module" fields have no relevance in this situation.

Alternatively, for code which has already been committed into the repository (or depot in Perforce terminology), a command like the following can be used to generate the topic text:

```
% p4 describe -du 132 > /tmp/review.txt
```

Here 132 is the change-number. This should be substituted with the change-number that you want to review.

It is also possible to create Perforce topics from already committed code. Simply enter the change number into either the "Start tag" or "End tag" field, and set "Module" to be //, which corresponds to the root of the depot.

3.4.7. Creating ClearCase Diff Topics

In its very basic form, a ClearCase diff file can be created for a single file using the following command:

```
% cleartool diff -serial_format -predecessor filename > /tmp/review.txt
```

This file can then be uploaded into Codestriker for reviewing. Of course, it is far more useful to be able to review a group of files at once. In ClearCase, to determine which files are a part of a view, you can use the following command from the top level of your checkout:

```
% cleartool lsco -cview -short -recurse
```

Combine the two commands together to generate a single file with all modified files can be achieved doing something like:

```
% cleartool lsco -cview -short -recurse | \
  xargs -n1 cleartool diff -serial_format -predecessor > /tmp/review.txt
```

Note if you are on a Win32 platform, and you have cygwin installed, you may need to add an intermediate sed command so that cleartool doesn't complain:

```
% cleartool lsco -cview -short -recurse | sed 's/\\/\//g' | \
  xargs -n1 cleartool diff -serial_format -predecessor > /tmp/review.txt
```

In all of the above commands, if a ClearCase repository has been specified in the Codestriker configuration, it is important that the above commands are run from the *top-level directory* of the view, so that the full vob pathname information is available within the diff files. This allows Codestriker to retrieve the full contents of files from the vob, so that a reviewer can see the changes applied to an entire file, rather than a small segment.

Note the above commands can be modified if there is only interest in a specific sub-directory in your view, for example, the **cleartool lsco** can be amended with the sub-directory of interest as follows:

```
% cleartool lsco -cview -short -recurse pathname/to/interested/dir | \
  xargs -n1 cleartool diff -serial_format -predecessor > /tmp/review.txt
```

3.4.8. Creating Visual SourceSafe Topics

Similar to CVS, SourceSafe diffs are creating by going to the top of your project, and issuing a command like the following in a cmd.exe window:

```
C:\Project\> ssdiff.pl > topic.txt
```

The `topic.txt` file will contain the source-code differences made with the current code set compared with the SourceSafe repository. This file can then be selected for the "Topic text upload" field in the create topic screen. The "Start tag", "End tag" and "Module" fields have no relevance in this situation.

The `ssdiff.pl` Perl script is a custom script that is contained in the Codestriker `bin` directory, which needs to be installed on client machines that need to run the above command. The configuration section at the start of the script should be updated appropriately before it is used.

Alternatively, for code which has already been committed to SourceSafe, it is possible to explicitly specify on the create topic screen, values for the "Start tag", "End tag" and "Module" fields, which will enable Codestriker to fetch the topic text directly from SourceSafe. In this instance, any SourceSafe label names can be used in the tag fields. If only one tag field is specified, the complete contents of files corresponding to the specified label will be retrieved. The "Module" field contains a path into the SourceSafe repository which indicates the files of interest. An example value could be `$/Project/Gui`, which would work on all files located within that path.

3.4.9. Creating Topics from Bug IDs

If you have Codestriker configured to make use of Scmbug integration it is possible to generate a Topic by entering a list of bug IDs (comma separated). To generate the topic in this way, ensure that the Start Tag, End Tag and Module fields are left blank and that the relevant bug IDs are entered into the Bug IDs field.

3.5. Reviewing Topics

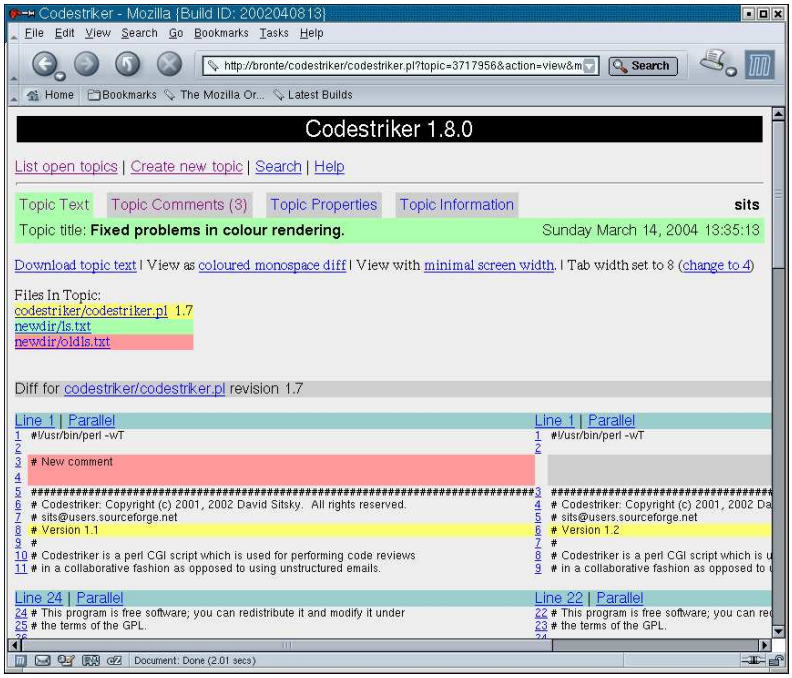
Reviewing a topic is achieved by going to the "View Topic" screen for a specific review topic. This can be accessed from either the "Topic List" screen (Section 3.2), or from a URL in a new topic notification email, which is sent to the author and reviewers.

3.5.1. Viewing a Topic

The "View Topic" screen is one of four tabs available for viewing aspects of a topic. An example "Topic Text"

tab display can be seen in Figure 3-5.

Figure 3-5. View Topic Screenshot



The top bar contains the following links to other Codestriker pages:

List open topics

Following this link will take you to the "Topic List" page (see Section 3.2), and will display all open topics in the system.

Create new topic

Following this link will take you to the "Create Topic" page (see Section 3.4).

Search

Following this link will take you to the "Topic Search" page (see Section 3.9).

Help

Following this link will show this help text.

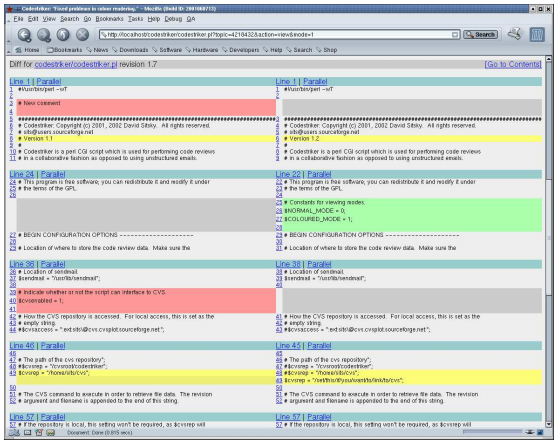
The next part of the screen shows the four available tabs for viewing different aspects of a topic. The "Topic Comments" tab is for viewing the comments that have been submitted against the topic. Clicking on this tab will take you to the "Topic Comments" page (see Section 3.6). The "Topic Properties" tab takes you to a page where you can view and edit the topic's attributes, such as the title, author, reviewer, the repository the code in the review has been made against, the project the review is associated with, and the topic description. Clicking on this tab will take you to the "Topic Properties" page (see Section 3.7). Finally the "Topic Information" tab contains topic metric data, both user-defined and entered, and those that can be automatically derived by Codestriker. Clicking on this tab will take you to the "Topic Information" page (see Section 3.8).

Following the "Download topic text" link will return in the browser the raw topic text which was entered when the review was created. This can be useful if the raw diff needs to be retrieved, so it can be used as a patch file. There are some links which affect the display, such as whether a mono-space font should be used to render the code changes (requires more screen real-estate), whether to preserve the code's line-breaking or not, and what tab-width to use.

This is followed by the table of contents of the review, which consists of the list of files which comprise the topic, whether they are added, removed or modified files, appropriately colour-coded. If a file has been modified, the revision of the file that has been modified is displayed. If filename is clicked, the browser will move to the anchor which corresponds to changes made to that file.

The rest of the "View Topic" page can be seen in Figure 3-6.

Figure 3-6. View Topic Detail Screenshot



This shows the start of the code changes for the file `codestriker/codestriker.pl`. Clicking on the filename will show revision history of this file, if Codestriker has been configured to link with a system, like ViewCVS. The "Go To Contents" link will move the browser back to the table of contents.

From this point, the screen in split into two sections, where the left side represents the old version of the file, and right side represents the new version. In the figure, the code in the block of red represents code which has been removed in the proposed change. Adjacent yellow blocks represent code changes, as can be seen with the version number change. The green block represents new code which has been added, in this instance, the new constants for viewing modes.

Each file is broken up into a series of "deltas". At the head of each delta block, for both the old and new versions of the file, there is an indication of what line number the delta started at. In this situation, clicking on the "Line 24" link on the left hand side, will open a new browser window, which will contain the complete original contents of `codestriker/codestriker.pl`, anchored at line 24. Clicking on the "Line 22" link on the right hand side will contain the complete contents of `codestriker/codestriker.pl` will *all* of the proposed changes applied. Clicking on "Parallel" will do a similar job, but will show both the original and new version of the complete file side-by-side. See Section 3.5.3 for more details on what can be done with these complete file-based views, which can assist enormously in the review process.

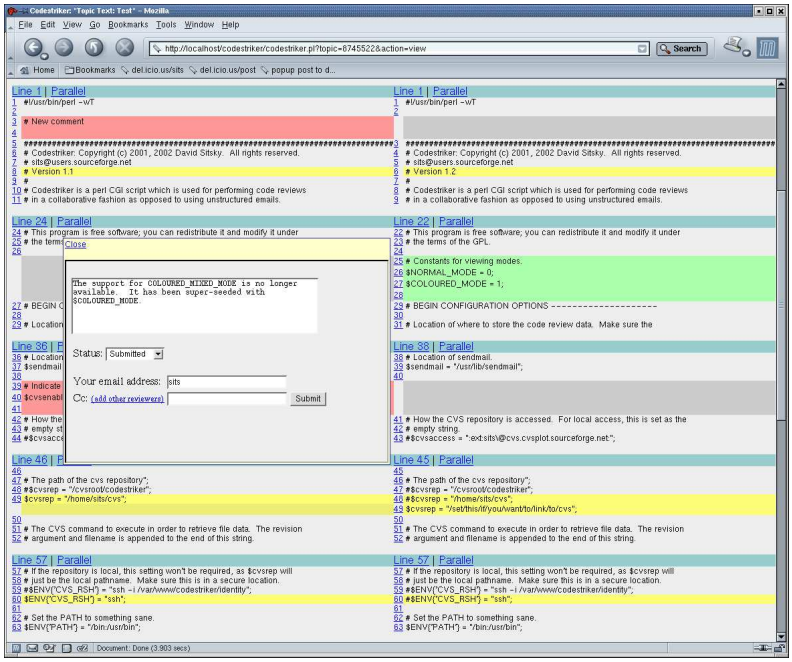
3.5.2. Adding a Comment

Every line of code in the display, is prefixed with a hyperlinked number. This number represents the line number of the file. If a comment needs to be made against a specific linenumber, click on the linenumber, and floating

window will come up with the "Add Comment" page (see Section 3.5.2). If a comment has already been made against a line, it will be red. Hovering over the line will bring up a tooltip window, containing the details of the comment made.

A comment against a line is made by clicking on the line number of interest. This will display a floating window, an example can be seen in Figure 3-7.

Figure 3-7. Add Comment Screenshot



As can be seen in the figure, the window contains a text box where the comment can be entered. Any formatting will be preserved, so that you can enter new code fragments, if required. Any comment metrics defined in the Codestriker configuration will appear here as a series of dropdown boxes. In this example, there is only the comment metric "Status" defined. As before, your email address is remembered in the browser's cookie, to prevent you from having to enter it each time you make a comment. When the "Submit" button is pressed, an email will be generated and sent to the author of the review, to the email address in the "Your email address" field as a reference, to the email address in the "Cc" field, just in case there is a requirement to send the email elsewhere, and to any other email addresses that have made a comment on this line, so that it is possible to get a form of discussion happening.

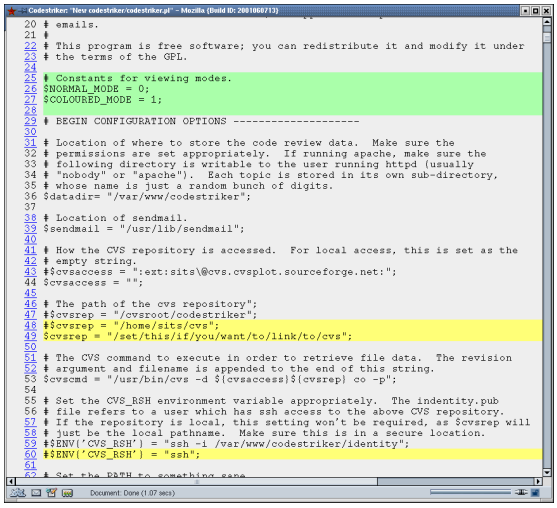
Once the comment has been accepted by the server, it will be automatically dismissed. If there was an error processing the comment, an error message will be displayed at the top of the window.

3.5.3. Viewing Complete old/new Files

As mention in Section 3.5.1, it is possible to view the entire contents of a file in a review in its original form, or in its proposed form, or side-by-side depending on whether the "Line" or "Parallel" link was selected. The review has to be linked to a SCM repository, so that it can fetch parts of the file which aren't a part of the review text.

When a "Line" link is clicked, a new browser window will come up with the contents of the file. This will be slow the first time, but in subsequent times will be fast since the browser window is not closed. An example of this window can be seen in Figure 3-8.

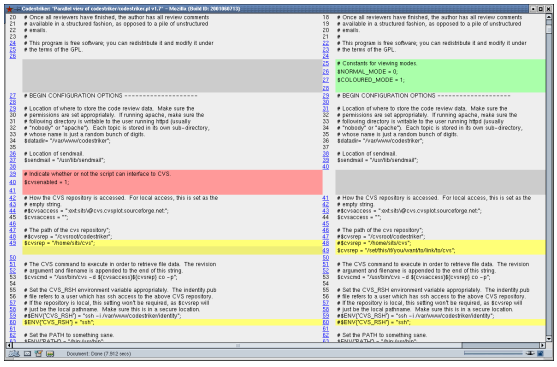
Figure 3-8. View File Screenshot



This view matches the new contents of `codestriker/codestriker.pl`, from the "View Topic" screen seen in Section 3.5.1. Each corresponding delta will have consecutive hyperlinked line numbers. Those lines which do not have a hyperlinked line number are outside the deltas in the review, but are a part of the file. Clicking on a hyperlinked line will bring the "Add Comment" window into display, in the same way it would as if the line number was clicked from the "View Topic" screen.

Seeing the entire contents of a file, either pre or post change is often essential to complete a review properly. Sometimes, it is necessary to see both versions of the file side-by-side. An example can be seen in Figure 3-9.

Figure 3-9. Parallel View File Screenshot



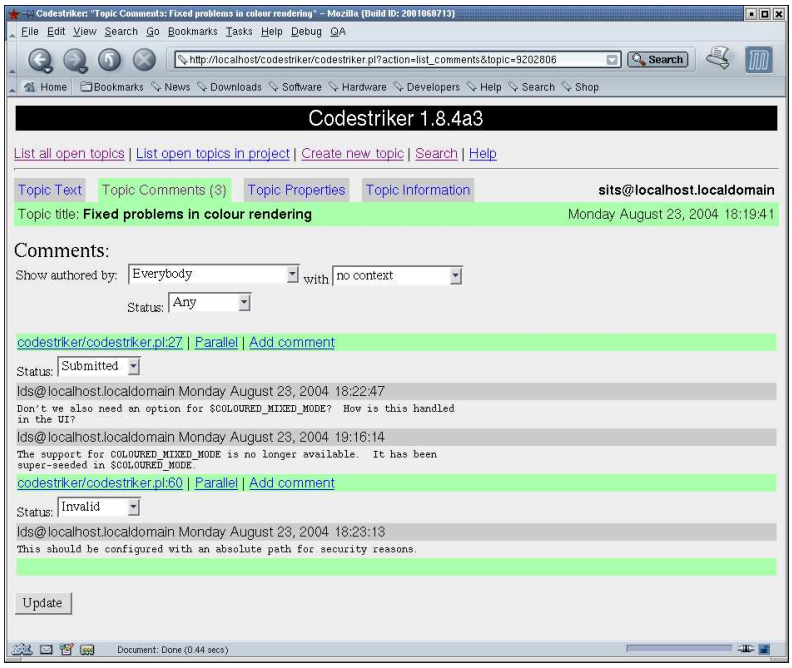
The same rules apply for adding comments - any hyperlinked line number can be clicked in order to add a comment.

3.6. Viewing Comments

From the "View Topic" page (see Section 3.5.1), the "Topic Comments" tab can be clicked to view the comments associated with a topic. This can be clicked at any stage to see what comments have been submitted against a review. It is often necessary for the author once a review has been completed, to work their way through the

comments from the browser, although they can do this with their email client as well (assuming `$allow_comment_email` is true in `codestriker.conf`), since each comment will generate an email message. An example screenshot of the "Topic Comments" page can be seen in Figure 3-10.

Figure 3-10. Topic Comments Screenshot



A separate heading is shown in the display, for each comment thread, which represents a series of comments made against a specific code line. For each thread, all of the comments made are shown, including author and date information. For each comment thread, the current values of the comment metrics associated with the thread will be displayed. In this example, there is only a single comment metric clled "Status". The metric values associated with the thread can be updated by changing the value of the metric dropdowns, and clicking the "Update" button at the bottom of the page.

For each comment thread, there are three links to other Codestriker screens. The first link will bring up a "View File" page (see Section 3.5.3) centered on the line number of the comment, "Parallel" will do the same, by using the parallel view, and the "Add comment" link will bring up the "Add Comment" window (see Section 3.5.2) for that line number.

In this particular example, there are two comment blocks, for file `codestriker/codestriker.pl`, on lines 27 and 60.

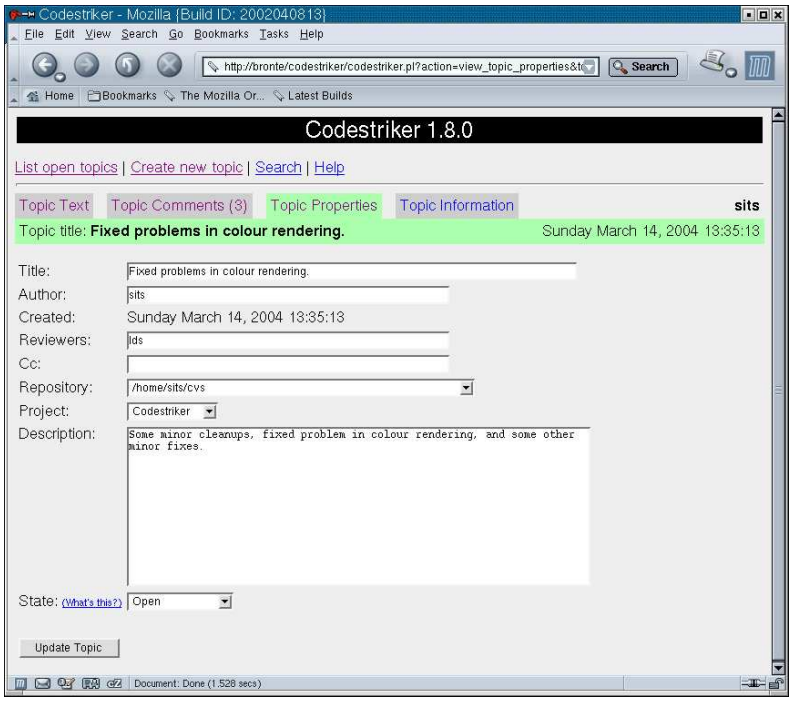
At the top of this page are a number of filtering options, which control what comments are displayed. Comments can be filtered by the author of the comment and/or specific comment metric values. In addition, for each comment block, it is also possible to display the context of the topic text the comment block is made against (as is done for the "Add Comment" page, see Section 3.5.2). These controls provide the capability for generating code review reports, which can then be used at code inspection meetings.

3.7. Topic Properties

From the "View Topic" page (see Section 3.5.1), the "Topic Properties" tab can be clicked to view and edit topic

properties. An example screenshot of this page can be seen in Figure 3-11.

Figure 3-11. Topic Properties Screenshot



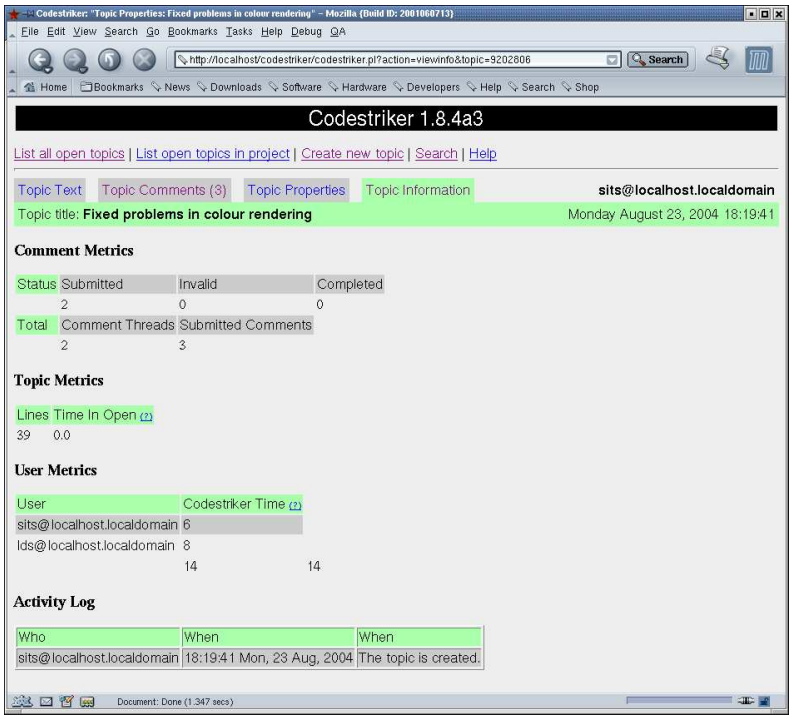
This page provides the opportunity to modify topic properties, such as the title, author, reviewers, cc, repository, project, description and the state of the topic. If Bugzilla integration is enabled, the list of associated bugids will be displayed here as well. Any changes in the author, reviewers and/or cc fields can generate new topic notification emails to alert the new parties involved in the review.

3.8. Topic Information

From the "View Topic" page (see Section 3.5.1), the "Topic Information" tab can be clicked to view and edit

topic information. An example screenshot of this page can be seen in Figure 3-12.

Figure 3-12. Topic Properties Screenshot



This page is divided into four sections. The "Comment Metrics" section provides an aggregate count for the number of comment threads with specific comment metric values. The "Total" subsection displays the number of comment threads in the topic, followed by the total number of comments submitted. This is followed by the "Topic Metrics" and "User Metrics" sections for this topic. If there are additional metrics defined here via the Codestriker configuration file, the user will be able to update their values here. The last part of the page is the "Activity Log", which provides an audit trail of all the important events that occurred in the topic's lifetime, such as creation and state changes.

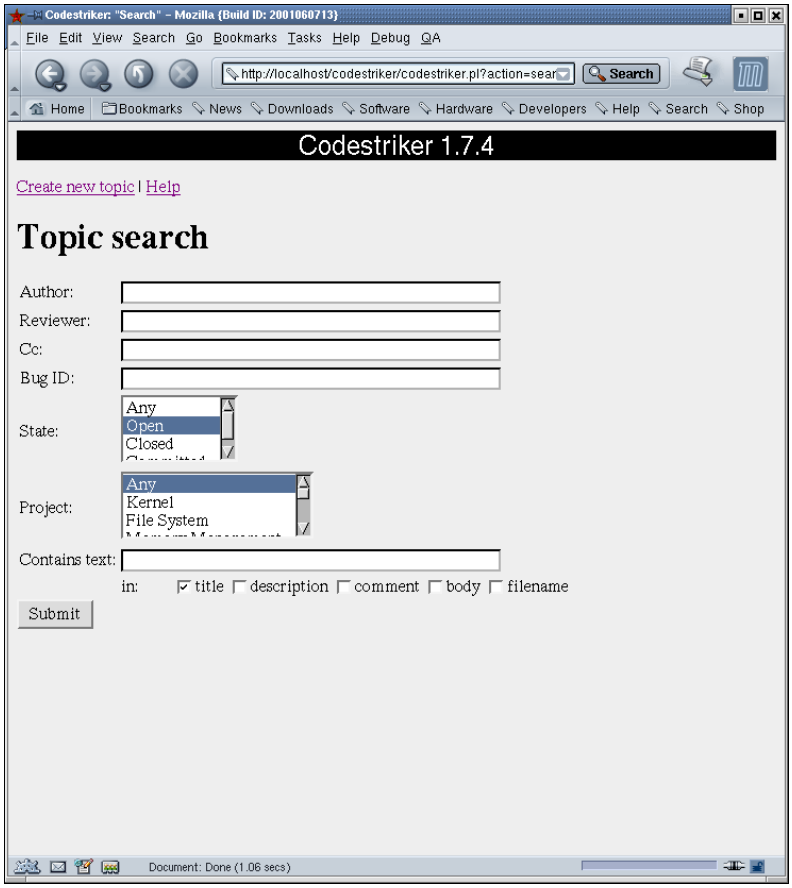
The Codestriker time metric is calculated by keeping timestamps of every refresh from the http server for the given topic. Now, just because you are not refreshing the topic does not mean that you are not looking at the topic. Codestriker will assume that you are looking at the topic for 4 minutes after the last refresh. If you click on a topic, and close the browser, it is logged as 4 minutes. The situation is unavoidable unless you put a camera on the computer and keep track of how long people are sitting at their computer, awake, and looking at the browser window. If this is not acceptable, users of Codestriker have the option to ask the reviewers to directly log the time spent in the review. The direct logging can be enabled via `$metrics_config` variable in the `codestriker.conf` file. The Codestriker time metric is "free" metric, being that it does not require any extra interactions with the user to operate. The trade off is that the accuracy of the data may not be as good as the direct logging done by a disciplined team. However, if your team is not very disciplined and can't be trusted to put in 100% accurate data, the Codestriker time metric will be better because it will not be biased by the reviewer's logging discipline.

The code that deals with all of the this is in `lib/Codestriker/Model/Metrics.pm`, in the `calculate_topic_view_time` method.

3.9. Topic Search

If Codestriker is used to review all new code before it is committed into a SCM, it builds up a very important development history. Since all this information is stored in a relational database, the information is searchable. The "Topic Search" screen provides an interface for locating topics, depending on a search criteria. The "Topic Search" screen can be seen in Section 3.9.

Figure 3-13. Topic Search Screenshot



By default, pressing "Submit" without typing anything extra will return all open topics in the system. This is because the "State" field is set to match any topic in state "Open", and the "Project" field is set to "Any", which will match any project, and there are no other constraints in the other fields. The search can be narrowed by entering in matching text in the "Author", "Reviewer", "Cc" and "Bug ID" fields.

The "Contains text" field in allows for searching in the following areas, depending on which checkboxes have been enabled (multiple checkboxes are ORed together):

title

: This will try to match the entered text with a topic's title.

description

: This will try to match the entered text with a topic's description.

comment

This will try to match the entered text with a comment associated with a topic.

body

This will try to match the entered text with the actual code associated with a topic.

filename

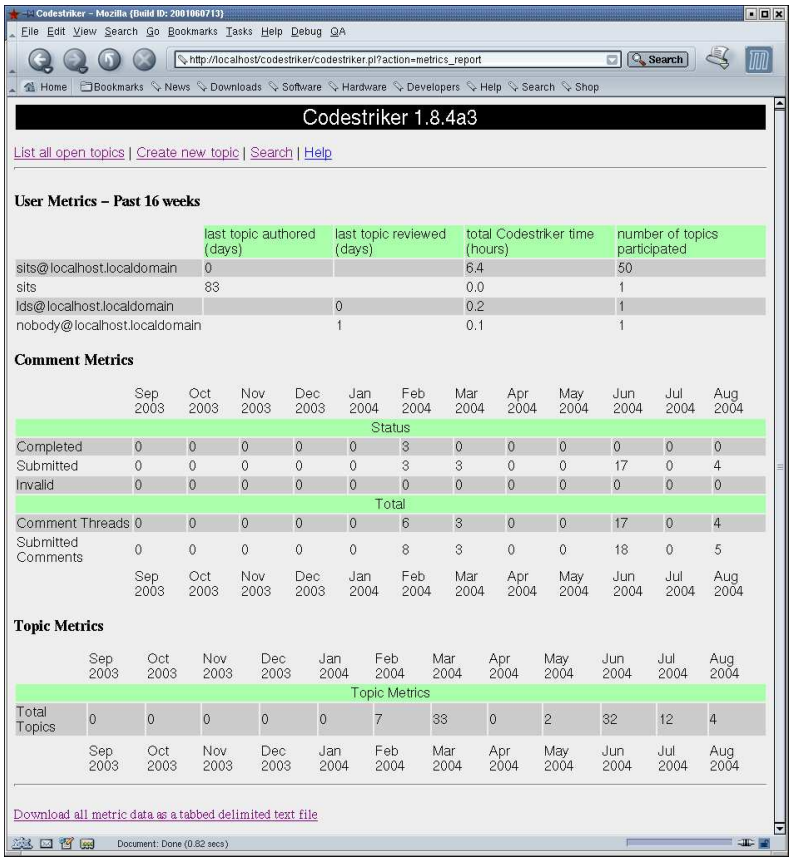
This will try to match the entered text with a filename that was a part of a topic.

In all cases, text entered will match the corresponding database text fields as a substring match. That is, entering "paul" in the "Author" field will match the author "paul@zot.com" or "joe_paul@zoo.com". It is also possible to use wildcards in the entered text, such as "joe*paul@zoo.com". Once the "Submit" button is pressed, the results are displayed in the "Topic List" page (see Section 3.2).

3.10. Metrics Report

From the "Topic list" screen (see Section 3.2), clicking on "Metrics Report" will take you to this page. An example screenshot can be seen in Figure 3-14.

Figure 3-14. Metrics Report Screenshot



As it can be seen, this page presents aggregated views for user metrics over the past 16 weeks, and topic metrics over last year, aggregated by month. For a real software team, this page will provide a wealth of detail as to how effective code reviews are. There is a link at the bottom of the page for retrieving this information as a CSV file, which can then be imported into other tools.

Chapter 4. Hacking

4.1. Codestriker Layout

This chapter is for developers which wish to contribute to Codestriker, and want a quick birds-eye view as to how the code is structured. The Codestriker modules are broken into the following sub-modules, which are located in `lib/Codestriker`:

Action

: Every action or gesture that can be performed on the Codestriker UI has an associated module for handling that action. The name of the module indicates what action it is handling.

BugDB

: Support for updating bug tracking systems when topics are created or their state is changed against specific bug reports. Currently, only Bugzilla is supported, but the API can support other bug-tracking systems.

DB

: A thin wrapper over the DBI interface, to create/remove database connections. Each database also has an associated module, to cater for subtle differences between database systems. Currently, there is support for MySQL, PostgreSQL, ODBC (SQL Server) and Oracle.

FileParser

: These modules are responsible for parsing the input text when topics are created. Currently, CVS unidiffs, PATCH unidiffs and Subversion diff files are supported. The API for these parsers is simply to accept input text, and to return an array of delta objects, which describe source code changes made to a specific file on specific lines. These delta objects are then used for rendering a topic, so that they are agnostic to which source control system they originated from.

Http

: This contains initial HTTP request processing, and support for generating an HTTP response, including the handling of cookies and compressed streams. This sub-module also contains the all-important Render.pm module, which is the work-horse for generating the complex HTML from the topic deltas.

Model

: This contains the persistent objects of the system, and the associated database operations on them. These objects include Topics, Comments, Files and Deltas. The Action classes eventually call these model objects when the persistent state of the system needs to be queried or modified.

Repository

: A number of objects implement the Repository API, which is used throughout the system. Repositories currently supported include a local CVS repository, a remote CVS repository whose data is accessed by HTTP or the pserver protocol, and Subversion. Future repositories can be easily added to this framework. The methods in the repository API include a way of retrieving the entire contents of a specified file and revision (which is used in the ViewFile action module), and the generation of a revision file log URL.

TopicListener

: This package contains a number of objects which are notified whenever a topic is changed, via a listener interface. Functionality such as email notification, Bugzilla integration, topic history recording are

implemented here as topic listeners.

In addition to the above modules, the following directories are of interest:

bin

This contains the all-important `install.pl` script, which is used for initialising the system, and for performing upgrades. This isn't the most pretty script in the world, partly for all of the old data-migration it has to do from old versions to the current version. This directory also contains `SubmitCodeReview.pl` which is a way of generating code reviews from the command line.

cgi-bin

This contains the codestriker.pl CGI script. It reads the "action" parameter from the URL, and delegates the handling of the request to the appropriate action object.

html

This contains the `codestriker.css` file.

templates

Almost all of the HTML generated by Codestriker (apart from the actual code rendering) can be customised by the templates within this directory. Each Action object typically has an associated template, where the view can be easily customized. There is also a header and footer template, which is included in every page, for site customization.

doc

This directory contains the source for the Codestriker guide written in docbook, contained in the `codestriker.sgml`, with the `Makefile` for building the various forms of the documentation.

4.2. Database Schema

The `bin/install.pl` script contains the **CREATE TABLE** statements that are used when the Codestriker database is initialised, in the `$table` hash. The following tables are defined:

project

A Codestriker database can consist of a number of projects, which have a name, id, description and creation date. Each code review topic, is linked to a specific project.

topic

A topic represents a code review that has been submitted to Codestriker. It contains attributes like the author, title, description, creation date, repository, and is linked to a member project.

participant

A participant is associated with a topic, and contains an email address, and the type of participant they are, such as the author, reviewer or observer (somebody was is CC'ed for the review).

commentdata

A comment is made against a specific review, and consists of the author and the comment text itself. It is linked with a specific commentstate object.

commentstate

A commentstate row represents a comment block - a list of comments made against a specific file and line number. The comment block has the notion of state, whether it is marked "submitted", "complete" or "invalid".

topicbug

This table records a many-to-many relationship between topics and bug ids. A topic may fix more than one bug id, and likewise, a bug id can be fixed by many topics.

topicfile

A topic consists on many file objects, which contain a filename, revision number and an indicator as to whether it is a text or binary file. A file consists of one or more delta objects.

delta

A delta is associated with a specific file object, and represents a single diff chunk or change. It represents for file X, at line number Y, the following change is a part of the review.

topichistory

Records a trail as to who modified what topic properties and when that occurred.

commentstatehistory

Records a trail for the modification of comment blocks.

topicviewhistory

Records who has viewed a topic and when.

topicmetric

Records the value of a metric against a specific topic.

topicusermetric

Records the value of a metric against a specific topic for a specific user.

4.3. Code Style Guide

- All comments must be complete sentences. Start with a capital letter, and end with a fullstop.
- Function calls should have no extra spaces around the `()`, so `function( x )`, should be written `function(x)`.
- Functions calls with more than one parameter should have spaces after the argument, so `function(x,y,z)`, should be `function(x, y, z)`.
- Limit line length where possible to 80 characters.

- All SQL statements should use uppercase for SQL keywords, and lowercase for everything else.

Chapter 5. Troubleshooting

If you have any problems, be sure to check first if it isn't a well-known Perl problem. The Bugzilla trouble-shooting page at http://www.bugzilla.org/docs216/html/troubleshooting.html documents them well. If the problem doesn't match, send a message (see Chapter 7), along with any relevant information (including relevant contents of your apache error log file), and I'll be happy to help you out.

Chapter 6. Future Plans

- Respository-specific topic-creation fields
- REST-style URLs
- User authentication
- Web-based configuration
- Per-user preferences for email
- Ability to review formatted design documents

Chapter 7. Contact Details

Please mail me any other suggestions you have to <codestriker-user@lists.sourceforge.net> or to me personally at <sits@users.sourceforge.net>. To subscribe to the Codestriker mailing list, go to the following URL: http://lists.sourceforge.net/lists/listinfo/codestriker-user. Contributions are most welcome!