Subversion and Trac Tutorial
Project in Automatic Control, FRT090

Johan Åkesson

1 Introduction

Trac and Subversion will be used in the course, respectively, for planning and follow-up of the project planning and version control of code and documents. Trac and Subversion (or similar systems) are commonly used both in industrial and open source software projects and are used in the course to further strengthen the skills of the course participants in these areas. The Trac site will be used continuously by the team supervisors to get an overview of the progress and status in the projects. It is therefore essential that the tools are used consistently and that the Trac site is up to date throughtout the project.

The objectives of this tutorial are:

- Perform initial set up of the Trac site by adding a milestone and a ticket.
- Make an initial commit of a document into the project Subversion repository.
- Practice cross referencing between Trac ticket pages and Subversion commit messages.
- Practice the Subversion update-commit cycle with different working copies.

After the tutorial, you should be able to continue working with Trac, e.g., to add additional milestones and tickets, and Subversion.

1.1 Subversion and version control

A version control system is useful when working on a collection of files that evolves over time, in particular if several people are modifying the files. For example, version control is often used in software projects to store the code in a way that several developers can access and contribute to the code without the need to send files by e-mail etc. Another example is when you are writing a paper with your colleagues and you need to efficiently and safely share your additions to the paper. A version control system also keeps track of the history of your contributions; most version control systems can be used to retrieve the

state of your files as of a given date. This means that there is no need to create local back-up copies. To make it short and sweet: version control means you can relax.

The version control system Subversion is installed on one of the departments servers. An excellent source of knowledge about using Subversion can be found at http://svnbook.red-bean.com/. Here you will find basically everything you need (and more) to use Subversion efficiently. Some basic guidelines for using Subversion are:

- A typical Subversion repository has three directories at the top level: 'trunk', 'tags', and 'branches'. Development usually takes place in 'trunk', so all the project files should reside there. 'branches' are typically used to create snapshots of the trunk to enable concurrent development, and 'tags' are used to mark releases. See the Subversion book for more info.

- Try to make atomic commits, meaning that each commit has a well defined scope. If several unrelated changes has been made to the code it is good to commit them separately. A good practice when using Subversion in association with Trac is to always work with a particular ticket in mind. In this case, commits are normally done in relation to tickets.

- Be careful to use the log-message facility. This makes it much easier to trace the origin of code changes. If Subversion is used with Trac, a reference to a ticket should normally be included in the log-message. Also, when a commit is made in relation to a ticket, it is useful to also add a reference to that commit in the Trac ticket.

1.2 The Trac project planning environment

In some larger, software-oriented projects involving several developers, it is useful to have a tool for planning, coordination, and documentation of the work. For this purpose, you will use Trac. Trac is a web and wiki-based planning tool in which tickets/bugs can be registered and assigned to developers, and milestones defined. Trac is also integrated with Subversion.

The user's guide for Trac is integrated with the system itself. The following guidelines summarize basic usage of Trac.

- Read the Trac documentation, and in particular the material related to wiki formatting and Trac references.

- The name space for wiki-pages is global. Therefore, it is important to use a convention that does not introduce conflicts. Here we use the CamelCase (http://en.wikipedia.org/wiki/CamelCase) strategy for naming wiki pages in combination with name mangling where CamelCase names are concatenated and separated with slashes ('/') to create a hierarchical structure. For example, if the wiki page 'GuideLines' is created at the top level, a typical name of a sub-page is then 'GuideLines/Trac'. Notice

that 'GuideLines/Trac' is considered by Trac as a global name and not a hierarchical one.

- Tickets typically follow a life cycle:
  - The ticket is created and a description is added.
  - The ticket is accepted by a developer that undertakes to perform the task specified in the ticket.
  - The ticket is resolved, usually by 'fixed', but there are also other options:
    * If a duplicate ticket is created, it can simply be resolved as 'duplicate'. This resolution is used also when a ticket is divided into two or more new tickets. Sometimes it is useful to create a very general ticket which is then divided into more specific ones. In such case, make sure that the new tickets refer back to the original ticket.
    * For some tickets it may be decided that the task will not be performed, for some reason. In this case, the ticket is resolved as 'wontfix'.
- Make frequent use of cross-references between tickets, wiki pages and subversion revision numbers. In particular, it is important to add a reference from a ticket to a subversion commit whenever a commit is made relating to the particular ticket. Conversely, try to always add a reference to a ticket in subversion commit comments. This makes it much easier to search and trace the origin of code changes. Tickets are referred to using the syntax '#1' (or 'ticket:1'), revisions by 'r1' (or 'chageset:1'). Please also see the Trac documentation for more information on wiki formatting and Trac references.
- The 'component' field of a ticket is used to classify tickets and group them into different categories. Different projects typically have the need for different components and are accordingly created based on the specifics of the particular project. Since adding components requires admin privileges, please contact the Trac administrator if new components need to be added.
- Once a ticket has been created, it is typically assigned to one of the team members by entering his/hers user ID in the Owner field of a ticket. In this way, Trac provides a means to allocate tasks to team members and for team members to get an overview of their individual tasks.

2 Tutorial Exercises

This tutorial will walk you through the first steps towards setting up and start working with you Trac planning site and with Subversion. Both Trac and Subversion are well documented and you are encouraged to read more about the tools to further improve your skills in working with these tools after the tutorial.

1. Set up a milestone and create a ticket in Trac.

   (a) Point your web browser to your project Trac site and log in with the identities you should have received by e-mail.

   (b) Have a look around and get acquainted with wiki editing by adding a welcoming text to the front page.

   (c) Click the "Admin" link in the menu bar and then the link "Milestones".

   (d) Start by removing the predefined milestones.

   (e) Create a new milestone, name it "Week_1" and set its completion date to match that of the project plan.

   (f) Once created, the milestone will appear in the list of milestones (currently only containing one milestone). Click the milestone name and add a description.

   (g) Click the "Roadmap" link in the menu bar and verify that the milestone you just created appears on this page.

   (h) Go back to the Admin page and click the "Components" link.

   (i) Remove the predefined components and create a new one with the name "Documentation". Leave the Owner field empty.

   (j) Click the "New Ticket" link to create a new ticket. The first ticket is about finishing the project plan and adding it to the Subversion repository. Enter the following information in the form:

   - Summary: "Write a project plan"
   - Description: A description of the task.
   - Type: "Task"
   - Milestone: "Week_1"
   - Priority: "Major"
   - Component: "Documentation"
   - Owner: A user ID of one of the team members.

   (k) Go back to the Roadmap page and click on the Week_1 milestone. Verify that the new ticket appears when you click on the link "Active tickets".

   (l) In order to quickly get an overview of the project status it is convenient to have lists of open and fixed tickets on the Trac front page. To achieve this, add the following lines to the wiki front page:

```
== Open Tickets ==
[[TicketQuery(status=new|assigned|reopened|accepted,format=table)]]

== Closed Tickets ==
[[TicketQuery(status=closed,format=table)]]
```

2. Commit code to the repository.

   (a) Create a new directory where you will check out your working copy.

   (b) Check out the content of your project repository and create a local working copy with the command:

   ```
   > svn checkout https://www2.control.lth.se/svn/REPO\_NAME/trunk REPO\_NAME
   ```

   I you are using Subversion under Windows, right-clicking in a directory shown in the file browser, e.g., "My Documents" and selecting "SVN Checkout..." brings up a dialog where the corresponding information can be entered.

   This operation will result in a new directory named `REPO_NAME` which will be referred to as the root of your working copy in the following.

   (c) Create a new directory in the root of you working copy for the project plan and add it to the repository.

   ```
   > mkdir project_plan
   > svn add project_plan
   ```

   In Windows, you may create the new directory inside the working copy root and then select the menu option "TortoiseSVN→Add" in the right-click menu. Notice that the new directory is not yet uploaded to the repository, just marked to be added in the next commit.

   (d) Copy the file(s) of the project plan to the new directory.

   (e) Add the file(s) to version control by using the `svn add` or the corresponding functionality in Windows.

   (f) Commit the files and write a commit message by placing yourself in the root of you working copy and enter the command:

   ```
   > cd ..
   > svn commit -m''Added the project plan to the repository, see #1.''
   ```

   In Windows, right-click the working copy directory and select "SVN Commit..." to bring up the commit dialog.

   Make a note of the revision number of the commit.

   (g) Make a comment at the corresponding Trac ticket page and make a comment referring to the commit. A suitable comment is:

   ```
   '''changeset:2'''
     Added the project plan to the repository, see #1.
   ```

   (h) Click the link "changeset:2" to see a summary of the commit.

   (i) Click the link "Browse Source" in the menu bar and verify that the new file(s) appears in the file browser.

3. Check out the code and make some modifications to the versioned files.

   - Open a new terminal window and create a temporary directory.

- Check out another working copy of the repository by following the same procedure as above.
- Make a change to one of the files in the project plan directory, for example add or remove some line breaks or change a formulation.
- Commit the changes. Don't forget to add a commit message with a cross reference to the Trac ticket number.
- Make a comment at the corresponding Trac ticket page and refer to the commit as described above.
- If you are happy with the project plan you may close the ticket by choosing the option "resolve as fixed".
- Click on the menu option "Timeline" to see a list of the operations you have performed in Trac and Subversion.

4. Update a working copy to check out file modifications.

   (a) Go back to the original working copy root.

   (b) Update the working copy to get the latest changes by entering the command:

   ```
   > svn update
   ```

   or equivalently, right-click the original working copy root directory and select "SVN Update".

   (c) Inspect the project plan files and verify that the changed made and committed from the other working copy has been downloaded.