Paper PP23

Hoarding Programs and Challenges with Programming Collaboration? Version Control to the Rescue!

Siva kumar Ramakrishnan, Vita Data Sciences, Waltham, MA, USA

Abstract:

In our programming experience, we must have come across 1) situations where programmers must have saved copies of previous versions of the programs as a back-up encompassing snapshots of the programs at various deliverables and 2) different copies of the programs worked on by different programmers gets saved as a back-up to reference for later period. This is a programming hoarders safe-haven and a pragmatic programmer's nightmare. In the long run this will cause an unmanageable number of files to keep track of with no traceability of the updates between each copy of the programs. As a solution to this challenge, in this paper we will show how with the use of version control, and with the proper versioning of programs we can tackle these situations. We will also discuss some great features available to us through version control tools which can be of immense help to the end-users.

Introduction:

In a general sense you may put a pause on a validated programs to achieve a particular deliverable, but a program is never a finished product. A program may have started from an initial idea, paused at a certain point in time to achieve a certain goal, but ideas keep evolving, requirements keep evolving and that often requires iterations, updates, and revisions to the programs. Due to the collaborative and evolving nature of the programs, two key aspects need to be tracked 1) users updating the programs and 2) changes to the programs at different points in time, otherwise users will be left with several different copies of the same source programs with different user edits, copies of different edits to a program to support various deliverable needs, or may just loose permanently the snapshot of the program which supported an earlier deliverable . The downsides to these uncontrolled edits are lack of traceability between updates to the programs, inability to identify the latest version of the program, difficulty reproducing results of a submission amongst many others.

Version control provides a solution to these challenges with the a) ability to provide the history of changes to any information on a file/document including the author, date, time, and written notes on the purpose of each change. b) ability to revert changes to a previous revision. c) ability to manage changes to any information on a file/document (addition, modification and/or deletion).
Version control also seamlessly answers the following questions: What was edited? Why was it edited? Who made the edit? When were the files edited?

An organization may use any of the several version control systems available in the market to achieve their goals, however the underlying concepts, incentive to use version control and challenges overlap.

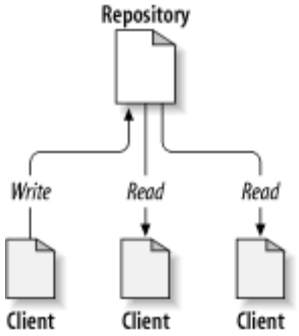
Before we delve further, we would like to make the readers familiar with the term repository as that will be quite frequently used.

What is a Repository?

A repository is a central store of data. The repository stores information in the form of a filesystem tree - a typical hierarchy of files and directories. Any number of clients connect to the repository, and then read or write to these files. By writing data, a client makes the information available to others; by reading data, the client receives information from others. [1]

Figure 1 [1]

A Typical Client/Server System



Functionalities of Version Control:

In general version control systems provide the users with the ability to

1. Add files to the version-controlled repository.
2. Commit files to the version-controlled repository.

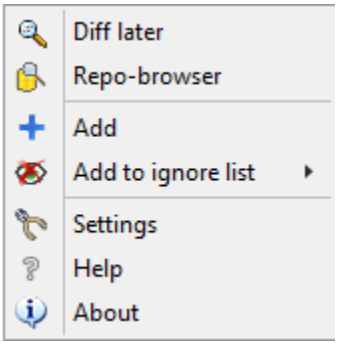
3. Revert the updates made to a file.
4. Show the modification log of the files.
5. Compare any previous version of the file with working copy.
6. Compare a file with a previous version from the version-controlled repository.
7. Save file to a previous revision from the version-controlled repository.
8. Delete files or folders from the version-controlled repository.
9. Check for modifications to files.

Below we will briefly look at some of the functionalities available through version control.

- **Add files to the version-controlled repository:**

  The very first step after creating the directory or a program is to add them to the version control repository.

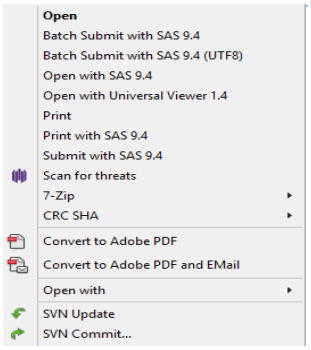
  Figure 2:



- **Commit files to the version-controlled repository:**

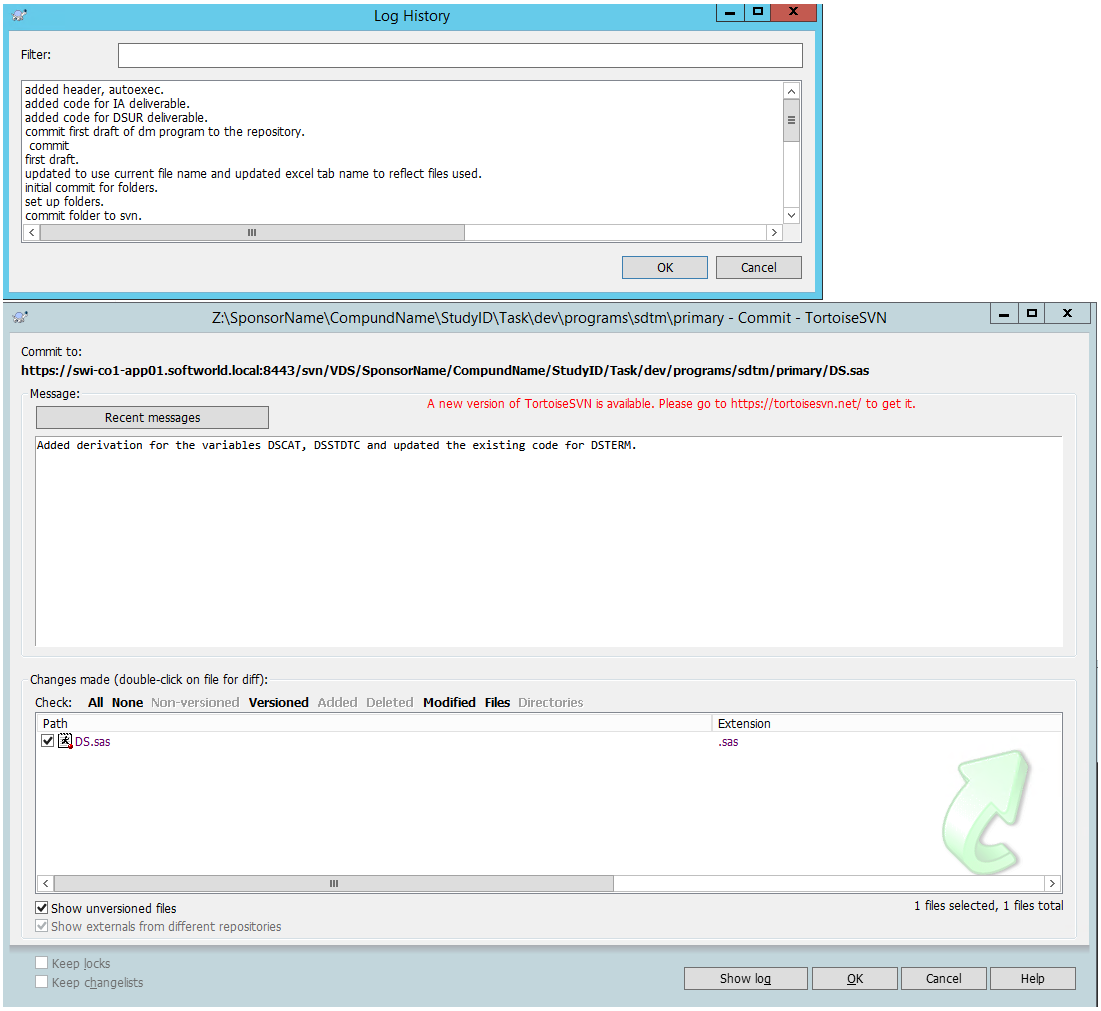
  Once a file is added to the repository and the development starts, the next logical step is to commit the files to the repository. A file can be committed as frequent as needed, but a general rule of thumb is for a programmer to commit the file at least once by the end of their working day and whenever a sizeable update is made to the program. Once the file is committed, it is available for other users to check-out and make edits.

Figure 3:



During the commit, the programmers are given an opportunity to enter the reason for their commit. A new reason can be entered, or the programmer can choose from the most recent commit messages used earlier, if the update to the program is similar in nature to other similar updates performed to other programs during an earlier commit.

Figure 4:



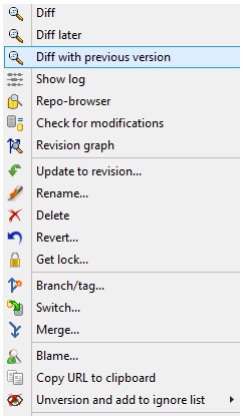
- **Revert the updates made to a file, check for modifications, and delete files:**

  There may come a time, when the programmer may want to scrap some of the most recent edits that they made. If not for version control, the programmer would have had to manually save an older copy of the program, or the programmer would have to manually go through the recent updates line by line, however with version control, all that the programmer will have to do is to choose 'Revert' from the context menu.

  Version control systems also provide the programmers options to

  - check for modifications to files and commit them if they exist,
  - compare their modifications to previous versions,
  - save previous versions separate from the current version in a version-controlled environment etc.
  - There are also options to SVN delete the files if necessary.

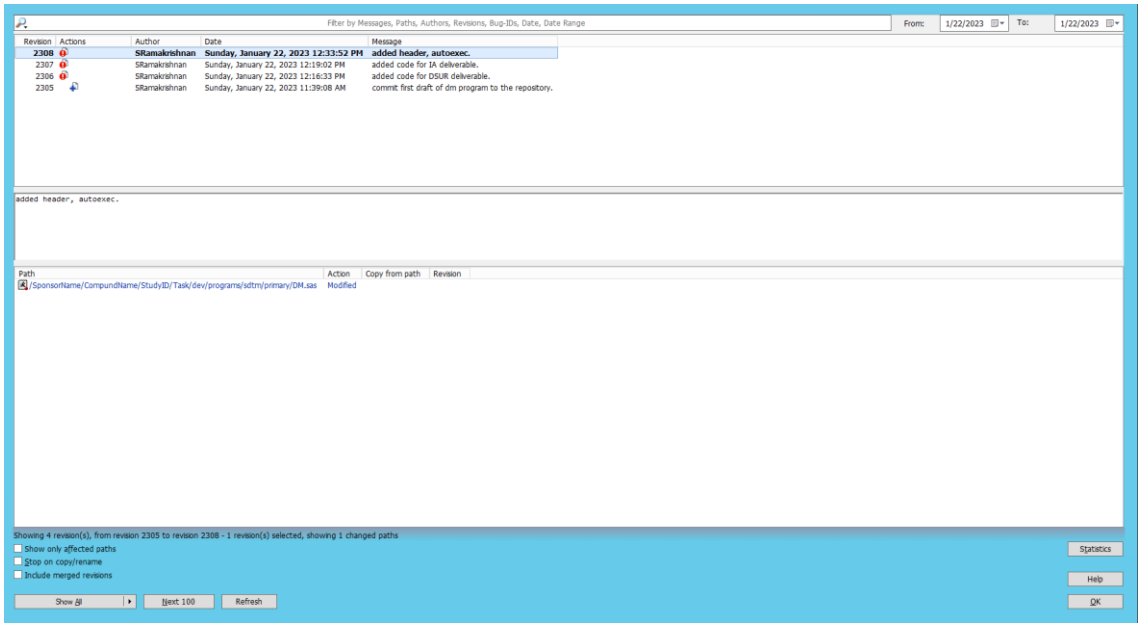
  Figure 5:



- **Show the modification log of the files:**

  Now, let's say a programmer has made several updates to a 'DM.sas' program over the course of several deliverables, version control makes it very straightforward to retrieve the version of the program that was used in the delivery of a particular submission. This is a very useful feature for program submissions to sponsors or regulatory agencies, while at the same time maintaining traceability, reason for changes and timestamp of changes to the programs.

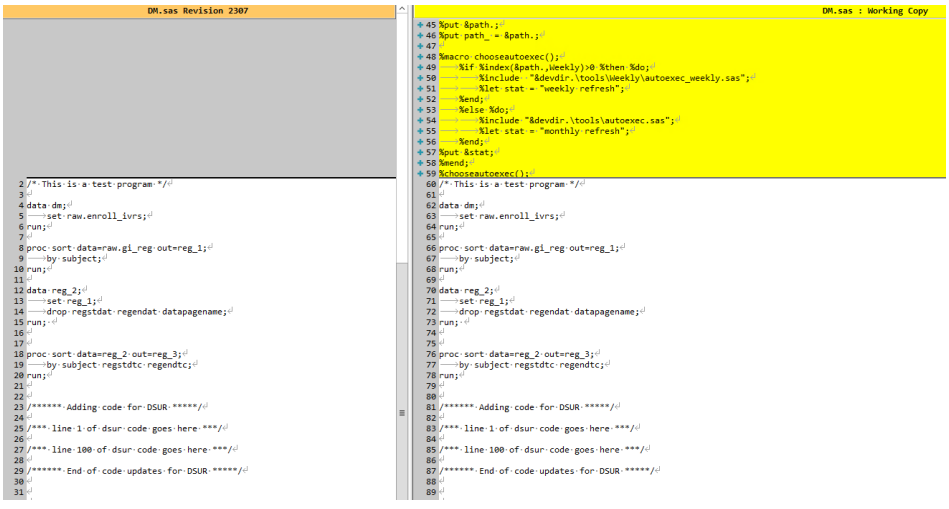
Figure 6:



- **Compare between different revisions or between revisions to a working copy:**

  As you can see from the below figure, the code to the left is a past revision (revision number: 2307) for 'DM.sas' program, while the code to the right is the working copy. The yellow highlighted code on the working copy is the latest addition that was added from revision 2307. With version control, a programmer can easily pinpoint the exact code that was either added, deleted, or changed between different revisions of the program or between one revision to the latest working copy.

  Figure 7:



Closing thoughts:

Apart from a plethora of use cases as discussed above, using version control system also allows us to keep a cleaner study area as shown in figures 8 and 9.

There is just a single program each for DM and DS in our example. When it comes to DM there has been several revisions to the program to accommodate for DSUR updates, IA updates and other miscellaneous updates over the lifecycle of the program. Instead of saving several copies of the program as in figure 10, version control gives the user the ability to just maintain a single copy while at the same time providing the option to retrieve the exact code that was used for a particular deliverable or maintained by a particular author.

Cleaner folder from using version control:

Figure 8:

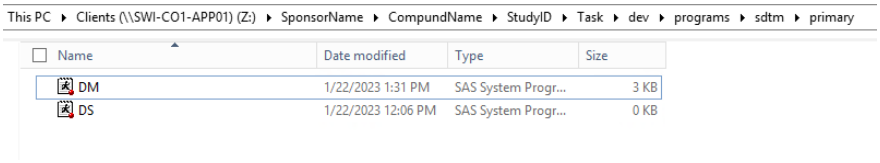


Figure 9:

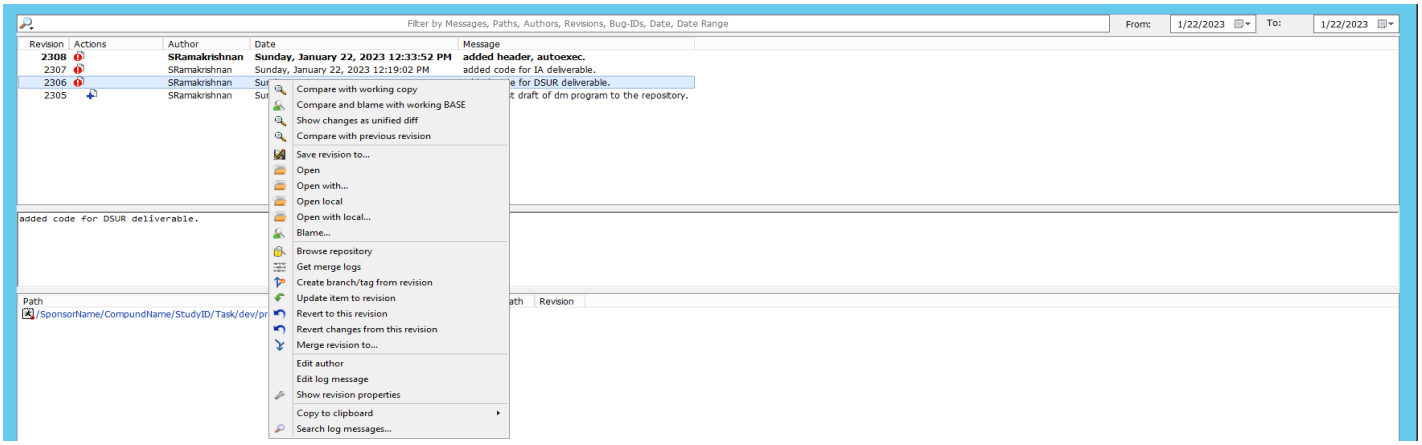
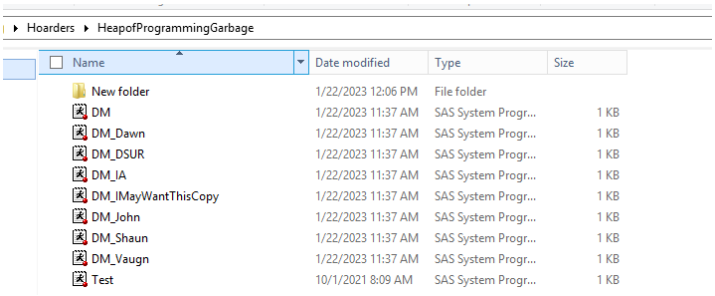


Figure 10 provides a glimpse of a would-be study area if there was no version control.

Figure 10:



Conclusion:

In conclusion, using version control in a controlled programming environment provides significant advantage over a non-version-controlled programming environment. It is simple, easy to trace back to an earlier version of the program, and know: What was edited? Why was it edited? Who made the edit? When were the files edited? while also saving significant time with maintenance.

References:

[1] https://tortoisesvn.net/

Contact Information:

Your comments and questions are valued. Please contact the author at:

Author Name: Siva kumar Ramakrishnan

Company: Vita Data Sciences

Address: 281 Winter St, suite# 100

City / Postcode: Waltham, MA, 02451

Work Phone: 765-212-8111

Email: siva@vitadatasciences.com

Web: www.vitadatasciences.com