Correctness of Data Mined from CVS

Christopher Thomson, Mike Holcombe
University of Sheffield
Department of Computer Science
Regent Court, 211 Portobello, Sheffield S1 4DP
+44 114 222 1980
{c.thomson, m.holcombe}@dcs.shef.ac.uk

ABSTRACT
Source code repositories managed by the popular CVS are frequently mined by researchers to validate various hypotheses about how the development of a software product progressed. This paper presents a study where the development process of 17 student teams was followed. It will show that in an environment where the teams were permitted to manage their own CVS repositories that several errors emerged that could lead to the misinterpretation of a hypothesis. These were classified into three types: type one errors which relate to the non-use of the system; type two errors that emerged from the direct manipulation of the repository; and type three errors from the limitation of CVS not to record file name changes.

Categories and Subject Descriptors
D.2.8 [**Software Engineering**]: Metrics – *Process metrics*

General Terms: Measurement, Experimentation.

Keywords: CVS, data mining, correctness.

1. INTRODUCTION
Source code management tools and their associated repositories have become important sources of information for data mining. The repositories often spanning several years of development offer the possibility for long term insights to development at low cost. The rise of open source software development and the CVS tool [3] has made many such repositories available without the restrictions typically imposed by industrial collaborators.

CVS is not without its limitations and other researchers have identified additional types and sources of information which can be used to inform analysis. These studies however typically focus on the information obtainable from CVS rather than the potential flaws inherent in using the repository produced. Whilst the discussion in this paper will be familiar to many researchers we hope that a clear documented classification of concerns will be useful to those planning future studies.

MSR'08, May 10–11, 2008, Leipzig, Germany.

This paper presents a study in which student developers working in teams to develop a software product used CVS. CVS was generally welcomed by the student teams, but few used it regularly. By the end of the project it was evident that four teams had become frustrated with it to the point that the CVS repository had been deleted and recreated resulting in the loss of the change history.

The errors discovered within this study should guide other researchers in the verification of the repositories that they are using. They may also suggest that some hypotheses will require repositories which have particular verified properties in order to assess them.

The most widely understood flaw in CVS is that of the name identity problem [4, 5, 8]. CVS does not store any information between file rename and move operations. Furthermore this identification process is further complicated by cut and paste and file duplication, which even other more advanced tools such as Subversion do not identify. To address these issues various tools and techniques have been developed [8].

Whilst CVS repositories have been used in many research studies the additional error types considered here will be of most relevance to studies conducted on student development projects and research which requires accurate data from the early stages of a project. Student projects are particularly vulnerable to errors made by the students because they don't understand how the version control system work, especially at the start of the course [6, 7].

2. EXPERIMENTAL SETUP
The results reported were obtained in a study of 17 self-selected teams of between 5-6 students working on one of four projects competitively. The development projects ran over a twelve week period where the student developers worked fifteen hours a week. Each project was provided by an industrial client who later used the software in his/her business.

The teams in question were all composed of second or third year undergraduate students; the second years studying computer science or software engineering and the third years on dual-honours programmes with maths or a European language. In terms of their computing curriculum all had studied similar courses including at least: programming skills in Java (10 ECT credits); systems analysis, design and testing skills (15 ECT credits); knowledge of database systems (5 ECT credits); knowledge of user interface design (5 ECT credits). The students had no previous instruction on using CVS; details were provided in a single one hour lecture at the start of the course and accompanying written notes.

The student teams were provided with two open source tools [2, 3] and instructions for managing their development by using CVS. A one hour tutorial was given in the use of CVS at the start of the project, and the instructor was available for consultation by the teams at a help desk session once a week for the whole development period. They were instructed to 'check-in' their files at least once a week. To avoid confusion each team had a nominated archivist who was required to ensure that this was done. For each check-in they were encouraged to provide a brief explanation in the comments field provided by CVS.

Each team was provided with a shared directory into which the whole team had read-write access. This was pre-populated with an empty CVS repository for their use. The teams also had the option to keep files in this directory separate from CVS in the case that they felt it was in appropriate to store them. The intention was that this facility could be used for bulky files (such as images) or documents supplied by the client. In reality several teams managed most of their projects in the normal directories.

3. METHOD

On the Friday of every week during the project the team directories are harvested, these formed the data set for analysis. Many of the files included are duplicates (as commonly the teams kept a checked out version of the CVS repository in the shared directory) and many files are written in natural language. The files were in various formats including; code in Microsoft Access, Java and PHP, and documentation in comments, structured databases and word documents.

The CVS repositories were analyzed briefly at the end of each week of the project. The analysis was made using the CVSPlot tool [1] which extracts basic statistics from CVS regarding the number of files in the repository and the cumulative number of lines after the submission. The graphs generated are based on log information retrieved from CVS and so show a true reflection of the exact time that the developer submits files to the repository. Minor modifications were made to the standard program to allow it to print the day and month (instead of month and year) and to process binary files. This analysis was used to show how the repositories were being used.

At the end of the project a through analysis was completed. For this a tool ('Sheffield Compare') was created that could extract various details from each copy of the repository made in each week. To aid this it includes an automated tool to spider a directory structure that is organized by team and sample point (in our case a directory for each week of the project). This automatically identified all CVS repositories and all files not in the repository.

The tool shows a list of files to a human operator from two sample points (in our case the start and end of a week). It automatically matches and aligns any files which have the same name that are also in the same directory. The operator views the remaining files to determine if any are unmatched files are the result of either moving the files or renaming them.

Sheffield Compare also included a number of analysis tools to aid the comparison of the files, including the ability to select specific file extensions, identify authors (using inline code comments), and compare files using 'Beyond Compare' a commercial diff tool. In the case of the comparison single files can be compared or a summary report of the whole data set. The purpose of this analysis was to identify unexpected changes which were not clearly reflected in the logs obtained from the CVS repositories available in the final week.

The first analysis was concerned with identifying the correctness the repositories in the team directories. This was calculated by firstly searching for directories named 'CVSROOT' (which are present in the root directory of each CVS repository) and then checking out each of the modules present.

Secondly for each file in each repository in each week a log of the history of the file was obtained using the CVS log command. The results of the command were noted and the timestamps compared across repositories collected in each of the weeks. This was to identify if the repository had been deleted and recreated as a whole, as it detected when the file history was missing between the two copies of the repository.

The number of errors in the analysis due to file renaming or moving was assessed by using the 'Sheffield Compare' tool to manually align the files between the weekly snapshots. This analysis however did not take account of any cut and paste or duplication activity.

Lastly the files not in CVS were counted to evaluate the overall use of CVS by the team.

4. ANALYSIS AND RESULTS

4.1 Type 1 Errors: Non-usage of CVS

The initial evaluation of the repositories revealed that all of the teams attempted to use the CVS system. In some cases they either abandoned the use of it or encountered problems. By the final week of development there were four teams that had some files present in their shared directory that were not in the CVS repository. Calculated across all the teams in the final revision this equated to 0.16 missing files per file in CVS (2869 files in CVS, 401 files not in CVS).

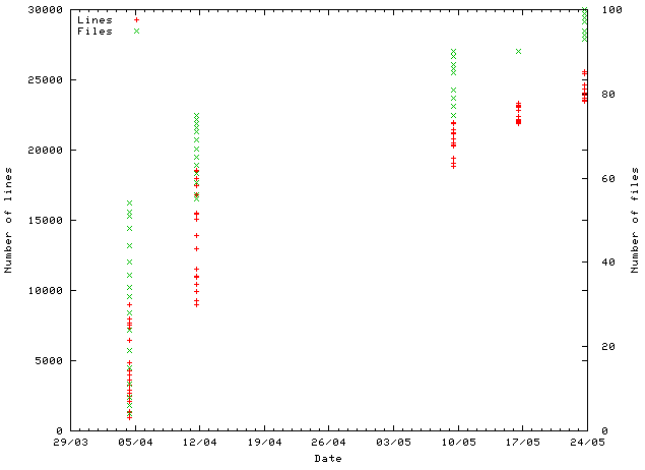

Figure 1. Example showing that for a single team files were added to CVS only once a week.

Analysis of the repositories showed that many teams used CVS exactly once a week, this was the minimum requirement for the module. This is illustrated by figure 1 (as generated by CVSPlot) which clearly shows weekly stepping as the code was added to the repository. This data is a true reflection of the exact time that the files were submitted as for instance shown by figure 2 which

shows file alterations without major steps. In this case only data points are present with a gap of three weeks (19/4 to 3/5) which was due to a vacation that fell towards the end of the twelve week period. This may not be particular problem in itself, but if a hypothesis was based on when files were changed much of the potential information could be lost.

Whilst this behavior could perhaps be a true representation of the periods when the developers worked on the project (thus inferring they conducted all their project work on a single day each week) this is highly unlikely. As stated in the setup the developers spent on average fifteen hours a week on the project, this would make for an exceptionally long working day, and probably not something that would be repeated throughout the project.

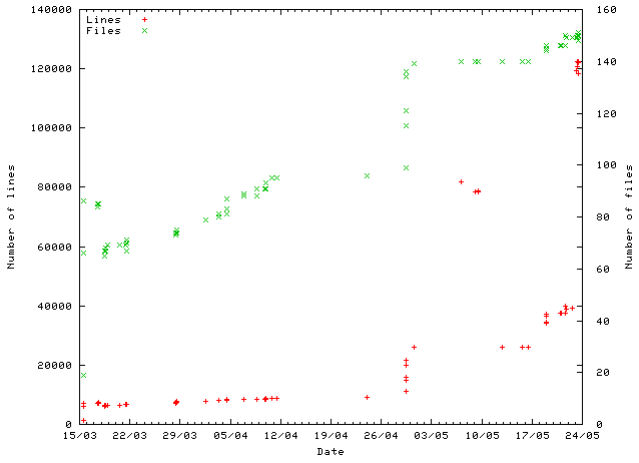

Figure 2. A CVS repository showing regular submissions.

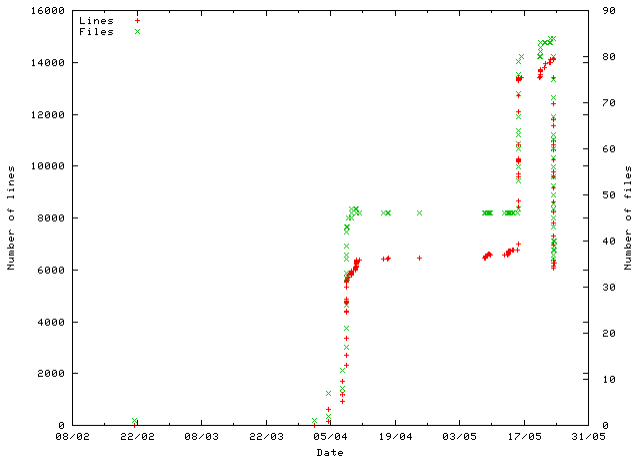

Figure 3. Example showing that for a single team files were added to CVS only once a week.

This is a type one error which result from the developers studied not using the repository as expected. That is to say that CVS can only record what it is given. The extreme case is fairly easy to identify, but in the case where some files are in the repository but the majority are not this becomes much harder. The stepwise submission of files is perhaps much harder to detect, take for example figure 3. There are still clear steps here, but also data points in between. This perhaps suggests that the majority of the team updated at the same time and one member updated it more regularly. However a pattern like this may be harder to detect automatically.

4.2 Type 2 Errors: Repository Corruption

When validating the repositories a further error was found with two repositories. In one case the repository did not checkout the files it clearly contained. In the other case the team had not been using CVS properly and had written files directly to the CVS directory, thus bypassing the versioning system. In this case CVS still worked correctly but only returned the fraction of the files created by the team before making this error. This was only detected when the analysis tool detected a set of recursive directories. This underscores the need to manually verify that the repository is valid and has not been corrupted.

The validation process also examined the logs of the CVS repositories on a weekly basis. This was to attempt to detect if the logs had been tampered with over time. The algorithm checked that the revision of each file from each sample point had the same commit timestamp in the log. The result of the analysis showed that four teams had deleted and recreated their repositories, two in week two, one in week three and one in week seven (table 1). Whilst not a lot of information was lost for teams A-C, the loss of the repository in week 7 could have been more problematic. We believe that this was intentional deletion as the developers were aware that we maintained a backup if they needed to recover files. This suggests that special care should be taken when a hypothesis requires information from the start of a project as in this case that information was missing from the final repository.

A similar effect to the deletion and recreation event may be present in many repositories. This can occur (although it was not observed in this experiment) if the repository is not used until late in the project. In this case the project will appear with a large number of completed files, with no history showing how the first version developed.

Table 1. CVS repositories deleted and recreated.

	A	B	C	D
Week	2	2	3	7
Files lost	6	7	7	24
Total revisions (final week)	203	264	1116	157

These two errors are of type two where the developers thought they were using the repository correctly but they have introduced errors which may hinder research. The case of directly manipulated directories is hard to detect, so some care must be taken to validate this by hand. A creation and deletion event is also hard to detect and can only be revealed by examining archived copies of the repository. Both of these problems can be avoided by only allowing indirect access to the repository via a CVS server daemon but this may not always be appropriate or applicable.

Whilst this is a clear case of misuse, such errors can potentially occur in any repository where the experimenter is not in complete control. Therefore any studies that use historical archives should check who had direct access to the repository and verify that they only used it properly.

4.3 Type 3 Errors: Unmanaged Operations

This final classification of errors is well known and relates to the rename and move operations which are not recorded by CVS. Whilst these are intentional changes by the developer they are not recorded by CVS. Therefore a naive analysis which ignores this possibility coupled with a hypothesis that requires measurement of change (of either new files, or internal modification) would lead to an erroneous interpretation.

Type 3 errors were analyzed by aligning all the files by hand to detect those which were renamed or moved. The analysis was conducted within the Sheffield Compare tool. The number of operations of this type was found to be fairly low (number of errors per file in the final revision = 0.05), over the teams there was no particular pattern to their relationship as shown in figure 3.

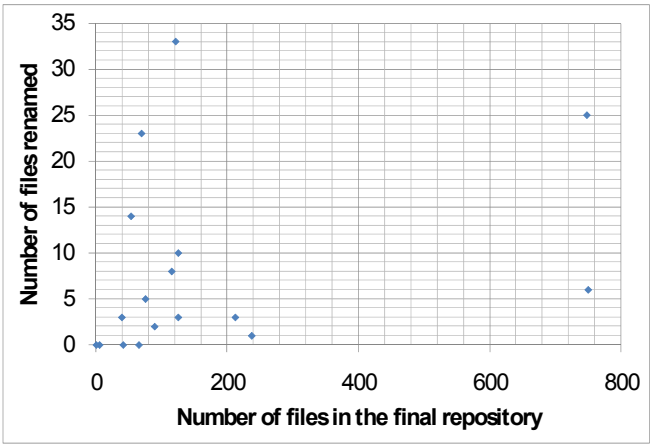

Figure 3. Plot showing the number of files renamed before the final version of the repository for each team.

5. Conclusions

CVS may be a good vehicle for looking at long term studies of software. This study however, has identified three categories of error which may occur when using it with students in short projects. Type one errors relate to the non-use of the system and were relatively common. The students perhaps had other priorities and delivered enough to comply with the requirements placed upon them and no more. Type two errors emerged from the direct manipulation of the repository where the students had unwittingly corrupted the data set by modifying the CVS directories directly. Lastly type three errors were encounter from the well known limitation of CVS not to record file name changes. However the error measured in this study as a result of this error was surprisingly low.

These observations are likely to be correct for most student projects, and so may have important implications for such studies [7]; however in industrial projects the developers are likely to be more familiar with CVS and consequently less likely to exhibit the same behavior as the students. This may or may not be the case with open source projects as it may be hard to determine the experience of the lead developers, particularly so if the repository is many years old.

Archives obtained from commercial development are perhaps less likely to exhibit some of these issues, however issues may arise if the research question is concerned with the early development of the project where repository management may be more lax. In particular it may be the case that for both commercial and open source projects that in some cases the repository is either established at some point after the start of project or is substantially reorganized.

These results have some implications for studies based on student repositories. In order to maintain the goodwill of the students it may not be possible to require them to hand in their projects on a more frequent basis, thus type one errors may be hard to avoid. Thus careful consideration must be placed on the choice of hypothesis to evaluate with student data. Enhanced instruction may be able to mitigate the effect of type two errors, but these could also be avoided by requiring the use of CVS and using the server access method. The number of type three errors was surprisingly low leading to the conclusion that this may not be important to avoid unless the hypothesis to be evaluated depends on this. If it is important then an analysis technique will be required to clean the data.

6. ACKNOWLEDGMENTS

This work was supported by an EPSRC grant: EP/D031516 - the Sheffield Software Engineering Observatory. During the period of research Thomson was also supported by an EPSRC studentship.

7. REFERENCES

[1] cvsplot.sourceforge.net.

[2] www.tortoisecvs.org.

[3] Berliner, B., CVS {II}: Parallelizing Software Development. in *Proceedings of the USENIX Winter 1990 Technical Conference*, (1990), USENIX Association, 341-352.

[4] Dig, D. and Johnson, R., The Role of Refactorings in API Evolution. in *ICSM '05: Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM'05)*, (2005), IEEE Computer Society, 389-398.

[5] Gorg, C. and Weisgerber, P., Detecting and Visualizing Refactorings from Software Archives. in *IWPC '05: Proceedings of the 13th International Workshop on Program Comprehension*, (2005), IEEE Computer Society, 205-214.

[6] Liu, Y., Stoulia, E., Wong, K. and German, D., Using CVS Historical Information to Understand How Students Develop Software. in *1st International Workshop on Mining Software Repositories*, (2004), 32-36.

[7] Mierle, K., Laven, K., Roweis, S. and Wilson, G., Mining student CVS repositories for performance indicators. in *MSR '05: Proceedings of the 2005 international workshop on Mining software repositories*, (2005), ACM, 1-5.

[8] Van Rysselberghe, F., Rieger, M. and Demeyer, S., Detecting move operations in versioning information. in *Software Maintenance and Reengineering, 2006. CSMR 2006. Proceedings of the 10th European Conference on*, (2006), 8 pp.