How Distributed Version Control Systems Impact Open Source Software Projects

Christian Rodríguez-Bustos
Dept. de Ing. de Sistemas e Industrial
Universidad Nacional de Colombia
Bogotá, Colombia
carodriguezb@unal.edu.co

Jairo Aponte
Dept. de Ing. de Sistemas e Industrial
Universidad Nacional de Colombia
Bogotá, Colombia
carodriguezb@unal.edu.co

***Abstract*—Centralized Version Control Systems have been used by many open source projects for a long time. However, in recent years several widely-known projects have migrated their repositories to Distributed Version Control Systems, such as Mercurial, Bazaar, and Git. Such systems have technical features that allow contributors to work in new ways, as various different workflows are possible. We plan to study this migration process to assess how developers' organization and their contributions are affected. As a first step, we present an analysis of the Mozilla repositories, which migrated from CVS to Mercurial in 2007. This analysis reveals both expected and unexpected aspects of the contributors' activities.**

***Keywords*—Contribution measuring; Distributed Version Control System; Mining Software Repositories; Mozilla, Open Source Software**

I. Introduction

In recent years, several Distributed Version Control Systems (DVCSs) such as Mercurial, Git and Bazaar emerged in the software field to provide a new way for versioning software artifacts. These tools have been adopted in the last years by many Open Source Software (OSS) projects, such as, Mozilla, Python, KDE, NetBeans, Eclipse, GNOME, etc., which used to employ CVS or SVN, two well-known centralized alternatives.

One of the main disadvantages of CVS and SVN, which frequently occur in OSS projects, is that contributors also called committers must be core developers (the ones who have writing permissions) in order to perform basic tasks, such as, reverting changes to a previously state, creating or merging branches, publishing changes with full revision history, etc. This limitation affects participation and authorship for new contributors, who have to follow a step-by-step joining process to become core developers [1]. Several authors [2][3] and developers from OSS projects[1][2] have mentioned that DVCSs allow non-core members to contribute and keep authorship easier than before, due to their characteristics (see Section II).

In this sense, we think some barriers that might restrict developers' participation and prevent researchers to clearly understand team's interactions and individual contributions can be reduced due to the emergence of DVCSs. Like some project leaders and developers[3], we also think it is worth assessing the true influence of DVCSs on OSS projects' workflows.

There are a number of aspects which need to be studied before we can clearly understand how a migration to a DVCS affects an OSS project. As an initial step towards this goal, this work aims at answering the following research question: *How does the migration from a CVS to a DVCS affect team members' contribution?*

We are addressing it through the analysis of (i) the authorship of changesests, (ii) the size and contribution of core developers; and (iii) the type of contribution based on the number of modified files before and after the migration.

II. Advantages of DVCSs

There are various papers that point out the advantages of using DVCSs versus centralized VCSs. In this section we summarize and categorize the most important ones into technical, structural, and research advantages.

A. Technical Advantages

From a technical point of view, the most important feature of DVCSs is that they allow individual developers to be servers or clients, like in peer-to-peer models. In this way, developers can work on source code without being connected to a central or remote repository (*offline mode*). The operations in this offline mode are performed faster since no network is involved.

Likewise, DVCSs work in terms of *changesets* (i.e., changes between one revision and the next one) instead of versions, where each individual file has its own separate revision history. This variation leads to easier branching and merging [4], and smaller repositories since less redundant information is stored. In addition, operations such as diff, log, branch and merge perform faster than in centralized systems. In this regard, David Miller, a senior Linux developer from the Red Hat project, mentioned two years after the migration of the Linux Kernel to BitKeeper (another DVCS): *"Before BitKeeper I was in merge hell every time a new kernel was released. Now I do real work instead of wasting time on repeated merging."*[4]

[1] http://lwn.net/Articles/246381/

[2] http://planet.mysql.com/entry/?id=13365

[3] James Henstridge, a Gnome project collaborator, shares our interest; he posted in his personal blog: "*It would be interesting to see the results of a study based on contributions to various projects that have already adopted DVCS.*"

[4] http://www.bitkeeper.com/press/2004-03-17.html

B. Structural Advantages

Regarding team structure, DVCSs allow teams to easily create, implement, or switch between different workflow models. Thus, the classic centralized model is no longer the only option: the *Integration-manager model* and the *Benevolent dictator model* are two examples of the alternatives allowed by DVCSs [4].

This kind of workflows allows teams to define intermediate roles that are responsible for testing, merging, reviewing and integrating changes from new developers. In contrast with centralized VCSs, the changes or contributions done in the intermediate repositories are kept in historical records. This represents a big advance for maintaining the authorship of changes done by non-core developers and for tracking the life cycle of branching, merging and administrative tasks performed by intermediate roles.

C. Research Advantages

This new generation of version control systems comes with the promise of new data, which will lead to new research questions related to how DVCSs affect processes, products, and people around software projects [2]. Moreover, this novel way of managing historical data brings some benefits to research in Mining Software Repositories and Software Evolution. First, the distributed repositories are smaller in size than the centralized ones, yet contain more information about contributions and workflows. Second, as a consequence of its reduced size, the data extraction and repositories cloning are faster than before. Third, it is also possible to recover the information related to the original committer or reviewer unlike the centralized repositories, where only committers (the ones who have writing permissions) appear as code contributors.

III. OSS Projects and DVCSs

Some OSS projects have considered some of the benefits mentioned in Section II, and have decided to try a distributed alternative for versioning. Several discussions and reasons in favor and against DVCSs can be found in the official communication channels of several OSS projects. As a way to exemplify these discussions, we present some relevant facts and comments about the use of these systems, reasons for migrating, effects of the migration, and developers' expectations.

- *Python project:* Python developers mentioned in the official Python blog[5] that one drawback of CVS and SVN is that non-core developers had limitations to contribute, due to the complexity when doing *"basic tasks such as reverting changes to a previously saved state, creating branches, publishing one's changes with full revision history, etc."* Also, they mentioned that *"there was no place to contain intermediate work"* such as reviews or testing performed by intermediate roles. They decided to start the migration to Mercurial in 2008.

- *Perl project:* After moving the Perl Project to GIT in 2008, in its official blog[6] was posted that now *"developers outside the core team can more easily work on experimental changes to Perl before proposing them for inclusion in the next release"*.

- *MySQL project:* The MySQL project team decided to use Bazaar as default version control system in 2008. Lenz Grimmer, a MySQL team member, said that he hoped *"that the change will significantly lower the barriers for external contributors and make it easier for their own developers to merge and accept changes that have been created by others and received sufficient community testing."*[7]

- *NetBeans project:* In 2009, the NetBeans project migrated to Mercurial, which, according to the development team, works faster than CVS, performs tasks in offline mode and maintains private forks more easily[8].

- *Mozilla project:* In 2006, the Mozilla Summer event took place, and became the starting point for finding a replacement for CVS within the Mozilla project. It was determined that CVS could not satisfy some requirements of the project[9] such as changesets, private/local branching, file forking/diverging with history, offline operations, pulling historical versions, and checking in changes to files in historical version, or keep committer ID separate from the actual person. After analyzing the pros and cons of available VCSs, they decided to migrate to Mercurial in 2007[10].

IV. Team Dynamics and DVCSs

According to the opinions cited on the previous section, one remarkable point is that DVCSs could generate a new way of contributing in the OSS communities, by facilitating the access to artifacts, allowing new users to maintain the authorship of their changes, tracking historical information of intermediate branches, facilitating the participation of non-core developers, and allowing teams to implement new workflows. In order to validate these assertions, we plan to analyze and compare some project features, before and after the migration to a DVCS. Specifically, for each project we are going to mine its centralized and decentralized repositories, and then, study these two data sets to detect and assess variations related to characteristics such as developers' contribution, workflows and roles. This analysis will be replicated with several projects to uncover patterns and draw more reliable conclusions about the effect of the migration decision.

To our knowledge, there are no published works that investigate this migration process and propose approaches to measure its impact. Therefore, based on the comments posted by developers and projects' leaders, and preliminary

[5] http://www.python.org/dev/peps/pep-0374/

[6] http://use.perl.org/articles/08/12/22/0830205.shtml

[7] http://planet.mysql.com/entry/?id=13353

[8] http://wiki.netbeans.org/HgMigrationReasons

[9] https://wiki.mozilla.org/Version_Control_System_Requirements

[10] http://soberbuildengineer.com/blog/2007/04/version-control-system-shootout-redux-redux/

studies regarding the analysis of developers' contribution, we started to analyze how the contribution has been affected by the migration of Mozilla repositories to Mercurial. Specifically, we decided to study the following variables, close related to developers' contribution.

A. Active/New Contributors per Month

We mentioned that many OSS projects intended to reduce the barriers for non-core developers. In order to determine whether this happened, we computed how many new different active contributors with at least one contribution join the project month by month, and compare this measure between both repositories. We unified contributors with same user and different domain, for example: *asasaki@mozilla* and *asasaki@netscape* were unified to *asasaki*.

B. Revisions/Changeset per Contributors/Month

The 20/80 rule proposed in [5] has been useful to characterize the developers' core of OSS projects. The rule means that approximately 20% of the contributors do 80% of the work. In this sense, we aimed at verifying if such proportion applies for the projects before and after the migration to a DVCS.

C. Files Added/Removed/Modified per Month

Another topic related to members' contribution is the rate of files modified. In this case we analyzed how many changes involve addition, deletion, or modification of files, and how many changes do not include files intervention month by month. We expect to find new information about the type of contribution in order to categorize them as a first step for defining contributors' roles.

V. Preliminary Study and Results

In a first stage of this research, we analyze the effects of migration on members' contribution within the Mozilla project. The migration process of this project lasted about 3 years, from 2007 to 2010. The Mozilla community decided to use Mercurial in order to satisfy most of their version control requirements (summarized in Section II), and overcome some difficulties generated by the use of CVS, their default control version system since 1998.

A. Data Sets

We considered the entire CVS and Mercurial historical data of the Mozilla project. In CVS were stored 212,396 changesets created from March 1999 to March 2007. In the Mercurial repository were found 68,828 changesets, created between March of 2007 and April of 2011.

B. Data Processing

To compare project features before and after migration, the CVS repository was converted to Mercurial using cvs2hg[11], a tool for migrating CVS repositories to Subversion, Git, Mercurial or Bazaar, with full historical data preservation. Then, customized Mercurial logs were generated and basic meta-data from each changeset such as date, author, node_id, and revision number were loaded into a relational data base using a set of Perl Scripts developed for this study. Besides, these scripts contain the queries used for computing the proposed metrics.

C. Active/New Contributors per Month Analysis

We found that after the migration to Mercurial, there were 18 new contributors on average per month, instead of four new contributors in CVS (see the boxplots in Figure 1). We also found that in CVS there were 669 different registered contributors) versus 933 in Mercurial. If we consider that CVS was used for 12 years and Mercurial have been in use only for four years, these increments are significative from a practical and a research standpoint. It may indicate that DVCSs strongly affect the rate of new contributors, or they better reflect the work that each developer does, or both.

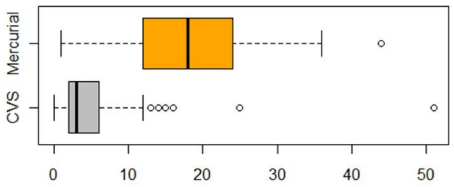

Figure 1 – New contributors per month

In this respect, these numbers can be interpreted as an indicator that all those members, who could not contribute directly in CVS and their changes needed to be submitted through core developers, can keep the authorship of their changes now. Additionally, it could be useful for sponsors, team leaders or intermediate roles to have full records of new and old contributors. Finally, this information could be useful for measuring the real growth of the Mozilla community around the development tasks, and for having a more accurate picture of its software processes.

D. Revisions/Changeset per Contributors Analysis

We found that for CVS the 20/80 rule is valid, i.e., the 80% of the changes was committed by 23.77% of the contributors, which is equivalent to 160 authors. For Mercurial, the same proportion of changes was committed by 10.63% of the contributors, which is equivalent to 99 authors. One can notice that the rule does not apply for the Mercurial repository. If the *main developers* are understood as the group of contributors who do the 80% of changes, these unexpected results indicate that the development core of Mozilla project decreased its size, after the migration to Mercurial. This contradicts the distributed essence that we expected to find. We plan to study this rule for several OSS projects that use DVCSs.

Another change identified and shown in Figure 2 is that after migration, the top 90 main contributors (shown in plot) did changes in higher proportions than before. We think this

[11] http://vc.gerg.ca/hg/cvs2svn/

may indicate that main developers in DVCSs have to do more administrative tasks such as review, merge, or revert changes. However, a deeper analysis of the contributions of main developers is necessary in order to categorize them, and identify the purposes of their changes.

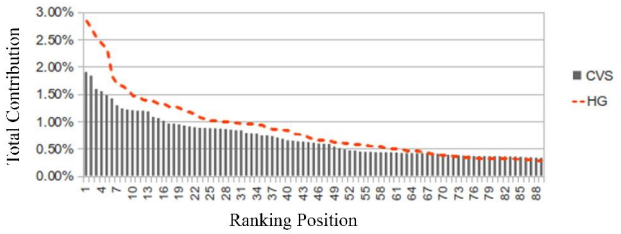

Figure 2 – Contributions of top 90 main contributors

E. Files Aadded/Removed/Modified per Month Analysis

We found that the average of changes that include file modifications does not change dramatically after migration. In contrast, the average of changes that add or delete files increases notoriously, as can be observed in Table 1.

TABLE 1 - AVERAGES OF CHANGESETS THAT MODIFY/ADD/DELETE FILES

Avg of changes by month with	CVS	Mercurial	Difference
files modified	1319	1224	- 7.2%
files added	118	216	+ 83.1%
files deleted	35	77	+ 120.0%
no files modified/added/deleted	1	31	+ 3000.0%

The average of changes that do not modify/add/delete files has relevant results. We analyzed manually the descriptions of some of those changes, and found that for CVS, contributors committed changes for annotating forgotten descriptions of previous changes. In Mercurial the changes that do not modify files were often related to merging, reverting, or reviewing other changes. Therefore, one way to categorize changes as administrative or code contribution can be achieved through the analysis of the number of files modified. Also, a deeper analysis is required to understand the increment of added and deleted files.

F. Revisions/Changeset per Month Preliminary Results

In the literature mentioned in the previous sections, it is stated that developers tend to commit changes more frequently when they are using a DVCS.

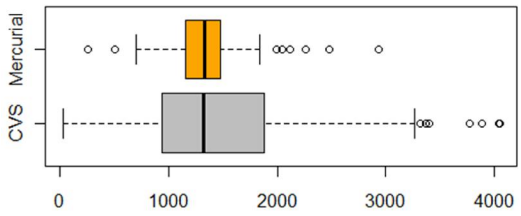

Figure 3 - Changes per month

Our preliminary results do not support this assertion because the average number of changes is similar for both repositories (see the boxplots in Figure 3). We plan to analyze this measure deeper to understand why the distribution of changes tends to be less biased in Mercurial than in CVS.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we introduced the study of the migration process from centralized to distributed version control systems on Open Source Software projects. We performed a preliminary study with the CVS and Mercurial data of the Mozilla project, by mining the metadata from both repositories and using them to analyze the effects of migration on the members' contribution to the project.

We found that Mercurial allows Mozilla team to record the authorship and contributions of all (core and non-core) members unlike CVS, where only core members appear as contributors. Besides, Mercurial provides stakeholders and researchers with more information to track the real growth of the community around the development tasks, and a more accurate picture of its software processes.

Regarding the main developers group, we found that its size decreases after migration: 80% of the changes were performed by a greater number of contributors. This unexpected finding contradicts the distributed essence that we expected to find in an OSS project that uses a DVCS. Although the amount of changes committed by each main developer increases, we argue that this could be caused by administrative tasks that they have to do such as merges and reviews. In order to clarify this hypothesis, we plan to categorize contributions according to their purpose and estimate the proportion between administrative and code contributions.

The results obtained so far in this preliminary study can be influenced by inner development processes such as general refactorings or widespread maintenance activities. In this sense, we also plan to compare the current results for the Mozilla project with other OSSs which have also migrated to a DVCS, and to study the effects of this migration process on projects workflows and member roles in order to uncover patterns and generalize the results.

VII. REFERENCES

[1] I. Herraiz, G. Robles, J.J. Amor, T. Romera, J.M. González Barahona, "The processes of joining in global distributed software projects*," In Proceedings of the 2006 international workshop on Global software development for the practitioner*, New York, NY, USA, 2006, 27-33.

[2] C. Bird, P. C. Rigby, E. T. Barr, D. J. Hamilton, D. M. German, P. Devanbu, "The promises and perils of mining git", *In Proceeding MSR '09 Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories*, 2009, 1-10

[3] B. de Alwis, J. Sillito, "Why are software projects moving from centralized to decentralized version control systems?", *In Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, IEEE Computer Society, 2009, 36-39

[4] N. B. Ruparelia, "The history of version control", *ACM SIGSOFT Software Engineering*, 2010, 35, 5-9

[5] S. Chacon, "Pro Git", Apress, 2009

[6] A. Mockus, R. T. Fielding, and J. D. Herbsleb. "Two case studies of open source software development: Apache and Mozilla". *ACM Transactions on Software Engineering Methodology*, 309–346, 2002.