Towards a Total Traceability of Research Collaboration:
a New Distributed Research Oriented Version Control System

Safa Taieb  
National Engineering School of Tunis  
Tunis, Tunisia  
safa.taieb@efrei.fr

Layth Slima  
EFREI  
Villejuif, France  
layth.sliman@efrei.fr

Benoit Charroux  
EFREI  
Villejuif, France  
benoit.charroux@efrei.fr

Yvan Stroppa  
LEO, CNRS  
Orleans, France  
vvan.stroppa@univ-orleans.fr

Fathelalem.F Ali  
Meio university  
Okinawa, Japan  
ali@mail.meio-u.ac.jp

Nejib Ben Hadj Alouane  
National Engineering School of Tunis  
OASIS  
Tunis, Tunisia  
nejib.bha@gmail.com

Abstract: Collaboration during scientific research activities has become an important issue due to the high competitive environments. Finding effective tools to facilitate collaborative project initiation, sharing ideas and data using the internet infrastructures is a challenge even with the most advanced technology and trained professionals. In this paper we describe a new online version control system equipped with a full action tracing system. Our tool is dedicated to people involved in the research activities. The tool aims to bring to researchers, developers, as well as domain experts stemming from multiple disciplines. The tools allows to co-develop scripts and computer programs to test and demonstrate research finding with a total traceability features, it allows exchange ideas, evaluate, test, reproduce and optimize research findings and artifacts and create a network for scientific community to be up to date with the latest development in this important field.

Keywords: Collaboration, traceability, co-development, reproducibility, scientific community and social network.

1. Introduction
Scientific collaborations used to be between scientists sharing the same geographical zone. Nowadays, researchers need more sufficient technologies to collaborate. Using new technologies of communication, collaboration has become much more easy and important to any researcher to move forward his research and validate his results.

Collaboration in research context is the fact of providing research collaborators with the ability to take advantages of existing research projects and trying to ameliorate or optimize them by investigating common questions and sharing resources and information.
Collaboration can occur in different steps of research production cycle including problem and requirement analyses, specification, modeling or design, prototyping, testing and benchmarking, and finally optimization [1].

Collaboration is the engine of research reproducibility. It makes it possible to contribute with other’s researches in order to improve their quality.

Collaboration in software development is the heart of the production process. Exchange among the software developers is essential to realize their common objectives. A lot of software tools managing collaboration are available to be used by developers to collaborate. We can distinguish between these tools referring to their use area, their functionalities, based on the high level approach to collaboration that the tool takes or considering the effort that the user make to collaborate [2].

Using such developing tools to enable scientists to collaborate is a possible solution. However, we can not confirm that it is the best solution to adopt.

In this paper, we will discuss collaboration in the research context and we will introduce a new solution to make researchers able to exchange and contribute, and providing them with an efficient traceability system.

The paper is structured as follows: the first part is the state of the art. In this part, we will analyze the most popular Version Control Systems VCSs and we will study the efficiency of using an existing tool for collaboration.

In the next part, we will present the new collaborative features of a research oriented platform based on a new solution for collaboration, and finally comes work concluding remarks.

2. State of The Art

Collaboration is an irrelevant element in the software as well as in the research production process.

In order “to keep many versions and configurations in an organized way” [3], Software Configuration Management SCM (called also source control or version control) is frequently used.

A variety of VCSs are available to manage the project evolution. Centralized Version Control Systems like Subversion SVN were built to offer source code control capacities such as branching and merging capabilities.

However, a centralized Version Control System VCS does not satisfy developers. They need more sophisticated tools enabling them to collaborate from distant places (on the Internet).

Thus, researchers need a collaborative tool offering security, speediness and distributed repositories.

Distributed VCSs like Git, Mercurial and Bazaar are frequently used in software development.

Git, the efficient and widely used tool, is “a fast, scalable, distributed revision control system with rich command set that provides both high level operations and full access to internals” [4]. It was initially developed for Linux system but it is also used for Windows.

It is equipped with a variety of tools to navigate the history of the system. It is known for its speediness, rapid branching and scalability which are important especially for huge projects.

Git is based on the concept of snapshots. After every commit, Git stores a picture describing the state of the repository at that moment.

Git associates a reference for every file. The reference is based on checksum mechanism. It is generated using a cryptographic hash function (Secure Hash Algorithm, SHA-1).

“A reference is a forty-character string composed of hexadecimal characters (0–9a–f)*” [15]. The objective of this function is to detect changes.

In the next commit, if there is a change, only modified files are stored. This concept brings a kind of efficiency in storage. So, we do not have to store all the files as version; we just need to store the changes. Figure 1 describes Git storage concept.
We can find several other DSCMs based on Git like GitHub. It is a web based hosting service to share and develop software. GitHub offers a graphical dashboard permitting new social perspectives of version control like profiling, social tagging and role-based action control.

As a direct concurrent of Git, Mercurial came with many similarities with Subversion but in a distributed context which make it easy to learn for those who are familiar with SVN. Major drawbacks of Mercurial include that it does not allow two parents’ mergence, unlike Git.

Bazaar is also a distributed VCS (DVCS) that supports many kinds of Workflows. It is used to fit almost any scenario of users and setups so it is easy to install and use. The good thing with this VCS is that it has a strong community that maintains it.

We find other alternatives in market that are based on different DVCSs and offering version control management capabilities.

Beanstalk is an alternative based on Subversion, Git and Mercurial. It offers the basic secure version control provided with notification messages about changes and an adapted interface of deployment. While Assembla have used Subversion, Git and Perforce build up a cloud based tool permitting Agile project management and scrum meeting using a video group chat. Another alternative is BitBucket which is based on Git and Mercurial. It offers REST APIs facilitating its integration [5].

In research context, many researchers want to find a space to collaborate with others and share their research findings but without caring much about the safety of their original projects.

The main purpose of collaboration in a research program is to get better results and to move forward the research. After a research contribution, the researcher wants to be able to switch to previous stable versions in any step during the research production process.

When the researcher gives the authorization to others to contribute with his research, he wants to keep a trace of his work and other traces of others’ contributions. The collaborators could be from different affiliations and not necessarily developers so they need a tool to assist them to make their contributions and facilitate their work.

Consequently, we do not need just simple VCS managing versions but we also need a user friendly tool assisting users to make changes.

To keep a trace of collaborators’ contributions, the first solution to come in mind is to use an existing VCS SCM. This could be the easiest solution to adopt.

Using this solution, we do not have to invent new VCS, we just need to add some changes to the SCM software code and to create an adapted user interface to help collaborators to make their contributions.

We can study the efficiency of such solution based on Git, one of the most popular and used DVCS. Git as we have mentioned above is a useful source code control system for software development.

### 2.1. Merging Conflicts Problems

As a distributed version control system, Git merges branches which have diverged from a git repository. Mergence take place when two distant users clone the same initial version, make changes on it and want to incorporate their work. It is done automatically when a user asks to pull (to extract) other’s changes or to push (to add) his changes.

The merging conflicts appear when the auto merging fails. It is always the case when the users are working on the same part of the code and one of them is adding something while the other delete a part of it so the person who is adding is using inexistent part of code in the other repository.

In such case Git ask the user to fix the problem manually in order to merge his modifications.

In research context, scientists are always working to resolve a specific part of the research project problem. Thus, they have to fix an objective and try to find the adequate solution. The solution implementation is
usually a small project which is different of huge projects in software development field. The frequency of the code modification is always low.

Merging conflicts in such context will be frequent which means that if we want to use this tool, we have to find a way to avoid merging conflicts.

This leads us to conflicts awareness. Awareness is “an understanding of the activities of others, which provides a context for your own activity” [6]. What we mean by awareness in this context is the early detection of conflicts in order to avoid it. When we talk about awareness, we can distinguish between Presence awareness that informs where others are looking in the code [7], and Change awareness informs where changes are being made in the code, which may help detect conflicts earlier. However, reporting, that is files, types, or program elements being changed may overload developers with notifications that are irrelevant to what they are doing [8], [9], [10], [11]. Although there are different trends to reduce the number of notifications, this kind of solutions still is not adapted to the research context where researchers are not necessarily developers and could not understand these notifications.

Another problem with resolving conflict is the file where git stores the way it has resolved a particular merge conflict in the past so that it can replay the same conflict resolution in the next similar mergence conflict. This file, called “.git/rr-cache” is not automatically copied from repository to another. It means that we will lose the history of manual resolving conflicts which means that every time the repository is cloned, we can have the same conflicts and we have to resolve it another time.

2.2. Data Set Problem

Git is a great tool for text-based files, such as source code, but it is not perfect for data [12]. Although it is not perfect for datasets, Git has already been used as the engine behind a couple of Open Data Ecosystems [13].

2.3. Supporting Adding Blanks

While creating his version, the user can make any modification including adding blanks. After committing his changes, we might have two similar versions with no significant differences. Handling this kind of versions must be reviewed. Detecting unchanged versions is not possible with DVCS like Git.

2.4. Source Code Modification Problem

Some version control systems like Git give the user the ability to add some changes. “GIT Hooks provide an easy way for everyone to extend git to their needs”[14]. Git Hooks are a kind of triggers called while performing a git action. To implement a hook, all what you have to do is to write a code in order to be executed while proceeding the git action. The problem with hooks is that they are not automatically copied from a repository to another which means that the user have to reinstall all his hooks if he loose his repository.

A possible solution to adapt the tool to the user needs is to modify some functionality in the source code of the software which could affect other functionalities so we no longer have the same efficient VCS.

In other terms, we do not have the complete control on the whole software artifacts to modify and adapt them to the wanted context.

Furthermore, adding some functionality to ameliorate the tool capabilities is no longer safe because we can’t predict the consequences on the version control system. Non predicted effects of changing software functionalities could deal with its basic VCS SCM capabilities.

In research context, we find several examples of research-oriented platforms enabling collaboration between scientists.

Colwiz is “an online platform offering secure spaces for collaboration, discussions and sharing materials” [16]. To collaborate with others on this platform, you have to be a member, create a new group and invite the wanted collaborators. The collaboration working space is the group home page and any contribution appears to the group as a new post where a member can add his thoughts, publications, files etc.

Another research platform, MediaWiki, has developed extra packages for collaboration. WorkingWiki is a software extension for the platform developed principally for research groups. “It integrates editing and running code with wiki editing. It operates on source files that are stored in standard wiki pages“[17].

To overcome the difficulties of distance research collaboration, Web-based platforms like Laboratree offer the possibility to scientists to exchange precious knowledge.

It is a secure virtual research environment for project communication, document and data sharing, and real-time informal exchanges of ideas.

Laboratree accomplishes document and data sharing with a drag-and-drop folder-based document tree. This feature includes permission-based document sharing, checkout for review and editing, full version control and tracking, and a user configurable metadata system to identify files [18].

All mentioned platforms allow their users to share the research activities’ source code and data. They facilitate the dissemination of the research project and offer the capacity to archive the associated folder.

The problem with these tools is the way to treat the research project. In these platforms, a research project is just as a set of files to manage not like a work in progress.
The research activity version does not contain any information about the results so we can’t have an idea about the project evolution.

Furthermore, the contributors are selected while authorizing the collaboration so we still have a limited collaboration. Nobody outside the research group can have access to the project or even knows about its existence.

These platforms are based on a VCS or inspired from a social network like twitter (posts and comments). Consequently, they form a kind of a social network for scientists enabling them to discuss ideas and share information.

Another limitation of these platforms is that the majority of these tools do not offer the ability to work online on the research project. They “allow viewers to export the source code to their workstations and work on it offline” [17].

Consequently, the efficiency of using one of the existing VCSs or even an existing platform is no more granted.

To manage research project, we have to build up an efficient tool of collaboration and find more adapted solution.

3. Proposed Solution in Run&Share Platform: Current Work

To overcome existing VCS problems and manage versions, we propose a solution based on the version control capabilities but aiming to improve several functionalities and adapt that to the research context and environment. This solution is implemented to manage collaboration for Run&Share (R&S) platform. Run&Share is a cloud based platform as a service offering the possibility for researchers to collaborate, simulate, evaluate and reproduce their research. Giving the ability to users to contribute with research projects means providing them with a versioning system to manage and display the collaborators’ contributions.

The concept of a research project version in Run&Share is different from other platforms or SCMs. It is based on the needs of scientists and adapted to the research process. The version in this platform includes information about the research production, the authors, the code and the results of the project simulation on the platform. The following component diagram describes the general architecture of Run&Share versioning system.

Run&Share’s versioning system is based on different levels of abstraction.

The first level is the collector. It is designed to collect information about the work of each contributor from the frontal web, like the original research project, a description of the contribution, the proposed source code, the input parameters and so on. In this step, any modification made by any collaborator is kept and stored in a temporary log file in the working directory.

![Proposed versioning System Architecture](image)
Using the generated log files, the retriever extracts the information and generates formatted files (XML format). These files will contain all what we need to make a new version and especially information about the modification and the collaborator.

The formatted files will be useful to make sure that the new version is a real contribution not just a copy of the original version so that we avoid having unchanged versions. It will be also important to reload the version context if we want to use them to switch from a version to another. Consequently, an RMC user can clone any authorized research project and make his changes. If he wants that his contribution will be counted in the research project evolution, he has to make real changes.

The retriever stores the generated files to be used in the following levels. The storage will be in a user directory. In this directory, we will store all the needed information about the user changes so we can give him the possibility to go back to his old version and provide him with a summary of his project evolution. Thus, we have two levels of version control. Managing the user personal versions and controlling the research project versions. The first level is dedicated to manage the user changes in his own working space so he can easily make modifications and go back to the versions required. The second level is dedicated to track the research project evolution. It focuses on the user’s real contributions.
Using the generated standardized files, the third component, the recorder, generates a matching object with the structure of a version which will be stored in a relational database dedicated to store versions of different contributions. In this level, we can avoid merge conflicts and grant pseudo independency between versions without losing link between them. This kind of independency will enable users to manage their versions or their original projects independently. Users can make changes in the previous version without being obliged to switch to this version. Having such independency will enable researchers or scientists to compare different versions of the project. They do not have to make extra effort to recreate another project with a lot of similarities with the current project.

Different distant users can start working in their remote repositories from an initial version without caring about other’s changes. When they finish their modifications, their versions will be merged as new versions referring the initial version.

An adapted and practical user interface is provided by the forth component to allow Run&Share platform users to display the evolution of their research projects. This architecture provides basic VCS capacities, but it offers more simplicity and the clarity of each component role.

The goal of this solution is to adapt the version control to research context. It is possible to develop this solution’s functionalities and make it adapted to build up a total traceability system.

We mean by total traceability the fact of tracking all users’ modifications. In fact, any change committed by user could be logged and stored the same way like the version in order to keep a trace of research project contributions.

Keeping a trace of all changes made by the researcher is easier using such architecture. We just need to collect these changes and treat it like we treat versions of contributions.
Figure 9. Search for Project Versions

Figure 10. View Project Versions.
Figure 11. Edit Project.

Figure 12. View New Versions of the Project.
Figure 13. *Clone Project.*

We can also clone a version.

Figure 14. *Clone Result.*
Using this architecture, adding or improving some functionalities is possible and safe because having such kind of multi-level architecture facilitates the control and the evolution of each part and consequently the maintenance becomes more efficient. R&S versioning system aims to enable researchers to have a benchmark of versions, inputs and outputs in order to contribute with the research project reproducibility and facilitate to encourage users to help with the scientific evolution.

4. Conclusion

In this paper, we have discussed the concept of collaboration in the research field. We have studied the efficiency of using existing software collaboration tool referring to Git, the most well known and the widely used DVCS. Then, we have introduced a research oriented collaboration tool enabling users to contribute with research production process which is Run&Share platform.

Run&Share comes with a new concept of a version seen with the eyes of a researcher and adapted to the research context. It facilitates the collaborators’ contributions by giving them an idea of the evolution of a research project. This platform provides its users with the ability to have information about the project simulation which means that it provide its users with a benchmark of inputs which could be useful for similar researchs, a benchmark of outputs which is important to see the evolution of the research project and simulation capacities helping them to test, evaluate and validate their research. The proposed solution of collaboration in this platform offers the possibility to be improved in order to have total traceability including any action made by the user during using the platform services.

Acknowledgements

We would like to thank the CNRS, the University of Orleans, and EFREI for their valuable support.

References