RESEARCH ARTICLE OPEN ACCESS

Traffical User Control System

[1]P.Sakila, [2]J.Iswarya

[1] Assistant professor, Dept.of.Computer science, Ponnaiyah Ramajayam institute of Science and Technology (Prist) Thanjavur

[2] Research Scholar, Dept.of.Computer science, Ponnaiyah Ramajayam institute of Science and Technology (Prist) Thanjavur

Abstract:

In this paper new adaptation control is an important tool for shield of date and collaboration between colleagues. New disseminated version control systems are increasing increasingly in style as successors to central systems like CVS and Subversion. Graphical user interfaces (GUIs) make it easier to interact with version control systems, but GUIs for distributed systems are still few and less mature than available for centralized systems. The purpose of this thesis was to propose specific GUI ideas to make distributed systems more accessible. A custom study was using some software engineers. Process

Introduction

That the vision is control some all software projects of adequate size and involving numerous developers today tend to. The purpose of a version control system is to act as a repository for the project data along with a complete development history, enabling any state of The VCS also helps developers, transparently allowing people to work on different parts of the project simultaneously. Currently most version control systems are centralized, with a single server as the common hub of all developers' clients. The server stores the single copy of the entire development history while clients check out a single state from the server, modify the data and then commit the result back to the server.

Repositories can then be synchronized, allowing changes made to one repository to be applied to another. Whereas centralized systems are always hub–based client–server networks, distributed systems are peer–to– peer based and have no distinction between client and server. They therefore do not have to adhere to any specific design when it comes to data flow. With the additional freedom of choice granted by distributed version control systems, new types of work flow become possible. However, this also risks making distributed systems more difficult to learn for new users. For many users, well-designed graphical user interfaces provide a quicker and more intuitive means of learning and working with systems, but distributed systems are still relatively new and few graphical frontends are available. By comparison, centralized systems have been around for much longer and have many available frontends as well as tools for integration with virtually any development environment.

Purpose of methods:

The purpose of this thesis is to propose graphical user interface techniques appropriate for making the features of distributed version control systems accessible to users. A DVCS can require new ways of thinking compared to centralized systems, therefore it is likely that a graphical user interface for such a system would benefit from corresponding new kinds of visualization and interaction. This thesis aims to investigate and define user

interface concepts that fulfill this requirement, in order to serve as a brief guide to developers wishing to develop graphical frontends.

The primary input for this thesis comes from analyzing existing version control systems, graphical frontends and usage patterns. The latter is largely based on a small survey conducted at Opera Software, aimed at discovering the areas of user interaction with the greatest need for simplification and/or visual enrichment through graphical user interfaces. Based on the survey results, a number of graphical user interface ideas are proposed that may help alleviate the identified problems. A brief evaluation of these proposals' effectiveness is also carried out by analyzing them with regards to the original problems and established user interface design guidelines.

By being aware of earlier revisions, version control also makes it possible for several people to work on the same file. Each of their changes can be recorded in the history and in many cases merged automatically – the system determines how all of the changes can be included, even though they were made independently from each other.

Other features

Current version control systems, including but not limited to distributed systems, often support modern technologies like Unicode and tunneling. Unicode support typically means the VCS can perform doffing on Unicode files and support Unicode characters in commit metadata, while tunneling is usually the means to communicate between peers (or client/server) through existing established network protocols like HTTP and SSH. As technology advances, so do expectations on applications. Many modern version control systems therefore have very similar feature sets in this area, possibly in part due to not wanting to appear "behind the times" by not supporting established technologies

As a part of this thesis a version control system usage survey was carried out. The survey had multiple goals:

1. Determine which aspects of centralized version control systems are in greatest need of improvement in terms of usability, and which tools and frontends are being used to Supplement them.
2. Determine which distributed systems are most popular.
3.Determine which aspects of distributed version control systems are in greatest need of improvement in terms of usability, and gather ideas and suggestions on how such improvements could be achieved via a graphical frontend.

Conclusion

Based on the above summary and evaluation, we find that the proposed features are appropriate for the identified problem areas and should help alleviate the problems experienced by users.

Since the survey results easily identified several problem areas with current distributed systems, more work should go into developing usable graphical frontends for these systems, as it will likely improve the chances of users successfully migrating from older centralized systems to newer distributed systems.

The features proposed in this report should constitute a good starting point for developing a working prototype frontend, which can then be evaluated by users and refined into a final product. In addition to the main proposed features, the minor features mentioned may also warrant closer examination when making an actual prototype.

References

[1] Rochkind, Marc J. (1975). The source code control system. In: IEEE Transactions on Software Engineering SE-1(4):364–370.

[2] Bazaar Benchmarking Results (2007)

[3] Berliner, Brian (1990). CVS II: Parallelizing software development. In: Proceedings of the USENIX Winter 1990 Technical Conference, pp. 341–352. USENIX Association, Berkeley, CA, USA.

[4] BitTorrent Introduction (2008

[5] Roundy, David (2005). Darcs: distributed version management in haskell. In: Haskell '05: Proceedings of the 2005 ACM SIGPLAN workshop on Haskell, pp. 1–4. ACM Press, New York, NY, USA. ISBN 1-59593-071-X.