Svn Obliterate - Contract Proposal

Table of Contents

Introduction

This document sets out a proposal for an agreement between a contractor and one or more sponsors, whereby the contractor will undertake to define, produce and deliver a Subversion feature known as “Obliterate” for and on behalf of the sponsor(s), conducting the work with the full involvement of the Subversion development community.

The Obliterate Feature

The facility to remove selected parts of history from a Subversion repository is known as the “Obliterate” feature.

The Obliterate feature has long been requested but not yet adequately specified or developed. The existing sponsors of Subversion development and also the volunteer contributors have thus far concentrated their efforts on other areas.

The reasons for wanting the feature can be classified broadly into the following two areas:

> Hiding secret or sensitive data that was accidentally committed to a location where it is visible to an inappropriate audience.
>
> Trimming down the repository size by removing data that has become obsolete or data that was committed accidentally.

One notable case of the former is where some sensitive text is inserted by accident into one revision of a long-standing and publicly visible file, and removed in another revision shortly afterwards, and the administrator wishes to permanently hide that piece of text in the repository's history without

hiding the whole file. One notable case of the latter is avoiding a boundlessly increasing repository size when frequently committing new revisions of large files of which only the most recent revision or few revisions are ever needed.

Goals of the Project

The purpose of the project is to define and to provide the “Obliterate” feature, or such parts of the feature as may be determined to be worth developing at this time.

The major goals of the project are:

To produce an “Obliterate” feature that satisfies the requirements of the sponsor(s) and is accepted by the Subversion community.

To define the feature's purpose and scope.

To design the feature.

To implement and test the feature.

To deliver the feature in suitable form(s).

To conduct the whole process in open participation with the Subversion community.

To involve the sponsor(s) at all stages with discussion of options, progress reports, review and testing.

Project Plan

Tasks

The tasks required are listed in their approximate sequence.

Each should specify: work required, deliverables, acceptance criteria, time & effort estimates.

Tasks:

User requirements:

Review similar capabilities in other VCSs

Summarize the archived ML and issue tracker discussions.

Gather and document the use cases that have been reported.

Write user-level requirements, by analysing use cases.

Architecture; scope; phases:

Decide on the architecture of the solution: e.g. whether it will be off-line or on-line (admin task) or on-line (client side).

Plan a phased progression of functionality with increasing scope.

Decide which phases are to be within the project. Specify the acceptance criteria for each phase.

Write functional requirements (requirements on software behaviour) for at least the early phases of functionality, from the user-level requirements and the architecture, covering both client and server sides.

Client-side behaviour, especially w.r.t. working copies:

Design the first phase of client-side behaviour, from the functional requirements.

Design a simple UI for the client-side behaviour, against which tests can be written.

Show how the client-side behaviour satisfies the use cases.

Write tests for the client-side behaviour, from the functional requirements.

Repository-transformation algorithm:

Design the repo-transformation algorithm against the functional requirements. Show how it satisfies the use cases.

Write tests for the repo-transformation algorithm.

Implement the repo-transformation algorithm as an off-line prototype, e.g. within "svnsync". Run the repo-transformation tests against it, and refine the algorithm.

Client-side behaviour:

Test the client-side behaviour on (unaware) working copies, by manually running the off-line prototype on the server. (See also some previous tests in the email archives.)

Revise the client-side behaviour (requirements, design and tests) in light of the findings.

Implement new client-side behaviour as required.

Test the new client-side behaviour, using the automated tests on the client and the off-line prototype on the server.

Design and implement on-line obliterate on the server (if required). Test with the same tests that were used earlier.

Design and implement client-side invocation (if required). Update and use the client-side tests.

Mile-stones

The following mile-stones are defined. This is the initial list, and will be reviewed and updated for mile-stone 2.

A mile-stone is reached when the documents mentioned are satisfactory to the sponsor, the contractor and the community, and when the functionality mentioned is tested, documented and integrated into the trunk.

MS0: Project plan agreed.

MS1: User requirements:

collected use cases;

user-level requirements.

MS2: Shape of project:

architecture;

phases of functionality and their acceptance criteria.

MS3: First phase of client-side behaviour:

functional requirements for this;

design, UI and use cases;

(tests are not required yet).

MS4: Repo-transformation prototype:

Algorithm documented, implemented, tested.

MS5: Behaviour of old clients reviewed; new client-side behaviour tested.

MS6 (if required): On-line server-side obliterate tested.

MS7 (if required): On-line client-side obliterate tested.

MS8 (etc., if required): Further phases as defined at MS2.

Integration

Integration with developments on trunk shall be done continuously, whether on trunk or on a branch kept up to date with trunk.

If working on a branch, work shall be merged into trunk at least at every defined milestone.

Time Estimates

A time and effort estimate shall be agreed for each mile-stone.

Notes

This section contains some thoughts that I have not yet put into a proper place in this proposal.

Outline Designs

The high-level design could go in different directions. It may well be worth outlining 2 or 3 different approaches, to the level of:

what protocols affected

what locking required

example API

example UI

what requirements & UC's fulfilled

what work required, in what areas

Planning

Impl. plan

Testing plan (API & UI tests to be automatic)

Documentation plan

Detailed design task should include:

API design & tests

UI design & tests

Initial documentation

Implementation & Testing task should contain:

Developer documentation

All code and documentation shall be in the main Subversion repository (maybe on a branch) unless in an agreed other repository (e.g. the Svn Book).

Delivery

Documentation

Provide adequate developer documentation, in the source tree or other suitable place.

Add suitable user documentation in “The Book” at <svnbook.red-bean.com>.

Provide trial versions

when functionality is available to be tested

as packaged source code

Provide a release according to the Subversion community's release procedures

ensure it will be included in the next official “minor” release

if the community's release schedule makes the expected official release date too far off, make a custom release

---

Terms of Contract

Co-operation and Time Limits

Specify time scales & recourse if not met, for:

Completion of each task...

Reaching consensus with community (on ML)...

Reaching agreement between sponsor and contractor...

Communication:

Contractor will provide progress reports to the community and to the sponsor, at least once a week.

Sponsor and contractor will respond to queries from the other by end of next working day.

Sponsor's sign-off of a task...

What if acceptance criteria not met?

Remuneration

Costs shall be based on the person-days of effort estimated and agreed before the start of each phase, at a rate to be agreed, regardless of the actual effort expended. An estimated completion date shall be agreed for each phase, but the actual completion date shall not affect the cost.

Expenses incurred by the contractor to meet specific requirements of the sponsor(s) shall be agreed in advance and billed at cost. Examples include travel to meet the sponsor, and hardware and software for system-specific testing.

Intellectual Property Rights

The sponsor(s) will execute the appropriate CLA for work completed.

Employer-employee Relationship

The Sponsor shall employ the Contractor in the capacity of a subcontractor. The terms shall be non-exclusive in respects that: the Contractor may engage in other work at the same time, for the same or other employers, on the same or other projects; the Sponsor may employ other contractors or staff to work on the same or other projects, in co-operation or separately.

The contractor undertakes to ensure that the work needed is done, not necessarily to do it all in person, and may sub-contract, use the work of volunteers, and use existing free software, to any extent.

Glossary

CLA – Contributor License Agreement – see <http://subversion.org/legal/individual-cla.html>.

Contractor – an individual undertaking the development of the Feature.

Feature – the Subversion software feature to be defined and developed in this project.

ML – Mailing List.

Obliterate – a common name for the Feature and also for many possible subsets, supersets and variants of the Feature.

Sponsor(s) – one or more companies or persons funding the development of the Feature.

Subversion – a Version Control System – see <http://subversion.tigris.org>.

UC – Use Case.

UI – User Interface.

About the Contractor

Contact details

Name: Julian Foad

Email: <julianfoad@btopenworld.com>

Qualifications & experience

Subversion committer on core C code for over 5 years as a volunteer, 11 months as an employee of CollabNet.

See my CV for further details: <http://www.foad.me.uk/career/JulianFoadCV.pdf>.