Proposal for improvement of Svn Tools

Marco Corvo - XI SuperB workshop

December 2, 2009

Svn Tools today

Svn Tools is currently made of a bunch of perl scripts whose main goal is to:

- manage the interaction with Subversion

Svn Tools today

Svn Tools is currently made of a bunch of perl scripts whose main goal is to:

- manage the interaction with Subversion
- bootstrap a working area (namely with *sbnewrel*)

Svn Tools today

Svn Tools is currently made of a bunch of perl scripts whose main goal is to:

- manage the interaction with Subversion
- bootstrap a working area (namely with *sbnewrel*)

In particular we have scripts that:

- create a private working area based on a given software release

Svn Tools today

Svn Tools is currently made of a bunch of perl scripts whose main goal is to:

- manage the interaction with Subversion
- bootstrap a working area (namely with *sbnewrel*)

In particular we have scripts that:

- create a private working area based on a given software release
- add specific packages to a private working area

Svn Tools today

Svn Tools is currently made of a bunch of perl scripts whose main goal is to:

- manage the interaction with Subversion
- bootstrap a working area (namely with *sbnewrel*)

In particular we have scripts that:

- create a private working area based on a given software release
- add specific packages to a private working area
- create *tags* or *usertags*

Svn Tools today

Svn Tools is currently made of a bunch of perl scripts whose main goal is to:

- manage the interaction with Subversion
- bootstrap a working area (namely with *sbnewrel*)

In particular we have scripts that:

- create a private working area based on a given software release
- add specific packages to a private working area
- create *tags* or *usertags*
- switch to different Subversion directories

Svn Tools today

Svn Tools is currently made of a bunch of perl scripts whose main goal is to:

- manage the interaction with Subversion
- bootstrap a working area (namely with *sbnewrel*)

In particular we have scripts that:

- create a private working area based on a given software release
- add specific packages to a private working area
- create *tags* or *usertags*
- switch to different Subversion directories
- print some useful info about packages

Svn Tools limits

Based on my personal experience, I could summarize my impressions saying that:

- most of the functions are re-written several times

Svn Tools limits

Based on my personal experience, I could summarize my impressions saying that:

- most of the functions are re-written several times
- interaction with Subversion is made through *system* calls and not api.

Svn Tools limits

Based on my personal experience, I could summarize my impressions saying that:

- most of the functions are re-written several times
- interaction with Subversion is made through *system* calls and not api.
- there are doubles of scripts, like e.g. **newrel** and **sbnewrel**

Svn Tools limits

Based on my personal experience, I could summarize my impressions saying that:

- most of the functions are re-written several times
- interaction with Subversion is made through *system* calls and not api.
- there are doubles of scripts, like e.g. **newrel** and **sbnewrel**
- IMHO looks like they've been written just as needs rose, that is without a coherent view of the functionalities

Evolution

Svn Tools is the principal front end for users managing packages in Subversion repos, this is why scripts should:

- be simple

Evolution

Svn Tools is the principal front end for users managing packages in Subversion repos, this is why scripts should:

- be simple
- provide a full interface to Subversion functionalities

Evolution

Svn Tools is the principal front end for users managing packages in Subversion repos, this is why scripts should:

- be simple
- provide a full interface to Subversion functionalities
- be well organized

Evolution

Svn Tools is the principal front end for users managing packages in Subversion repos, this is why scripts should:

- be simple
- provide a full interface to Subversion functionalities
- be well organized
- be factorized

Proposals

- Subversion comes with APIs written both in perl and in python.

Proposals

- Subversion comes with APIs written both in perl and in python.
- After a look at perl APIs I realized that they lack at least a couple of the most useful things: documentation and working examples.

Proposals

- Subversion comes with APIs written both in perl and in python.
- After a look at perl APIs I realized that they lack at least a couple of the most useful things: documentation and working examples.
- This is why I tried to look at python APIs. I personally have no strong enough technical arguments against perl, but in general I feel more comfortable with python.

Proposals

- Subversion comes with APIs written both in perl and in python.
- After a look at perl APIs I realized that they lack at least a couple of the most useful things: documentation and working examples.
- This is why I tried to look at python APIs. I personally have no strong enough technical arguments against perl, but in general I feel more comfortable with python.
- Another argument is that we are trying to evaluate a new build system for SuperB, that is SCons, and SCons is a python tool. This would push in the direction of a limited number of languages used for software development.

cont'd

Besides a great improvement would be to factorize common functions into modules. This would allow:

- a clear specification of the user interface

cont'd

Besides a great improvement would be to factorize common functions into modules. This would allow:

- a clear specification of the user interface
- the possibility to add new functionalities in a straight forward way reusing existing modules

cont'd

Besides a great improvement would be to factorize common functions into modules. This would allow:

- a clear specification of the user interface
- the possibility to add new functionalities in a straight forward way reusing existing modules
- e redefinition of the interaction of scripts with Subversion which can minimize the client-server communication

Users' contribution

A great contribution could come also from users' experience and use cases.

Users' contribution

A great contribution could come also from users' experience and use cases.
These simple considerations came out just with a 'almost blind' usage of Svn Tools so an extensive experience of many users could help finding out other improvements and use case definitions.