Novo wrote on Tue, 12 January 2010 07:07
After a short research I chose "monotone".

Pros:
1) It is just one executable (plus a couple of dlls). You don't really need to install it.
2) It is quite powerful. Actually, GIT borrowed a lot of ideas from monothone.
3) It is easy to use. Monotone has not that big set of commands.
4) Repository and workspace can be located in different places. (You can have different workplaces for the same branch. Actually, I'm using this for a different purpose.)

I like your writing mistake about pros and cons, IMO your pros are cons .
1) How extensible is it? If we want to create a hook for some operation or replace one part with our i.e. diff?
2) ... (the counterparts are too) ...
3) Easy to use not always means less commands, monotone stay in my way because it's doing simple things complicated. I've tried once to help pidgin development and implement some features which i needed but I've hit monotone wall and I quited. A scm should not stay in your way, it should be expressive enough not to impose limits but not too expressive to let you doing same thing in thousand different ways.
4) IMO it just complicate my life

Novo wrote
The only thing I'm missing about monotone is GUI, which can be easily developed using U++, because monotone stores repository in SQLITE database.

While others want a GUI on top of scm tool I want it to be integrated with my tools. I.e. I create project in theIDE (or other ide), i'm adding new source files, do some codding, then i initialize a repository (from theIDE) and do initial commit. Supposing that i have a working program and want to test new stuff. I'm creating a branch (from theIDE AnySCM->NewBranch), theIDE reloads all opened files and these contain data from new branch, note that i've not changed project location, so the branch reside in same directory and i may switch to other branch if i want and that's cool because i don't have to create a different package in theIDE just to try new things (i'm reusing the same interface, just switching from one branch to another). I'm doing some modification to this branch, i'm testing changes and if it's ok i'm merging it in master branch
(AnySCM->Merge<branch-name>).
IMO that's a nice way to get stuff done.
Novo wrote
1) GIT keeps repository and workspace in the same directory. In order to checkout another branch you need to clone repository. Basically, this means that you cannot store several projects in one repository. This is not a problem with monotone. In my case "branch" is often equal to "project".
2) U++ is a set of "packages". When I create a new project I want to assemble it from several "packages". This is possible with GIT, but in this case each package should be represented as a separate GIT repository. This seems to be way too complicated.

I'd like to get more feedback about work-flows with distributed VCS from you guys.

1) I think you don't have to clone the repo to checkout another branch. You do the checkout and reuse the same space and if you want to switch to master branch just checkout it.

```
~$ git branch
  master *
  your-cool-branch
~$ git checkout your-cool-branch
# now where your project is, you have your-cool-branch source code in place
# working...
~$ git commit -a
~$ git checkout master
# yay, we are on master branch on the same location
# and i think that's cool
```

2) Not every package should be a separate repository, i would make assembly(nest?) a repository, perhaps only for MyApps i wouldn't do that because packages from it may not be related one to other.

Just my 2 cents.

I think we should consider others dvcs for scm but IMO the leading scm which will be used for a long time from now will be git, hg and bzr.
Git - surfer from code fragmentation(shell code, perl, python, C, tcl, etc.), making it a bit unportable.
Bzr - a bit slow, it's written in python but cross-platform is achieved.
Hg - i don't have an opinion, i've used it only once (tried the tutorial) so no real life experience with it.