How to develop in this repository ?

Organization of the repository

Projects

The repository is divided into several more or less independent projects. Of course, there are dependencies between projects but each project can have its own managers, access right, versions, etc...
For example, anatomist, aims, axon, morphologist, brainrat, cortical_surface... are projects.
The projects are the first level of directory in the repository in the directory https://bioproj.extra.cea.fr/neurosvn/brainvisa.

Components

A project may be divided into several components. This is the second level of directories in the repository. It is optional, some projects like *axon* for example, are not divided into subparts. The idea is that sooner or later, components will become independently downloadable packages.
Components are used in the following cases:

License

Several components are often used to separate parts of the projects that have not the same **license**. For example, a project can have a part that have a GPL license because it uses GPL libraries, and another part that is less constrained because it does not use any GPL software.

For example, anatomist project has 2 components:
- *anatomist-free*: BSD-like license, less restrictive than GPL
- *anatomist-gpl*: this part uses PyQt library, Python bindings of Qt library, which is under the GPL license.
Other projects have a *private* component because this part is not open source. For example *morphologist/morphologist-private*.

Sub-projects

The components can also divide the project into several sub-projects according topics or target audience.

For example, the project *morphologist* has a sub-project baby which is related to T1 MRI processing but specifically for baby brain images. So the topic is a bit different and it is a distinct toolbox in BrainVISA than the regular Morphologist toolbox. Moreover this sub-project is only internal for the moment and is not distributed on the web.
Another example is the component *freesurfer_plugin* in *cortical_surface* project which contains a BrainVISA toolbox enabling to combine freesurfer analysis and Morphologist analysis. This toolbox is distinct from the Cortical Surface toolbox developed by the LSIS lab.

Versions

In each component directory, there are version directories. There is always at least a *trunk* directory which contains the main (development) version of the source files.
In addition to the trunk directory, you may find the following directories:
- *branches/X.Y* This is what we call a *bug_fix* or *stable* branch. It is a copy of the trunk branch at a time it is stable when we want to release new packages with this version. This branch can be modified but only with bug fixes, not with big new features.
- *tags/X.Y.Z* This is what we call a *release* branch or a patch version or a tag. It is a copy of the X.Y branch at the time we create a new release package. This copy should not be modified. It is saved only for history to be able to retrieve the state of the source files in any X.Y.Z version.

Developing in existing projects

Getting the sources

To develop in the repository, you first have to get the sources and compile them. In general, we do not get the whole repository with all branches because it is too big and it is not useful to have a copy of all the repository.
So the first question you have to ask yourself is: **which version of the source do I need ?**

A few rules to answer this question:

- **New features should be developed in the trunk version**. This version is not for end users and is not packaged. To create new packages, we should first create a new bug_fix branch from the trunk branch and it corresponds to a major or minor version change.

- **Small modifications and bug fixes can be developed in the bug_fix branch**. The modifications will be take into account in the next package created from the bug_fix branch. It corresponds to a patch version change.

- **Modifications in the bug_fix branch should be reported in the trunk branch.**

-> In conclusion, you should get the sources of the trunk and bug_fix branches of at least the project in which you develop. For the other projects of the repository, if you just need to compile them, you could get only the bug_fix branch.

To know how to get the sources and compile, see the wiki page about how to setup a development environment for BrainVISA projects.

Committing modifications

To commit modifications in the repository:

- **Modify** the files in your local copy of the repository. Use the *svn add* command if you need to create new files or directories.
- Use *bv_maker configure build* to **update the build** directory according to the modifications. This step is optional if you only modify existing python files on a Unix system. Indeed, python files are not compiled and on Unix the python files are not copied in the build directory but only symlinked.
- **Test** your modifications by running the program from your build directory.
- **Check** the modified files with *svn status* command and possibly with *svn diff* to be sure that you will commit only the modification you intented to.
- **Commit** the modification using *svn commit* and enter a comment describing as precisely as possible the modification in english. If the modification is related to an issue in Bioproj, indicate the reference to the issue in the comment with #issue_number. The commit will send the modifications to svn server, so it will be available for other users of the repository.

Be careful, *svn* commands can be used only in a local copy of the repository. If you have several projects and/or several versions of projects, you have in fact several local copies of different parts of the repository. The local copies are under the *trunk* or *branches/x.y* directory of each projects.
So, if you need to do modifications in several projects, it is not possible to do it in one commit command, you will have to run the *svn commit* command in each project.

Using Bioproj issues

Bioproj interface offers practical features to manage issues. It is possible to enter a new issue in a specific project via the menu *New issue*. The issue can report a bug or ask for a new feature. It is useful to keep a track of the problems that have to be fixed and to have a well organised history of the modifications of the repository. Indeed, it is possible to associate subversion commits and Bioproj issues. This enables to easily retrieve all the modications in the repository related to a specific bug fix or feature development.

Each Bioproj issue has a number that can be used to reference this issue using the following syntax: ***#issue_number***.

When committing a modification in subversion repository, it is possible to add a reference to an issue writing *#issue_number* in the comment. This indication will be replaced by a link to the issue in Bioproj when visualizing the revision comment in Bioproj interface. And a link to the revision will be automatically added in the issue's page in Bioproj interface.

It is also possible to automatically close an issue via a commit in the repository by indicating in the comment ***Fixes #issue_number*** or ***Closes #issue_number***.

For example, in this wiki page, the code #4280 generates the following link to #4280 issue.

In the same way, it is possible to reference revisions with the code r<revision_number>, for example r46535.

Reporting modifications from one branch to another (merging)

In case of a modification in the *bug_fix* branch, it should generally be reported in the *trunk* branch. **Do not do this manually**, you would risk forgetting to report some modifications and so introducing problems in the trunk branch.

The command ***svn merge*** enable to easily report modification from one branch to another. In addition, we develop a script ***bv_check_svn_merge*** which checks if there are modifications in the a bug_fix branch that have not been reported in the trunk branch. In this case, the script also shows the commands that should be run to merge the branches: a command *svn merge* that merges the server bug_fix branch and your local copy of trunk branch, and then a *svn commit* command to submit the merge modifications in the trunk branch. This script is available in *development/brainvisa-svn* project.

Be careful: before merging, use *svn status* to check that you do not have uncommitted modifications in your local copy of the trunk branch. You would risk committing them with the merge by mistake.

It is also possible to merge a modification from the *trunk* branch to the *bug_fix* branch but we try to avoid that because the merges in subversion become painful when merges have been done in both directions (*bug_fix* -> *trunk* and *trunk* -> *bug_fix*), and all changes in the *trunk* branch should not be reported to *bug_fix*. **We strongly recommend to merge branches only in one direction: *bug_fix* -> *trunk***.

Creating new branches

Distributed packages are created from the bug_fix branch of the sources. So, when we want to integrate new features developped in the trunk to the next release of Brainvisa, we create a new bug_fix branch. To ease the creation of such branches, we create the scripts *bv_create_branch*. See this wiki page for details.

Creating a new project

If you wish, it is possible to create a new project in the Brainvisa repository to host your development. In this case, ask it to one of the managers of the repository. Indeed, admin rights are needed to create a new directory at the root of Brainvisa repository.

It is also possible to create a new Bioproj project associated to the new subversion directory but it is not mandatory.

Access rights to the new project can be customized if needed.

To be able to compile the new project with brainvisa-cmake, it is necessary to add CMake configuration files to the project. See the following wiki page for more information about this.

Distribution policy

Development hints & tips