Managing code checkout of large projects

This topic describes the differences in storing your source code in a standard build directory and a detached source directory when generating a snapshot.

Definitions

When you generate a new snapshot of a project, a new snapshot directory is created as a subdirectory of the `odasa/projects` directory. This subdirectory contains a `snapshot` file that defines the location of the source code for the version you wish to analyze, in addition to the build commands, which are used when the snapshot database is generated. A copy of the source code may be checked out into one of two locations:

- **Standard source directory**—within each snapshot directory. In this case, each snapshot directory contains a complete version of the source code. With this setup, if you define new analyses for the project then results for historic snapshots can be calculated from the archived source code.
- **Detached source directory**—a single, alternative directory. In this case, the code in the source directory is updated every time a new snapshot is added and built. With this setup, you cannot easily rebuild a historic snapshot because the old version of the source code has been overwritten with a new version. However, this method uses less disk space.

Standard source directories

For most systems, the standard method of storing a complete version of the source code in each snapshot directory is appropriate. You configure the `project` file with a `checkout` element containing a command that defines how to retrieve a full version of the source code. For example:

- `git clone " https://github.com/bagder/curl.git " ${src}`—clone the Git repository for Curl into the standard source location
- `svn co " http://svn.ruby-lang.org/repos/ruby/trunk " ${src}`—checkout the Subversion repository for Ruby into the standard source location.

where `${src}` is automatically expanded to the location of the new snapshot. For example: `/home/example/odasa/projects/Hadoop/revision-2012-December-12--22-12-19/src.`

Detached source directories

You should use a detached source directory in the following situations:

- **Source code can only be built from an absolute path**—you can define a detached source directory in the required location on the file system
- **Disk space is low**—if you define a detached source directory then a single copy of each source code file is stored on the server, so less disk space is required
- **Source code is very large**—if you define a detached source directory then, once you have downloaded the initial repository, only new and updated files need to be retrieved during the snapshot creation.

The project file is defined with a `<checkout>` command that defines how to update the source code to the required version. For example:

- `git pull " https://github.com/bagder/curl.git " ${src}`—pull the latest changes from Git into the detached source directory

- `svn up " http://svn.ruby-lang.org/repos/ruby/trunk " ${src}`—update the detached source directory with the latest changes from Subversion.

where `${src}` is automatically expanded to the detached location for the project. The detached location is stored in the project file using the `<source-location>` element, for example: `<source-location>/home/example/Ruby_source</source-location>`.

Note that when reusing the same source directory for multiple builds, some build systems leave build artifacts in the source directory. These artifacts could potentially be picked up by later builds and then influence the results. In such cases care must be taken to ensure that the source directory is properly cleaned between builds (e.g. with `make clean` or similar commands).