Organizing the support for the TDR software development

R. Stroili

Università degli Studi di Padova & INFN

outline

- requirements
  - re-use of BaBar code
  - start from scratch
- code support
  - source code management
  - gforge
  - release management

requirements

- need to understand the needs for the computing effort of the SuperB project
- two options:
  1) use BaBar software infrastructure and code
     - think seriously about this option
     - most probably it's an overkill
     - this is a temporary solution
       - the SuperB software infrastructure will be quite different from the BaBar one
       - start from scratch
  2) setup a software repository for a distributed community
     - CVS + AFS as in BaBar?
     - need to implement an access and authorization policy

reuse of the BaBar code

- is it flexible enough to accomodate the need for varying quickly part of the code?
  - to build a release it takes O(24) hours?
    - if there're no compilation problems
- if you want to use BaBar simulation you need also the analysis code
- most likely one has to choose a given release and put it on an independent development path

reuse of the BaBar code

- probably the best thing (at least for outsiders) is to have a single place where the release and infrastructure is available and where to work on it
  - not the BaBar distributed environment
  - need to set up for a distributed community
    - accounts
    - disk space
    - services
  - need to have somebody supporting and maintaining the release and all the required packages

start from scratch

- standalone projects for detailed studies at the beginning
  - sometime in the future some of them should be merged
  - try to *impose* some commonalities
- fast and detailed simulation
- if the projects are small it's easy to implement a distributed computing environment
  - *little* code to move around, comile and link
- use tools with good functionalities and not too obsolete
- plan in advance a strategy that brings us to the final architecture
- what is done now should go in the final software environment
  - at least what will continue to be of interest

source code management

- use Subversion (svn)
  - close to CVS
  - fixes some problems
    - i.e. client-server mechanism for distributed access
      - uses apache as a remote server
  - atomic commits
  - supports binary diffs
  - used for major projects
    - apache
    - KDE
    - gnome
    - ... many others
  - actively supported

source code management

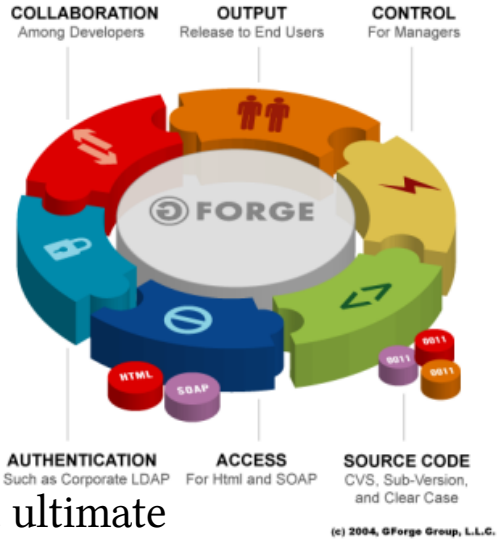
- svn has a different philosophy with tags and branches with respect to cvs
  - more flexible?
- available on
  - many Linux distributions
  - MacOSX 10.4
  - windows (via cygwin)
  - binaries also for Solaris, HP and xxxBSD
- migration from cvs should be quite easy
  - as reported by the ROOT team
- the repository doesn't need to be on AFS

release management

- Subversion is only a first step toward a code management system
  - in BaBar we use SRT (Software Release Tool), HyperNews, ...
- in Padova we are investigating gforge as a possible tool to manage the SuperB software

gforge

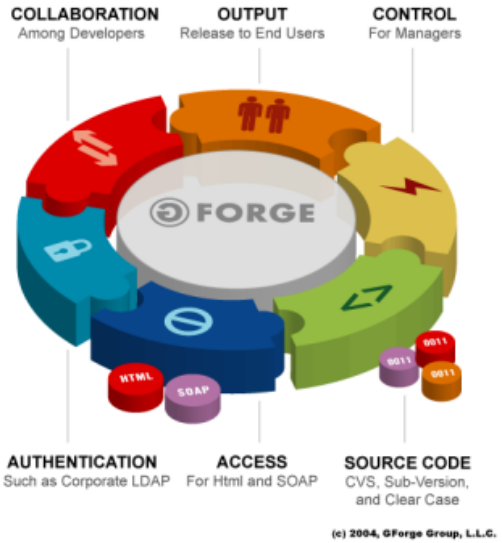
- gforge is a collaborative development software
  - Unlimited numbers of projects, each with its own
    - CVS or Subversion repository automatically created
    - Access control to repositories
    - Statistics gathering from repositories
    - Automatically send diffs to mailing lists on commit with cvssyncmail and svncommitemail plugins
    - Mailing lists
    - Forums
      - ability to respond via email
      - multiple views, including nested, flat, threaded, and ultimate
    - Trackers
      - ability to respond to tracker items via email
      - ability to link CVS and SVN commits to specific tracker and task items with cvstracker and svntracker plugins
      - unlimited numbers of *custom* fields, with unlimited elements in each field
      - saved power queries and quick browse filters



gforge

- Role-based access controls allow for easy permission setting for members of projects. Entire projects or portions of projects may be set to private to control access.
- Task managers
  - powerful MS Project Integration (Enterprise CDE only)
  - Gantt charting
- Document management with approval queue
- Surveys
- News
- File Release System
  - tgz, rpm, ...?
- Wiki

- Command Line Interface
- Tinderbox Integration



gforge

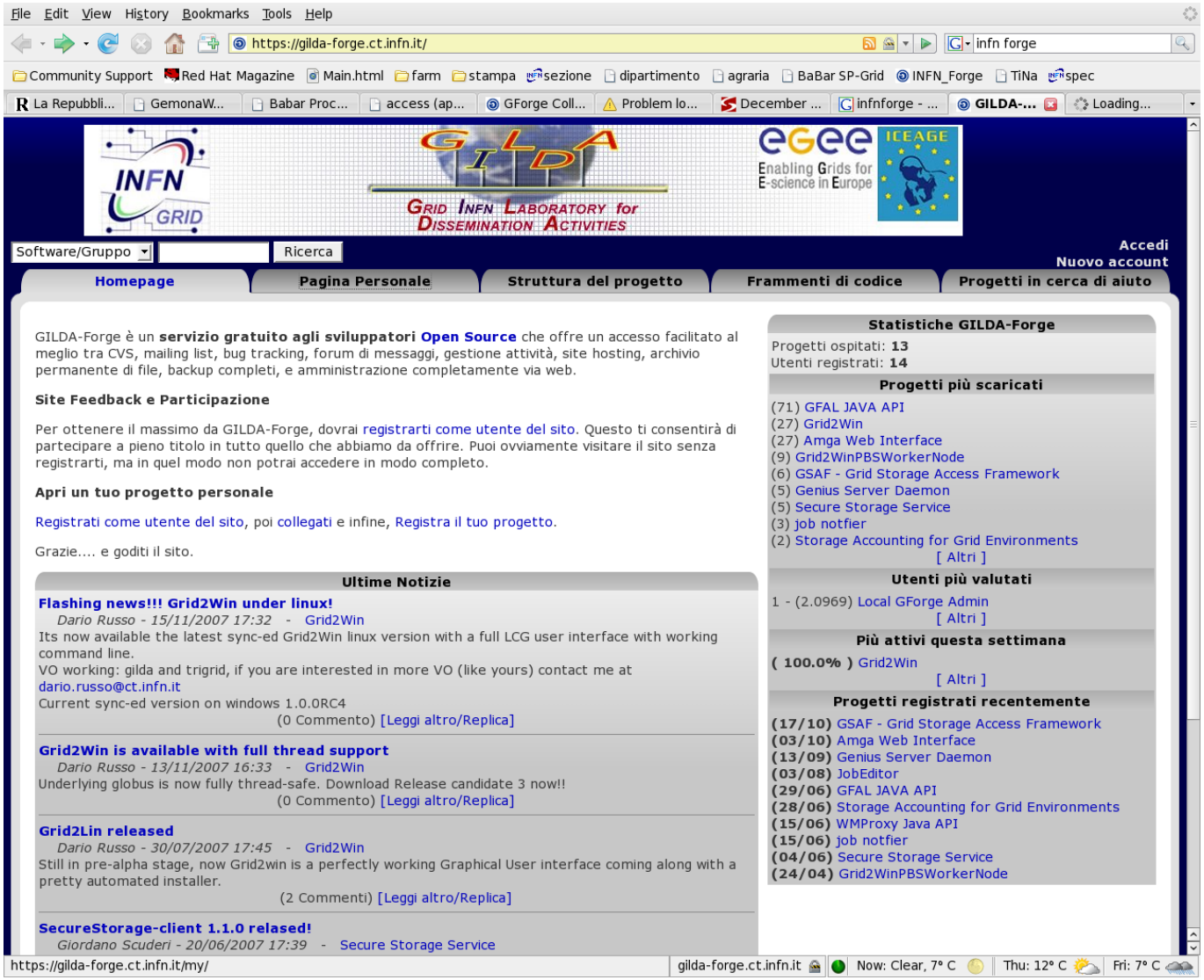
- a project admin can:
  - grant access to users, assign and tune their roles
  - update public project description
  - enable/disable features (eg. enable wiki, mail notifications)
  - customize trackers, forums, mailing lists (eg. add new forum, customize a forum's scope, add subprojects under tasks)
  - add release packages
  - shortly said: full rights for the project (deletion included)
- the gforge admin can:
  - customize access policies, enable projects
  - customize site-wide (may need php editing)

gforge

- gforge pros
  - unified place for projects of a geo-wide organization
  - unified and customizable look'n feel
  - users just search in a single place!
  - project manager can check tasks progress
  - flexible (works well without unwanted components)
  - wide admin/users community
- gforge cons
  - developers used with other tools or working methodology are reluctant with moving to a new tool
  - may be difficult to install (hint: “Use the debian, Luke!”)
  - may be tricky to configure/maintain
  - may need a fixed *admin timeslot* for a while, if no one else knows the system.

gforge

- how it looks like
- estimate few months to setup



software distribution

- the software distribution in BaBar is not optimal
  - you have to import the full release
- possible improvements
  - use package standards
    - rpm, deb?
      - it helps in deploying production on the GRID
  - decouple development production and user parts
    - possible split
      - base
        - probably what is in $BFROOT/bin or part of it
      - reconstruction production
        - only Elf, the shared libraries and tcl scripts
      - simulation production
        - only Moose, the shared libraries and tcl scripts

software distribution

- development
  - libraries
  - header files
    - no source files are needed
      - only those on which one is working (get through cvs, svn)
- the packages should be better defined, for each component
  - code packages
  - tcl packages
- the packages know their dependencies
- we could use yum or apt to distribute the releases
- issues:
  - manpower to manage this structure
  - access to the repository

software distribution

- set up a yum or apt repository for distributing packages
  - standard linux packages
  - also
    - geant4
    - clhep
    - ...
- other issue: decide on one or two (at most) architectures
  - just for now