Software development tools summary

Also sprach Giulio

- *The main point I would like to make is that when talking about software development one has to think about the whole process, from how to plan for new features to deployment, not about the single separate tools and steps*
- *I subscribe to this point of view, since we are at the very beginning of the journey and can take advantage of organized vs blobbing development*

- Good development planning and policies should also avoid a heavy turn to QA tools
  - Which are nevertheless usefull
    - And should be worth their price
- Some weeks ago Andrea Di Simone circulated a proposal for coding policy
  - Maybe it's the right time to agree on a base policy and start to follow it
    - Even if I'm not sure we can make it until the development is made on top of legacy BaBar code…

When will we decide what/how much BaBar code will be kept by SuperB?

- QA tools (or systems) could be also useful in our quest of “parallelizable code”
  - Especially running dynamic analysis ones, like valgrind or igprof

  ⇒ Extend, or go into thoroughly, the analysis to all SuperB code
    - Maybe this will also help us identifying code to be rewritten from scratch

- I realized that there's much more to "building" a software than simply "running make"
  - There is integration with VCS, QA/Unit test tools, cross compilation, integration with externals, packaging, distribution…

    → Etics is certainly a complete (and almost "keys in hand" tool), but I would keep investigating other solutions trying not to shoot a fly with a bazooka.
- Is our VCS suitable? Git promises to solve a lot of problems but, at least at a first glance, needs a clear and clean definition of development workflow (everything in Git is a branch, and you can have remotes with many branches to commit to…).

→ Do we have one?

- Planning, planning and planning
  - With a (possibly) clear picture of the complexity of our software (which shouldn't be too hard to define, given baBar experience…) identify the tools which adapt easier and faster
    - I'm thinking about the "small build unit -> Scram V1 model" choice, for example…
  - More coordination among developers (Fast and Full) to identify common strategies and policies