Building tools for SuperB

Marco Corvo

CNRS and INFN

Caltech XV SuperB General Meeting

December 16, 2010

Current build system

SoftRelTools was the standard in BaBar, but it has several limitations. Besides technologies have improved since SRT was developed.

SRT issues:

- SRT changed a lot over years
- It supports very old and no more used OSes and software
- Hand written Makefiles, which are difficult to manage and debug
- Impractical code dependencies and complex dependency management (what goes where, etc.)
- Online/SRT base issues where you want most flexibility & agility depended on this huge blob of SRT base

In other words: difficult to clean up or reorganize, Better to write from scratch

Solution(s) for these issues

1. Write SRT from scratch
   - Possible, but not very practical (at least in terms of man power)
2. Use available third party tools
   - Autotools
   - SCons
   - CMake

What are *Autotools*?

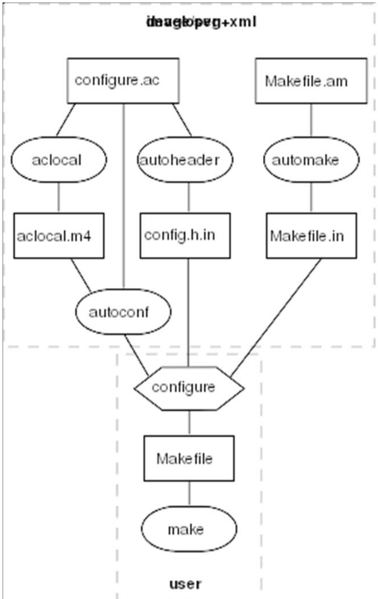
PROS

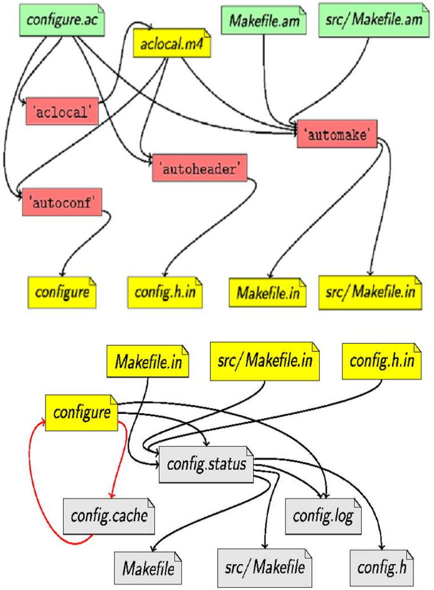
- Complete tool chain of several programs, each with different "macro" syntax
- Easy to use for users (./configure && make && make install)

CONS

- Same as point one of Pros (too many programs)
- Creates big build scripts and helper files even for a hello world example
- Hard to extend, hard to understand

Autotools flowcharts





What is *Scons*?

SCons is an Open Source software construction tool: it's a cross-platform substitute for the classic Make utility with integrated functionality similar to autoconf/automake and compiler caches such as ccache.

- Written in Python (a real OO programming language)
- Reliable, automatic dependency analysis built-in for C, C++ and Fortran
- Built-in support for C, C++, D, Java, Fortran, Yacc, Lex, Qt and SWIG, and building TeX and LaTeX documents
- Improved support for parallel builds

Very similar to CMake (features, cross platform support, behaviour) has the advantage of being written in Python. From my experience not so intuitive as CMake. Need to spend some time to get comfortable with it.

What is *CMake*?

1. Generates native build environments
   - UNIX/Linux: **Makefiles**
   - Windows: **VS Projects/Workspaces**
   - Mac OS: **Xcode**
2. Opensource
3. Cross-platform
4. Integrates testing and packaging systems

CMake features

1. Manage complex, large build environments (KDE4)
2. Very Flexible and Extensible
   - Support for Macros
   - Modules for finding/configuring software (bunch of modules already available)
   - Extend CMake for new platforms and languages
   - Create custom targets/commands
   - Run external programs
3. Very simple, intuitive syntax
4. Support for regular expressions (*nix style)
5. Support for In-Source and Out-of-Source builds
6. Cross Compiling
7. Integrated Testing and Packaging (Ctest, CPack)

Why Use *CMake*?

Pros

1. CMake depends only on C++ compiler
2. CMake supports great variety of platforms (basically every ***ix, Mac OS, Windows**)
3. CMake generates only Makefiles for all supported platforms
4. CMake additionally can produce project files for IDE's (KDevelop, XCode, VStudio)

Why Use *CMake*?

PROS (cont'd)

1. More usefull error messages when making a mistake in editing input files
2. Easy to use configure-like framework
3. CMake has simple syntax
4. CMake has a testing framework
5. CMake is faster than autotools (does not use libtools)

Furthermore, talking with CMS people, they also would use CMake if they were to write from scratch their build system

Why Use *CMake*?

Special interesting features

CMake combines further subsystems

1. CTest: used to automate updating (using CVS for example), configuring, building, testing, performing memory checking and submitting results to a CDash or Dart dashboard system
2. CPack: software packaging tool which can be used with or without CMake and is able to generate many different flavours of installers (RPM, Debian, DragNDrop, PackageMaker)
3. CDash: CDash is an open source, web-based software testing server. CDash aggregates, analyzes and displays the results of software testing processes submitted from clients located around the world. Developers depend on CDash to convey the state of a software system.

Current status of FastSim build

- Currently the prototype to build SuperB software with *CMake* works with the Head (trunk) of FastSim V0.2.6
- We forsee to release a FastSim V0.2.7 with fully working CMake support (likely in January) with the following features:
  - Full build of a Release
  - Build of single package or bunch of packages based on a given release
  - Support for Linux SL4 and SL5. We still have problems with Mac OSX due to some link issue

Current status of FastSim build

- CMakeLists files in place for every FastSim package
- Bunch of *CMake* macros and scripts to configure the release
  - Third party packages configuration and management (CLHEP, Root. . .)
  - Specific platform settings (compiler definitions and flags)
  - Bash script to run cmake executable in a more friendly way
- Already ongoing tests using the CTest framework

Short term

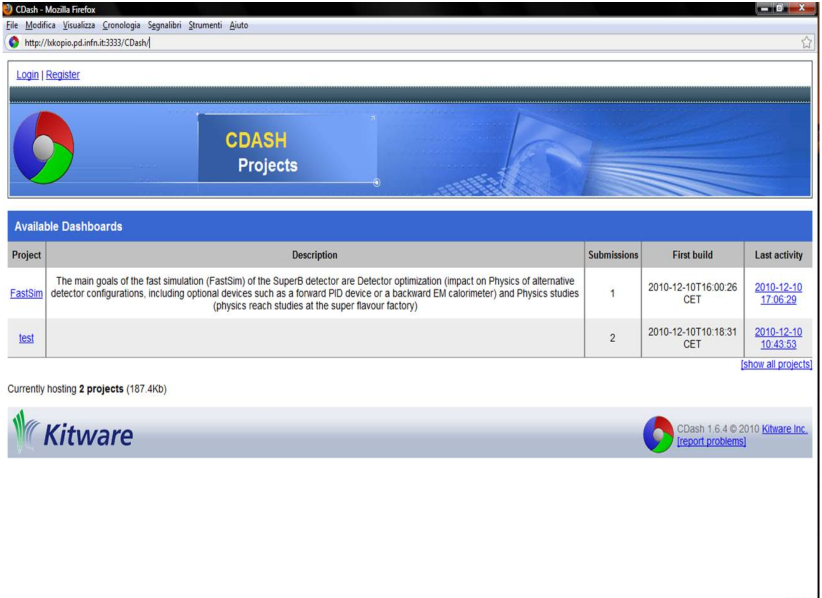
- First impression and first experience with FastSim is good. *CMake* is simple to use, flexible and has a large number of modules to set up and manage third party software (for FastSim I used *CMake* modules to configure Root, CLHEP and Boost libs)
- Current system is still a prototype which needs further and deeper work in order to turn it into a stable and widely usable one, in particular in relation to Mac OSX linking issues

Future plans

Future plans (next 2/3 months) consider developing prototypes with *CMake* combined with *CPack* (low priority by now), *CTest* and *CDash* (higher priority)

1. *CTest*
   - Useful to configure, build, test, perform memory check (e.g. via Valgrind)
   - Submits results to a CDash web site
2. *CDash*
   - Stores build information (history, failures, warnings, logs)
   - Useful (also) to set up alarms, notifications and build statistics
3. *CPack*
   - CPack can be used also without *CMake* as a standalone tool
   - Same syntax as *CMake*
   - Support for many different package generators (RPM, Debian, OSX, Cygwin)

CDash set up in Padova


CDash - Mozilla Firefox
File Modifica Visualizza Cronologia Segnalibri Strumenti Aiuto
http://lxkopio.pd.infn.it:3333/CDash/
Login | Register
CDASH
Projects

Available Dashboards

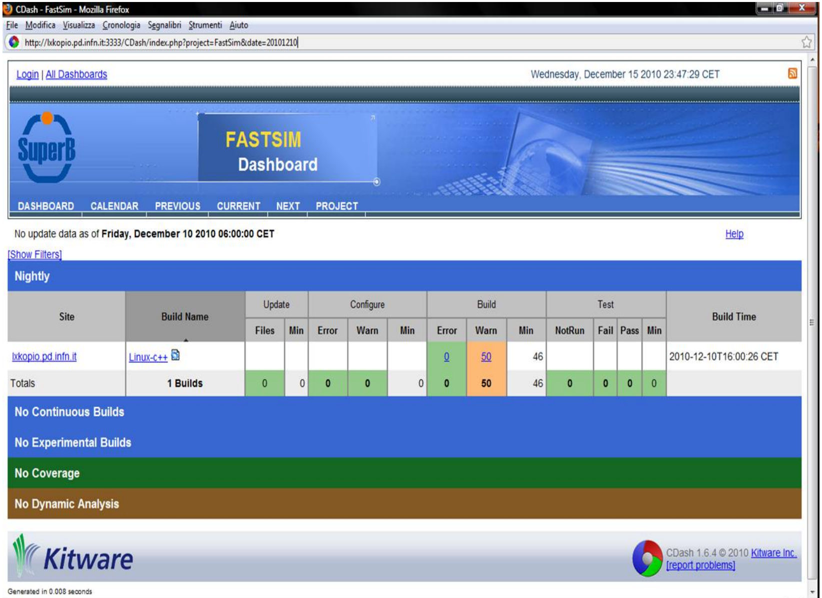
Project	Description	Submissions	First build	Last activity
FastSim	The main goals of the fast simulation (FastSim) of the SuperB detector are Detector optimization (impact on Physics of alternative detector configurations, including optional devices such as a forward PID device or a backward EM calorimeter) and Physics studies (physics reach studies at the super flavour factory)	1	2010-12-10T16:00:26 CET	2010-12-10 17:06:29
test		2	2010-12-10T10:18:31 CET	2010-12-10 10:43:53

[show all projects]

Currently hosting **2 projects** (187.4Kb)

CDash 1.6.4 © 2010 Kitware Inc. [report problems]

CDash set up in Padova


CDash - FastSim - Mozilla Firefox
File Modifica Visualizza Cronologia Segnalibri Strumenti Aiuto
http://lxkopio.pd.infn.it:3333/CDash/index.php?project=FastSim&date=20101210
Login | All Dashboards
Wednesday, December 15 2010 23:47:29 CET
SuperB
FASTSIM Dashboard
DASHBOARD CALENDAR PREVIOUS CURRENT NEXT PROJECT
Help
[Show Filters]

No update data as of **Friday, December 10 2010 06:00:00 CET**

Nightly

Site	Build Name	Update		Configure			Build			Test				Build Time
		Files	Min	Error	Warn	Min	Error	Warn	Min	NotRun	Fail	Pass	Min	
lxkopio.pd.infn.it	Linux-c++						0	50	46					2010-12-10T16:00:26 CET
Totals	**1 Builds**	0	0	**0**	**0**	0	**0**	**50**	46	**0**	**0**	**0**	0	

No Continuous Builds

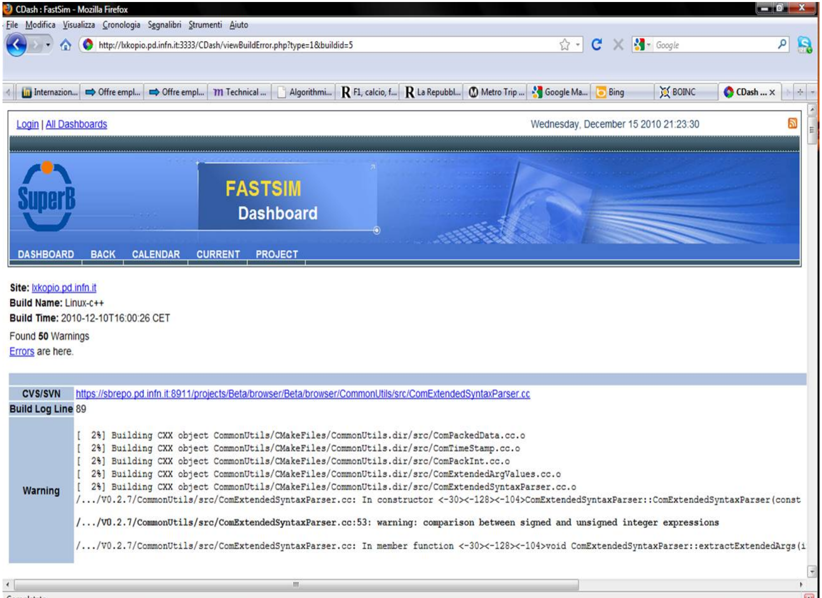
No Experimental Builds

No Coverage

No Dynamic Analysis

Kitware
CDash 1.6.4 © 2010 Kitware Inc.
[report problems]
Generated in 0.008 seconds

CDash set up in Padova


CDash : FastSim - Mozilla Firefox
File Modifica Visualizza Cronologia Segnalibri Strumenti Aiuto
http://lxkopio.pd.infn.it:3333/CDash/viewBuildError.php?type=1&buildid=5
Login | All Dashboards
Wednesday, December 15 2010 21:23:30
SuperB
FASTSIM Dashboard
DASHBOARD BACK CALENDAR CURRENT PROJECT

Site: lxkopio.pd.infn.it
Build Name: Linux-c++
Build Time: 2010-12-10T16:00:26 CET

Found **50** Warnings
Errors are here.

CVS/SVN	https://sbrepo.pd.infn.it:8911/projects/Beta/browser/Beta/browser/CommonUtils/src/ComExtendedSyntaxParser.cc
Build Log Line	89

Warning

```
[  2%] Building CXX object CommonUtils/CMakeFiles/CommonUtils.dir/src/ComPackedData.cc.o
[  2%] Building CXX object CommonUtils/CMakeFiles/CommonUtils.dir/src/ComTimeStamp.cc.o
[  2%] Building CXX object CommonUtils/CMakeFiles/CommonUtils.dir/src/ComPackInt.cc.o
[  2%] Building CXX object CommonUtils/CMakeFiles/CommonUtils.dir/src/ComExtendedArgValues.cc.o
[  2%] Building CXX object CommonUtils/CMakeFiles/CommonUtils.dir/src/ComExtendedSyntaxParser.cc.o
/.../V0.2.7/CommonUtils/src/ComExtendedSyntaxParser.cc: In constructor <-30><-128><-104>ComExtendedSyntaxParser::ComExtendedSyntaxParser(const

/.../V0.2.7/CommonUtils/src/ComExtendedSyntaxParser.cc:53: warning: comparison between signed and unsigned integer expressions

/.../V0.2.7/CommonUtils/src/ComExtendedSyntaxParser.cc: In member function <-30><-128><-104>void ComExtendedSyntaxParser::extractExtendedArgs(i
```