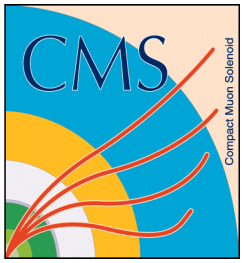


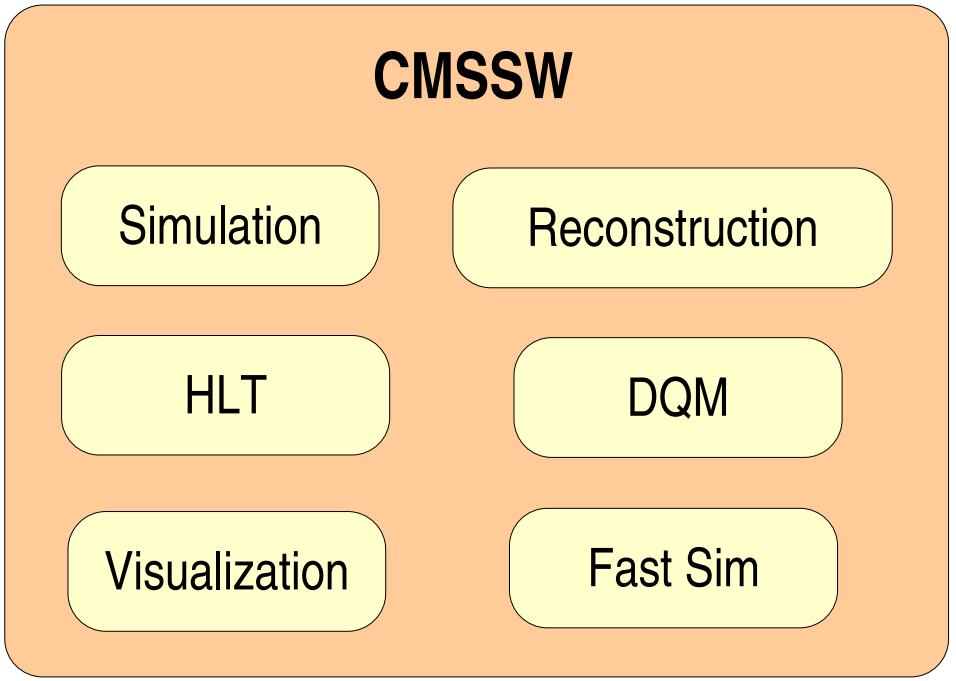
The Development and Release Process for the CMS Software Project

S. Argiro'
CERN and INFN

Introduction

- In this talk we will review the development and release cycle used in CMSSW.

- CMSSW is the name of the new Online/Offline project for the CMS experiment



Goals and Difficulties

Goals

- Allow fast development of a complex system
- Build often all the code relevant for development
- Speed up integration
- Keep code consistent and prevent divergences of development lines
- Allow fast release (for bug fixes, conference time, etc.) and distribution

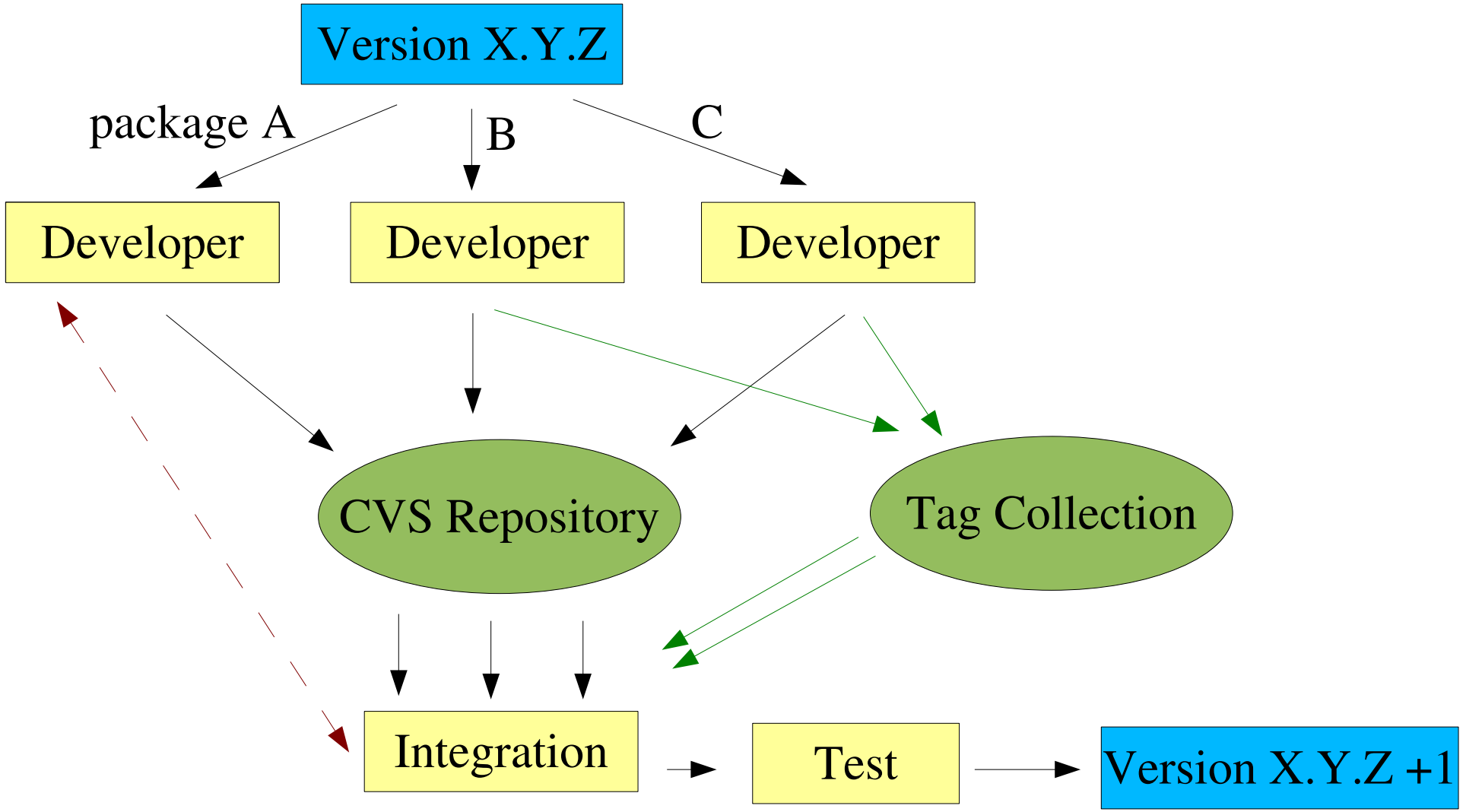
Difficulties

- Large and geographically dispersed collaboration
- Different skill levels

Release Timing and Planning

- For each major release the area coordinators try to set goals and functionalities that should be met.
- Monthly major releases
- Weekly integration releases

Development Cycle Overview



Development

SCRAM

- The key tool for the development of CMSSW is the SCRAM build system. SCRAM holds a database of (project,version) .
- The system configures the environment for one or several (project,version) in the developer's area.
- The developer then checks out the code of interest, and the build and runtime are configured transparently.
- This means the developer always works against an existing release, never against the CVS Head.

Code Organization

- All the code is kept in a single CVS repository
- The code is organized in a two-level hierarchy

Subsystem

Package

- One package -> one library
- Every package has two administrators who can commit and submit tags and developers who can only commit.
- Rather tight control on who can do what via CVS-pm

60 Active Developers
408 Packages
230000 lines of code

Communication and Integration

Code Submission

When the package manager considers the package is ready to be integrated, he submits a particular TAG to the system, for inclusion in the nightly build and possibly in the upcoming release.

Tag Collection

We have been using a simple, CVS-based tag collection mechanism.
A server-side script collects tags in a CVS file.

We are now deploying a web-based tag collection system on the basis of the one used by BaBar.

CMS Tag Collector: Release Management Page - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://pccms211.cern.ch:8080///CmsTagCollector/cgi-bin/ReleasePage

Getting Started Latest Headlines

CMS Software Page | nicos webpage with build results | CMS Tag Collector: Release...

Show Pending Tag Requests

CMS Tag Collector maintained by: David Lange

Ported for CMS from WebSRT.

Create Tag List

CMSSW_0_4_0_pre2

Release currently Open

Requests will be Committed

CalibCalorimetry/EcalDBInterfa	(i) V00-01-01*
CalibCalorimetry/HcalAlgos	(i) V02-00-01*
CalibCalorimetry/HcalPlugins	(i) V02-00-02*
CalibFormats/CaloObjects	(i) V00-00-01*
CalibFormats/HcalObjects	(i) V02-01-00*
CalibFormats/SiStripObjects	(i) V00-00-00*
CalibTracker/Records	(i) V02-00-00*
CalibTracker/SiStripConnectivi	(i) V02-00-00*
CalibTracker/SiStripPedestals	(i) V02-00-00*
CondCore/CSCPlugins	(i) V00-00-01*
CondCore/CalibPlugins	(i) V00-00-00*
CondCore/DBCommon	(i) V00-01-00*
CondCore/DTPlugins	(i) V01-00-01*
CondCore/ESSources	(i) V00-02-00*
CondCore/EcalPlugins	(i) V00-00-02*
CondCore/HcalPlugins	(i) V02-00-00*
CondCore/MetaDataService	(i) V00-00-06*
CondCore/PluginSystem	(i) V00-01-01*
CondCore/SiStripPlugins	(i) V01-00-00*
CondFormats/Alignment	(i) V00-00-00*
CondFormats/CSCObjects	(i) V01-00-01*
CondFormats/Calibration	(i) V00-01-00*
CondFormats/DTMapping	(i) V01-00-00*

Done

Nightly Build

- Nightly builds are available to ease and speed up the work of the integrator and of the release manager.
- All nightly builds are published to the SCRAM database
- We use a modified version of the NICOS system.
- When a package does not compile or link, a message is sent to developers and package administrators
- A web page shows the build report
- We plan to run Unit tests and notify failures
- Code metrics (dependency graph, include checking)

Test Results

Click to see Test log files TestLog	Click to see Test log Tool Checker
Click to see Dependencies log Dependencies	RuleChecker Files Rule Checker
Include Checker Log2 Include Checker	Click to see Warning Filter log Warning Filter

Build results for individual packages, failures first

Container	Package	Build	QA Test	Test	Administrator(s)
RecoTracker	RoadSearchCloudMaker	✗	N/A	N/A	gutsche@fnal.gov burkett@fnal.gov stevew@pizero.colorado.edu
RecoTracker	RoadSearchSeedFinder	✗	N/A	N/A	gutsche@fnal.gov burkett@fnal.gov stevew@pizero.colorado.edu
CalibCalorimetry	EcalDBInterface	✓	N/A	N/A	ricky.egeland@cern.ch
CalibCalorimetry	HcalAlgos	✓	N/A	N/A	Shahram.Rahatlou@cern.ch ratnikov@fnal.gov jmmans@physics.umn.edu wfisher@fnal.gov
CalibCalorimetry	HcalPlugins	✓	N/A	N/A	Shahram.Rahatlou@cern.ch ratnikov@fnal.gov
CalibCalorimetry	HcalStandardModules	✓	N/A	N/A	jmmans@physics.umn.edu wfisher@fnal.gov
CalibFormats	CaloObjects	✓	N/A	N/A	Jeremy.Mans@cern.ch
CalibFormats	HcalObjects	✓	N/A	N/A	Paolo.Meridiani@cern.ch jmmans@physics.umn.edu ratnikov@fnal.gov
CalibFormats	SiStripObjects	✓	N/A	N/A	Giacomo.Bruno@cern.ch
CalibTracker	Records	✓	N/A	N/A	Robert.Bainbridge@cern.ch
CalibTracker	SiStripConnectivity	✓	N/A	N/A	Robert.Bainbridge@cern.ch Suchandra.Dutta@cern.ch Giacomo.Bruno@cern.ch
CalibTracker	SiStripPedestals	✓	N/A	N/A	Robert.Bainbridge@cern.ch Giacomo.Bruno@cern.ch domenico.giordano@ba.infn.it

Testing

- We are using **CppUnit** as a unit test framework and plan to use **Oval** for validation.

- Plans on using test coverage reports.

- Testing and validation:

  1. Are a quality assurance tool
  2. Save time
  3. Make the integrator's life easier

- Problem: getting the developers to write tests in a uniform and useful way.

- We plan on having different stages of testing and validation depending on the nature (devel/prod) of the release.

Development Tools

We make use of several development tools that were setup over the past years

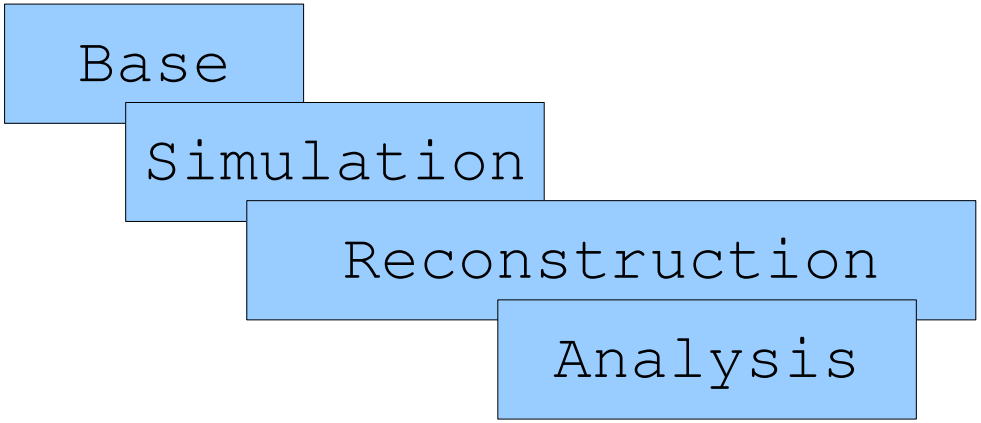
rule checker : check compliance of coding style

include checker : check inclusion of unnecessary headers

dependency checker : make dependency graphs and help finding circular deps

Distribution

- We use the RPM system to distribute CMSSW
- The distribution system used in CMS is described in another two contributions to this conference (see talks by Andreas Nowack and Klaus Rabbertz)
- We plan on modularizing the distribution kit



Differences with previous CMS experience

Previously:

1. Separate projects for framework, simulation, reconstruction, etc
   configuration problems
   (easier distribution)

2. Staged release process (order of dependency)

   long time needed to release
   (cut down to 1 or 2 days)

   divergent lines of development

Plans for improvement

- Optimize the Build (parallelization)
- Modularize the distribution kit
- Optimize the Testing infrastructure

Conclusions

The CMSSW project started in March 2005
We started developing new development procedures in August 2005

The procedure is successful in delivering rapidly
(as much as three releases in a week !)