

# Software Development Environment with **SCRAM V1**

Natalia Ratnikova  
Fermilab CD/CMS  
For **EDM** Project

# This Presentation Includes:

- Introduction into **CMS** software development process
  - ✓ SCRAM tool
  - ✓ Software projects
  - ✓ CVS repositories
  - ✓ Configuration management
  - ✓ Release cycle
  - ✓ Software environment
  - ✓ Software distribution
- **SCRAM** commands
- Examples for **EDM**

# SCRAM tool

- ❑ The main task of SCRAM is to ensure that all developers are working with the same consistent set of external products, libraries, environments and source codes. Configuration management methods ensure that this is possible.
- ❑ The build and runtime environments are constructed automatically from the description documents of the required external products and the project-specific environment.

# Software Projects

*SCRAM project is a releasable unit covering a particular software domain.*

## □ Basic CMS projects:

- COBRA - *basic analysis framework*
- OSCAR - *Geant4 based simulation*
- FAMOS - *fast simulation*
- ORCA - *reconstruction*
- IGUANACMS - *visualization*

## □ Package-oriented structure:

- ✓ Packages provide libraries
- ✓ Subsystems are collections of packages
- ✓ One path to all *include files*:
  - \$(LOCALTOP)/\$(SCRAM\_SOURCEDIR)
  - #include “*SubSystem / Package / interface / X.h*”

# CVS repositories

## *CVS infrastructure at CERN:*

- ✓ standardized source code repositories
- ✓ standard directory structure for code organization.
- ✓ a cvs server supporting p-server, k-server and ssh access.
- ✓ CVSPm for distributed management of authority and access
- ✓ SCRAMToolBox for configuration definition and versioning
  - Contact persons:
    - [Nick.Sinanis@cern.ch](mailto:Nick.Sinanis@cern.ch)
    - [Shaun.Ashby@cern.ch](mailto:Shaun.Ashby@cern.ch)

# Configuration management

- ❑ Projects need to share the same configuration so that they can inter-operate. SCRAM provides a mechanism for importing independently maintained configuration documents automatically.
- ❑ SCRAM project configuration is defined by 2 elements:
  - ✓ project configuration definition
    - EDM/config -
      - contains build templates, requirements, scram version, boot file, top-level BuildFile, etc.
  - ✓ external tools configuration requirements:
    - SCRAMToolBox/CMSconfiguration
- ❑ SITE-dependent configuration setup for external products (tools).

# Release cycle

- Projects are developed at a different pace.
  - ✓ Each project may have its own conventions, defined by the project manager.
  - ✓ CMS (and LCG) releases are usually preceded by a series of pre-releases
    - prereleases are often compiled with debug flag
    - prereleases have a shorter lifetime and are not being packaged for a binary distribution
- Released installations are made available through the scram database.
- Old installations are gradually removed.

# Software environment

*General software environment available upon login to the system (or source setup file):*

- ❑ Provides path to CMS utilities and tools:
  - ✓ access to *scram* and *scramdb*
  - ✓ commands to set access to cvs
- ❑ Defines default **cernlib** environment,
- ❑ Commands to set up environment for non-scram managed projects (e.g. **CMKIN**).
- ❑ Runtime environment for scram-managed projects is controlled by SCRAM.

# Software distribution

- ❑ SCRAM projects can be `bootstrapped' from a single description document in which the structure and download information of other required project documents and components is declared.
  - `scram project -b config/bootsrc`
- ❑ CMSInstall tool provides binary distributions in rpm format, including externals and build environment for the CMS official releases:
  - `xcmsi.pl`
- ❑ DAR tool distributes applications in binary form, including official and user's private code, and the runtime environment.
  - *Not used in development.*

# SCRAM commands

- ❑ Online help:
  - ✓ `scram help` ; `scram <command> help`
- ❑ Commands used for development:
  - ✓ `scram project <Project> <Version>`
  - ✓ `scram build` ; `eval `scram runtime -csh``
- ❑ Query commands:
  - ✓ `scram version`; `scram list`; `scram arch`
  - ✓ `scram tool list`; `scram tool info <toolname>`
- ❑ Release management commands:
  - ✓ `scram install`; `scram db`; `scram remove`
- ❑ Configuration management commands:
  - ✓ `scram setup`; `scram tool remove`

# Example for EDM development

- ❑ `scram list EDM`
  - ✓ *lists available project releases*
- ❑ `scram project EDM EDM_0_0_1_pre1`
  - ✓ *creates empty directory structure in the cwd, associated with the base installation of project version.*
- ❑ `cmscvsroot EDMProto` or `project EDMProto;`
  - ✓ *sets CVSROOT for pserver or kserver respectively*
- ❑ `cd EDM_0_0_1_pre1/src;`
- ❑ `cvs co Event`
  - ✓ *checks out most recent revision of the source code for Event subsystem*
- ❑ `scram b`
  - ✓ *rebuilds all modified source code*
- ❑ `eval `scram runtime -csh``
  - ✓ *sets the runtime environment in tcsh or csh session*
- ❑ *Now you can run test application: Provenance\_t*

# Additional Information Sources

- ❑ SCRAM documentation:  
<http://spi.cern.ch/cgi-bin/scrampage.cgi>
- ❑ SCRAM page in Savannah:  
<http://savannah.cern.ch/projects/scram>
- ❑ "Moving to SCRAM Version 1" . CMS Software Tutorial:  
<http://agenda.cern.ch/fullAgenda.php?ida=a043735>
- ❑ SCRAM Version 1.0 Release: New Features, Implications.  
*More technical:*  
<http://agenda.cern.ch/age?a043441>
- ❑ ORCA developers mailing lists archives:  
<http://cmsdoc.cern.ch/orca/#mailinglists>