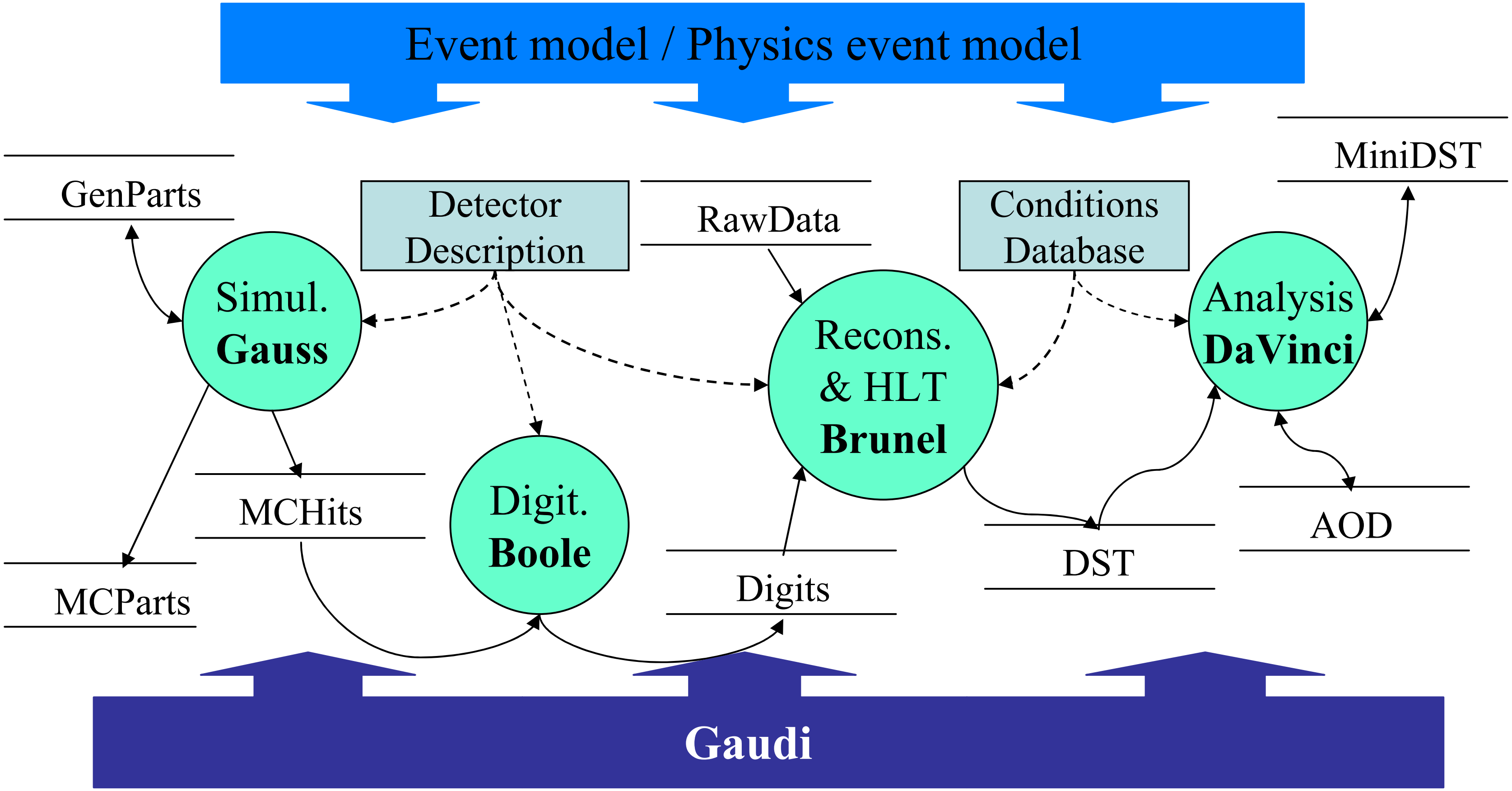
1

Overview of LHCb applications and software environment

LHCb applications



Event model / Physics event model

GenParts

Detector Description

RawData

Conditions Database

MiniDST

Simul. **Gauss**

Recons. & HLT **Brunel**

Analysis **DaVinci**

Digit. **Boole**

MCHits

MCParts

Digits

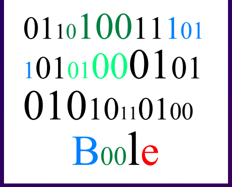
DST

AOD

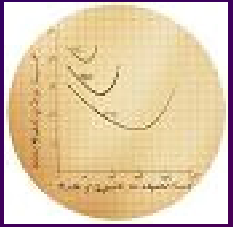
Gaudi

Main LHCb applications

- **Gauss**
  - Event generation and GEANT4 simulation



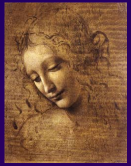
- **Boole**
  - Detector response and digitization
  - Output in same format as real data



- **Moore**
  - Trigger reconstruction and HLT selection
  - Runs both online (in trigger farm) and offline

- **Brunel**
  - Event reconstruction
  - Output Tracks, Particle ID, “ProtoParticles”

More main LHCb applications



- **DaVinci**
  - Physics analysis framework
  - Manipulate particles and vertices to identify and measure physics processes

- **Panoramix**
  - Event and geometry display
  - Scripting based on Python

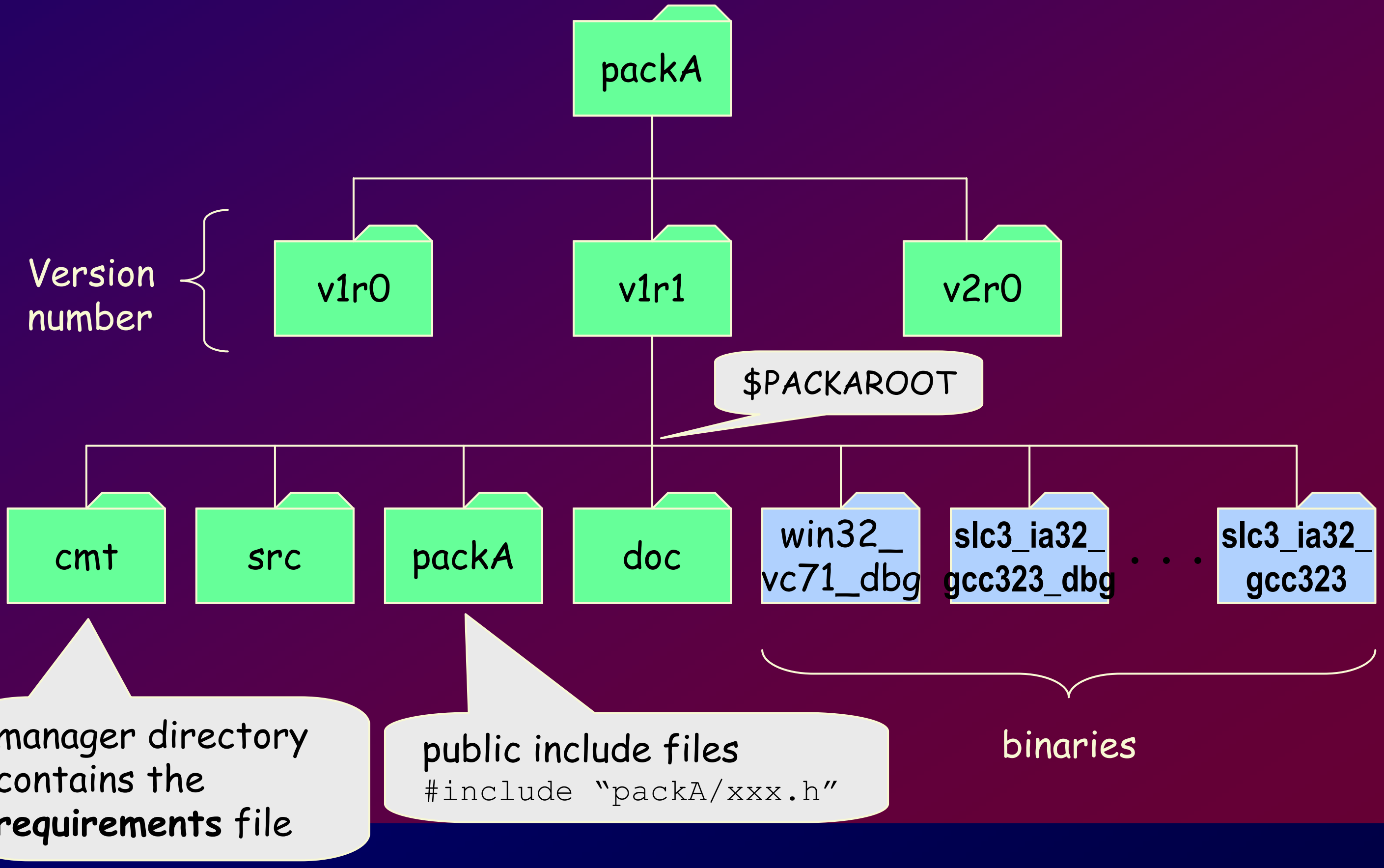
- **Ganga**
  - User interface for handling job preparation, submission and retrieval (e.g. on the grid)

+BENDER
+ONLINE
+EULER
+ test_beams
+…

Applications are built from packages

- Package Definition:
  - Collection of related classes in a logically cohesive physical unit
  - Minimal entity that can be versioned
- Reflects on
  - Logical structure of the application
  - Organizational structure of the development team

Package: Structure



packA

Version number

v1r0

v1r1

v2r0

$PACKAROOT

cmt

src

packA

doc

win32_ vc71_dbg

slc3_ia32_ gcc323_dbg

. . .

slc3_ia32_ gcc323

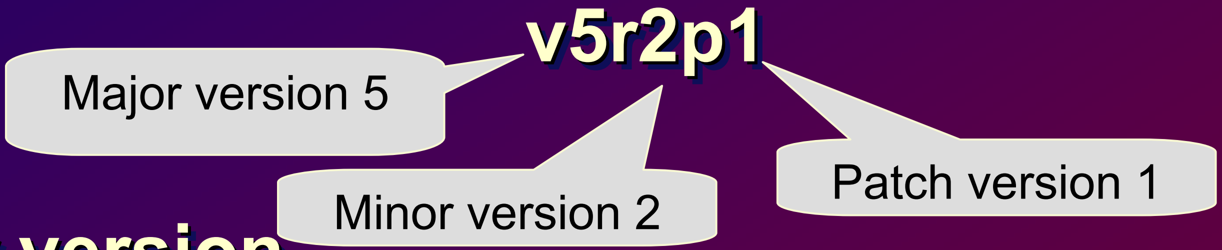
manager directory contains the **requirements** file

public include files
`#include "packA/xxx.h"`

binaries

Package versions

Packages have several versions:



v5r2p1

Major version 5

Minor version 2

Patch version 1

Major version

- Indicates a change in the interface: all packages that use it may have to change

Minor version

- Indicates an internal only change

Patch version

- Not usually present. A minor bug fix to an existing release

Project

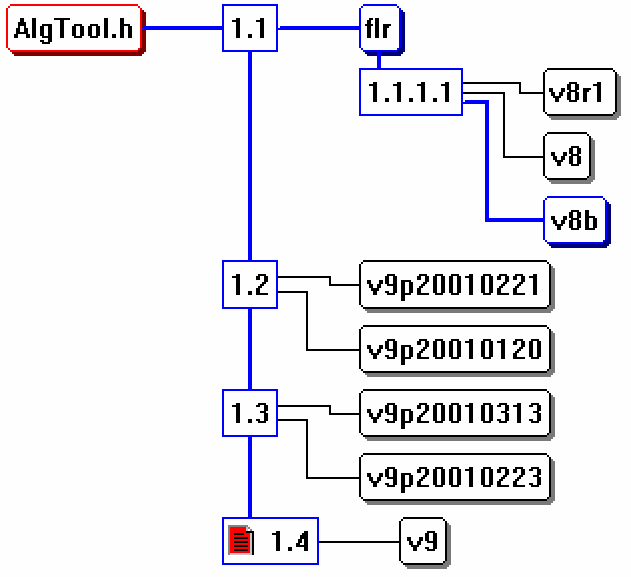
- Projects are a collection of packages that are released together
  - One project per application (e.g. Brunel, DaVinci)
  - Several independent projects for components (e.g. Lbcom, Rec, Phys)
  - Two projects for the framework (Gaudi, LHCb)
- Users work in the environment defined for a given version of the chosen project
  - e.g. DaVinciEnv v17r5

CVS

Version Control System

- Record the history of your source files
- Helps you if you are part of a group of people working on the same project.

(Repository, Module, File, Version, Tag)



CVS: Common Repository

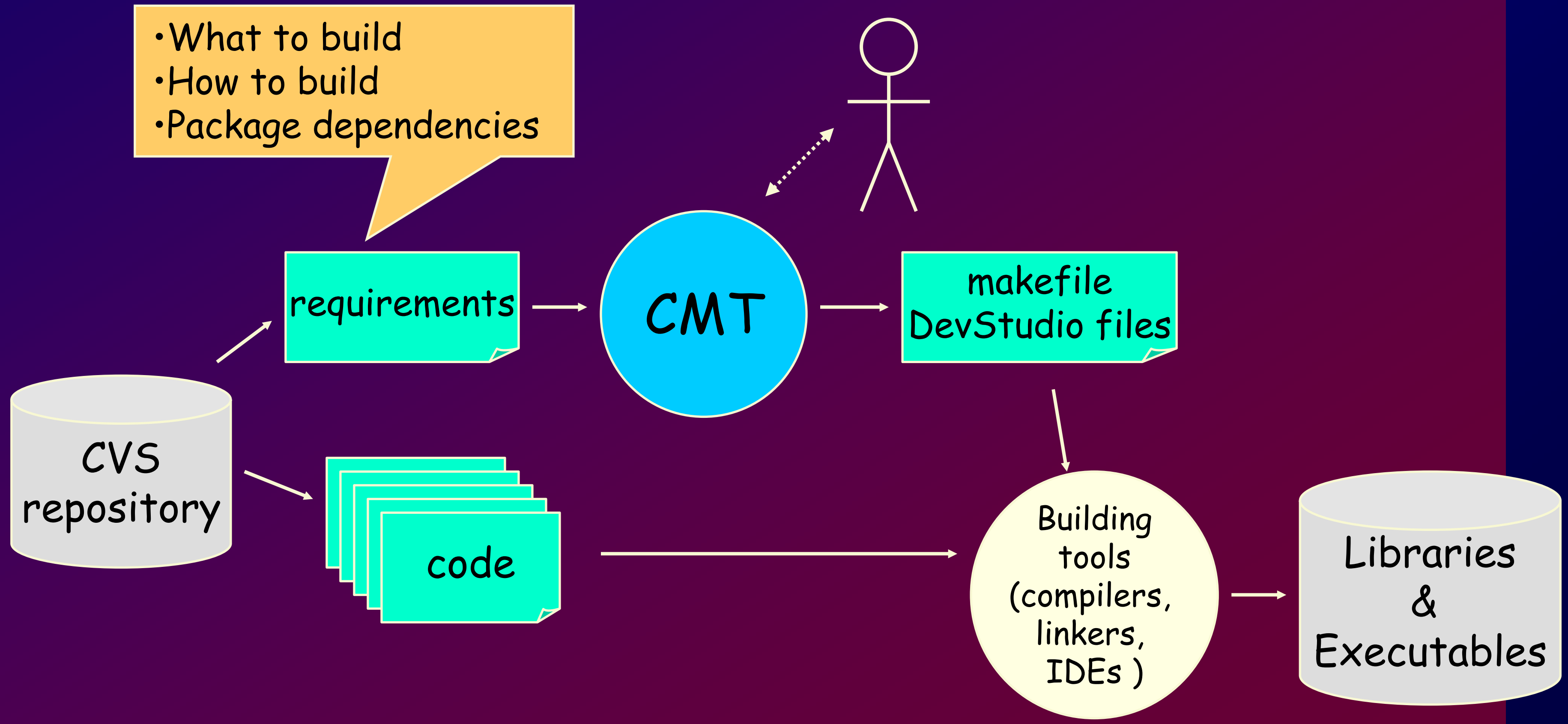
- LHCb Repository on CERN-IT CVS server
  - Web browsable
    - http://isscvs.cern.ch/cgi-bin/cvsweb.cgi/?cvsroot=lhcb
    - http://isscvs.cern.ch/cgi-bin/cvsweb.cgi/?cvsroot=Gaudi
  - World readable if authenticated
    - Kerberos authentication (e.g. AFS on CERN Linux)
      - Configured by LHCb group login at CERN
    - SSH authentication (e.g. from Windows)
    - Detailed instructions at http://cvs.web.cern.ch/cvs/howto.html#accessing
  - For write access
    - register with Florence.Ranjard@cern.ch

CMT

Configuration Management Tool written by C. Arnault (LAL, Orsay)

- It is based around the notion of *Package*
- Provides a set of *tools for automating* the configuration and building packages
- It has been adopted by LHCb (other experiments are also using it)

How we use CMT



- What to build
- How to build
- Package dependencies

CMT: Requirements file

```
package           MyPackage
version           v1r0

# Structure, i.e. directories to process.
branches          cmt doc src

# Package does not export any public include files
include_path none

# Used packages.
use GaudiAlg      v*

# Component library building rule
library          MyPackage    ../src/*.cpp

# define component library link options
apply_pattern component_library library=MyPackage
```

CMT and projects

- CMTPATH
  - The directories to look for CMT packages
  - Initialised to ~/cmtuser in LHCb login
- CMTCONFIG
  - The “default” configuration
- <Project>Env [<version>]
  - Adds to the CMTPATH the path where the project packages are located and their dependent projects
- <Project>_release_area
  - Specifies the path to a project, in case it does not reside in the default release area
  - Set to be equal $LHCBRELEASES
    - /afs/cern.ch/lhcb/software/releases @CERN
    - /software/lhcb/lhcb @UNI-DORTMUND

CMT: Basic Commands

- **cmt config**
  - Configures the package (creates setup and make files)
- **source setup.csh**
  - Sets environment
- **cmt show uses**
  - Show dependencies and actual versions used
- **cmt show macro <macro>**
  - Show the value of a macro for the current configuration
- **cmt binclean**
  - Clean all binaries (libraries, executables, dictionaries,etc.)
- **cmt broadcast <command>**
  - Recursive CMT command in all used packages found on first component of CMTPATH
  - e.g. cmt broadcast gmake

Package Categories

- ***Program***: is a package that contains a main routine and a list of dependent packages needed to link it.
- ***Library***: contains a list of classes and the list of dependent packages needed to compile it.
- ***Package group***: contains a list of other packages with their version number (e.g. GaudiSys)
- ***Interface package***: interfacing to packages not managed with CMT (e.g. Python, GSL, ROOT,…)

Link vs. *Component* Libraries

- Link libraries are need for linking the program (static or dynamic)
  - Traditional libraries.
- Component libraries are loaded at run-time (*ApplicationMgr.DLLs* property)
  - Collection of components (Algorithms, Tools, Services, etc.)
  - Plug-in

Component Libraries

Components_load.cpp

```
#include “GaudiKernel/DeclareFactoryEntries.h”
DECLARE_FACTORY_ENTRIES ( Components ) {
  DECLARE_ALGORITHM( MyAlgorithm )
  DECLARE_SERVICE( MyService )
  DECLARE_TOOL( MyTool )
}
```

Your components need to be added here

Components_dll.cpp

```
#include “GaudiKernel/LoadFactoryEntries.h”
LOAD_FACTORY_ENTRIES ( Components )
```

No change needed

Getting a package

- The “getpack” command
  - Script combining “cvs checkout” + “cmt config”

```
> getpack [hat/]<package> [<version>] [head]
```

  - If no version given, it suggests the **latest** version of package
    - N.B. Suggested version is not necessarily consistent with current environment; especially if you are not using the latest environment

Building a package

- Working in the /cmt directory
  - <package>/<version>/cmt
- Invoke the make command

```
> make [target] [tag=<configuration>] [clean]

         configurations:   $CMTCONFIG (default)
                           $CMTDEB (for debug)
```

- Set the run time environment
  - Not needed for building
  - MANDATORY TO RUN THE PROGRAM

```
> source setup.csh [-tag=<configuration>]
```

CMT: 5 magic lines

- Experience shows that 90-95% of “misterios” problems (compile-time and run-time) are due to misconfiguration
- Please pay attention to correct configuration!
- Many problems could be detected/eliminated by 5 magic CMT-lines
  - cmt show uses [| grep cmt]
    - Show dependencies and actual versions used
  - cmt br make binclean
    - Cleanup ALL LOCAL packages
  - cmt bt cmt config
    - Re-configures all local package
  - cmt br make
    - Re-build all local packages
  - source setup.csh
    - Sets environment

Emacs customisation

- A customisation of emacs for LHCb:
  - Templates for creation of files
    - E.g. MyAlgorithm.h, .cpp, <Components>_load.cpp, <Components>_dll.cpp, requirements etc.
  - Various shortcuts for code insertions
  - Optionally, load an EDT keypad emulation
- Add following lines to ~/.emacs:

```
(load (expand-file-name "$EMACSDIR/edt"))
(load (expand-file-name "$EMACSDIR/lhcb"))
```

  - Or copy from $EMACSDIR/.emacs

Exercise

Now read the web page attached to this lesson in the agenda and work through the exercises