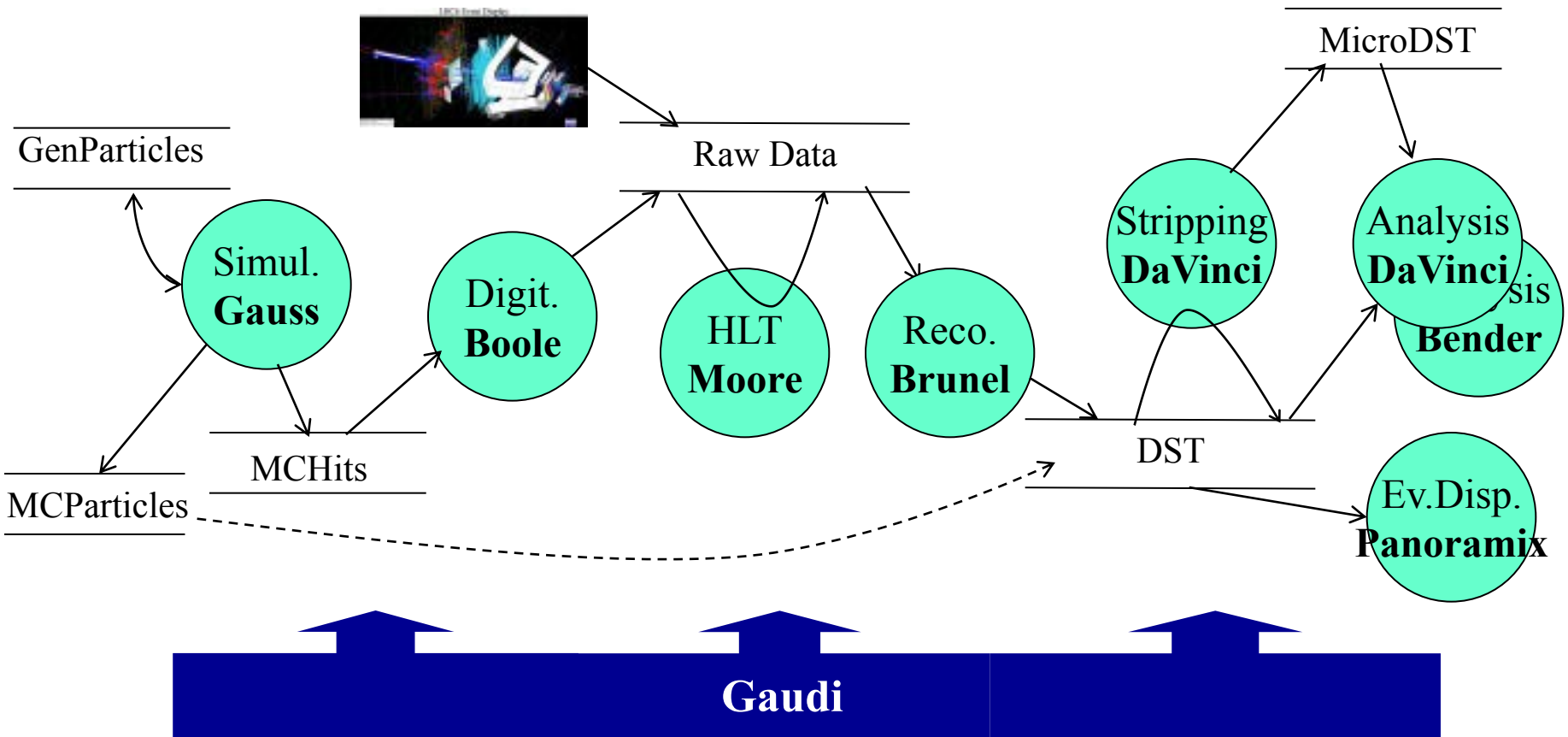


Overview of LHCb applications and software environment

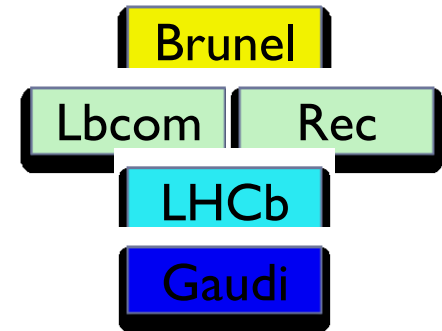
LHCb applications

Event Model / Detector Description / Conditions



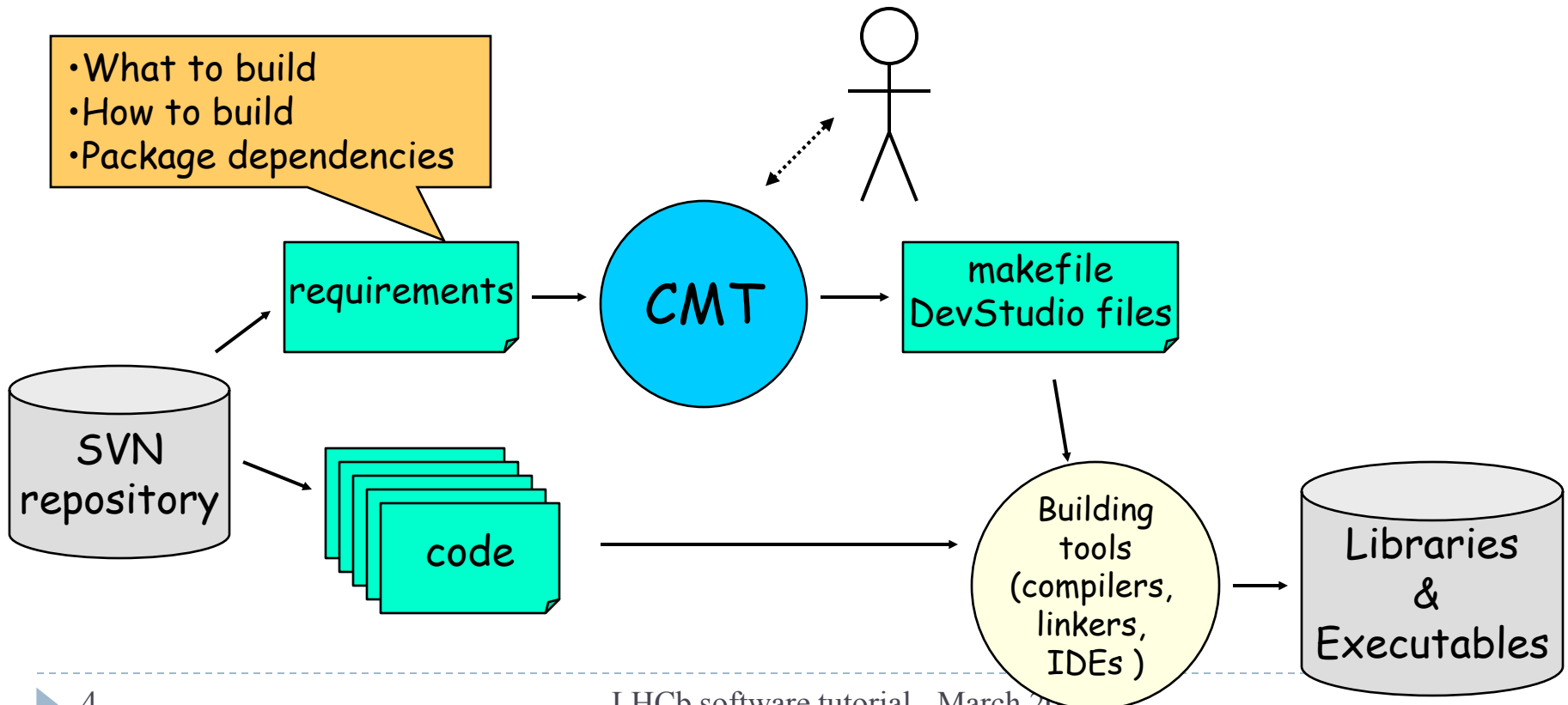
Software organisation

- ▶ Applications are released as a stack of *Projects*
 - ▶ One project per application
 - e.g. Brunel, DaVinci
 - ▶ Several independent projects for components
 - e.g. Lbcom, Rec, Hlt, Analysis, Stripping
 - ▶ Four projects for the framework
 - Gaudi, LHCb, Phys, Online
- ▶ Projects are collections of *Packages*
 - ▶ Group of classes in a logically cohesive physical unit.
 - ▶ Packaging structure reflects on:
 - Logical structure of the application
 - Organizational structure of the development team
 - ▶ Package is minimal entity that can be versioned
 - ▶ A project version uniquely defines the versions of the packages it contains, and of the projects it depends on
 - ▶ Users work in the environment defined for a given version of the chosen project
 - e.g. SetupProject DaVinci v28r1p1

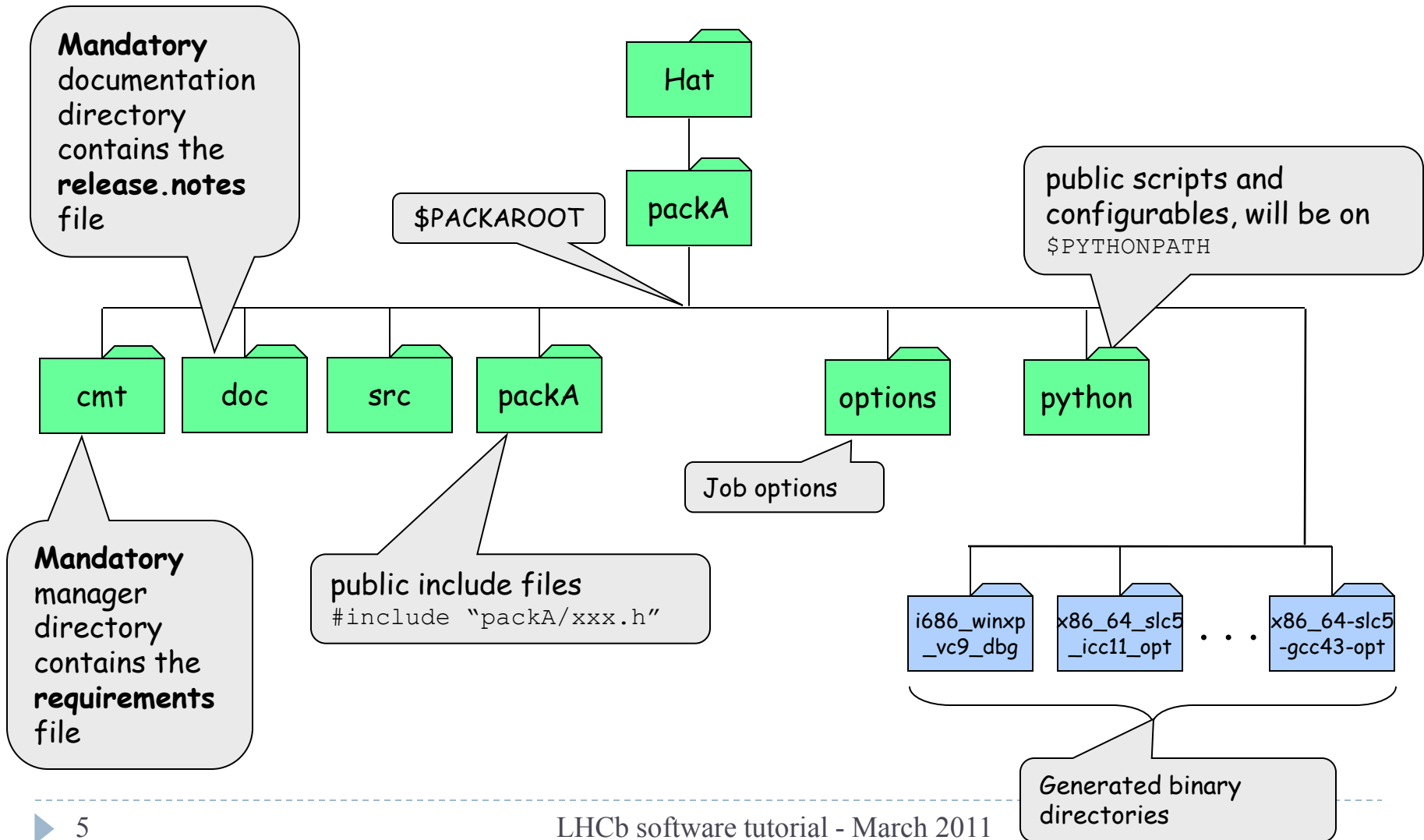


CMT (Configuration Management Tool)

- ▶ Based around the notion of package
- ▶ Provides a set of tools for automating the configuration and building of packages

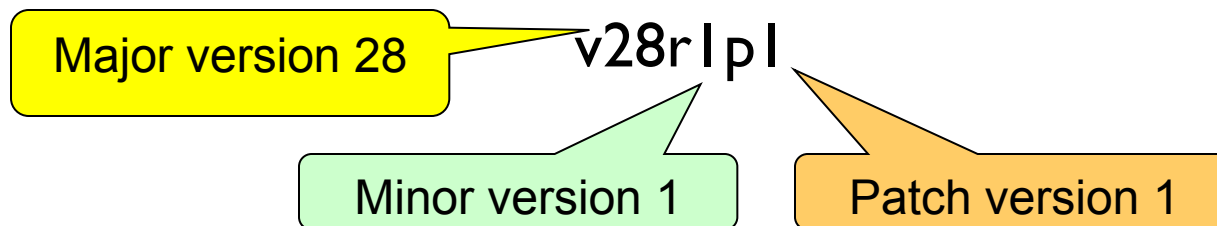


Package: Structure



Package versions

- ▶ Packages have several versions
 - ▶ Defined in cmt requirements file
 - ▶ Version number formatted according to convention:



- ▶ 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 always present. A minor bug fix to an existing release

Package Categories

- ▶ **Application**: is a package that sets up the environment and contains the job options needed for configuring and running a program.
- ▶ **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. LHCbSys)
- ▶ **Interface package**: interfacing to packages not managed with CMT (e.g. Python, GSL, ROOT,...)

Link vs. Component Libraries

- ▶ Link libraries are **linked** to the program executable
 - (static or dynamic linking)
- ▶ Base classes, event and geometry data classes
 - ▶ Loaded at program start
 - ▶ Must be found on LD_LIBRARY_PATH
- ▶ Component libraries are **loaded on demand** at run-time
 - ▶ Collection of components (Algorithms, Tools, Services, etc.)
 - ▶ “Plug-ins”
 - ▶ No need to recompile or re-link program if plug-in changes

CMT: requirements file

```
package           MyPackage
version           v1r0

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

# 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: Basic Commands

- ▶ **cmt config**
 - ❑ Configures the package (creates setup and make files)
 - ❑ Invoked automatically by getpack (see later slide)
- ▶ **cmt show uses**
 - ❑ Show dependencies and actual versions used
- ▶ **cmt make**
 - ❑ build the current package
- ▶ **cmt broadcast <command>**
 - ❑ Recursive CMT command in all used packages found in the current CMT project
 - ❑ e.g. `cmt broadcast cmt make`
- ▶ **cmt run <command>**
 - ❑ Executes <command> in current package environment

Setting the CMT environment

- ▶ Create a working directory for a given project

```
> setenv<Project> [<version>]
```

- ▶ actually an alias for

```
SetupProject --build-env <Project> [<version>]
```

- ▶ creates working directory and `cd` to it:

```
~/cmtuser/<Project>_<version>
```

- ▶ Packages are searched for in Projects that are on

CMTPROJECTPATH

- ▶ **Default is** `${User_release_area}:${LHCBPROJECTPATH}`
- ▶ **Add additional paths with** `--nightly` and `--dev-dir` switches of `SetupProject`

Getting a package

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

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

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

Building a package

- ▶ Work in the **/cmt** directory
 - `<hat>/<package>/cmt`
- ▶ Set the CMT configuration (`$CMTCONFIG`)
 - Defines platform, compiler, directories where to find binaries
 - slc5 default: `x86_64-slc5-gcc43-opt`
 - `LbLogin -c i686-slc5-gcc43-dbg`
 - `setenv CMTCONFIG $CMTDEB`
- ▶ Invoke the make command, via `cmt`

```
> cmt make [-j] [target] [clean] [binclean]
```

- Putting “`cmt`” in front of any command ensures that the command is executed in a shell that has defined all the environment described by the requirements file

Running the application

- ▶ Set the run time environment

```
> SetupProject <project> <version>
```

- ▶ Takes into account value of CMTCONFIG
- ▶ Needed once at beginning of session, then only if environment changes (new packages checked out, requirements file changed, CMTCONFIG changed, etc.)

- ▶ Execute the program

```
➤ gaudirun.py job.py
```

- ▶ But see also **cmt run** command in hands on

SVN (Version Control System)

- ▶ Record the history of your source files
- ▶ Share your development with other people
 - ▶ Commit often
 - ▶ But only codes that compiles and builds
 - ▶ Tested by nightly builds
- ▶ LHCb and Gaudi Repositories reside on CERN-IT SVN server
 - ▶ Private repositories are also possible, but must be managed by yourself, see <http://svn.web.cern.ch/svn/howto.php#creating-requesting>

Accessing LHCb SVN server

▶ Web browsable

▶ LHCb:

- Trac: <https://svnweb.cern.ch/trac/lhcb/browser>
- WebSVN: <http://svnweb.cern.ch/world/wsvn/lhcb>

▶ Gaudi:

- Trac: <https://svnweb.cern.ch/trac/gaudi/browser>
- WebSVN: <http://svnweb.cern.ch/world/wsvn/gaudi>

▶ World readable if authenticated

- ▶ SSH authentication
- ▶ Automatic if authenticated in CERN AFS cell (lxplus)

▶ For write access

- Register your account in the e-group [lhcb-svn-writers](#)

▶ Detailed instructions at

<https://twiki.cern.ch/twiki/bin/view/LHCb/SVNUUsageGuidelines>

Emacs customisation

- ▶ A customisation of emacs for LHCb:
 - ▶ Templates for creation of files
 - E.g. MyAlgorithm.h, MyAlgorithm.cpp, requirements, release.notes 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