Software Installation and Releases

Jan Engels

DESY

EUDET: Annual Meeting 2007

8th October 2007

Software Engineering

- Creating/Maintaining a complex software framework is certainly a very difficult task to manage. Some of the most common problems that can show up as the project grows are listed below:
  - Increase of dependencies (external/internal)
  - Lack of documentation
  - Lack of communication between people/groups
  - Increase of requirements
  - Increase of support/maintenance costs
  - Problem of keeping standards alive
  - Different coding arts (one of them aka spaghetti-code :)
  - Migration problems (support for other platforms/OS's ...)
  - And **many** others not listed here...

Software Engineering

- Even a team of experienced administrators would have a hard time trying to manually handle all the tasks bound to the management of a large software project:
  - Maintaining
  - Releasing
  - Testing
  - Support

- Automating things is inevitable!

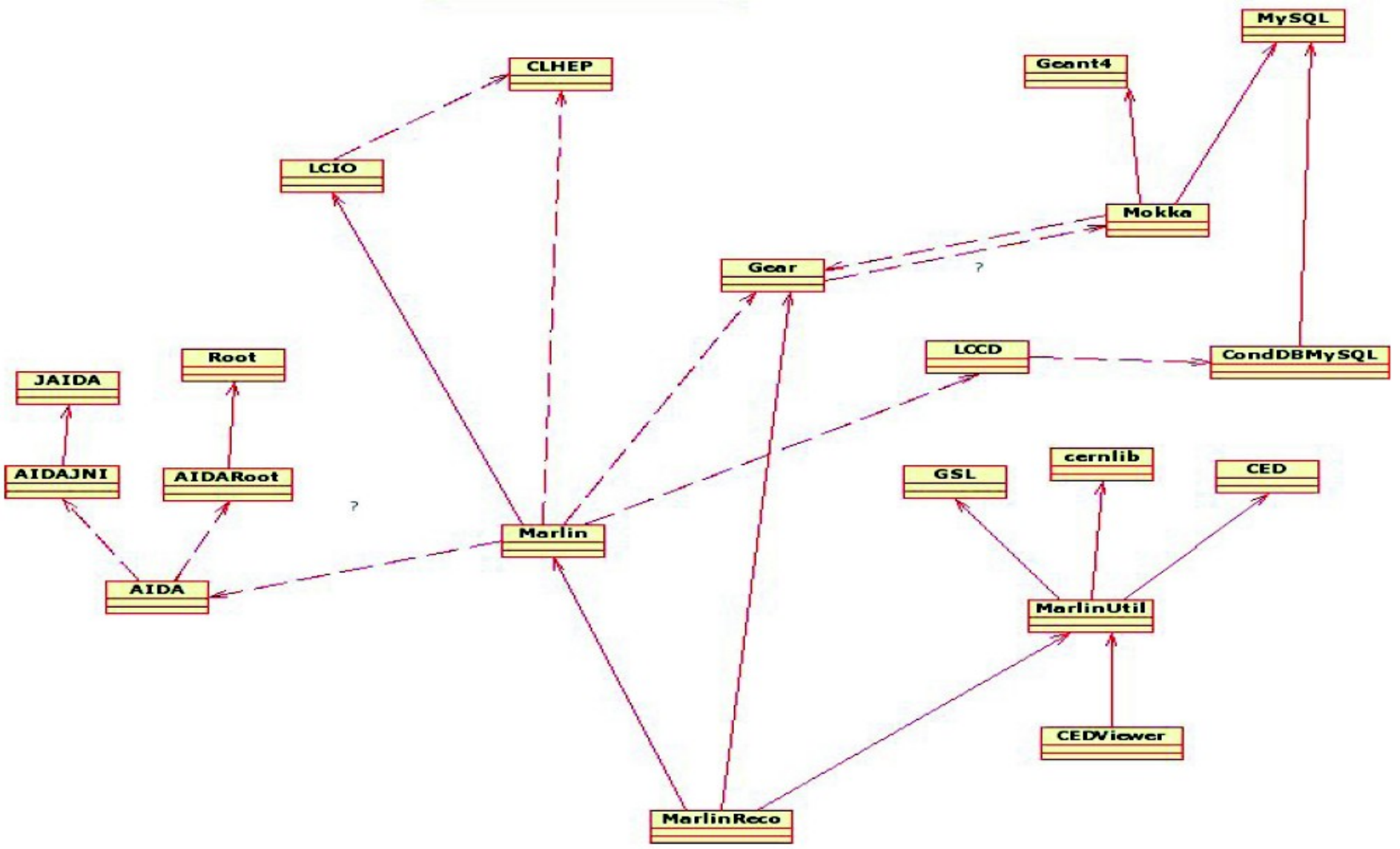
Automating Software Installation

- Automating the installation of a large software framework can be the first step to overcome some of the problems listed before! If properly done it can do some of the following:
  - Automatic checking of dependencies
  - Greatly reduce complexity and installation times
  - Installations without any user-intervention
    - Crucial for performing nightly builds
  - Integrated Testing

Less maintenance/support costs

Increase of productivity!

ILC Software Package Dependencies



ILC Software Installation

- **ILCInstall**
  - Python script for installing LDC Software framework
    - LCIO, GEAR, LCCD, Marlin + modules ...
  - Support for external packages
    - CLHEP, GSL, CERNLIB, QT, ...
  - Automatic checking of dependencies
  - Install whole framework without user-intervention
  - Fully configurable (via configuration file)
    - Versions
    - Download related stuff
    - Use/Link/Install packages
    - Dynamic assignment of environment/cmake variables
  - Modular & Extendable
    - Abstract class for Marlin modules
  - Recently changed to use CMake as the build manager

ILC Software Releases (AFS)

- **/afs/desy.de/group/it/ilcsoft/**
  - **CLHEP/**
    - 2.0.2.1
    - 2.0.2.2
  - **GSL/**
    - 1.6
    - 1.8
  - **CERNLIB/**
    - 2005
    - 2006
  - **CondDBMySQL/**
    - ILC-0-5-10
  - **...**

Software with **no** dependencies

Software **with** dependencies

- **/afs/desy.de/group/it/ilcsoft/v01-01/**
  - **CLHEP/**
    - 2.0.2.2 -> link to ../CLHEP/2.0.2.2
  - **GSL/**
    - 1.8 -> link to ../GSL/1.8
  - **CERNLIB/**
    - 2006 -> link to ../CERNLIB/2006
  - **Marlin/**
    - V00-09-09
  - **MarlinUtil/**
    - V00-04
  - **MarlinReco/**
    - V00-04
  - **...**

Linking against Releases in AFS

- You can easily modify a release configuration file to your needs, for example if you want to install PandoraPFA locally and link it against the ILC Software release "v01-01" in AFS for SL3 you can do it with the following steps:
  - Download ilcinstall:
    - http://www-zeuthen.desy.de/lc-cgi-bin/cvsweb.cgi/ilcinstall/?cvsroot=ilctools
  - Copy from directory "releases" configuration file "release_v01-01_sl3.cfg"
  - Modify it by:
    - Installing only the packages you want to work on:
      - ilcsoft.install(PandoraPFA("HEAD"))
      - ilcsoft.module("PandoraPFA").download.type="ccvssh"
      - ilcsoft.module("PandoraPFA").download.username="engels"
      - ilcsoft.module("PandoraPFA").download.password="******"
    - Change dependencies from install to link:
      - ilcsoft.link(LCIO(ilcPath+ "lcio/v01-08-04"))
      - ilcsoft.link(Marlin(ilcPath+ "Marlin/v00-09-09"))
      - ...

Build with ILCInstall generated file (ILCSoft.cmake): cmake -C ../../../ILCSoft.cmake ..

Build-System Manager

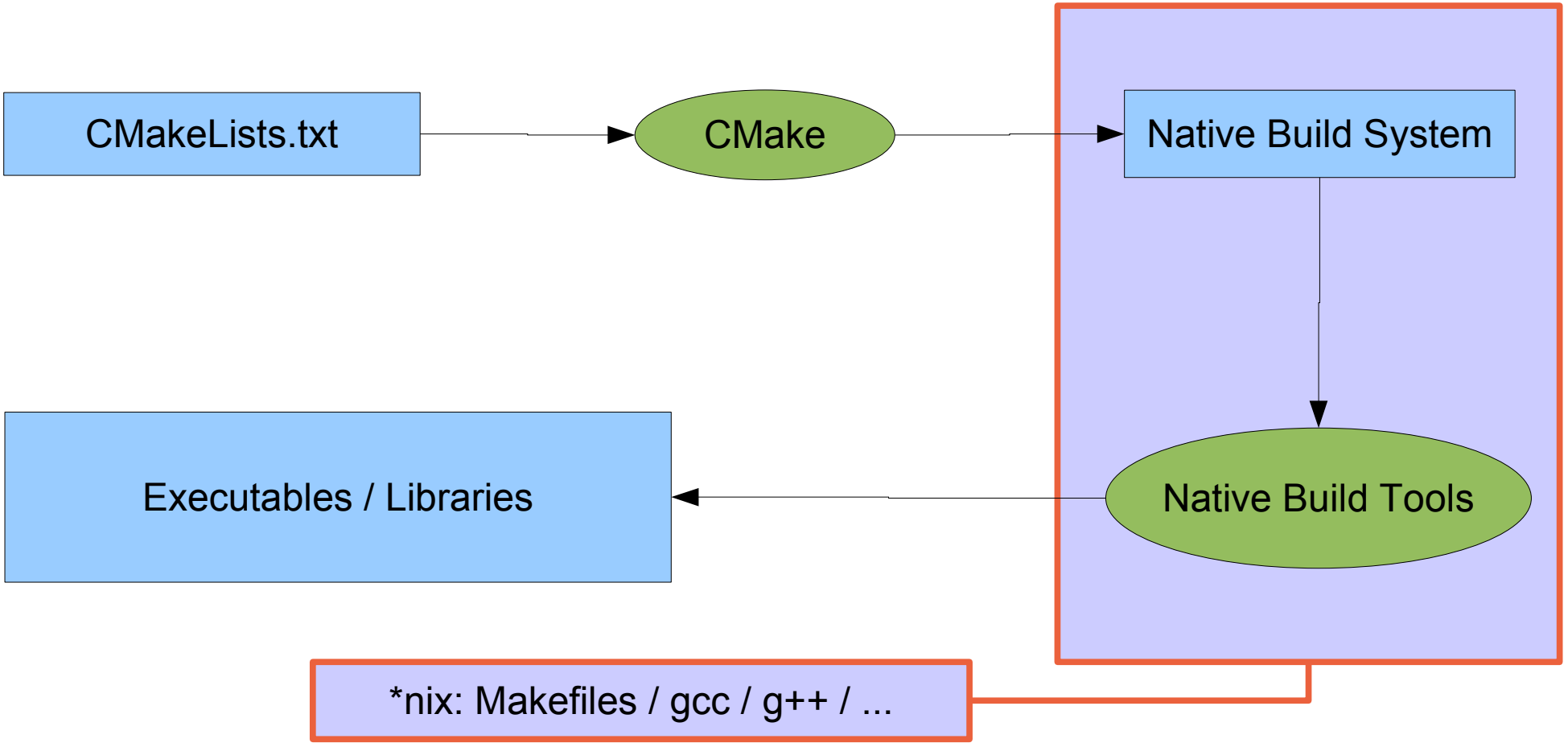
- One of the most "*annoying problems*" we have in the word of computer science is having such a heterogeneous environment of hardware architectures / operating systems / software compilers / ...
  - We need standards!
  - We need different levels of abstraction!
- Typical users are not interested in having to deal with software dependencies or hardware/platform-dependent build issues
- Administrators want to spend as little time as possible dealing with these kinds of problems...

CMake

- **CMake**:
  - Cross-platform build-system manager
  - Generates native build environments
    - UNIX/Linux -> Makefiles
    - Windows -> VS Projects/Workspaces
    - Apple -> Xcode
  - Takes as input simple text files (CMakeLists.txt)
    - Very simple, intuitive syntax!
    - Support for regular expressions (*nix style)
  - In-source vs Out-of-source builds
  - Flexible, extensible & powerful
    - Macros, Modules for finding/configuring software, custom targets ....
  - Integrated Testing & Packaging (Ctest, CPack)
  - Open-Source :)
  - Check www.cmake.org!

Successor of GNU auto-tools

CMake's Build-System Generator



CMake Modules

- CMake Modules
  - FindJava.cmake
  - FindZLIB.cmake
  - FindQt4.cmake
  - ...

- Variables returned by these modules:
  - *ZLIB_INCLUDE_DIRS* "/usr/include"
  - *ZLIB_LIBRARIES* "/usr/lib/libz.so"
  - *ZLIB_DEFINITIONS* "-DUSE_ZLIB"

- Use the variables in your project for building/linking against the desired modules!

CMake Trees & Builds

- The **Source Tree** contains:
  - CMake input files (CmakeLists.txt)
  - Source && Header files (*.cc / *.h)

- The **Binary Tree** contains:
  - Build system files (Makefiles)
  - Build-Output files:
    - Libraries
    - Executables
    - Any other build generated file

- Source and Binary trees may be:
  - In the **same directory** (**in-source** build)
  - In **different directories** (**out-of-source** build)

Using CMake

- Create a build directory ("**out-of-source** build" concept)
- Configure the package for your system
- Build the package
- Install it!

Similar to Auto Tools

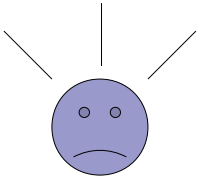
- **mkdir build ; cd build**
- **cmake [options] <source_tree>**
  - *e.g. cmake -C ../**BuildSetup.cmake** ..*
- **make**
- **make install**

Script containing all important stuff for configuring a package.
Edit & change according to your system before calling cmake!

- The last 2 steps can be merged into one:
  - "**make install**"

Global file for defining paths

Set *CMAKE_MODULE_PATH* in **10 different BuildSetup.cmake** files??!!



- CMake can use **more than one** -C option:
  - *cmake -C ../BuildSetup.cmake -C ~/ILCSoft.cmake ..*
- Latest file overwrites values from previous file
- Use for defining **paths** (<PKG>_HOME variables)
- Check /afs/desy.de/group/it/ilcsoft/v01-01/ILCSoft.cmake

Adapting a Marlin Package to CMake

- Prerequisites in Marlin README!

- Copy 4 files from *$MARLIN/examples/mymarlin*
  - **CmakeLists.txt**
    - change the project name and add missing dependencies
      - PROJECT(mymarlin)
      - SET(${PROJECT_NAME}_DEPENDS "Marlin;LCIO;GEAR;..."
  - **mymarlinConfig.cmake.in**
    - rename to <MyProcessor>Config.cmake
  - **BuildSetup.cmake**
    - Edit & change according to your system setup!
  - **cmake_uninstall.cmake.in**
    - No changes needed!

CMake with ILCInstall

```
### RAIDA.cfg ########################
# configuration file for installing RAIDA
# into "/data/ilcsoft/RAIDA/v01-03"
#####################################
ilcsoft = ILCSoft("/data/ilcsoft")

ilcsoft.useCMake = True

# install RAIDA v01-03
ilcsoft.install( RAIDA( "v01-03" ))

# example for setting cmake build variables
ilcsoft.module( "RAIDA" ).envcmake["BUILD_RAIDA_EXAMPLE"] = "ON"

# link ROOT
ilcsoft.link( ROOT( "/afs/desy.de/group/it/ilcsoft/root/5.08.00" ))

# CMake Modules
ilcsoft.install( CMakeModules( "v01-02" ))

# End of configuration file
```

Here we tell ILCInstall to use CMake

Make sure cmake is in your **$PATH**

Afterwards run script with:
ilcsoft-install -i RAIDA.cfg

Status of the ILCSoft Install/Build-System

ILCInstall:

- Current version: v01-02-01
- Currently Supported modules:
  - LCIO, GEAR, LCCD, RAIDA, Marlin, MarlinUtil, MarlinReco, CED, CEDViewer, PandoraPFA, LCFIVertex, SiliconDigi, Overlay, Eutelescope, Mokka, CLHEP/HepPDT, GSL, CERNLIB, CondDBMySQL, QT, CMake, CmakeModules, ROOT, Geant4, AIDAJNI, JAIDA, Java, MySQL

Recently added
Improved
No install/build support

CMake:

- CmakeModules - Current version: v01-02
- Currently Supported:
  - LCIO, GEAR, LCCD, RAIDA, Marlin, Overlay, MarlinUtil, MarlinReco, CED, CEDViewer, PandoraPFA, LCFIVertex, SiliconDigi, Eutelescope, CLHEP/HepPDT, CERNLIB, CondDBMySQL, GSL, ROOT, Java, AIDAJNI/JAIDA

Summary & Outlook

- **Summary:**
  - Most of the LDC Software can already be installed in a fully configurable & automated way!
  - Nightly builds already being done for testing the software under SL3
  - Shared libraries & Marlin plugin's support with CMake
  - Reached higher levels of Hardware/Platform-abstraction with CMake
- **Outlook**:
  - New release of ILCSoft available at AFS soon! (v01-02)
  - More elaborated integrated testing (Ctest)
    - Regression tests, Black box testing, Test Dashboards...
  - Nightly builds for SL4
  - Installation of LDC Software on the Grid
  - Interface for ilc-job-submissions to the Grid
  - Support for MarlinTPC, Calice ??

Thank you! Your feedback is welcome!