Hepsoft - an approach for up to date multi-platform deployment of HEP specific software

S. Roiser
European Organization for Nuclear Research CERN
E-mail: stefan.roiser@cern.ch

Abstract.
LHC experiments are depending on a rich palette of software components to build their specific applications. These underlying software components include the ROOT analysis framework, the Geant4 simulation toolkit, monte carlo generators, grid middle-ware, graphics libraries, scripting languages, databases, tools, etc. which are provided centrally in up to date versions on multiple platforms (Linux, Mac, Windows). Until recently this set of packages has been tested and released in a tree like structure as a consistent set of versions across operating systems, architectures and compilers for LHC experiments only. Because of the tree like deployment these releases were only usable in connection with a configuration management tool which provided the proper build and run-time environments and was hindering other parties outside LHC from easily using this palette of packages.

In a new approach the releases will be grouped in "flat structure" such that interested parties can start using it without configuration management, retaining all the above mentioned advantages. In addition to an increased usability the software shall also be distributed via system provided package deployment systems (rpm, apt, etc.). The approach of software deployment is following the ideas of providing a wide range of HEP specific software packages and tools in a coherent, up to date and modular way on multiple platforms. The target audience for such software deployments are individual developers or smaller development groups / experiments who don’t have the resources to maintain this kind of infrastructure. This new software deployment strategy has already been successfully implemented for groups at CERN.

1. Introduction
This paper provides an idea on how to possibly improve the software deployment infrastructure for the common software layers currently used by LHC experiments and at CERN in general. The organisation of the AA software stack will be discussed in section 2 followed by section 3 concentrating on its current usage. In section 4 a proposal will be discussed on how this deployment can be changed in order to facilitate its usage also outside LHC. Sections 5 on the current status of this new deployment will conclude this paper.

2. Organisation of the AA software stack
The Applications Area (AA) [1] of the LHC Computing Grid project (LCG) [2] provides coordination and installation of the common software stack for LHC experiments such as Atlas and LHCb and is divided into three parts (see Figure 1).
• **Experiment specific software** such as analysis and reconstruction software. These applications are provided by the LHC experiments themselves and are building on top of the other two layers provided by AA.

• **AA projects** are developed within AA. This software packages typically provide special or optimized functionality for the needs of High Energy Physics experiments. Examples are the ROOT data analysis framework [3], the COOL conditions database, the CORAL relational database abstraction layer and the POOL persistency project [4].

• **AA externals** are software packages which are taken from external sources and re-compiled for the AA provided platforms. Currently the total number of these packages is 102 and the functionality provided within ranges from mathematical libraries, scripting languages, databases, I/O, compilers, tools, etc.

All these software layers are furthermore provided on different operating systems (Scientific Linux CERN [5], Mac OSX, Windows), architectures (32 and 64 bit) and compilers (gcc, icc, llvm, VC).

### 2.1. Special needs for the AA software stack

The AA software stack is special in several circumstances. Below a list of these special needs is provided

• The software stack shall produce **relocatable binaries**. This is needed because the binaries are used in several different environments, such as a centrally available file-system (Andrew File System - AFS), are shipped on the computing grid for use within different batch farms and shall be usable for individual users on their personal computers. Each of these installations can be ending up in a different root directory and as such the installation e.g. must not contain any hard-coded paths.

• All software packages that are combined to a common software release are specified by **exact version numbers**. This policy for AA projects was developed in order to know at all times the exact state of the software stack. This is needed e.g. to know which bug fixes are provided for certain packages.

• The whole software stack is provided on **multiple platforms** whose constituents are several operating systems, architectures, compilers and optimization levels. This implies that an effort has to be done to recompile all packages an all platforms whenever possible. In general
Another policy of AA is the provision of **rebuildable packages** with minimal effort also outside the AA environment. This is currently implemented for all linux and mac platforms and allows e.g. users to rebuild certain software packages outside the AA environment or to rebuild the whole software stack for their preferred e.g. linux distribution.

- The list of packages provided by AA is not complete in the sense that some parts will be taken from the underlying operating system. As a consequence of this policy there is a **fuzzy frontier to the operating system** layer. This means that whenever necessary packages are moved from the OS layer to AA or vice versa. There are packages which need to be controlled at all times, e.g. version of the Gnu Scientific Library [6]. Other packages are provided within AA because they are either not provided on all platforms or in different / too old versions, for example the Qt [7] graphical widget tool kit.

### 3. Current use of AA software by clients

The common software layers (see Figure 1) are provided to the experiments in so called “LCG Configurations” [3]. A LCG Configuration denotes a release of all common software, specifying the exact package version of each constituent and platforms/architectures provided. “LCG Configurations” are provided on demand by LHC experiments which are discussed in a bi-weekly steering meeting. A major new release series is started whenever there are substantial changes in the software stack, such as changing major versions of key components (e.g. Python [9], ROOT [3], etc.). On top of major releases usually a series of bug-fix releases are done. In the past there were usually 2-3 major releases per year with 5-6 patch releases in each series.

The deployment of the whole AA software stack is done in a tree structure. Inside a root directory the next levels are the package name, the version number and the platform. From there on the “usual” package installation starts, e.g. with the bin, lib and share directory (see Fig 2). In order to provide a proper run-time environment to the user a configuration management tool is needed to put together all necessary environment variables (e.g. PATH, LIBPATH, PYTHONPATH etc.). Without this tool, currently we use CMT [10, 11]), the user will not be able to have a proper run-time environment being setup. Due to the nature of the software deployment the individual entries in the directory tree have to be concatenated and
this can result in long values for each of the necessary variables, making it also hard to manage them. With some shells the maximum allowed number of characters for environment variables has even been exceeded with this approach.

4. Proposal for changing the software deployment infrastructure

4.1. Flat deployment

Following the restrictions currently observed using a tree like software deployment, a new deployment strategy of AA software is being proposed. This deployment will gather all software packages in a flat infrastructure and provide one directory for libraries, binaries, etc. (see Fig 3). The advantage of this deployment will be the easier handling of the installation by the user as no special configuration management tool will be needed to setup compile or run-time environments. Furthermore even some special environment variables (e.g. PYTHONPATH) can be skipped completely as all python modules will be grouped together with the python installation and found automatically.

In the previous tree like deployment a package could have been used in several releases if its version number has not changed and no dependent package did. In this new infrastructure the packages will need to be deployed for each release again which will result in an increased need of storage space. On the other hand the deployment of all packages again for each release will guarantee the proper handling of dependencies between packages which could have been omitted in the previous case.

Another simplification for the user will be that he is not bound to a special configuration management tool for building his own software but can use whatever build tool he prefers.

4.2. Combining flat and directory deployment

To overcome the remaining issue of space duplication a further extension to the deployment can be developed (see Fig 4). In this scenario the packages would be again deployed in a tree like structure as it is currently being done. On top of this tree like structure a thin layer for the software deployment interface to the user will be provided. This thin layer will contain the flat directory structure as discussed above and the necessary files (binaries, headers, etc.) will be hard-linked into this structure from the tree like installation.

This way one could benefit from the fact that software packages can be re-used and also changes to the deployment can be done easily by un-linking, re-linking any changes e.g. for new

Figure 3. Proposed flat deployment of AA software stack
package versions. On the other hand the maintenance of such a deployment will again require some tool to maintain such an infrastructure, but such a tool will not be linked to the way a user will use the software installation. Deployed to the user in a flat way it can still be used with any preferred software building tool and also the run-time environment will stay easy to be setup.

5. Current status
The idea of a flat software deployment has been tried out in a prototype installation. This installation “HEPSOFT” [12] is currently being used by the CERN Theory group both in an installation in the Andrew File System (AFS) and CERNVM [13].

6. Conclusion
This paper provides an idea of improving the way how to install software packages for the HEP community and especially the software packages used for the LHC experiments. A prototype of such an installation is currently available and in production by users at CERN.

References
[1] Roiser S and Pfeiffer A 2007 Configuration, Build and Distribution of LCG Applications Area Software for the LHC Experiments (2007 IEEE Nuclear Science Symposium and Medical Imaging Conference)
[2] Worldwide lhc computing grid http://lcg.web.cern.ch/LCG/
[3] Root http://root.cern.ch/root
[4] Dullmann D et al. 2003 The LCG POOL Project General Overview and Project Structure (2003 Conference for Computing in High Energy and Nuclear Physics (CHEP))
[5] Scientific linux cern http://linux.web.cern.ch/linux/
[6] The GNU scientific library (GSL) http://www.gnu.org/software/gsl
[7] Qt http://qt.nokia.com/
[8] Lcg configurations http://lcgsoft.cern.ch
[9] Python http://www.python.org
[10] Arnault C 2000 CMT : a Software Configuration Management Tool (2000 International Conference on Computing in High Energy and Nuclear Physics)
[11] CMT http://www.cmtsitem.org
[12] Hepsoft /afs/cern.ch/sw/lcg/hepsoft
[13] Cernvm http://cernvm.cern.ch

Figure 4. Combining tree and flat deployment