

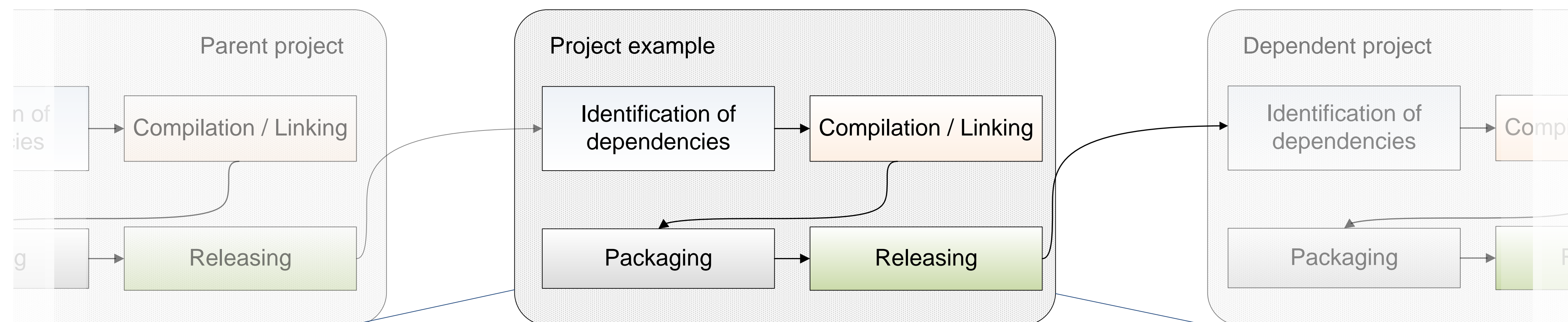
A C/C++ Build System Based on Maven for the LHC Controls System

J. Nguyen Xuan, B. Copy, CERN, Geneva, Switzerland
 M. Dönszelmann, Bogazici University, Istanbul, Turkey

Introduction

The CERN accelerator controls system, mainly written in Java and C/C++, consists nowadays of 50 projects and 150 active developers. The Controls group has decided to unify the development process and standards (e.g. project layout) using Apache Maven and Sonatype Nexus. Maven is the de-facto build tool for Java, a plugin (Maven NAR) adapts the build process for native programming languages for different operating systems and platforms. Our approach was to combine the best of the two worlds: NAR/Nexus and Makefiles. Maven NAR manages the dependencies, the versioning and creates a file with the linker and compiler options to include the dependencies. The Makefiles carry the build process to generate the binaries. Finally the resulting artifacts (binaries, header files, metadata) are versioned and stored in a central Nexus repository.

Workflow and concept of reuse



Let's look at those 4 steps with a concrete example, using Maven, Nexus and Makefiles

Identification of dependencies

```

1 <project>
2   <groupId>cern.cmw.cpp</groupId>
3   <artifactId>cmw-rbac</artifactId>
4   <packaging>nar</packaging>
5   <version>3.7.0</version>
6
7   <dependencies>
8     <dependency>
9       <groupId>cern.cmw.cpp</groupId>
10      <artifactId>cmw-serializer</artifactId>
11      <version>[1.0.0,2.0.0)</version>
12      <type>nar</type>
13    </dependency>
14    <dependency>
15      <groupId>cern.cmw.cpp</groupId>
16      <artifactId>cmw-util</artifactId>
17      <version>[1.0.0,2.0.0)</version>
18      <type>nar</type>
19    </dependency>
20    <dependency>
21      <groupId>com.zeroc</groupId>
22      <artifactId>IceUtil</artifactId>
23      <version>3.2.3</version>
24      <type>nar</type>
25    </dependency>
26    <dependency>
27      <groupId>org.boost</groupId>
28      <artifactId>boost</artifactId>
29      <version>1.33.1</version>
30      <type>nar</type>
31    </dependency>
32  </dependencies>
33 </project>
  
```

A project is described by a file called **pom.xml** (Project Object Model). It contains metadata along with a list of dependencies. The project is identified by a GAV (groupId, artifactId, version).

This particular pom belongs to a C++ project. What differs it from a Java project is the type of the packaging and the dependencies: **NAR**.

1. mvn nar:makedep

Compilation / Linking

```

1 CXX = /usr/bin/g++
2 DEPENDENT_COMPILER_OPTIONS += -I$(R)cmw-serializer-1.3.1-noarch/include \
3 -I$(R)cmw-util-1.2.3-noarch/include \
4 -I$(R)IceUtil-3.2.3-noarch/include \
5 -I$(R)boost-1.33.1-noarch/include
6
7 DEPENDENT_LINKER_OPTIONS += -L$(R)cmw-serializer-1.3.1-1386-SLC5-gpp-static/lib/1386-SLC5-gpp-static \
8 -L$(R)cmw-util-1.386-SLC5-gpp-static/lib/1386-SLC5-gpp-static \
9 -L$(R)IceUtil-3.2.3-1386-SLC5-gpp-static/lib/1386-SLC5-gpp-static \
10 -Lcmw-serializer-1cmw-util-1cmw-util-1cmw-util
  
```

The generated Makefile contains the **compiler and linker options**. It has to be included by the main Makefile which handles the compilation process.

```

Name      Size      Type
cmw-rbac  9 items  folder
AccessManager.h  5.5 KB  C header
AccessMgr.h      4.2 KB  C header
AdminUserReport.h  4.0 KB  C header
CheckingMgr.h    4.9 KB  C header
CredentialHeader.h  1.8 KB  C header
LogContext.h     5.9 KB  C header
mainHeader.h     1007 bytes C header
mainMgr.h        6.6 KB  C header
Transaction.h    2.1 KB  C header
lib        1 item  folder
LIBS        1 item  folder
libcmw-rbac.a  3.4 MB  AR archive
  
```

The results of the build are put in a specific folder. The **naming and the directory structure** must be respected.

2. mvn compile

Packaging

Name	Size	Type
cmw-rbac-3.7.0-noarch.nar	18.7 KB	ZIP archive
cmw-rbac-3.7.0-1386-SLC5-gpp-static.nar	864.6 KB	ZIP archive

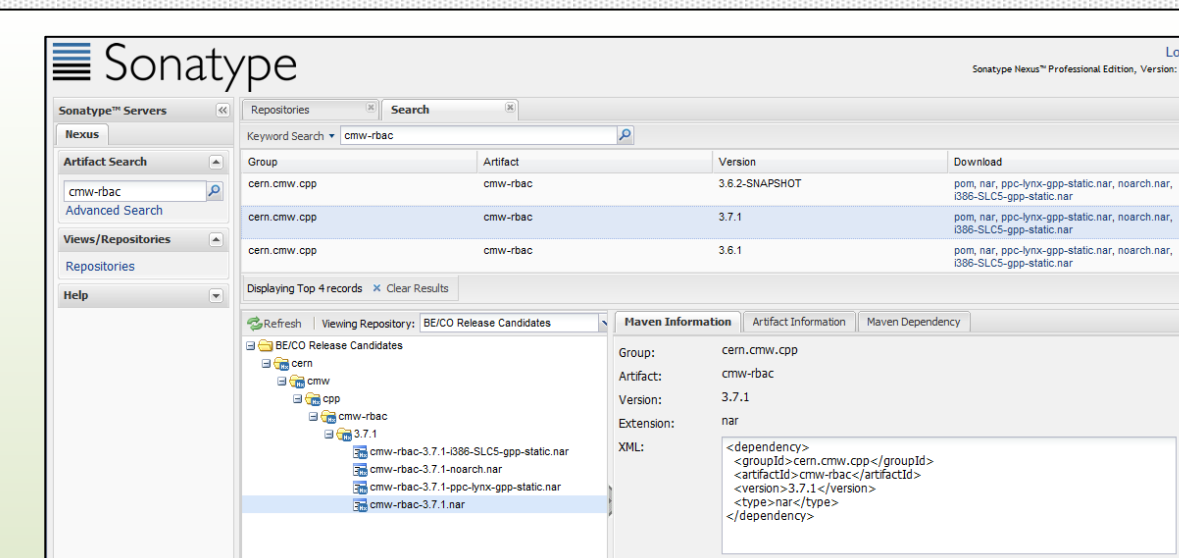
With Maven NAR, mainly **two types** of artifacts are typically published:

- 1) Public headers
- 2) Platform dependent Library or Binary

Several platform dependent artifacts can be published. When a Maven build is triggered, the right artifact for the targeted platform will be automatically picked.

3. mvn package

Releasing



A **binary repository** (Maven standard Sonatype Nexus) hosts all the versioned artifacts. A web interface allows **searching** for available artifacts.

4. mvn deploy

Maven goals:

1. **mvn nar:makedep** - This goal from the NAR plugin **downloads** and **unpacks** the dependencies locally, and **generates a Makefile** with the dependencies information.
2. **mvn compile** - This goal relies on Makefiles. It simply calls **make** to start the build process.
3. **mvn package** - Thanks to the **strict directory structure**, Maven NAR knows where to pick up the different binaries and packages them into several **artifacts**.
4. **mvn deploy** - **Uploading** the artifacts to Nexus.

Comparison of this implementation based on Maven with the previous implementation based on GNU Makefiles

	Maven NAR	Makefiles implementation
Cross-compilation	✔ Custom compilers can be defined	✔ Well supported by Makefiles
Dependencies management	✔ Automatically managed	✘ Managed by hand
Versioning	✔ A version must be defined in the pom.xml	✘ Not existing or manually managed
Directory standards	✔ Conventions are required	✘ Not existing

Conclusions

Early experiments were conducted in the scope of the Controls group's Testbed. Some existing projects have been successfully converted to this solution and some projects start from scratch using this implementation. Using the same build tool as the Java developers allows the C/C++ teams to benefit from the same infrastructure (continuous integration server, binary repository,...) without changing their habits with Makefiles. Standards are also enforced since the developers have to follow the strict conventions for directories naming and structure. Moreover, the development/release/deployment process is unified for the Java and the C/C++ teams.

Additional features have to be implemented to fit the developers requirements such as the possibility to use a framework to run unit tests. We plan to integrate the CERN modifications back to the official NAR plugin in order to contribute to its community.