LP DAM - GL

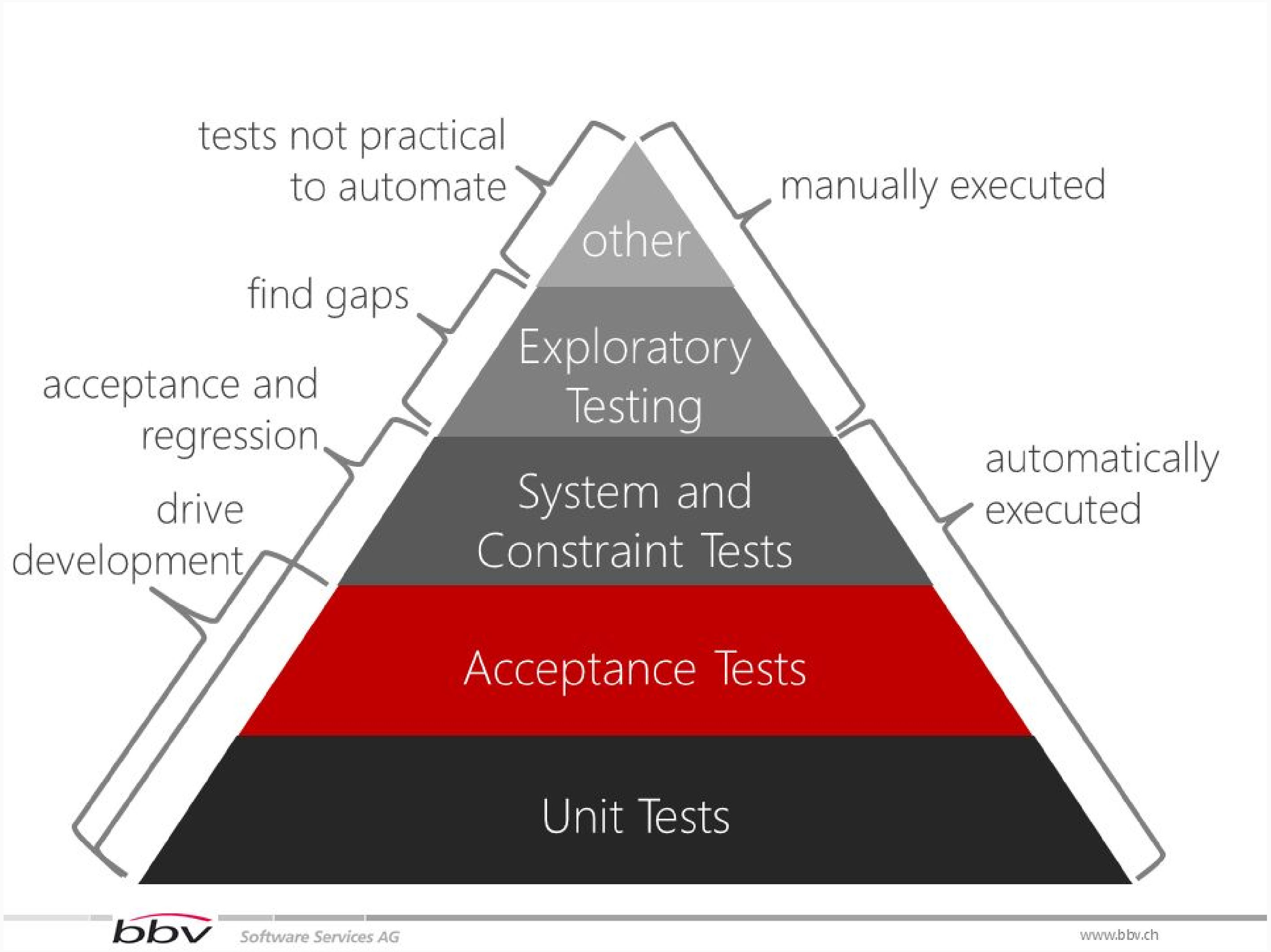
Packaging, automation

Based on Simon Urli and Sébastien Mosser courses

05/12/2016

Cécile Camillieri

Spoiler alert…



Development process

Develop features and tests

↓

Launch tests

↓

Develop…

↓

Launch tests…

↓

Release and/or deployment
of a stable version

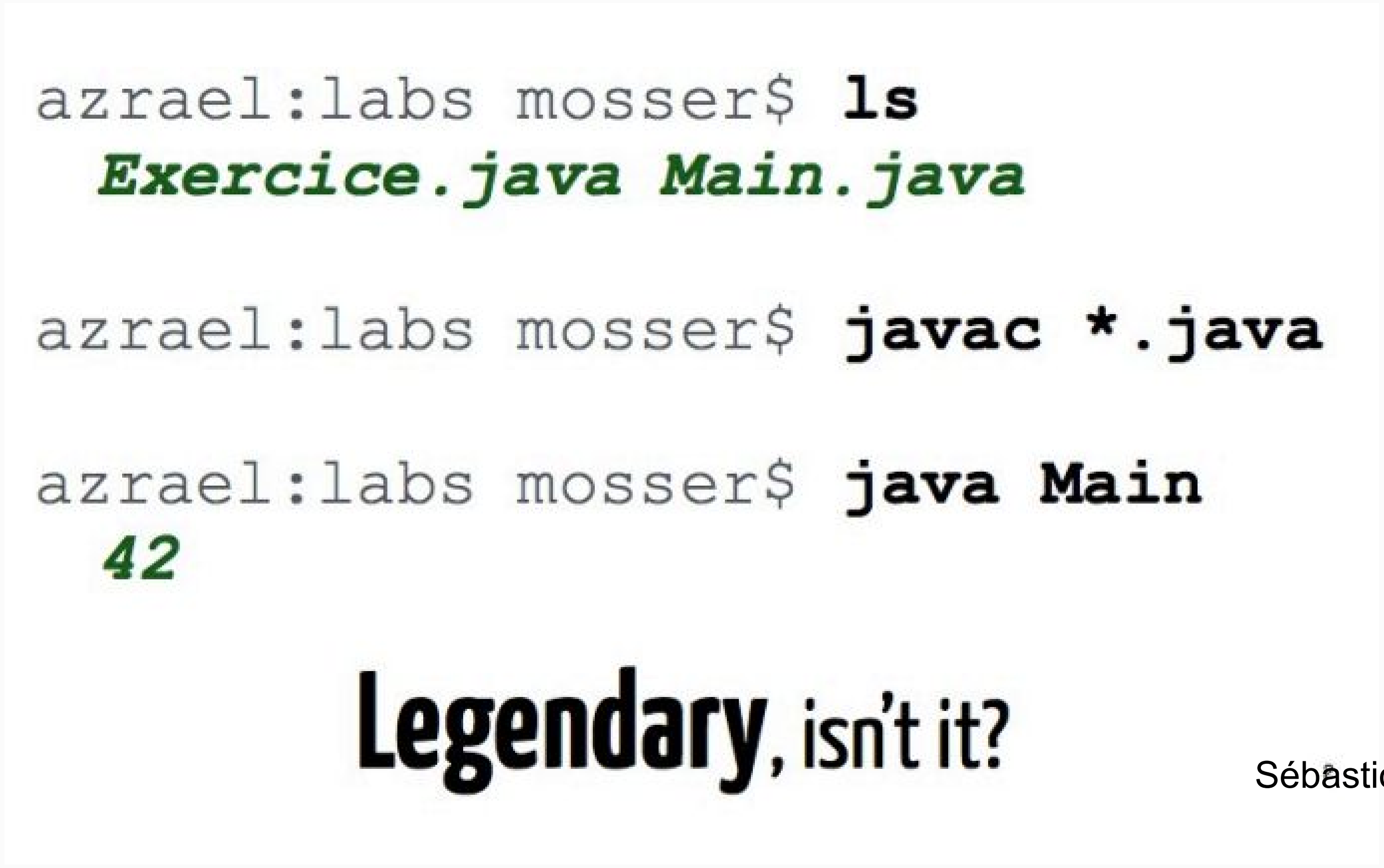
Development process

Develop features and tests	**code**
↓	
Launch tests	**commands**
↓	
Develop…	**code**
↓	
Launch tests…	**commands**
↓	
Release and/or deployment of a stable version	**commands**

Development process

Develop features and tests	**code**
↓	
Launch tests	**commands -> script?**
↓	
Develop…	**code**
↓	
Launch tests…	**commands -> script?**
↓	
Release and/or deployment of a stable version	**commands -> script?**

Compiling during lab sessions



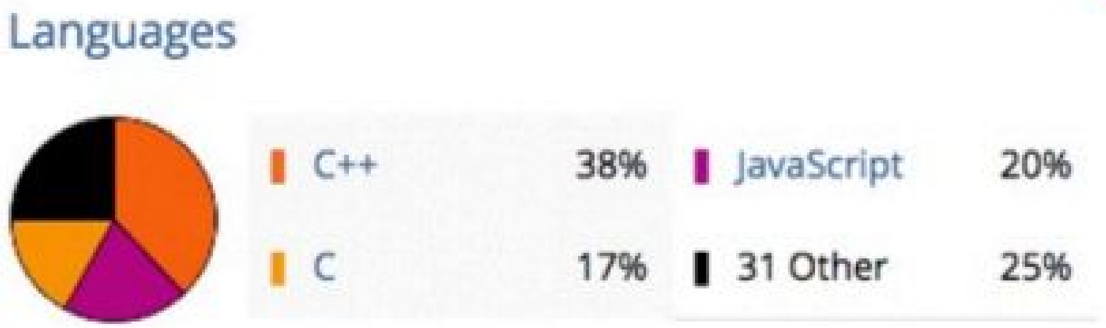
```
azrael:labs mosser$ ls
  Exercice.java Main.java

azrael:labs mosser$ javac *.java

azrael:labs mosser$ java Main
  42
```

Legendary, isn't it?

Sébastion Mosser

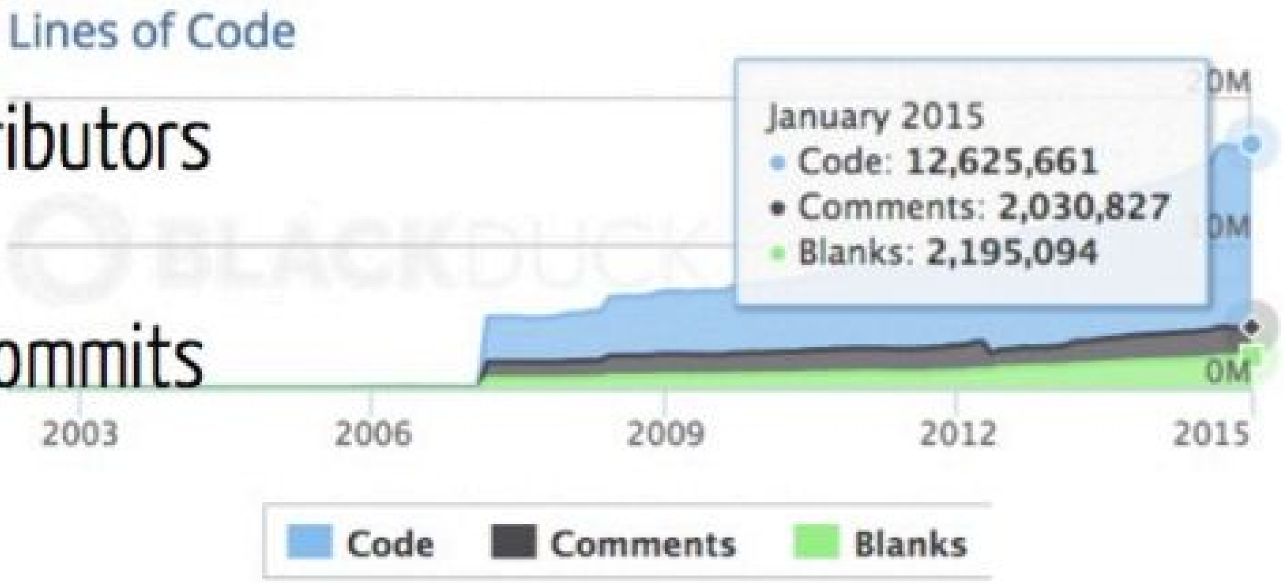
How far can you go like this ?

Code in Real-Life™



3,254 contributors

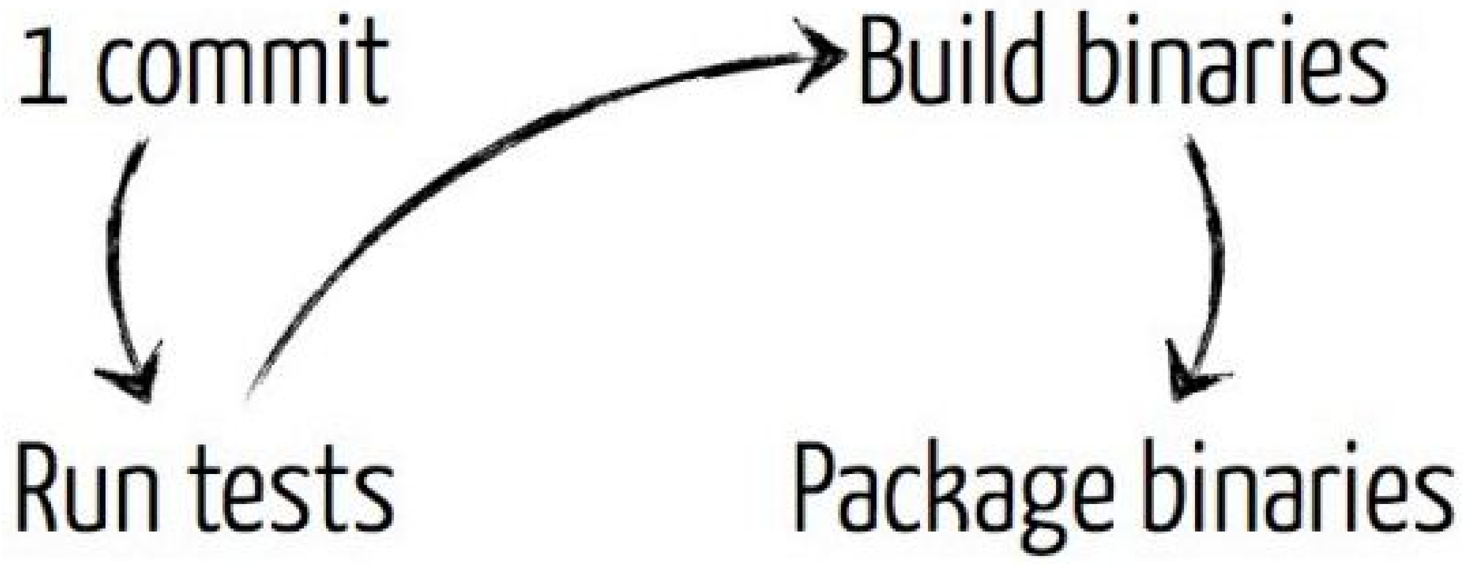
230,442 commits



http://www.ohloh.net/p/firefox

4

Ok, That's Impressive… So What?

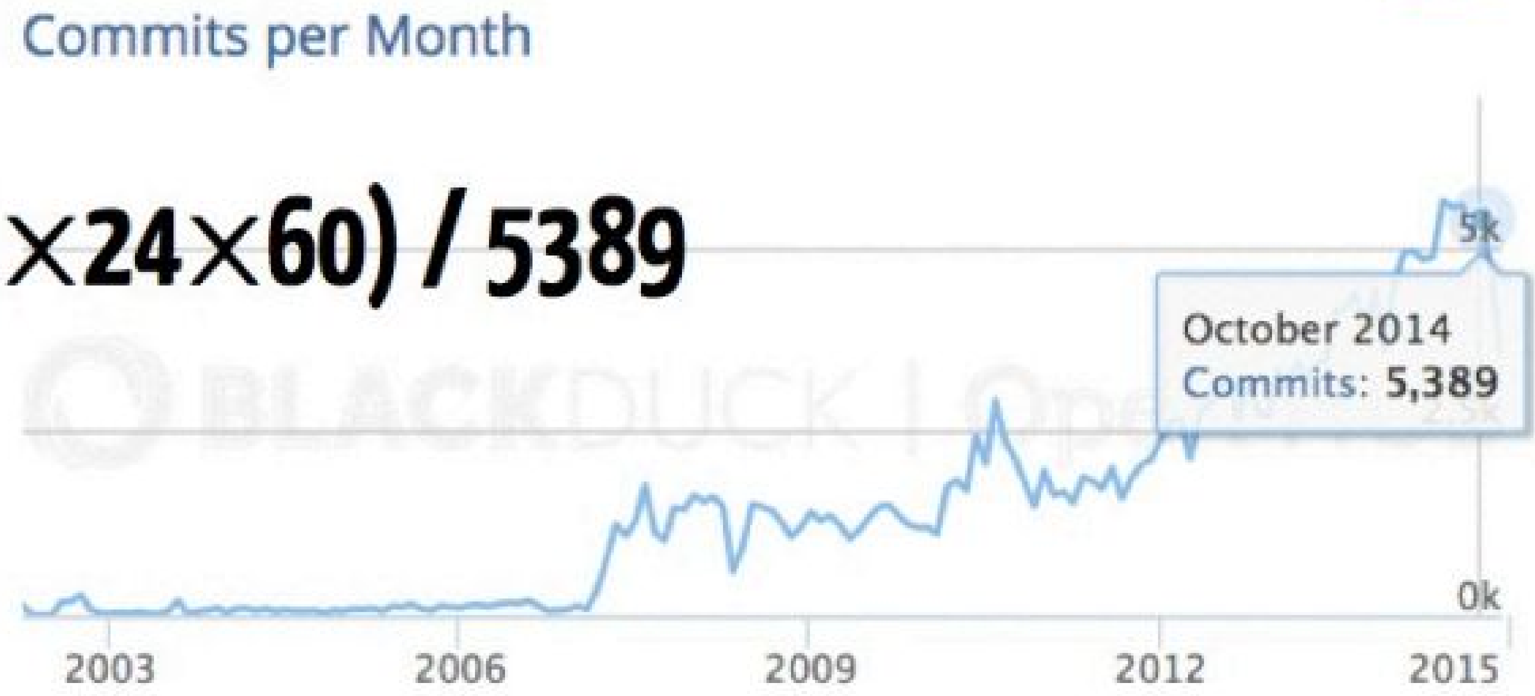


Ok, That's Impressive... So What?

$$\begin{tikzcd}
\text{1 commit} \arrow[d] & \text{Build binaries} \arrow[d] \\
\text{Run tests} \arrow[ur, bend left] & \text{Package binaries}
\end{tikzcd}$$

137 hours! (08.2012)

4403 Commits Last Month!



(31×24×60) / 5389

~1commit each 8 minutes

http://www.ohloh.net/p/firefox

6

One **cannot** do it by hand

Towards automation

Automate = **no human intervention**

=> Speed

=> Avoid mistakes

=> Concentrate on the code

« Lorsque vous écrivez une commande plus de trois fois, pensez à l'automatiser. »

- Raphaël Marvie

What to automate?

- Creation of a new project: structure, configuration, …
- Creation of a new element: class, resource file, …
- Tests execution
- Publication of tests results
- Packaging: creation of .jar, .zip, …
- Deployment of software
- Documentation creation
- Documentation deployment
- ...

Maven

What’s maven?

- Descendant of Make and Ant
- Similar to Gradle

- Objectives

  => Handle dependencies

  => Automate tasks (compile, test, …)

- Configuration with a **pom.xml** file

- JS equivalent:

  npm for dependencies, grunt for tasks

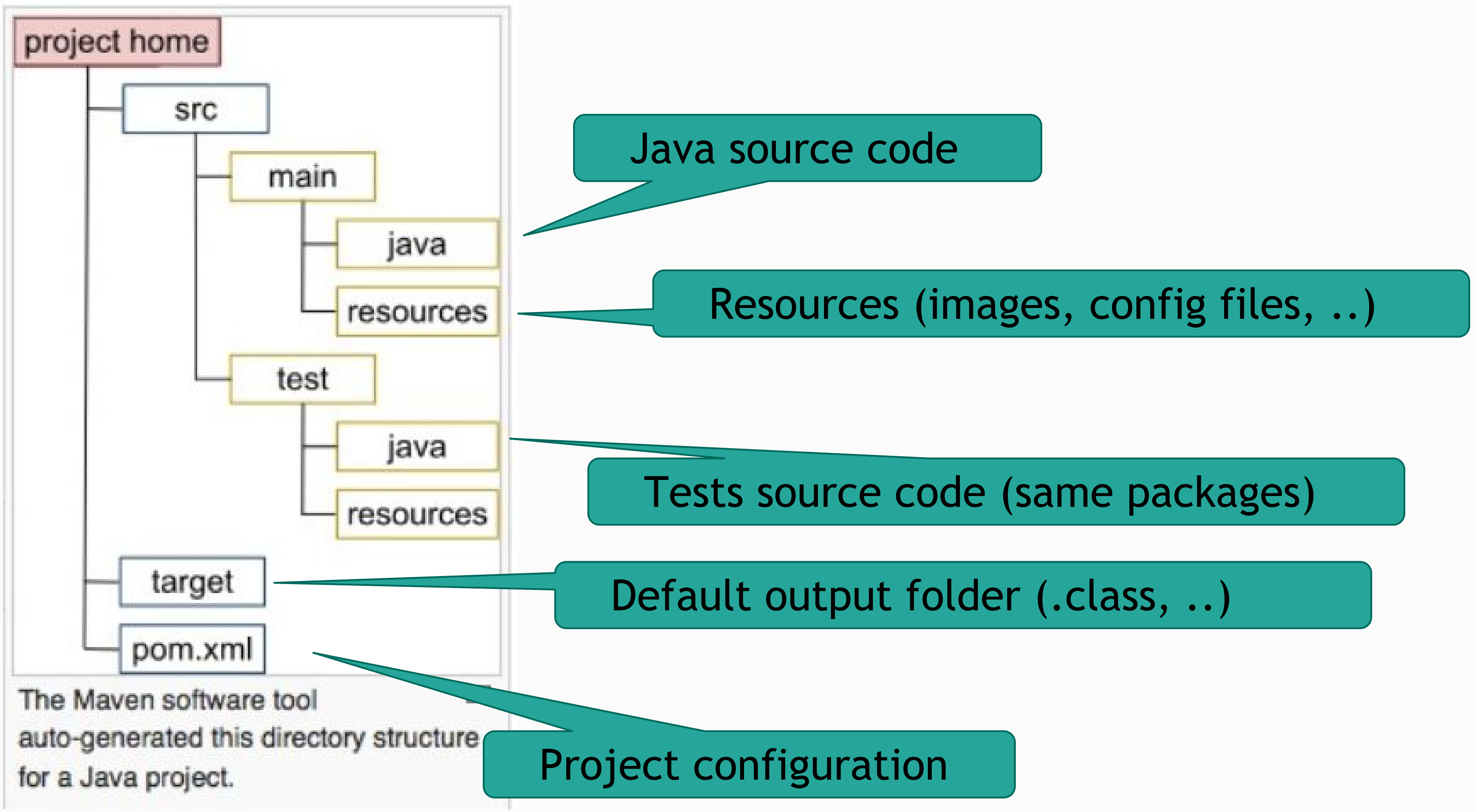
Creating a maven project

- Choose a predefined archetype
- In a command line:

```
mvn -B archetype:generate \
-DarchetypeGroupId=org.apache.maven.archetypes \
-DgroupId=com.mycompany.app -DartifactId=my-app
```

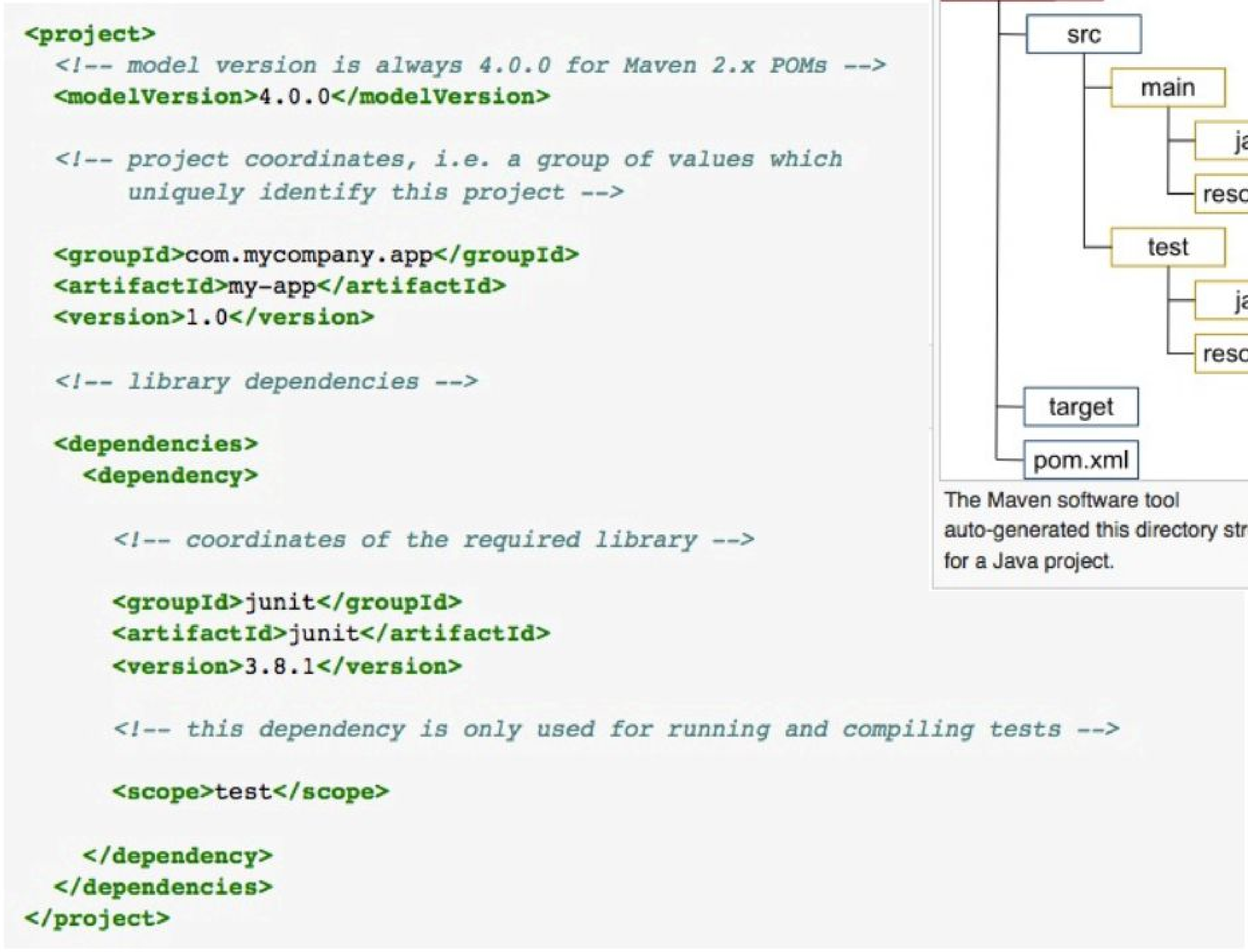
- Project structure
- pom.xml with some dependencies/settings

Maven project structure

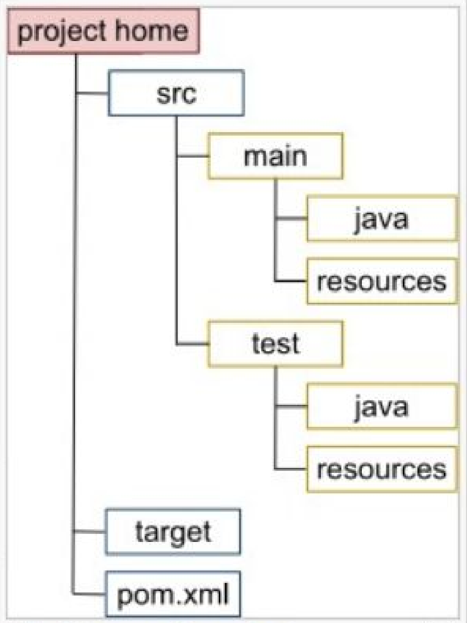

project home
src
main
java
resources
test
java
resources
target
pom.xml
Java source code
Resources (images, config files, ..)
Tests source code (same packages)
Default output folder (.class, ..)
Project configuration

The Maven software tool auto-generated this directory structure for a Java project.

pom.xml

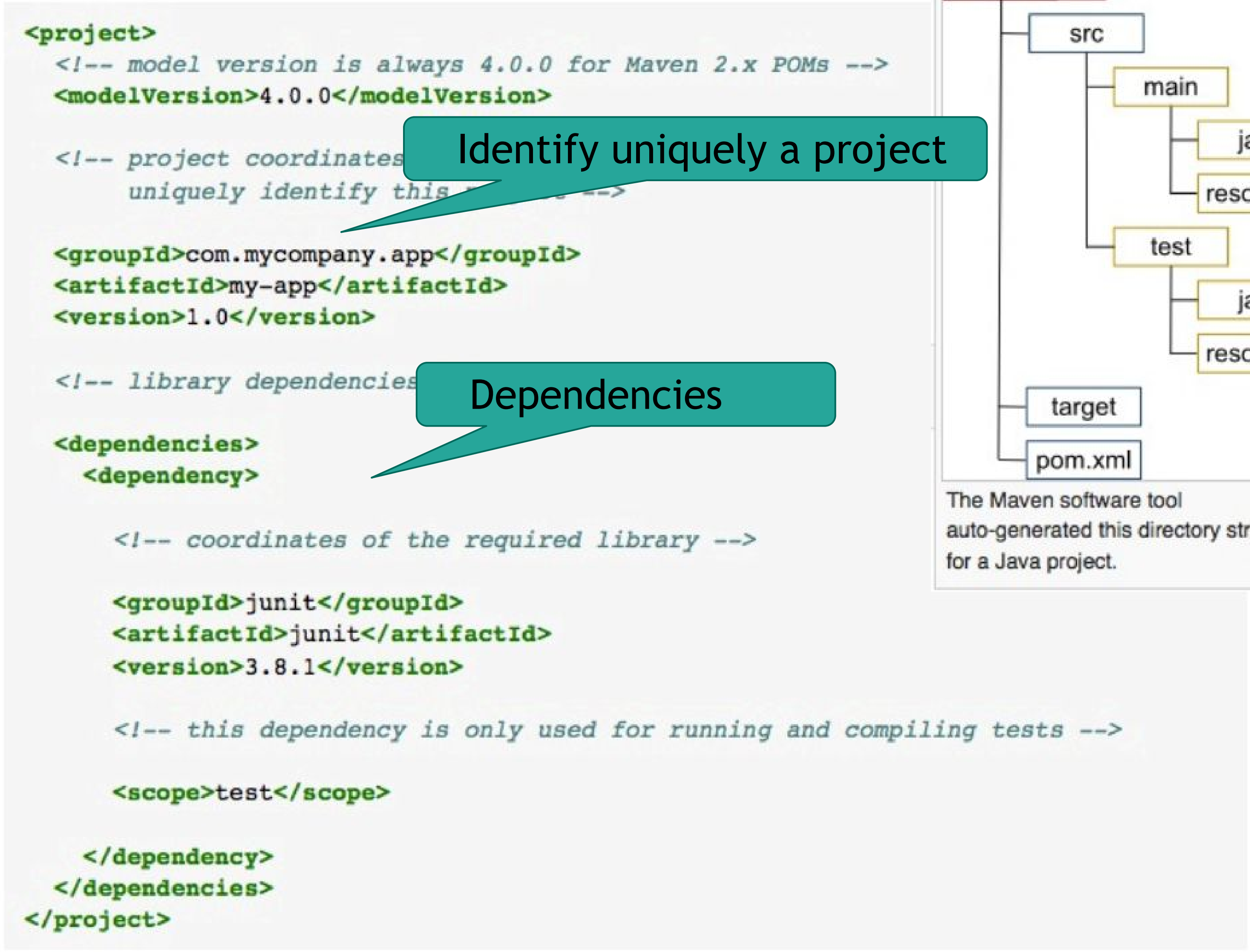


```xml
<project>
  <!-- model version is always 4.0.0 for Maven 2.x POMs -->
  <modelVersion>4.0.0</modelVersion>

  <!-- project coordinates, i.e. a group of values which
       uniquely identify this project -->

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0</version>

  <!-- library dependencies -->

  <dependencies>
    <dependency>

      <!-- coordinates of the required library -->

      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>

      <!-- this dependency is only used for running and compiling tests -->

      <scope>test</scope>

    </dependency>
  </dependencies>
</project>
```



The Maven software tool auto-generated this directory structure for a Java project.

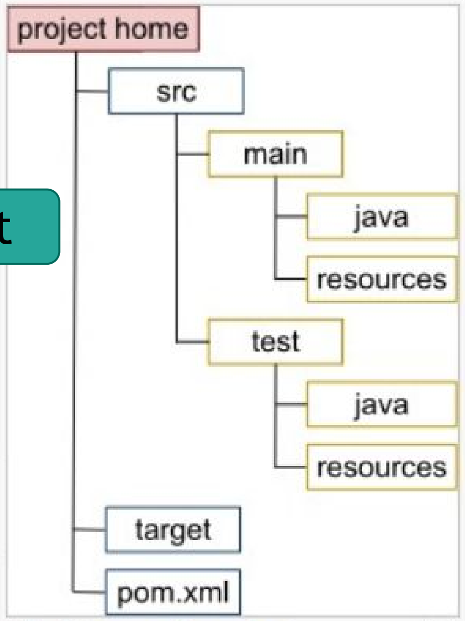
pom.xml



```
<project>
  <!-- model version is always 4.0.0 for Maven 2.x POMs -->
  <modelVersion>4.0.0</modelVersion>

  <!-- project coordinates
       uniquely identify this         -->

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0</version>

  <!-- library dependencies

  <dependencies>
    <dependency>

      <!-- coordinates of the required library -->

      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>

      <!-- this dependency is only used for running and compiling tests -->

      <scope>test</scope>

    </dependency>
  </dependencies>
</project>
```

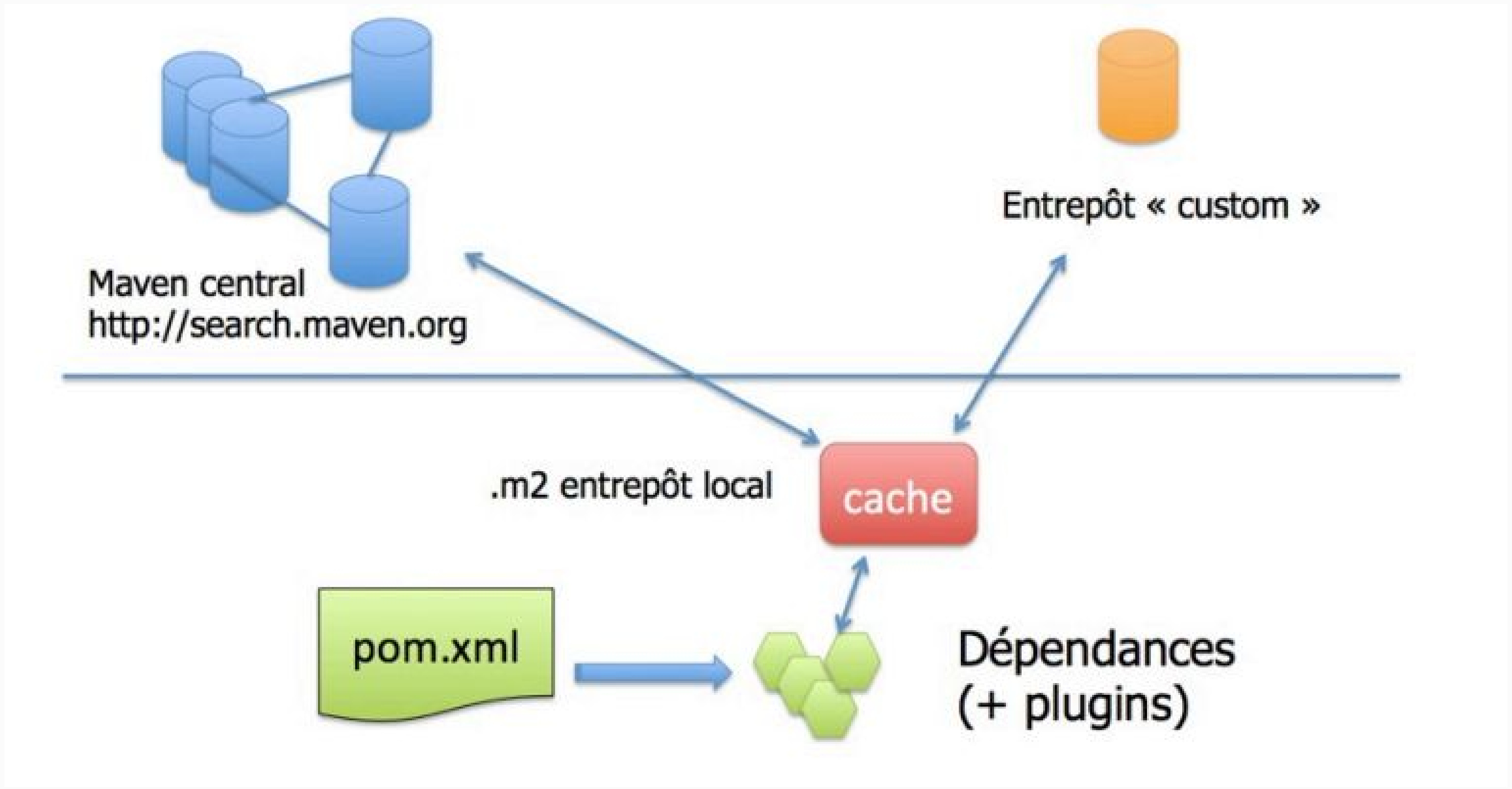
Identify uniquely a project

Dependencies

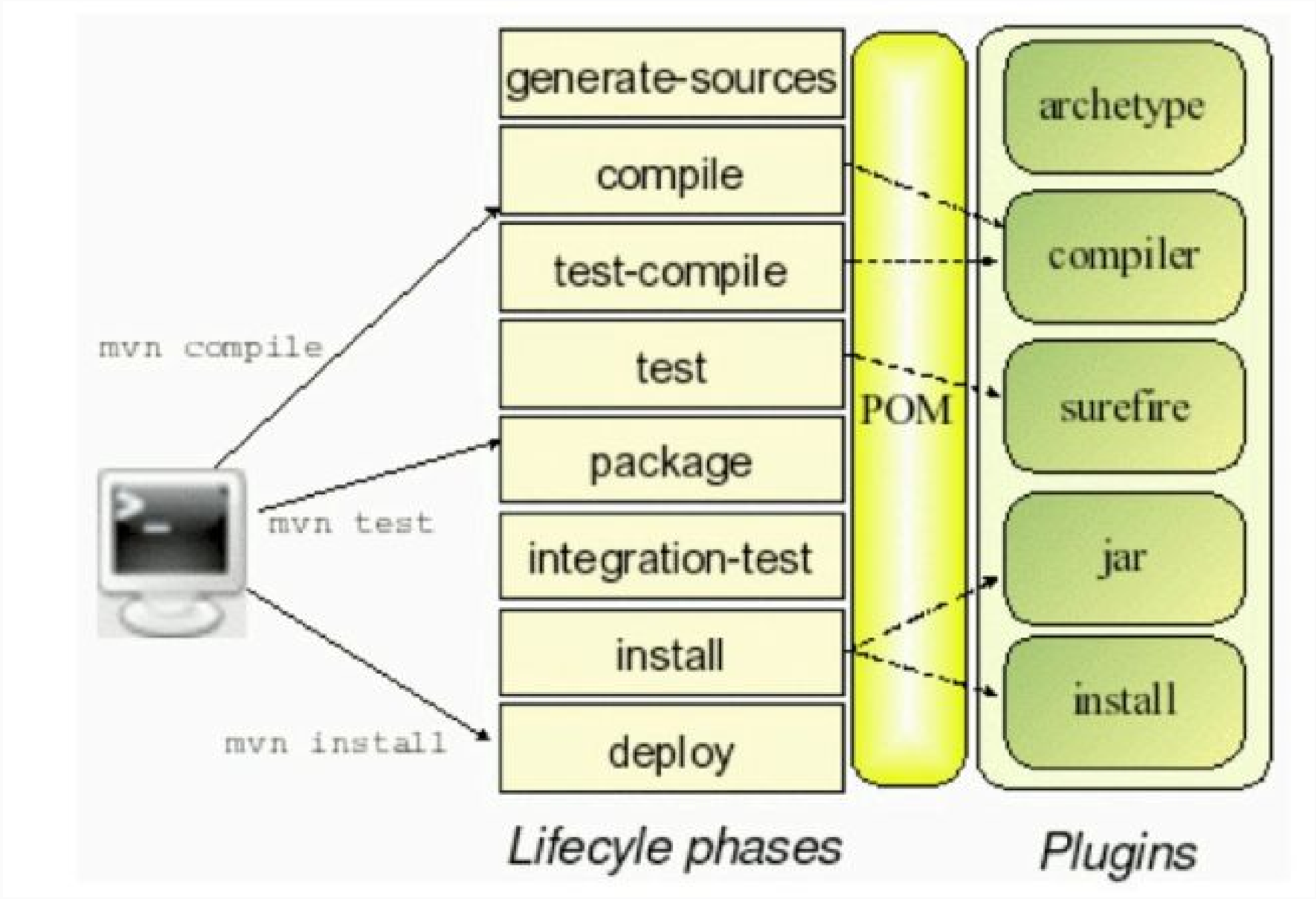


The Maven software tool auto-generated this directory structure for a Java project.

Maven repositories



Default maven lifecycle



Maven goals

- **mvn clean**: clean the project (by default, delete the target folder)
- **mvn compile**: compile the project
- **mvn test**: compile the project and tests and run the tests
- **mvn package**: compile + test then creates a package containing binaries (.jar, .war)
- **mvn install:** package + install the package locally
- **mvn deploy**: install + deploy the package on a remote maven repository

Simple pom example

```
<project>

    <groupId>fr.unice.iut</groupId>
    <artifactId>simple</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <packaging>jar</packaging>

    <dependencies> … </dependencies>

    <build>
        <plugins> … </plugins>
    </build>
</project>
```

Identify uniquely the project

Package into a jar

Required dependencies

External plugins

Simple pom example

```
<dependencies>
     <dependency>
          <groupId>org.json</groupId>
          <artifactId>json</artifactId>
          <version>20151123</version>
     </dependency>

     <dependency>
          <groupId>junit</groupId>
          <artifactId>junit</artifactId>
          <version>4.12</version>
          <scope>test</scope>
     </dependency>
</dependencies>
```

Serialization to/from Json

Is only resolved during the test phase

Simple pom example

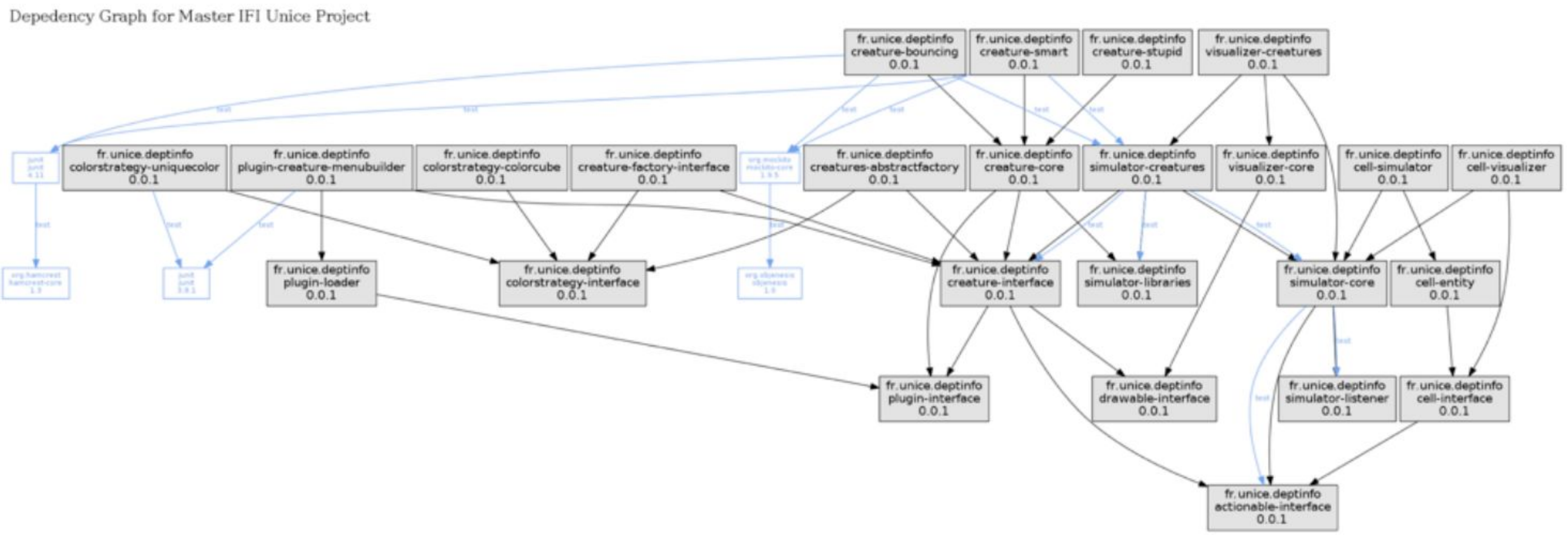
```
<build><plugins>
        <plugin>
           <artifactId>maven-assembly-plugin</artifactId>

           <configuration>
               <archive> <manifest>
                      <mainClass>fr.unice.iut.simple.Main</mainClass>
               </manifest> </archive>
               <descriptorRefs>
                   <descriptorRef>jar-with-dependencies</descriptorRef>
               </descriptorRefs>
           </configuration>
        </plugin>
</plugins></build>
```

Unique identifier of the plugin

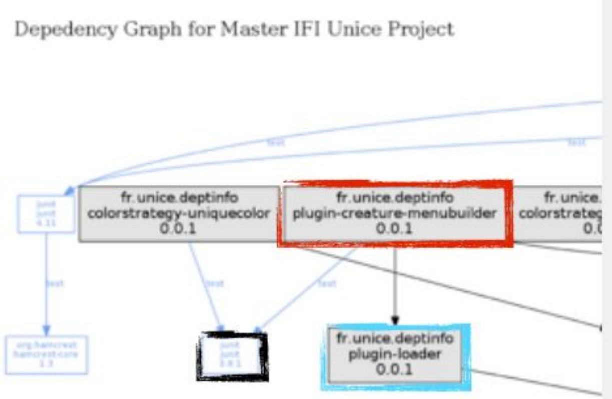
Main class for executable jar

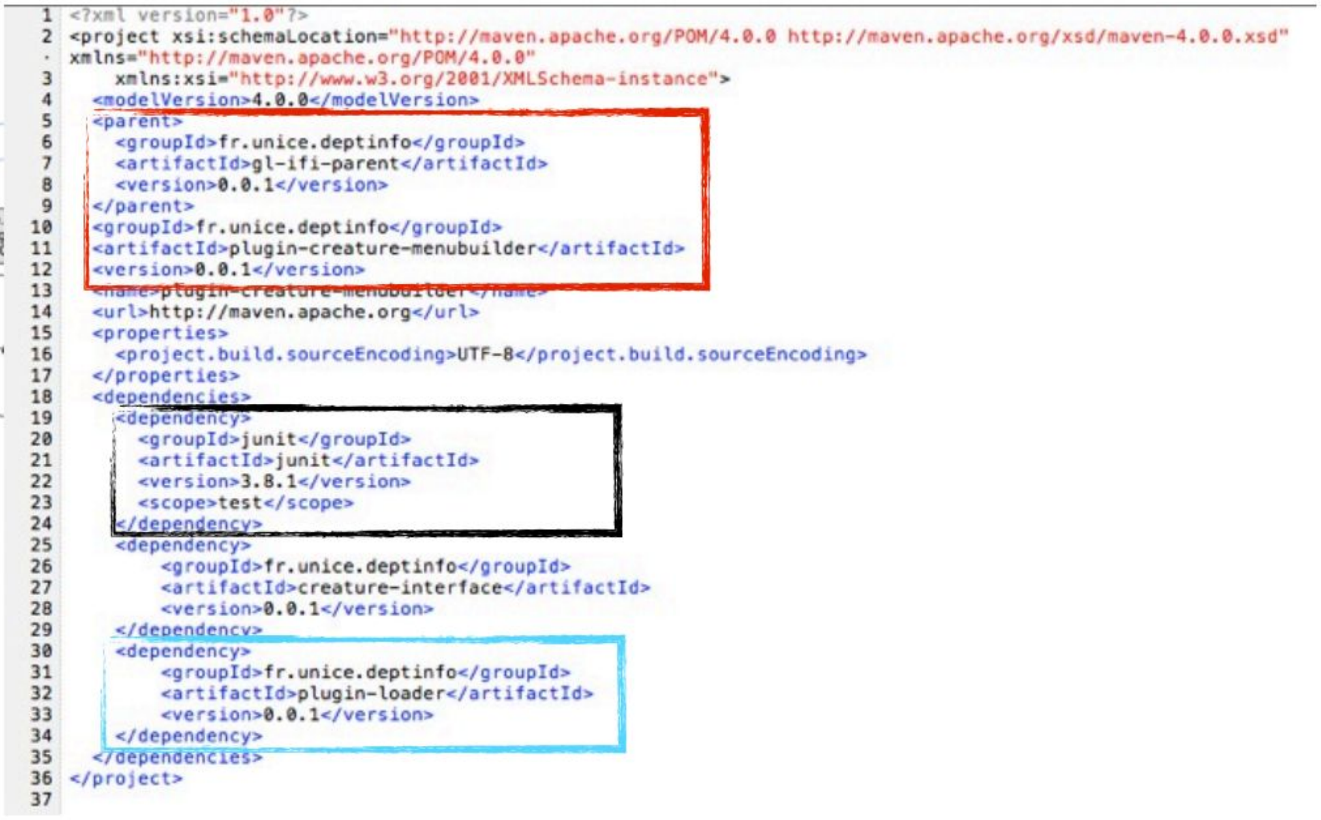
Configuration of the plugin

Going further: dependencies


Depedency Graph for Master IFI Unice Project
fr.unice.deptinfo creature-bouncing 0.0.1
fr.unice.deptinfo creature-smart 0.0.1
fr.unice.deptinfo creature-stupid 0.0.1
fr.unice.deptinfo visualizer-creatures 0.0.1
junit junit 4.11
fr.unice.deptinfo colorstrategy-uniquecolor 0.0.1
fr.unice.deptinfo plugin-creature-menubuilder 0.0.1
fr.unice.deptinfo colorstrategy-colorcube 0.0.1
fr.unice.deptinfo creature-factory-interface 0.0.1
org.mockito mockito-core 1.9.5
fr.unice.deptinfo creatures-abstractfactory 0.0.1
fr.unice.deptinfo creature-core 0.0.1
fr.unice.deptinfo simulator-creatures 0.0.1
fr.unice.deptinfo visualizer-core 0.0.1
fr.unice.deptinfo cell-simulator 0.0.1
fr.unice.deptinfo cell-visualizer 0.0.1
org.hamcrest hamcrest-core 1.3
junit junit 3.8.1
fr.unice.deptinfo plugin-loader 0.0.1
fr.unice.deptinfo colorstrategy-interface 0.0.1
org.objenesis objenesis 1.0
fr.unice.deptinfo creature-interface 0.0.1
fr.unice.deptinfo simulator-libraries 0.0.1
fr.unice.deptinfo simulator-core 0.0.1
fr.unice.deptinfo cell-entity 0.0.1
fr.unice.deptinfo plugin-interface 0.0.1
fr.unice.deptinfo drawable-interface 0.0.1
fr.unice.deptinfo simulator-listener 0.0.1
fr.unice.deptinfo cell-interface 0.0.1
fr.unice.deptinfo actionable-interface 0.0.1
test

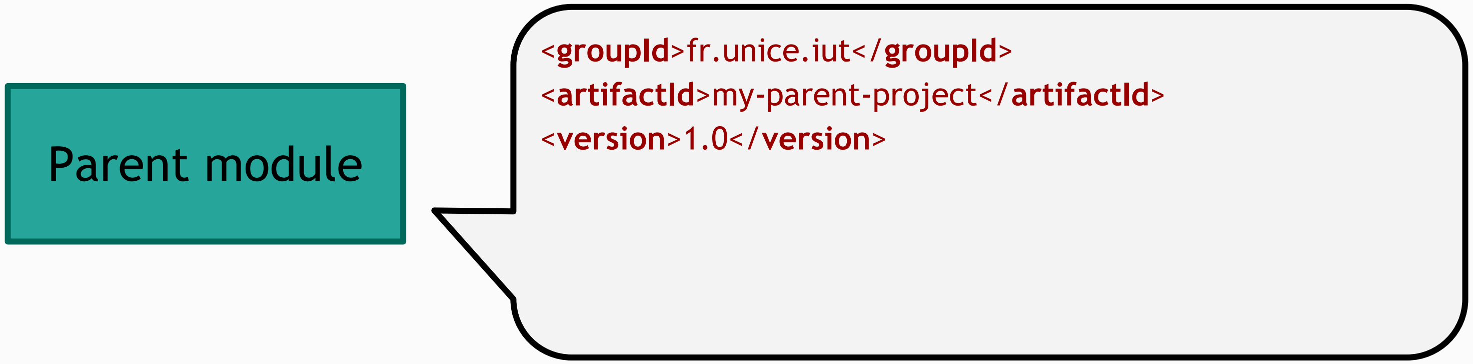
Going further: dependencies


Depedency Graph for Master IFI Unice Project
fr.unice.deptinfo colorstrategy-uniquecolor 0.0.1
fr.unice.deptinfo plugin-creature-menubuilder 0.0.1
fr.unice. colorstrateg 0.0
junit junit 4.11
org.hamcrest hamcrest-core 1.3
junit junit 3.8.1
fr.unice.deptinfo plugin-loader 0.0.1
test



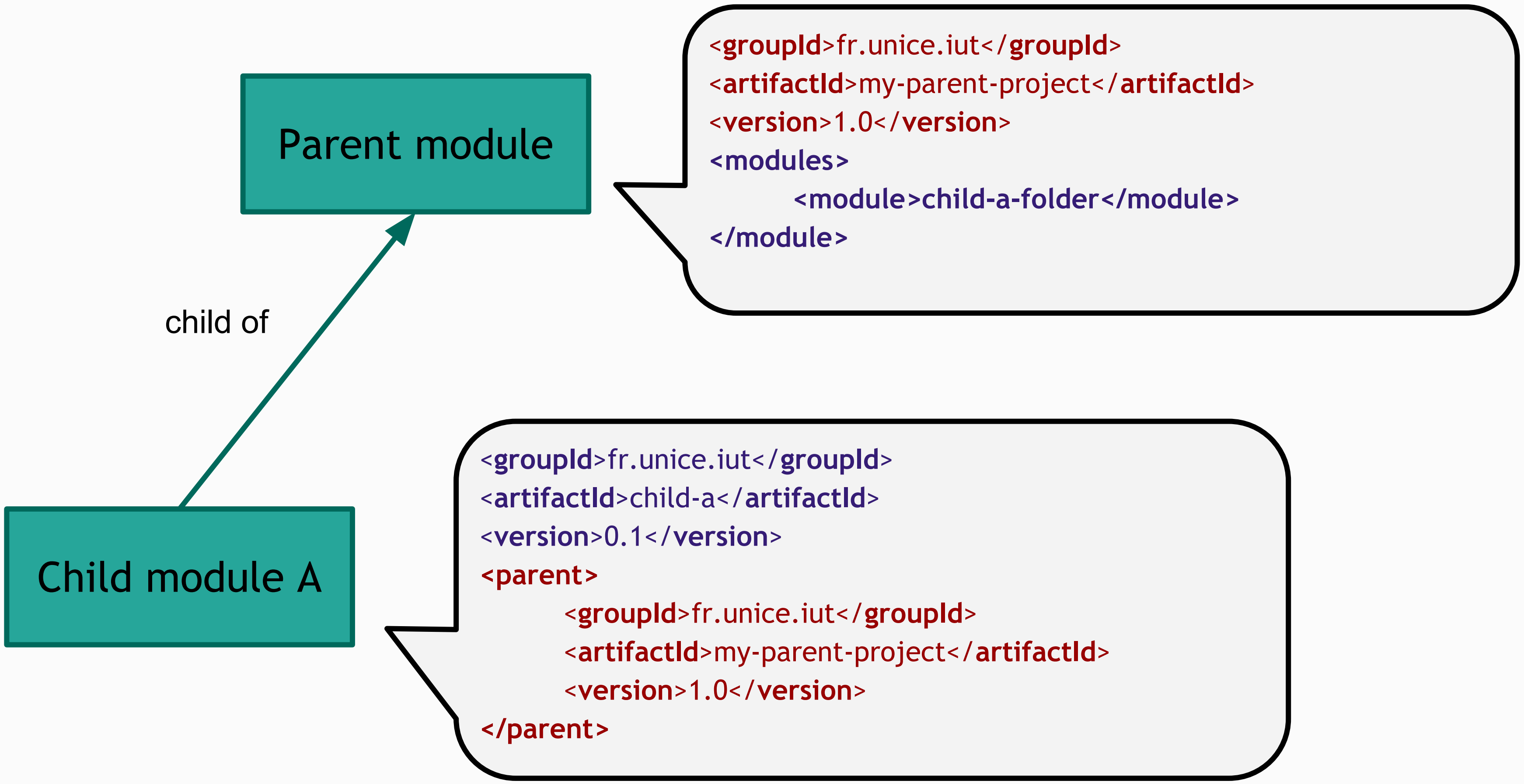
```xml
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>fr.unice.deptinfo</groupId>
    <artifactId>gl-ifi-parent</artifactId>
    <version>0.0.1</version>
  </parent>
  <groupId>fr.unice.deptinfo</groupId>
  <artifactId>plugin-creature-menubuilder</artifactId>
  <version>0.0.1</version>
  <name>plugin-creature-menubuilder</name>
  <url>http://maven.apache.org</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>fr.unice.deptinfo</groupId>
        <artifactId>creature-interface</artifactId>
        <version>0.0.1</version>
    </dependency>
    <dependency>
        <groupId>fr.unice.deptinfo</groupId>
        <artifactId>plugin-loader</artifactId>
        <version>0.0.1</version>
    </dependency>
  </dependencies>
</project>
```

Going further: modules hierarchy



```
<groupId>fr.unice.iut</groupId>
<artifactId>my-parent-project</artifactId>
<version>1.0</version>
```

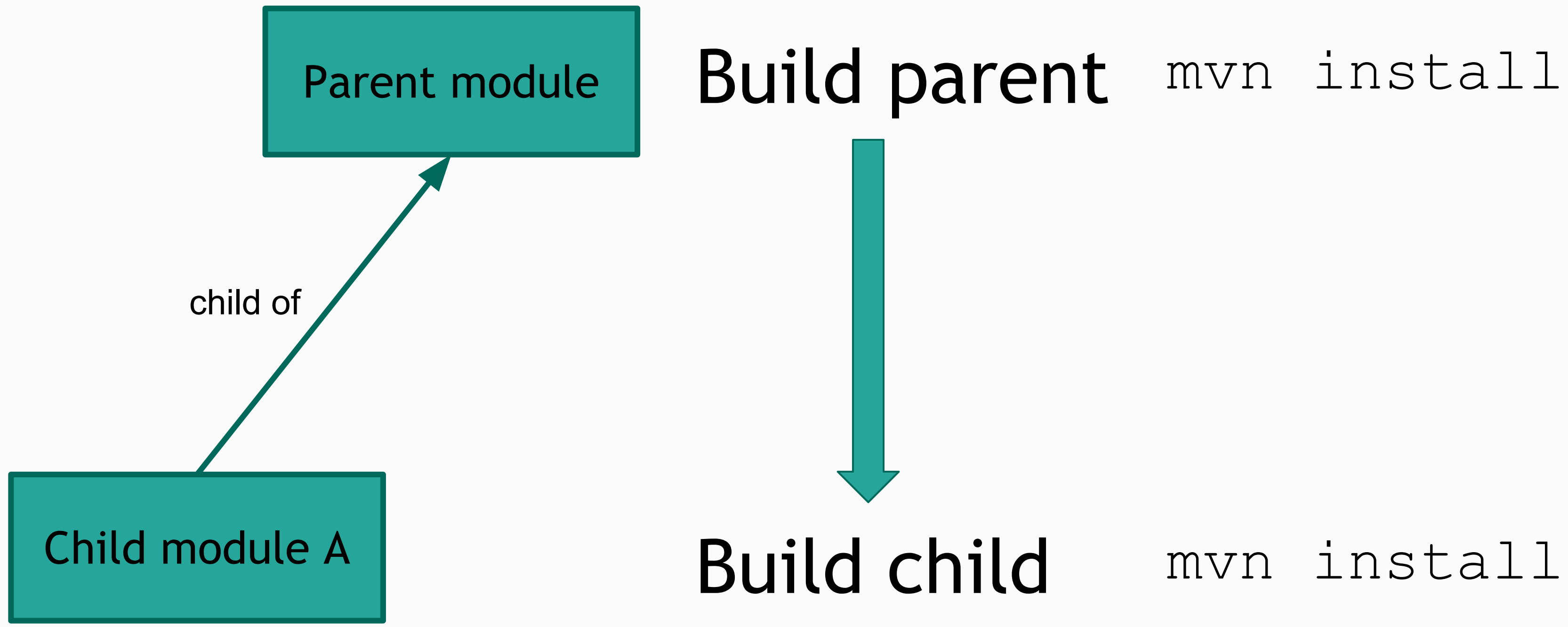
Going further: modules hierarchy



```xml
<groupId>fr.unice.iut</groupId>
<artifactId>my-parent-project</artifactId>
<version>1.0</version>
<modules>
        <module>child-a-folder</module>
</module>
```

```xml
<groupId>fr.unice.iut</groupId>
<artifactId>child-a</artifactId>
<version>0.1</version>
<parent>
        <groupId>fr.unice.iut</groupId>
        <artifactId>my-parent-project</artifactId>
        <version>1.0</version>
</parent>
```

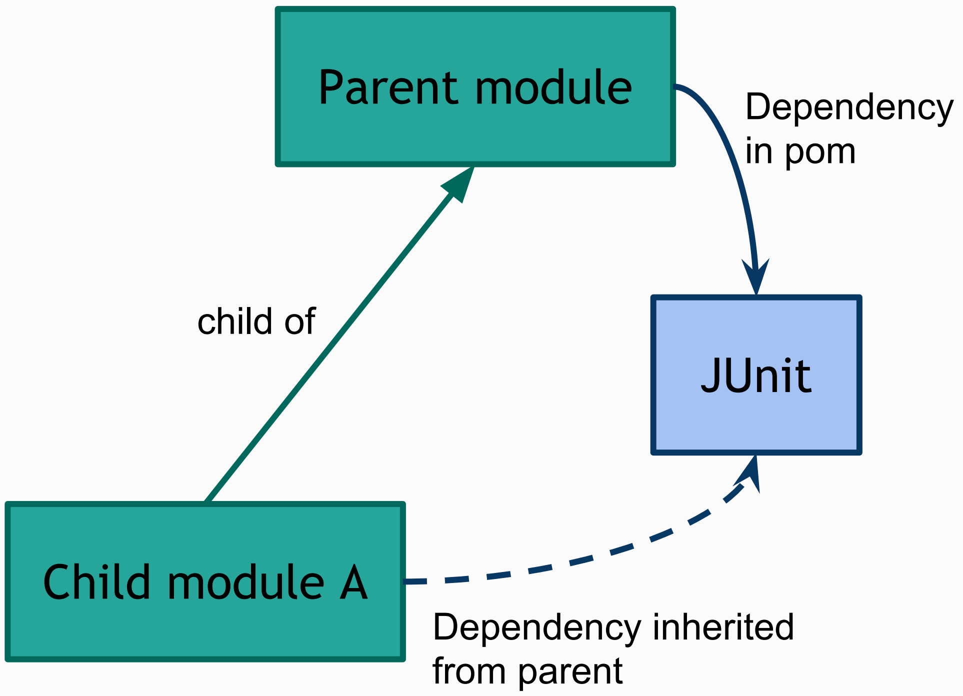
Going further: modules hierarchy



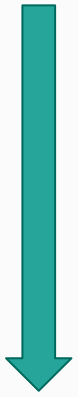
```
mvn install
```

```
mvn install
```

Going further: modules hierarchy

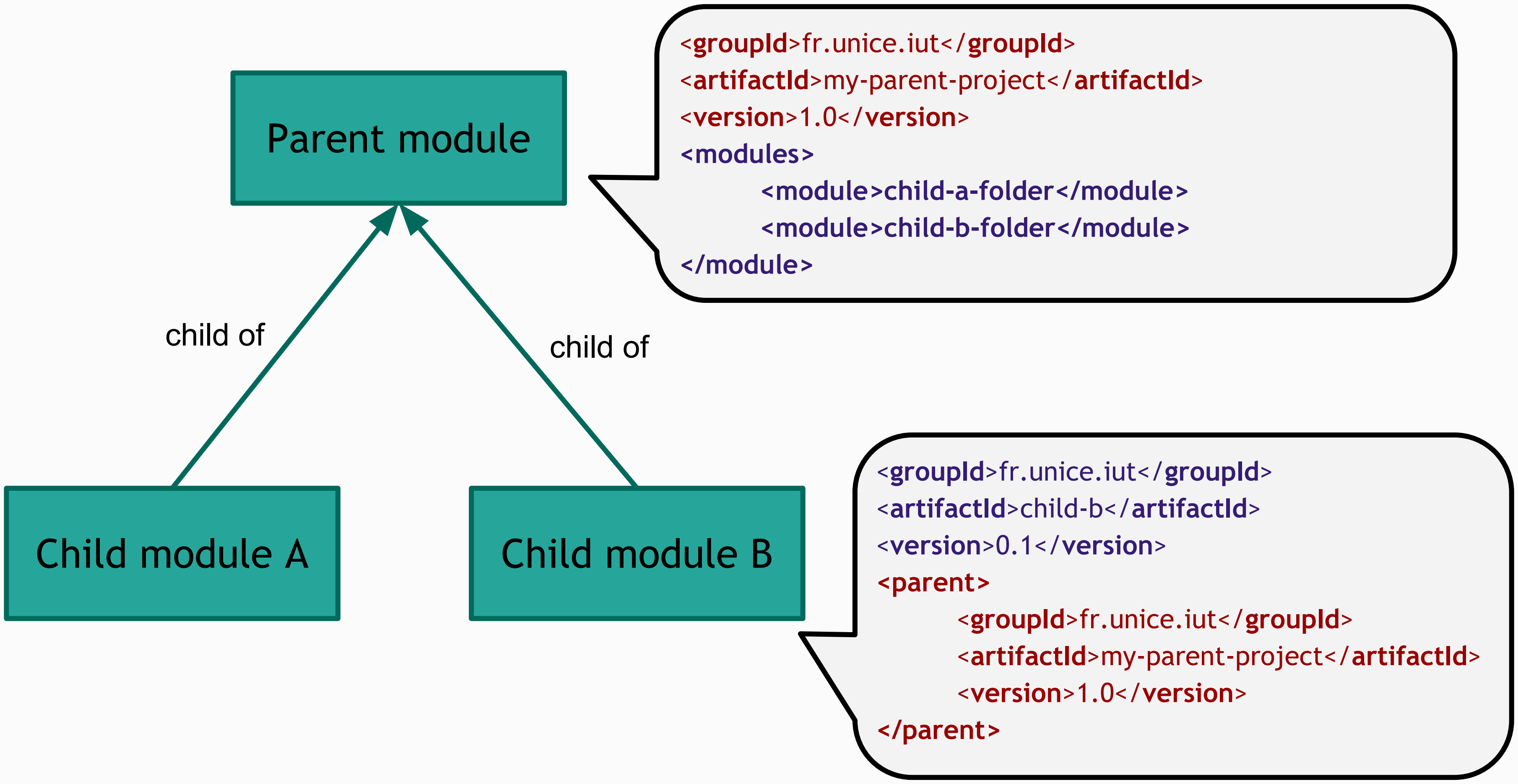


Parent’s dependencies



Child’s dependencies

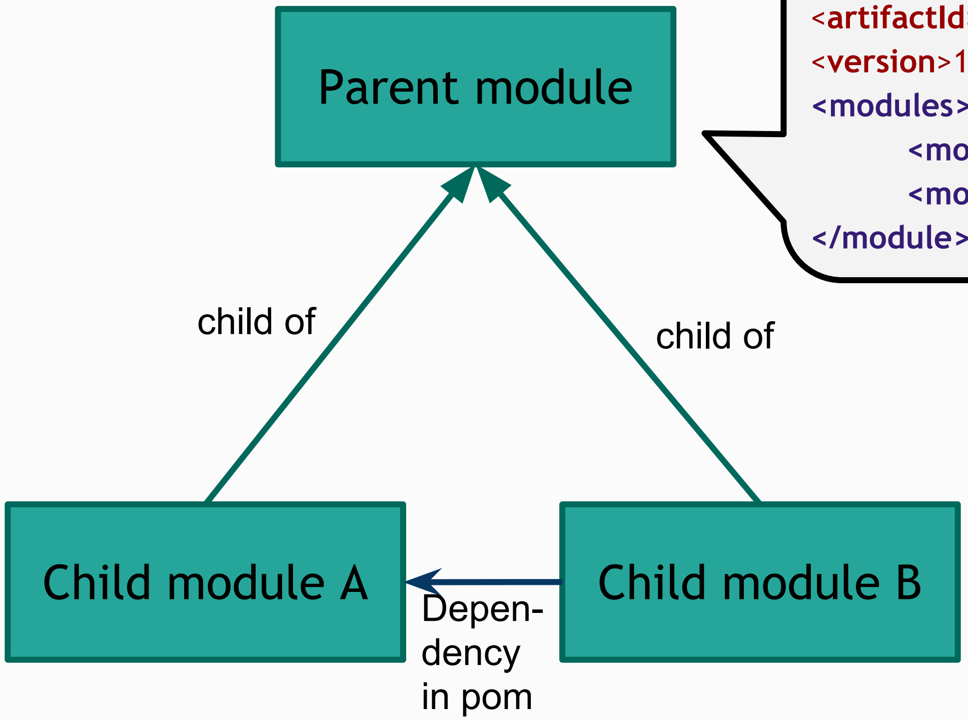
Going further: modules hierarchy



```
<groupId>fr.unice.iut</groupId>
<artifactId>my-parent-project</artifactId>
<version>1.0</version>
<modules>
        <module>child-a-folder</module>
        <module>child-b-folder</module>
</module>
```

```
<groupId>fr.unice.iut</groupId>
<artifactId>child-b</artifactId>
<version>0.1</version>
<parent>
        <groupId>fr.unice.iut</groupId>
        <artifactId>my-parent-project</artifactId>
        <version>1.0</version>
</parent>
```

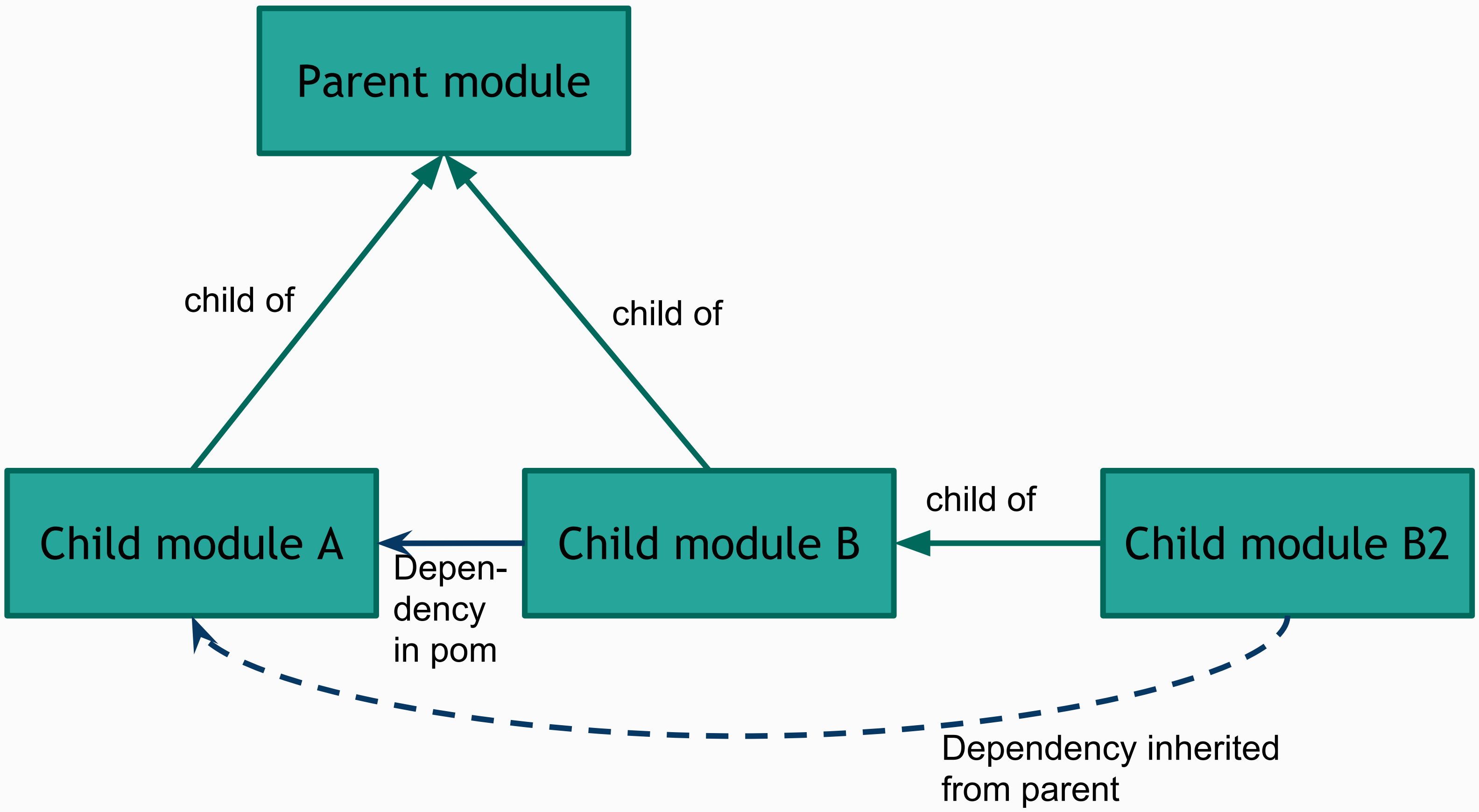
Going further: modules hierarchy



```
<groupId>fr.unice.iut</groupId>
<artifactId>my-parent-project</artifactId>
<version>1.0</version>
<modules>
        <module>child-a-folder</module>
        <module>child-b-folder</module>
</module>
```

```
<groupId>fr.unice.iut</groupId>
<artifactId>child-b</artifactId>
<version>0.1</version>
<parent>
        <groupId>fr.unice.iut</groupId>
        <artifactId>my-parent-project</artifactId>
        <version>1.0</version>
</parent>
<dependencies><dependency>
        <groupId>fr.unice.iut</groupId>
        <artifactId>child-a</artifactId>
        <version>0.1</version>
</dependency></dependencies>
```

Going further: modules hierarchy



Maven to...

Build...

Test...

Deploy...

Sidenote

Gradle

Simple gradle project

```
apply plugin: 'java'
repositories {
    mavenCentral()
}
dependencies {
    compile group: 'cc', name: 'cc', version: '3.2.2'
    testCompile group: 'junit', name: 'junit', version: '4.+'
}
jar {
    manifest {
        attributes 'Main-Class': 'fr.unice.iut.simple.Main'
    }
}
```

Look for dependencies in maven central

Some dependency

JUnit only for tests

Jar configuration

?