Agile Software Development

Produced by

Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

http://www.wit.ie

http://elearning.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit

Maven

- Objectives:
  - Provide a standard development infrastructure across projects
  - Make the development process transparent
  - Decrease training for new developers.
  - Bring together tools in a uniform way
  - Prevent inconsistent setups
  - Divert energy to application development activities
- Maven is a process of applying patterns to a build infrastructure in order to provide a coherent view of software projects

Maven

- Principles:
    - Convention over configuration
    - Declarative execution
    - Reuse of build logic
    - Coherent organization of dependencies

Convention over configuration

- Provide sensible default strategies for the most common build tasks
- The primary conventions to promote a familiar development environment are:
    1. Standard directory layout for projects
    2. Standardised set of build phases – lifecycle phases
    3. A single Maven project produces a single output/artifact
    4. Standard naming conventions
- Conventions are Maven’s understanding of how a project is typically built. This built-in project knowledge simplifies and facilitates project builds.
- Leverages its built-in project knowledge to help users understand a complex project's structure and potential variations in the build process.

Reuse of build logic

- Reuse of build logic principle:
    - Maven encapsulating build logic into modules called plugins.
    - A plugin's components, called mojos, perform build tasks.
        - MOJO - **M**aven plain **O**ld **J**ava **O**bjects
- Maven acts as a framework which coordinates the execution of plugins in a well defined way.
- Some plugins are standard, others are downloaded on demand.

Declarative Execution

- Ant's typical target names are standardised into a set of well-defined and well-known build lifecycle phases.

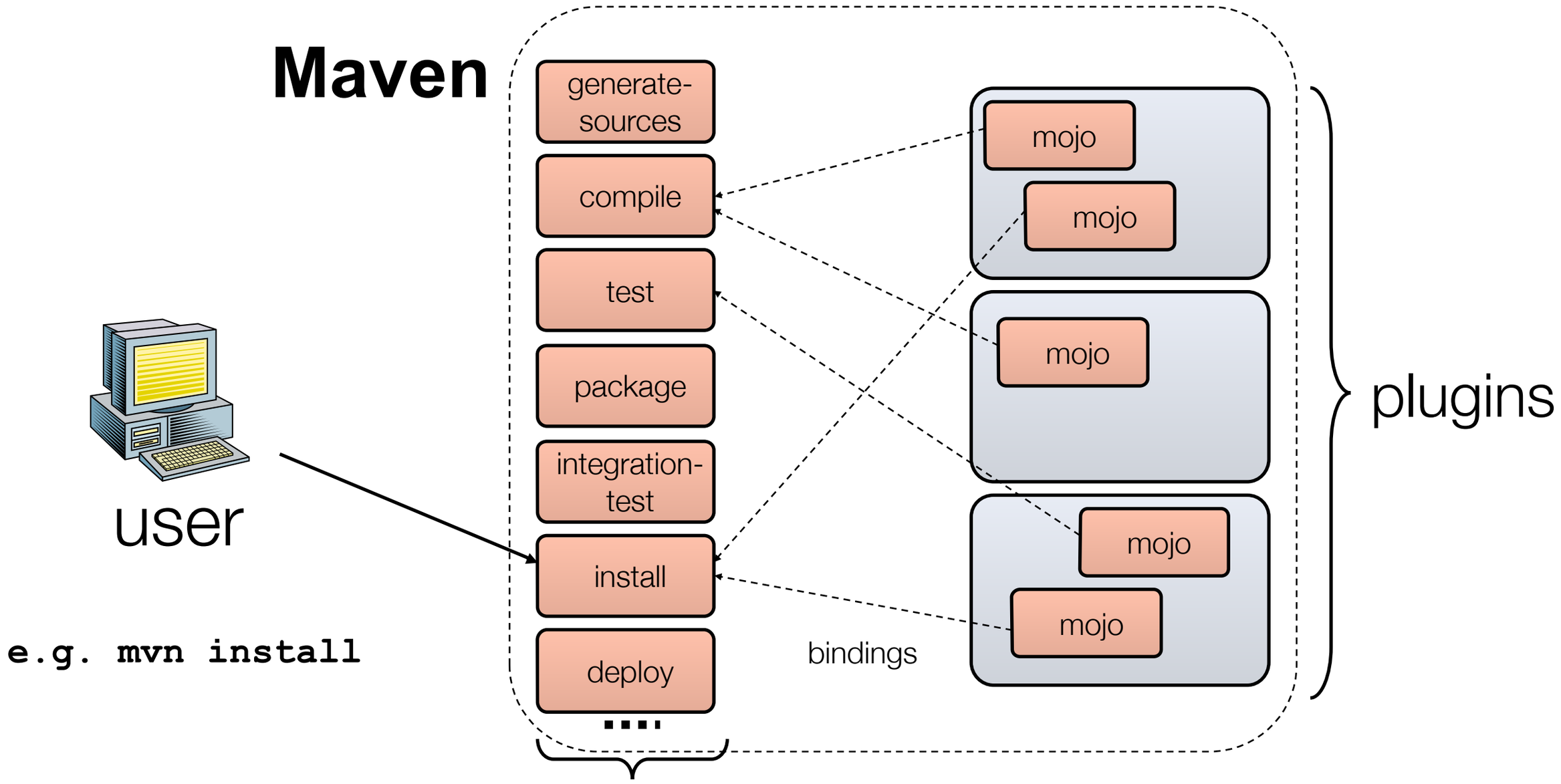
- A lifecycle phase invokes the relevant plugins (the mojos) to do the work.

- The phase to plugin bindings are hardwired (for standard plugins).

- User configures a plugin declaratively in the POM (Project Object Model) file.

- Configuration only necessary for non-standard cases

Declarative Execution



Well-known phases

Declarative Execution

- When user invokes a lifecycle phase, all its predecessors are also executed, if necessary, e.g. *mvn package*.

- You can also invoke plugins directly.

  Format: *mvn plugin-name:goal*

  e.g. mvn jetty:run

Declarative Execution

- The POM file (pom.xml) is Maven's description of a single project
- It drives Maven’s execution for a project
    - e.g configuring a plugin for a particular phase.
- Contains metadata about the project
    - Location of directories, Developers/Contributors, Extra plugins required, Special plugin configuration, Jars required (3rd party and in-house), Repositories to search for plugins/jars, etc.
- A project’s POM inherits from the Super POM.
    - All standard project information (e.g. directory structure) is held in the Super POM (principle).

Minimalist POM

```
<project>

  <modelVersion>4.0.0</modelVersion>

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

  <dependencies>

   <dependency>
     <groupId>com.thoughtworks.xstream</groupId>
     <artifactId>xstream</artifactId>
     <version>1.3.1</version>
   </dependency>

  </dependencies>

</project>
```

Minimalist POM elements

- groupId - indicates the unique identifier of the organization or group that created the project. Typically based on the fully qualified domain name of the organization

- artifactId - indicates the unique base name of the primary artifact being generated by this project. A typical artifact produced by Maven would have the form <artifactId>-<version>.<extension> (for example, myapp-1.0.jar)

- packaging - indicates the package format to be used for this artifact (JAR, WAR, EAR, etc.). It also indicates a specific life cycle to use as part of the build process.

Coherent organization of dependencies

- Three related concepts: Artifact; Dependencies; Repositories

```
<project>
      ……
  <dependencies>

    <dependency>
      <groupId>com.thoughtworks.xstream</groupId>
      <artifactId>xstream</artifactId>
      <version>1.3.1</version>
    </dependency>

   </dependencies>

</project>
```

This project has a dependency on version 1.3.1 of the artifact with id xstream, produced by the com.thoughtworks.xstream group.

Coherent organization of dependencies

- All artifacts/dependencies are stored in repositories
  - Local and remote repositories
- The local repository is searched first, then remote ones
- Dependencies are automatically downloaded (from remote repositories) and installed (in local repository) for future use
- Maven knows about some remote repositories, e.g.

  http://ibiblio.org/maven2

- Other remote repositories can be listed in the project POM or in Maven's configuration file (setting.xml)

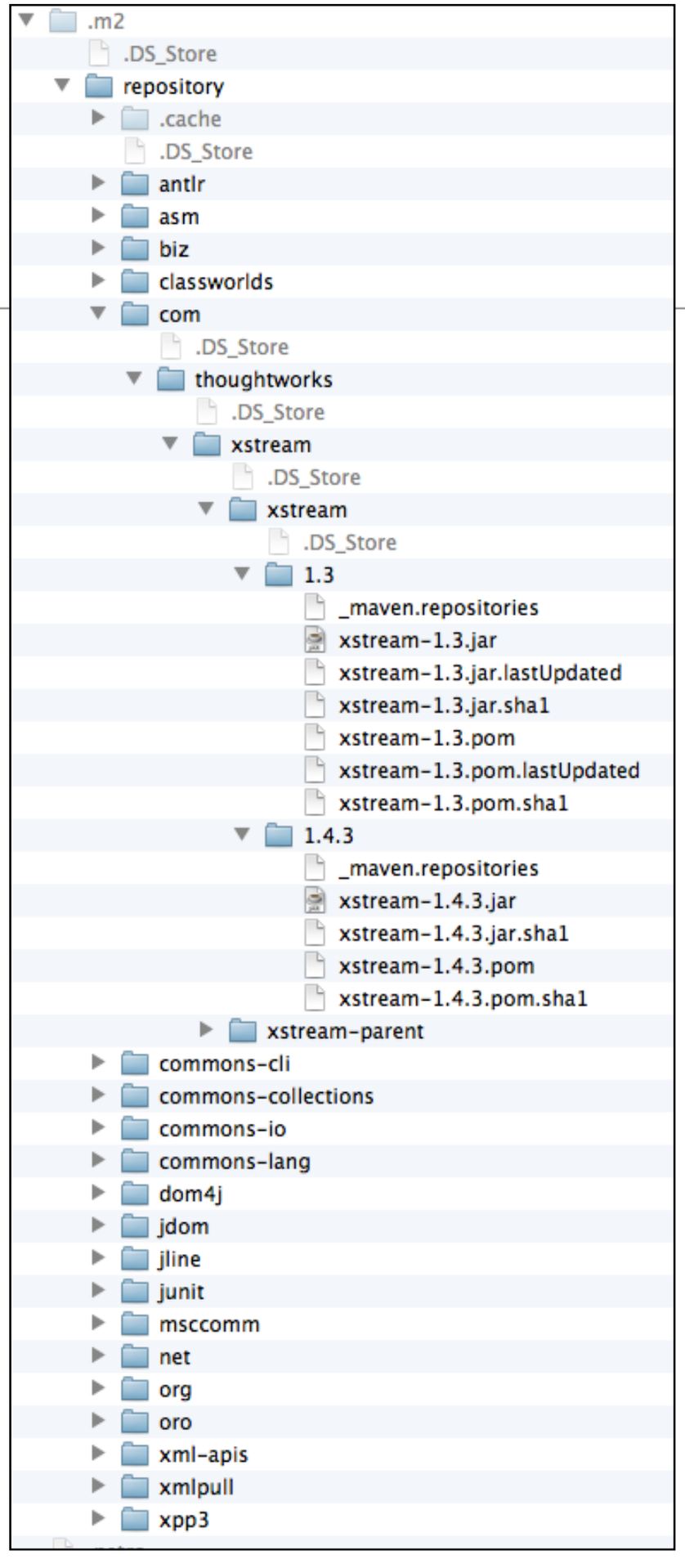
Local repositories.

- After installing and running Maven for the first time a local repository is automatically created and populated with some standard artifacts

    - Default Local repository location: *Home/.m2/repository*

- Plugins are also stored in repositories.

- In theory a repository is an abstract storage mechanism, but in practice it is a directory structure in your file system

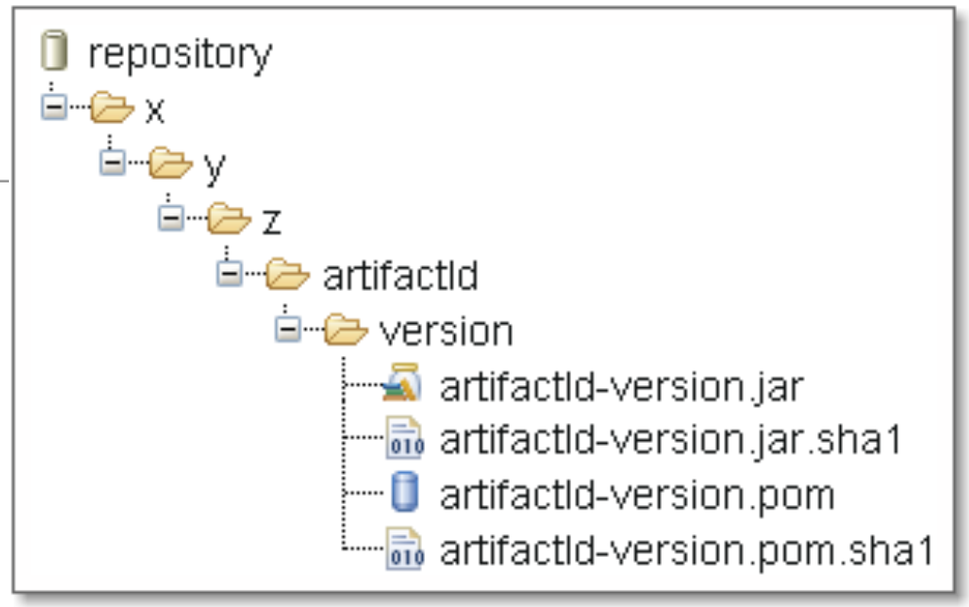


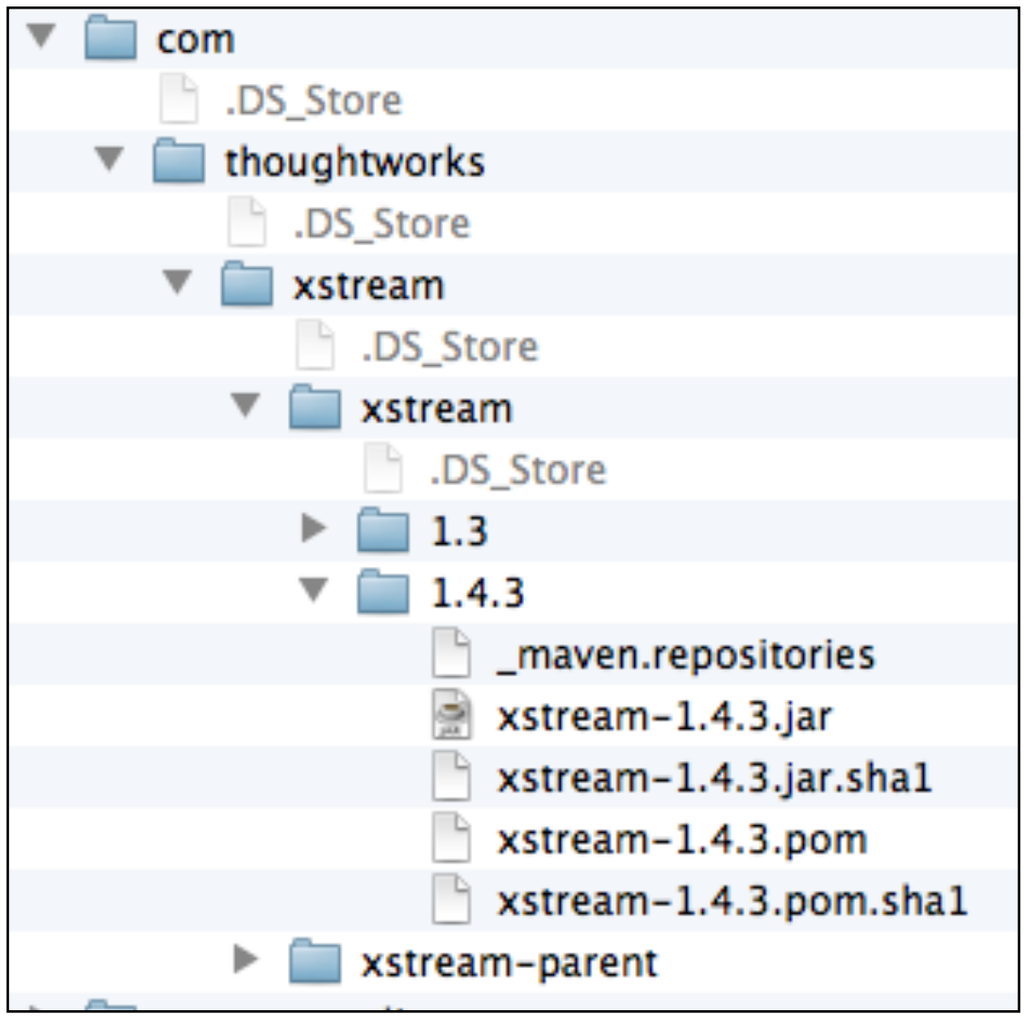
Repository structure.

- Repository structure centered around dependency coordinates schema.

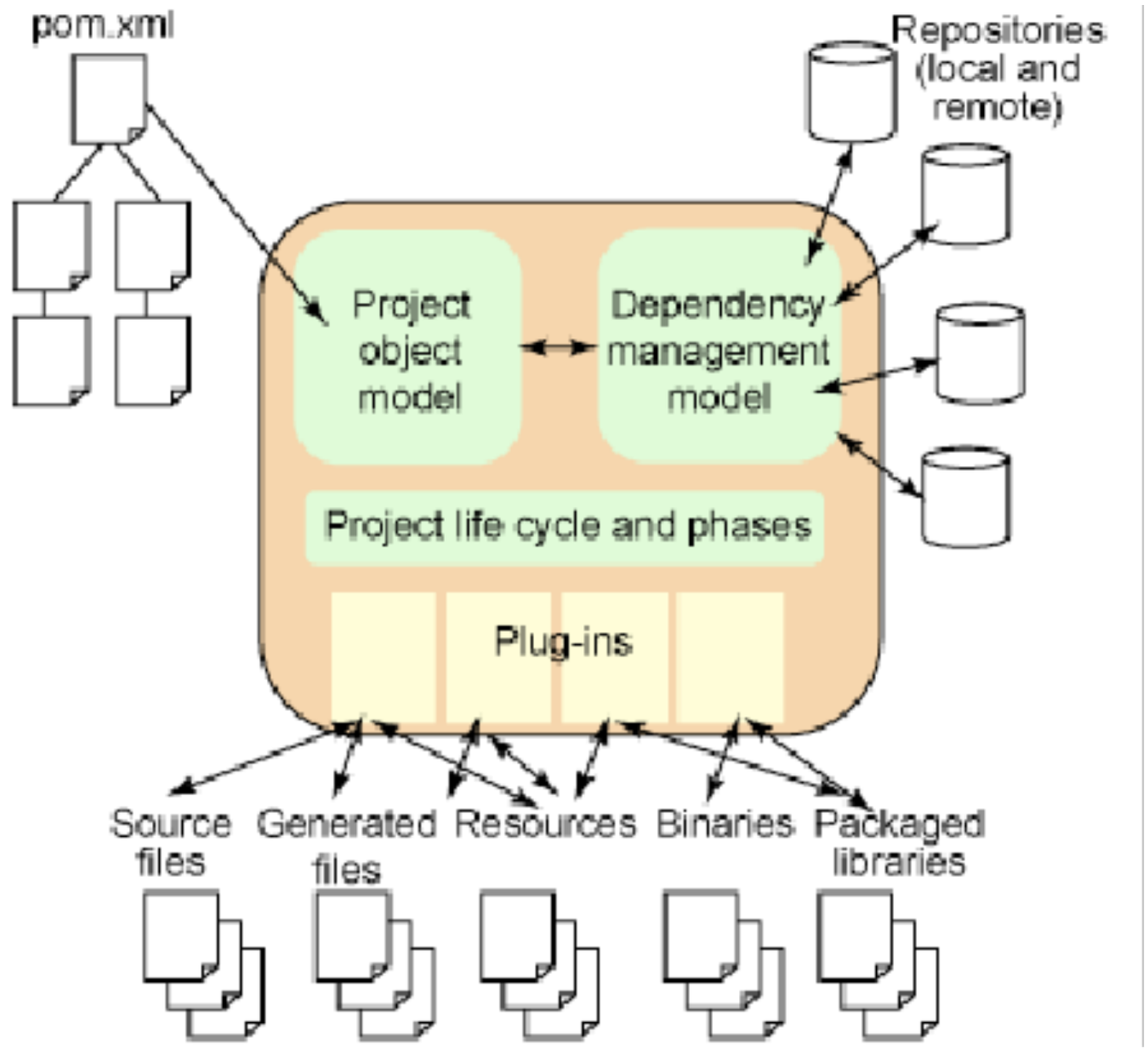
- Maven uses artifact's id, group id. and version to navigate to the correct folder.

- If the groupId is a fully qualified domain name such as x.y.z then it is fully expanded.





The full picture.



Archetypes.

- An archetype is a template project structure
- Many archetype options:
  - maven-archetype-webapp – Web application (WAR) project template
  - maven-archetype-j2ee-simple – J2EE project (EAR) with directories and subprojects for the EJBs, servlets, etc.
  - maven-archetype-quickstart (default) – simple Java project (JAR)
- Create a new project folder structure with the archetype plugin, invoking the create goal

```
mvn archetype:create
    -DgroupId=[your project's group id]
    -DartifactId=[your project's artifact id]
    -DarchetypeArtifactId=[artifact type]
```

Quickstart archetype.

- Folder structure for ‘quickstart’ archetype

- The base directory name is taken from artifactid.
- A minimal POM is included in base directory.