About Tycho, Maven, p2 and Target-Platforms: From Pain to Best Practice

Sascha Müller (sa.mueller@dlr.de)
Philipp M. Fischer (philipp.fischer@dlr.de)
Tobias Schlauch (tobias.schlauch@dlr.de)
Agenda

1. Projects layouts
2. Integrating Tycho / Maven
3. Our Jenkins setup
4. How we setup new projects
What we hope to deliver to you

Tycho and Maven based build infrastructure
- Automated builds and testing
- Good integration with Eclipse technologies
- But how to set it up?

The Pain
- Only little documentation
- Answers scattered through various forum posts, books and tutorials
- Difficult to answer all projects needs with “one ultimate answer”

We hope to deliver a compilation of answers to make the start easy for you!
Our Main Project

Virtual Satellite (VirSat)

- Eclipse based framework for spacecraft related applications
- For example modelling tools
- Used for research and productive projects by engineers

Need for a building and testing infrastructure that can handle all cases!
Our simple Eclipse project structure

**Default project setup**
- All plugins on the same level of the folder hierarchy
- Master pom referenced by ..\myProject\pom.xml
- Use working sets to structure plugins in the IDE

```xml
<parent>
  <artifactId>de.dlr.sc.virsat</artifactId>
  <groupId>de.dlr.sc.virsat</groupId>
  <version>4.5.0-SNAPSHOT</version>
  <relativePath>../de.dlr.sc.virsat/pom.xml</relativePath>
</parent>
```
Getting Maven to use Tycho

Setting up the tycho plugins in the master pom

- Requires adding several plugins
- Some very straightforward to configure
- But some need specific settings to work well

Let’s discuss them in the following!
Getting your warnings evaluated on Jenkins

Setting up the compiler

- Add tycho-compiler-plugin to master pom
- Set showWarnings flag so Jenkins can pick up on Java warnings
Using the IDE target platform in the build

Explicitly declaring the target platform

- Maven/Tycho needs dependencies to build
- Eclipse needs dependencies to build
- Use cases: All projects with source code

Same target platform in development and build environment!
Getting dependencies directly with Maven

Declaring repositories in the master pom

- Change in product often means change in the target platform
- By declaring repositories Maven/Tycho can get all dependencies
- Use case: Projects without own source code
  - Example: VirSat IDE

(all features developed in CORE)
Setting up your testing environment

Testing in headless mode
- Use case: Regular unit tests
- Can be combined with jacoco plugin for coverage reports

Run in headless mode
Continue building even if there is a test failure.
Keeping version numbers up to date

**Using ant script to update version numbers**

- A lot of files specify the plugin version
- Need to be updated when software version is incremented

Automatically update version numbers with an ant script!

```xml
<!-- ==============================================================
Task: updateVersion - Update Build Version in POM for deployment
 ============================================================== -->
<replaceregexp byline="true">
  <regexp pattern="&lt;version&gt;${version.pattern}.SNAPSHOT&lt;/version&gt;" />
  <substitution expression="&lt;version&gt;${version.new}.SNAPSHOT&lt;/version&gt;" />
  <fileset dir="..">
    <exclude name="de.dlr.sc.**/target/"/>
    <include name="de.dlr.sc.**/pom.xml" />
  </fileset>
</replaceregexp>
```
Mirroring external dependencies

Directly referencing external repositories
- Impacts loading times of the target platform
- External site might be down
- Resolving dependencies may take a lot of time

Mirror external dependencies into a local p2 repository

Locations
The following list of locations will be used to collect plug-ins for this target definition.

- http://p2-mirror.sc.dlr.de/adapt/releases/12x/120/
- http://p2-mirror.sc.dlr.de/projects/subversive/download/eclipse/6.0/neon-site/
- http://p2-mirror.sc.dlr.de/releases/oxygen
- http://p2-mirror.sc.dlr.de/tools/orbit/downloads/drops/R20160520211859/repository/
How we deal with dependent projects – Deployment flow

Building by deploying to p2 repositories
- Build dependent artifacts
- Use Jenkins to deploy them to a p2 repository
- Reference deployed artifacts in target platform
How we deal with dependent projects – Configuring the target platform

Target platform of the dependent project

- Option 1: Fixed release numbers
  - Use cases: Release builds, research projects

- Option 2: Always latest build
  - Use cases: Development builds, testing
  - Version number „0.0.0“
  - Can be set by opening target platform with text editor

```xml
<location includeAllPlatforms="false" includeConfigurePhase="true" includeMode="planner" includeSource="true" type="InstallableUnit">
  <unit id="de.dlr.sc.virsat.project.feature.feature.group" version="0.0.0"/>
  <unit id="de.dlr.sc.virsat.svn.feature.feature.group" version="0.0.0"/>
  <unit id="de.dlr.sc.virsat.test.feature.feature.group" version="0.0.0"/>
  <unit id="de.dlr.sc.virsat.uiengine.feature.feature.group" version="0.0.0"/>
  <repository location="file:/U/VirSat/VirSat_Jenkins_Deploy/p2/VirSat4_Core_Application/development/"/>
</location>
```
Why we need multiple build jobs per project

Our Git setup following GitFlow

Different branch types require different build setups!
How we have setup our Jenkins build jobs (1)

**Development build Job**
- Triggers from any push to the development branch
- Triggers dependent projects development builds
- Old builds are cleared
- Target platform uses „0.0.0“ version qualifiers

**Features build Job**
- Triggered upon any push to any branch called feature/*
  - or for a new merge request
- Merge requests to development are only allowed if the build succeeds
How we have setup our Jenkins build jobs (2)

Integration build job
- Setup like the Development build job, but for the integration branch

Release build job
- Builds from manually specified tag
- Build is triggered manually
- Target platform uses version qualifiers for this release
- Each build is persisted and qualified with build job number
- Flushes m2 repository for a clean build
How we setup new dependent projects

Use a template project
- Prepared setup of target platform, master pom, etc.
- ANT script for configuration (e.g. project name)
Thank You!
Evaluate the Sessions
Sign in and vote at eclipsecon.org

-1 0 +1