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

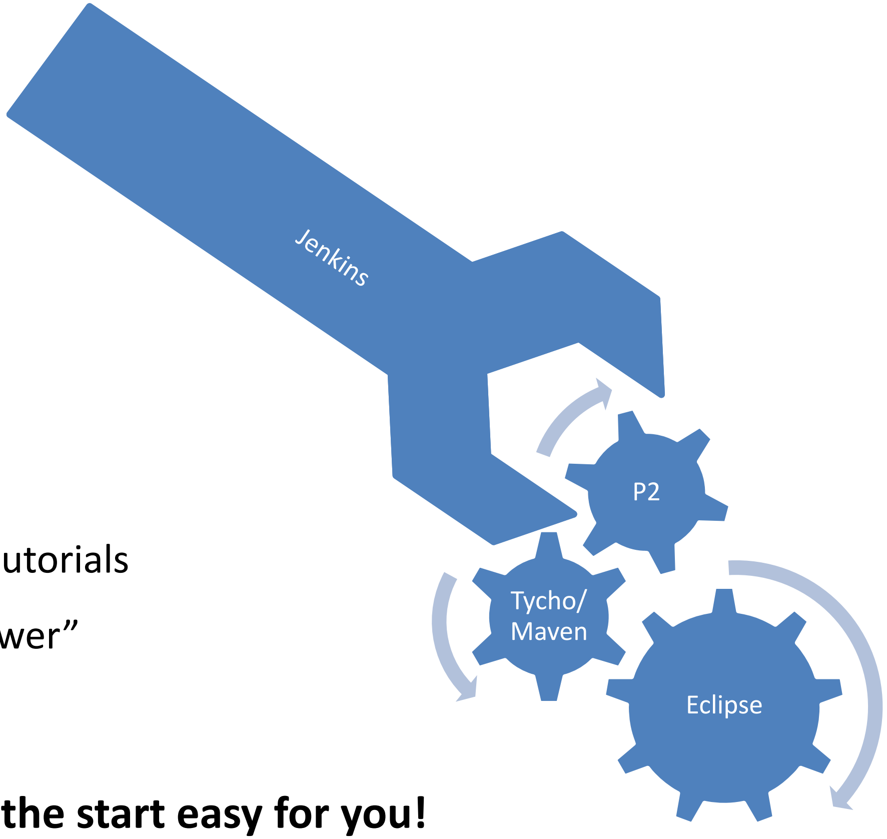
What we hope to deliver to you

Tycho and Maven based build infrastructure

- Automated builds and testing
- Good integration with Eclipse technologies
- But how to set it up?

The Pain

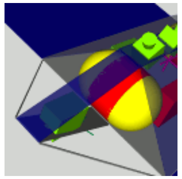
- Only little documentation
- Answers scattered through various forum posts, books and tutorials
- Difficult to answer all projects needs with “one ultimate answer”



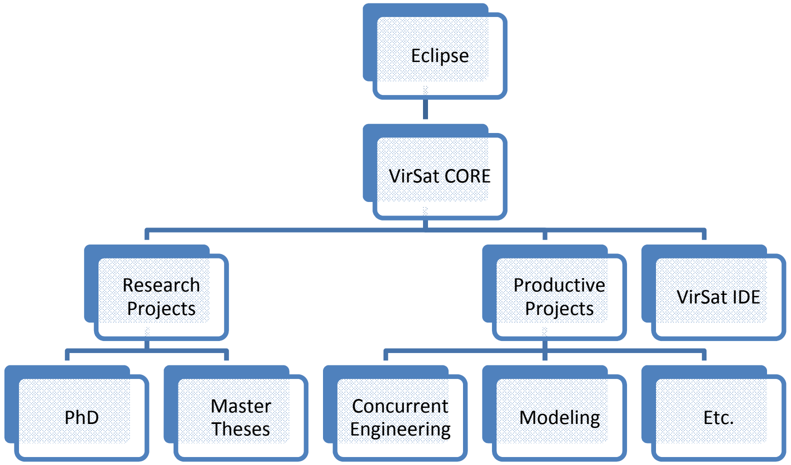
We hope to deliver a compilation of answers to make the start easy for you!

Our Main Project

Virtual Satellite (VirSat)

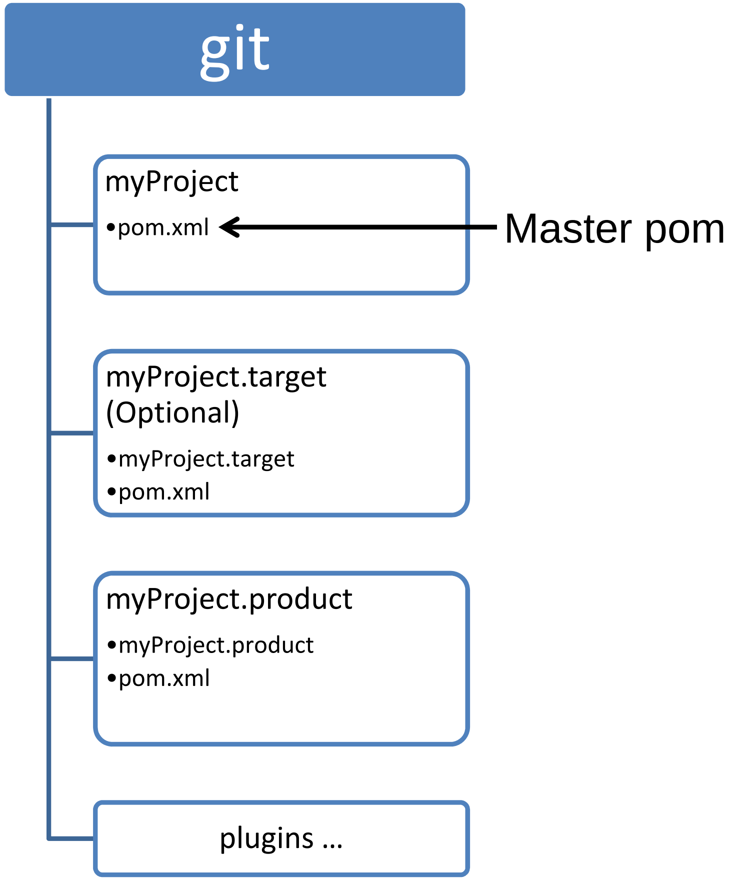


- Eclipse based framework for spacecraft related applications
- For example modelling tools
- Used for research and productive projects by engineers



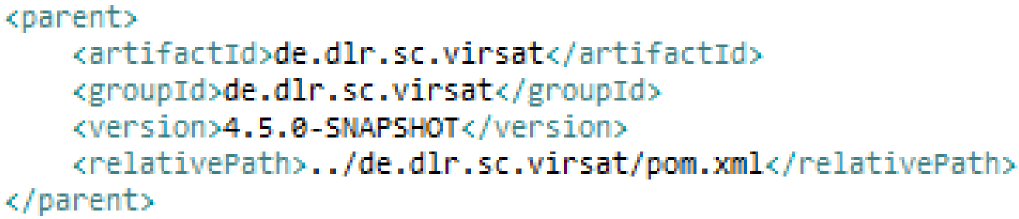
Need for a building and testing infrastructure that can handle all cases!

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



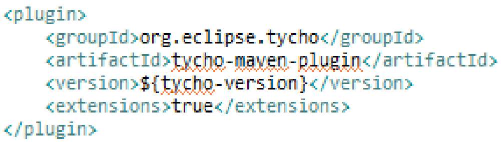
```
<parent>
    <artifactId>de.dlr.sc.virsat</artifactId>
    <groupId>de.dlr.sc.virsat</groupId>
    <version>4.5.0-SNAPSHOT</version>
    <relativePath>../de.dlr.sc.virsat/pom.xml</relativePath>
</parent>
```

Getting Maven to use Tycho

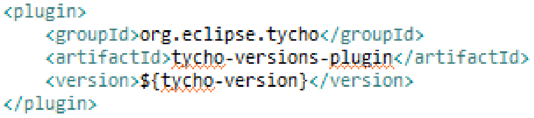
Setting up the tycho plugins in the master pom

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

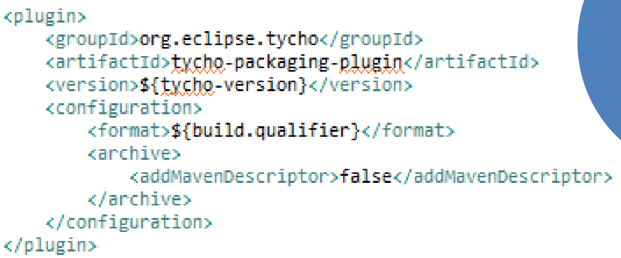
Let‘s discuss them in the following!



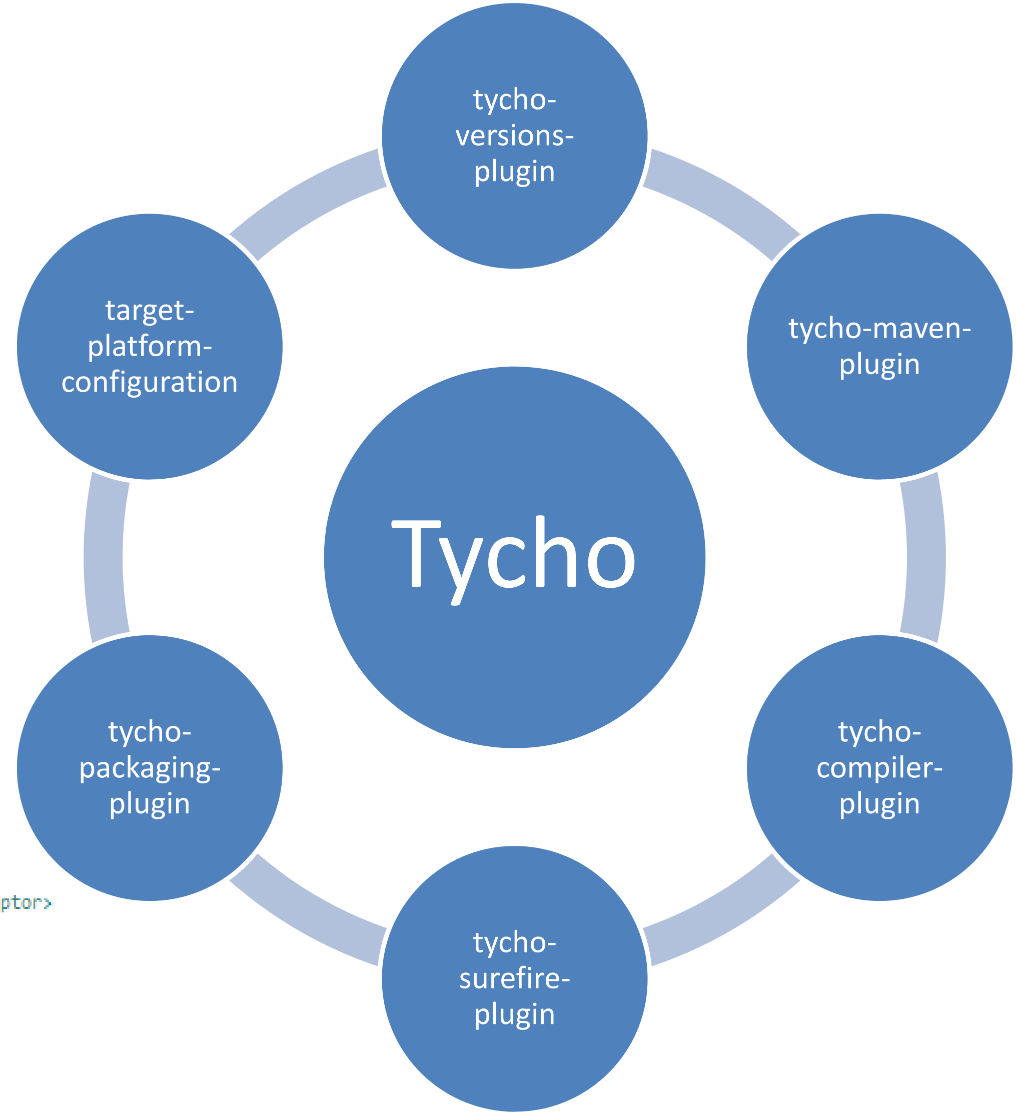
```
<plugin>
    <groupId>org.eclipse.tycho</groupId>
    <artifactId>tycho-maven-plugin</artifactId>
    <version>${tycho-version}</version>
    <extensions>true</extensions>
</plugin>
```



```
<plugin>
    <groupId>org.eclipse.tycho</groupId>
    <artifactId>tycho-versions-plugin</artifactId>
    <version>${tycho-version}</version>
</plugin>
```



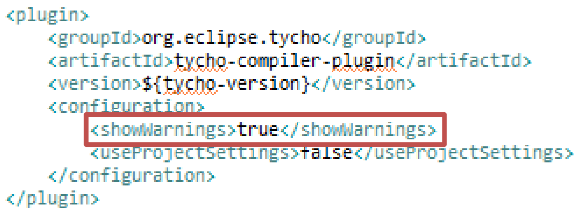
```
<plugin>
    <groupId>org.eclipse.tycho</groupId>
    <artifactId>tycho-packaging-plugin</artifactId>
    <version>${tycho-version}</version>
    <configuration>
        <format>${build.qualifier}</format>
        <archive>
            <addMavenDescriptor>false</addMavenDescriptor>
        </archive>
    </configuration>
</plugin>
```



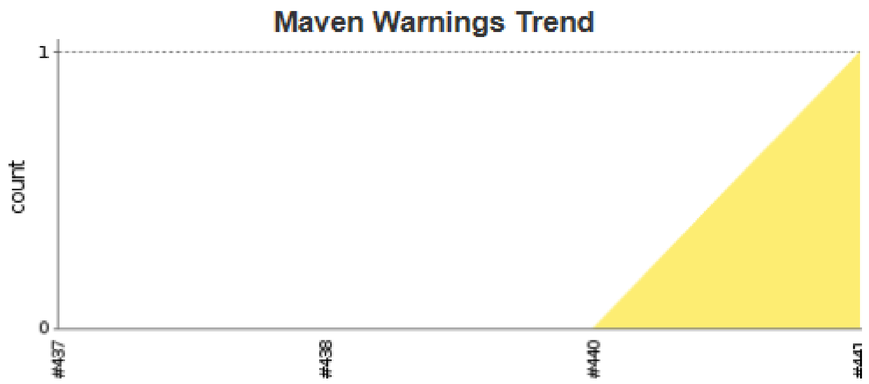
Getting your warnings evaluated on Jenkins

Setting up the compiler

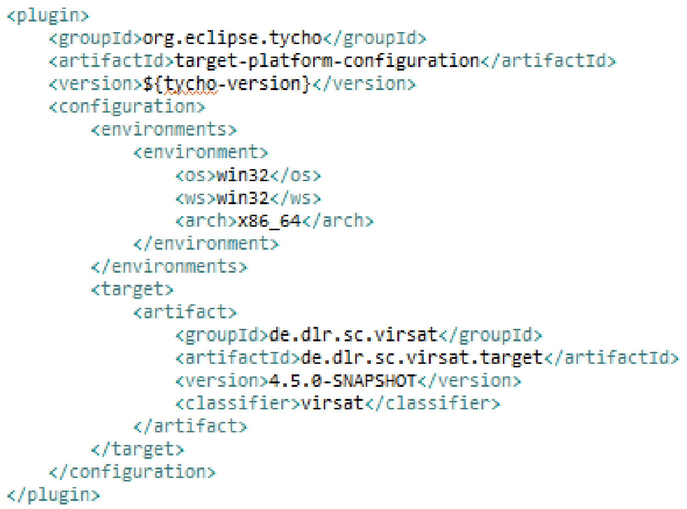
- Add tycho-compiler-plugin to master pom
- Set showWarnings flag so Jenkins can pick up on Java warnings



```
<plugin>
    <groupId>org.eclipse.tycho</groupId>
    <artifactId>tycho-compiler-plugin</artifactId>
    <version>${tycho-version}</version>
    <configuration>
        <showWarnings>true</showWarnings>
        <useProjectSettings>false</useProjectSettings>
    </configuration>
</plugin>
```



Using the IDE target platform in the build

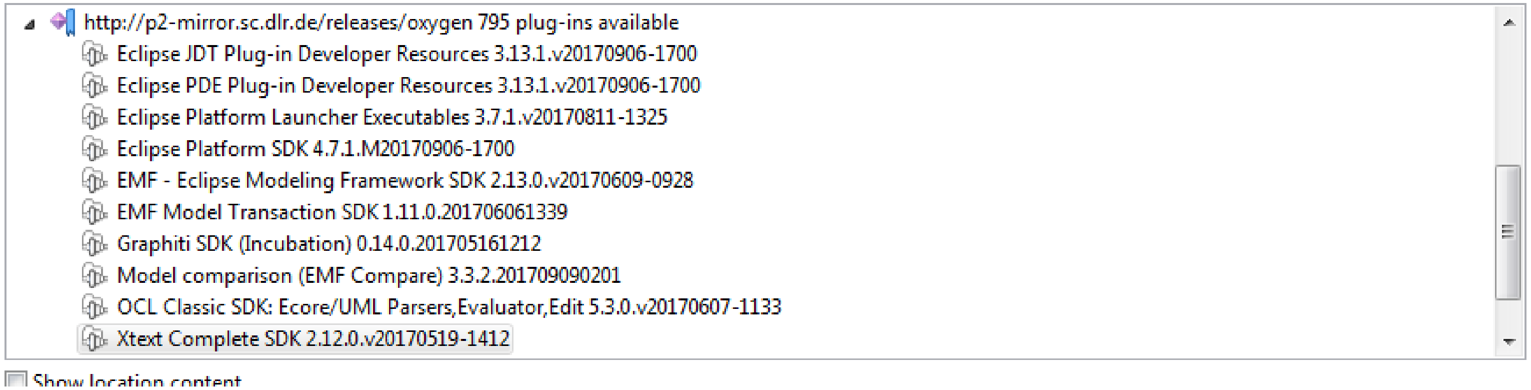


```
<plugin>
    <groupId>org.eclipse.tycho</groupId>
    <artifactId>target-platform-configuration</artifactId>
    <version>${tycho-version}</version>
    <configuration>
        <environments>
            <environment>
                <os>win32</os>
                <ws>win32</ws>
                <arch>x86_64</arch>
            </environment>
        </environments>
        <target>
            <artifact>
                <groupId>de.dlr.sc.virsat</groupId>
                <artifactId>de.dlr.sc.virsat.target</artifactId>
                <version>4.5.0-SNAPSHOT</version>
                <classifier>virsat</classifier>
            </artifact>
        </target>
    </configuration>
</plugin>
```

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!



http://p2-mirror.sc.dlr.de/releases/oxygen 795 plug-ins available
- Eclipse JDT Plug-in Developer Resources 3.13.1.v20170906-1700
- Eclipse PDE Plug-in Developer Resources 3.13.1.v20170906-1700
- Eclipse Platform Launcher Executables 3.7.1.v20170811-1325
- Eclipse Platform SDK 4.7.1.M20170906-1700
- EMF - Eclipse Modeling Framework SDK 2.13.0.v20170609-0928
- EMF Model Transaction SDK 1.11.0.201706061339
- Graphiti SDK (Incubation) 0.14.0.201705161212
- Model comparison (EMF Compare) 3.3.2.201709090201
- OCL Classic SDK: Ecore/UML Parsers,Evaluator,Edit 5.3.0.v20170607-1133
- Xtext Complete SDK 2.12.0.v20170519-1412

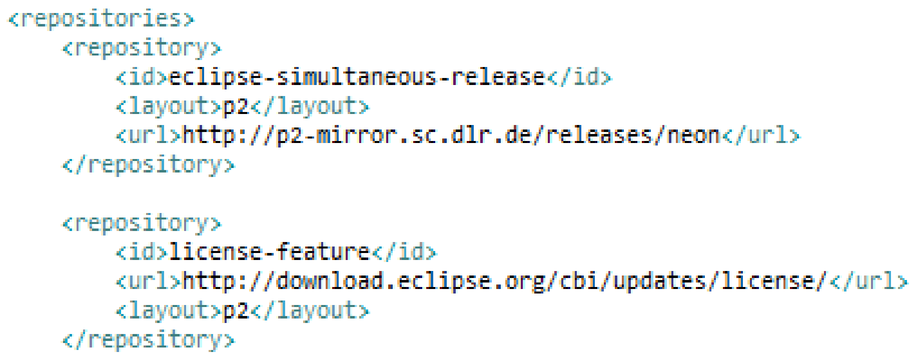
□ Show location content

Getting dependencies directly with Maven

Declaring repositories in the master pom

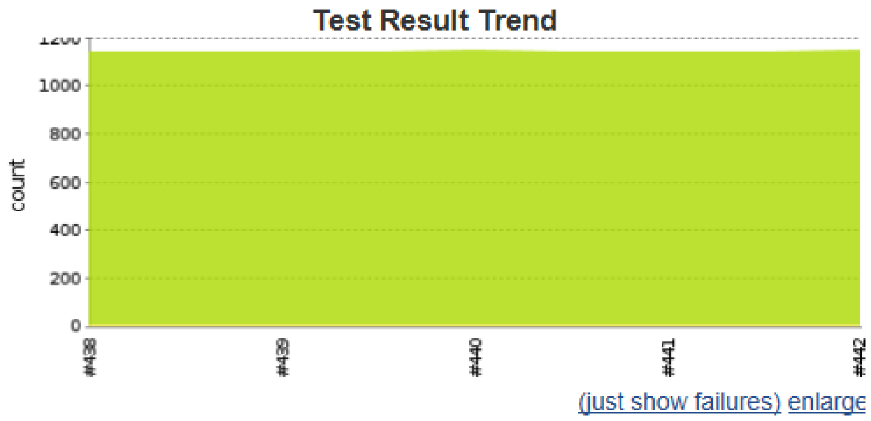
- Change in product often means change in the target platform
- By declaring repositories Maven/Tycho can get all dependencies
- Use case: Projects without own source code
  - Example: VirSat IDE

    (all features developed in CORE)



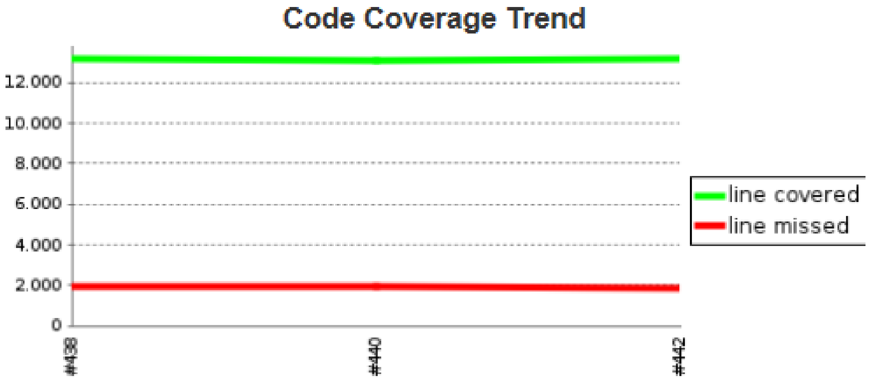
```
<repositories>
    <repository>
        <id>eclipse-simultaneous-release</id>
        <layout>p2</layout>
        <url>http://p2-mirror.sc.dlr.de/releases/neon</url>
    </repository>

    <repository>
        <id>license-feature</id>
        <url>http://download.eclipse.org/cbi/updates/license/</url>
        <layout>p2</layout>
    </repository>
```

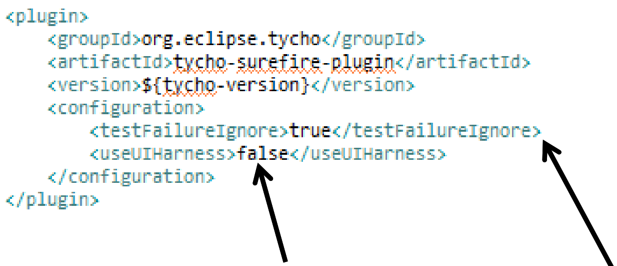
Setting up your testing environment



Testing in headless mode

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





```
<plugin>
    <groupId>org.eclipse.tycho</groupId>
    <artifactId>tycho-surefire-plugin</artifactId>
    <version>${tycho-version}</version>
    <configuration>
        <testFailureIgnore>true</testFailureIgnore>
        <useUIHarness>false</useUIHarness>
    </configuration>
</plugin>
```

Run in headless mode

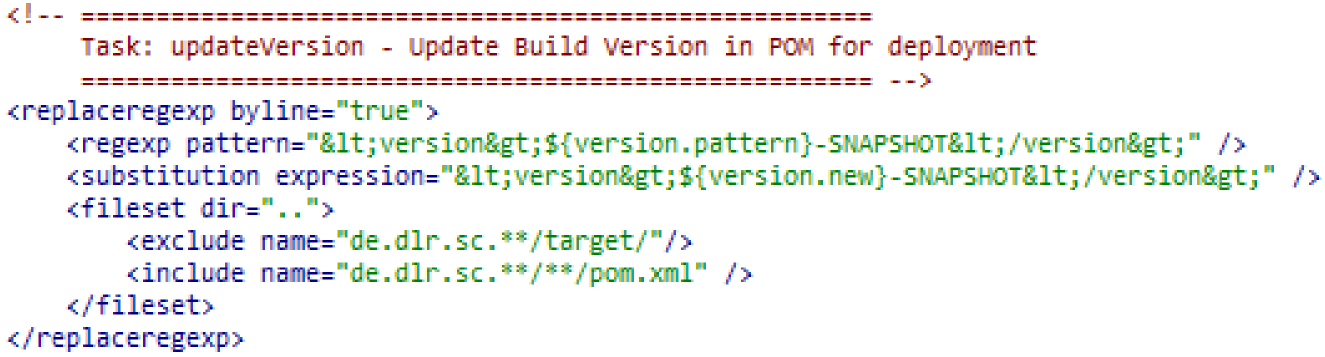
Continue building even if there is a test failure.

Keeping version numbers up to date

Using ant script to update version numbers

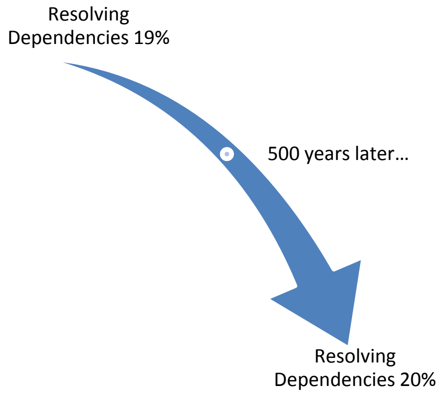
- A lot of files specify the plugin version
- Need to be updated when software version is incremented

Automatically update version numbers with an ant script!



```
<!-- =====================================================
     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



Resolving Dependencies 19%

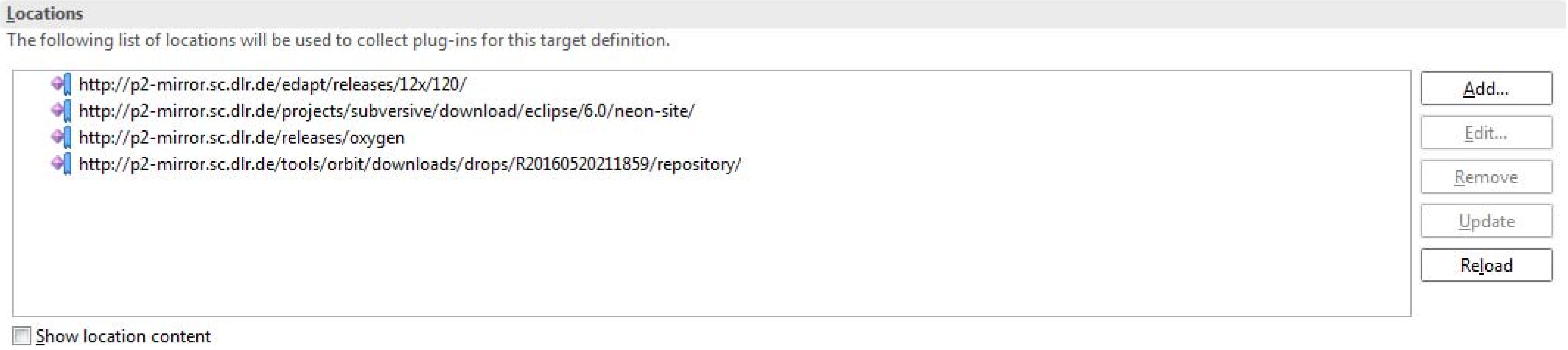
500 years later…

Resolving Dependencies 20%

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.

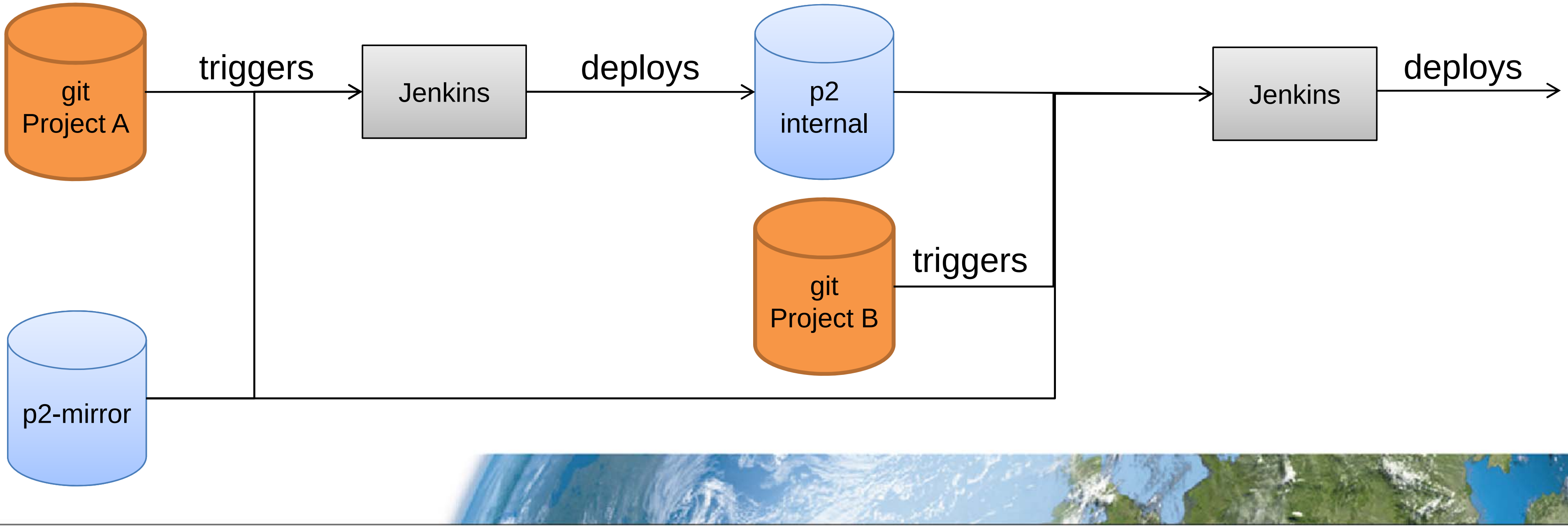
- http://p2-mirror.sc.dlr.de/edapt/releases/12x/120/
- http://p2-mirror.sc.dlr.de/projects/subversive/download/eclipse/6.0/neon-site/
- http://p2-mirror.sc.dlr.de/releases/oxygen
- http://p2-mirror.sc.dlr.de/tools/orbit/downloads/drops/R20160520211859/repository/

□ Show location content

How we deal with dependent projects – Deployment flow

Building by deploying to p2 repositories

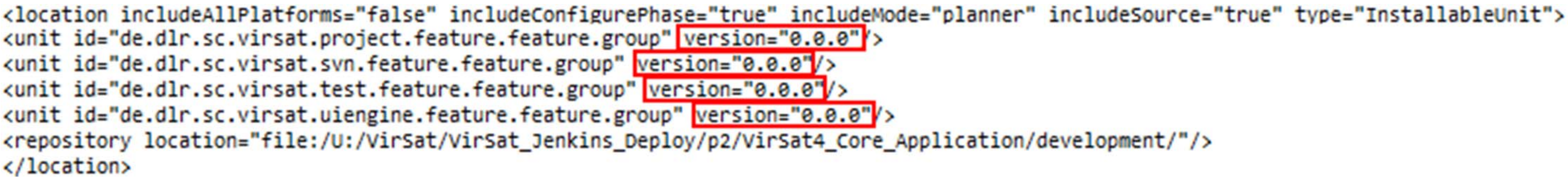
- Build dependent artifacts
- Use Jenkins to deploy them to a p2 repository
- Reference deployed artifacts in target platform



How we deal with dependent projects – Configuring the target platform

Target platform of the dependent project

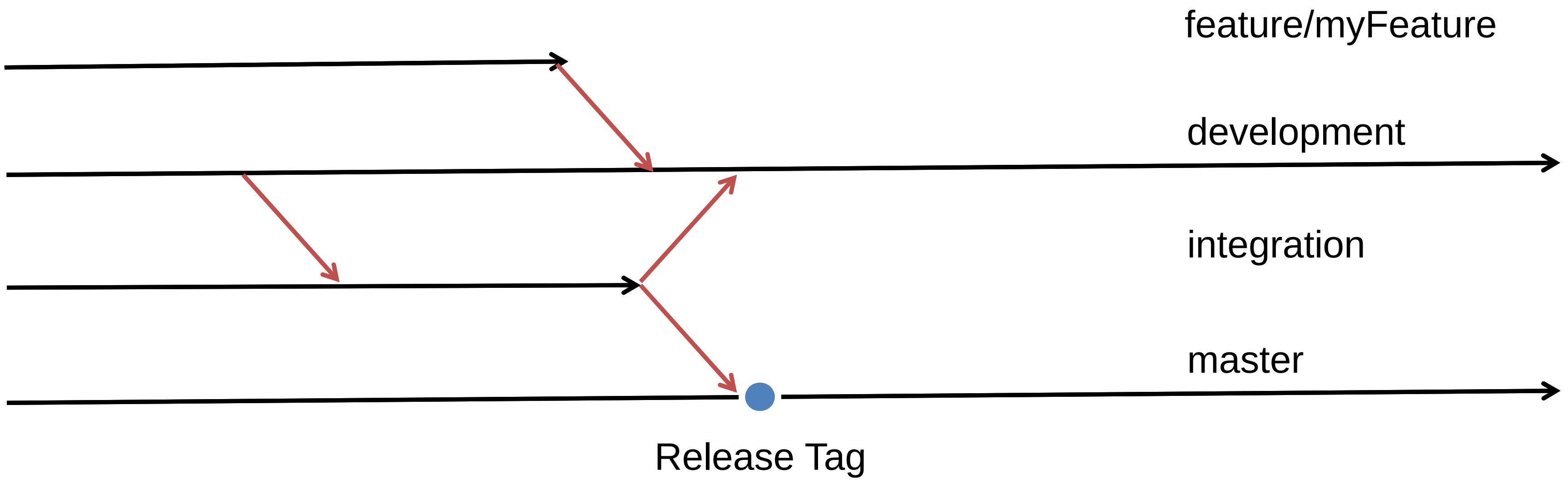
- Option 1: Fixed release numbers
  - Use cases: Release builds, research projects
- Option 2: Always latest build
  - Use cases: Development builds, testing
  - Version number „0.0.0“
  - Can be set by opening target platform with text editor



```
<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

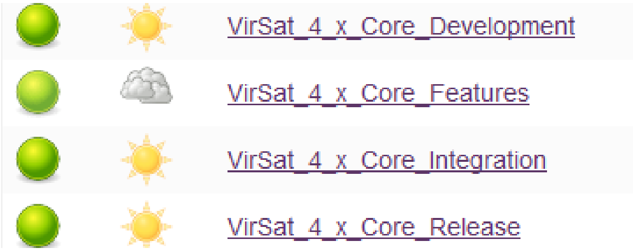


Different branch types require different build setups!

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



●	☀	VirSat_4_x_Core_Development
●	☁	VirSat_4_x_Core_Features
●	☀	VirSat_4_x_Core_Integration
●	☀	VirSat_4_x_Core_Release

Features build Job

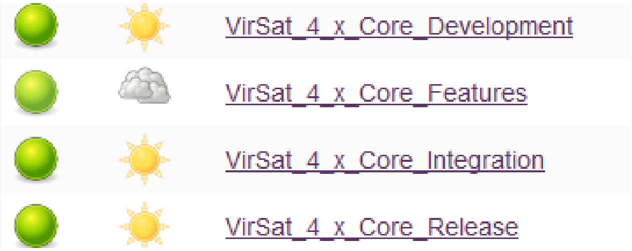
- Triggered upon any push to any branch called **feature/***

  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



● ☀ VirSat_4_x_Core_Development

● ☁ VirSat_4_x_Core_Features

● ☀ VirSat_4_x_Core_Integration

● ☀ VirSat_4_x_Core_Release

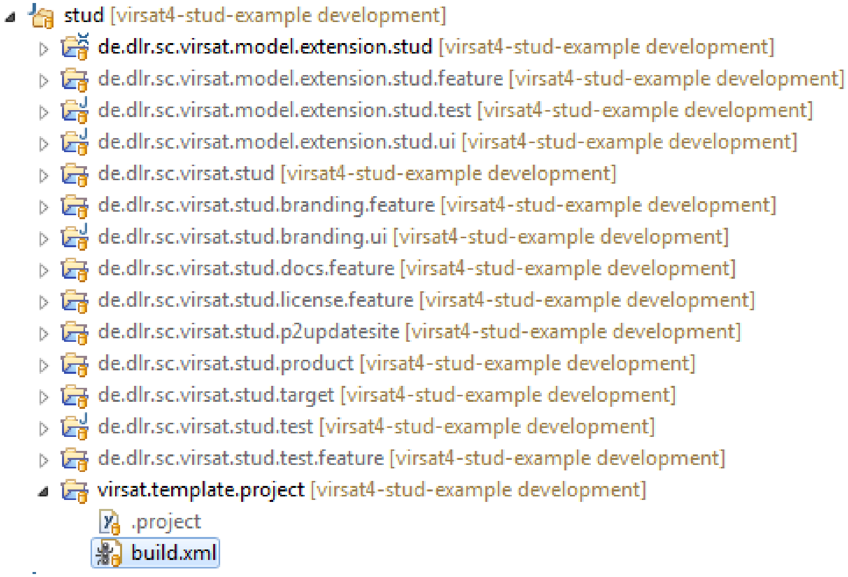
Release build job

- 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

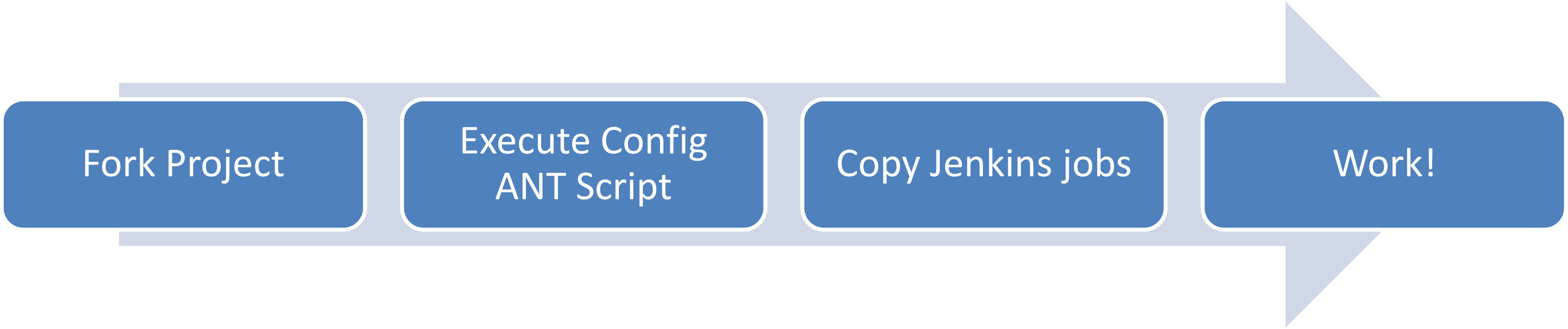
How we setup new dependent projects

Use a template project

- Prepared setup of target platform, master pom, etc.
- ANT script for configuration (e.g. project name)



```
stud [virsat4-stud-example development]
  de.dlr.sc.virsat.model.extension.stud [virsat4-stud-example development]
  de.dlr.sc.virsat.model.extension.stud.feature [virsat4-stud-example development]
  de.dlr.sc.virsat.model.extension.stud.test [virsat4-stud-example development]
  de.dlr.sc.virsat.model.extension.stud.ui [virsat4-stud-example development]
  de.dlr.sc.virsat.stud [virsat4-stud-example development]
  de.dlr.sc.virsat.stud.branding.feature [virsat4-stud-example development]
  de.dlr.sc.virsat.stud.branding.ui [virsat4-stud-example development]
  de.dlr.sc.virsat.stud.docs.feature [virsat4-stud-example development]
  de.dlr.sc.virsat.stud.license.feature [virsat4-stud-example development]
  de.dlr.sc.virsat.stud.p2updatesite [virsat4-stud-example development]
  de.dlr.sc.virsat.stud.product [virsat4-stud-example development]
  de.dlr.sc.virsat.stud.target [virsat4-stud-example development]
  de.dlr.sc.virsat.stud.test [virsat4-stud-example development]
  de.dlr.sc.virsat.stud.test.feature [virsat4-stud-example development]
  virsat.template.project [virsat4-stud-example development]
    .project
    build.xml
```



Thank You!

eclipsecon Europe 2017

OSGi™ Community Event 2017

Evaluate the Sessions

Sign in and vote at **eclipsecon.org**

- 1 0 + 1