

# Is Maven a solution to occurring build path problems?

Dominik Grether

[www.matsim.org](http://www.matsim.org)

Transport Systems Planning and Transport Telematics  
Technical University Berlin

18.07.2008

# Introduction

## Why?

- MATSim benefits from libraries, e.g. Geotools

# Introduction

## Why?

- MATSim benefits from libraries, e.g. Geotools
- Libraries need to be added to build path manually when checkout

# Introduction

## Why?

- MATSim benefits from libraries, e.g. Geotools
- Libraries need to be added to build path manually when checkout
- Manual addition time consuming, order not specified

# Introduction

## Why?

- MATSim benefits from libraries, e.g. Geotools
- Libraries need to be added to build path manually when checkout
- Manual addition time consuming, order not specified
- Correctness/Runability of code dependend on a not specified order

# Preliminaries

## Java class names

- Unique identifier of Java class: fully qualified name, e.g. `java.lang.String`
- See also [http://java.sun.com/docs/books/jvms/second\\_edition/html/Concepts.doc.html#20207](http://java.sun.com/docs/books/jvms/second_edition/html/Concepts.doc.html#20207)
- However no “.jar” information used to identify class

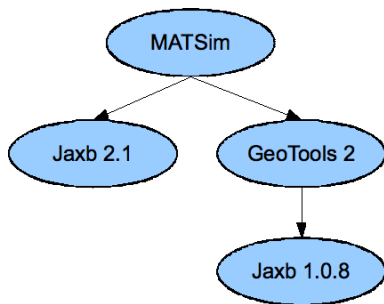
# Preliminaries

## Java classpath

- Classes loaded by `java.lang.ClassLoader`
- Java classpath specifies locations for `ClassLoader` to search classes
- First name matching with fully qualified name is loaded

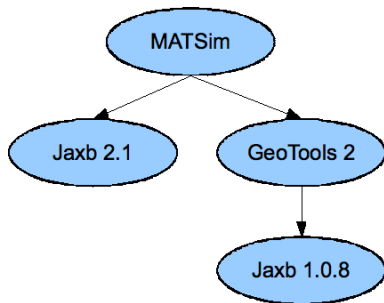
# Problem

- MATSim using Jaxb 2.1 and Geotools 2



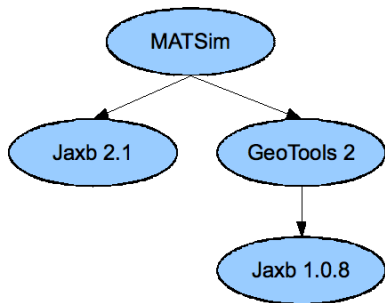
# Problem

- MATSim using Jaxb 2.1 and Geotools 2
- Geotools 2 using Jaxb 1.0.8



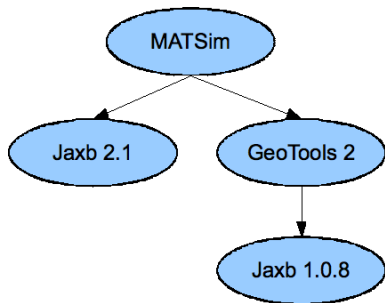
# Problem

- MATSim using Jaxb 2.1 and Geotools 2
- Geotools 2 using Jaxb 1.0.8
- Same fully qualified class names in Jaxb versions



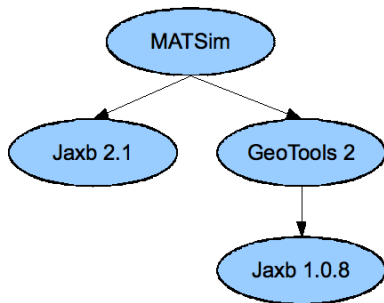
# Problem

- MATSim using Jaxb 2.1 and Geotools 2
- Geotools 2 using Jaxb 1.0.8
- Same fully qualified class names in Jaxb versions
- Depending on order of libraries in classpath Jaxb 2.1 OR Jaxb 1.0.8 classes loaded



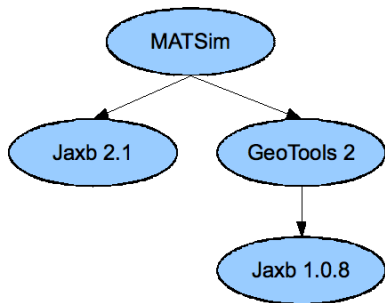
# Problem

- MATSim using Jaxb 2.1 and Geotools 2
- Geotools 2 using Jaxb 1.0.8
- Same fully qualified class names in Jaxb versions
- Depending on order of libraries in classpath Jaxb 2.1 OR Jaxb 1.0.8 classes loaded
- Jaxb code runs only with one version



# Problem

- MATSim using Jaxb 2.1 and Geotools 2
- Geotools 2 using Jaxb 1.0.8
- Same fully qualified class names in Jaxb versions
- Depending on order of libraries in classpath Jaxb 2.1 OR Jaxb 1.0.8 classes loaded
- Jaxb code runs only with one version
- Problem known as Jar Hell, see [http://en.wikipedia.org/wiki/JAR\\_hell#JAR\\_hell](http://en.wikipedia.org/wiki/JAR_hell#JAR_hell)



# Maven

## Introduction

- Maven: software project management and comprehension tool
- `www.maven.apache.org`
- Dependency management including automatic updating, dependency closures
- “Eierlegende Wollmilchsau”
- More features: <http://maven.apache.org/maven-features.html>

# Maven

## Project object model

- Project object model: POM
- “Fundamental unit of work in Maven”
- XML definition of:
  - Dependency management
  - Remote repositories
  - Build logic, e.g. source directory, target directory
  - Subprojects, e.g. Matsim OTFVis
- Tool/IDE independent

# Maven

## POM Example

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.matsim</groupId>
  <artifactId>matsim</artifactId>
  <name>MATSim</name>
  <version>0.1</version>
  <description></description>
  <url>matsim.org</url>
  <inceptionYear>2006</inceptionYear>
  <licenses>
    <license>
      <name>GNU General Public License</name>
      <url>{BASEDIR}/COPYING</url>
      <comments>See also the LICENSE file</comments>
    </license>
  </licenses>
  ...

```

# Maven

## POM Example

```
...
<build>
  <sourceDirectory>src/</sourceDirectory>
  <testSourceDirectory>test/src/</testSourceDirectory>
  <outputDirectory>bin/</outputDirectory>
  <testOutputDirectory>bin/</testOutputDirectory>
  <resources>
    <resource>
      <directory>./</directory>
      <includes>
        <include>res/**/*.*</include>
        <include>dtd/**/*.*</include>
      </includes>
    </resource>
  </resources>
...

```

# Maven

## POM Example

```
...
<plugins>
  <plugin>
    <artifactId>maven-compiler-plugin</artifactId>
    <configuration>
      <source>1.5</source>
      <target>1.5</target>
    </configuration>
  </plugin>
  <plugin>
    <artifactId>maven-assembly-plugin</artifactId>
    <configuration>
      <descriptorRefs>
        <descriptorRef>jar-with-dependencies</descriptorRef>
      </descriptorRefs>
    </configuration>
  </plugin>
</plugins>
</build>
...
```

# Maven

## POM Example

```
...
<repositories>
  <repository>
    <id>geotools</id>
    <name>Geotools repository</name>
    <url>http://maven.geotools.fr/repository</url>
  </repository>
  <repository>
    <id>ibiblio</id>
    <name>Ibiblio - the public's library and digital archive</name>
    <url>http://www.ibiblio.org/maven2</url>
  </repository>
</repositories>
```

# Maven

## POM Example

```
...
<dependencies>
  <dependency>
    <groupId>trove</groupId>
    <artifactId>trove</artifactId>
    <version>2.0.3</version>
    <scope>system</scope>
    <systemPath>${basedir}/libs/trove-2.0.3/lib/trove-2.0.3.jar</systemPath>
  </dependency>
  ...
  <dependency>
    <groupId>org.geotools</groupId>
    <artifactId>gt2-main</artifactId>
    <version>2.3.5</version>
  </dependency>
  <dependency>
    <groupId>org.geotools</groupId>
    <artifactId>gt2-shapefile</artifactId>
    <version>2.3.5</version>
  </dependency>
</dependencies>
```

# Maven

## Maven Lifecycle

- Maven build is divided into several subgoals, e.g.
  - compile
  - test
  - package
  - verify
  - install
  - deploy
- See: `http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html`

# Maven

## Dependency management

- Maven comes with sophisticated dependency management
- Central online repository for libraries
- Downloaded to central repository (`~/.m2/`)
- Supports also “system scope” libraries not in repository
- Transient dependencies: dependencies of dependencies
- Maven resolves transient dependencies: Nearest definition is used, if equal first definition
- $\Rightarrow$  Maven filling Java specification gap

# Maven

## Installation

- [matsim.org](http://matsim.org) → documentation → building matsim with maven
- or <http://matsim.org/node/133>

# Maven

Demo

Demo

# Summary

- There might be more convenient solutions
- Good experiences with Ant (also [apache.org](http://apache.org))
- Maven based on experiences with Ant, can be seen as enhancement
- Maven would solve our dependency problems...
- ...and in the future some other tasks

# References

- Java VM Spec [http://java.sun.com/docs/books/jvms/second\\_edition/html/VMSpecTOC.doc.html](http://java.sun.com/docs/books/jvms/second_edition/html/VMSpecTOC.doc.html)
- Jar hell at Wikipedia  
[http://en.wikipedia.org/wiki/JAR\\_hell#JAR\\_hell](http://en.wikipedia.org/wiki/JAR_hell#JAR_hell)
- Specification change request  
<http://www.jcp.org/en/jsr/detail?id=277>
- Maven book (beta)  
[http://www.sonatype.com/community/definitive\\_guide.html](http://www.sonatype.com/community/definitive_guide.html)
- Maven home <http://maven.apache.org>
- Getting started (5 min) <http://maven.apache.org/guides/getting-started/maven-in-five-minutes.html>
- Getting started <http://maven.apache.org/guides/getting-started/index.html>
- POM overview <http://maven.apache.org/guides/introduction/introduction-to-the-pom.html>