

# Module, Versionen, CI & Release

**Martin Eigenbrodt,  
[Martin.Eigenbrodt@innoq.com](mailto:Martin.Eigenbrodt@innoq.com)**

# Über mich

- ▶ **Consultant bei innoQ DE**
- ▶ **JavaEE**
- ▶ **Build**

# Vorweg...

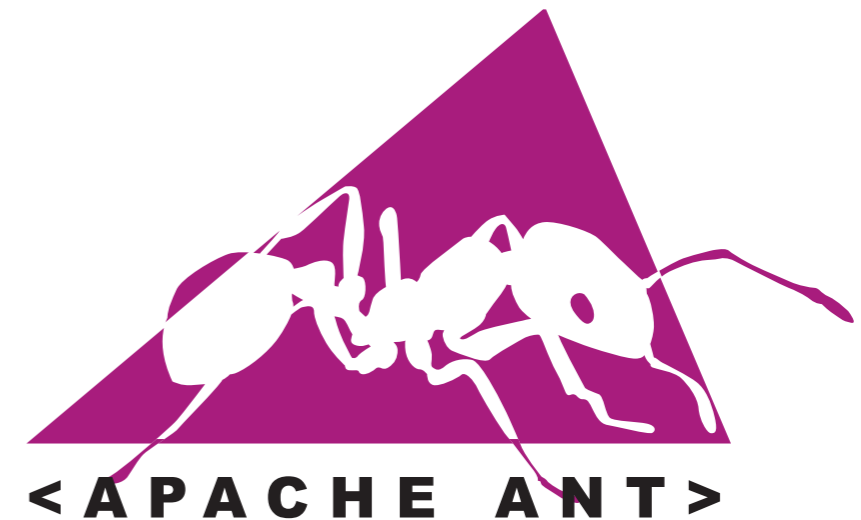
- ▶ **Konzepte im Vordergrund**
- ▶ **Ant, Ivy & Hudson nur Beispiel**
- ▶ **Denkanregung**
- ▶ **Keine „Out of the box“ Lösung**

# Tools

- ▶ **Ant**
- ▶ **Ivy**
- ▶ **Hudson**

# Apache Ant

- ▶ **Plattform unabhängiges Make**
- ▶ **Viele Tasks, z.b. Kopieren, Compilieren, Zippen**
- ▶ **Erweiterbar**
- ▶ **Apache Lizenz**



# build.xml

```
<project name="MyProject" default="dist" basedir=".">
  <property name="src" value="src"/>
  <property name="build" value="build"/>
  <property name="dist" value="dist"/>

  <target name="compile">
    <javac srcdir="${src}" destdir="${build}"/>
  </target>

  <target name="dist" depends="compile">
    <mkdir dir="${dist}/lib"/>
    <jar jarfile="${dist}/lib/MyProject.jar" basedir="${build}"/>
  </target>
</project>
```

# Apache Ivy

- ▶ **Dependency Manager**
- ▶ **Erweiterung zu Ant**
- ▶ **Flexibel**



# ivy.xml

```
<ivy-module version="2.0">
<info organisation="org.innoq.build" module="example" />
<configurations>
  <conf name="default" />
  <conf name="test" visibility="private" />
</configurations>

<dependencies>
  <dependency org="junit" name="junit" rev="4.8.1"
conf="test->*" />
</dependencies>
</ivy-module>
```

# Hudson

- ▶ **CI-Server**
- ▶ **Leicht konfigurierbar**
- ▶ **Plugin Architektur**



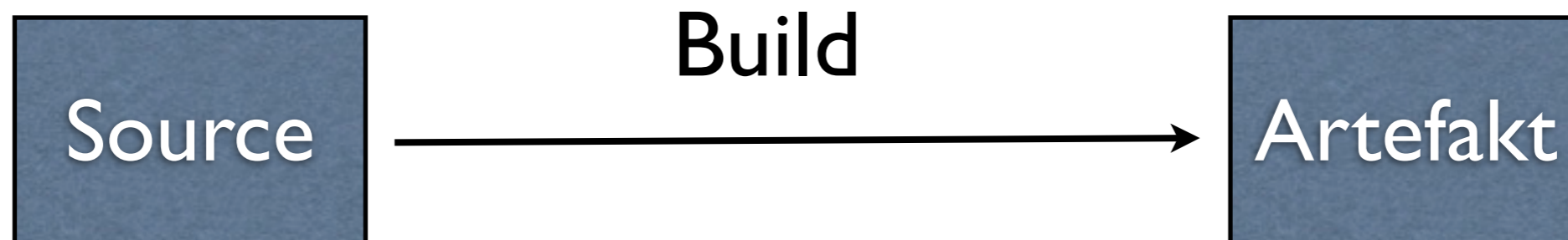


# Begriffe

(und Schlussfolgerungen)

# Build

**Ein Build ist die Transformation von ein oder mehreren Quellen in ein Zielartefakt**



# Release

**Ein „Release“ ist die Weitergabe einer Software an eine bestimmte Gruppe von Personen**

# Release

**Ein „Release“ ist die Weitergabe einer Software an eine bestimmte Gruppe von Personen**

Auch die weitergegeben Software heißt Release

# Continuous Integration

Continuous Integration is a software development practice where members of a team integrate their work frequently [...].

Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.

(M. Fowler)

# Continuous Integration

- ▶ **Integration muss stetig erfolgen**
- ▶ **automatisiert**
- ▶ **kein Delta zum Release**

# Continuous Integration

- ▶ **CI. Build = Release Build**

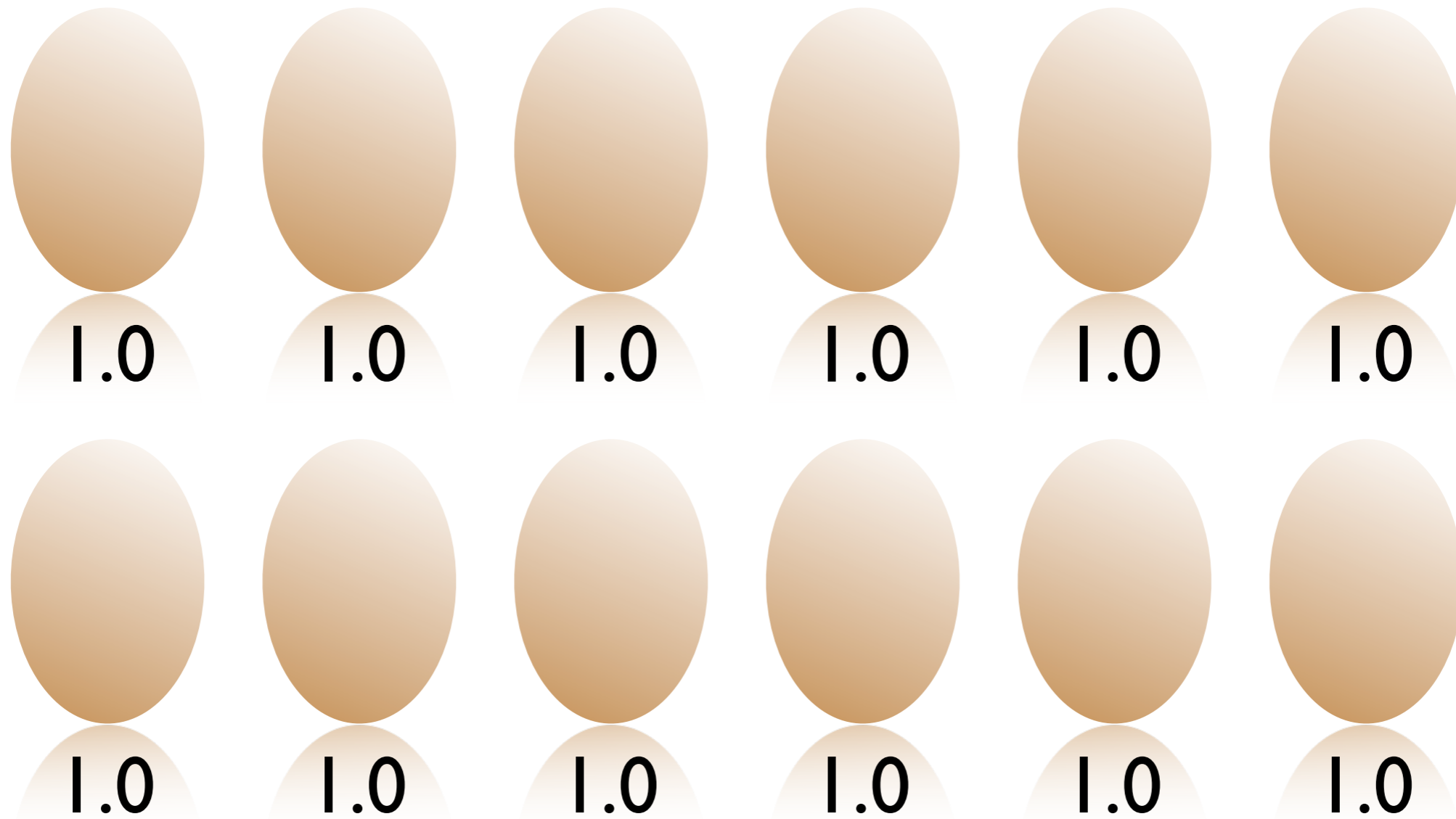
# Continuous Integration

- ▶ **CI. Build = Release Build**
- ▶ ~~**Release Build**~~

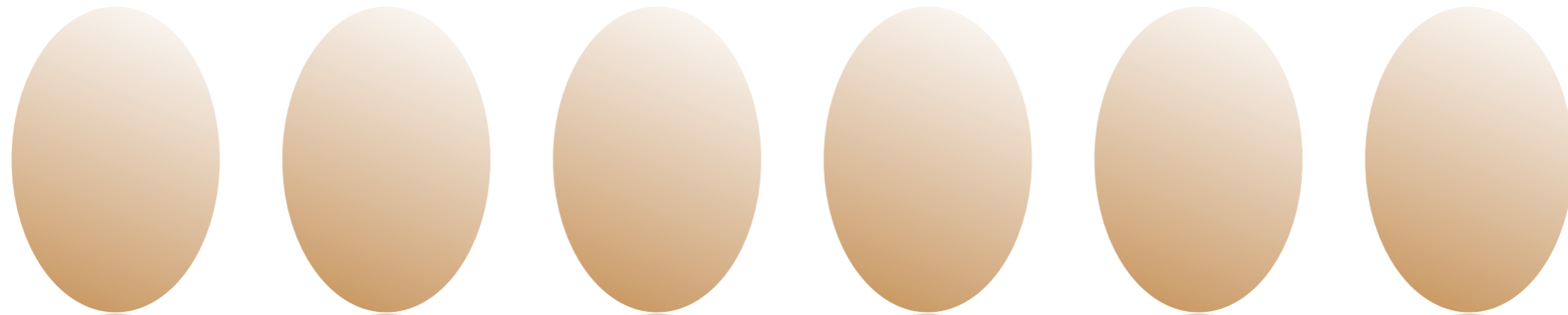
# Continuous Integration

- ▶ **CI. Build = Release Build**
- ▶ ~~**Release Build**~~
- ▶ **Jeder CI-Builds ist ein  
potentielles Release**

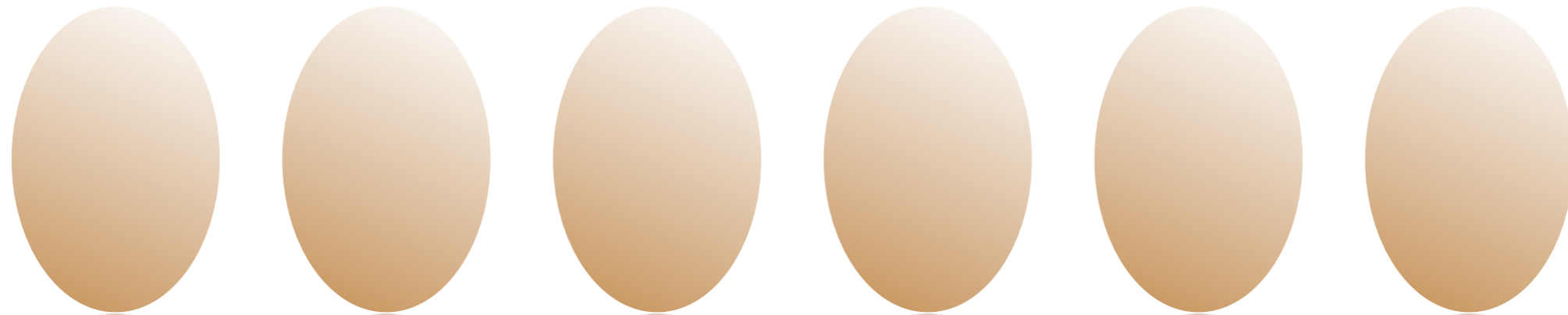
# Finde den Unterschied...



# Besser?

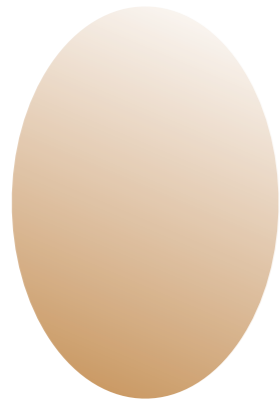


I.0-Snapshot I.0-Snapshot I.0-Snapshot I.0-Snapshot I.0-Snapshot I.0-Snapshot

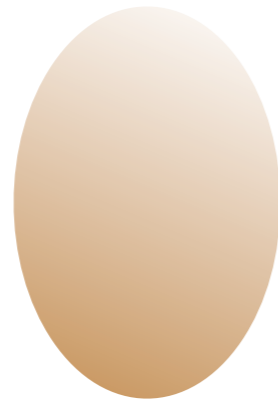


I.0-Snapshot I.0-Snapshot I.0-Snapshot I.0-Snapshot I.0-Snapshot I.0-Snapshot

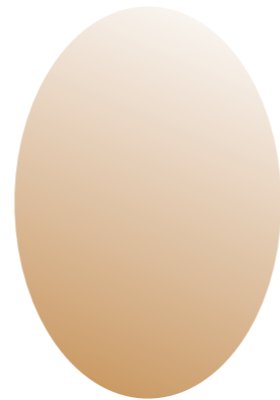
# Ah!



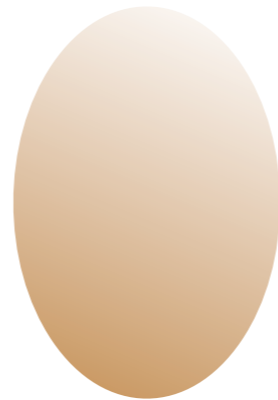
1.0.1



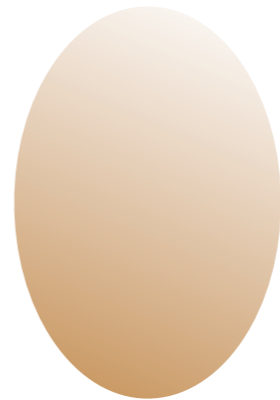
1.0.2



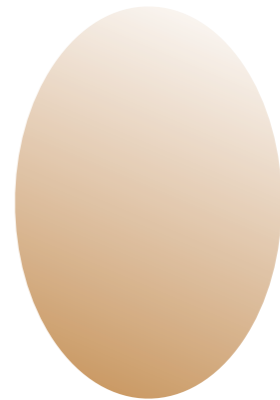
1.0.3



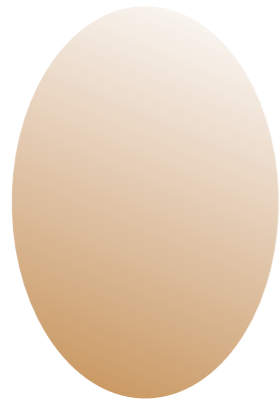
1.0.4



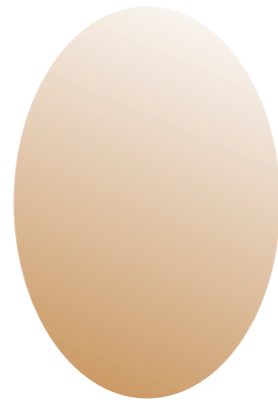
1.0.5



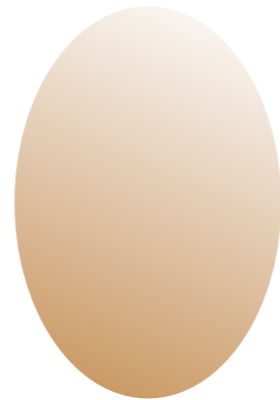
1.0.6



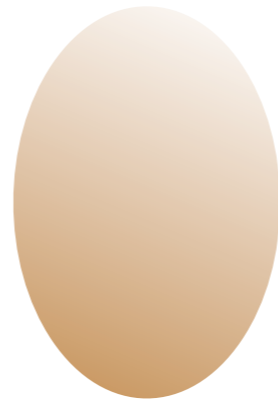
1.0.7



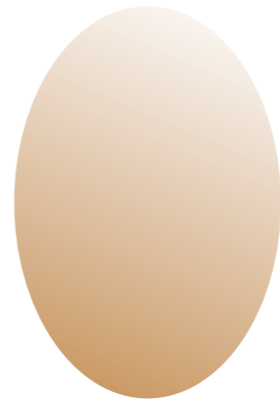
1.0.8



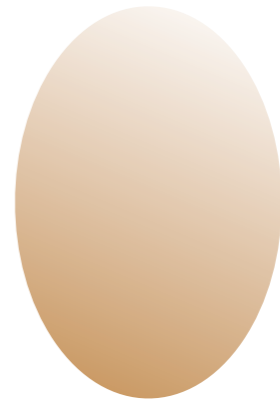
1.0.9



1.0.10



1.0.11



1.0.12

# Versionsnummer

Identifiziert zusammen mit einem Namen **eindeutig** ein Artefakt

# Versionsnummer

Identifiziert zusammen mit einem Namen **eindeutig** ein Artefakt

- ▶ Mehrfacher Builds -> verschiedene Versionen

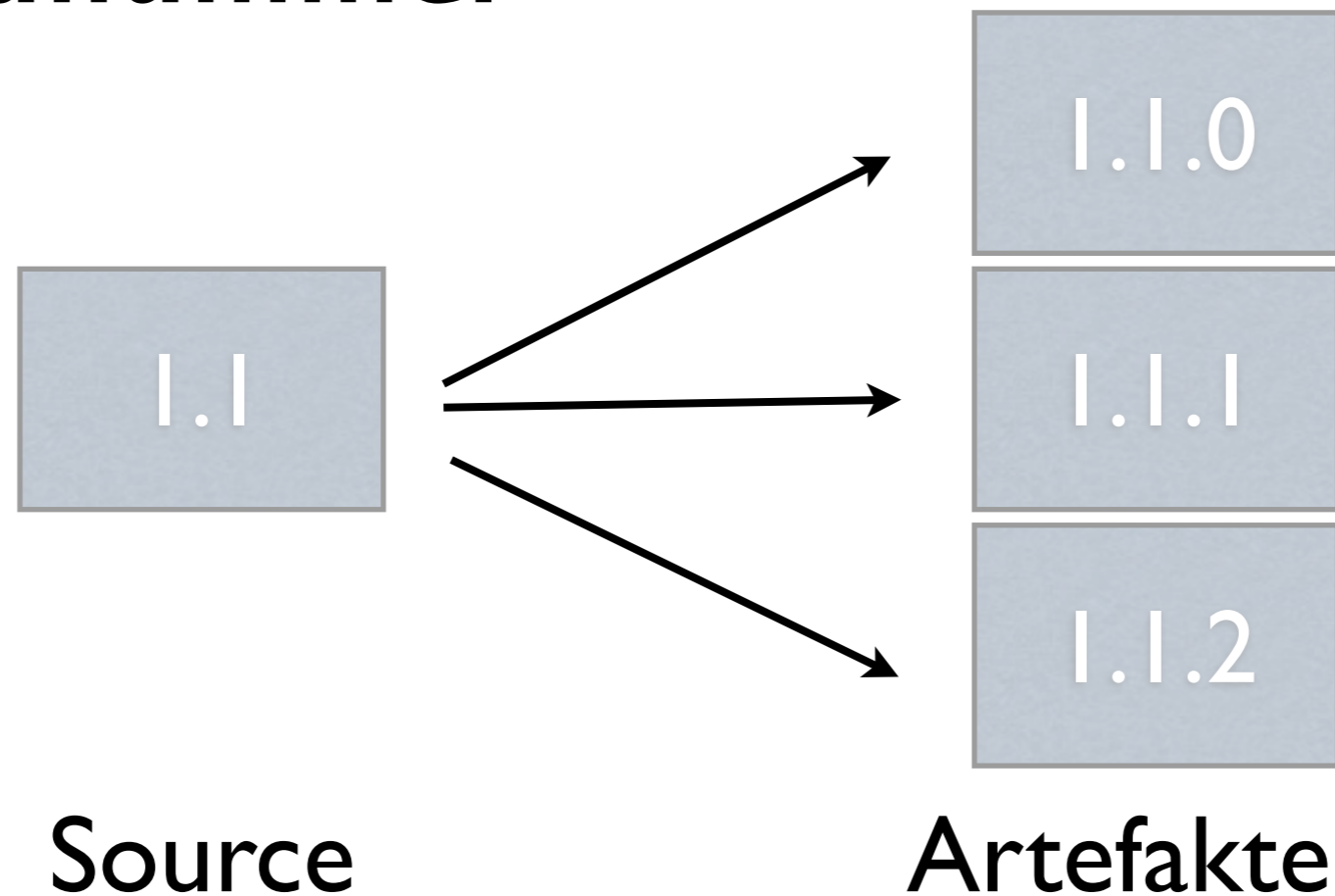
# Versionsnummer

Identifiziert zusammen mit einem Namen **eindeutig** ein Artefakt

- ▶ Mehrfacher Builds -> verschiedene Versionen
- ▶ -> Versionsnummer nicht in den Sourcen

# Versionsnummer

Zum Beispiel durch automatische  
Buildnummer



# Anforderungen

- ▶ **eindeutige Versionsnummer**
- ▶ **Jeder Build ein Release**
- ▶ **Reproduzierbar**
- ▶ **Sourcen eindeutig zugeordnet**
- ▶ **Abhängigkeiten eindeutig zugeordnet**

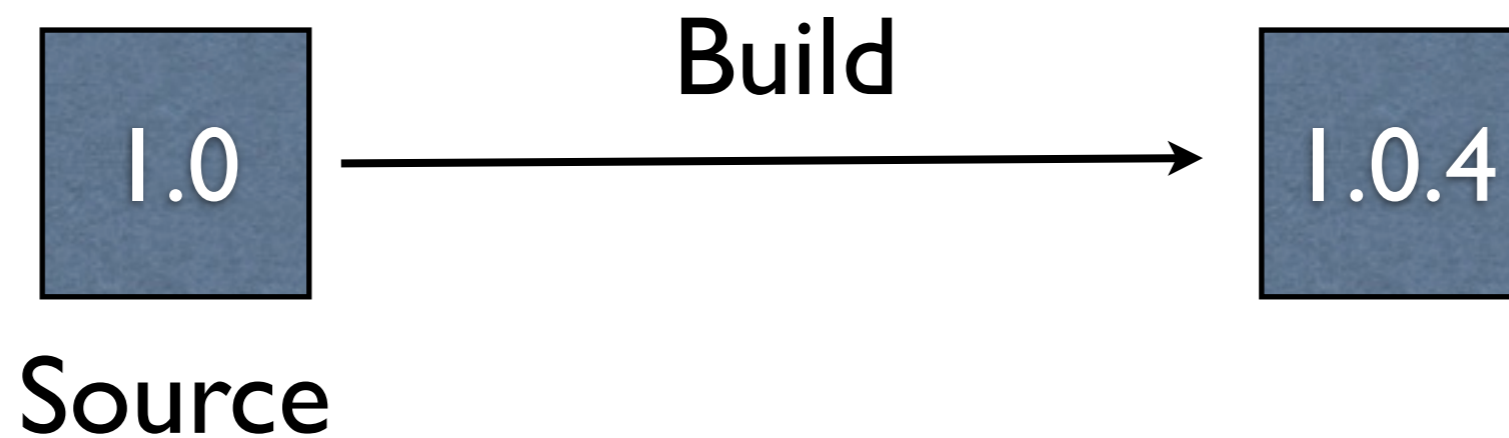
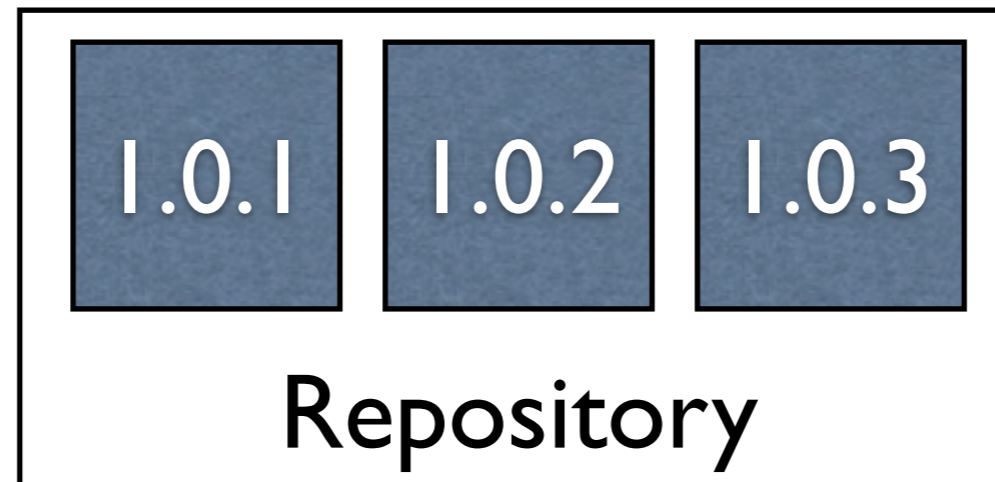
# Umsetzung

# Buildnummer

## Alternativen

- ▶ **automatisch erhöhte Zahl im Source**
- ▶ **Zeitstempel**
- ▶ **Nächste Zahl auf Basis existierender Artefakte**

# Buildnummer



# Zuordnung Sourcen

## Möglichkeiten der Umsetzung

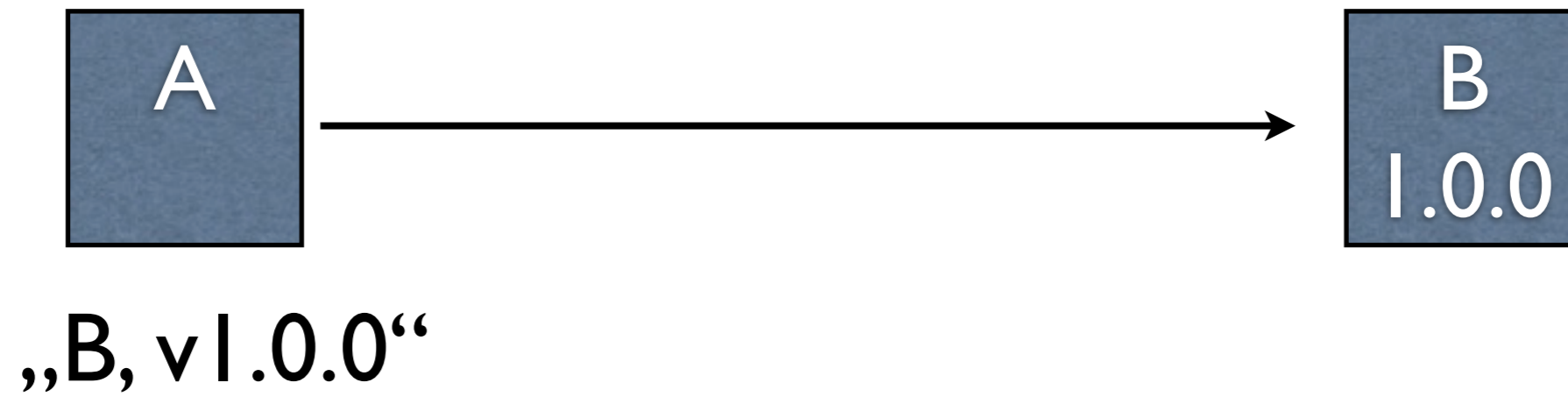
- ▶ **Taggen bei jedem Build**
- ▶ **Svn url und Revision in Metadaten**

# Demo

# Buildnumber & SVN

# Abhängigkeiten

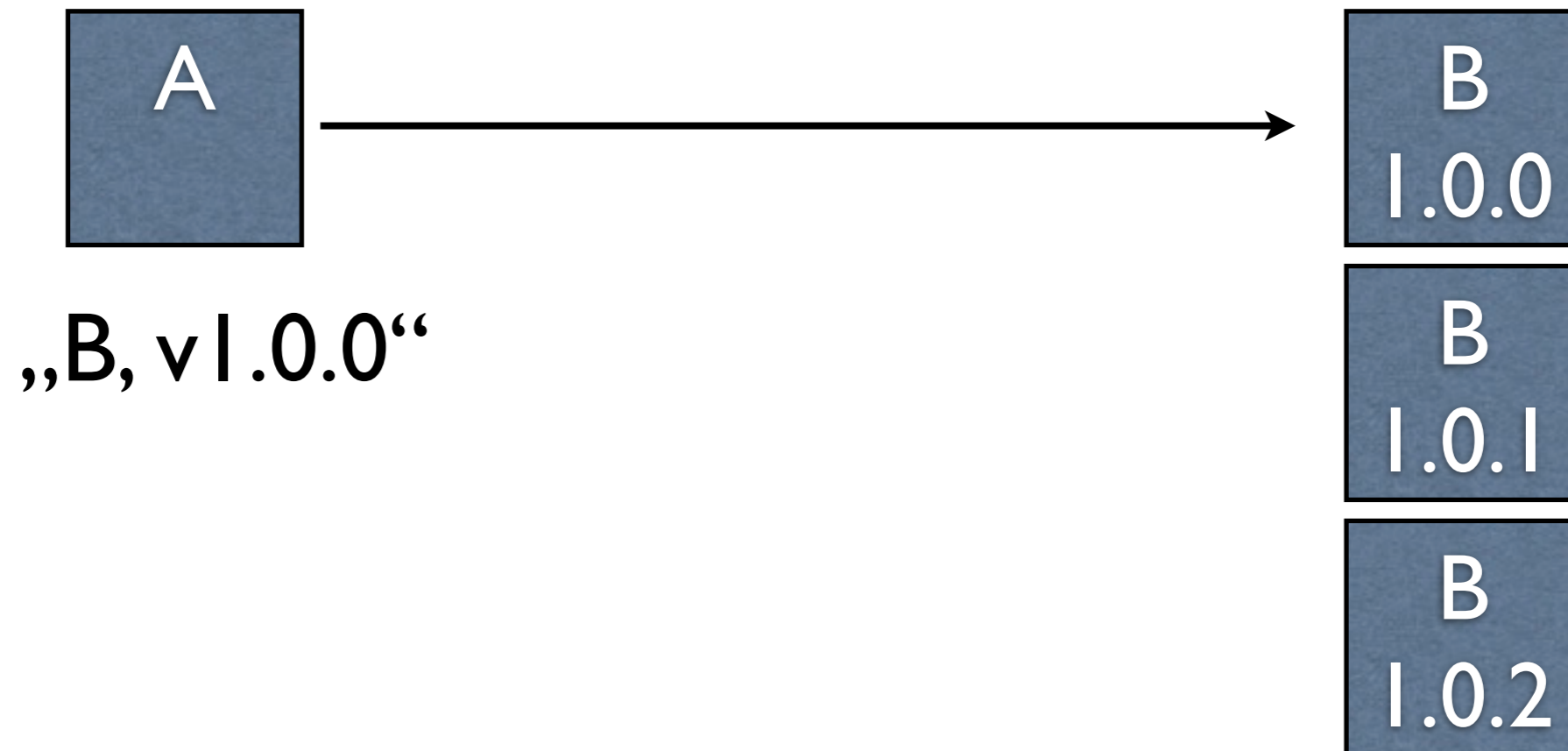
# Statische Abhängigkeit



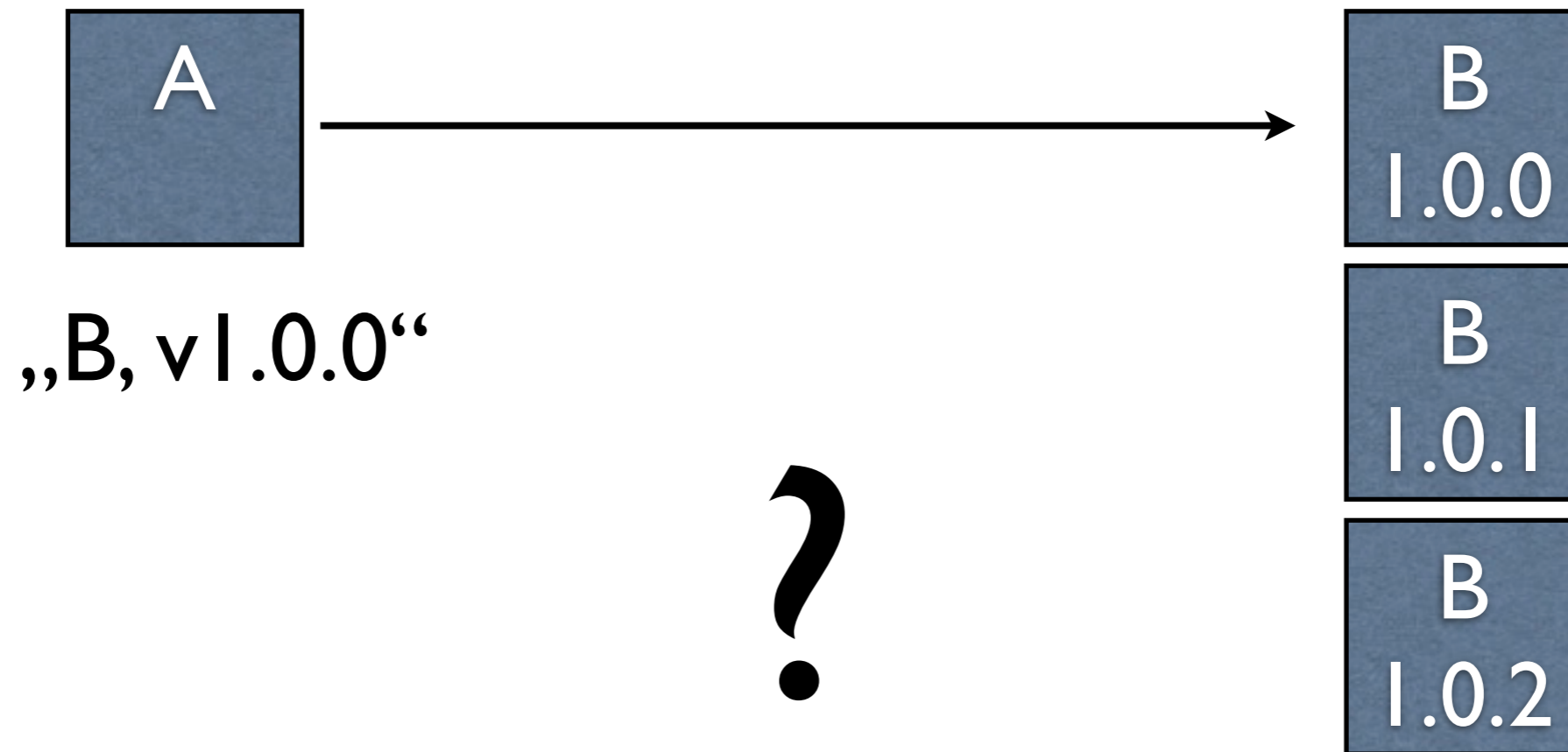
# Statische Abhängigkeit



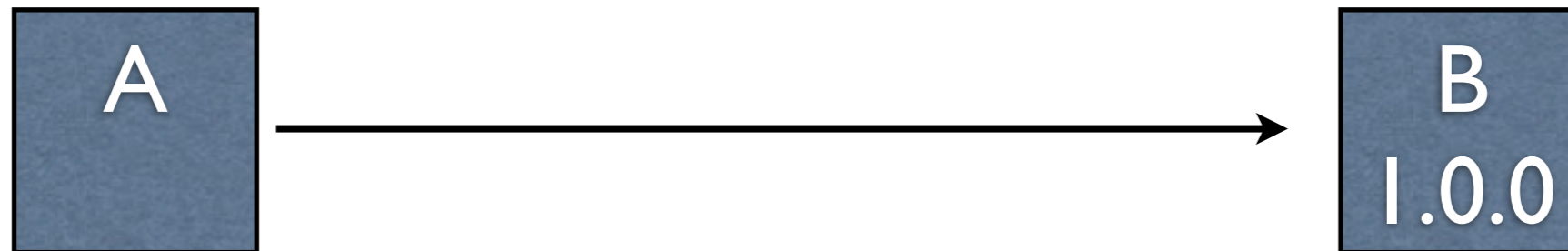
# Statische Abhängigkeit



# Statische Abhängigkeit



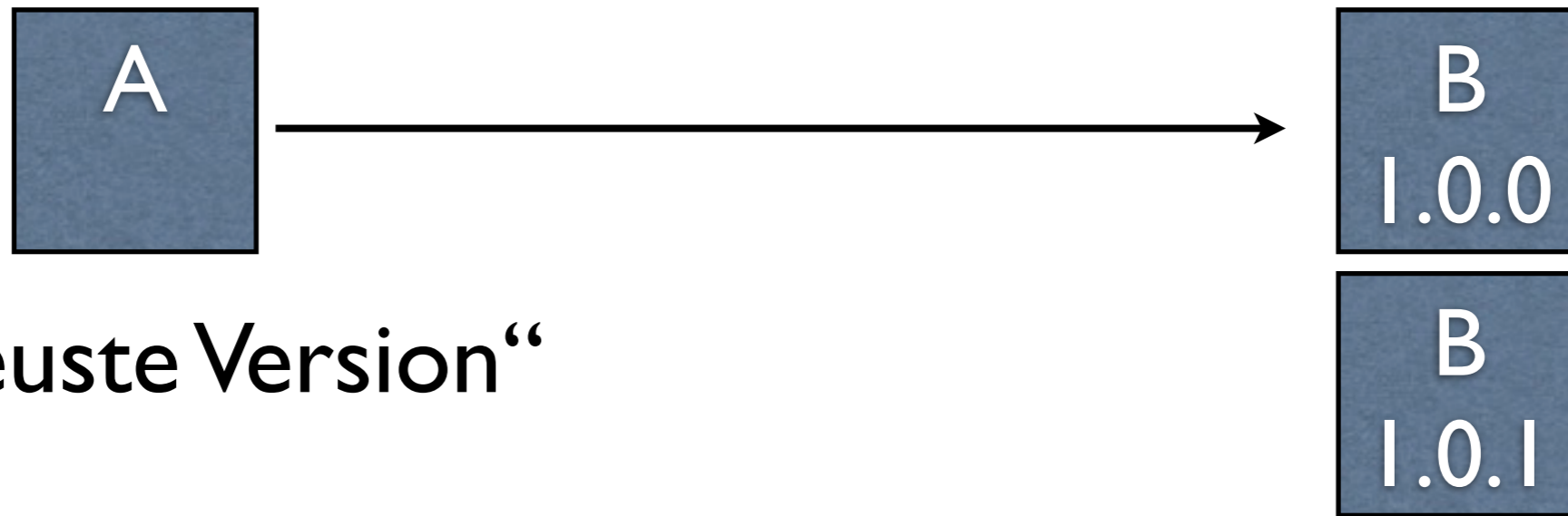
# Dynamische Abhängigkeit



„B, neuste Version“

Keine Eindeutige Zuordnung!

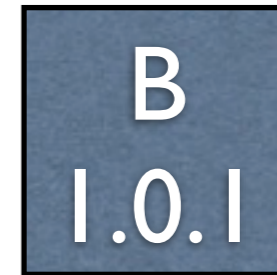
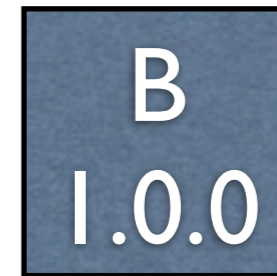
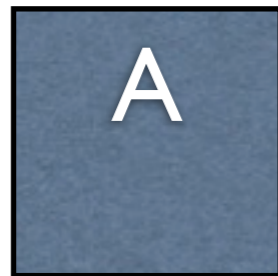
# Dynamische Abhängigkeit



„B, neuste Version“

Keine Eindeutige Zuordnung!

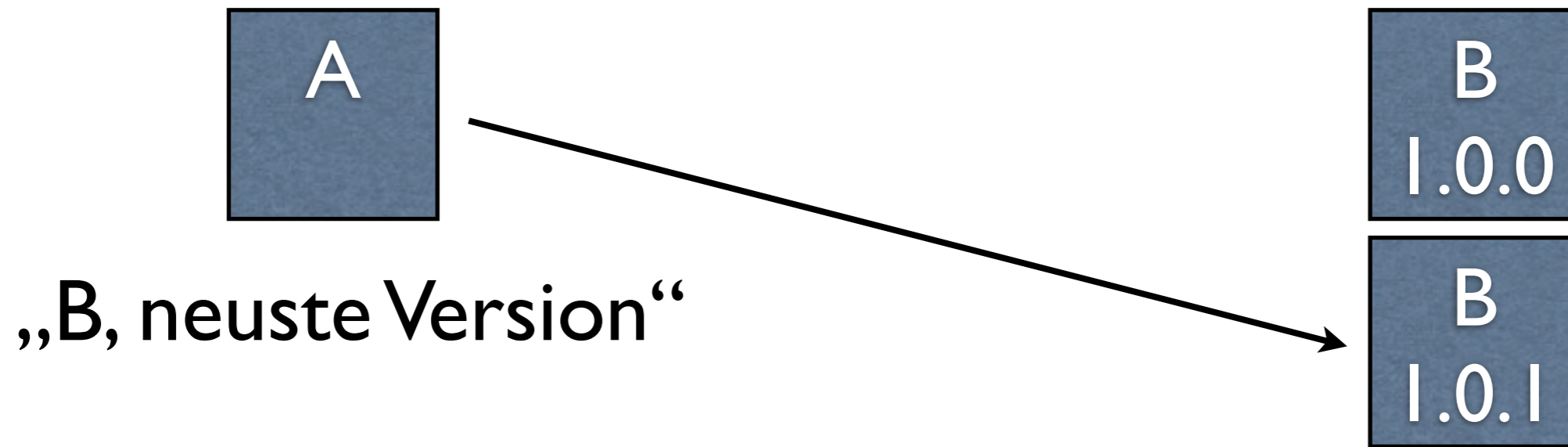
# Dynamische Abhängigkeit



„B, neuste Version“

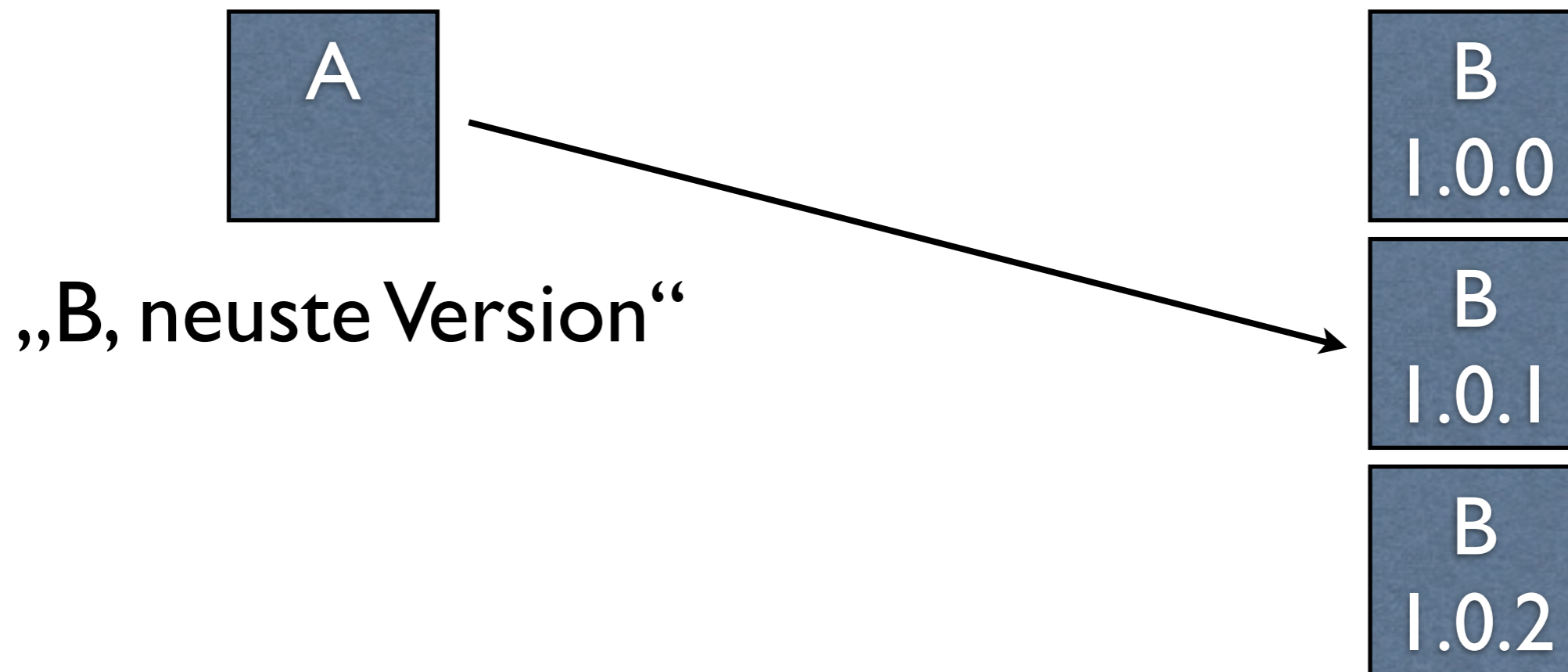
Keine Eindeutige Zuordnung!

# Dynamische Abhängigkeit



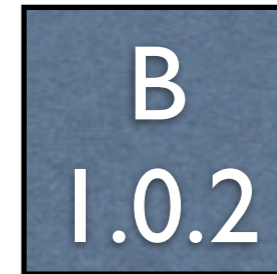
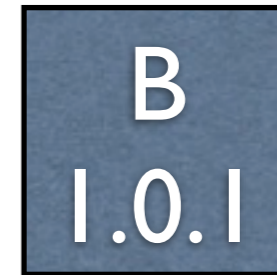
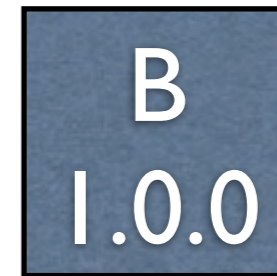
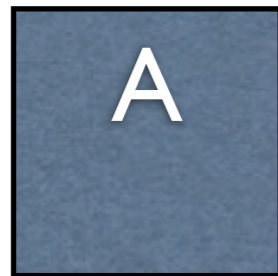
Keine Eindeutige Zuordnung!

# Dynamische Abhängigkeit



Keine Eindeutige Zuordnung!

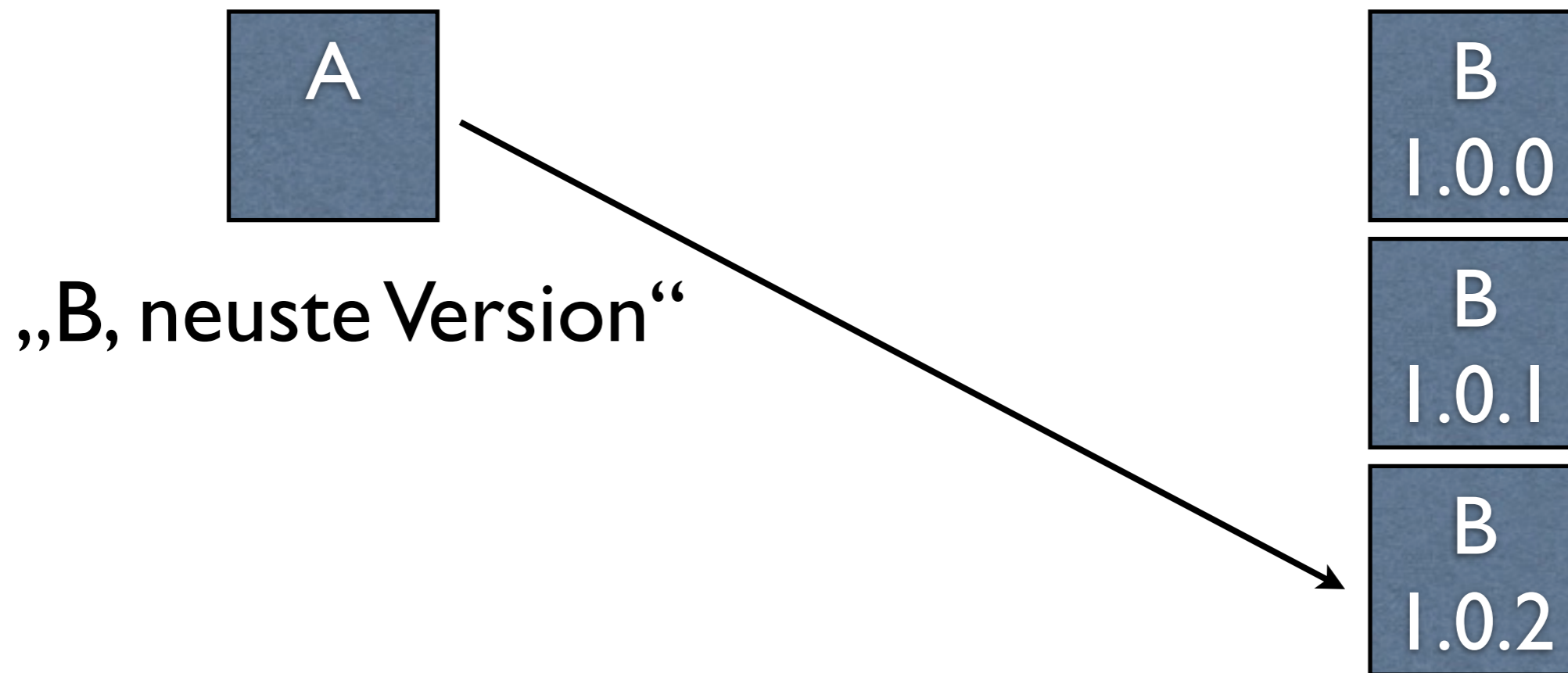
# Dynamische Abhängigkeit



„B, neuste Version“

Keine Eindeutige Zuordnung!

# Dynamische Abhängigkeit



Keine Eindeutige Zuordnung!

# Abhängigkeiten

## Problem beim Referenzieren von Abhängigkeiten

- ▶ „Module B, Version 1.0.1“ ist zu statisch

# Abhängigkeiten

## Problem beim Referenzieren von Abhängigkeiten

- ▶ „Module B, Version 1.0.1“ ist zu statisch
- ▶ „Module B, neuste Version“ ist nicht eindeutig

# Abhängigkeiten

## Problem beim Referenzieren von Abhängigkeiten

- ▶ „Module B, Version 1.0.1“ ist zu statisch
- ▶ „Module B, neuste Version“ ist nicht eindeutig

**Maven unterscheidet deshalb Entwicklungs- und Release Builds.**

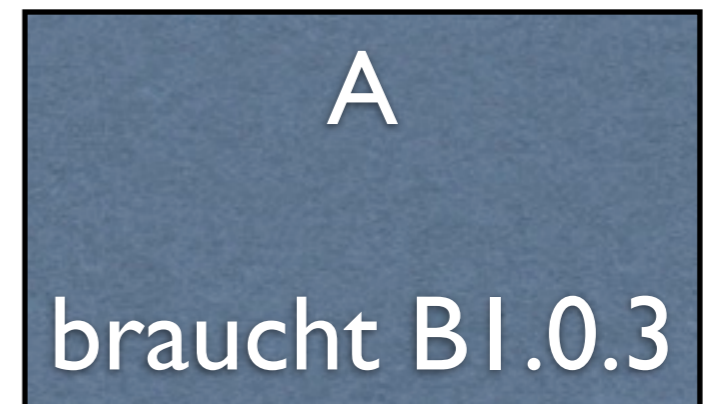
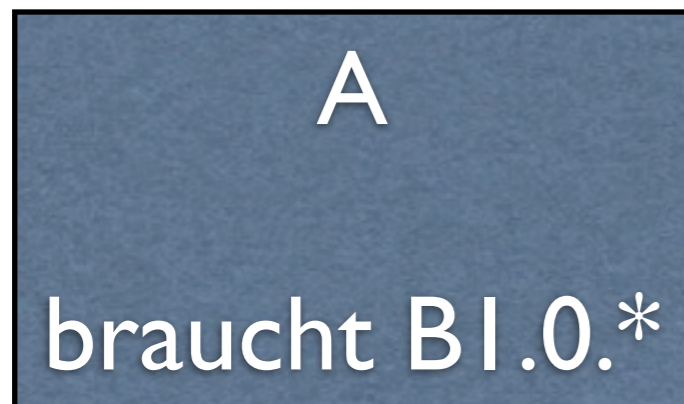
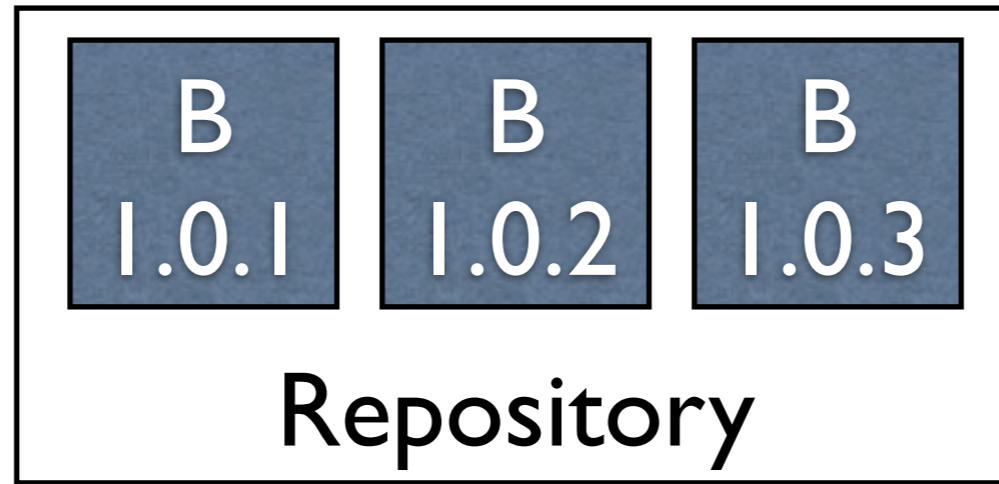
# Abhängigkeiten

## Problem beim Referenzieren von Abhängigkeiten

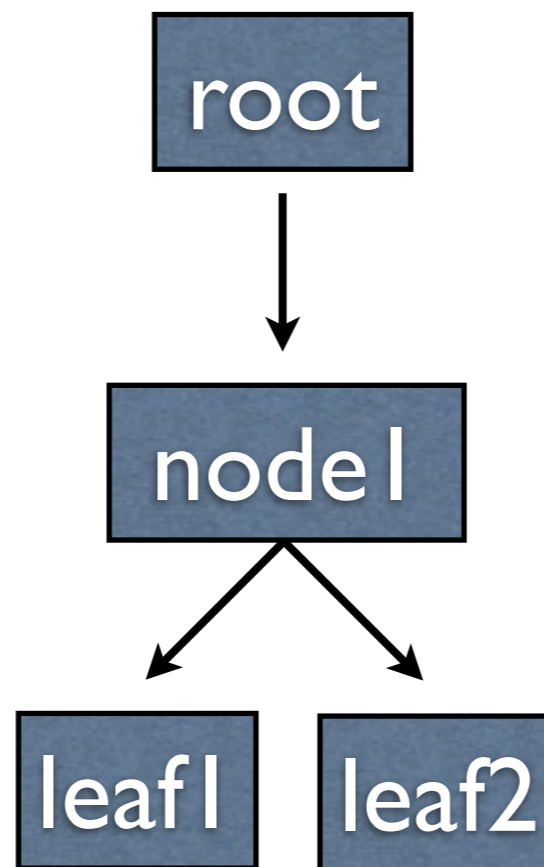
- ▶ „Module B, Version 1.0.1“ ist zu statisch
- ▶ „Module B, neuste Version“ ist nicht eindeutig

Maven unterscheidet deshalb Entwicklungs- und Release Builds.

Eleganter: Dynamische Version + ersetzen zur Buildzeit



# Demo



# Modularisierung

**Wir modularisieren Software  
... wir sollte sie auch so bauen**

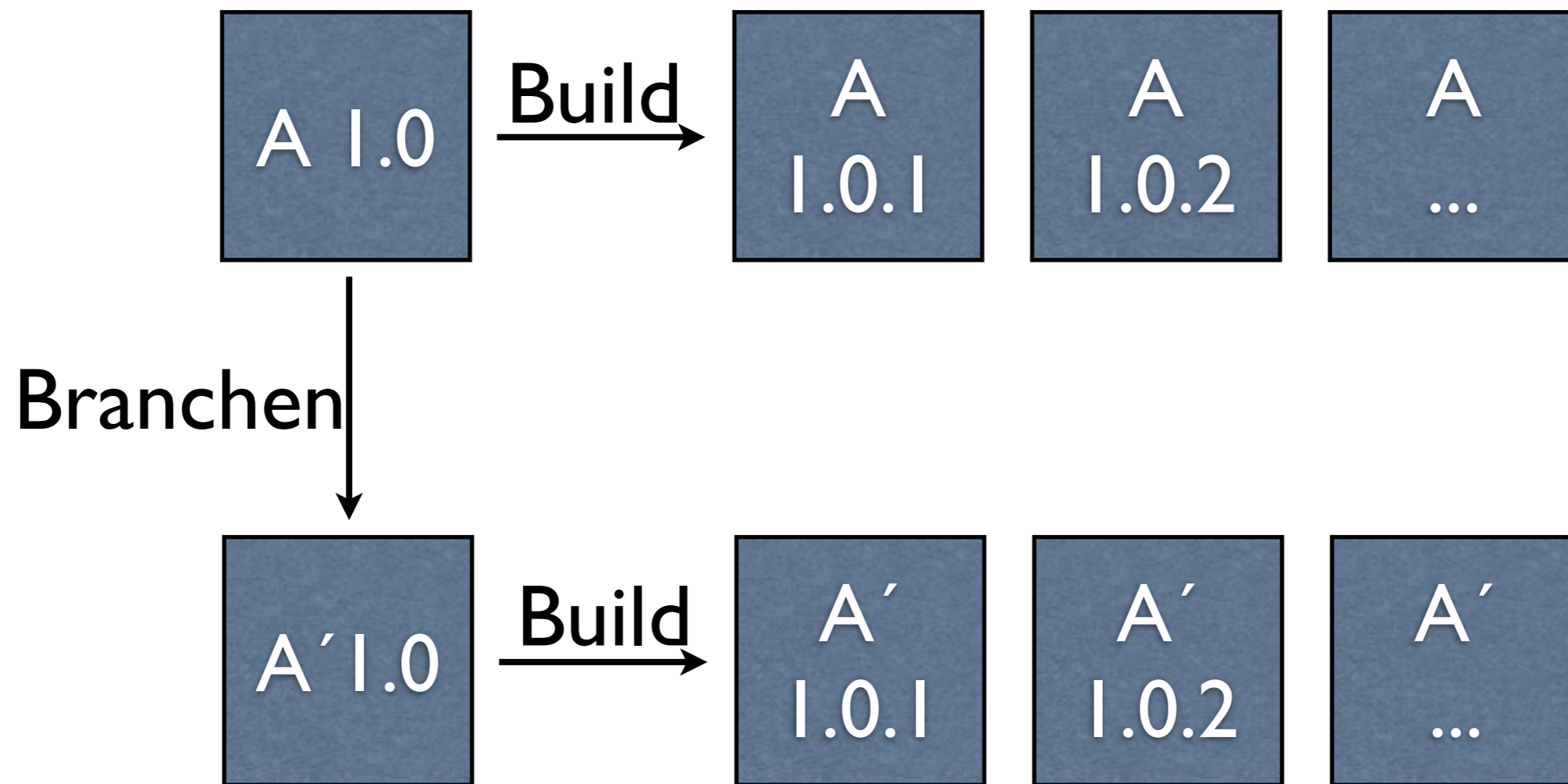
# Demos

# Branchen

- ▶ **Kopie einer Quelle, Paralleles entstehen neuer Artefakte**
- ▶ **Der Name muss das abbilden**

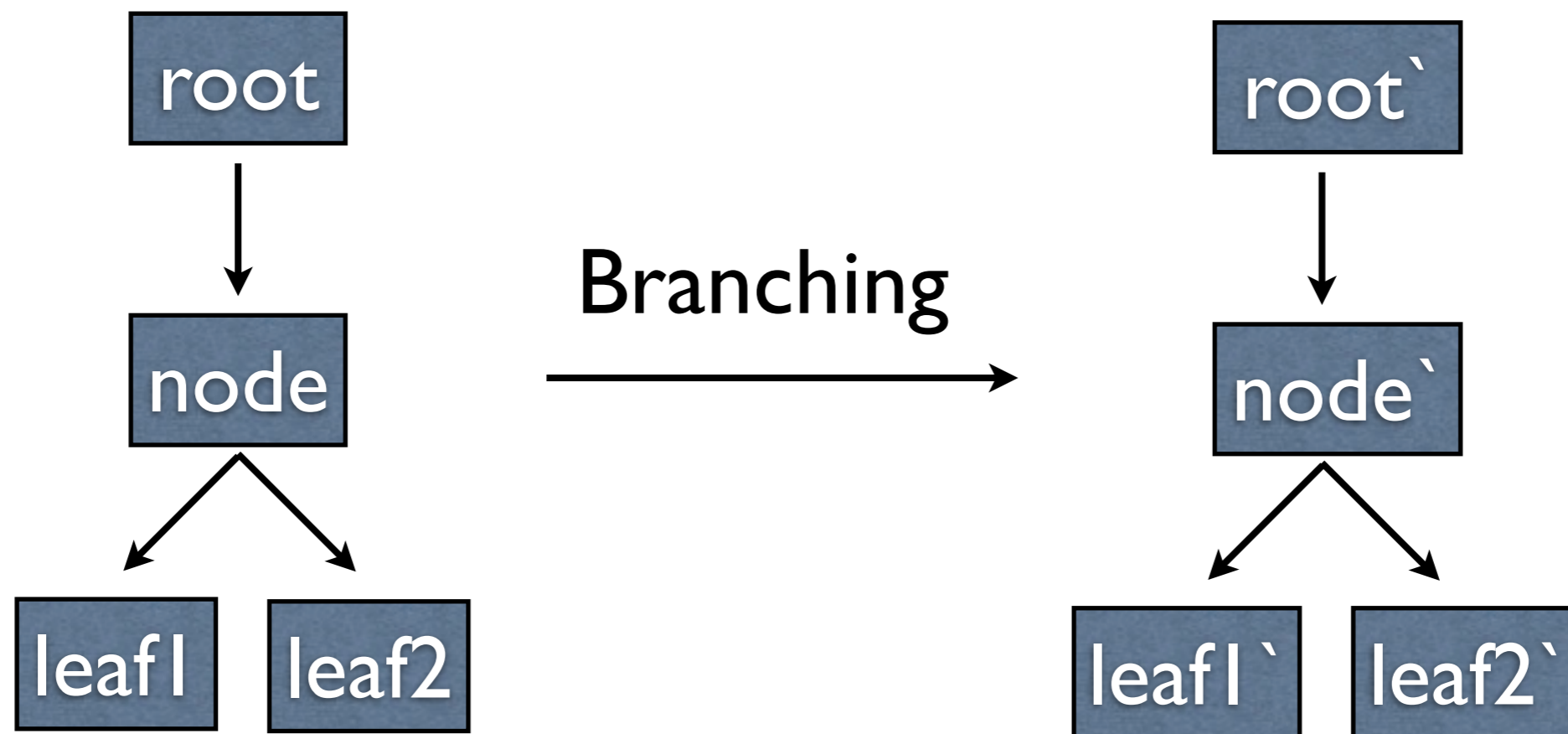
# Branchen

## Umbenennung des Moduls



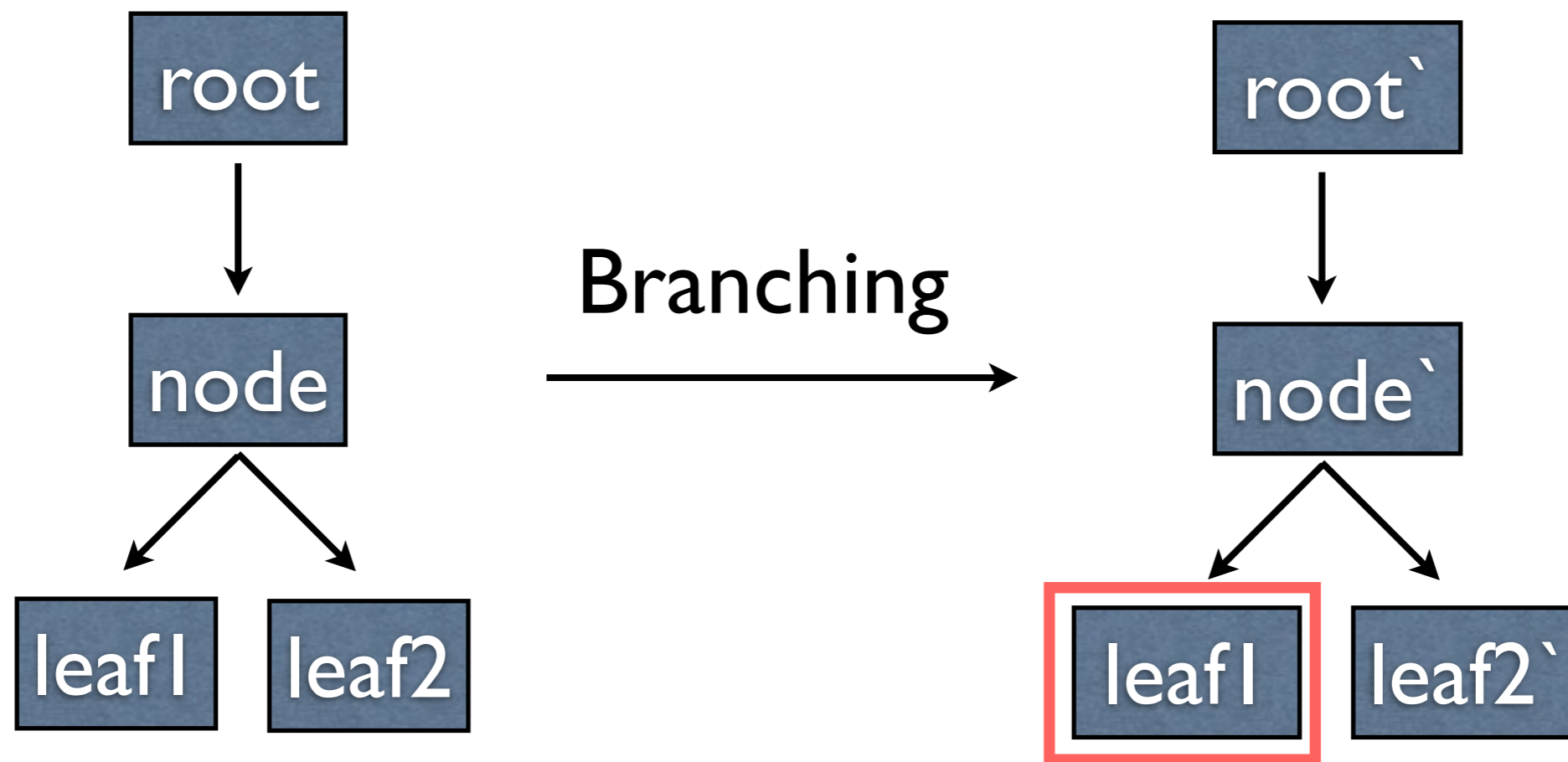
# Branchen

Erhalten von Abhängigkeiten



# Partielles Branchen

Bugfix leaf2 in von Abhängigkeiten



# Wishlist

**Das Tooling sollte unterstützen:**

- Update der Metainformationen für totales und partielles Branchen**
- Anlegen der Branches im SCM**
- Anlegen der Projekte im CI Server**

# Build Promotion

- ▶ **Viele Artefakte durch CI**
- ▶ **Kopieren relevanter Artefakte – „promotion“**
- ▶ **Promotion ist ein guter Zeitpunkt für's taggen**
- ▶ **Regelmäßiges löschen unwichtiger Artefakte**

# Status

Leider geht nicht alles hier gezeigt „out of the Box“

Eindeutige Versionsnummer	ivy:buildnumber
svn Url + Rev. in Metadaten	svn info mit svnAnt, regex Suchen + Ersetzen mit Ant
dynamische Revision ersetzen	ivy:publish
Upstream/Downstream Build in ant	ivy:buildlist + subant
Modularisiert Bauen in Hudson	hudson ivy plugin

# Status

<p>Branchen – Umbenennen von Abhängigkeiten</p>	<p>ivy:buildlist, eigener ivy/ant task zum Update der Attribute</p>
<p>Branchen – Svn Erstellung partieller B.</p>	<p>ivy:buildlist, ivy:retrieve, eigener ivy/ant task, svnAnt</p>
<p>Build Promotion</p>	<p>ivy:install, keine Hudson integration</p>
<p>Löschen in Repository</p>	<p>manuell, skripte..</p>

# Fazit

- ▶ **Denken!**
- ▶ **CI Build = Release**
- ▶ **Modularisieren**
- ▶ **Branchen ist mehr als svn copy...**

# Fragen?

**Vielen Dank!**  
**[Martin.Eigenbrodt@innoq.com](mailto:Martin.Eigenbrodt@innoq.com)**