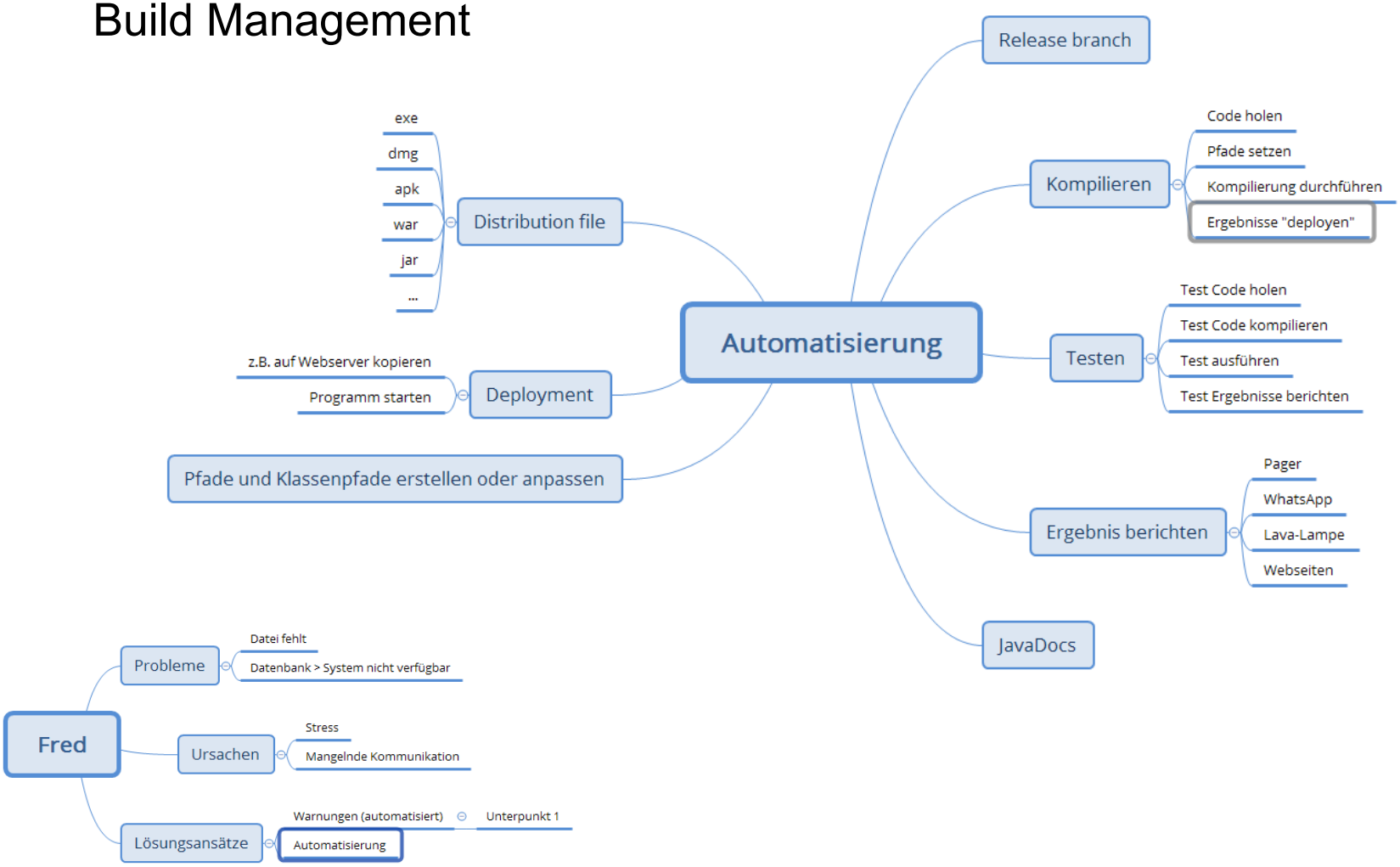
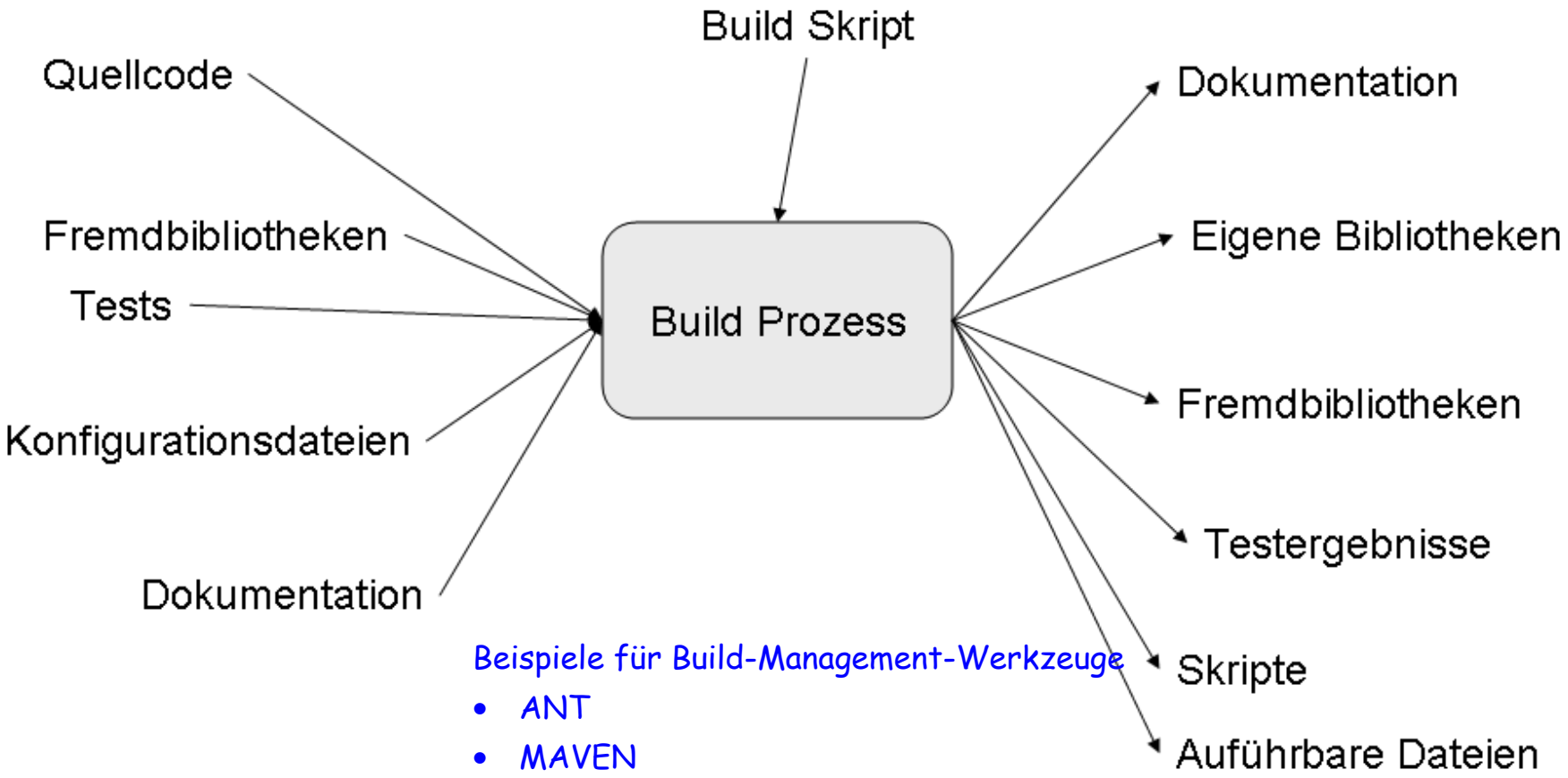
Build Management





Beispiele für Build-Management-Werkzeuge

- ANT
- MAVEN
- Gradle
- Grunt
- SBT
- C: make

Build-Management mit ANT

Lernziel

- wissen, für was ANT gut ist und was es macht
- wissen, wie man Variablen in ANT definiert und verwendt
- wissen, was ein Target ist
- wissen, wie man die Reihenfolge von Targets bestimmt
- ANT-Skript lesen, verstehen und ändern können

Calculator.java | CalculatorTest.java | build.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- WARNING: Eclipse auto-generated file.
    <property environment="env"/>
    <property name="ECLIPSE_HOME" value="C:/Tools/Eclipse/eclipse/"/>
    <property name="junit.output.dir" value="junit"/>
    <property name="debuglevel" value="source,lines,vars"/>
    <property name="target" value="1.8"/>
    <property name="source" value="1.8"/>
    <path id="JUnit 4.libraryclasspath">
    <path id="SOTE2-2018-06-01.classpath">
    <target name="init">
    <target name="clean">
    <target depends="clean" name="cleanall"/>
    <target depends="build-subprojects,build-project" name="build"/>
    <target name="build-subprojects"/>
    <target depends="init" name="build-project">
        <echo message="${ant.project.name}: ${ant.file}"/>
        <javac debug="true" debuglevel="${debuglevel}" destdir="bin" includeantruntime="false" source="${source}" target="${target}">
            <src path="src"/>
            <src path="test"/>
            <classpath refid="SOTE2-2018-06-01.classpath"/>
        </javac>
    </target>
    <target description="Build all projects which reference this project. Useful to propagate changes." name="build-refprojects"/>
    <target description="copy Eclipse compiler jars to ant lib directory" name="init-eclipse-compiler">
    <target description="compile project with Eclipse compiler" name="build-eclipse-compiler">
    <target name="CalculatorTest (1)">
    <target name="junitreport">
</project>
```

Definition von Variablen ("property"). z.B. hat die Variable "target" den Wert "1.8"

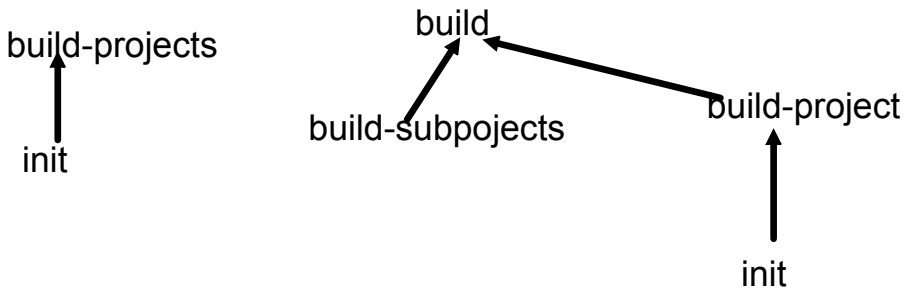
Verwendung von Variablen: Mit ${target} wird der Wert der Variablen target referenziert. D.h. der Wert wird mit 1.8 ersetzt

Target sind Aufgabenpakete, das mehrere Aufgaben ("Tasks") enthalten

```
    <path id= SOTE2-2018-06-01.classpath >
    <target name="init">
    <target name="clean">
    <target depends="clean" name="cleanall"/>
    <target depends="build-subprojects,build-project" name="build"/>
    <target name="build-subprojects"/>
    <target depends="init" name="build-project">
    <target description="Build all projects which reference this project. Useful to propagate changes." name="build-refprojects"/>
    <target description="copy Eclipse compiler jars to ant lib directory" name="init-eclipse-compiler">
    <target description="compile project with Eclipse compiler" name="build-eclipse-compiler">
    <target name="CalculatorTest (1)">
    <target name="junitreport">
</project>
```

Ein Target wird erst ausgeführt, wenn die Target, die unter "depends" genannt sind, ausgeführt wurden.

Wenn das Target "build-projeckt aufgerufen wird, ist die Aufrufsequenz die folgende:



```
<target name="init">
    <mkdir dir="bin"/>
    <copy includeemptydirs="false" todir="bin">
        <fileset dir="src">
            <exclude name="**/*.launch"/>
            <exclude name="**/*.java"/>
        </fileset>
    </copy>
    <copy includeemptydirs="false" todir="bin">
        <fileset dir="test">
    </copy>
</target>
```

Target beinhalten Aufgaben (Tasks).

z.B. beinhaltete das Target "init" drei Tasks:

- mkdir
- copy
- copy

```
<target depends="init" name="build-project">
    <echo message="${ant.project.name}: ${ant.file}"/>
    <javac debug="true" debuglevel="${debuglevel}" destdir="bin" includeantruntime="false" source="${source}" target="${target}">
        <src path="src"/>
        <src path="test"/>
        <classpath refid="SOTE2-2018-06-01.classpath"/>
    </javac>
</target>
```

ANT bietet eine Vielzahl vordefinierter Tasks an

- Verzeichnis anlegen mkdir
- Verzeichnis löschen deldir
- Kopieren
- Kompilieren
- Komprimieren
- Tests durchführen
- Testergebnisse als Webseite aufbereitet
- Testergebnisse kommunizieren....

Zudem gibt es die Möglichkeit, eigene Tasks zu schreiben (s. nächsten Stunde).

Vorteile eine automatisierten Build-Managements

- Zeitsparend
- Schneller
- Fehlerfreier
- Häufiger durchgeführt => Fehler früher erkennen => Fehler früher und einfach reparieren
- ...