Buildsysteme im Architekturüberblick

Basierend auf der Workshopreihe “Buildsysteme”

Thomas Wieger, Marco Klemm,
Darco Palic, Michael Jerger

Was erwartet uns?

Was bietet Ant?

Warum Maven?

Was könnten wir besser machen?

Erwartungshaltung an Buildsysteme

Zentrale Forderung CRISP

- **Complete**

Erwartungshaltung an Buildsysteme

Zentrale Forderung CRISP

- **Complete**
- **Repeatable**

Erwartungshaltung an Buildsysteme

Zentrale Forderung CRISP

- **Complete**
- **Repeatable**
- **Informative**

Erwartungshaltung an Buildsysteme

Zentrale Forderung CRISP

- **Complete**
- **Repeatable**
- **Informative**
- **Schedulable**

Erwartungshaltung an Buildsysteme

Zentrale Forderung CRISP

- **Complete**
- **Repeatable**
- **Informative**
- **Schedulable**
- **Portable**

Erwartungshaltung an Buildsysteme

Darüber hinaus:

- **Multiprojektfähigkeit**

Erwartungshaltung an Buildsysteme

Darüber hinaus:

- **Multiprojektfähigkeit**
- **IDE-Integration nach DRY-Prinzip**

Erwartungshaltung an Buildsysteme

Darüber hinaus:

- **Multiprojektfähigkeit**
- **IDE-Integration nach DRY-Prinzip**
- **performant**

Erwartungshaltung an Buildsysteme

Darüber hinaus:

- **Multiprojektfähigkeit**
- **IDE-Integration nach DRY-Prinzip**
- **performant**
- **Verwaltung von 3rd Party-Libs**

Erwartungshaltung an Buildsysteme

Darüber hinaus:

- **Multiprojektfähigkeit**
- **IDE-Integration nach DRY-Prinzip**
- **performant**
- **Verwaltung von 3rd Party-Libs**
- **Reifegrad**

Erwartungshaltung an Buildsysteme

Darüber hinaus:

- **Multiprojektfähigkeit**
- **IDE-Integration nach DRY-Prinzip**
- **performant**
- **Verwaltung von 3rd Party-Libs**
- **Reifegrad**
- **einfach und angemessen**

Erwartungshaltung an Buildsysteme

Darüber hinaus:

- Multiprojektfähigkeit
- IDE-Integration nach DRY-Prinzip
- performant
- Verwaltung von 3rd Party-Libs
- Reifegrad
- einfach und angemessen
- Convention over Configuration

Erwartungshaltung an Buildsysteme

Darüber hinaus:

- Multiprojektfähigkeit
- IDE-Integration nach DRY-Prinzip
- performant
- Verwaltung von 3rd Party-Libs
- Reifegrad
- einfach und angemessen
- Convention over Configuration
- leicht anpassbar

Erstes Beispiel: Ant

Ant ist eine umfassende Sammlung von Build-Werkzeugen, d. h.

- **alles Wichtige vorhanden**
- **deklarative Beschreibung der Buildreihenfolge**
- **Dependecies nur zwischen Targets**

Einordnung von ANT

Erfahrungen mit ANT

- **Manchmal ist Imperative Verarbeitung wünschenswert (Reihenfolgen, Schleifen ...)**
- **Builddefinition in XML ist gesprächig**
- **Buildsprache hat deutliche Lücken (Vererbung, Modularisierung, Verständlichkeit)**

Zweites Beispiel: Maven

Maven ist ein vollständiges Buildsystem

- Multiprojekt Support
- Ein 3rd Party Bibliothekskonzept
- Mut zu meist sinnvollen Konventionen
- Integriertes Reporting
- Scopes für Test, Compile und Packaging

Einordnung von Maven

Erfahrungen mit Maven

- Arbeiten ohne Netzzugang ist fast unmöglich
- Transitivität im Compile-Scope verursacht unerwartete Abhängigkeiten
- Die Einschränkung auf ein Artefact pro Projekt
- Multiprojekt Build für viele Maven Module fehlerhaft
- Projekt- und 3rd Party Lib-Abhängigkeiten werden nicht unterschieden
- Unterschiedlicher Reifegrad der Maven Module

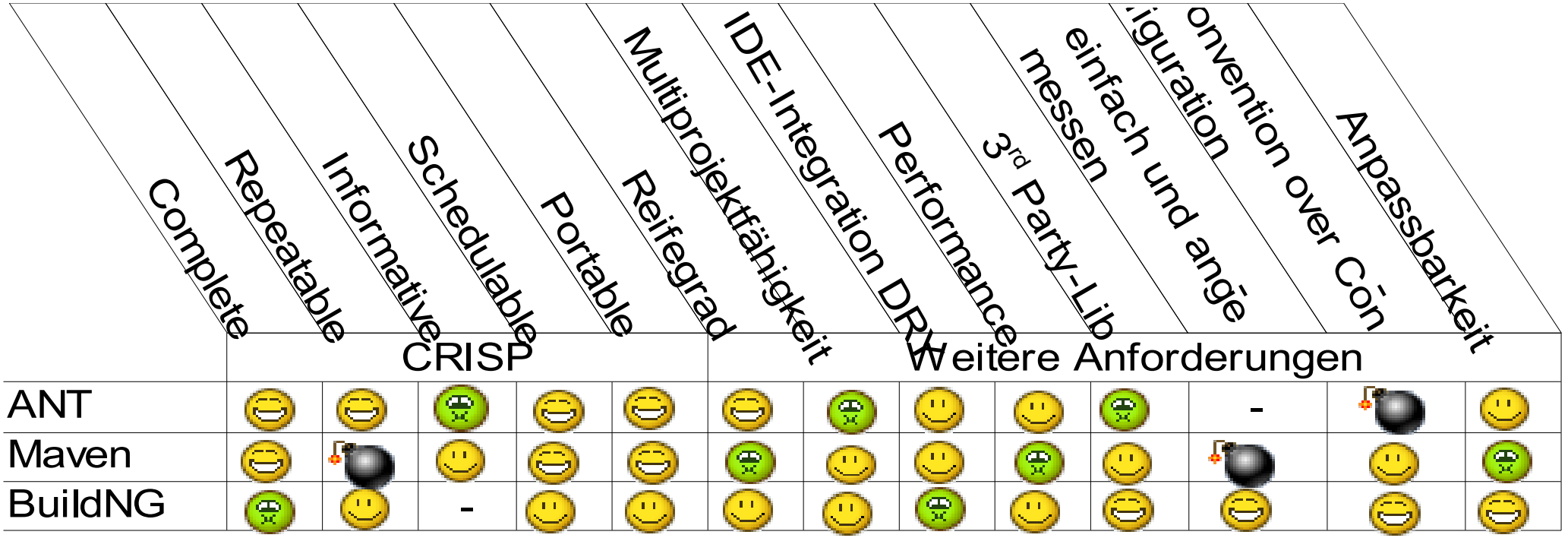
Fragen, die offen bleiben

- **Warum XML und weitere ConfigFiles als Buildsprache?**
- **Gibt es einen angemessenen Mittelweg in der Bibliotheksverwaltung?**
-

Buildng – was ist drin

- kein XML – jeder build ist ein Java-Programm
- nutzt ANT – bietet ein fluent interface für Ant in Java an (elegAnt)
- unterscheidet Abhängigkeiten zu anderen Projekten und zu Bibliotheken (jars)
- keine Versionen von Bibliotheken
- Bibliotheken liegen im Dateisystem, nicht in irgendeinem Repository. Keine Vorgaben zur Verzeichnisstruktur, keine Verwaltung von Versionen
- Compile-Abhängigkeiten sind nicht-transitiv, Test/Runtime-Abhängigkeiten sind transitiv
- sinnvolle Konventionen (a la Maven)

Einordnung BuildNG



Buildng – was fehlt noch

- Infrastruktur für Berichte
- builder für ejb-module, ear
- Ide Integration