# 54. Werkzeuge für das Bau-Management



Prof. Dr. rer. nat. Uwe Aßmann
Institut für Software- und Multimediatechnik
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
TU Dresden
http://st.inf.tu-dresden.de
Version 12-0.1, 31.01.13

1) Baumanagement
2) Das Baumanagement-System von GNU
3) Aufgabenmanagement (Fehler, Anforderungen)

In 2012/13 weggelassen
nur zur Info



# 54.1 Bau-Management-Werkzeuge

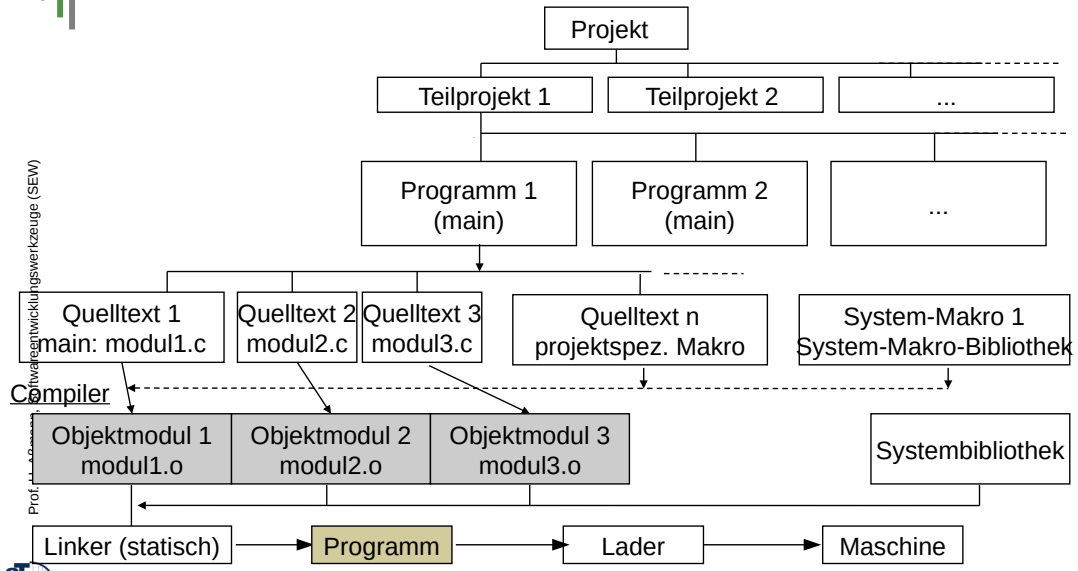


... Kommando:
info automake



# Aufbau realer Projekte mit statischem Linken (Technikraum C/C++)








# Make



- *Make* ist ein Bauprogramm, das *abhängige Zieldateien (targets)* aus *Quelldateien (sources, Originaldateien)* durch *Ausführen eines Kommandos* erzeugt.
  - Stuart Feldman, 1977, als Teil von UNIX
  - http://de.wikipedia.org/wiki/Make
- Abhängigkeiten werden in **Bau-Spezifikationen (makefiles)** spezifiziert mit Hilfe von *Abhängigkeiten* und *Aktionen*:
  - `B.o: B.c C.h D.h` // A ist abhängig von B, C und D
    - `gcc -c B.c` // Aktion, um B.o aus B.c zu generieren
- Generische Regel:
  - Falls (Zieldatei älter als eine der Quelldateien)
  - führe die Aktion aus, d.h. regeneriere
- Im Wesentlichen ist Make ein Regelinterpreter mit lazy-Evaluation für die Regeln (träge Auswertung)
  - lazy funktionale Sprache bieten die Funktionalität umsonst an

## Makefile für Generierung von pdfs aus tex



```
LATEX=latex
MAIN=document.tex
SOURCES=$(MAIN) figure.tex
PDFLATEX=pdflatex
G=gen
tex_it:       $(SOURCES)
        $(LATEX) $(MAIN) && cp $(MAIN).dvi $(MAIN).dvi~
        dvips -Ppdf -o $G/$(MAIN).ps $(MAIN)
        ps2pdf $G/$(MAIN).ps
```



## Generische Regeln



```
$G/%.ps: %.pdf
        pdftops $< $@
        psnup -2 $@ > ${@:.ps=}-2p.ps
        psselect -r ${@:.ps=}-2p.ps | psnup -2 |psselect -r > ${@:.ps=}-4p.ps
```



## Operatoren (wildcard, basename...)



```
BIBS=$(wildcard *.bib)

MAIN=$(basename $(shell grep -l documentclass *.tex))

# #### Replacement Operatoren: #######
BIBHTMLS=${BIBS:%.bib=$G/%-bib.html}
```



## Schleifen



```
architectures:
        for j in $(ARCHITECTURES) ; do mkdir $$j ; done;
links:
        for j in $(ARCHITECTURES) ; do cd $$j; for i in $(SOURCES) ; do ln -s ../$$i
. ; done; cd ..; done;
cleanlinks:
       for j in $(ARCHITECTURES) ; do cd $$j; for i in $(SOURCES) ; do mv $$i $
(HOME)/backup; done; cd ..; done;
       for j in $(ARCHITECTURES) ; do cd $$j; rm -f *.c *.cc *.C; cd ..; done;
```



## Ant – Bauwerkzeug für Java



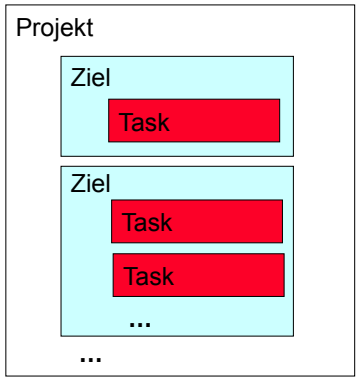
Zur Build-Beschreibung zusammenhängender Komponenten werden anstatt *Make*-Files durch *Ant* Beschreibungen für *Tasks* eingesetzt, die ein bestimmtes Ziel realisieren.

*Ant* (*Another neat tool*) ist ein **Framework** zur Automatisierung des Build-Prozesses aus Komponenten.

Ein ***Ant*-Buildfile** build.xml enthält
- genau ein **Projekt**,
- ein bis mehrere **Ziele (Targets)**, die Aspekte des Projektes (z.B. Dokumentation erstellen, Release- oder Debug-Version erstellen…) beschreiben sowie
- **Tasks** meist in Java zur Umsetzung der Ziele.

Es unterstützt Wildcards und Filesets.

*Ant* stellt „nur“ das Framework zur Organisation und Verwaltung der Projekte und Ziele dar, die weitergehende Logik (z.B. inkrementelles Compilieren) steht in der Verantwortung der Tasks.




**Quelle:** [ 7, S. 142 ff. ]



## *Ant* (2)



Es werden zahlreiche vorgefertigte Tasks mitgeliefert, z.B. für
- Übersetzen (`javac`, `JspC`)
- Archivieren (`jar`, `zip`, `rpm`)
- Dokumentation (`Javadoc`)
- Dateipf ege (`Checksum`, `Copy`, `Delete`, `Move`…)
- Test (`Junit`)

ANT kann mit selbst geschriebenen Tasks (Java-Klassen) erweitert werden.

Aufruf: **`ant [options] [target]`**

`target` bestimmt dabei, welches im Buildf le aufgeführte Ziel realisiert werden soll, falls der Parameter weggelassen wird, wird das default-target verwendet. Beispielaufrufe:

| | |
|---|---|
| **`ant compile`** | Ziel „compile“ umsetzen |
| **`ant clean compile`** | Ziele „clean“ und „compile“ umsetzen |
| **`ant -buildfile test.xml`** | Buildf le test.xml verwenden, default-Ziel |

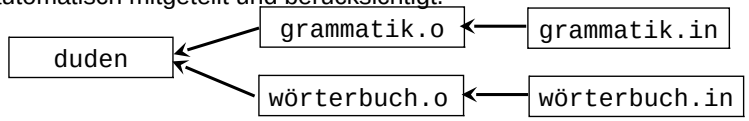
Buildf les werden im XML-Format erstellt, damit einfache Weiterverarbeitung durch Werkzeuge möglich.



## Ant XML Tasks (3)



Ableitungsbeziehungen der **Komponenten** untereinander bilden einen *Abhängigkeits-graphen.* Pfeil A <-- B bedeutet „A ist abgeleitet aus B“. Abhängigkeiten werden Task **`Javac`** automatisch mitgeteilt und berücksichtigt.




```
<project name=„duden" default=„compile" basedir="./..">     Projekt
    <target name="init">                                    Ziel
      <mkdir dir="${output}"/>                              Task
      <mkdir dir="${report}"/>                              Task
    </target>

    <target name="compile" depends="init">                  Ziel
      <javac srcdir="${src}/duden"  destdir="${output}">    Task
      </javac>
     </target>
    …
</project>
```

**Quelle:** [ 7, S. 127 ff. ]



## 54.2 Das Baumanagementsystem von GNU

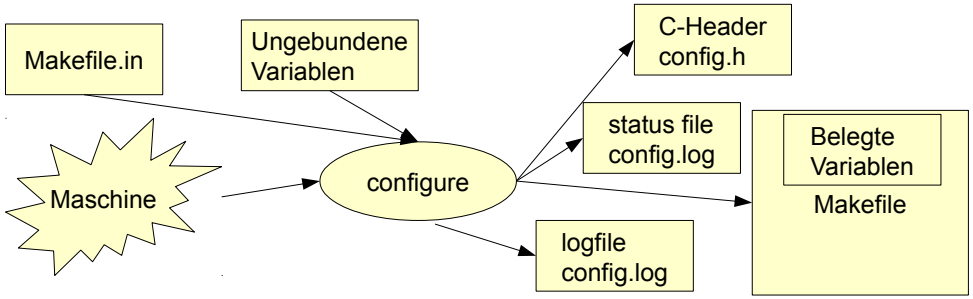


... Kommando:
info automake

## configure



- http://www.gnu.org/software/autoconf/
- configure ist ein Shellskript, ein Generator für Bau-Spezifikationen (Makefiles)
  - untersucht die Maschine, ob bestimmte Bibliotheken vorhanden sind
    - in welcher Variante sie vorhanden sind
  - Erzeugt parametrisierte Aufrufe an C-Übersetzer und andere Werkzeuge
    - durch das Setzen von Variablen (compilation switches), die Plattformabhängigkeiten ausdrücken (für Makefiles *und* C/C++-Programme
- configure sorgt für Portabilität der GNU-Programme






## Installation mit configure und make



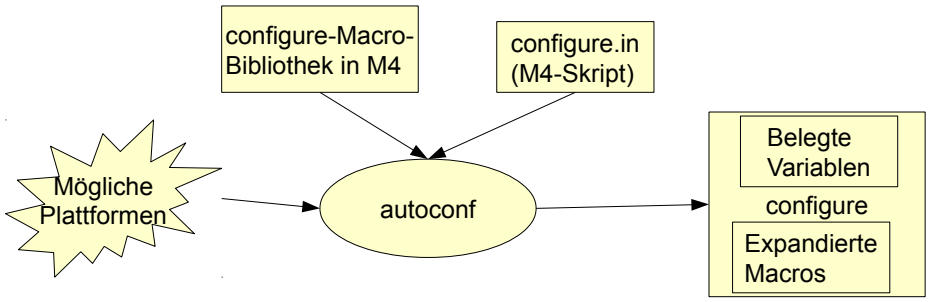
- unzip <package>
- ./configure
  - // Untersucht Maschine. Hinterlässt Makefile, config.log, config.status, config.h
- ./make
  - // baut das System
- ./make check
  - // Prüft, ob alles gutging
- ./make install
  - // installiert das Programm vom lokalen Dir in ein globales
- ./make uninstall
  - // deinstalliert das Programm



## autoconf



- http://www.gnu.org/software/autoconf/
- Autoconf generiert configure-Skripte aus M4-Präprozessorskripten, d.h. M4-Makros (wie Funktionen aufzurufen)
  - die Makros erzeugen Abschnitte des configure-Skriptes
  - Autoconf Makroarchiv: http://ac-archive.sourceforge.net/






## M4-Makros für Java-Entwicklung



- http://ac-archive.sourceforge.net ist ein Projekt, das M4-Makros für die Erzeugung von configure-Skripten sammelt
- Zum Beispiel http://ac-archive.sourceforge.net/ac-archive/ac_try_run_javac.html:
- ac_check_class (ac-archive)  // prüft, ob Klasse vorhanden
- ac_check_junit (ac-archive)  // prüft, ob jUnit vorhanden
- ac_check_rqrd_class (ac-archive) // etc.
- ac_java_options (ac-archive)
- ac_prog_jar (ac-archive)
- ac_prog_java (ac-archive)
- ac_prog_java_cc (ac-archive)
- ac_prog_java_works (ac-archive)
- ac_prog_javac (ac-archive)
- ac_prog_javac_works (ac-archive)
- ac_prog_javadoc (ac-archive)
- ac_prog_javah (ac-archive)
- ac_try_compile_java (ac-archive)
- ac_try_run_javac (ac-archive)

## Autoproject



- http://packages.debian.org/unstable/devel/autoproject
- autoproject initialisiert in einem leeren Directory ein GNU-konformantes Projekt
  - installierbar auf vielen Plattformen (deployable) mit Hilfe des Werkzeuges *configure*
  - varianten-konfigurierbar
  - erzeugt ein Baumanagement-Sytem
    - das mit *autoconf* configure-skripten erzeugen kann
    - *und automake*



# 54.3 Aufgabenmanagement (Fehler- und Änderungsmanagement)





## Aufgabenmanagement



Es beinhaltet die Erfassung, Registratur, Speicherung und Verfolgung von

- Fehlermeldungen
- Änderungsanforderungen

und steuert alle dazu notwendigen Vorgänge:

- **Bewertung** der Änderung: Nach der Notwendigkeit der Auswirkungen, wobei jede Änderung einer Version zu einer neuen Version führt
- **Planung und Entscheid** des Änderungsvorgehens: Vorgehen nach einer def nierten Änderungsprozedur mit Durchlauf geforderter Zustände(V-Modell).
- Einleitung und Überwachung der **Änderungsdurchführung**: Darstellung des Änderungsgeschehens in einem Logbuch, dessen Ein-tragungen sich immer auf def nierte Versionen von Komponenten beziehen. Daraus sollen Änderungsstatistiken auf Abruf generierbar sein.
- Abschluß und **Auswertung der Änderung**:
  - Alle Änderungen sollen nachvollziehbar und rekonstrukturierbar sein.
  - Es ist eine Historie zu führen, die alle Änderungsdaten einschließlich eines ausführlichen Kommentars enthält.
  - Als Vorgehensbaustein „Problem- und Änderungsmanagement“ im V-Modell XT realisiert.



## Aufgabenmanagement-Tools



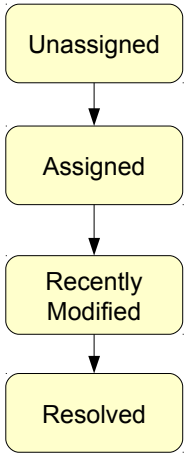
| | | |
|---|---|---|
| Bugzilla | Mozilla (OSS) | www.bugzilla.org |
| Mantis | OSS | http://www.mantisbt.org/ |
| | | |

http://en.wikipedia.org/wiki/Comparison_of_issue_tracking_systems

# Aufgabenmanagment (Fehler und Änderungen) mit Mantis



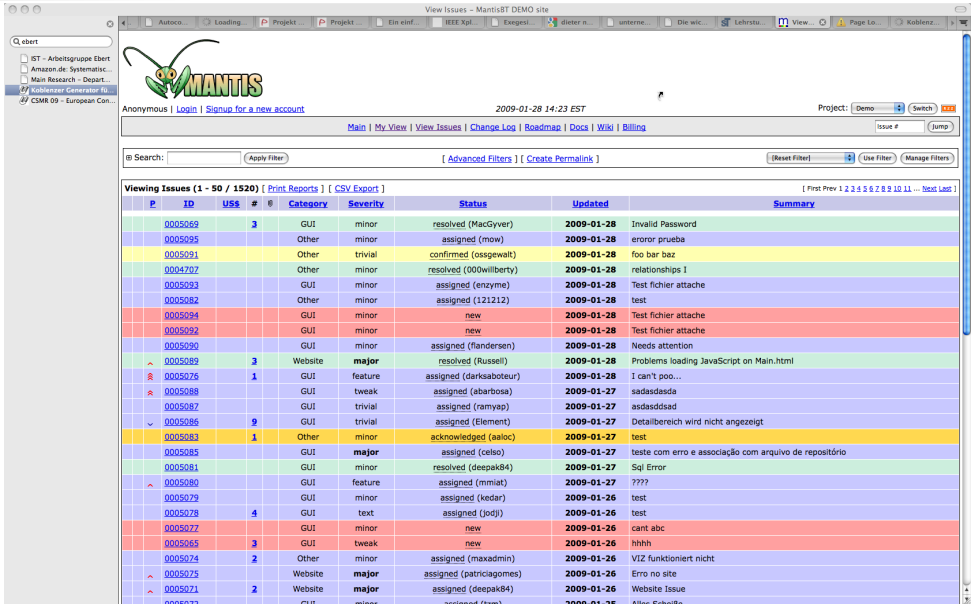
- Mantis ist ein webbasiertes Aufgabenmanagement-System (issue tracking system)
  - Zustandsmaschine für Fehler und Änderungswünsche
  - Generierung von emails über Statusänderungen
  - Visualisierung von Zuständen und Bearbeitern






# Überblick über Aufgaben [www.mantisbt.org]

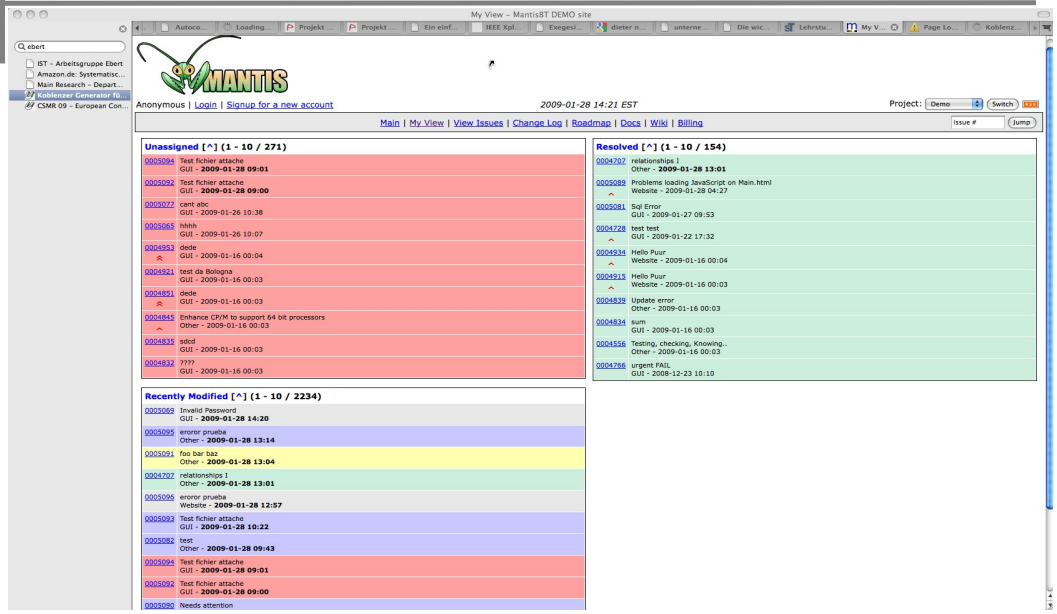







# Überblick über Zustände der Aufgaben








# The End