technische universität
dortmund

Software- Engineering für langlebige Systeme

Thomas Ruhroth | Dortmund
SoSe 2014

Updates

VL8

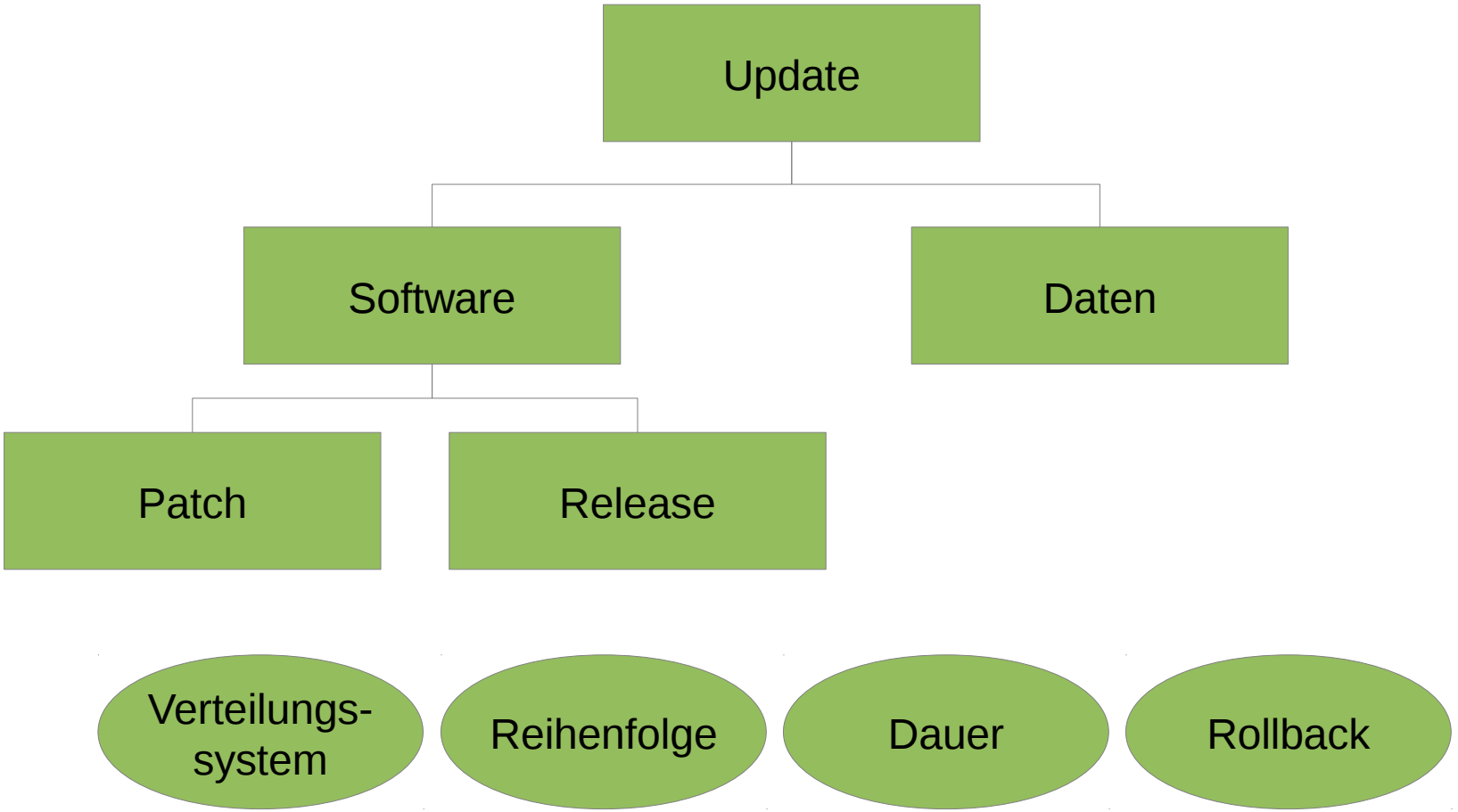
- Update-Arten
- Programm-Update
- Daten-Update

- Ziele:
  - Arten und Probleme von Updates kennen

Updates

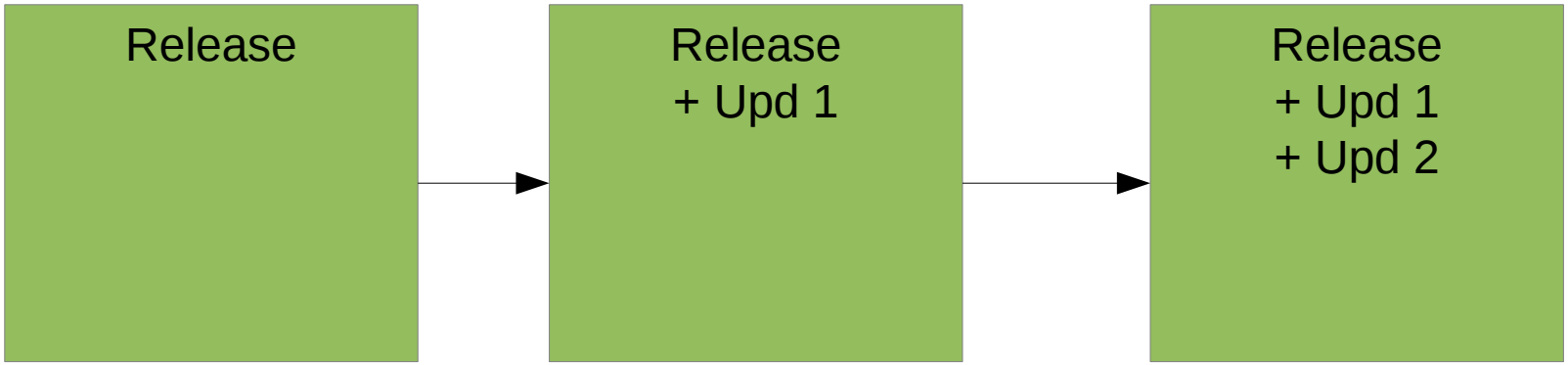
- Installation auf einen neuen Stand bringen
- Systemunterbrechung gering halten
- Sicherheit wiederherstellen
- Fehler beseitigen
- Features hinzufügen

Eine unvollständige Taxometrie



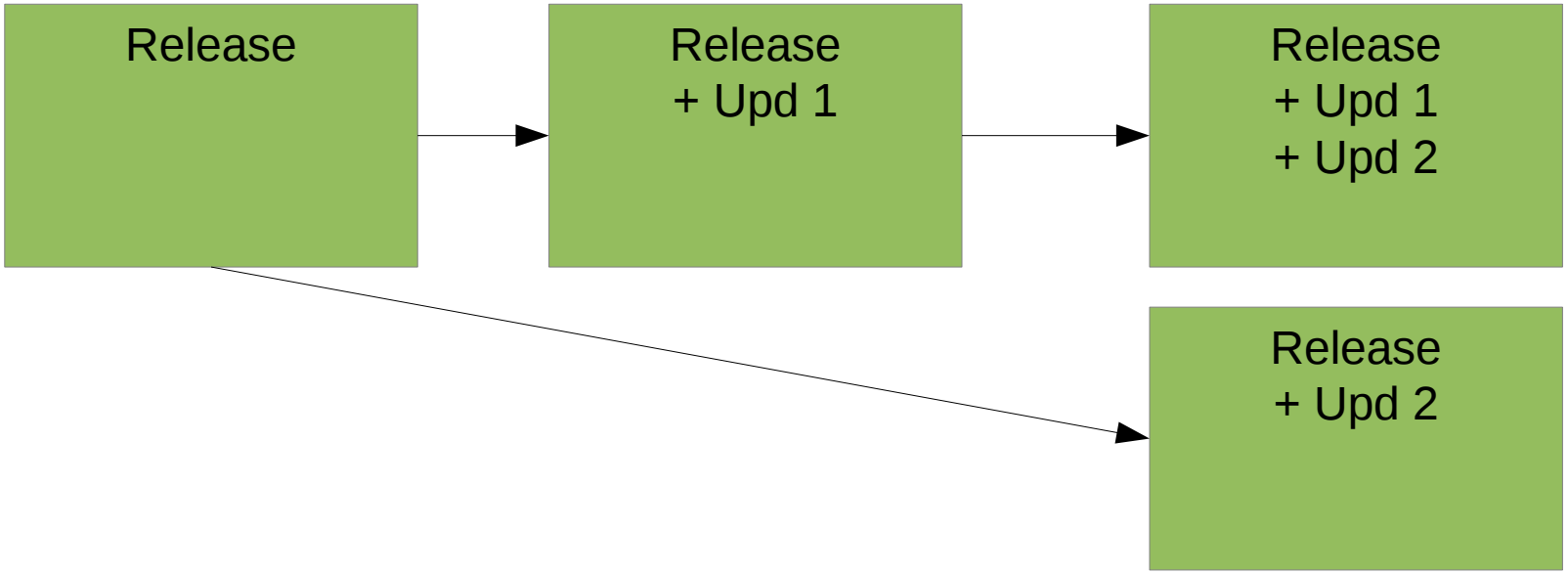
Anforderungen: Reihenfolge

- Alle-Szenario: Alle Updates werden immer durchgeführt
  - Update kann nur von einer festgelegten Version aus durchgeführt werden
  - Mehrere Updateschritte in einem sind möglich
    - Oft „Überspringen“ genannt



Anforderungen: Reihenfolge

- Einzel-Szenario: Updates können unabhängig eingespielt werden
  - Update kann für eine beliebige „kompatible“ Version erfolgen
  - Nicht alle Versionen beinhalten alle Updates



Anforderung: Rollback

- Zurücksetzen auf einen vorherigen (hoffentlich funktionstüchtigen) Stand
- Betrifft:
  - Software
  - Konfigurationen
  - Daten
- Wichtig bei kritischen System
  - Börse, Banken
  - Krankenhäuser, Notfalldienste (Polizei, Feuerwehr)
  - Raumfahrt, Satelliten

Software

- Parallel-Installation
  - Vollständige neue Installation
  - Übernahme von Einstellungen automatisch oder manuell
- Teilersetzung
  - Teile der Software werden ersetzt
- Patch
  - Teile von Dateien werden geändert/ersetzt

Parallel-Installation

- Vorteil:
  - Wenig Aufwand, da ähnlich einer Neuinstallation
  - Einmalige Übernahme der Einstellungen
  - Alte Installation kann weiter genutzt werden
    - Nur wenn Datenhaltung unverändert
    - Rollback-Option
  - Alle-Szenario
- Nachteile:
  - Benötigt viel Ressourcen (Speicherplatz, Installationverteilung)
  - Ungeeignet für kleine Änderungen
    - Aufwand/Nutzen stehen nicht in Relation
  - Alle-Szenario
- Nutzung: Meist für neue Releases

Teilersetzung

- Vorteil:
  - Nutzung weniger Ressourcen als Parallel-Installation
  - Konfiguration kann in weiten Teilen unverändert bleiben
  - Rollback
    - Sichern der Altkomponenten und bei Bedarf Rückersetzung
  - Einzel-Szenario/Alle-Szenario
- Nachteile:
  - Versionsmanagement für alle Teilsysteme/Komponenten nötig
  - Einzel-Szenario/Alle-Szenario
- Nutzung:
  - Gemeinsam genutzte Komponenten
  - Linux/Unix (DEP, RPM)

Patch

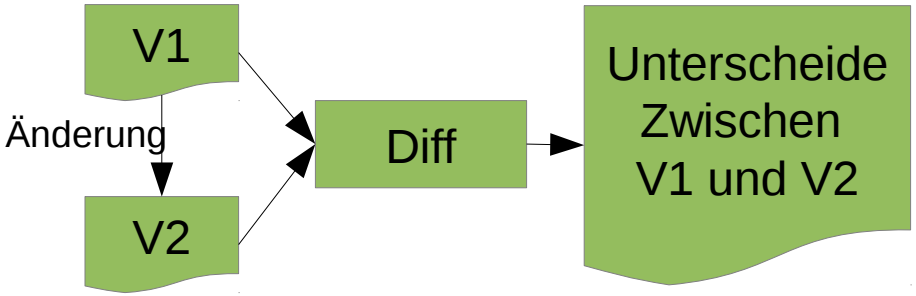
- Teile von Dateien werden geändert/ersetzt
- Vorteil:
  - Geringe Nutzung von Ressourcen
  - Konfiguration kann in weiten Teilen unverändert bleiben
  - Einzel-Szenario
- Nachteil:
  - Versionierung aller Dateien
  - Keine linearen Versionierungskennzeichnen möglich
  - Rollback schwierig (aber möglich)
- Nutzung:
  - Sicherheitspatches unabhängig von funktionalen Patches einspielen
  - Windows Update

Patchen

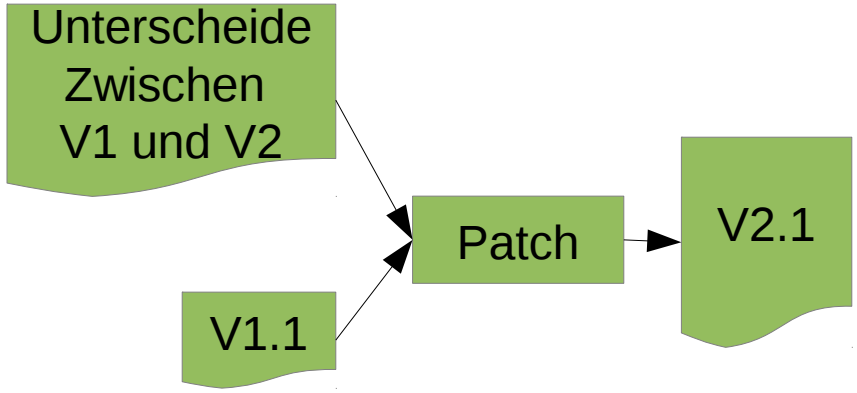
- Kernproblem:
  - Finden der Stellen in einer Datei die geändert werden müssen
  - Anpassen der Dateien
- Gibt es für binäre Dateien (auch für ausführbare Dateien) und Text-Dateien
- Im folgenden am Beispiel von Textdateien

Linux: Die Programme Diff und Patch

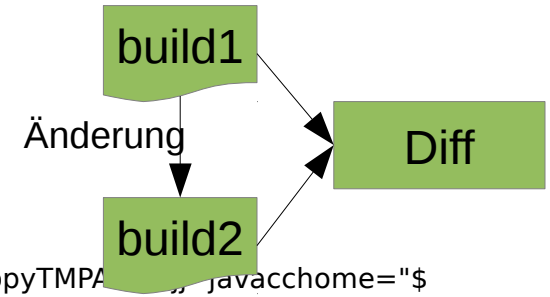
- Diff erzeugt Änderungsinformationen zwischen zwei Versionen einer Datei



- Patch versucht die Änderungen in eine Dateien einzuarbeiten

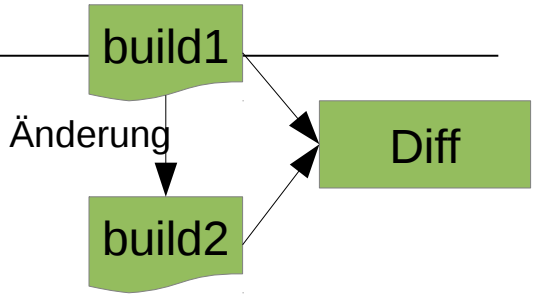


Diff



- >Diff build1.xml build2.xml

```
63,65c63,65
<           <javacc target="${copyTMPAST}/JavaccAST.jj" outputdirectory="${copyTMPA                    javacchome="$
{javaccRoot}"/>
<           <jjdoc javacchome="${javaccRoot}" target="${copyTMPAST}/JavaccAST.jj"/>
<           <move file="JavaccAST.html" todir="./info" />
---
>           <javacc target="${copyTMPAST}/javaccAST.jj" outputdirectory="${copyTMPAST}/jj" javacchome="$
{javaccRoot}"/>
>           <jjdoc javacchome="${javaccRoot}" target="${copyTMPAST}/javaccAST.jj"/>
>           <move file="javaccAST.html" todir="./info" />
69c69
<      <target name="init" depends="generateFiles">
---
>      <target name="init">
104c104
<      <target name="generate4Java" depends="init4Java">
---
>      <target name="generate4Java" depends="init4Java, generateFiles">
133c133
<           <javac includeantruntime="false"
---
>           <javac
```



Diff

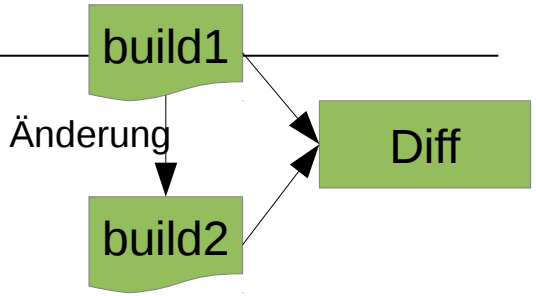
- >Diff build1.xml build2.xml

```
63,65c63,65
<          <javacc target="${copyTMPAST}/JavaccAST.jj" outputdirectory="${copyTMPAST}/jj" javacchome="$
{javaccRoot}"/>
<          <jjdoc javacchome="${javaccRoot}" target="${copyTMPAST}/JavaccAST.jj"/>
<          <move file="JavaccAST.html" todir="./info" />
---
>          <javacc target="${copyTMPAST}/javaccAST.jj" outputdirectory="${copyTMPAST}/jj" javacchome="$
{javaccRoot}"/>
>          <jjdoc javacchome="${javaccRoot}" target="${copyTMPAST}/javaccAST.jj"/>
>          <move file="javaccAST.html" todir="./info" />
69c69
<      <target name="init" depends="generateFiles">
---
>      <target name="init">
104c104
<      <target name="generate4Java" depends="init4Java">
---
>      <target name="generate4Java" depends="init4Java, generateFiles">
133c133
<          <javac includeantruntime="false"
---
>          <javac
```

Es fehlen Context-Informationen

Context-Informationen

- Zeilen vor und nach der Änderung
  - Erkennung der Änderungsstellen, falls sich die Zeilennummern verschoben haben
- Verschiedene Möglichkeiten:
  - -c, -C NUM, --context[=NUM]
    output NUM (default 3) lines of copied context
  - -u, -U NUM, --unified[=NUM]
    output NUM (default 3) lines of unified context
  - -e, --ed
    output an ed script
  - -n, --rcs
    output an RCS format diff



- >Diff -c build1.xml build2.xml

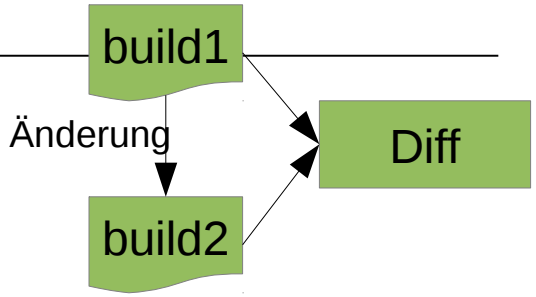
```
*** build1.xml  2013-07-01 13:03:32.168052586 +0200
--- build2.xml  2013-07-01 13:03:52.544051678 +0200
***************
*** 60,72 ****
            <move file="java_cleaned.html" todir="./info" />

            <jjtree target="${ASTParserJJ}" outputdirectory="${copyTMPAST}" javacchome="${javaccRoot}"/>
!           <javacc target="${copyTMPAST}/JavaccAST.jj" outputdirectory="${copyTMPAST}/jj" javacchome="${javaccRoot}"/>
!           <jjdoc javacchome="${javaccRoot}" target="${copyTMPAST}/JavaccAST.jj"/>
!           <move file="JavaccAST.html" todir="./info" />
      </target>


!     <target name="init" depends="generateFiles">
            <mkdir dir="${copyDir}" />
            <mkdir dir="${copyTMP}" />
      </target>
--- 60,72 ----
            <move file="java_cleaned.html" todir="./info" />

            <jjtree target="${ASTParserJJ}" outputdirectory="${copyTMPAST}" javacchome="${javaccRoot}"/>
!           <javacc target="${copyTMPAST}/javaccAST.jj" outputdirectory="${copyTMPAST}/jj" javacchome="${javaccRoot}"/>
!           <jjdoc javacchome="${javaccRoot}" target="${copyTMPAST}/javaccAST.jj"/>
!           <move file="javaccAST.html" todir="./info" />
      </target>


!     <target name="init">
            <mkdir dir="${copyDir}" />
            <mkdir dir="${copyTMP}" />
      </target>
```



- >Diff -u build1.xml build2.xml

```
--- build1.xml          2013-07-01 13:03:32.168052586 +0200
+++ build2.xml          2013-07-01 13:03:52.544051678 +0200
@@ -60,13 +60,13 @@
        <move file="java_cleaned.html" todir="./info" />

        <jjtree target="${ASTParserJJ}" outputdirectory="${copyTMPAST}" javacchome="${javaccRoot}"/>
-       <javacc target="${copyTMPAST}/JavaccAST.jj" outputdirectory="${copyTMPAST}/jj" javacchome="${javaccRoot}"/>
-       <jjdoc javacchome="${javaccRoot}" target="${copyTMPAST}/JavaccAST.jj"/>
-       <move file="JavaccAST.html" todir="./info" />
+       <javacc target="${copyTMPAST}/javaccAST.jj" outputdirectory="${copyTMPAST}/jj" javacchome="${javaccRoot}"/>
+       <jjdoc javacchome="${javaccRoot}" target="${copyTMPAST}/javaccAST.jj"/>
+       <move file="javaccAST.html" todir="./info" />
        </target>


-      <target name="init" depends="generateFiles">
+      <target name="init">
        <mkdir dir="${copyDir}" />
        <mkdir dir="${copyTMP}" />
        </target>
```

Patch

Usage: patch [OPTION]... [ORIGFILE [PATCHFILE]]

Input options:

-p NUM --strip=NUM Strip NUM leading components from file names.
-F LINES --fuzz LINES Set the fuzz factor to LINES for inexact matching.
-l --ignore-whitespace Ignore white space changes between patch and input.

-c --context Interpret the patch as a context difference.
-e --ed Interpret the patch as an ed script.
-n --normal Interpret the patch as a normal difference.
-u --unified Interpret the patch as a unified difference.

-N --forward Ignore patches that appear to be reversed or already applied.
-R --reverse Assume patches were created with old and new files swapped.

-i PATCHFILE --input=PATCHFILE Read patch from PATCHFILE instead of stdin.

Patch

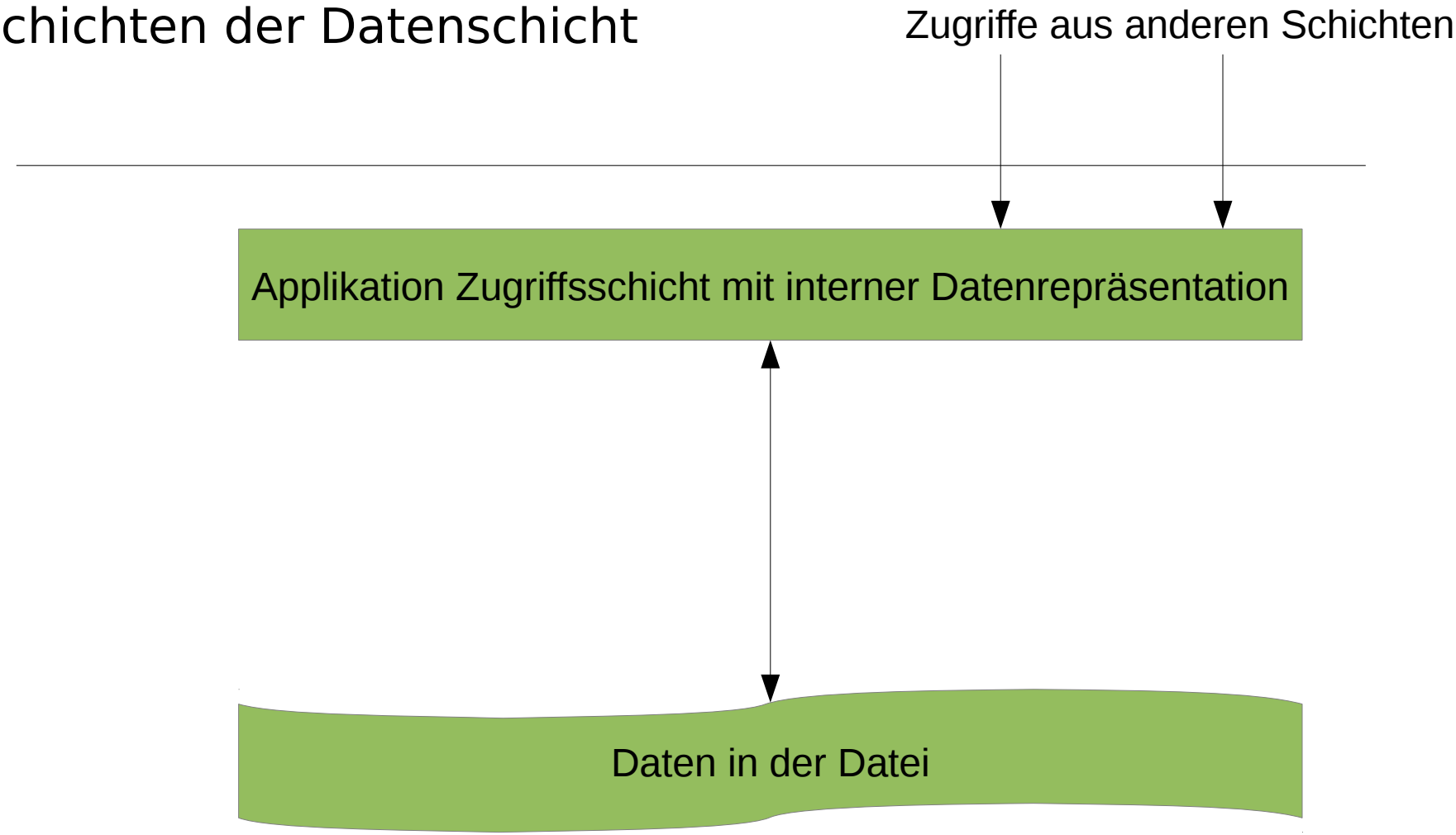
- Patch -u build1.xml diff.txt
  - Ergibt wieder build2.xml
- Patch kann nach Veränderungen der Dateien angewandt werden, solange die Patchstellen unverändert sind
- Whitspaces und Contexte können durch Parameter angepasst werden
  - Ignorieren von Whitespraces
  - Größe des Context
- Eingeschränkt auch für binäre Dateien nutzbar

Datenmigration

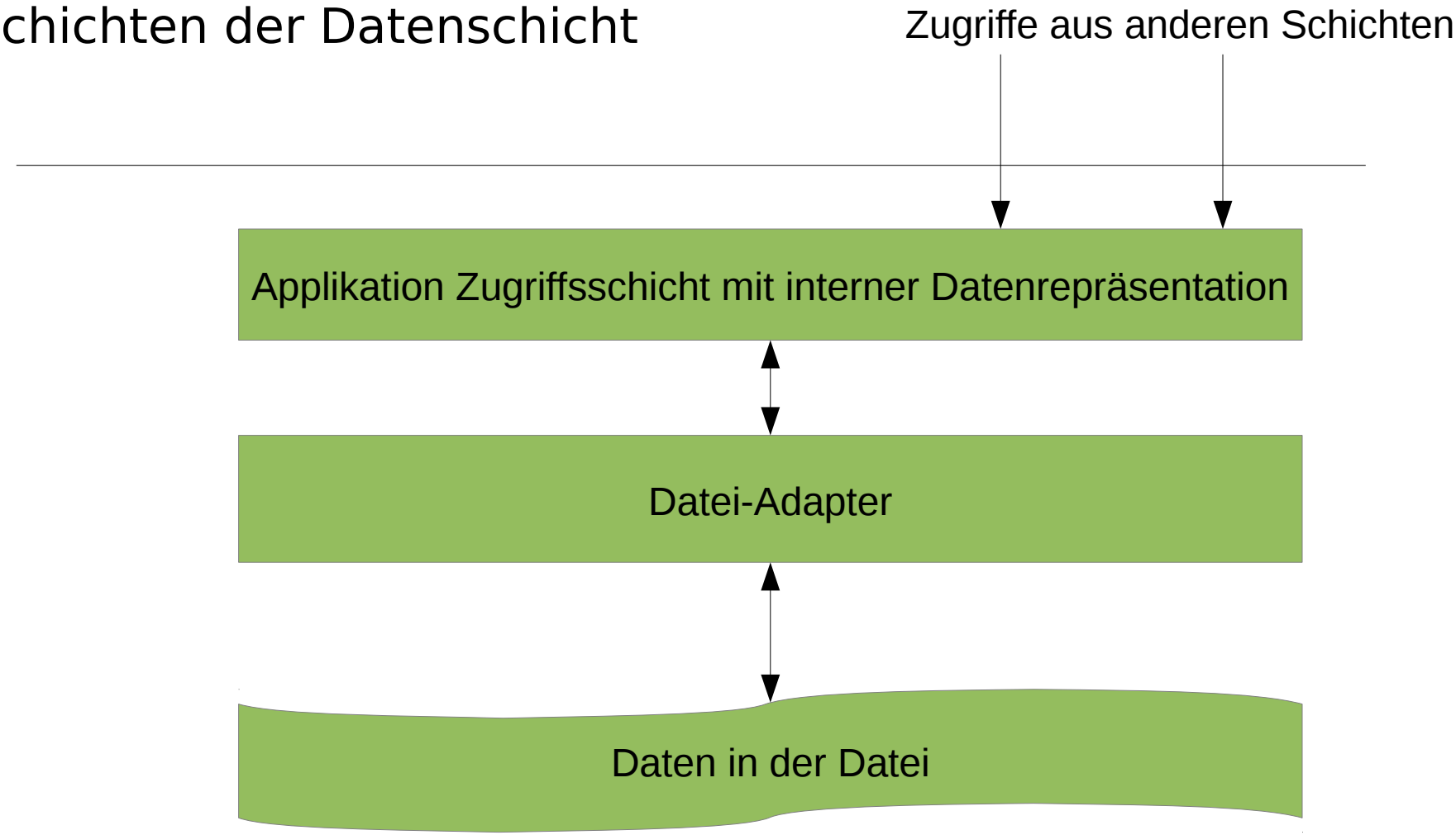
Daten-Migration

- Problem: Datenhaltungen (Dateien, Datenbankschemata) können durch ein Update verändert werden.
  - Alte Daten müssen migriert werden
  - Daten müssen ergänzt werden
    - Defaultwerte
    - Abgeleitete Werte
    - Neuaufnahme
      - Manuell
      - Automatisch aus anderen Datenquellen
- Hilfreich: Schichtenarchitektur

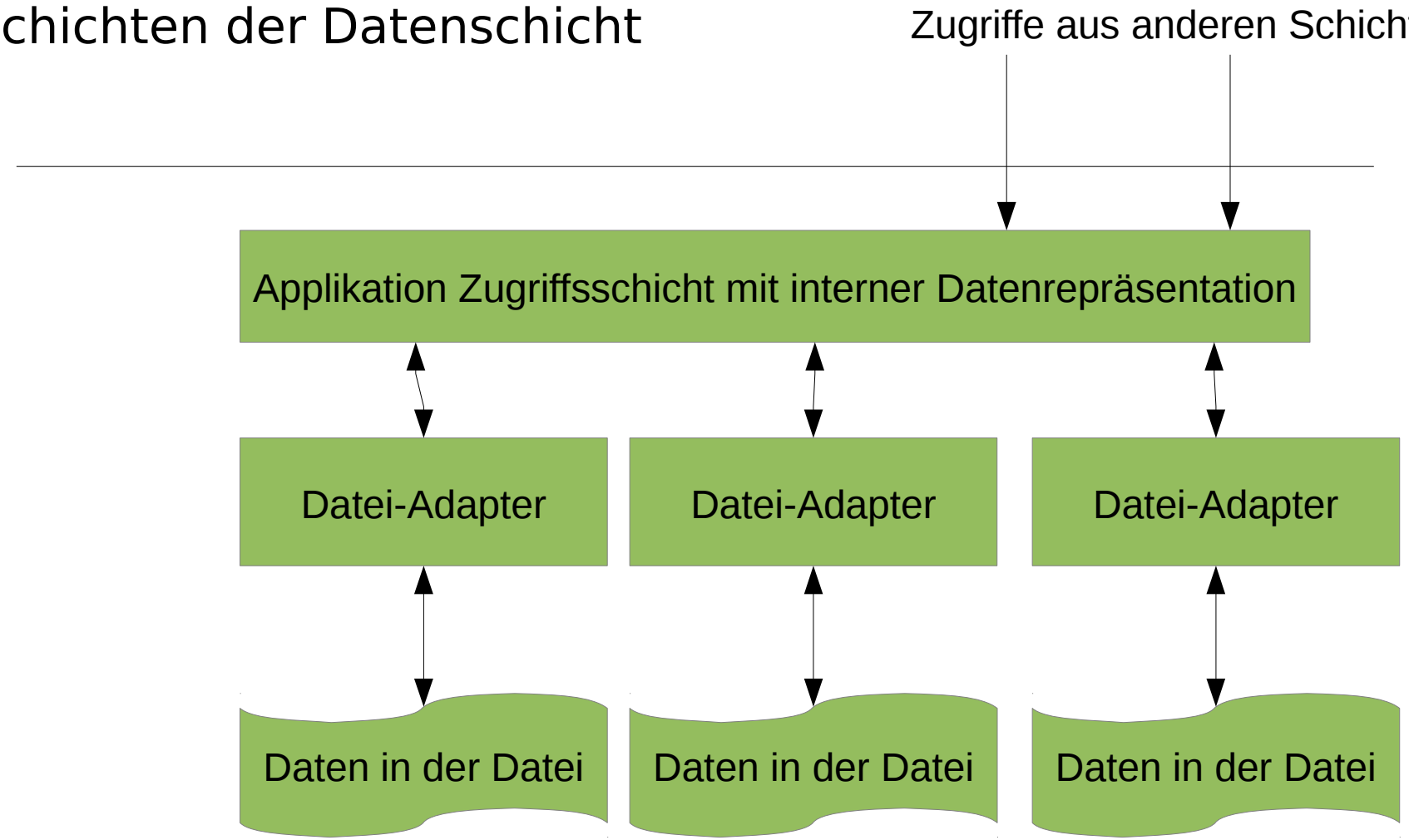
Schichten der Datenschicht



Schichten der Datenschicht



Schichten der Datenschicht



Datei-Adapter in der Datenschicht

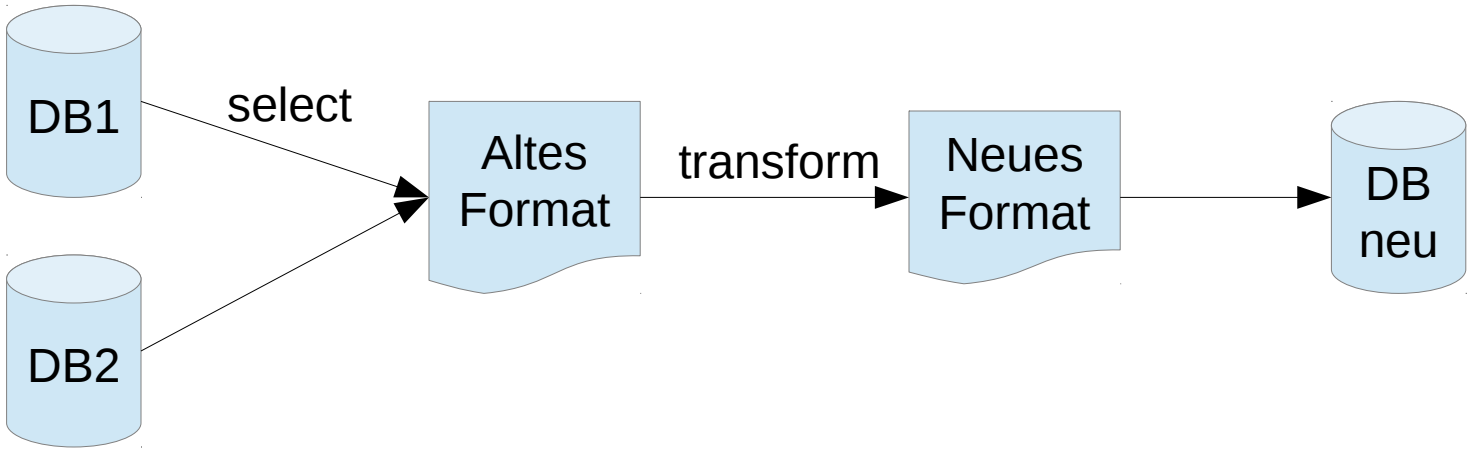
- Vorteil:
  - Daten können in verschiedenen Formaten geladen und gespeichert werden
  - Überspringen von Versionen möglich
  - Nutzung von Fremddaten
- Nachteil:
  - Änderung der internen Strukturen nur eingeschränkt möglich
  - Wartungsaufwand für verschiedene Adaptoren

Datenbanken

- Vorgehen Datenbanken bei kleinen Datenmengen wie bei Dateien möglich
- Vollmigration
  - Daten werden in eine neue Datenbank überführt
- Große Datenbanken werden meist „online“ angepasst
  - Schema für einen Rollback kompatible?
    - Evtl. Einschränkungen in der Struktur des neuen Schemas
  - Wie schnell lassen sich die Änderungen einspielen?
    - Ausfalldauer des Systems minimieren
  - Problem: Verteilte Datenbanken
    - Konsistenz zwischen den Versionen

Extract, Transform, Load (ETL)

- Extraktion
  - der relevanten Daten aus verschiedenen Quellen
- Transformation
  - der Daten in das Schema und Format der Zieldatenbank
- Laden
  - der Daten in die Zieldatenbank



Transformieren von Datenbankdaten

- Nicht eingehaltene Normalform
- Unvollständige Daten
- Trennen der nichtatomaren Daten
- Auffüllen von fehlenden Informationen

Key	Name	Menge
143	Mehl	1kg
157	Karotten	7
195	Tomaten	10 Stk

Key	Name	Menge	MU
143	Mehl	1	kg
157	Karotten	7	Stk
195	Tomaten	10	Stk

Problem Fehler in den Daten

- Entstehung
  - Abstürze
  - Programmierfehler
    - Auch korrigierte Programmfehler können Daten verfälscht haben
  - Fehler aus vorherigen Datenübernahmen
  - Änderungen ohne alte Daten konvertiert zu haben
  - ....
- Probleme
  - Datenübernahme „läuft nicht durch“
  - Berechnungen -> Fehlerfortpflanzung
  - Teilweise nicht (einfach) zu finden
  - Doppelte Daten
  - Nicht erreichbare Daten (Tote Daten)

Umgang mit fehlerhaften Daten

- Gruppenbildung und dann Gruppenweise behandeln
  - Bei großen Datenmengen bei denen gleiche Fehler häufig auftreten
- Fehler per Hand korrigieren
- Fehler durch andere (redundante) Quellen finden und korrigieren
  - z.B. Adressdatenbanken

Nächste Woche:

Technical Writing