



Versionskontrolle

Begriffe

- **Version**
 - Ein Objekt in einem abgespeicherten Bearbeitungszustand.
- **Revision** (lat.): wiederanschauen
 - Ein geändertes Element, das seinen Vorgänger ersetzt.
- **Konfiguration** (lat.): Erschaffung, Entwurf, Formation
 - Naturwissenschaften: Gruppierung von Elementen zu Systemen
 - Informatik: Selektion von Revisionen der Komponenten eines Systems
- **Variante** (lat.):
 - Ein Element, das mit Elementen gleicher Funktion ko-existiert.
- **Repository**
 - Ort, wo Versionen von Objekten (z.B. Dateien) gespeichert werden.
- **Arbeitsumgebung (Workspace, Sandbox)**
 - Ein nur einem Entwickler zur Verfügung stehender Raum, indem versionierte Objekte modifiziert oder transformiert werden.

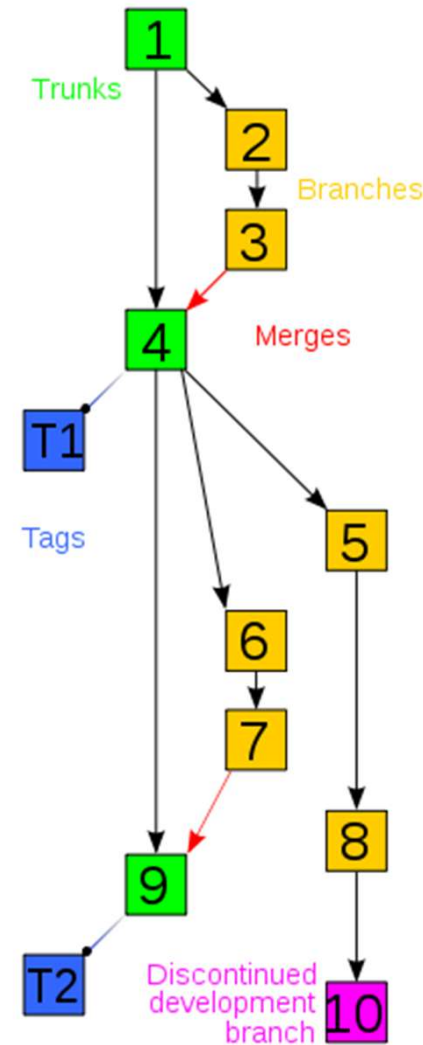
Ziele

- Versionskontrolle
 - Wiederherstellung früherer Versionen
 - Rekonstruktion ausgelieferter Versionen
 - Parallele Entwicklung von verschiedenen Lösungen
 - Dokumentation der Änderungshistorie
 - Klärung der Verantwortlichkeit
- Konfigurationsmanagement
 - Selektion der Komponenten eines Systems
 - Handhabung von Varianten
- (Tele-)Kooperation
 - Arbeitsmodelle
 - Arbeitsteilung, Prozessketten

Themen

Source: https://en.wikipedia.org/wiki/File:Revision_controlled_project_visualization-2010-24-02.svg

- Versionskontrolle
 - o Identifikation
 - o Werkzeuge, Speicherungstechniken
 - o Zugriffskontrollstrategien
 - o Änderungshistorie
- Konfigurationsmanagement
 - o Selektion der System-Komponenten
 - o Konsistenz

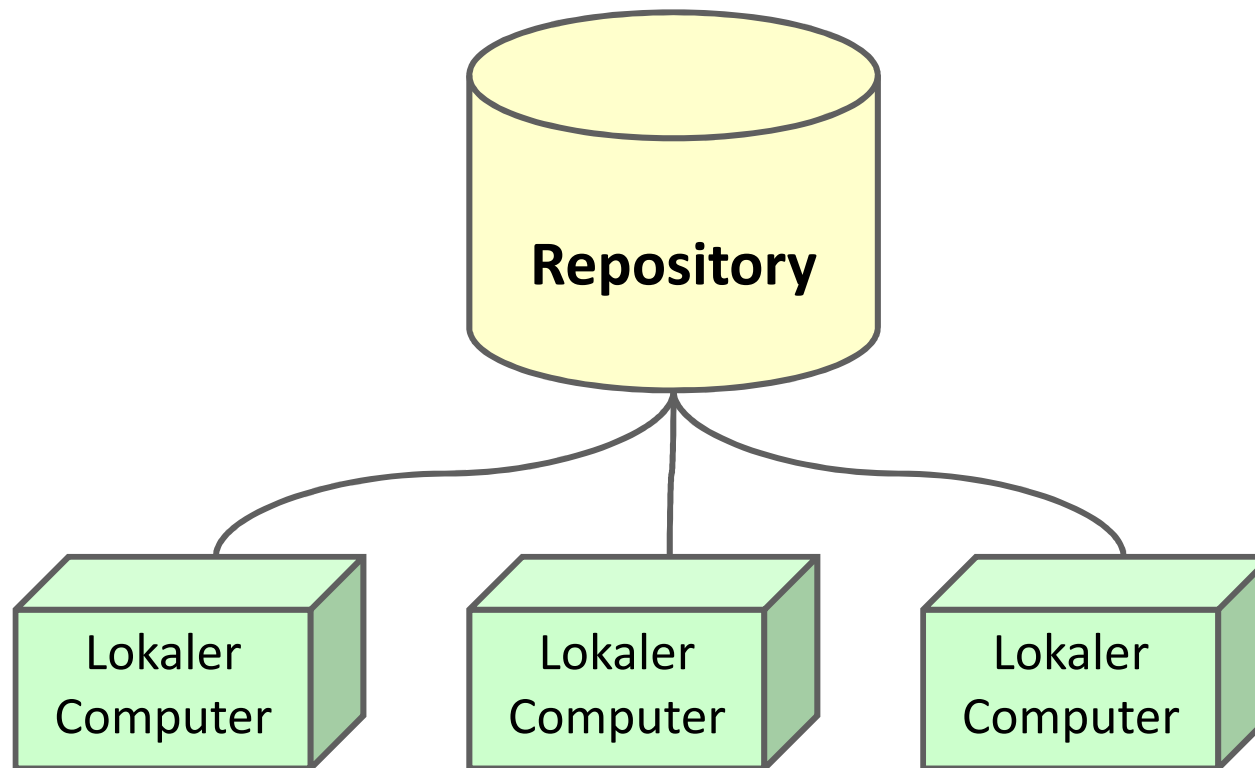


Beispiel einer Richtlinie für Versionsbezeichnungen

Versionsbezeichnung

- Die Version vor dem ersten Review eines Dokuments oder Programms erhält die Bezeichnung „Entwurf“.
- Jede im Rahmen eines Reviews mit dem Kunden vorgelegte Version erhält eine neue Hauptziffer (1.0, 2.0 usw.).
- Versionen, welche zwischen Reviews an den Kunden ausgehändigt werden, erhalten eine neue, erste Nebenziffer (1.1, 1.2 usw.).
- Versionen zwischen zwei dem Kunden ausgehändigten Versionen erhalten eine neue, zweite Nebenziffer (1.1.1, 1.1.2 usw.). In der dem Kunden ausgehändigten Version wird nur die Hauptziffer und die erste Nebenziffer angegeben.

Zentralisiertes Repository



Werkzeuge/Produkte

RCS

CVS

Subversion (-> SWE 2)

MKS Source Integrity

ClearCase

Perforce

Quelle: Martin Pfeiffer, Software-Engineering 2, Skript, Hochschule Pforzheim

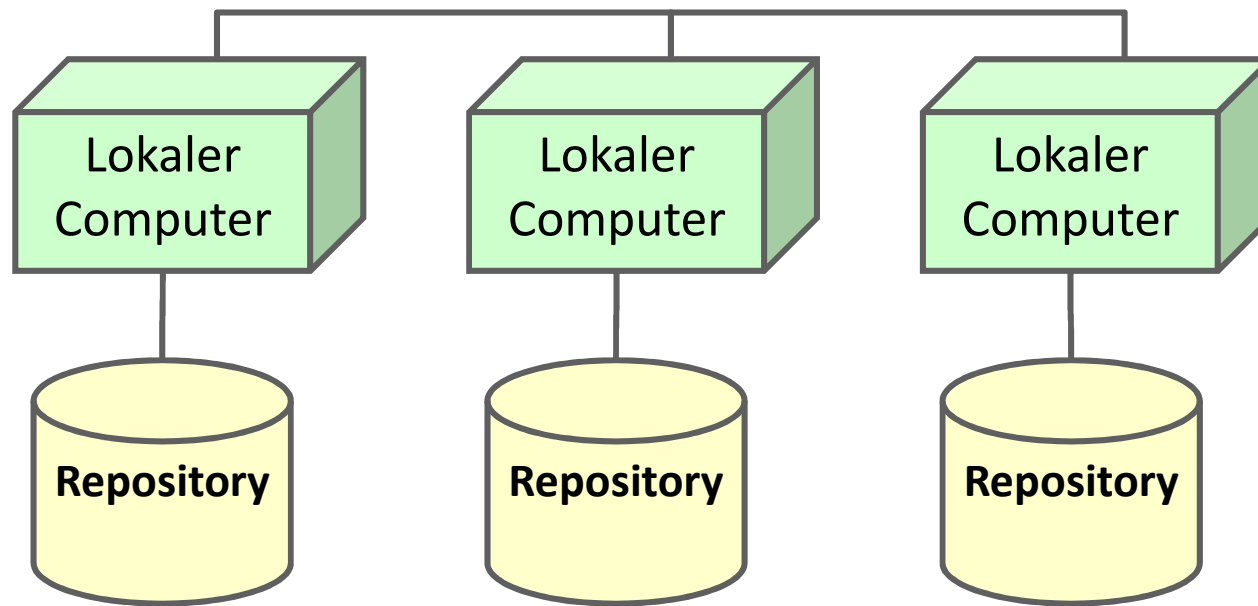


Verteiltes Repository

Werkzeuge/Produkte

Mercurial

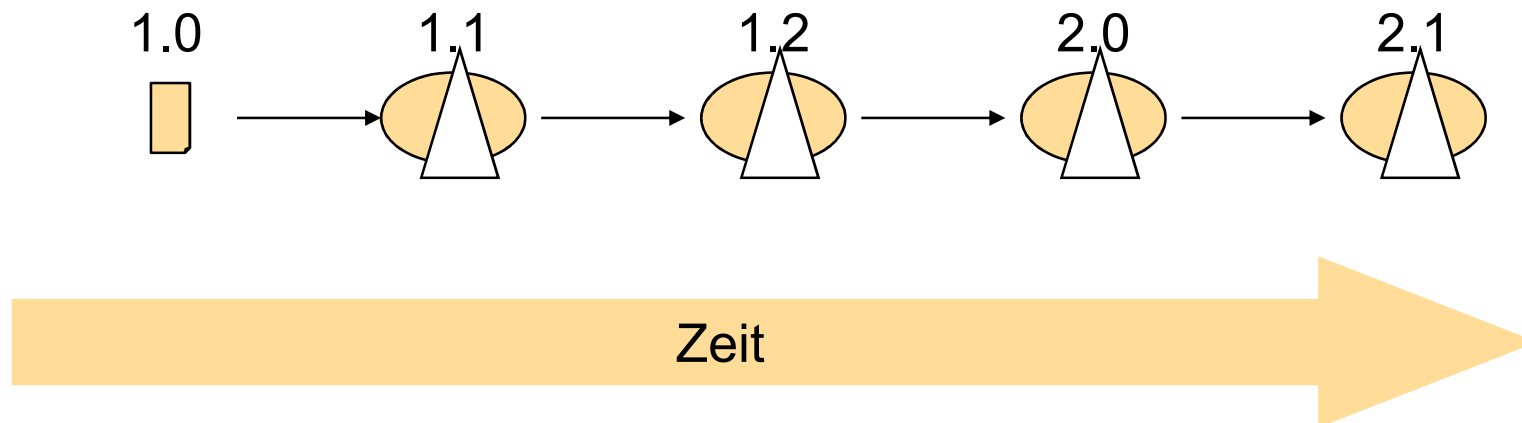
Git



Quelle : Martin Pfeiffer, Software-Engineering 2, Skript, Hochschule Pforzheim

Speicherung von Versionen – Vorwärts-Deltas

- Speichere Vorwärts-Deltas, nicht die vollständige Datei
 - Das Delta vom Nachfolger zum Vorgänger wird gespeichert
 - Basis sind
 - eingefügte,
 - gelöschte und
 - modifizierte Zeilen
 - Kostet wenig Speicher, aber mehr Verarbeitungszeit , insbesondere für die augenblickliche Version



Implementierung von Vorwärts-Deltas

```
#include „liste.h“  
// Class ListenElement  
ListenElement::ListenElement( ) { // X::X()  
    next = 0;  
    previous = 0;  
    objectPointer = 0;  
}  
  
ListenElement::ListenElement (const el ...  
    next = 0; // X::X( X & )  
    previous = 0;  
    objectPointer = el.objectPointer;  
}
```

2.1

Implementiert durch ed-script
(Unix)

save

Delta:

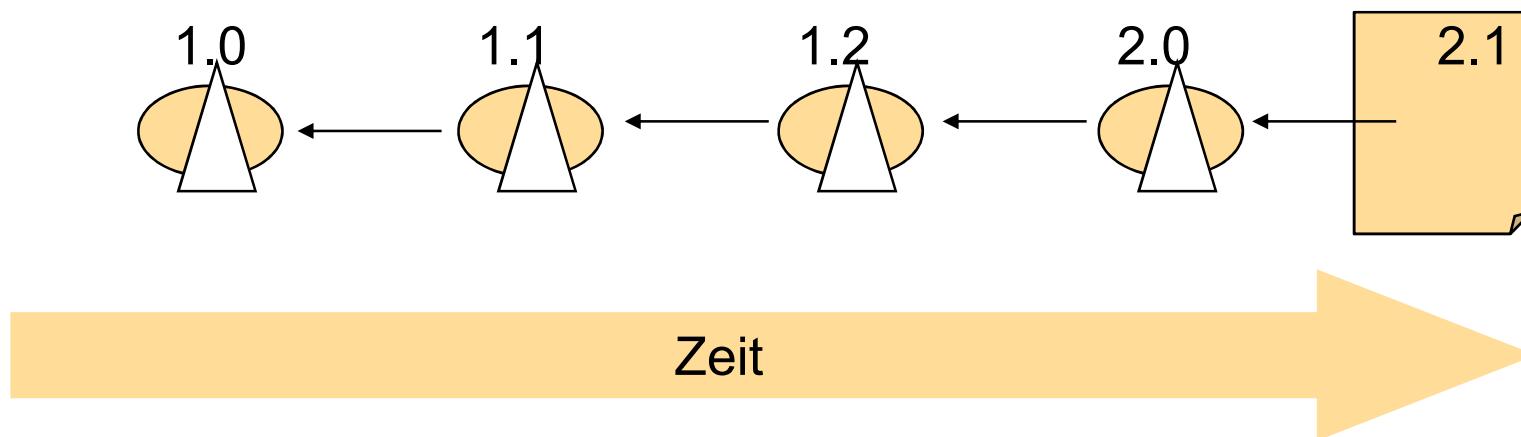
- insert(5, „previous = 0;“);
- replace(9, „next = 0; // x::X(X &)“);

Speicherung von Versionen – Rückwärts-Deltas

- Speichere Rückwärts-Deltas, nicht die vollständige Datei
 - o Das Delta vom Vorgänger zu seinem Nachfolger wird gespeichert
 - o Basis sind
 - eingefügte,
 - gelöschte und
 - modifizierte Zeilen
 - o Kostet wenig Speicher, aber mehr Verarbeitungszeit, insbesondere für die vorherigen Version, aktuelle Version kostet keinen Aufwand

1980s (Dissertation Walter Tichy: RCS)

Aktuelle Version als
komplettes Dokument



Implementierung von Rückwärts-Deltas

```
#include „liste.h“  
  
// Class ListenElement  
ListenElement::ListenElement( ) { // X::X()  
    next = 0;  
    previous = 0;  
    objectPointer = 0;  
}  
  
ListenElement::ListenElement (const el ...  
    next = 0;           // X::X( X & )  
    previous = 0;  
    objectPointer = el.objectPointer;  
}
```

2.1

Implementiert durch by ed-script
(Unix)

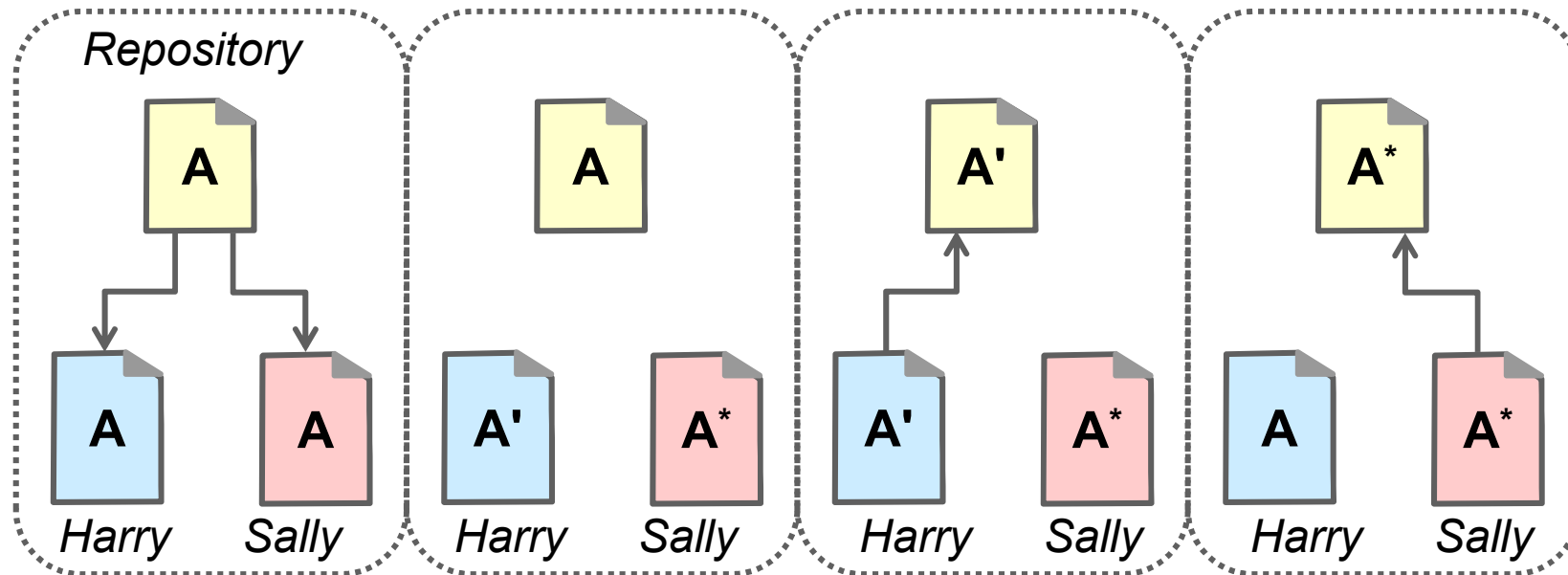
save

Delta:

- delete(5);
 - replace(9, „next = 0;“);
-
- Vergleich:
 - 2 Zeilen statt z.B. 450 lines
 - Walter Tichy: “Delta techniques uses about 10% of the original space as extra space”



Zugriffskontrollstrategien



Two users read a document from the repository by creating local copies

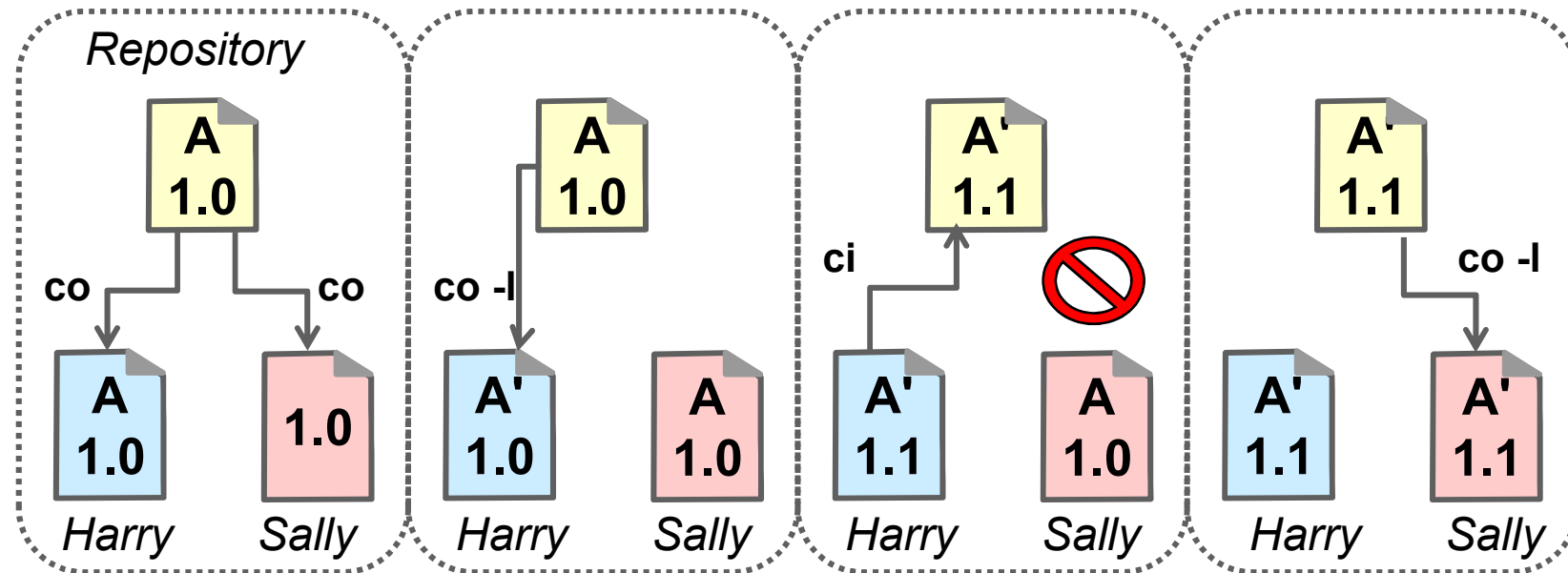
Both change the document independently from each other.

One user, Harry, writes back his changes into the repository.

The second user, Sally, writes back her changes into the repository, **overwriting the changes of Harry.**



Zugriffskontrollstrategien – Konservative Strategie (lock-modify-unlock)



Two users, Harry and Sally, read a document from the repository by creating local copies.

Before Harry make changes, he checks out the file with a lock.
Harry edits his file.

Harry checks in the changed file and releases the lock.
While the file is locked, Sally cannot make changes to her file.

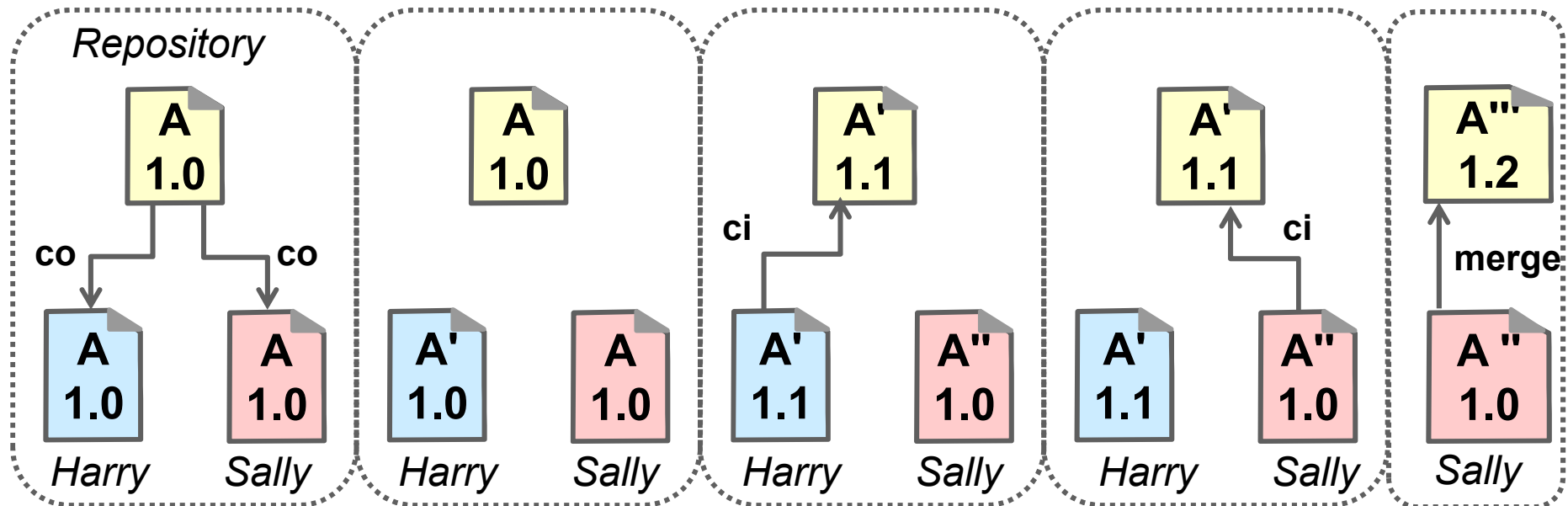


Before Sally make changes, she checks out the file with a lock.

She gets the version of the file that was changed by Harry.



Zugriffskontrollstrategien – Optimistische Strategie (copy-modify-merge)



Two users, Harry and Sally, read a document from the repository by creating local copies.

Harry edits his local file, Sally edits her local file.

Harry checks in the changed file.

On Sally's attempt to check in her changed file she gets an "out-of-date" message.

Harry's changes do not get lost, but Sally has to merge the files.



Beispiel:

Revision Control System (RCS, Tichy) - Funktionen



- **RCS-Initialisierung (rcs)**
 - o Versionsdatei erstellen
- **Check In / Commit (ci)**
 - o Datei ins Repository (RCS Unterverzeichnis)
 - o hebt vorher gesetzte Sperre auf
 - o fordert log message an
- **Check Out (co)**
 - o aktuelle oder spezifizierte Version
 - o mit oder ohne Sperre
- **Differenzen ermitteln (rcsdiff)**
 - o Vergleicht Versionen
- **Verschmelzen (rcsmerge)**
 - o 3-Wege-Verschmelzen mit gemeinsamer Basisversion
- **Info (rlog)**
 - o Aktuelle Version, Verzweigungen, Autor, Anzeige der Änderungen, Log-Texte, ...
- **Eingebettete Info (\$Id\$)**

```
/* $Id: Kreis.c,v 1.2 2005/01/06 16:45:20 labuser Exp labuser $ */
```

Dateiname

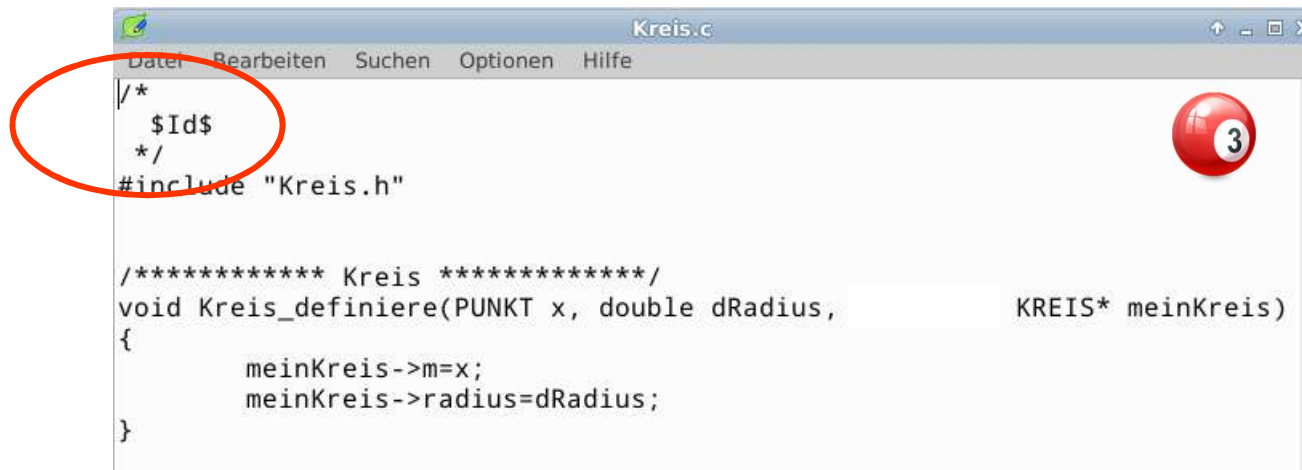
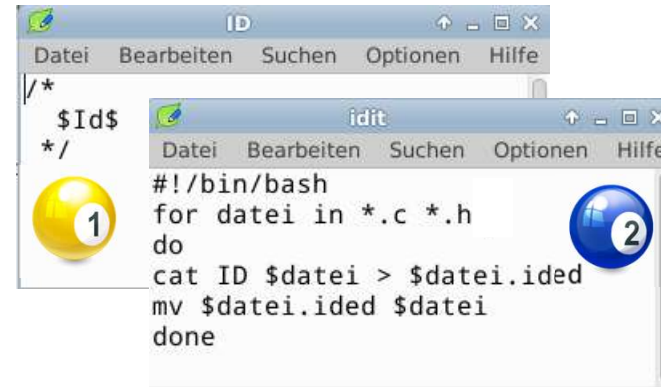
ci-Zeit

Autor

Halter der Sperre



Arbeiten mit RCS



RCS - Einrichten

4


```
> mkdir Repository
> ln -s Repository RCS
rcc -i -m0.1:"Initialization" Kreis.c
RCS file: RCS/Kreis.c,v
rcc: RCS/Kreis.c,v: RCS file empty
```

RCS Verzeichnis auch als symbolic link möglich

5

```
> ci Kreis.c
RCS/Kreis.c,v <-- Kreis.c
enter description, terminated with ↵
single '.' or end of file:
NOTE: This is NOT the log message!
>> First Entry
>> .
initial revision: 1.1
done
```

Kreis.c verschwindet aus dem Verzeichnis



Name	Größe	Typ	Änderungsdatum
RCS	4,0 KB	Verknüpfung mit Repository	2015-03-07 00:19:33
Repository	4,0 KB	Ordner	2015-03-07 00:19:33
Dreieck.c	1,6 KB	C-Quelltext	2015-03-06 23:57:17
Dreieck.h	636 bytes	C-Header	2015-03-06 23:57:17
Kreis.h	733 bytes	C-Header	2015-03-06 23:57:17
main.c	2,7 KB	C-Quelltext	2015-03-06 23:57:17
minmax.h	331 bytes	C-Header	2015-03-06 23:57:17
Punkt.c	239 bytes	C-Quelltext	2015-03-06 23:57:17
Punkt.h	384 bytes	C-Header	2015-03-06 23:57:17
Quadrat.c	1,3 KB	C-Quelltext	2015-03-06 23:57:17
Quadrat.h	719 bytes	C-Header	2015-03-06 23:57:17
Rechteck.c	1,5 KB	C-Quelltext	2015-03-06 23:57:17
Rechteck.h	695 bytes	C-Header	2015-03-06 23:57:17
Strecke.c	338 bytes	C-Quelltext	2015-03-06 23:57:17
Strecke.h	447 bytes	C-Header	2015-03-06 23:57:17

Kreis.c,v erscheint im Repository

Praktische Informatik 2	Software-Engineering 2	6 BoundingBox - Plain - Flat - RCS	Repository
Name	Größe	Typ	Änderungsdatum
Kreis.c,v	1,1 KB	Einfaches Textdokument	2015-03-07 00:20:23



Checkout

6 > co Kreis.c
RCS/Kreis.c,v --> Kreis.c
revision 1.1
Done
Kreis.c wird als lokale Kopie angelegt

> ll Kreis.c
-r--r--r-- 1 labuser users 990 Mär 7 00:33 Kreis.c
Kreis.c ist nur lesbar

```
/*  
 $Id: Kreis.c,v 1.1 2015/03/06 23:20:10 labuser Exp $  
 */  
#include "Kreis.h"
```

RCS Meta-Datei-Informationen

7 co -l Kreis.c
RCS/Kreis.c,v --> Kreis.c
revision 1.1 (locked)
Done
Kreis.c ist nun schreibbar

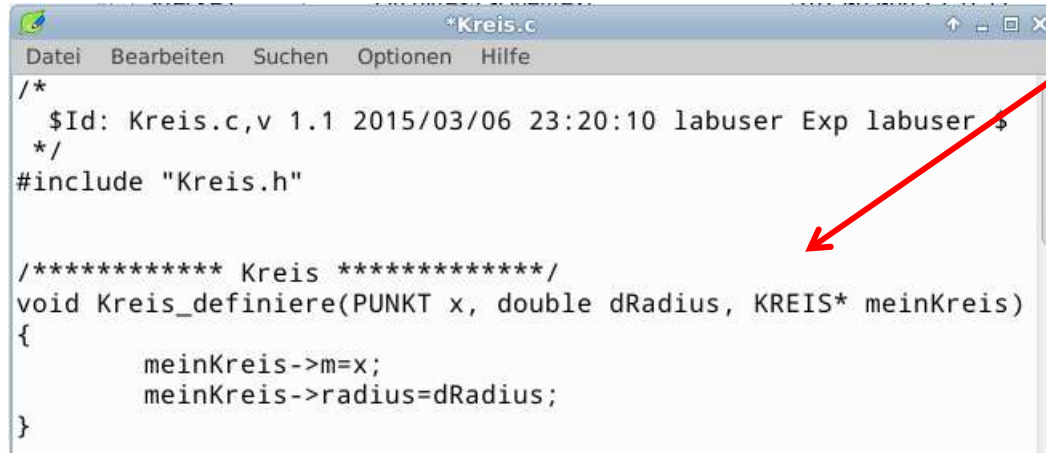
> ll Kreis.c
-rw-r--r-- 1 labuser users 998 Mär 7 00:41 Kreis.c

```
/*  
 $Id: Kreis.c,v 1.1 2015/03/06 23:20:10 labuser Exp labuser $  
 */  
#include "Kreis.h"
```

Die Datei ist von diesem user gesperrt.

Checkout

8 > leafpad Kreis.c



```
/*
 $Id: Kreis.c,v 1.1 2015/03/06 23:20:10 labuser Exp labuser $
 */
#include "Kreis.h"

/***** Kreis *****/
void Kreis_definiere(PUNKT x, double dRadius, KREIS* meinKreis)
{
    meinKreis->m=x;
    meinKreis->radius=dRadius;
}
```

Kreis.c wird hier geändert

9 ci Kreis.c

RCS/Kreis.c,v <-- Kreis.c

new revision: 1.2; previous revision: 1.1

enter log message, terminated with single '.' or end of file:

>> Removed instrumentation for splint

>> .

done

10 > co Kreis.c

RCS/Kreis.c,v --> Kreis.c

revision 1.2

done

Version aktualisiert



```
<Kreis.c>
Datei Bearbeiten Suchen Optionen Hilfe
/*
 $Id: Kreis.c,v 1.2 2015/03/07 00:00:05 labuser Exp $
 */
#include "Kreis.h"
```



Ziele der Versionskontrolle

Ziel

- Wiederherstellung früherer Versionen
- Rekonstruktion ausgelieferter Versionen
- Parallele Entwicklung verschiedener Lösungen
- Dokumentation der Änderungshistorie
- Klärung der Verantwortlichkeit

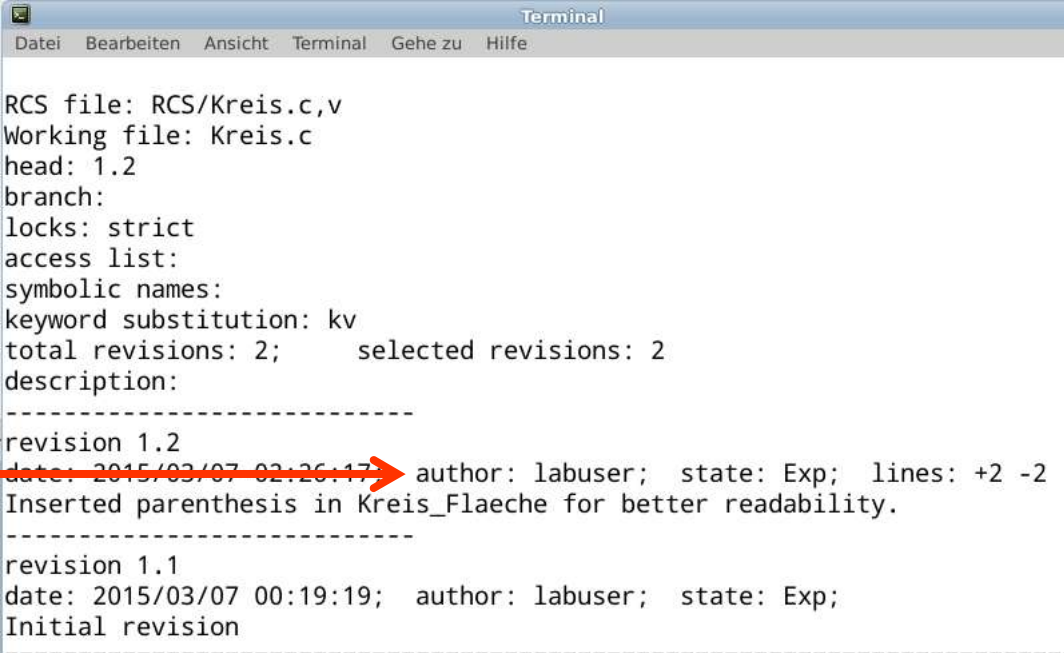
Umsetzung mit RCS

> co -r1.1 Kreis.c

hans> co -r1.1 Kreis.c

jana> co -r1.2 Kreis.c

rlog Kreis.c



```
Terminal
Datei  Bearbeiten  Ansicht  Terminal  Gehe zu  Hilfe

RCS file: RCS/Kreis.c,v
Working file: Kreis.c
head: 1.2
branch:
locks: strict
access list:
symbolic names:
keyword substitution: kv
total revisions: 2;    selected revisions: 2
description:
-----
revision 1.2
date: 2015/03/07 02:26:17; author: labuser; state: Exp; lines: +2 -2
Inserted parenthesis in Kreis_Flaeche for better readability.
-----
revision 1.1
date: 2015/03/07 00:19:19; author: labuser; state: Exp;
Initial revision
-----
```

Änderungshistorie

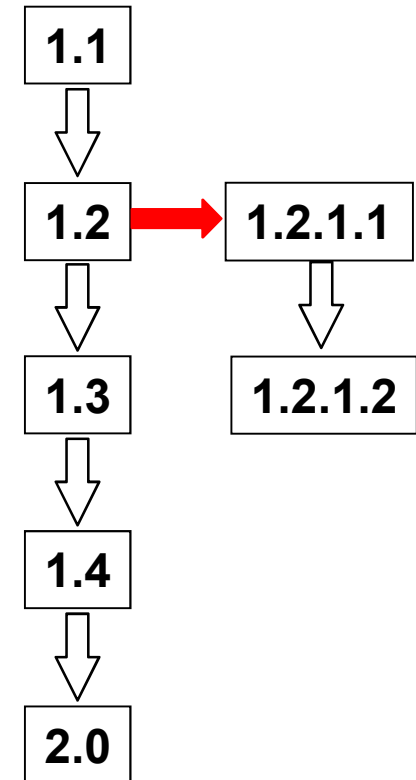
```
rcsdiff -r1.1 -r 1.2 Kreis.c
rcsdiff: RCS/1.2,v: No such file or directory
=====
RCS file: RCS/Kreis.c,v
retrieving revision 1.1
retrieving revision 1.2
diff -r1.1 -r1.2
2c2
<  $Id: Kreis.c,v 1.1 2015/03/07 00:19:19 labuser Exp $
---
>  $Id: Kreis.c,v 1.2 2015/03/07 02:26:17 labuser Exp $
16c16
<  return PI*meinKreis->radius*meinKreis->radius;
---
>  return PI*(meinKreis->radius*meinKreis->radius);
```

Parallele Entwicklung verschiedener Lösungen durch Verzweigungen (Branches)

```
co -l1.1 Kreis.c
ci -l Kreis.c

ci Kreis.c
ci -r1.2.1 Kreis.c
co -l -r1.2.1.1 Kreis.c
#edit Kreis.c
ci -r1.2.1.2 Kreis.c
```

- Damit wird ein Zweig (branch) an der Revision 1.2 erzeugt.
- Diese Version der Datei `Kreis.c` wird als Version 1.2.1.1 gespeichert.
- Auf der Hauptrevision kann kein Branch angelegt werden, dort wird ja sowieso angefügt.
- Aufwendig, erfordert Disziplin



Konfigurationsmanagement

Probleme

- Komplexe Systeme
 - Unterschiedliche Komponenten stehen in vielen unterschiedlichen Beziehungen zueinander
- Unterschiedliche Ausprägungen
 - Komponenten in verschiedenen Versionen
 - Beziehungen in verschiedenen Versionen
 - Verschiedenartige Komposition
- Mehrere Beteiligte
- Zeitliche Parallelität
 - Mehrere Ausprägungen existieren zur selben Zeit

Ziel

Herstellung und Beibehaltung (=Sicherstellung) des ordnungsgemäßen Zustands

Definitionen – DIN EN ISO 10007:2004-12

Konfigurationsmanagement (KM)

- Koordinierte Tätigkeiten zur Leitung und Lenkung der Konfiguration.

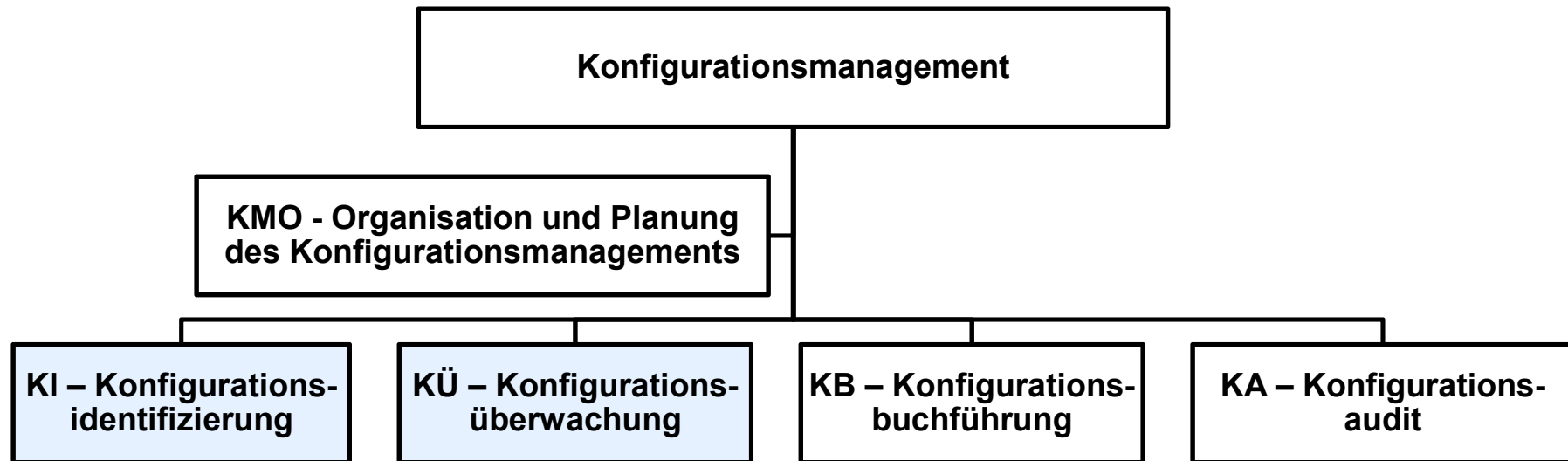
Konfiguration

- Miteinander verbundene funktionale und physische Merkmale eines Produkts, wie sie in den Produktkonfigurationsangaben beschrieben sind.

Produktkonfigurationsangaben

- Anforderungen an Entwicklung, Realisierung, Verifizierung, an Funktionstüchtigkeit im Betrieb und zur Unterstützung des Produkts.

Konfigurationsmanagement - Teilgebiete



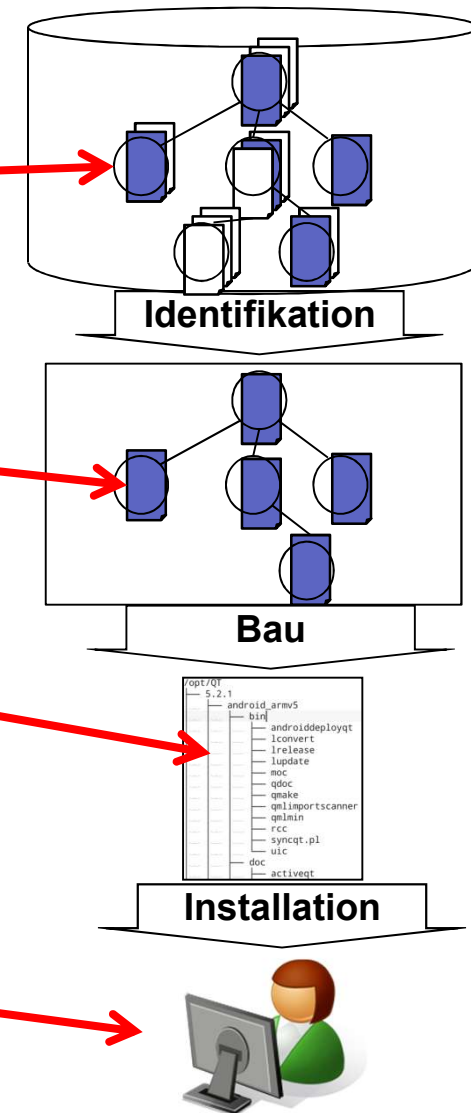
- Ursprünge in den 1950er Jahren

Ziele

- Ziel des KM ist es, den Grad der Erfüllung physischer und funktionaler Anforderungen an eine Konfigurationseinheit zu dokumentieren und diesbezüglich volle Transparenz herzustellen.
- Diese Transparenz führt zu Planbarkeit, Nachvollziehbarkeit und Reproduzierbarkeit: jedes an einer Konfigurationseinheit beteiligte Person soll die richtige und zutreffende Dokumentation verwenden.

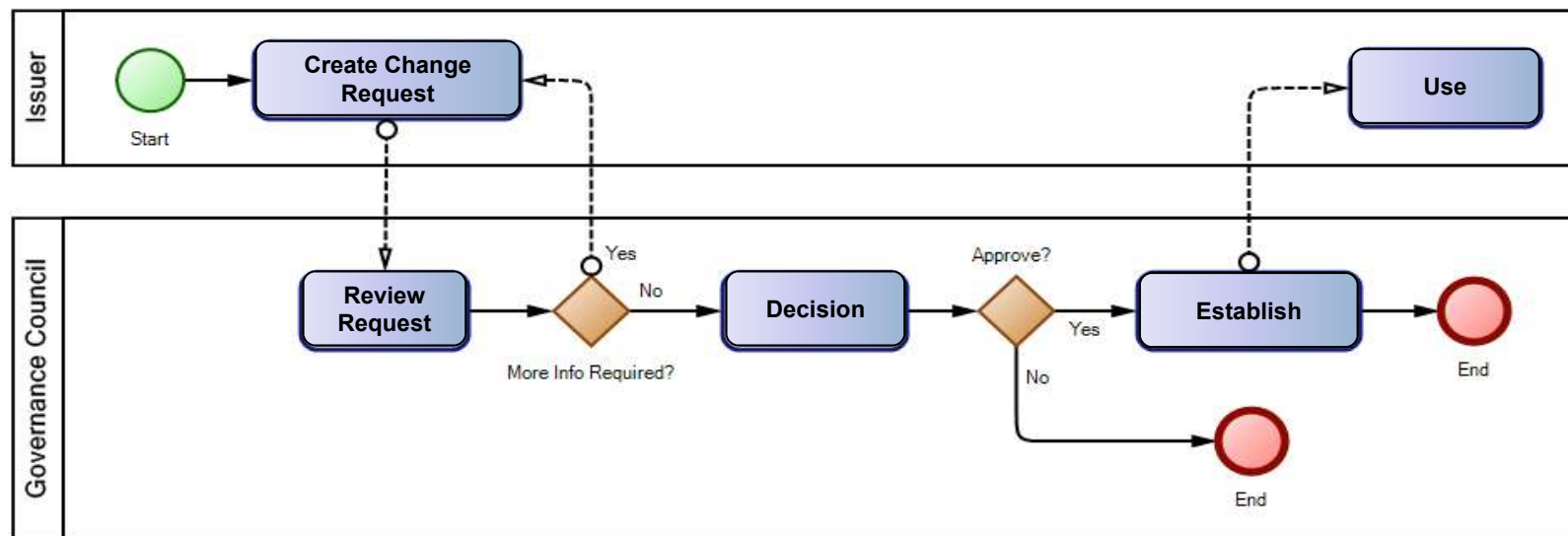
Konfigurationsidentifizierung (KI)

- Auswahl der Konfigurationseinheiten (engl. configuration items)
 - Welche Artefakte sind relevant?
- Dokumentation der Merkmale der Konfigurationseinheiten
 - Produktkonfigurationsangaben
- Definition der Produktstruktur
 - Wie werden die Artefakte verwaltet?
- System zur Bezeichnung
 - Identifizierbarkeit (Seriennummern, Ablagesystem etc.)



Konfigurationsüberwachung (KÜ)

- Dokumentation und Begründung von Änderungen
 - Was wird warum und wann geändert?
- Beurteilung von Änderungen
 - Welche Auswirkungen hat eine Änderung?
- Freigabe von Änderungen
 - Wer darf was ändern?
- Genehmigung von Ausnahmen



Konfigurationsbuchführung (KB)

- Rückverfolgung von Änderungen
 - Wer hat was wann wo geändert?
- Welchen Qualitätsstand hat eine Konfigurationseinheit?
 - Nebenprodukte (Metainformationen)

Beispiel

- Änderungshistorie

Problem

- Zusammenführung
über alle Konfigurations-
einheiten

Certified Tester
Foundation Level Syllabus



Revision History

Version	Date	Remarks
ISTQB 2011	Effective 1-Apr-2011	Certified Tester Foundation Level Syllabus Maintenance Release – see Appendix E – Release Notes
ISTQB 2010	Effective 30-Mar-2010	Certified Tester Foundation Level Syllabus Maintenance Release – see Appendix E – Release Notes
ISTQB 2007	01-May-2007	Certified Tester Foundation Level Syllabus Maintenance Release
ISTQB 2005	01-July-2005	Certified Tester Foundation Level Syllabus
ASQF V2.2	July-2003	ASQF Syllabus Foundation Level Version 2.2 "Lehrplan Grundlagen des Software-testens"
ISEB V2.0	25-Feb-1999	ISEB Software Testing Foundation Syllabus V2.0 25 February 1999

Konfigurationsaudit (KA)

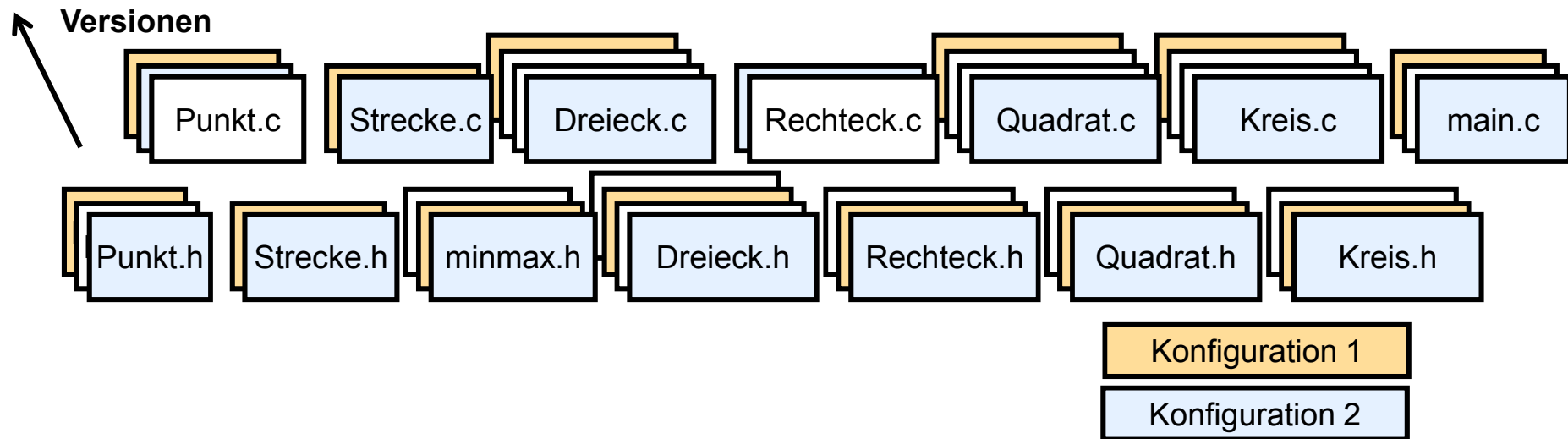
Zu festgelegten Meilensteinen

- Konfigurationsaudit
 - Stimmen die Dokumente formal?
 - Wurde richtig gearbeitet?
- Physisches Konfigurationsaudit
 - Stimmt das Produkt mit der Dokumentation überein?

Organisation und Planung des Konfigurationsmanagements (KMO)

- Festlegungen
 - Konfigurationseinheiten
 - Verantwortlichkeiten
 - Änderungsprozess
 - Bezeichnung von Artefakten
 - Varianten
- Zeitplanung
 - Audits
- Ressourcenplanung

Konfiguration des Beispielprojektes geo



Wie wird die Konfiguration gekennzeichnet?

Wo wird die Information gespeichert?

Mögliche Lösung:

- Markierung mit logischen Namen (Tagging, Baselining)

Tagging mit RCS

```
>rcs -NR1_0_0: RCS/*
RCS file: RCS/Dreieck.c,v
done
...
RCS file: RCS/Strecke.h,v
Done
> rlog Dreieck.c
RCS file: RCS/Dreieck.c,v
Working file: Dreieck.c
head: 1.1
branch:
locks: strict
access list:
symbolic names:
    R1_0_0: 1.1
keyword substitution: kv
total revisions: 1;      selected revisions: 1
description:
-----
revision 1.1
date: 2015/03/07 00:19:19;  author: labuser;  state: Exp;
Initial revision
> co -rR1_0_0 Kreis.c
RCS/Kreis.c,v --> Kreis.c
revision 1.1
done
```

- Erstellt logischen Namen für Head-Release.
- Dieser Name kann bei `co` verwendet werden um die auszucheckende Version zu kennzeichnen.
- Logische Namen können neu zugewiesen werden, aber immer nur mit einer Version verbunden sein.
- Eine Version kann aber mehrere logische Namen erhalten.

RCS

Vorteile

- Schnell eingerichtet
- Mehrbenutzerfähig
- Speicherplatzersparnis
- Einfach, überschaubar
- Automatisierbar
- Erweiterbar
 - z. B. für Berichte

Nachteile

- Korrumpierbar
 - Zugriffsrechte
 - Offenes Dateiformat
- Datei-basiert
- Löschungen und Hinzufügen von Konfigurationseinheiten (Verzeichnisse, Dateien) schwer durchführbar

RCS gilt deshalb als veraltet.

Nachfolger: z.B. subversion (SVN)

Variantenmanagement

Variantenmanagement

In den meisten Versionierungswerkzeugen gibt es kein Variantenkonzept, sondern es werden Konventionen verwendet oder existierende Versionierungskonzepte „missbraucht“.

- Kopien der Revisionsgraphen (separate Repositories)

- o Zugriffspfade

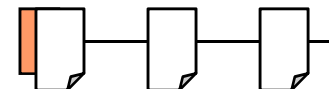
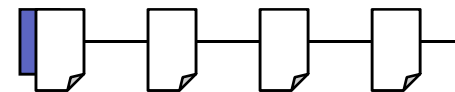
- o Namen

A/bsp.c,v

B/bsp.c,v

bspA.c,v

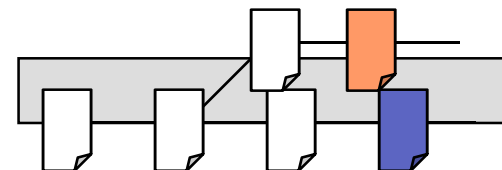
bspB.c,v



- o Seitenrevisionen innerhalb eines Repositories

bspA.c,v 1.3.1.2

bspB.c,v 2.1

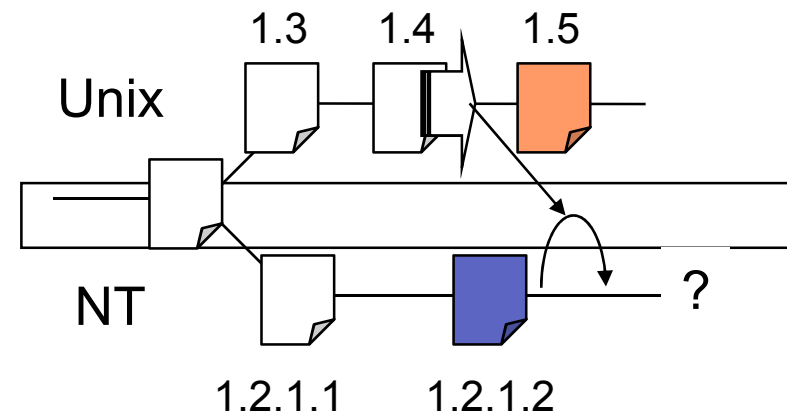


Propagieren von Änderungen

- Gründe
 - Nachziehen von Änderungen, die alle Varianten betreffen
 - Nachziehen von Fehler-Korrekturen (Bug-Fixes) in aktuelleren Revisionen
 - Parallele Weiterentwicklung von Revisionen synchronisieren

```
co -l 1.2.1.2 bsp.c
```

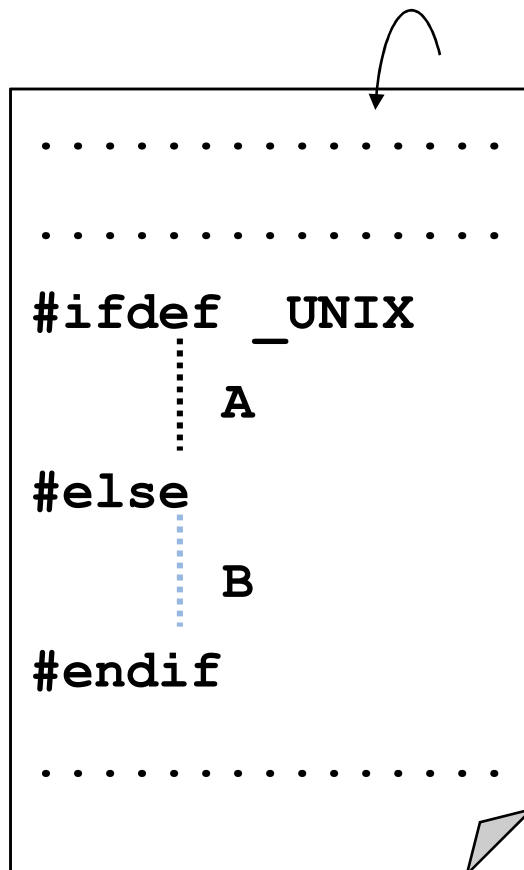
```
racsmerge -r1.4 -r1.5 bsp.c
```



Versionen & Varianten - Orthogonal organisiert

- Bedingte Übersetzung

Separierung der Varianten und der unter einer Abstraktion gemeinsamen Anteile



```
#define _UNIX  
oder  
cc -D_UNIX ...
```

Konzepte orthogonal:

- Propagieren unnötig
- keine verteilte Entwicklung

