Softwaretechnik C - Softwaremanagement

LE 10: Konfigurationsmanagement

Fachhochschule Dortmund
University of Applied Sciences

Organisatorisches

- Vorlesungsfrei 22.12.2022 – 04.01.2023
- Vorlesung LE 11 am 10.01.2023
  - In Präsenz

Agenda

- Konfigurationsmanagement
  - Einleitung
  - Änderungsmanagement
  - Versionskontrolle
  - Systemerstellung
  - Releasemanagement
  - Zusammenfassung

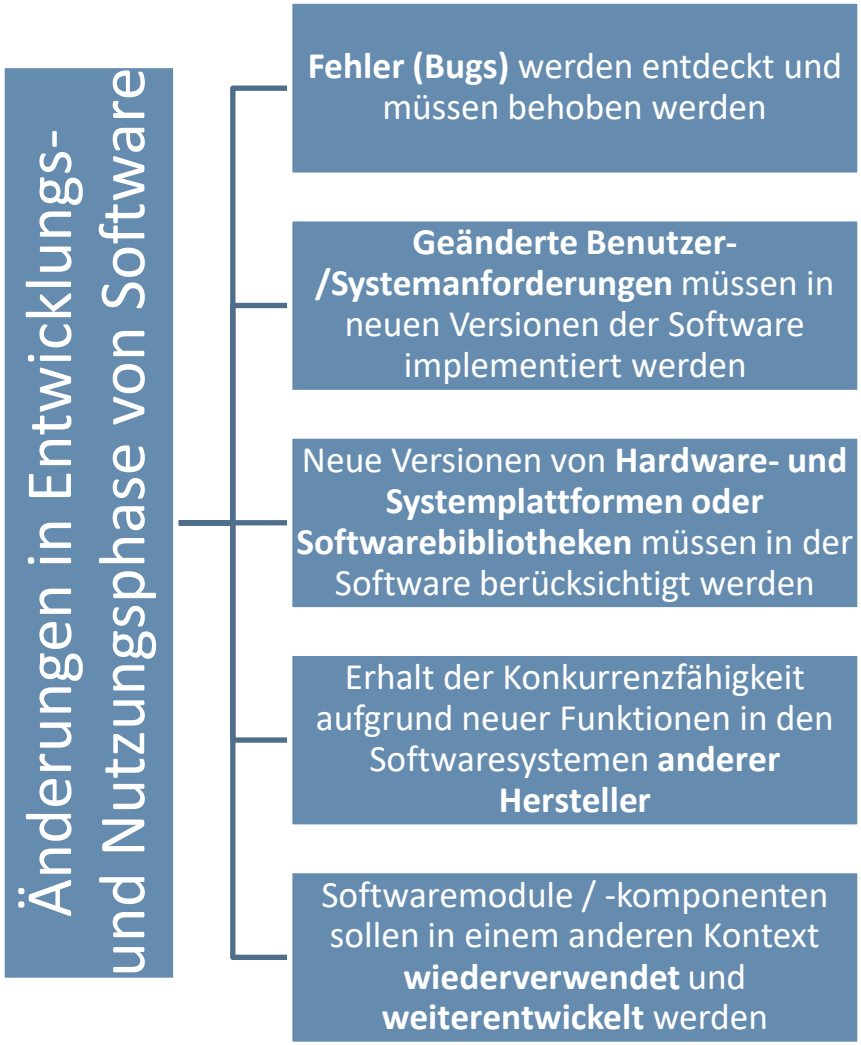
- Konfigurationsmanagement
  - Einleitung
  - Änderungsmanagement
  - Versionskontrolle
  - Systemerstellung
  - Releasemanagement
  - Zusammenfassung

Einleitung

Was führt zu Änderungen von Software (Ursachen, Auslöser, …)?

- …

Einleitung


Änderungen in Entwicklungs- und Nutzungsphase von Software
Fehler (Bugs) werden entdeckt und müssen behoben werden
Geänderte Benutzer-/Systemanforderungen müssen in neuen Versionen der Software implementiert werden
Neue Versionen von **Hardware- und Systemplattformen oder Softwarebibliotheken** müssen in der Software berücksichtigt werden
Erhalt der Konkurrenzfähigkeit aufgrund neuer Funktionen in den Softwaresystemen **anderer Hersteller**
Softwaremodule / -komponenten sollen in einem anderen Kontext **wiederverwendet** und **weiterentwickelt** werden

Einleitung

- Wenn **Änderungen** an einer Software vorgenommen werden, wird eine **neue Version** des Softwaresystems erzeugt
- Die meisten Softwaresysteme sind somit als ein **Satz von Versionen** vorstellbar, die jeweils gewartet, gepflegt und verwaltet werden müssen
- Konfigurationsmanagement befasst sich mit den Richtlinien, Prozessen und Werkzeugen für die **Verwaltung** sich ändernder Softwaresysteme
- Dabei können sich **mehrere Versionen** des gleichen Softwaresystems **gleichzeitig** in der **Entwicklung** und in **produktiver Nutzung** befinden

Historie

- Konfigurationsmanagement wurde ursprünglich von der amerikanischen Raumfahrtindustrie in den 1950er Jahren eingeführt
- Ursache: stetig steigende Produktkomplexität
  - Raumfahrzeuge unterlagen während Entwicklung vielfach undokumentierten Änderungen
  - Raumfahrzeuge wurden im Test oftmals zerstört
  - Hersteller waren selbst bei erfolgreichen Tests nicht in der Lage, das gleiche Modell noch einmal nachzubauen
  - von einer Serienfertigung ganz zu schweigen (Pläne veraltet bzw. Prototyp vernichtet)

Historie

- Als Reaktion auf diese Probleme und den dabei entstandenen Informationsverlust wurden Methoden des Konfigurationsmanagements erarbeitet

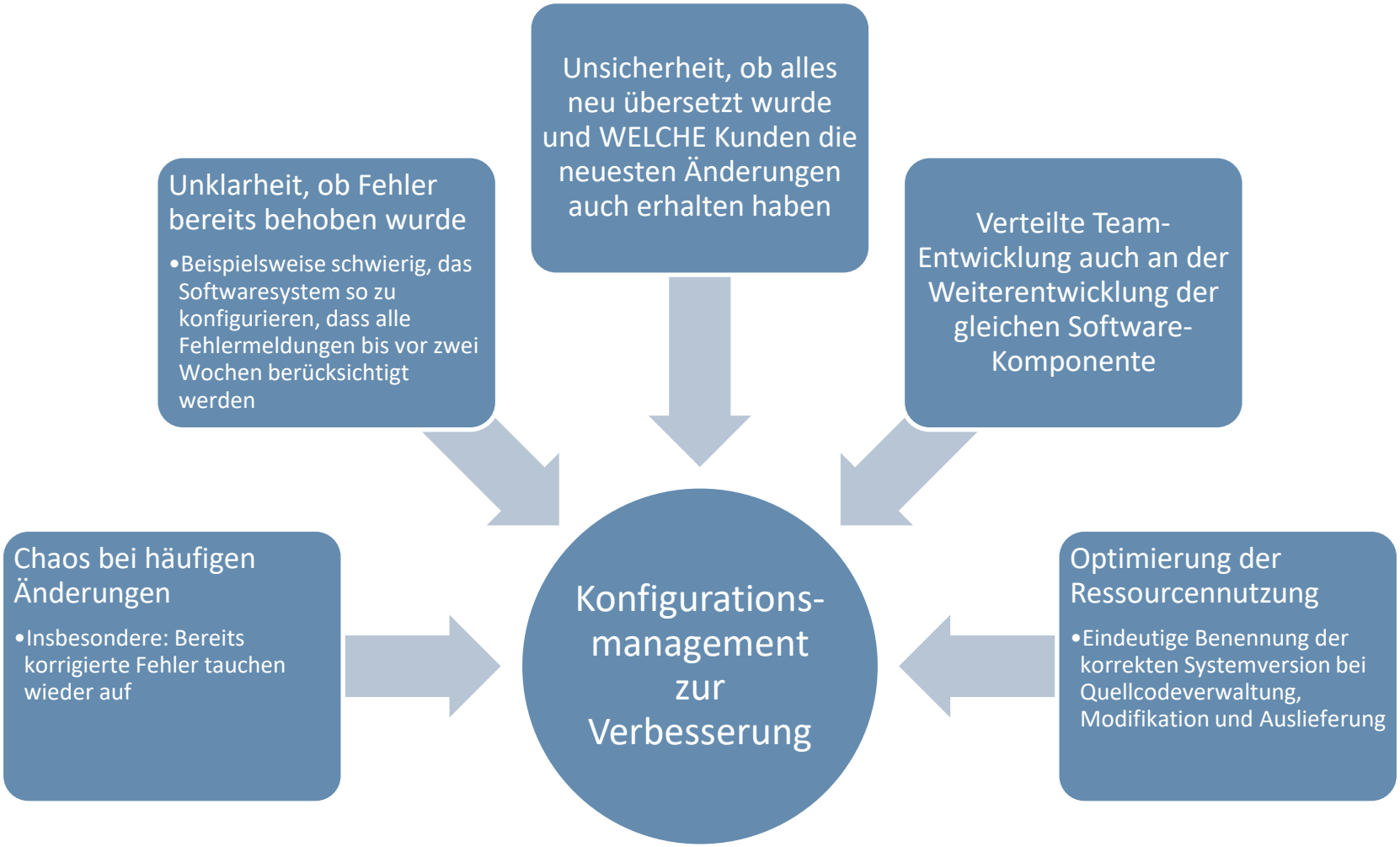
- Erste Arbeitsmittel damals: (Kartei-) Karten, auf denen der Konfigurationsstatus eines Produkts manuell beschrieben wurde
  - Einzelne Bestandteile und jeweilige Versionen der Produkt-Komponenten

Begriffe und Definitionen

Konfiguration	Summe miteinander verbundener funktioneller und physischer Merkmale eines Produkts [zu einem Zeitpunkt / in einer Ausprägung], wie sie in den Produktkonfigurationsangaben beschrieben sind
Produktkonfigurationsbeschreibung	Anforderungen an Entwicklung, Realisierung, Verifizierung, Funktionstüchtigkeit und Unterstützung des Produkts
Konfigurationsmanagement	Koordinierte Tätigkeiten zur Leitung und Lenkung von Konfigurationen

Quelle: ISO 10007:2003

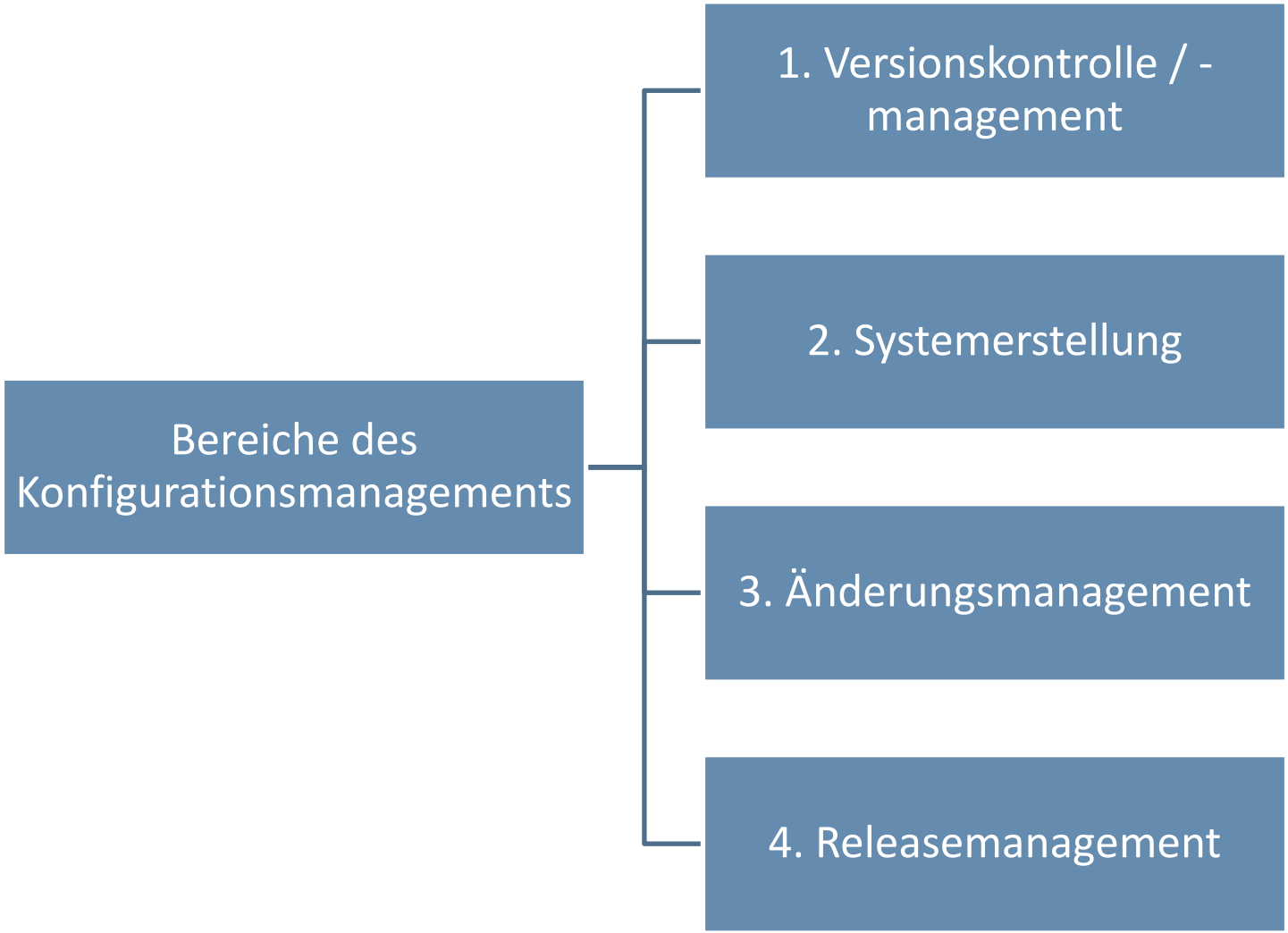
Motivation


Unsicherheit, ob alles neu übersetzt wurde und WELCHE Kunden die neuesten Änderungen auch erhalten haben
Unklarheit, ob Fehler bereits behoben wurde
•Beispielsweise schwierig, das Softwaresystem so zu konfigurieren, dass alle Fehlermeldungen bis vor zwei Wochen berücksichtigt werden
Verteilte Team-Entwicklung auch an der Weiterentwicklung der gleichen Software-Komponente
Chaos bei häufigen Änderungen
•Insbesondere: Bereits korrigierte Fehler tauchen wieder auf
Konfigurations-management zur Verbesserung
Optimierung der Ressourcennutzung
•Eindeutige Benennung der korrekten Systemversion bei Quellcodeverwaltung, Modifikation und Auslieferung

Richtlinien und Verfahren

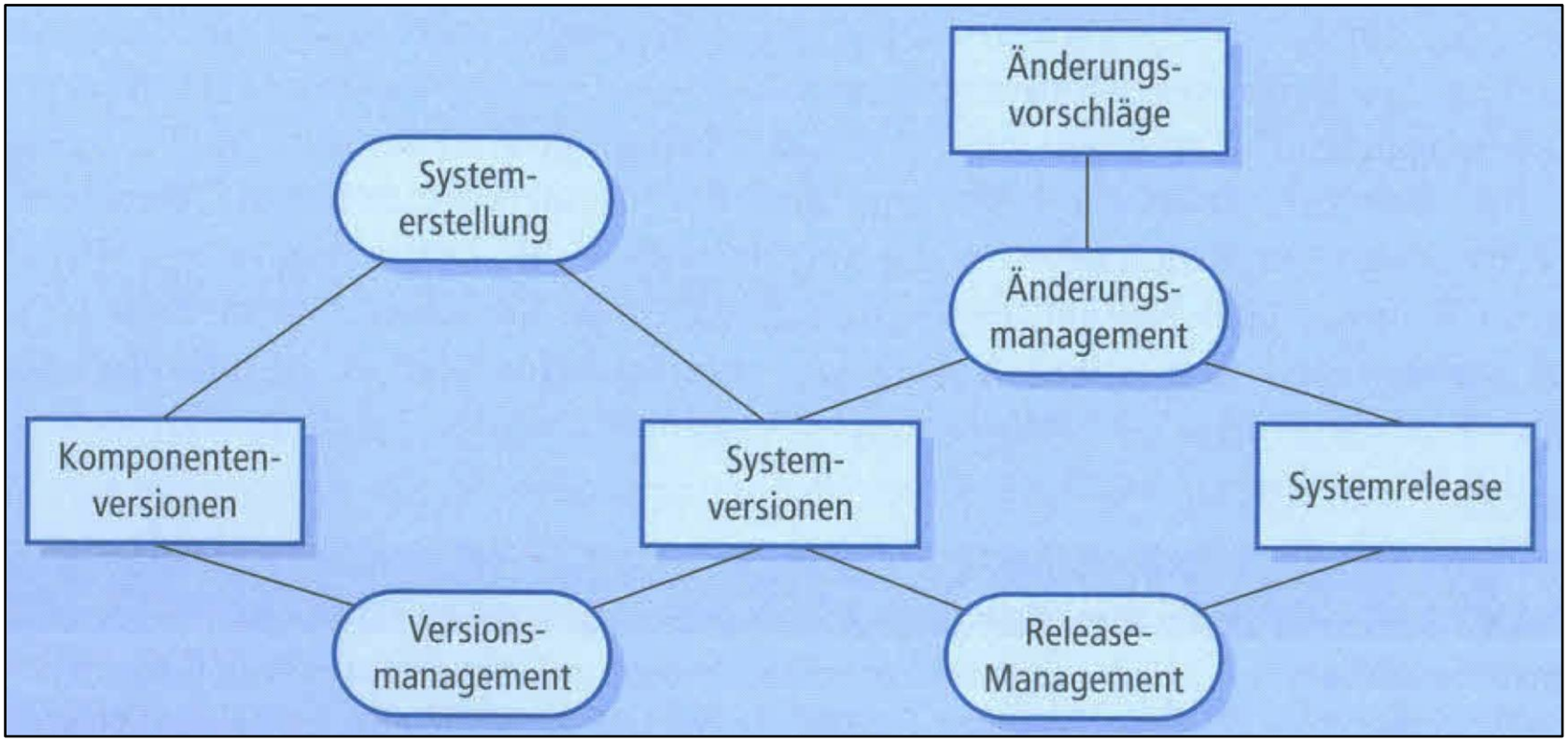
- In den **Richtlinien und Verfahren** des Konfigurationsmanagements wird festgelegt,
  - Wie vorgeschlagene **Systemänderungen aufgezeichnet und bearbeitet** werden
  - Wie **entschieden** wird, **welche** Systemkomponenten zu ändern sind
  - Wie die verschiedenen **Versionen** des Systems und seiner Komponenten **verwaltet** werden und
  - Wie Änderungen **an die Kunden weitergegeben** werden

Bereiche des Konfigurationsmanagements

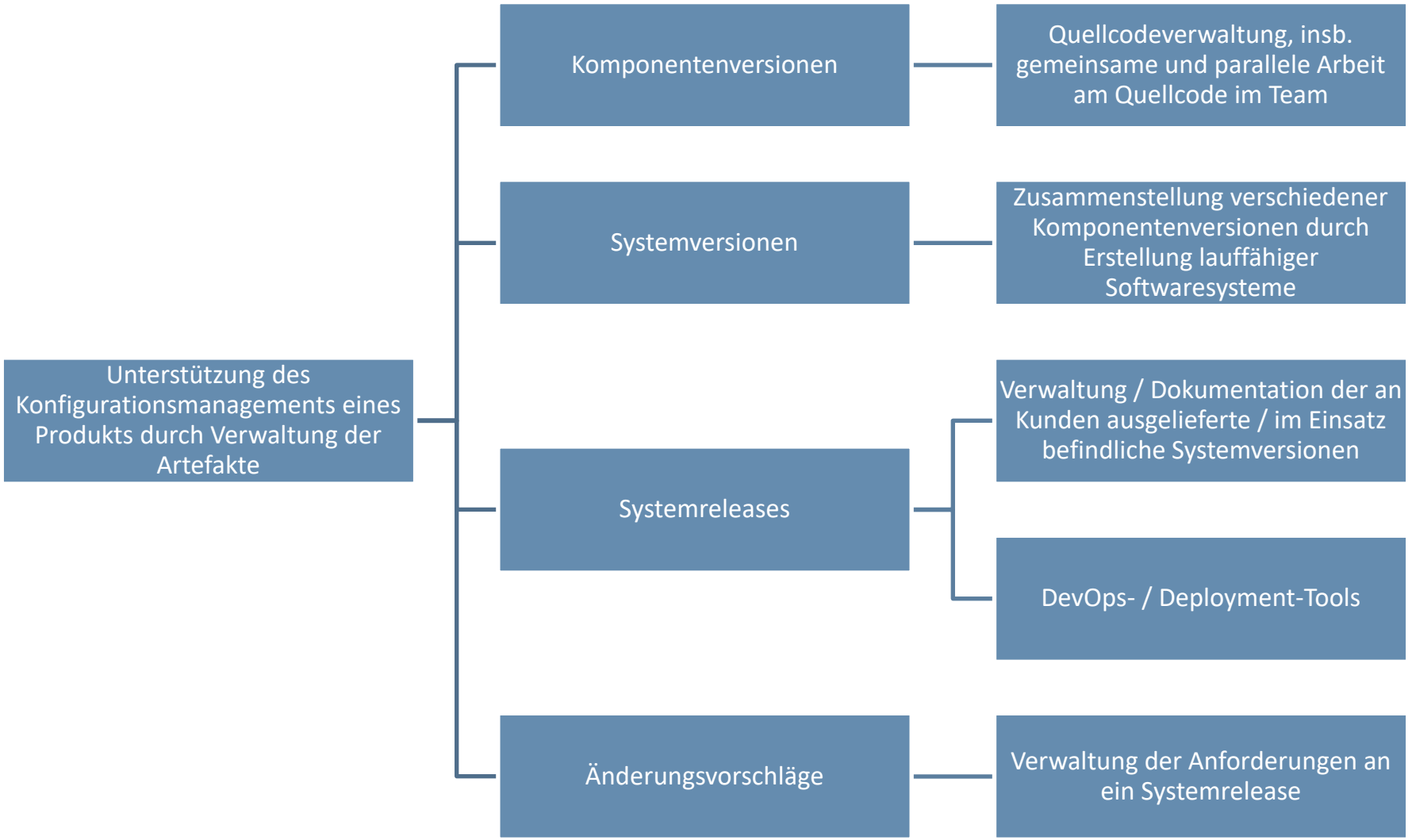


Bereiche und Artefakte des Konfigurationsmanagements

Fachhochschule Dortmund
University of Applied Sciences

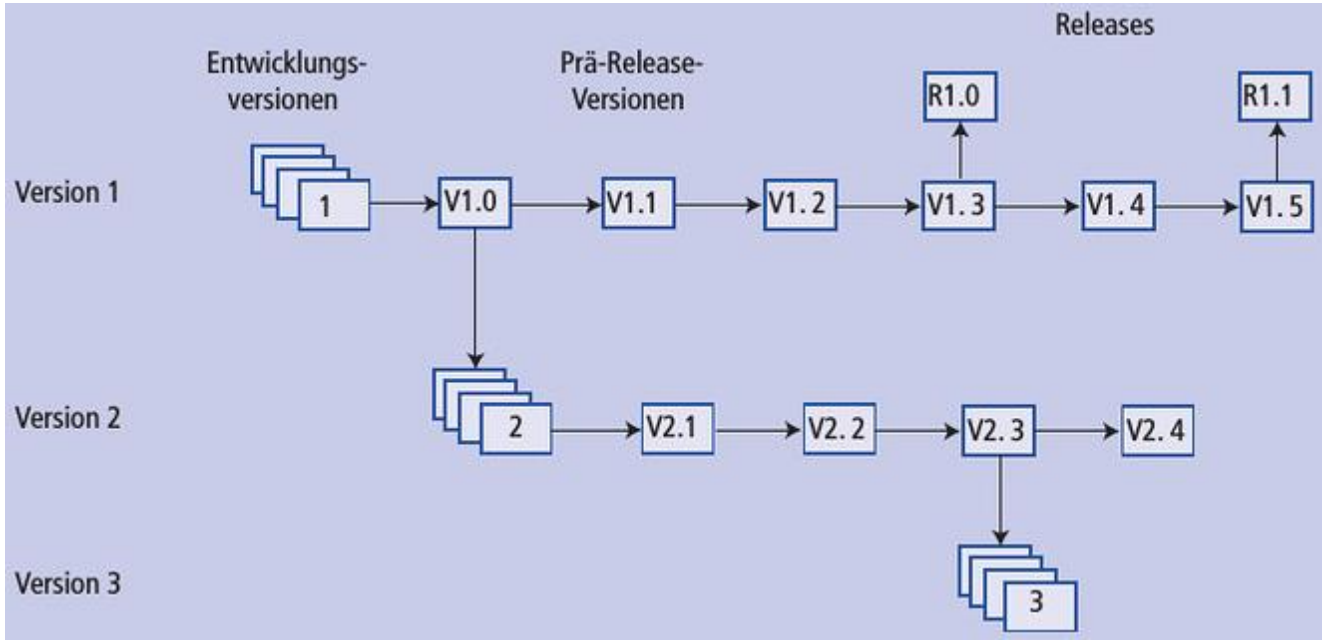


Werkzeuge


Unterstützung des Konfigurationsmanagements eines Produkts durch Verwaltung der Artefakte
Komponentenversionen
Quellcodeverwaltung, insb. gemeinsame und parallele Arbeit am Quellcode im Team
Systemversionen
Zusammenstellung verschiedener Komponentenversionen durch Erstellung lauffähiger Softwaresysteme
Systemreleases
Verwaltung / Dokumentation der an Kunden ausgelieferte / im Einsatz befindliche Systemversionen
DevOps- / Deployment-Tools
Änderungsvorschläge
Verwaltung der Anforderungen an ein Systemrelease

Werkzeuge

- Unterstützung der Entwicklung eines Softwareprodukts den Phasen:
  1. Entwicklungsphase
  2. Systemtestphase
  3. Releasephase



Werkzeuge

- **Bandbreite** der KM-Werkzeuge reicht von **einfachen Werkzeugen**, die **eine einzige Aufgabe** des KMs übernehmen (z.B. Fehlerverfolgung) bis hin zu ganzen Tool-Sammlungen und **Werkzeugketten**, die **sämtliche Aktivitäten** des Konfigurationsmanagements unterstützen
- **Agile Entwicklung**, bei der Komponenten und Systeme mehrmals am Tag geändert werden, wäre **ohne KM-Werkzeuge undenkbar**
- Versionen von Softwarekomponenten werden in einem gemeinsam genutzten **Projekt-Repository** gespeichert, aus dem die Entwickler:innen diese in ihren privaten Arbeitsbereich kopieren
  - Lokale Änderungen können mit dem gemeinsam genutzten Repository synchronisiert und von allen Entwickler:innen genutzt werden

Werkzeuge

- Vielzahl von Software-Werkzeugen verfügbar, u. a.:
  - Trac, Redmine (Open Source)
  - Git, GitLab (Community Edition: Open Source)
  - Jenkins (Open Source)
  - Atlassian-Produkte (u.a. BitBucket, kommerziell)
  - Rational ClearCase von IBM (kommerziell)
  - Rational Synergy von IBM (ehemals Telelogic Synergy, kommerziell)
  - Team Foundation Server von Microsoft (kommerziell)
  - …

Verankerung im Qualitätsmanagement

- Manchmal wird das KM auch als **Teil des Qualitätsmanagements** der Softwareentwicklung angesehen
  - Nach Entwicklung einer neuen Softwareversion wird diese an QM-Team übergeben, das das Software-System auf die Qualitätsanforderungen überprüft
  - Damit wird es zu einem gesteuerten System, was bedeutet, dass:
    - vor der Umsetzung Einigkeit über alle Änderungen hergestellt werden muss und
    - sämtliche Änderungen aufgezeichnet werden müssen

- Verwendung von KM-Standards ist Voraussetzung für eine **Qualitätszertifizierung nach ISO 9000, CMM und CMMI**

- Konfigurationsmanagement
  - Einleitung
  - Änderungsmanagement
  - Versionskontrolle
  - Systemerstellung
  - Releasemanagement
  - Zusammenfassung

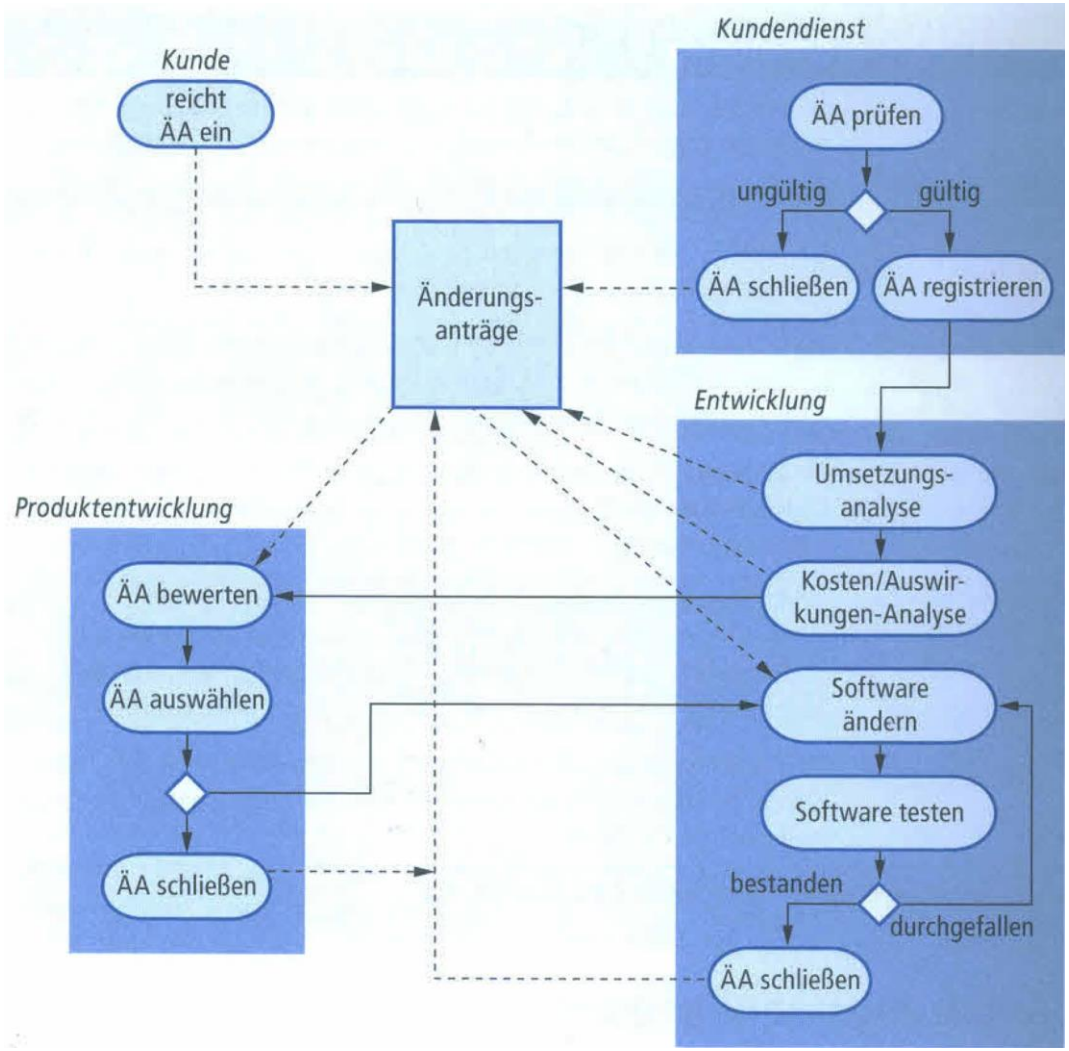
Änderungsmanagement

- Bei großen Softwaresystemen sind **Änderungen an der Tagesordnung**
- Damit Änderungen **kontrolliert und geplant** erfolgen, benötigen Sie einen festgelegten Änderungsmanagementprozess
  - Siehe auch Change Request Prozess in der SWT-C-Vorlesung zu „Anforderungsmanagement“

Änderungsmanagement

- Wird angewendet, **nachdem** ein Softwaresystem für Kunden freigegeben wurde
- Änderungsanträge für **Individualsoftware** unterscheiden sich von denen für **Standardsoftware**
  - Bei **Standardsoftware** kommen Änderungsanträge meist vom Vertrieb, Account Management oder Entwicklungsteam
    - Spiegeln in der Regel die **Kundenrückmeldungen** und -**wünsche** sowie die Analyse von **Konkurrenzprodukten** wider
- **Produktmanagement entscheidet** unter strategischen und ökonomischen Erwägungen, welche Änderungsanträge genehmigt werden

Alternativer CR-Prozess



Änderungsantrag

Formular für Änderungsanträge			
Projekt:	SICSA/AppProcessing	**Nummer:**	23/02
Antragsteller:	I. Sommerville	**Datum:**	20.01.02
Beantragte Änderung:	Der Status der Bewerber (abgelehnt, angenommen usw.) sollte in der Bewerberliste mit angezeigt werden.		
Zuständig für die Änderungsanalyse:	R. Looek	**Datum der Analyse:**	25.01.09
Betroffene Komponenten:	ApplicantListDisplay, StatusUpdater		
Verknüpfte Komponenten:	StudentDatabase		
Beurteilung der Änderung:	Relativ einfach zu implementieren, da nur die Farbe der Anzeige je nach Status geändert werden muss. Es muss eine Tabelle ergänzt werden, die den Status mit den Farben verbindet. Es sind keine Änderungen an den verknüpften Komponenten erforderlich.		
Änderungspriorität:	Mittel		
Implementierung der Änderung:			
Geschätzter Aufwand:	2 Stunden		
Datum der Vorlage beim Genehmigungsteam:	28.01.09	**Datum der Entscheidung:**	30.01.09
Entscheidung:	Änderung akzeptieren. Änderung in Release 1.2 implementieren.		
Zuständig für die Implementierung:			
Datum der Vorlage bei der QS:		**Datum der Änderung:**	
Datum der Vorlage beim KM:		**Entscheidung der QS:**	
Kommentare:			

Änderungsantrag

- Folgende **wichtige Faktoren** sollten bei der Entscheidung berücksichtigt werden, ob ein Änderungsantrag genehmigt wird:

1. Mögliche Folgen, wenn die Änderungen **nicht** vorgenommen würden
   - Systemabsturz vs. Farbnuancen bei der Darstellung der GUI
2. **Nutzen** der Änderung
   - Wer profitiert von der Änderung: einer, wenige oder alle Benutzer?
3. **Kosten** der Änderung
   - Umfangreiche und zeitaufwändige Änderungen => hohe Kosten
   - Änderung betrifft zahlreiche Systemkomponenten => hohe Wahrscheinlichkeit neuer eingebauter Fehler => hohe (Folge-)Kosten
4. **Produktreleasezyklus**
   - Wenn gerade eine neue Version an den Kunden ausgeliefert wurde, macht es Sinn, die Implementierung der Änderung bis zum nächsten Release zu verschieben

Änderungsantrag

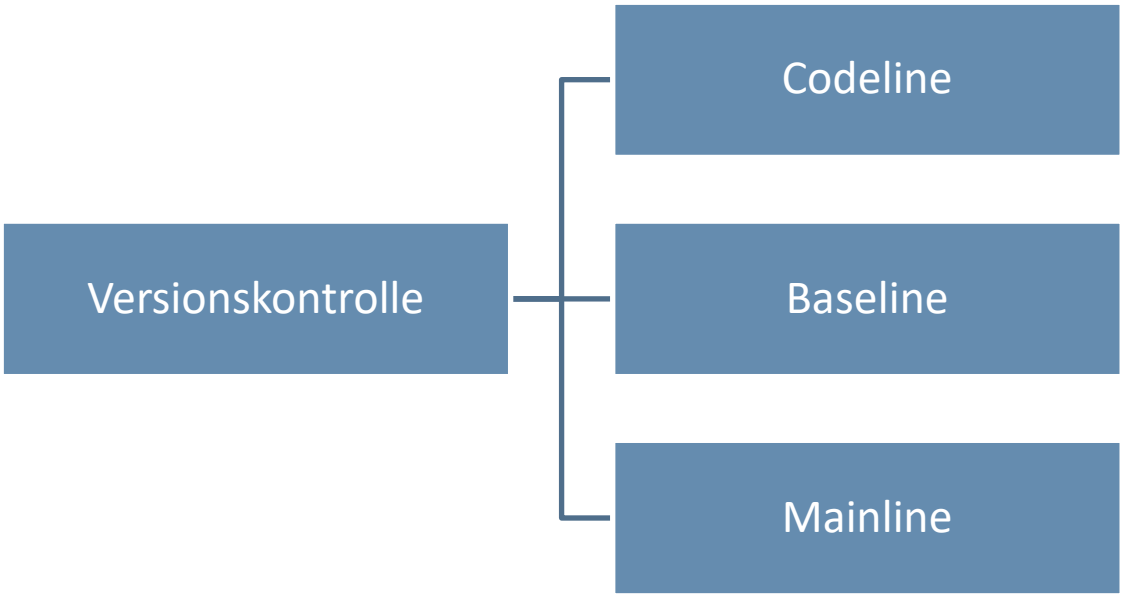
- **Agile Methoden** betonen die Notwendigkeit, den **Kunden** direkt in den Prozess der Änderungs-Priorisierung miteinzubinden
  - Product Owner entscheidet zusammen mit dem Team, welche Änderungen im nächsten Sprint zu implementieren sind
  - Das ist für Individualsoftware effektiv und sinnvoll
  - Bei der Produktentwicklung von Standardsoftware wird dies problematisch, da es keinen konkreten Kunden gibt
  - In solchen Fällen muss das Team selbst Prioritäten festlegen und nachvollziehbar dokumentieren oder Methoden des Crowd-RE nutzen
- Änderungen an Softwaremodulen und –komponenten müssen **dokumentiert** werden
  - Sogenannte **Versionsgeschichte** einer Softwarekomponente
  - Idealfall: Protokoll in Form eines standardisierten Kommentarkopfs zu Beginn des Quellcodes der Komponente (u.a. Verweis auf zugehörigen Änderungsantrag)

- Konfigurationsmanagement
  - Einleitung
  - Änderungsmanagement
  - Versionskontrolle
  - Systemerstellung
  - Releasemanagement
  - Zusammenfassung

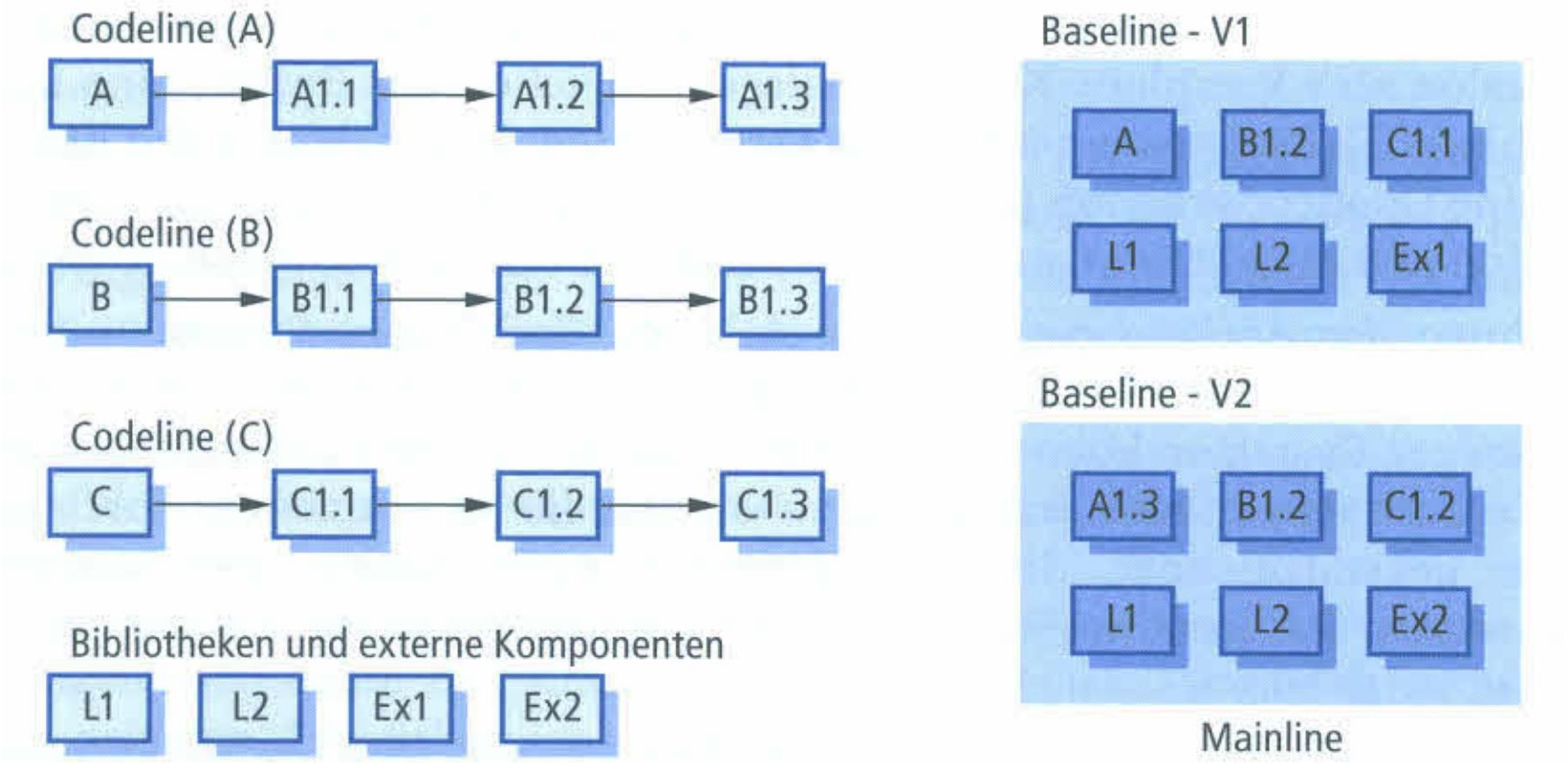
Einleitung

- Versionskontrolle
  - dient der **Verwaltung und Verfolgung**
    - verschiedener Versionen von **Software-Komponenten** und
    - verschiedener Versionen von **Software-Systemen**, in denen diese Komponenten verwendet werden
  - Stellt sicher, dass sich Entwickler:innen, die zeitgleich an den Versionen arbeiten, ihre Änderungen **nicht gegenseitig überschreiben**

Begriffe und Definitionen



Codeline vs. Baseline



Baselines

- Baselines können mit einer **Konfigurationssprache** spezifiziert werden
  - Eine **Baseline-Spezifikation** legt fest, welche Komponenten zu einer bestimmten Version eines Softwaresystems gehören sollen
  - Dabei kann explizit eine Komponentenversion (bspw. „C.1.2“) oder nur den Komponentenbezeichner (bspw. „C“) angeben werden
  - Angabe des Komponentenbezeichners (bspw. „C“) bewirkt Verwendung der jeweils aktuellsten Version der Komponente in der Baseline

- Baselines sind wichtig zur **Erstellung einer speziellen Version eines Gesamtsystems**, z.B. wenn:
  - Kunden individuelle Systemversionen einer Produktlinie bekommen
  - Ausgelieferte SW-Version neu erstellt werden muss, da der Kunde Fehler entdeckt hat

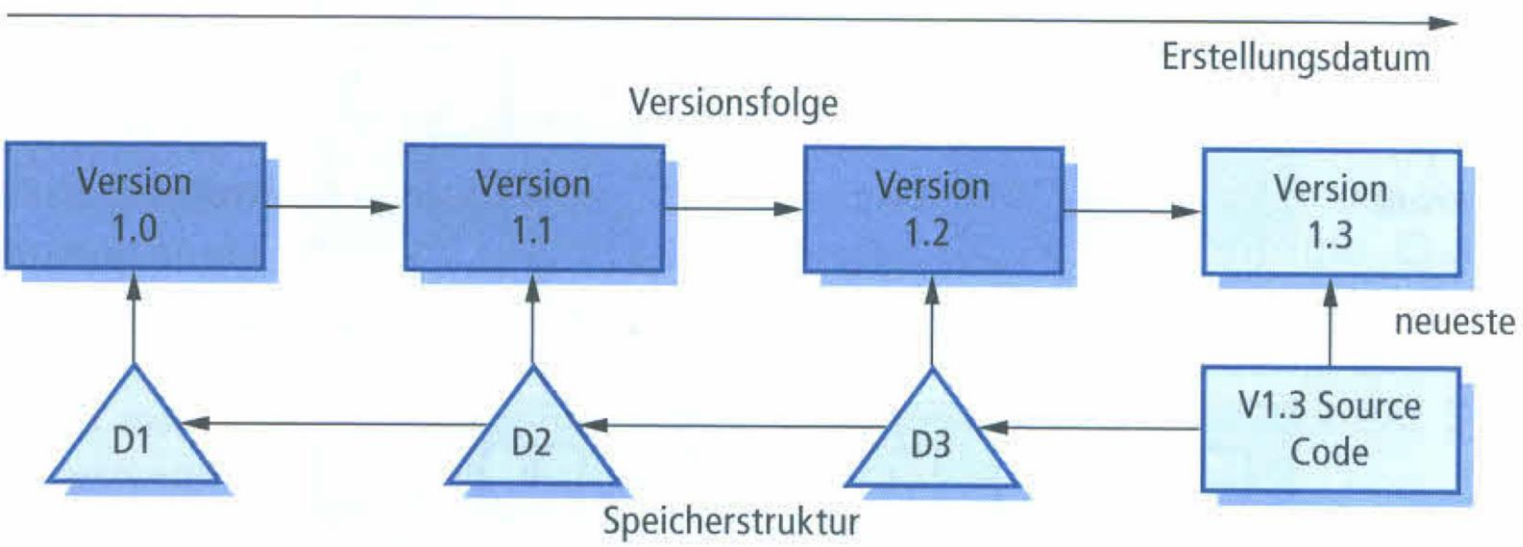
Werkzeugunterstützung

Wichtige Funktionen eines Versionsmanagement-Werkzeugs:

1. **Versions- und Releasebezeichnung**: ButtonManager 1.3 = dritte Version in der Codeline 1
2. **Aufzeichnen der Änderungshistorie**: Protokollierung der vorgenommenen Änderungen an einer Komponente bzw. einem System
3. **Unabhängige Entwicklung**: Eincheck-/Auscheck-Mechanismus, Konfliktvermeidung beim parallelen Bearbeiten und anschließenden Einchecken
4. **Projektunterstützung**: Versionskontrollsysteme unterstützen die Entwicklung verschiedener Projekte, die gleiche Komponenten nutzen. Projektdateien können ein-/ausgecheckt werden.
5. **Speicherverwaltung**: Das Versionskontrollsystem speichert eine Liste der Unterschiede (Deltas) von einer Version zur nächsten => besitzt hohe Priorität

Speicherverwaltung mit Deltas

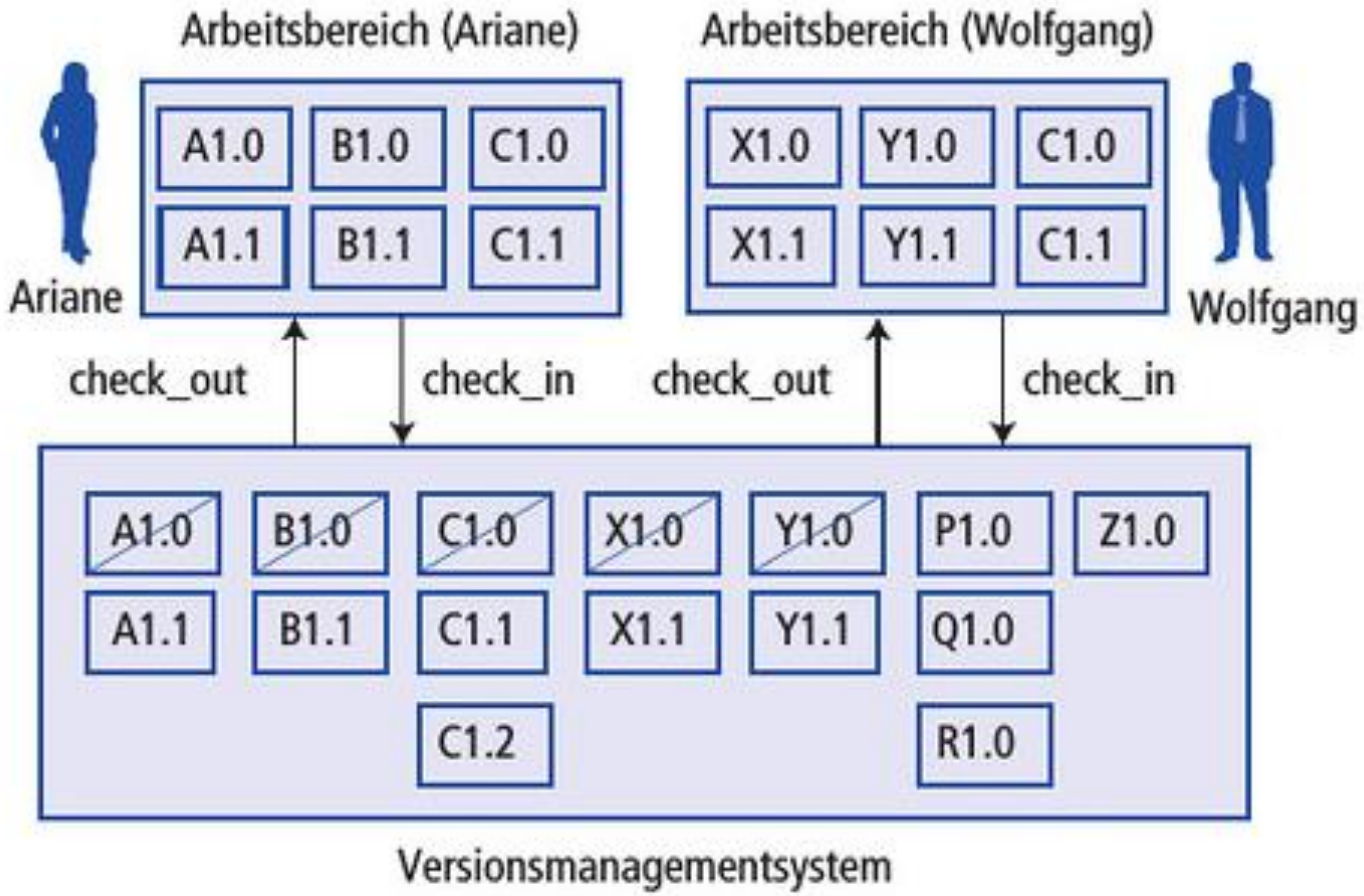
- **Speicherverwaltung** hat hohe Priorität
  - Versionsmanagement- (VM-) System speichert neue Version **vollständig**
  - Zudem wird ein **Delta** (d.h. Liste der Differenzen) zwischen neuer Version und Vorgängerversion, die die Ausgangsbasis war, gespeichert
  - Bsp.: Speicherstruktur einer Komponente in einer bestimmten Version



Unabhängige Softwareentwicklung im Team

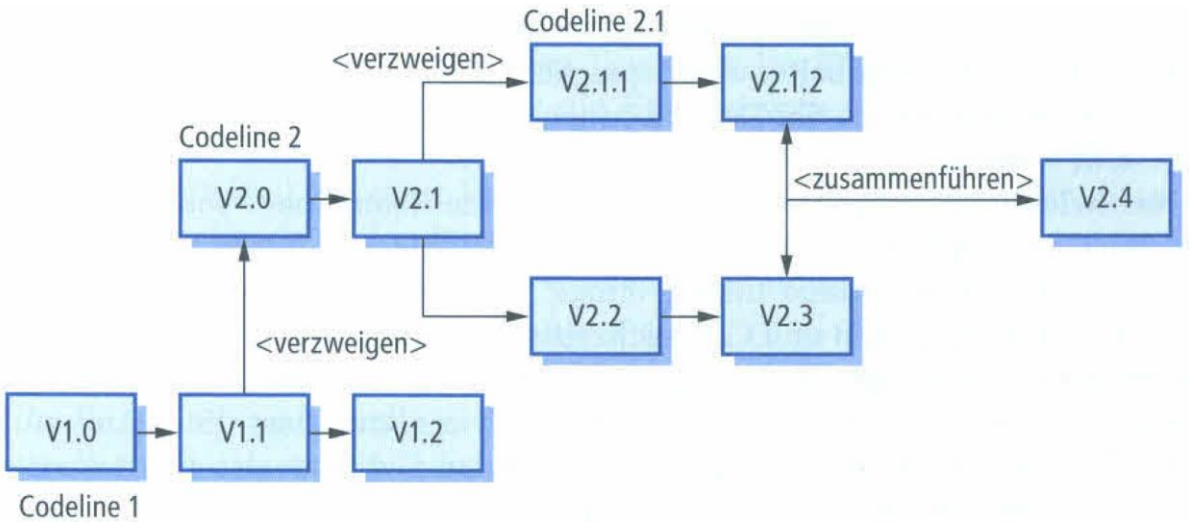
- Zur Unterstützung einer **unabhängigen Entwicklung ohne Konflikte** verwenden VM-Systeme das Konzept eines **öffentlichen Datenarchivs (Repository)** und eines **privaten Arbeitsbereichs**
  1. Entwickler:innen laden Softwarekomponenten aus dem öffentlichen Repository in ihren privaten Arbeitsbereich => auschecken
  2. Nun können sie die Komponenten nach Belieben bearbeiten
  3. Nach Bearbeitung wird Komponente wieder im (öffentl.) Repository eingecheckt
  4. Wenn zwei oder mehr Entwickler:innen gleichzeitig an einer Komponente arbeiten, muss jeder von ihnen die Komponente aus dem Datenarchiv auschecken
  5. VM-System stellt beim Einchecken dieser N geänderten Komponenten sicher, dass verschiedene Versionen auch verschiedene Versionsbezeichner bekommen

Unabhängige Softwareentwicklung im Team



Unabhängige Softwareentwicklung im Team

- **Unabhängige Entwicklung** von Teilen der gleichen Softwarekomponente kann dazu führen, dass **Codelines verzweigen**
- Anstelle einer linearen Folge von Versionen, die Änderungen über einen Zeitraum widerspiegeln, kann es **mehrere unabhängige Zweige (branches)** geben



- Durchaus üblich in der verteilten & teamgestützten Softwareentwicklung

Unabhängige Softwareentwicklung im Team

- Irgendwann kann es notwendig werden, verschiedene **Codeline-Zweige (branches)** wieder **zusammenzuführen**
  - Zur Zusammenführung implementierter Änderungen einer Softwarekomponente aus verschiedenen Verzweigungen zu einer neuen Version eines Zweigs / einer Codeline
  - Wenn die vorgenommenen Änderungen absolut unterschiedliche Teile des Codes betreffen, kann das VM-System die Komponenten-versionen automatisch unter Einbindung der dazugehörigen Deltas zusammenführen
  - Häufig überschneiden sich jedoch die Änderungen, sodass **Konflikte** entstehen
  - In diesem Fall müssen die Änderungen manuell so aufeinander abgestimmt werden, dass sie kompatibel zueinander sind

Werkzeuge

- git, CVS und SVN im Open-Source-Bereich
- Bazaar, git und mercurial als verteilte Sourcecode-Managementsysteme
- Team Foundation Server von Microsoft
- Rational Synergy von IBM
- codeBeamer von Intland Software

- Konfigurationsmanagement
  - Einleitung
  - Änderungsmanagement
  - Versionskontrolle
  - Systemerstellung
  - Releasemanagement
  - Zusammenfassung

Einleitung

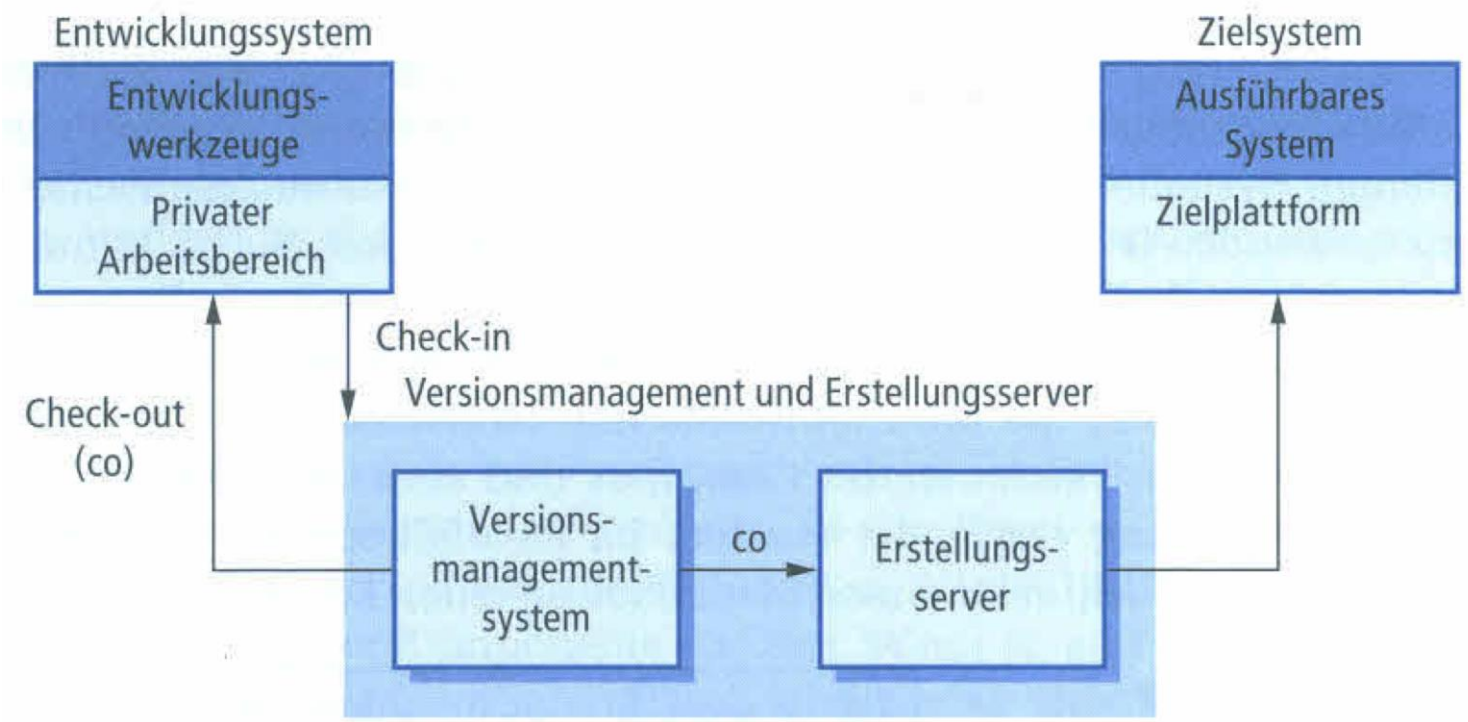
- Bei der Systemerstellung wird durch Kompilieren und Zusammenführen der Softwarekomponenten, externen Bibliotheken, Konfigurationsdateien usw. ein **vollständiges und ausführbares System** erzeugt

- Werkzeuge der Systemerstellung und Versionskontrolle müssen miteinander kommunizieren, da für die Erstellung Komponentenversionen aus dem Repository ausgecheckt werden, die vom VM-System verwaltet werden

- Die Konfigurationsbeschreibung, die verwendet wird, um eine Baseline zu kennzeichnen, wird auch vom Werkzeug der Systemerstellung verwendet

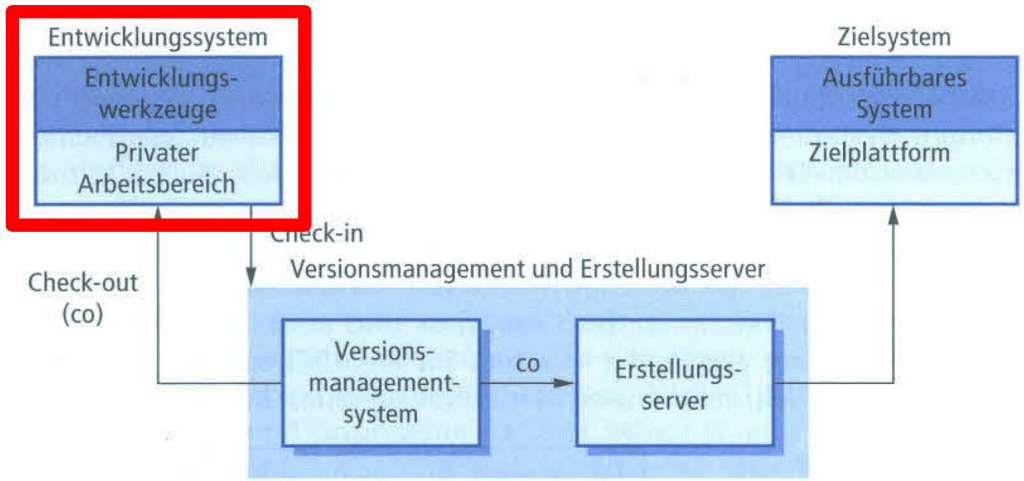
Einleitung

- Oftmals komplexer Prozess, da verschiedene Systemplattformen beteiligt sein können



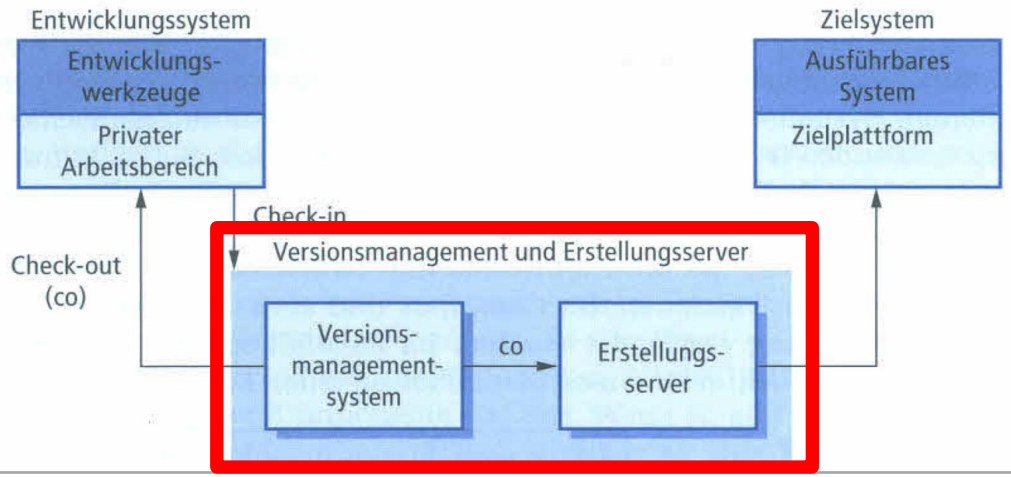
Prozess der Systemerstellung

- Entwicklungssystem
  - Vorgehen: Entwickler:innen checken Code aus dem VM-System aus und nehmen Änderungen im privaten Arbeitsbereich vor
  - I.d.R. wird eine Systemversion im privaten Arbeitsbereich erstellt, um das SW-System in der Entwicklungsumgebung zu testen, bevor die Änderungen in das VM-System hochgeladen werden
  - In diesem Fall müssen für die ausgecheckten Komponentenversionen im privaten Arbeitsbereich lokale Erstellungswerkzeuge verwendet werden



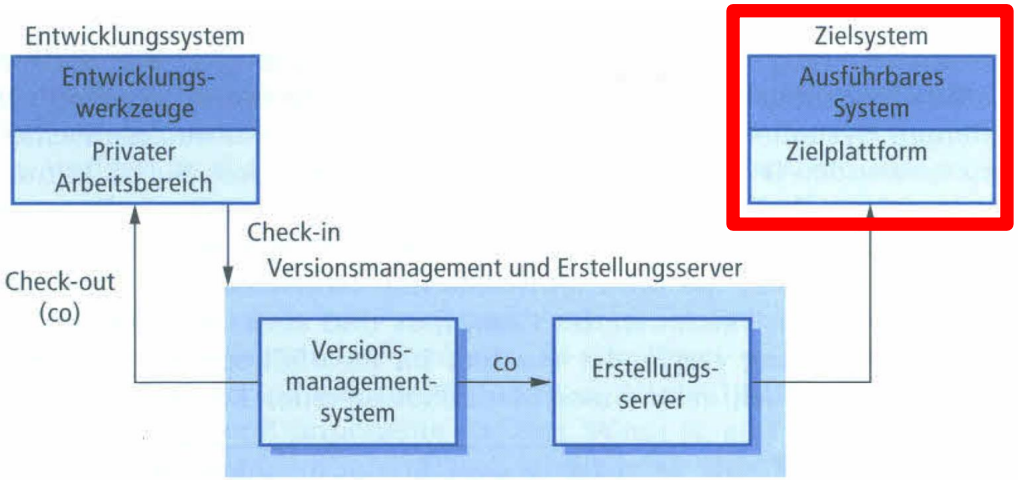
Prozess der Systemerstellung

- Erstellungsserver
  - Mit dem Erstellungsserver wird die endgültige, ausführbare Version des Systems erstellt
  - Erstellungsserver arbeitet sehr eng mit dem Versionskontrollsystem zusammen
  - Entwickler checken den Code in das VM-System ein, bevor der Erstellungsprozess startet
  - Systemerstellung ist hierbei evtl. auf externe Bibliotheken angewiesen, die nicht Teil des Versionskontrollsystems sind



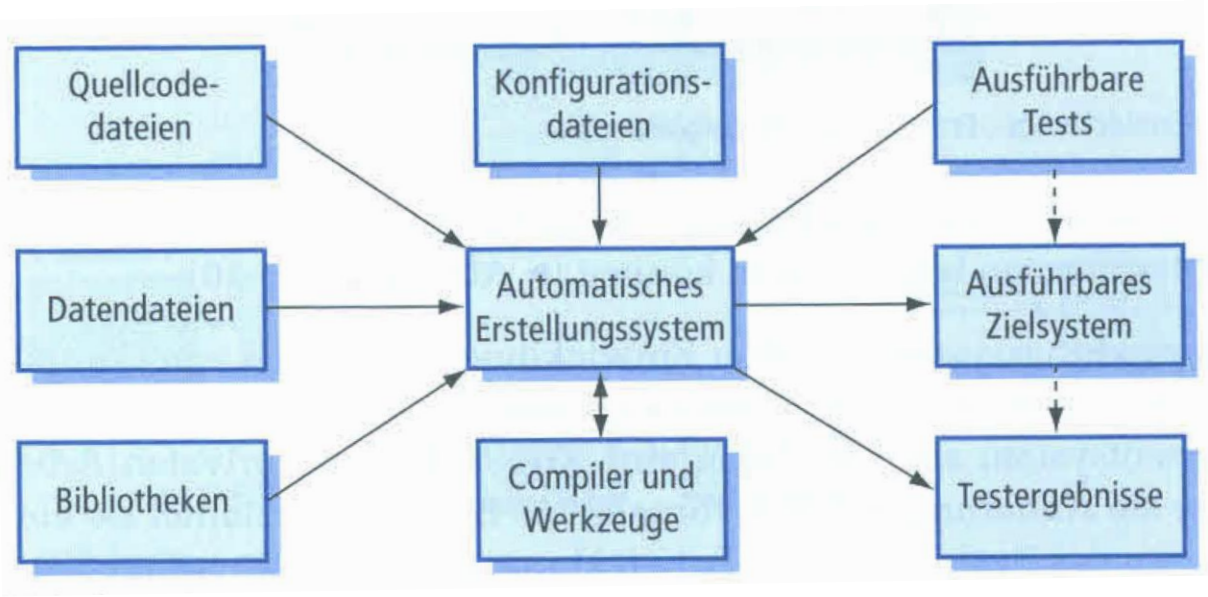
Prozess der Systemerstellung

- Zielsystem
  - Hier wird das kompilierte, zusammengeführte & vollständige System ausgeführt
  - Bei Echtzeitsystemen und eingebetteten Systemen ist die Zielumgebung oftmals kleiner und einfacher als die Entwicklungsumgebung (z.B.: Mobiltelefon)
  - Bei großen, umfangreichen Softwaresystemen können zur Zielplattform noch Datenbanken und andere Systeme gehören, die nicht auf den Entwicklungssystemen installiert werden können



Prozess der Systemerstellung

- **Hohe Komplexität** der Systemerstellung sollte durch ein **automatisches** Erstellungswerkzeug verwaltet werden
  - Außer bei sehr kleinen Softwareprojekten



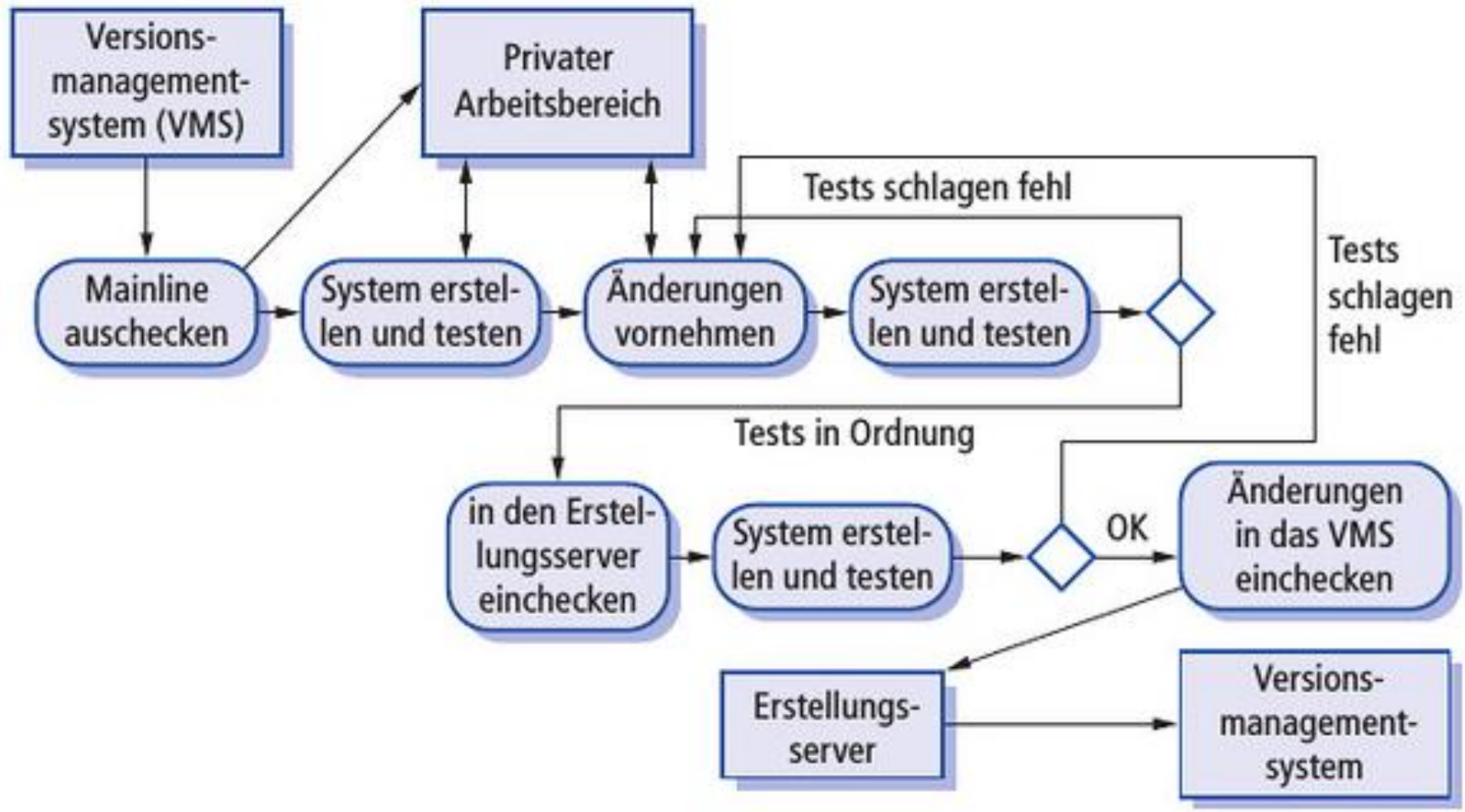
Prozess der Systemerstellung

- Kompilieren des Gesamtsystems kann äußerst **rechenzeitintensiv** sein, u.U. mehrere Stunden
- Werkzeuge zur Systemerstellung sind so ausgelegt, dass sie den Kompilierungsaufwand so gering wie möglich halten

1. Hierbei prüfen sie für jede Komponente, ob sie bereits kompiliert wurde
2. Nur wenn eine Softwarekomponente noch nicht kompiliert wurde, wird sie kompiliert
3. Deshalb muss es eine Möglichkeit geben, den Quellcode einer Softwarekomponente eindeutig mit dem Objektcode zu verknüpfen
4. Das wird über Signaturen erreicht:
   - Jeder Datei, die den Quellcode einer Komponente speichert, wird eine eindeutige Signatur zugewiesen
   - der entsprechende Objektcode erhält die entsprechende Signatur

Prozess der Systemerstellung
Continuous Integration

Fachhochschule Dortmund
University of Applied Sciences



- Konfigurationsmanagement
  - Einleitung
  - Änderungsmanagement
  - Versionskontrolle
  - Systemerstellung
  - Releasemanagement
  - Zusammenfassung

Einleitung

- Ein **Release** eines Softwaresystems ist eine Version eines Softwaresystems, das an den Kunden **ausgeliefert** wird

- Bei größeren Softwaresystemen kann prinzipiell zwischen zwei Release-Arten differenziert werden:
  1. Größere Releases beinhalten wichtige neue Funktionalitäten
  2. Kleinere Releases beheben Softwarefehler und gemeldete Kundenprobleme

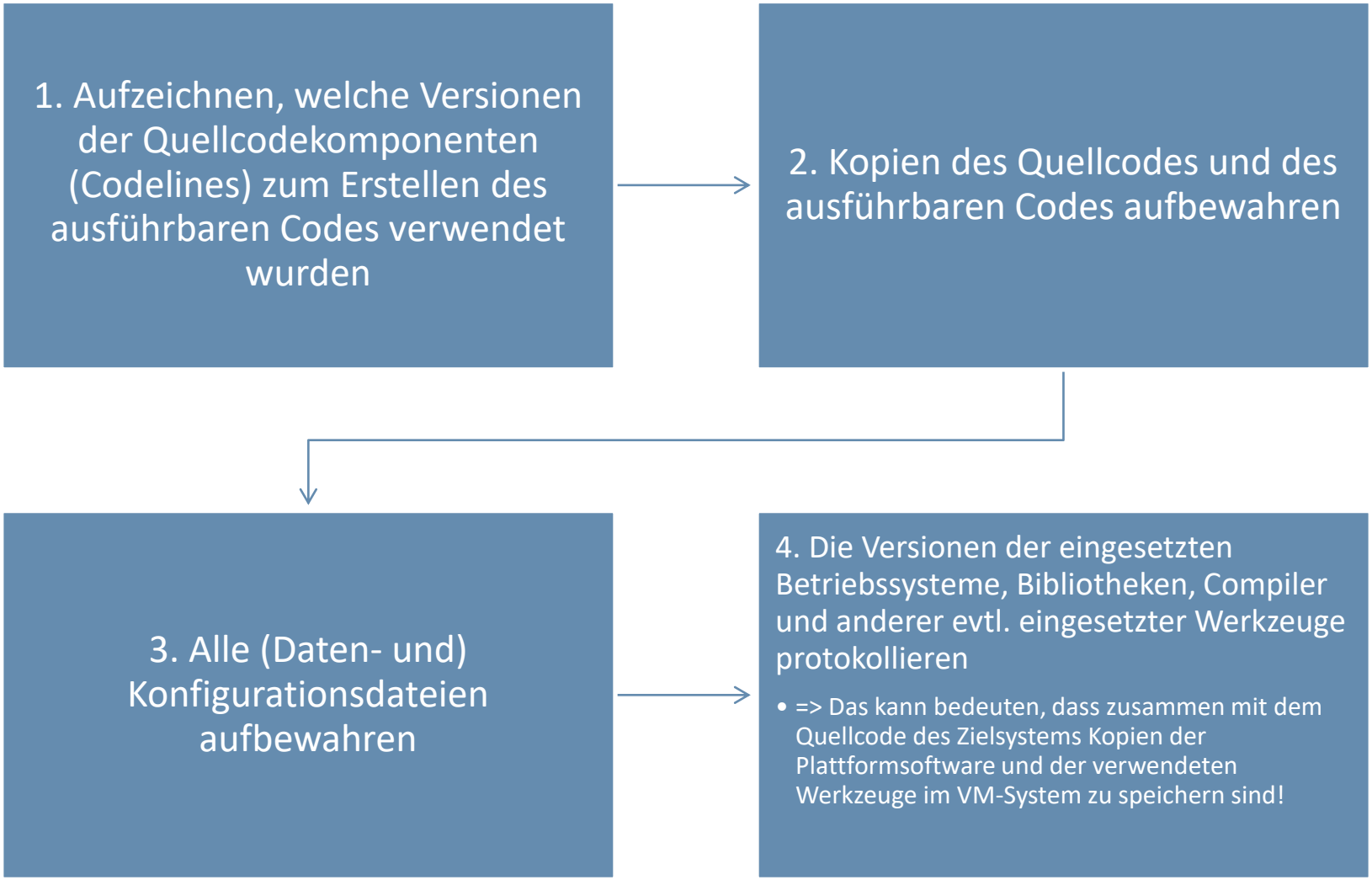
- Große Releases sind häufig kostenpflichtig, während kleinere Releases meist kostenfrei sind (Ökonomische Relevanz eines Release)

Einleitung

- Bei kundenspezifischer Individualsoftware oder Softwareproduktlinien ist die **Verwaltung** der Systemreleases ein **komplexer Prozess**
  - Evtl. muss für **jeden Kunden** ein **spezielles** Release erstellt werden oder einzelne Kunden arbeiten gleichzeitig mit verschiedenen Releases der Software
  - => Softwareunternehmen, die ein spezielles Softwareprodukt vertreiben, müssen oftmals Hunderte verschiedene Releases von einem Softwareprodukt verwalten
  - => KM-Systeme und -Prozesse müssen so ausgelegt sein, dass sie Auskunft darüber geben, welche Kunden mit welchen Releases des Systems arbeiten und welche Beziehungen zwischen den Releases und Systemversionen bestehen

- **Reproduktion** eines bestimmten Release / Software-Konfiguration kann erforderlich sein bspw. zur Fehlerreproduktion oder Weiterentwicklung

Dokumentation eines Releases



Vorbereitung und Verteilung

- Die Releaseerstellung ist ein **komplexer Prozess** und muss geplant werden
  - Ggf. sind auch (personelle) Aufwände im Rahmen des Vorgehensmodells einzuplanen, bspw. Release-Sprint

- Der **Zeitpunkt** einer Releaseerstellung und –bereitstellung muss geplant werden
  - Marketing-Aspekte müssen berücksichtigt werden
  - Auch Kunden sind betroffen und entsprechend zu berücksichtigen, bspw. weil neue Funktionen mit Personalschulungen eingeführt werden müssen

- Organisatorisches
- Qualitätsmanagement
  - Einleitung
  - Änderungsmanagement
  - Versionskontrolle
  - Systemerstellung
  - Releasemanagement
  - Zusammenfassung

Zusammenfassung

- **Konfigurationsmanagement** ist das Management eines sich weiterentwickelnden Softwaresystems

- Zur Wartung eines Softwaresystems wird ein **Konfigurations-Management-Team** zusammengestellt, das sicherstellt, dass
  - die Änderungen auf kontrollierte Weise integriert werden und
  - Protokolle mit Details zu implementierten Änderungen gespeichert werden

- Wichtigste Konfigurationsmanagementprozesse befassen sich mit
  1. Änderungsmanagement
  2. Versionskontrolle
  3. Systemerstellung
  4. Releasemanagement

Zusammenfassung

- Änderungsmanagement
  - Bewertung und Umsetzung von Änderungsanträgen unterschiedlicher Stakeholder
- Versionskontrolle
  - Verfolgung der verschiedenen Versionen der einzelnen Softwarekomponenten bzw. des gesamten Softwaresystems
- Systemerstellung
  - Prozess der Zusammenführung aller Systembestandteile zu einem vollständigen und ausführbaren Softwaresystem auf einer Zielplattform
  - Häufiges Erstellen und Testen als wichtiger Prozess um Fehler und Probleme frühzeitig identifizieren zu können
- Releasemanagement
  - Entscheidung über den Zeitpunkt der Systemreleases, Dokumentation der Systemreleases

Herzlichen Dank für Ihre Aufmerksamkeit !