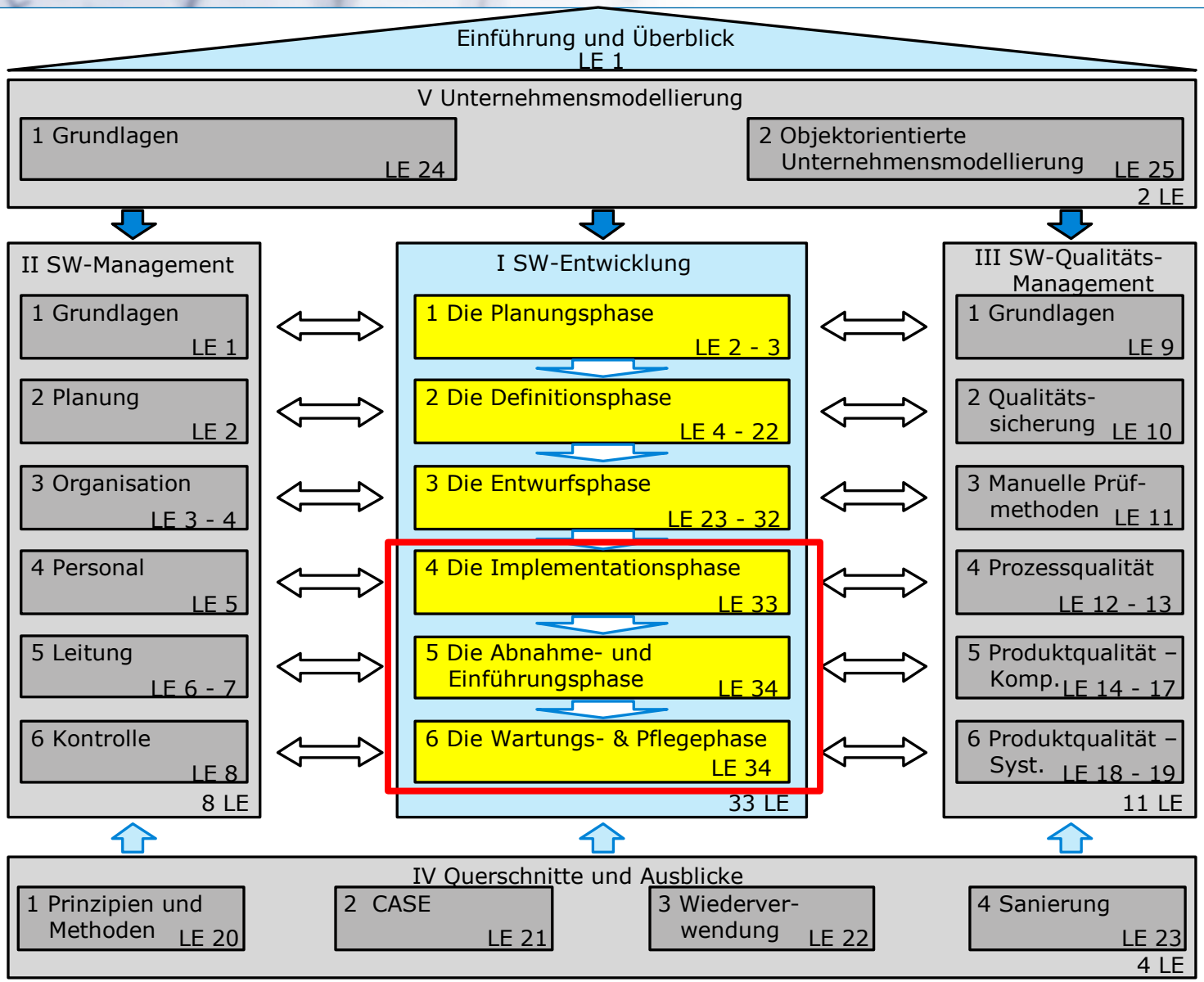
Vorlesung Softwaretechnik
- Implementierungsphase -
- Abnahme- und Einführungsphase–
- Wartungs- und Pflegephase -

Prof. Klaus-Peter Fähnrich

Wintersemester 2008/2009



Legende:

= Übergabe von Teilprodukten

= Informationsaustausch

= Einfluss

= Unterstützung

LE = Lehreinheit

Die in dieser Vorlesung Behandelten Themen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Überblick LE 33

LE 33: Implementierungsphase

- Einführung und Überblick
- Prinzipien der Implementierung
  - Verbalisierung
  - Problemadäquate Datentypen
  - Verfeinerung
  - Integrierte Dokumentation
- Zur Psychologie des Programmierers
  Selbstkontrolliertes Programmieren

LE 34

5. Abnahme- und Einführungsphase

6. Wartungs- und Pflegephase

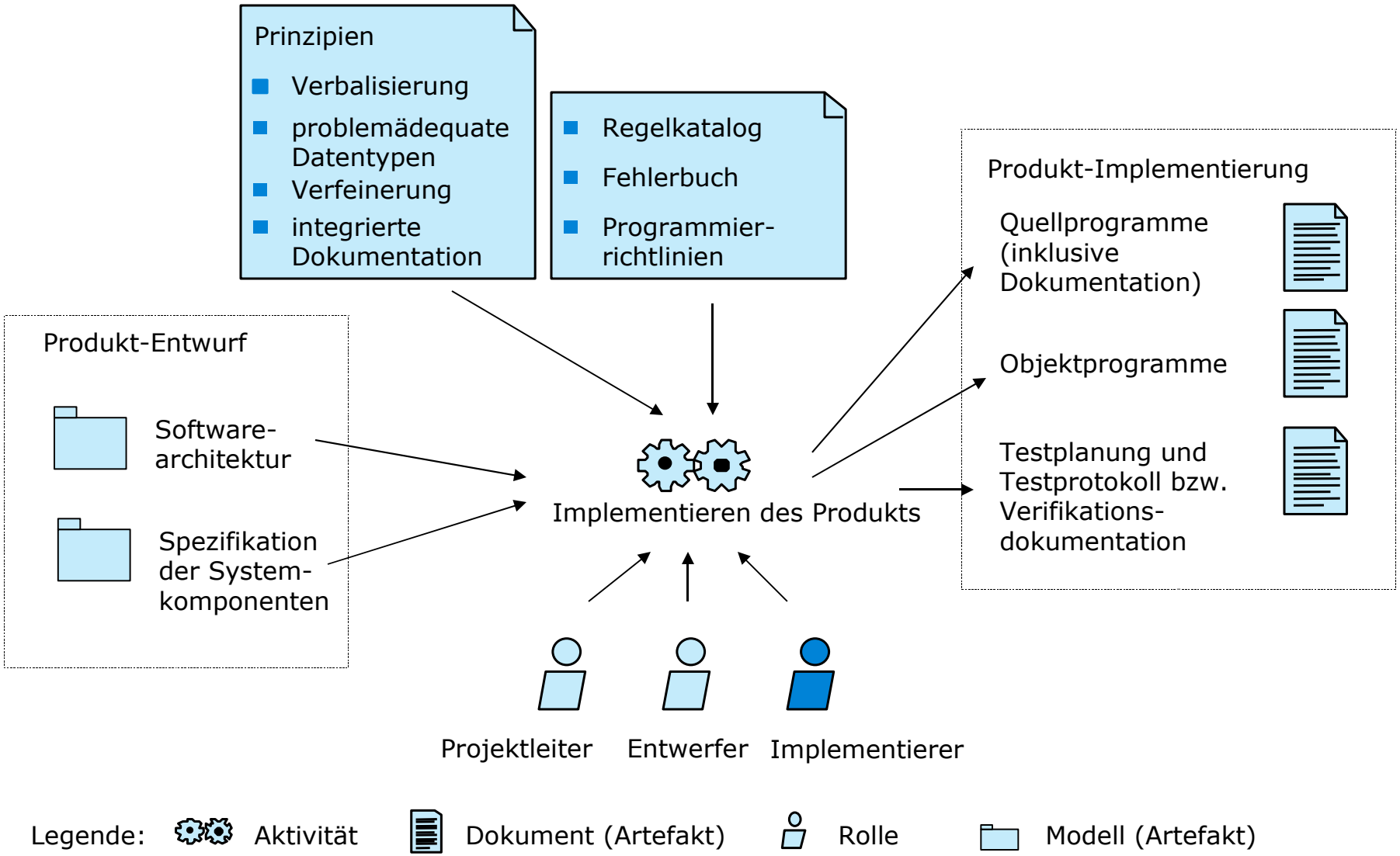
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Einführung

- Aufgabe des Programmierens: Aus vorgegebenen Spezifikationen für eine Systemkomponente die Systemkomponente zu implementieren, d. h. die geforderten Leistungen in Form eines oder mehrerer Programme zu realisieren
- Implementierungsphase
  - Durchführung der Programmiertätigkeiten
  - Eingebettet zwischen der Entwurfsphase und der Abnahme- und Einführungsphase.
- Voraussetzungen
  - In der Entwurfsphase wurde eine Software-Architektur entworfen, die zu geeigneten Systemkomponenten geführt hat.
  - Abhängig von der Entwurfsmethode kann eine Systemkomponente folgendermaßen aussehen:
    - Objektorientierter Entwurf: Klassen
    - Modularer Entwurf: funktionales Modul, Datenobjekt-Modul, Datentyp-Modul
    - Strukturierter Entwurf: funktionales Modul.
- Voraussetzungen
  - Für jede Systemkomponente existiert eine Spezifikation.
  - Die Software-Architektur ist so ausgelegt, dass die Implementierungen umfangsmäßig pro Funktion, Zugriffsoperation bzw. Operation wenige Seiten nicht überschreiten.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Implementierungsprozess



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Einführung

- Aktivitäten
  - Konzeption von Datenstrukturen und Algorithmen
  - Strukturierung des Programms durch geeignete Verfeinerungsebenen
  - Dokumentation der Problemlösung und der Implementierungsentscheidungen
  - Umsetzung der Konzepte in die Konstrukte der verwendeten Programmiersprache
  - Angaben zur Zeit- und Speicherkomplexität
  - Test oder Verifikation des Programms einschl. Testplanung und Testfallerstellung
- Auch »Programmieren im Kleinen« genannt.
- Teilprodukte
  - Quellprogramm einschl. integrierter Dokumentation
  - Objektprogramm
  - Testplanung und Testprotokoll bzw. Verifikationsdokumentation
- Alle Teilprodukte aller Systemkomponenten müssen integriert und einem Systemtest unterzogen werden
- Rollen
  - Implementierer / Programmierer / Algorithmenkonstrukteur.
- Basiskonzepte zur Implementierung der Systemkomponenten:
  - Kontrollstrukturen
  - Entscheidungstabellen
- Bei der Implementierung sollten bestimmte Prinzipien eingehalten werden.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Verbalisierung I

- Gedanken und Vorstellungen in Worten ausdrücken und damit ins Bewusstsein bringen.
- Gute Verbalisierung
  - Aussagekräftige, mnemonische Namensgebung
  - Geeignete Kommentare
  - Selbstdokumentierende Programmiersprache
- Bezeichnerwahl
  - Funktion bzw. Aufgabe zum Ausdruck bringen
  - Keine problemfreie oder technische Bezeichner
  - Verwendung einzelner Buchstaben ungeeignet.
  - Beispiel
    - `Feld1, Feld2, Zaehler` (problemfreie, technische Bezeichner)
      Besser:
    - `Messreihe1, Messreihe2, Anzahlzeichen` (problembezogene Bezeichner)
  - Durch benannte Konstanten: bessere Lesbarkeit
    - Programm außerdem änderungsfreundlicher
  - Kurze Bezeichner nicht aussagekräftig und wegen geringer Redundanz leicht Tippfehler
  - Erhöhte Schreibaufwand durch lange Bezeichner wird durch die Vorteile mehr als ausgeglichen.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Verbalisierung II

- Geeignete Kommentare wählen
  - Den else-Teil jeder Auswahl kommentieren
    - Welche Bedingungen gelten im else-Teil
    - Beispiel

```
if (A < 5) { ...}
else  //A >= 5
{ ...}
```

- Kurzkommentare vermeiden
  - Information besser in Namen unterbringen
  - Beispiel:
    - `I = I + 1;  //I wird um Eins erhöht`
    - `Lagermenge = Lagermenge + 1;`
- Wichtige Kommentare: In Kommentarkasten
- Grad der Kommentierung
  - Abhängig, ob Sprache selbstdokumentierend
  - ADA vorbildlich
  - C++ und Java sind ein schlechtes Beispiel
- Vorteile der Verbalisierung
  - Leichte Einarbeitung in fremde Programme bzw. Wiedereinarbeitung in eigene Programme
  - Erleichtert »code reviews«, Modifikationen und Wartung
  - Verbesserte Lesbarkeit der Programme

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Datentypen

- Daten- und Kontrollstrukturen eines Problems sollen sich in der programmiersprachlichen Lösung möglichst unverfälscht widerspiegeln.
- Programmiersprache sollte folgende Eigenschaften bezogen auf Datenstrukturen besitzen:
  - Umfangreiches Repertoire an Basistypen;
  - Geeignete Typkonstruktoren;
  - Benutzerdefinierbare Typen.
- Aufgabe des Programmierers: Angebot an Konzepten einer Programmiersprache optimal zur problemnahen Lösungsformulierung verwenden.
- Regeln
  - Können die Daten durch Basistypen beschrieben werden, dann ist der geeignete Basistyp auszuwählen.
  - Der Wertebereich sollte so festgelegt werden, dass er möglichst genau das Problem widerspiegelt – unter Umständen durch Einschränkungen des Basistyps.
- Bei OO-Entwicklung
  - Typen von Attributen durch elementare Klassen modellieren, gilt auch für Aufzählungstypen
- Vorteile problemadäquater Datentypen
  - Verständliche, leicht lesbare, selbstdokumentierende und wartbare Programme
  - Statische und dynamische Typprüfungen verbessern die Qualität des jeweiligen Programms
  - Die Daten des Problems werden 1:1 in Datentypen des Programms abgebildet, d. h. Wertebereiche werden weder über- noch unterspezifiziert.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Verfeinerung

- Dient dazu, ein Programm durch Abstraktionsebenen zu strukturieren.
- Verfeinerungsstruktur kann auf 2 Arten im Quellprogramm sichtbar gemacht werden.
  - Die oberste Verfeinerungsebene – bestehend aus abstrakten Daten und abstrakten Anweisungen – ist kompakt beschrieben.
    - Die Realisierung jeder Verfeinerung wird an anderer Stelle beschrieben.
  - Alle Verfeinerungsebenen sind substituiert
    - Die übergeordneten Verfeinerungen werden als Kommentare gekennzeichnet.
- Vorteile
  - Entwicklungsprozess im Quellprogramm dokumentiert
  - Leichtere und schnellere Einarbeitung in ein Programm
    - Details können zunächst übergangen werden
  - Entwicklungsentscheidungen können besser nachvollzogen werden
  - Obere Verfeinerungsschichten können in natürlicher Sprache formuliert werden.
  - Ein Programm wird zweidimensional strukturiert: sowohl durch Kontrollstrukturen als auch durch Verfeinerungsschichten.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Dokumentation

- Dokumentation
  - Integraler Bestandteil jedes Programms
- Angaben
  - Kurzbeschreibung des Programms
  - Verwaltungsinformationen
  - Kommentierung des Quellcodes
- Programmvorspann
  - Programmname: Name, der das Programm möglichst genau beschreibt
  - Aufgabe: Kurzgefasste Beschreibung des Programms einschl. der Angabe, ob ein GUI- , ein Fachkonzept- oder ein Datenhaltungs-Programm bzw. eine entsprechende Klasse (bei OOP) vorliegt.
  - Zeit- und Speicherkomplexität des Programms
  - Name der Programmautoren
  - Versionsnummer Datum (Release-Nummer, Level-Nummer)
- Bearbeitungszustände (V-Modell)
  - **geplant**: Neues Programm enthält die Versionsnummer 1.0 und den Status »geplant«
  - **in Bearbeitung** : Das Programm befindet sich im privaten Entwicklungsbereich des Implementierers oder unter seiner Kontrolle in der Produktbibliothek.
  - **vorgelegt** : Programm ist aus Implementierersicht fertig und wird in die Konfigurationsverwaltung übernommen. Vom Status »vorgelegt« ab führen Modifikationen zu einer Fortschreibung der Versionsangabe
  - **akzeptiert** : Das Programm wurde von der Qualitätssicherung überprüft und freigegeben. Es darf nur innerhalb einer neuen Version geändert werden.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Beispiel: Dokumentation in Java

- Dokumentationskommentar: `/** Kommentar */`
- Vom Java-Programm Javadoc ausgewertet
  - Automatische Dokumentation im HTML-Format
- Spezielle Befehle: Beginn mit @-Zeichen
  - `@author` und `@version`
- HTML-Befehle einfügbar wie `<HR>` und `<B>..<\B>`
- Mit dem HTML-`<A>`-Befehl können Hyperlinks auf andere relevante Dokumente gesetzt werden.
- Beide Befehlsarten müssen am Zeilenanfang beginnen, wobei Leerzeichen und ein Stern überlesen werden.
- Quellcode-Kommentierung
- Wichtig: Geeignete Kommentierung der Operationen einer Klasse
- Neben der Aufgabenbeschreibung jeder Operation ist die Bedeutung der Parameter zu kommentieren, wenn dies aus dem Parameternamen nicht eindeutig ersichtlich ist
  - `@param` Name und Bezeichnung von Parametern
  - `@return` Beschreibung Ergebnisparameter
  - `@exception` Name & Beschreibung von Ausnahmen
  - `@see` Verweis auf andere Klassen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Prinzip der integrierten Dokumentation

- Dokumentation muss integraler Bestandteil der Software-Entwicklung sein:
  - Bei Nachdokumentation am Ende der Codeerstellung sind wichtige Informationen, die während der Entwicklung angefallen sind, oft nicht mehr vorhanden.
  - Entwicklungsentscheidungen (z. B.: Warum wurde welche Alternative gewählt?) müssen dokumentiert werden, um bei Modifikationen und Neuentwicklungen bereits gemachte Erfahrungen auswerten zu können.
  - Aufwand für die Dokumentation wird reduziert, wenn zu dem Zeitpunkt, an dem die Information anfällt von demjenigen, der sie erzeugt oder verarbeitet, auch dokumentiert wird.
- Das Prinzips der integrierten Dokumentation...
  - reduziert den Aufwand zur Dokumentenerstellung
  - stellt sicher, dass keine Informationen verloren gehen
  - garantiert die rechtzeitige Verfügbarkeit der Dokumentation
  - erfordert die entwicklungsbegleitende Dokumentation
- Eine gute Dokumentation bildet die Voraussetzung für
  - leichte Einarbeitung in ein Produkt bei Personalwechsel oder durch neue Mitarbeiter;
  - gute Wartbarkeit des Produkts.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

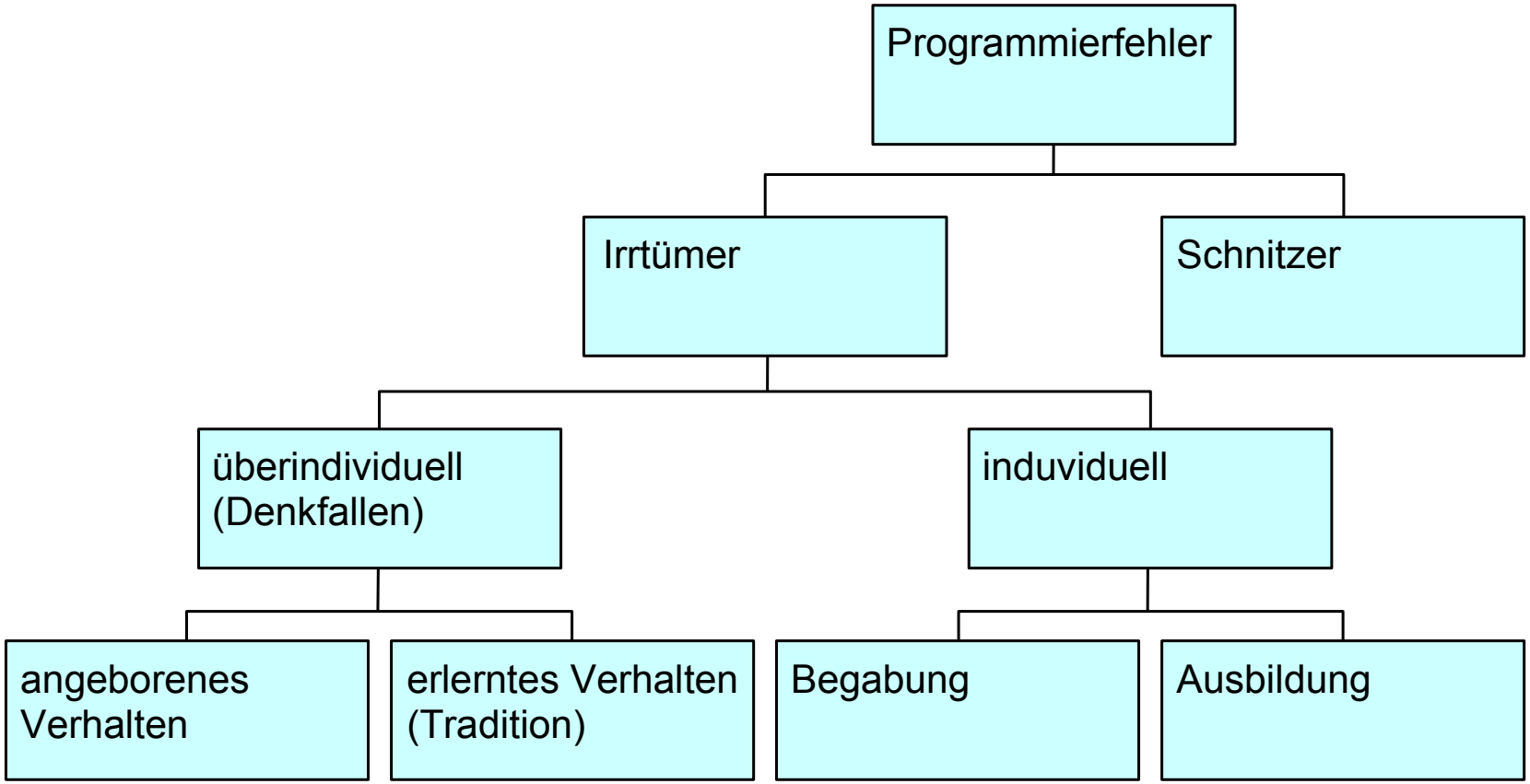
Fehler

- Jeder Programmierer macht Fehler!
  - Viele dieser Fehler sind auf Eigenheiten der menschlichen Wahrnehmung und des menschlichen Denkens zurückzuführen.
  - Denkpsychologie spielt eine Rolle
  - Assoziationen und Einstellungen des Menschen beeinflussen sein Denken.
  - Die meisten Programmierfehler lassen sich auf bestimmte Denkstrukturen und -prinzipien zurückführen.
  - Durch die Beachtung einiger Regeln lassen sich diese Fehler daher vermeiden.

- Hintergrundwissen
  - Jeder Mensch benutzt bei seinen Handlungen bewusst oder unbewusst angeborenes oder erlerntes Hintergrundwissen.
  - Dieses Hintergrundwissen ist Teil des Wissens, das einer Bevölkerungsgruppe, z. B. den Programmierern, gemeinsam ist.
  - Programmierfehler lassen sich nun danach klassifizieren, ob das Hintergrundwissen für die Erledigung der Aufgabe angemessen ist oder nicht.

Klassifizierung von Fehlern

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

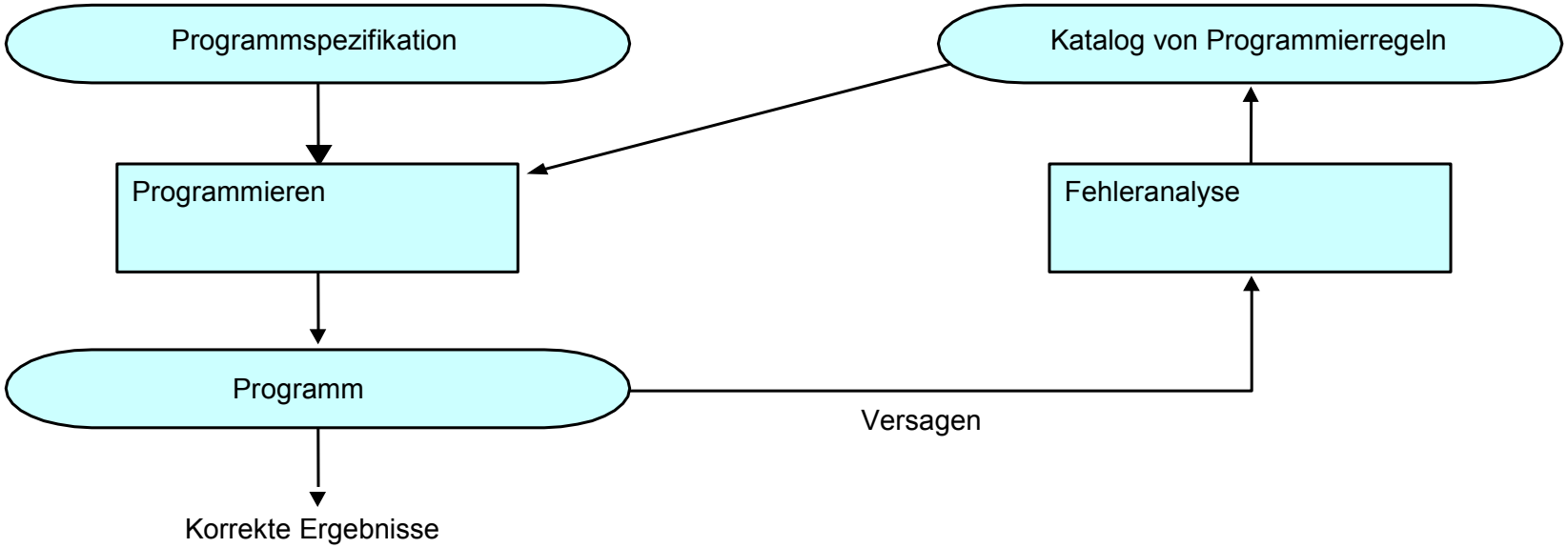
Prinzipien des Verhalten- und Denkmodells

- Übergeordnete Prinzipien
  - Scheinwerferprinzip
  - Sparsamkeits- bzw. Ökonomieprinzip
- Die »angeborenen Lehrmeister«
  - Strukturerwartung
    (Prägnanzprinzip, kategorisches Denken)
  - Kausalitätserwartung
    (lineares Ursache-Wirkungs-Denken)
  - Anlage zur Induktion (Überschätzung bestätigender Informationen)
- Bedingungen des Denkens
  - Assoziationen
  - Einstellungen.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Regelkreis des selbstkontr. Programmierens

- Der erfahrene Programmierer lernt aus seinen Fehlern
  - Er hält sich an Regeln und Vorschriften, die der Vermeidung dieser Fehler dienen.
  - In diesem Sinne ist das Programmieren eine Erfahrungswissenschaft.
  - Jeder Programmierer sollte für sich einen Katalog von Programmierregeln aufstellen und fortlaufend verbessern.
  - Es entsteht ein Regelkreis des selbstkontrollierten Programmierens.



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Regelkreis des selbstkontr. Programmierens

- Das Lernen aus Fehlern ist besonders gut geeignet, um Denkfallen zu vermeiden.
  - Im Laufe des Anpassungsprozesses wird der »Scheinwerfer der Aufmerksamkeit« auf die kritischen Punkte gerichtet.
  - Jeder Programmierer sollte einen Regelkatalog bereithalten und an seine Bedürfnisse anpassen.
  - Es sollte ein Fehlerbuch angelegt werden, das eine Sammlung typischer Fehler enthält.
    - Ein Fehler sollte eingetragen werden, wenn...
      - die Fehlersuche lange gedauert hat
      - die durch den Fehler verursachten Kosten hoch waren oder
      - der Fehler lange unentdeckt geblieben ist.
- Folgende Daten sollten im Fehlerbuch erfasst werden:
  - Laufende Fehlernummer
  - Datum (wann entstanden, wann entdeckt)
  - Programmname mit Versionsangabe
  - Fehlerkurzbeschreibung (Titel)
  - Ursache (Verhaltensmechanismus)
  - Rückverfolgung
    - Gab es schon Fehler derselben Sorte?
    - Warum war eine früher vorgeschlagene Gegenmaßnahme nicht wirksam?
  - Programmierregel, Gegenmaßnahme
  - Ausführliche Fehlerbeschreibung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Heuristiken

- Hat man ein Problem und keine Lösungsidee, dann kann die bewusste Aktivierung von Heuristiken helfen, Denkblockaden aufzubrechen und gewohnte Denkbahnen zu verlassen
- Heuristiken sind Lösungsfindeverfahren, die auf Hypothesen, Analogien oder Erfahrungen aufgebaut sind.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Literatur

- Balzert H., Lehrbuch der Softwaretechnik, 2. Auflage, Spektrum Akademischer Verlag,Heidelberg, 2000.

- Grams T., Denkfallen und Programmierfehler, Springer Verlag, Berlin, 1990

- Kernighan B.W., Plauger P.J., The Elements of Programming Style, 2. Auflage, McGraw-Hill, New York, 1978

- Shneiderman B., Software Psychology – Human Factors in Computer and Information Systems, Wintrop Publishers, 1980

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Überblick LE 34 (1)

LE 33: Implementierungsphase

LE 34

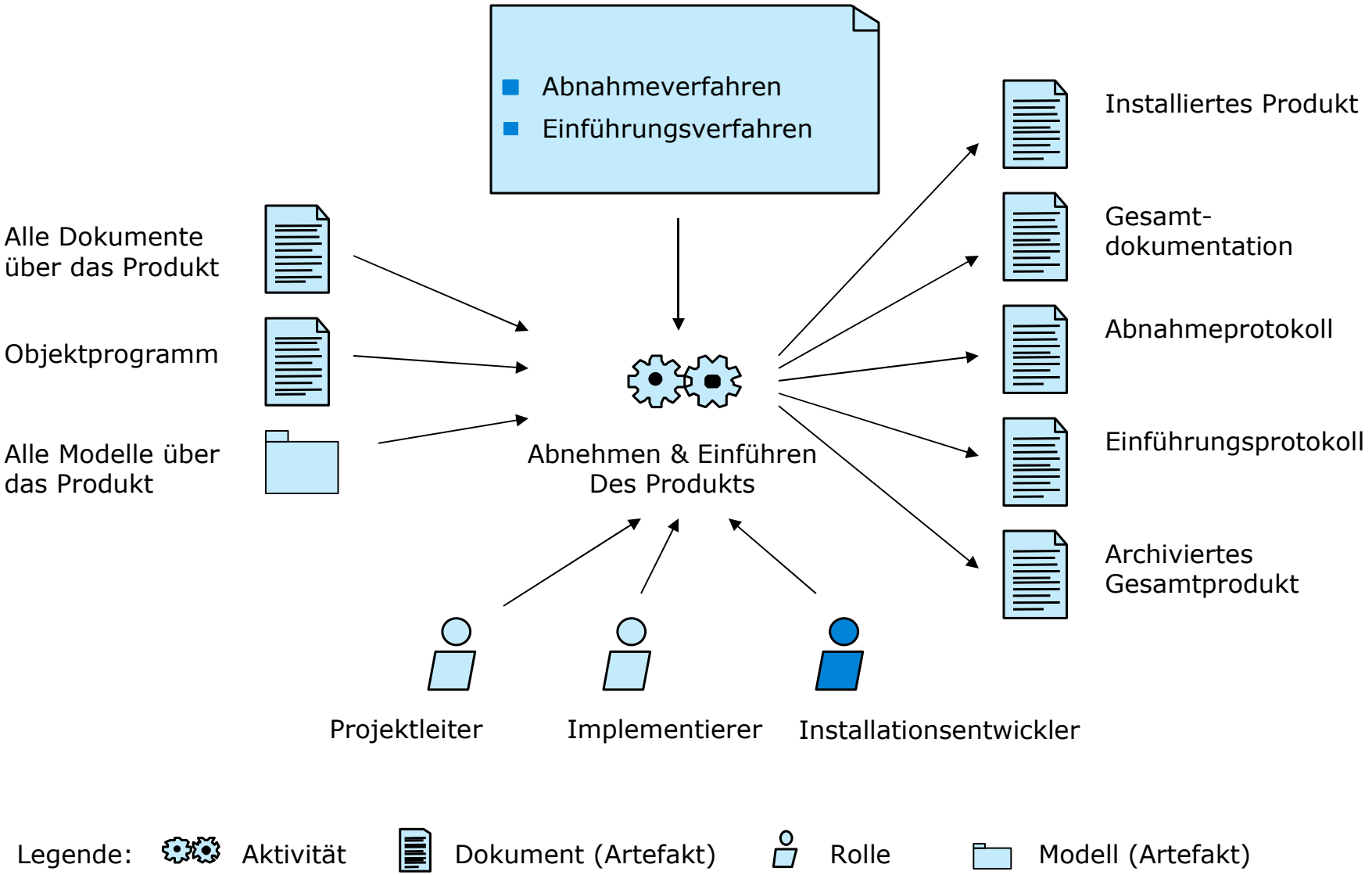
5. Abnahme- und Einführungsphase

- Abnahmephase
- Einführungsphase

6. Wartungs- und Pflegephase

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Abnahme- und Einführungsprozess



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Abnahme

- Abnahme- & Einführungsphase
  - Das fertiggestellte Gesamtprodukt wird abgenommen und beim Anwender eingeführt, d. h. in Betrieb genommen.
  - Ab diesem Zeitpunkt unterliegt das Produkt dann der Wartung & Pflege.
  - Rolle
    - Installationsentwickler (deployment manager).
- Die Abnahmephase: Tätigkeiten
  - Übergabe des Gesamtprodukts einschließlich der gesamten Dokumentation an den Auftraggeber.
  - Mit der Übernahme verbunden ist im Allgemeinem ein Abnahmetest.
  - Innerhalb einer Abnahme-Testserie ist es auch sinnvoll Belastungs- oder Stresstests durchzuführen.
  - Das Ergebnis der Abnahmephase ist ein Abnahmeprotokoll.

Abnahmephase

- Abnahme
  - Nach erfolgreichen Tests durch den Auftraggeber
  - Formale Abnahme: (schriftliche) Erklärung der Annahme (im juristischen Sinne) eines Produkts durch den Auftraggeber
- Externer Auftraggeber
  - Abnahmetest hängt auch davon ab, ob der Auftraggeber das Produkt...
    - nur nutzt, aber nicht wartet und pflegt
    - nutzt und selbst wartet und pflegt
  - Welche Alternative der Auftraggeber wählt, sollte bereits bei der Auftragsvergabe bekannt sein. Die relevanten Qualitätsmerkmale hängen von der gewählten Alternative ab.
- Produktnutzung: Qualitätsmerkmale Usability, Integrity, Efficiency, Correctness und Reliability sind wesentlich
- Wartung & Pflege: Merkmale Maintainability, Testability und Flexibility kommen hinzu
- Abnahmetest
  - Erfüllung der Qualitätsmerkmale prüfen
  - Macht Auftraggeber Wartung & Pflege selbst, dann benötigt er...
    - die gesamte Entwurfs- & Implementierungsdokumentation
    - eine sorgfältige Einführung in die Architektur
- Tätigkeiten
  - Installation des Produkts
  - Schulung der Benutzer und des Betriebspersonals
  - Inbetriebnahme des Produkts

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Einführungsphase

- Tätigkeiten
  - Installation des Produkts: Einrichtung des Produkts in dessen Zielumgebung zum Zwecke des Betriebs.
  - Schulung der Benutzer und des Betriebspersonals: Nach der Installation des Produkts sind die Benutzer in die Handhabung des Produkts einzuweisen.
  - Inbetriebnahme des Produkts: Übergang zwischen Installation und Betrieb
- Einführungsprotokoll
  - Alle Vorkommnisse, die in der Einführungsphase auftreten, werden festgehalten.
- Einführung muss sorgfältig geplant werden.
- Umfangreiche Produkteinführungen
  - Wie Innovationseinführungen zu behandeln. Allgemeine Charakteristika beachten, die bei Innovationseinführungen eine Rolle spielen.
- Umstellung
  - Zeitplanung, u.U. mit Netzplänen
  - Wichtige Aufgabe:
    - Umstellung der Datenbestände
    - Manuelle Karteien müssen oft erst aufbereitet oder zusammengestellt werden, bevor sie für die neue Datenverwaltung erfasst werden können
- Inbetriebnahme auf drei Arten möglich:
  - direkte Umstellung
  - Parallellauf
  - Versuchslauf.

Überblick LE 34 (2)

LE 33: Implementierungsphase

LE 34

5. Abnahme- und Einführungsphase

6. Wartungs- und Pflegephase

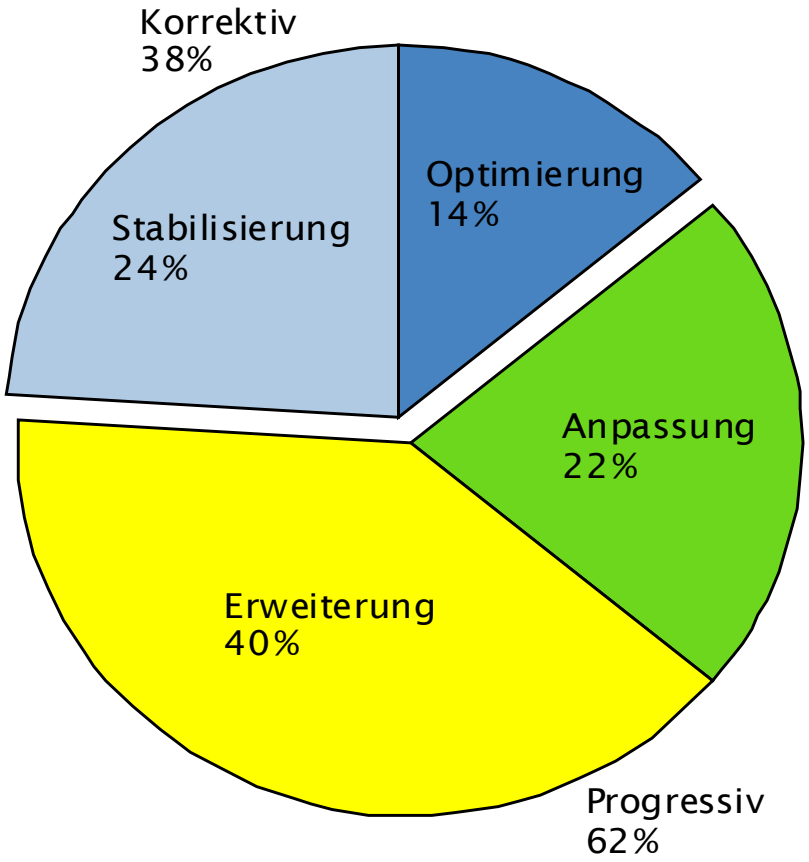
- Aufwandsverteilung
- Vergleich Wartung und Pflege
- Aufwand für ein Produkt
- Verbesserung der Wartung

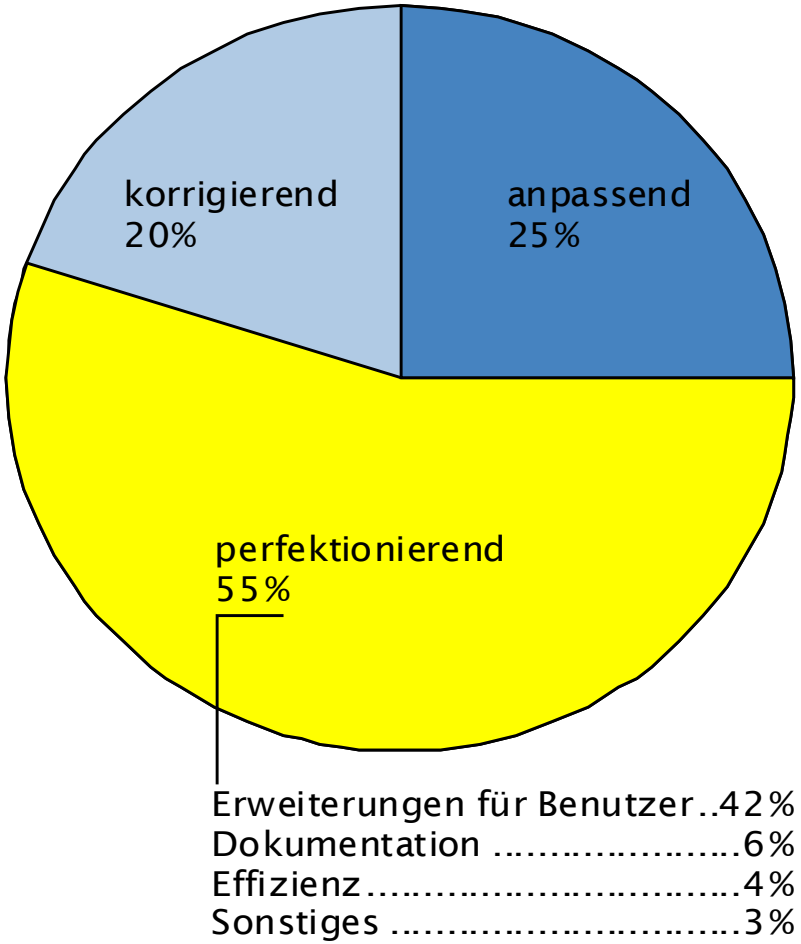
Wartungs- und Pflegephase

- Wartung und Pflege
  - Beginnt mit der erfolgreichen Abnahme und Einführung einer Software
- Nach der Inbetriebnahme eine Produktes...
  - treten im täglichen Betrieb Fehler auf
  - ändern sich die Umweltbedingungen (neue Systemsoftware, neue Hardware, neue organisatorische Einbettung, ...)
  - entstehen neue Wünsche und Anforderungen (neue Funktionen, geänderte Benutzungsoberfläche, erhöhte Geschwindigkeit, ...)
- Alterung von Software
  - Software, bei der nicht ständig Fehler behoben und Anpassungen sowohl an die Umwelt als auch an neue Anforderungen vorgenommen werden, altert und ist irgendwann veraltet
  - Sie kann dann nicht mehr für den ursprünglich vorgesehenen Zweck eingesetzt werden
  - »Software veraltet in dem Maße, wie sie mit der Wirklichkeit nicht Schritt hält« [Sneed 83].
- 4 Kategorien der Wartungs- & Pflege
  - korrektive Tätigkeiten
    - Stabilisierung / Korrektur
    - Optimierung / Leistungsverbesserung
  - progressive Tätigkeiten
    - Anpassung / Änderung
    - Erweiterung.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Aufwandsverteilung





UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Vergleich Wartung und Pflege

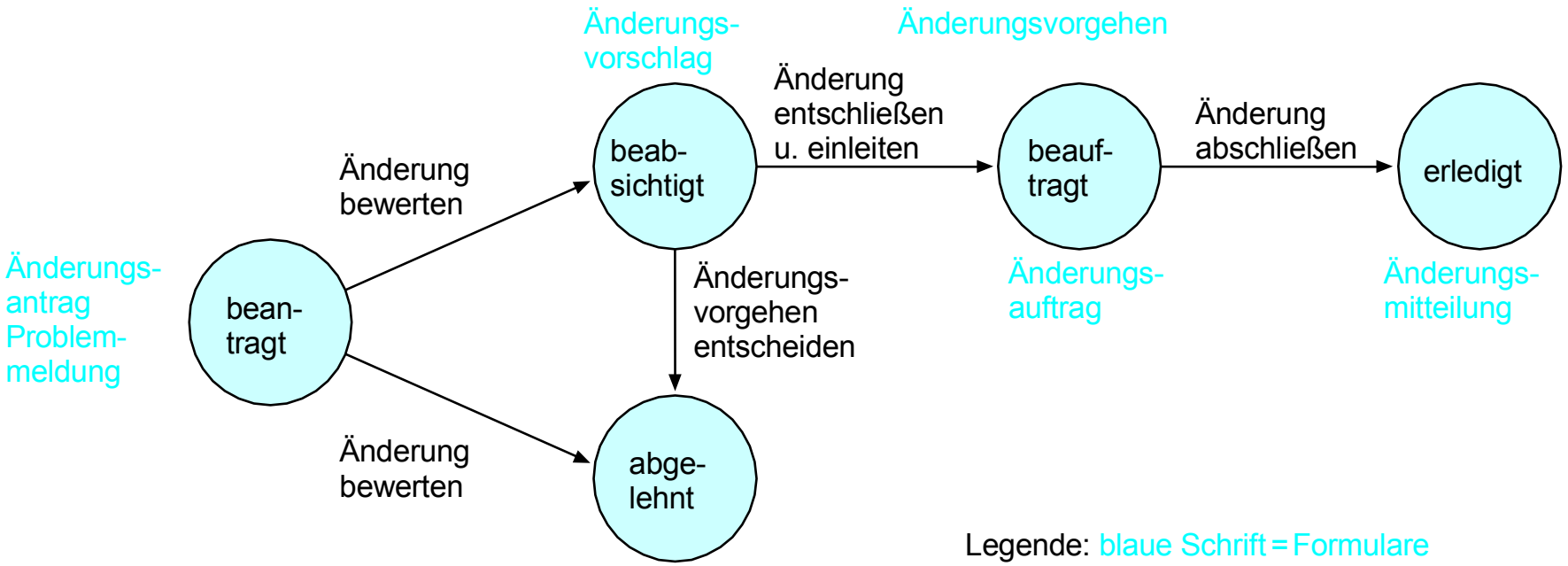
- **Wartung**: Lokalisierung und Behebung von Fehlerursachen von in Betrieb befindlichen Software-Produkten, wenn die Fehlerwirkung bekannt ist.
- **Pflege** : Lokalisierung und Durchführung von Änderungen und Erweiterungen von in Betrieb befindlichen Software-Produkten, wenn die Art der gewünschten Änderungen/Erweiterungen festliegt.
- Charakteristika von Wartungsaktivitäten
  - Ausgangsbasis ist ein fehlerhaftes bzw. inkonsistentes Produkt
  - Abweichungen zwischen Teilprodukten sind zu lokalisieren und zu beheben
  - Die Korrektur einzelner Fehler hat nur begrenzte Auswirkungen auf das Gesamtprodukt
  - Die Fehlerkorrekturen konzentrieren sich i.Allg. auf die Implementierung
  - Ereignisgesteuert, d.h. nicht vorhersehbar und daher schwer planbar und kontrollierbar.
- Charakteristika von Pflegeaktivitäten
  - Ausgangsbasis ist ein konsistentes Produkt, in das gezielt – unter Beibehaltung der Konsistenz – Änderungen und Erweiterungen einzubringen sind
  - Bandbreite bei Änderungen & Erweiterungen kann von kleinen bis zu großen Modifikationen gehen
  - Änderungen und Erweiterungen sind in allen Teilprodukten durchzuführen
    - Produkt-Definition
    - Produkt-Entwurf
    - Produkt-Implementierung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Aufwand für ein Produkt

Verbesserung der Wartung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme



Legende: blaue Schrift = Formulare

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Zusammenfassung (1)

Eine Software-Entwicklung endet mit der Abnahme und Einführung des Produkts.

Der Auftraggeber (extern oder intern) testet in der Abnahmephase das fertig-gestellte Produkt gegen die in der Produktdefinition festgelegten Anforderungen.

Nach erfolgter Abnahme wird in der Einführungsphase das Produkt beim externen Auftraggeber (Individualsoftware) oder bei Pilotkunden (Standard-software) installiert und in Betrieb genommen. Die Inbetriebnahme kann durch direkte Umstellung, Parallellauf oder einen Versuchslauf erfolgen.

Mit der erfolgreichen Einführung und Freigabe des Produkts ist die Produkt-entwicklung beendet.

Mit dem Betriebsbeginn eines Produkts beginnt seine Wartung und Pflege.
Die Wartung umfasst folgende Aufgaben :

- Stabilisierung eines Produkts durch Fehlerkorrekturen,
- Optimierung des Produkts durch Leistungsverbesserungen.

Die Pflege erledigt folgende Aufgaben :

- Änderungen eines Produkts durch Anpassung an die geänderte Umwelt
- Erweiterungen eines Produkts durch funktionale Ergänzungen.

Maintenance ist der amerikanische Oberbegriff für die Wartung und Pflege.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Zusammenfassung (2)

Betrachtet man den Lebenszyklus (life cycle) eines Software-Produkts, dann betragen

- die Entwicklungskosten zwischen 20% und 33% und
- Die Wartungs- und Pflegekosten zwischen 80% und 67%

der gesamten Lebenszykluskosten.

Da die Pflegeaktivitäten und die Wartungsaktivitäten unterschiedliche Charakteristika besitzen, sollten beide Aktivitäten getrennt werden.

Pflegeaktivitäten sollten –abgesehen von minimalen Änderungen- als Weiterentwicklung eines Software-Produkts angesehen werden und den normalen Software-Entwicklungsprozess durchlaufen (Erstellen einer neuen Produktversion). Eine Pflegephase gibt es dann nicht mehr. Bei älteren Produkten muss geprüft werden, ob eine Wartung und Weiterentwicklung noch ökonomisch ist. Wenn nein, dann ist zu entscheiden, ob eine Sanierung (Reengineering) sinnvoll ist oder ob das Produkt durch ein neues zu ersetzen ist.

Eine effektive Wartung wird durch eine geeignete Wartungsorganisation erreicht. Unverzichtbar ist ein Konfigurations- und Änderungsmanagement. Außerdem sind die Vor- und Nachteile einer organisatorisch unabhängigen Wartung abzuwägen.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Literatur

- Balzert H., Lehrbuch der Softwaretechnik, 2. Auflage, Spektrum Akademischer Verlag,Heidelberg, 2000.

- Balzert H., Ökonomische Software-Wartung durch adäquate Software-Konstruktion, in: Softwarewartung, BI-Verlag, Mannheim, 1988, S. 20-86

- Grady R. B., Practical software metrics for Projekt Management and Process, Prentice Hall, Englewood Cliffs, 1992

- McCall J.A., Richards P.K., Walters G.F., Factors in Software Quality, Development Center, Rome Air, 1983

- Zelkowitz M. V., Shaw A.C., Gannon J.D., Principles of Software Engineering and Design, Prentice Hall, Englewood, Cliffs, 1979