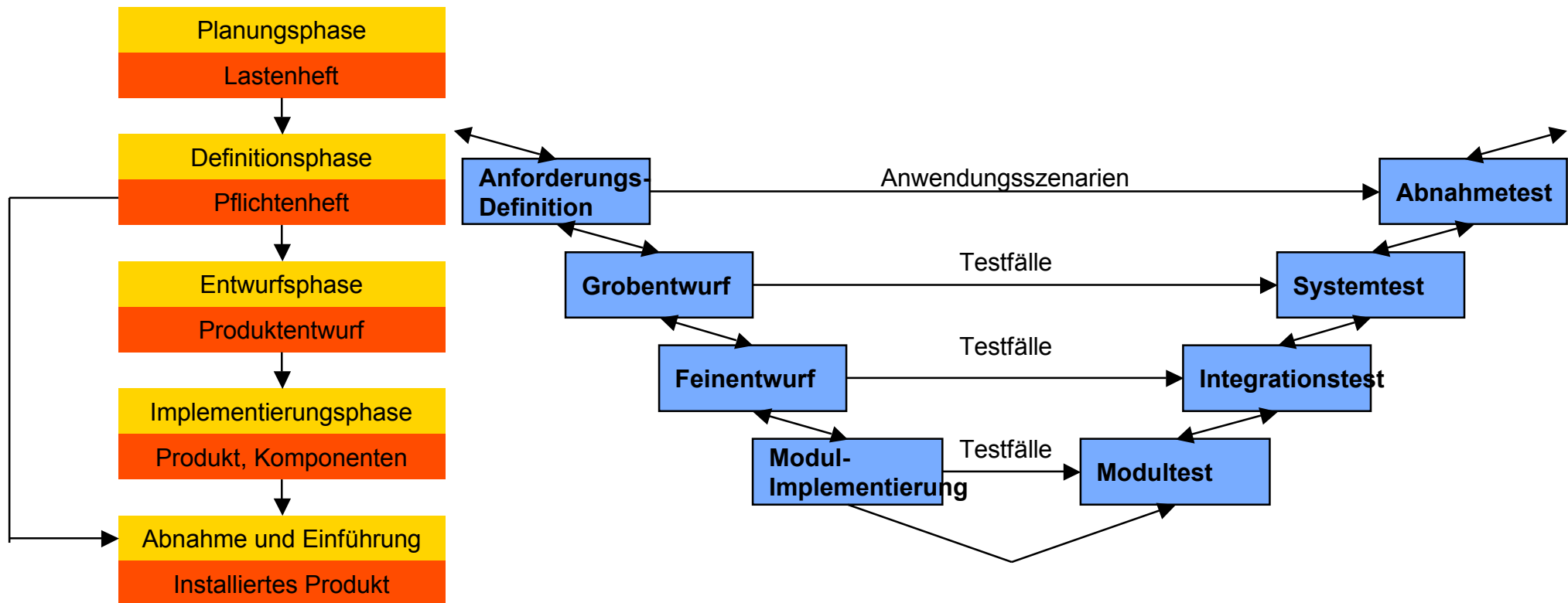


Vorlesung "Software-Engineering"

Prof. Ralf Möller, TUHH, Arbeitsbereich STS

- Vorige Vorlesung
 - Spezifikationstechniken
- Heute:
 - Fortsetzung: Testverfahren
 - Konstruktive Qualitätssicherung

Einordnung



Testverfahren: Dynamische Verfahren

- Software wird mit Testdaten ausgeführt
- Ausführung in realer Umgebung
- Prinzip des Stichprobenverfahrens
- Notwendigkeit zur Auswahl von Testfällen und Testdaten

Unterscheidung Testfälle - Testdaten

■ Testfall:

- eine aus der Spezifikation oder dem Programm abgeleitete Menge von Eingabedaten zusammen mit den zugehörigen erwarteten Ergebnissen

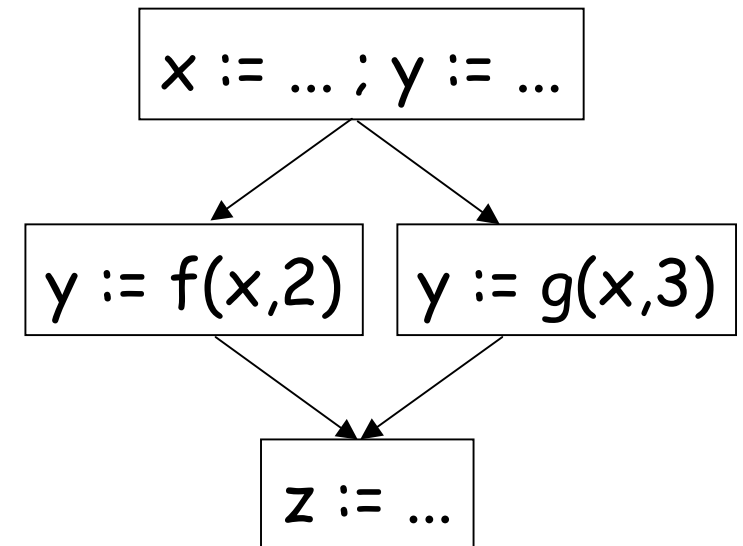
■ Testdaten:

- Teilmenge der Eingabedaten der Testfälle, mit denen das Programm tatsächlich ausgeführt wird

Kontrollflußbezogene Verfahren (1)

- Darstellung der Programm(-teile) als Kontrollflußgraphen
 - Anweisungen (Anweisungsblöcke) als Knoten des Graphen
 - Kontrollfluss als gerichtete Kanten zwischen Knoten
 - Zweig: Einheit aus einer Kante und den dadurch verbundenen Knoten
 - Pfad: Sequenz von Knoten und Kanten, die am Startknoten beginnt und am Endknoten endet

```
x := ... ; y := ...  
IF x > 5 AND y < 2  
THEN y := f(x,2)  
ELSE y := g(x,3)  
z := ...
```



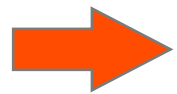
Kontrollflussbezogene Verfahren (2)

- Prinzip der "Überdeckung von Kontrollstrukturen" durch Testfälle (Coverage Analysis):
 - Gezieltes Durchlaufen (von Teilen) der Kontrollstrukturen durch geeignete Gestaltung der Testfälle
 - Angestrebter/erreichter Überdeckungsgrad in Prozent angegeben

Arten der Überdeckung von Kontrollstrukturen

- Anweisungsüberdeckung:
 - Alle Anweisungen werden mindestens einmal ausgeführt
- Zweigüberdeckung:
 - Alle Verzweigungen im Kontrollfluss werden mindestens einmal verfolgt
- Bedingungsüberdeckung:
 - Alle booleschen Wertekonstellationen von (Teil-) Bedingungen werden einmal berücksichtigt
- Pfadüberdeckung:
 - Durchlaufen aller Pfade von Start- zum Endknoten (außer bei sehr kleinen Programmen nur theoretisch möglich)

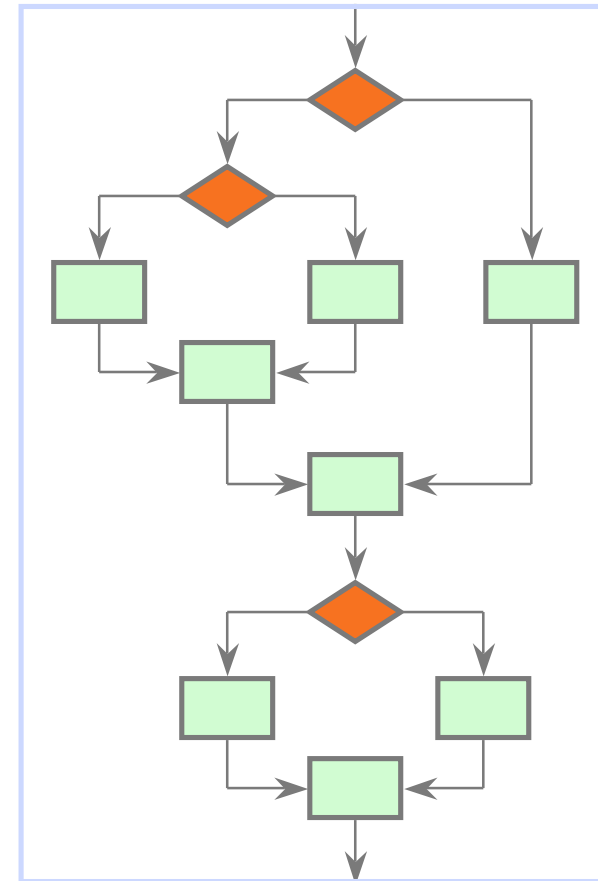
Coverage Analysis



Validierung der Vollständigkeit von Testreihen anhand von Metriken

Arten:

- Anweisungsüberdeckung (statement coverage)
-> zu schwach
- Entscheidungsüberdeckung (branch coverage)
-> minimum mandatory testing requirement
- Pfadüberdeckung (path coverage)
-> in der Praxis nicht durchführbar



Datenflußbezogene Verfahren (1)

- Erweiterung der Kontrollflußgraphen um Datenflüsse
- Unterscheidung verschiedener Situationen hinsichtlich des Zugriffs auf Variablen im Programm:
 - Definition von Variablenwerten (Wertzuweisung)
 - Berechnende Benutzung (zur Ermittlung eines Wertes in Ausdrücken)
 - Prädikative Benutzung (Auswertung der Bedingungen in bedingten Anweisungen oder Schleifen)

Datenflußbezogene Verfahren (2)

- Prinzip des Verfahrens:
 - Für jeden Definitionsknoten werden definitionsfreie Teilpfade zu allen Benutzungsknoten ermittelt
 - Die Auswahl der Testdaten muß das Durchlaufen jedes identifizierten definitionsfreien Teilpfades sicherstellen

Datenflußbezogene Verfahren (3)

- Definition von drei Funktionen als Basis der Datenflußverfahren:
 - Funktion def bildet jeden Knoten auf die Menge der darin global definierten Variablen ab
 - Funktion c-use ordnet jedem Knoten die in ihm berechnend benutzten Variablen zu
 - Funktion p-use weist jeder Kante die Menge von Variablen zu, deren prädikative Auswertung zum Durchlaufen der Kante notwendig ist
- Definition der Menge $\text{du}(v, n)$, wobei $v \in \text{def}(n)$:
 - Falls ein Teilpfad von n nach n_i existiert, auf dem v nicht neu definiert wird
 - $n_i \in \text{du}(v, n)$ falls $v \in \text{c-use}(n_i)$
 - $n_i \in \text{du}(v, n)$ falls (n_i, n_k) existiert mit $v \in \text{p-use}((n_i, n_k))$
- Auswahl von Testfällen:
 - All-uses-Kriterium:
 - Für jeden Knoten n und jede Variable $v \in \text{def}(n)$ wird mindestens ein definitionsfreier Pfad von n zu allen Elementen von $\text{du}(v, n)$ ausgeführt.

Funktionale Verfahren

- Überprüfung der in der Spezifikation festgelegten Funktionalität
 - Programmstruktur irrelevant
 - Beste Grundlage: formale Spezifikation
- Bestimmung der Testdaten:
 - Bildung "funktionaler Äquivalenzklassen":
Eingabe- und Ausgabemengen, die jeweils zu einer (Teil-) Funktionalität gehören
 - Alle Werte einer Äquivalenzklasse verursachen ein identisches funktionales Verhalten eines Programms
- Auswahl von Testdaten aus den Äquivalenzklassen:
 - Zufällig
 - Test spezieller Werte (z.B. 0, nil)
 - Grenzwertanalyse (primär Grenzbereiche der Eingabemengen als Testdaten)

Testen von Modulen

- Aufdecken von Fehlern in der isolierten Modulrealisierung
- Modultest wird meist als Teilphase der Implementierung betrachtet
- Notwendigkeit einer Testumgebung, die die anderen Module simuliert und das Modul selbst benutzbar macht (Fehler in Testumgebung?)

Testumgebungen

- (Wiederholten) Aufruf des Testobjektes ermöglichen
- Eingabedaten für Testobjekt bereitstellen
- Externe Ressourcen und deren Ergebnisse/Aktivitäten simulieren (z.B. importierte Module)
- Ergebnisse der Testausführung ausgeben und speichern

Testen von Modulverbindungen

- Testen von Subsystemen
(= Teilmengen von verknüpften Modulen)
- Prüfung der Kommunikation zwischen den Modulen
- Schrittweiser Aufbau größerer Subsysteme durch Hinzufügen weiterer Module/Subsysteme
(inkrementelles Testen, Integrationstest)
- Ausführung meist Bottom-up, jedoch auch Top-Down-Vorgehensweise möglich
- Ebenfalls Testumgebung notwendig, "Stub"-Moduln bzw. Treiber-Moduln erforderlich

Zusammenfassung

■ Junit

- Einfaches Framework für den Unit-Test von Java-Programmen.
- Testfälle werden parallel zum eigentlichen Code entwickelt.
- Die Tests können vollautomatisch ausgeführt werden.

Qualitätssicherung durch Tests

- Ein Test ist prinzipiell nur geeignet, evtl. vorhandene Fehler eines Systems zu Tage treten zu lassen, nicht jedoch die Abwesenheit von Fehlern zu zeigen (vgl. Verifikation)
- Der **Überdeckungsgrad** ist ein Maß für den Grad der Vollständigkeit eines Tests
- Ein **Regressionstest** ist ein Testverfahren, bei dem eine Sammlung von Testfällen erstellt wird, die bei Änderungen am System (automatisch) erneut überprüft werden
- Bei einer Individualsoftware findet ein **Abnahmetest** statt
- Beim **Alpha-Test** für Standardsoftware wird das System in der Zielumgebung des Herstellers durch ausgewählte Anwender erprobt
- Beim **Beta-Test** für Standardsoftware wird das System bei ausgewählten Zielkunden in einer eigenen Umgebung zur Probenutzung zur Verfügung gestellt. Auftretende Probleme und Fehler werden protokolliert. Beta-Tester erhalten typischerweise einen Preisnachlaß auf das endgültige Produkt. Typischerweise werden Beta-Tests iterativ mit aufeinanderfolgenden "*release candidates*" (RC) durchgeführt.

Testen: Zusammenfassung

- Rolle der Spezifikation beim Test
- Testsystematik
- Reproduzierbarkeit von Tests
(insbesondere nach Änderungen)
- Tests decken Fehler auf
- Tests helfen nicht, Fehler auszuschließen
- Herstellungsprozeß bedeutsam:
 - > Qualitätsmanagement, konstruktive Verfahren
 - Test helfen, Fehler zu vermeiden

Konstruktive Verfahren

- Annahme: Die Qualität eines Produktes wird maßgeblich durch die Qualität des Herstellungsprozesses bestimmt
 - Qualität ins Bewußtsein der Projektmitglieder bringen
 - Rückverfolgbarkeit gewährleisten
 - Verantwortlichkeiten und Zuständigkeiten schaffen (Planstellen)
 - -> Organisatorischen Rahmen für Software-Herstellungsprozeß schaffen

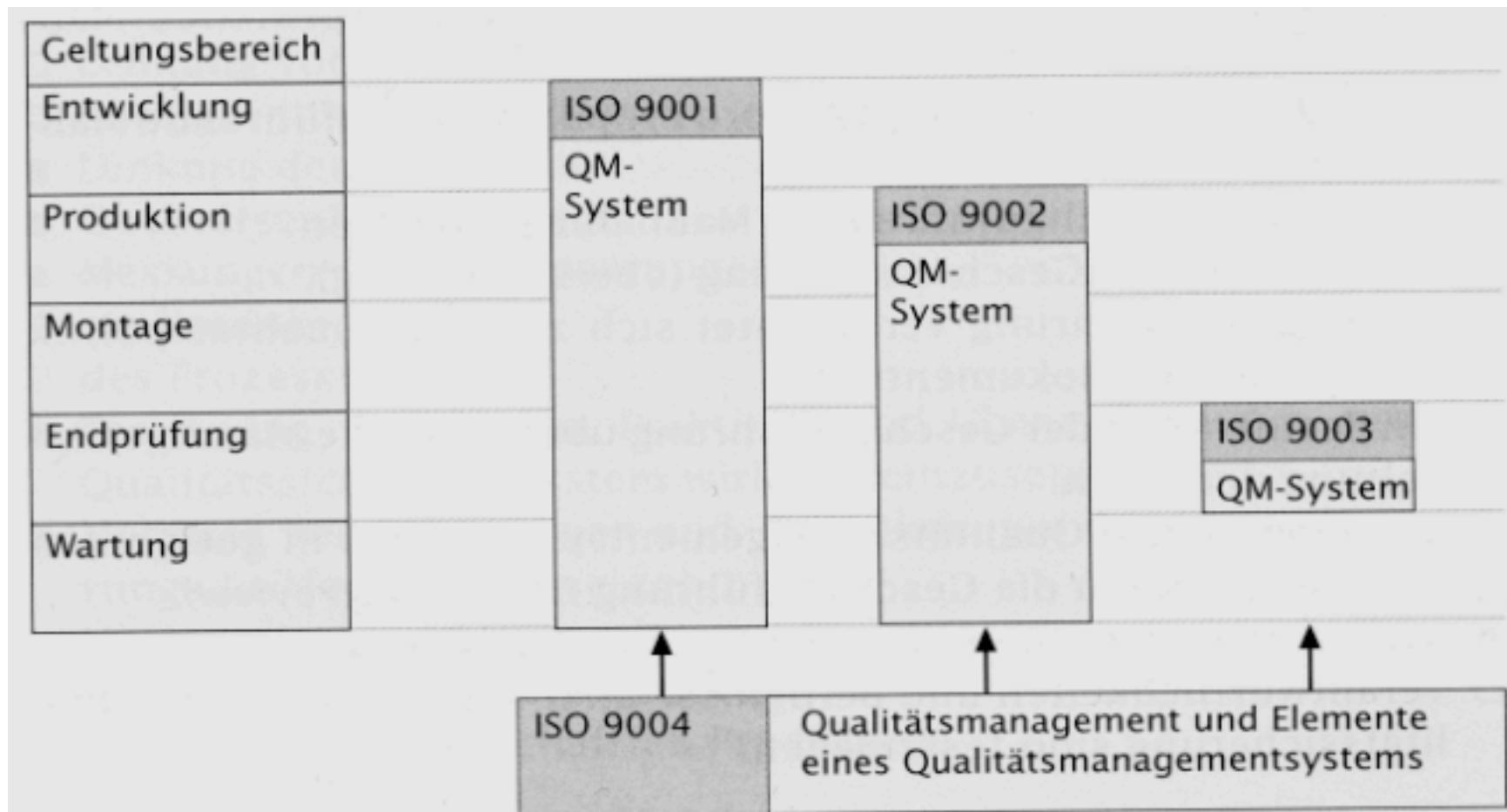
ISO 9000 (erstmalig standardisiert 1987)

- Das ISO 9000 Normenwerk legt für ein Auftraggeber-Lieferanten-Verhältnis einen allgemeinen, übergeordneten, organisatorischen Rahmen zur Qualitätssicherung von Produkten (insbesondere ISO 9001).
- Geprägt durch industrielle Fertigung von Produkten, aber ausgeweitet auf Erstellung von Software (9000-3)
- Zertifizierung nach 9001 bzw. 9000-3 durch Zertifizierungsstelle, wenn ein Qualitätsmanagementsystem vorhanden.
- Zertifiziert wird der Herstellungsprozeß,
NICHT DAS PRODUKT

ISO 9000 - Struktur (1)

ISO 8402	Begriffsbestimmungen
↓	
ISO 9000-1	Leitfaden zur Auswahl und Anwendung
ISO 9000-2	Allgemeiner Leitfaden zur Anwendung von 9001, 9002 und 9003
ISO 9000-3	Leitfaden zur Anwendung von 9001 auf Software
ISO 9000-4	Leitfaden zum Management von Zuverlässigkeitsprogrammen

ISO 9000 - Struktur (2)



ISO 9000-3

■ Aufbau ISO 9000-3:

- Rahmen
- Lebenszyklustätigkeiten
- Unterstützende Tätigkeiten

■ Inhalt ISO 9000-3:

- Entwicklung
- Lieferung
- Wartung von Software

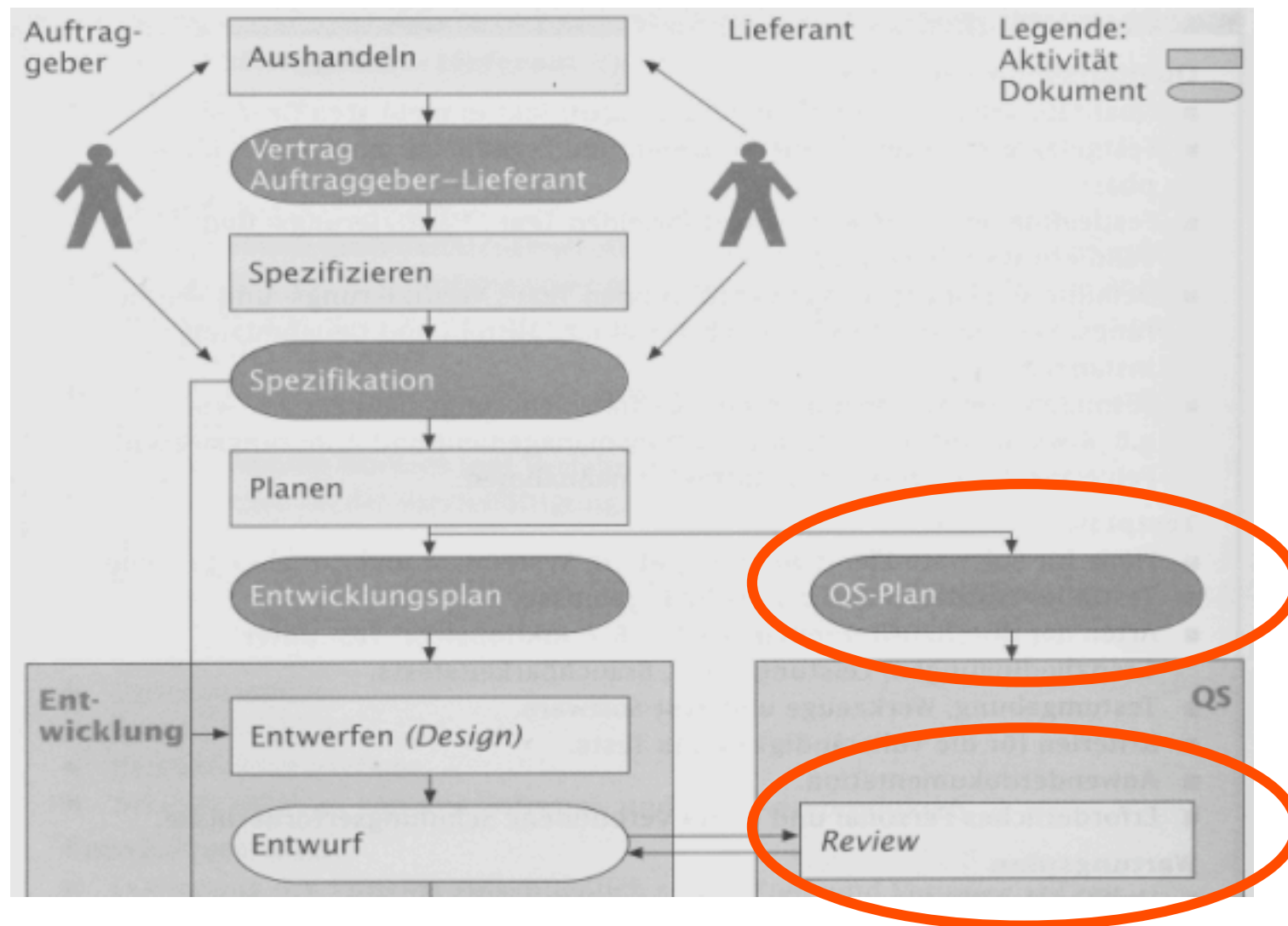
■ Maßnahmen:

- der Geschäftsführung
- der Mitarbeiter der Qualitätssicherung

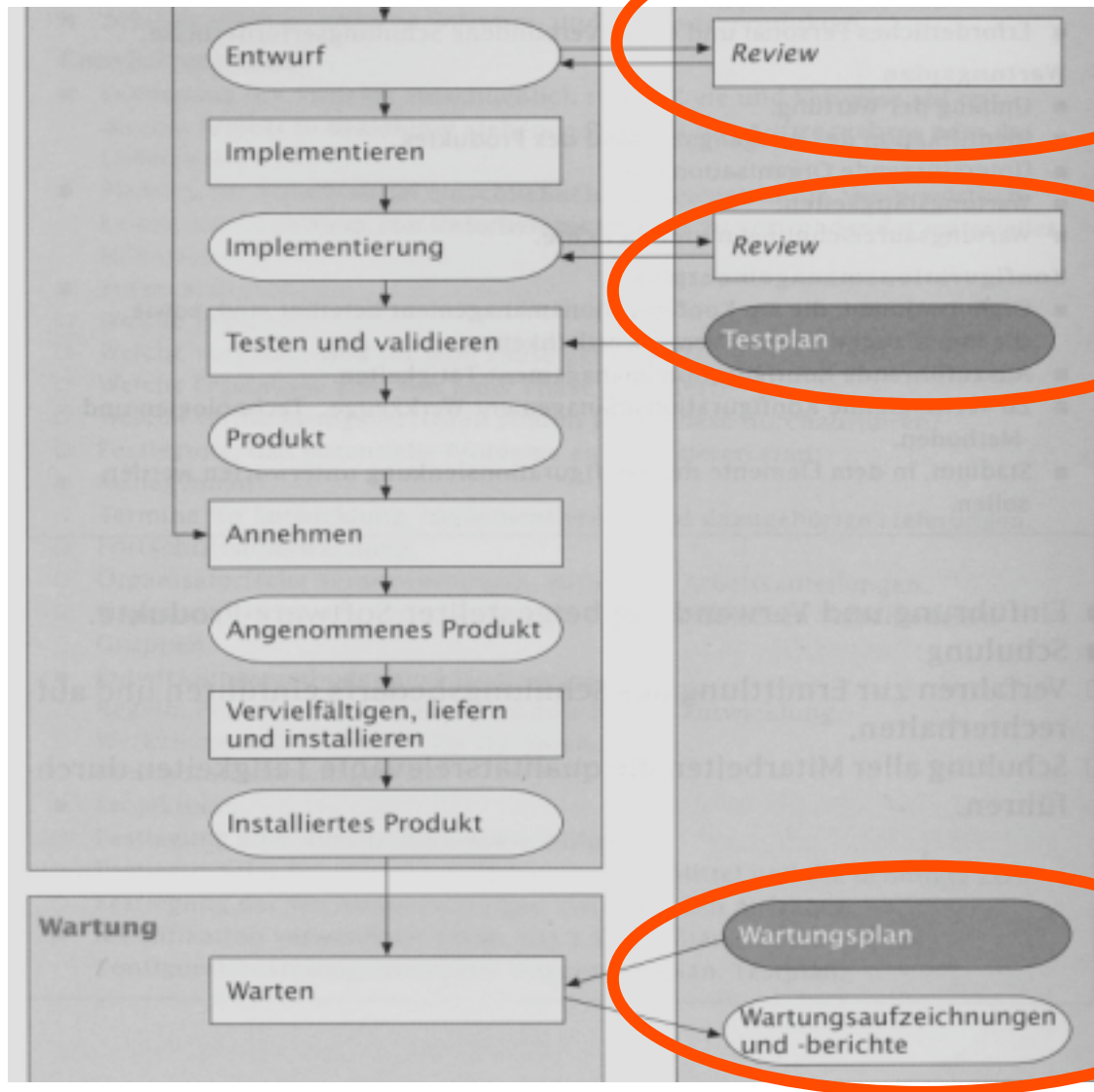
■ Anwendungsformen:

- Darlegung der Qualitätssicherung gegenüber Dritten
- Aufbau und Verbesserung eines QS-Systems

ISO 9000-3 - Implizites Vorgehensmodell



ISO 9000-3 - Implizites Vorgehensmodell



ISO 9000-3 - Implizites Vorgehensmodell

■ Phasenunabhängige Tätigkeiten:

■ Konfigurationsmanagement:

- | Identifikation und Rückverfolgbarkeit d. Konfiguration
- | Lenkung von Änderungen
- | Konfigurations-Statusbericht

■ Lenkung der Dokumente

■ Qualitätsaufzeichnungen

■ Messungen und Verbesserungen:

- | am Produkt
- | des Prozesses

■ Festlegung von Regeln, Praktiken und Übereinkommen für den Einsatz eines Qualitätsmanagementsystems

■ Nutzung von Werkzeugen und Techniken, um QS-Leitfaden umzusetzen

■ Unterauftragsmanagement

ISO 9000 - Bewertung

■ Vorteile:

- Aufmerksamkeit auf Qualitätssicherung
- Externe Zertifizierung und Wiederholung d. Audit
- Festlegen von Anforderungen
- Erleichtert die Akquisition von Aufträgen
- Weniger Produkthaftungsrisiko
- Stärkung Qualitätsbewußtsein

■ Nachteile:

- Unsystematischer Aufbau
- Keine sauber Trennung zwischen fachlichen, Management- und QS-Aufgaben
- Gefahr Software-Bürokratie
- Gefahr mangelnde Flexibilität
- Hoher Aufwand für Einführung und Pflege

Andere Modelle

- TQM (Total Quality Management)
- CMM (Capability Maturity Model)
- SPICE (Software Process Improvement and Capability dEtermination)

- Siehe: Balzert, Lehrbuch der Softwaretechnik Band 2

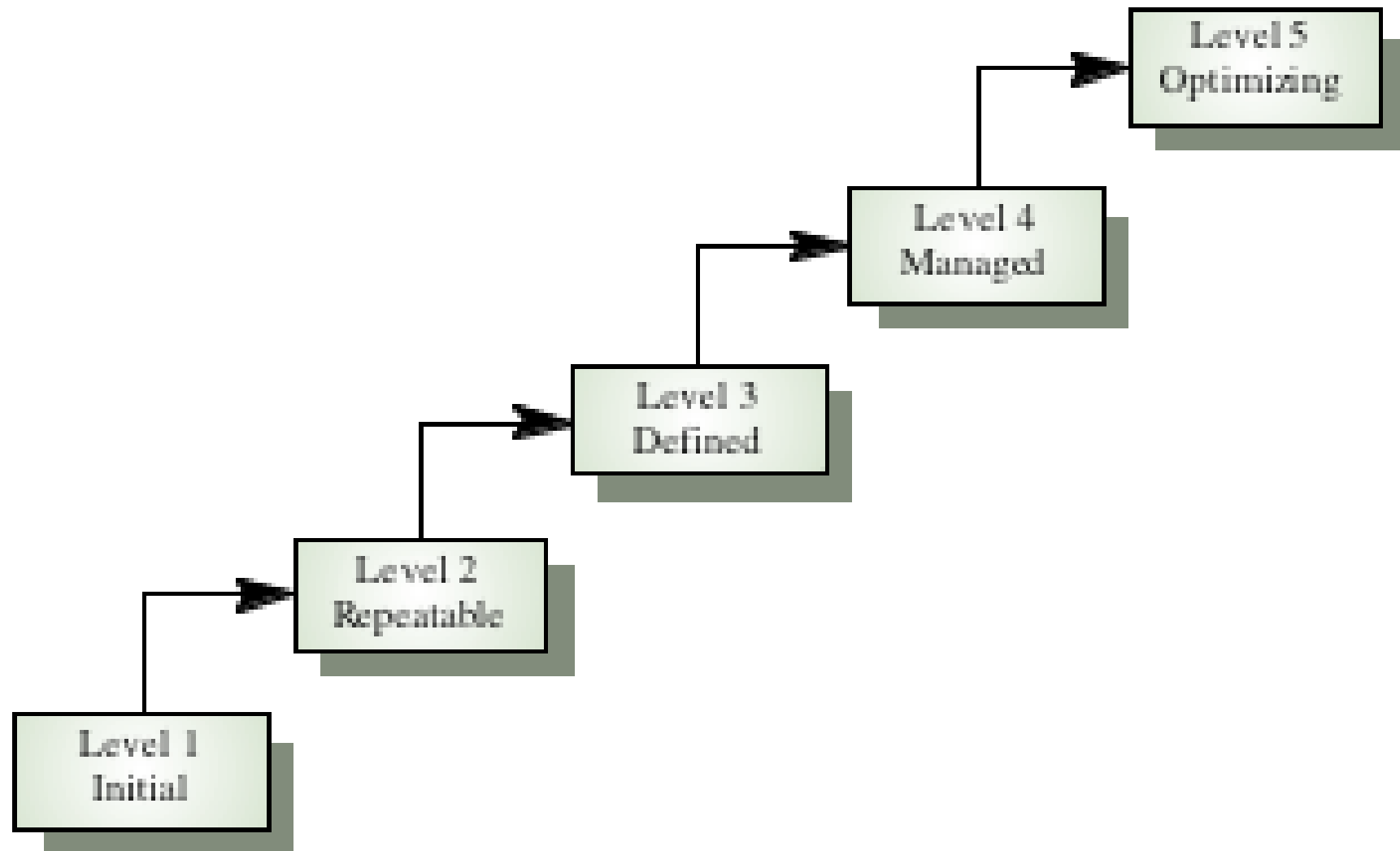
Über das Bewußtwerden und Explizitmachen

- Phasen der Software-Entwicklung
- Organisationsprinzipien
- Projektmanagement
- Vorgehensmodelle
- ...

-> Software Engineering ist mehr als Programmentwicklung

- It's all about processes ...
... and people.

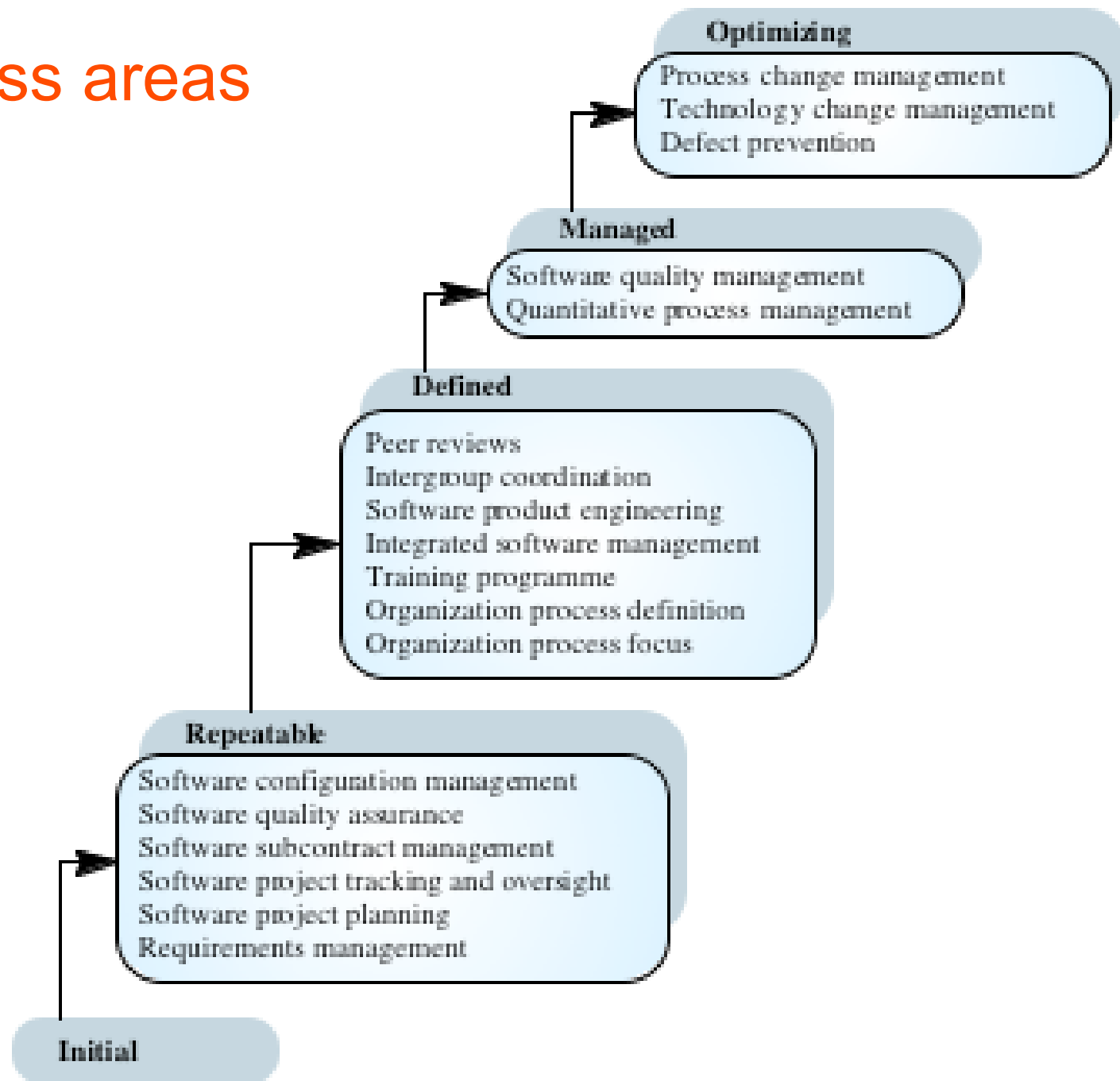
Processes: The SEI capability maturity model



Maturity model levels

- Initial
 - Essentially uncontrolled
- Repeatable
 - Product management procedures defined and used
- Defined
 - Process management procedures and strategies defined and used
- Managed
 - Quality management strategies defined and used
- Optimising
 - Process improvement strategies defined and used

Key process areas



The CMM and ISO 9000

- There is a clear correlation between the key processes in the CMM and the quality management processes in ISO 9000
- The CMM is more detailed and prescriptive and includes a framework for improvement
- Organisations rated as level 2 in the CMM are likely to be ISO 9000 compliant