Martin Glinz Harald Gall

Software Engineering

Wintersemester 2005/06

Kapitel 7

Validierung und Verifikation

Universität Zürich
Institut für Informatik

7.1 Grundlagen

7.2 Fehlermodell

7.3 Grundsätze der Prüfung von Software

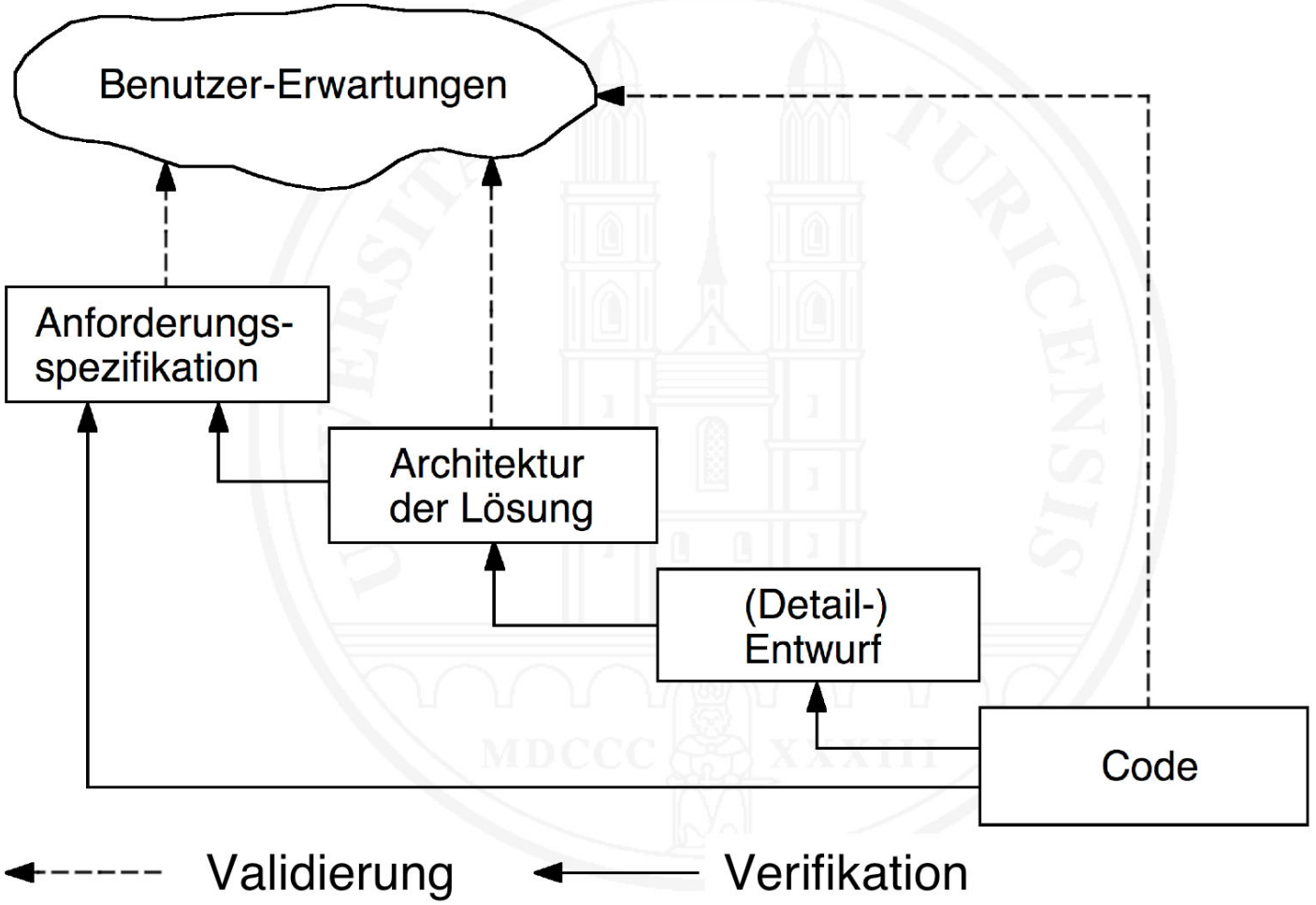
Validierung und Verifikation

- Tun wir das Richtige? ➪ *Validierung*
- Tun wir es richtig? ➪ *Verifikation* [Boehm 1981]

Validierung (validation) – der Prozess der Beurteilung eines Systems oder einer Komponente während oder am Ende des Entwicklungsprozesses, mit dem Ziel, festzustellen, ob die spezifizierten Anforderungen erfüllt sind. (IEEE 610.12).

Verifikation (verification) – (1) der Prozess der Beurteilung eines Systems oder einer Komponente mit dem Ziel, festzustellen, ob die Resultate einer gegebenen Entwicklungsphase den Vorgaben für diese Phase entsprechen, (2) der formale Beweis der Korrektheit eines Programms. (IEEE 610.12).

V&V im Entwicklungsprozess



7.1 Grundlagen

7.2 Fehlermodell

7.3 Grundsätze der Prüfung von Software

Fehler ist nicht gleich Fehler

- Insbesondere müssen Fehler, Fehlerursachen
- und Ausfälle auseinandergehalten werden
- Saubere Terminologie notwendig

Fehlerterminologie

- Eine Person begeht einen Irrtum (mistake) .
- Als mögliche Folge davon enthält die Software einen Defekt (defect, fault).
- Wird der Defekt durch Inspizieren der Software gefunden, so ergibt das einen Befund (finding).
- Bei der Ausführung von Software mit einem Defekt kommt es zu einem Fehler (error): Die tatsächlichen Ergebnisse weichen von den erwarteten / den richtigen ab.
- Dies kann zum Ausfall (failure) eines software-basierten Systems führen.
- Wird ein Fehler festgestellt, so muss die Fehlerursache gefunden und behoben werden (Fehlerbeseitigung, Debugging)

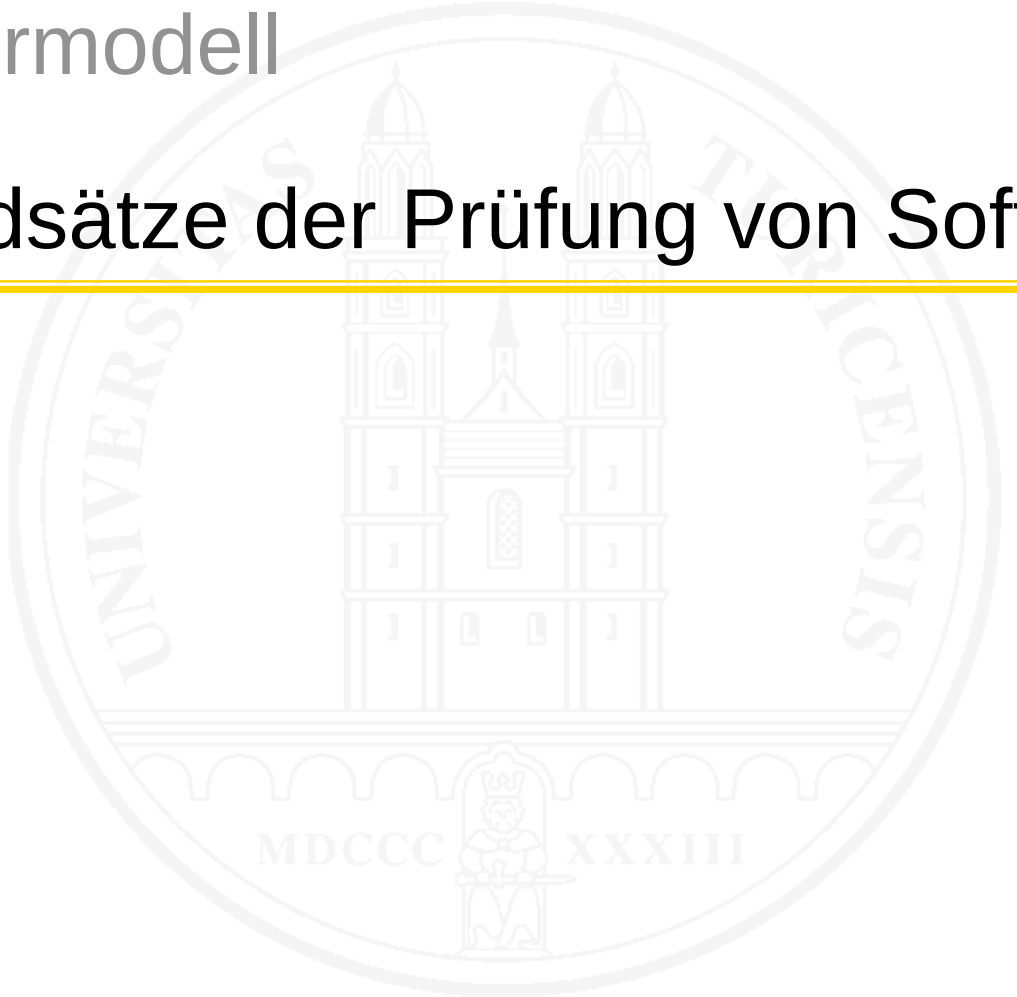
Umgangssprachliche vs. exakte Terminologie

- Umgangssprachlich wird in allen Fällen häufig nur von Fehler (error, bug) gesprochen.
- Im Qualitätsmanagement ist die genaue Unterscheidung oft wichtig:
  - Ein Defekt hat häufig mehrere Fehler zur Folge
  - Ein Defekt führt nicht immer zu einem Fehler
  - Nicht jeder Fehler hat einen Defekt als Ursache
  - Nicht jeder Fehler hat einen Ausfall zur Folge
  - Defekte ≠ Fehler ≠ Ausfälle !
- Defekte werden auch begangene Fehler oder Fehlerursachen genannt
- Fehler werden manchmal präziser bezeichnet als gefundene, erkannte oder festgestellte Fehler

7.1 Grundlagen

7.2 Fehlermodell

7.3 Grundsätze der Prüfung von Software



Prüfgrundsätze

- Nur gegen Vorgaben (Anforderungen oder Vergleichsresultate) prüfen
- Systematisch prüfen
  - Prüfstrategie festlegen
  - Prüfung (im Rahmen des Projektmanagements) planen
  - Prüfvorschriften erstellen
  - Prüfungen nach Vorschrift durchführen
  - Prüfergebnisse zusammenfassen und dokumentieren
- Prüfverfahren müssen wohldefiniert sein und reproduzierbare Ergebnisse liefern
- Prüfergebnisse müssen dokumentiert werden
- Beim Prüfen erkannte Fehler müssen anschließend korrigiert werden (indem die verursachenden Defekte erkannt und behoben werden)

Prüfverfahren

Statische Verfahren

- Review (Inspektion, Walkthrough)
- Statische Analyse
- Korrektheitsbeweis (formale Verifikation)
- Model Checking
- Messen

Dynamische Verfahren

- Testen
- Simulieren
- Prototypisieren

Prüfstrategie

- Qualitätsmerkmale gewichten
- Für jedes Qualitätsmerkmal das Risiko abschätzen, dass der Kunde reklamiert
- Je geringer das Risiko, desto weniger Prüfung ist erforderlich
- Prüfvorgehen festlegen
  - zu prüfende Artefakte (was)
  - gebunden an welche Meilensteine (wann)
  - Prüfverfahren (wie)
- Grobe Aufwandsabschätzung
- Ergebnisse in die Projektplanung übernehmen

Literatur

Boehm, B. (1981). *Software Engineering Economics*. Englewood Cliffs, N.J.: Prentice Hall.

IEEE (1988). *Standard Dictionary of Measures to Produce Reliable Software*. IEEE Std 982.1-1988. IEEE Computer Society Press.

Liggesmeyer, P. (2002). *Software-Qualität: Testen, Analysieren und Verifizieren von Software*. Berlin: Spektrum Akademischer Verlag.

Pol, M., T. Koomen, A. Spillner (2000). *Management und Optimierung des Testprozesses*. Heidelberg: dpunkt.verlag.

Zeller, A. (2006). *Why Programs Fail: A Guide to Systematic Debugging*. San Francisco: Morgan Kaufmann und Heidelberg: dpunkt.verlag.

Im Skript [M. Glinz (2005). *Software Engineering*. Vorlesungsskript, Universität Zürich] lesen Sie Kapitel 9.3.

Im Begleittext zur Vorlesung [S.L. Pfleeger, J. Atlee (2006). *Software Engineering: Theory and Practice*, 3rd edition. Upper Saddle River, N.J.: Pearson Education International] lesen Sie Kapitel 4.9 und 8.1.