Lernzettel

Softwaretechnik und Programmierparadigmen

Universität: Technische Universität Berlin
Kurs/Modul: Softwaretechnik und Programmierparadigmen
Erstellungsdatum: September 19, 2025

Lernzettel: Softwaretechnik und Programmierparadigmen

(1) Grundlagen der Softwaretechnik
Die Softwaretechnik beschreibt die systematische Herstellung von Software. Zentrale Aspekte sind:

- Entwicklungsmethoden zur Planung, Umsetzung und Wartung,
- Techniken der Projektorganisation, Rollen und Kommunikation,
- Qualitätssicherung, Metriken und Verifikation,
- Anforderungsanalyse als Grundlage aller weiteren Schritte.

(2) Vorgehensmodelle und Prozessorganisation

- Planungsorientierte Modelle (z. B. Wasserfall, V-Modell) legen Phasen fest.
- Iterative Modelle (z. B. inkrementell, iterativ) ermöglichen schrittweise Verbesserung.
- Agile Ansätze (Scrum, Extreme Programming) betonen Flexibilität, frühe Tests und enge Zusammenarbeit.

(3) Anforderungsanalyse

- Stakeholder-Identifikation, Erhebung, Priorisierung.
- Funktionale vs. nicht-funktionale Anforderungen.
- Use Cases, Szenarien und Verträge zur Spezifikation.
- Requirements Engineering als Grundlage für Design und Implementierung.

(4) Architektur und Modularisierung

- Architektur: Struktur des Systems in Teilkomponenten, Schnittstellen und Beziehungen.
- Prinzipien: Kapselung, lose Kopplung, hohe Kohäsion.
- Muster: Layered Architecture, Clean Architecture, Microservices (Beispiele).
- Modularisierung ermöglicht klare Verantwortlichkeiten und Wiederverwendbarkeit.

(5) Verhaltensbeschreibung

- Beschreibt, wie das System in typischen Situationen reagieren soll.
- Use-Case-Szenarien, Abstufungen von Reaktionspfaden, Zustandsdiagramme.

- Modellbasierte Beschreibungen unterstützen frühes Feedback.

(6) Qualitätssicherung

- Tests: Unit-, Integrations-, Systemtests, ggf. Akzeptanztests.
- Verifikation durch den Hoare-Kalkül:

$$\{P\}\ C\ \{Q\}$$

- Metriken: Fehlerrate, Testabdeckung, Zykluszeit, Stabilität.
- Reviews, Audits und formale Ansätze ergänzen die Praxis.

(7) Architekturklassifikation

- Architekturen unterscheiden sich nach Zweck: Anwendungsarchitekturen vs. Systemarchitekturen.
- Beispiele: monolithische, serviceorientierte (SOA), ereignisgesteuerte Architekturen (EDA).
- Abgrenzung anhand Eigenschaften wie Skalierbarkeit, Wartbarkeit, Latenz.

(8) Programmierparadigmen

Objektorientierte Programmierung (OOP):

- Klassen, Objekte, Kapselung, Vererbung, Polymorphie.
- Fokus: Status und Verhalten, Wiederverwendung durch Hierarchien.

Modellorientierte Programmierung (modellgetrieben):

- Modelle als zentrale Repräsentation von Systemen.
- Transformationen von Modellen zu Implementierungen, Generierung von Code aus Modellen.

Funktionale Programmierung:

- Fokus auf Funktionen, Unveränderlichkeit, Nebenwirkungsfreiheit.
- Höhere Ordnung, Funktionskomposition, Referentielle Transparenz.

Logische Programmierung:

- Deklarativ, Regeln und Fakten, Abfragen über logische Schlaufen.
- Typische Sprache: Prolog; Schlaufen durch Rückwärtsverkettung.

(9) Verbindungen und Vergleich der Paradigmen

- Unterschiede in Wartbarkeit, Testbarkeit und Parallelität.
- Mischformen: mehrparadigmäße Sprachen (z. B. Java, Scala, F) unterstützen mehrere Paradigmen.
- Auswahlkriterien: Anforderungen, Teamkompetenz, Wartungsbedürfnisse.