technische universität
dortmund

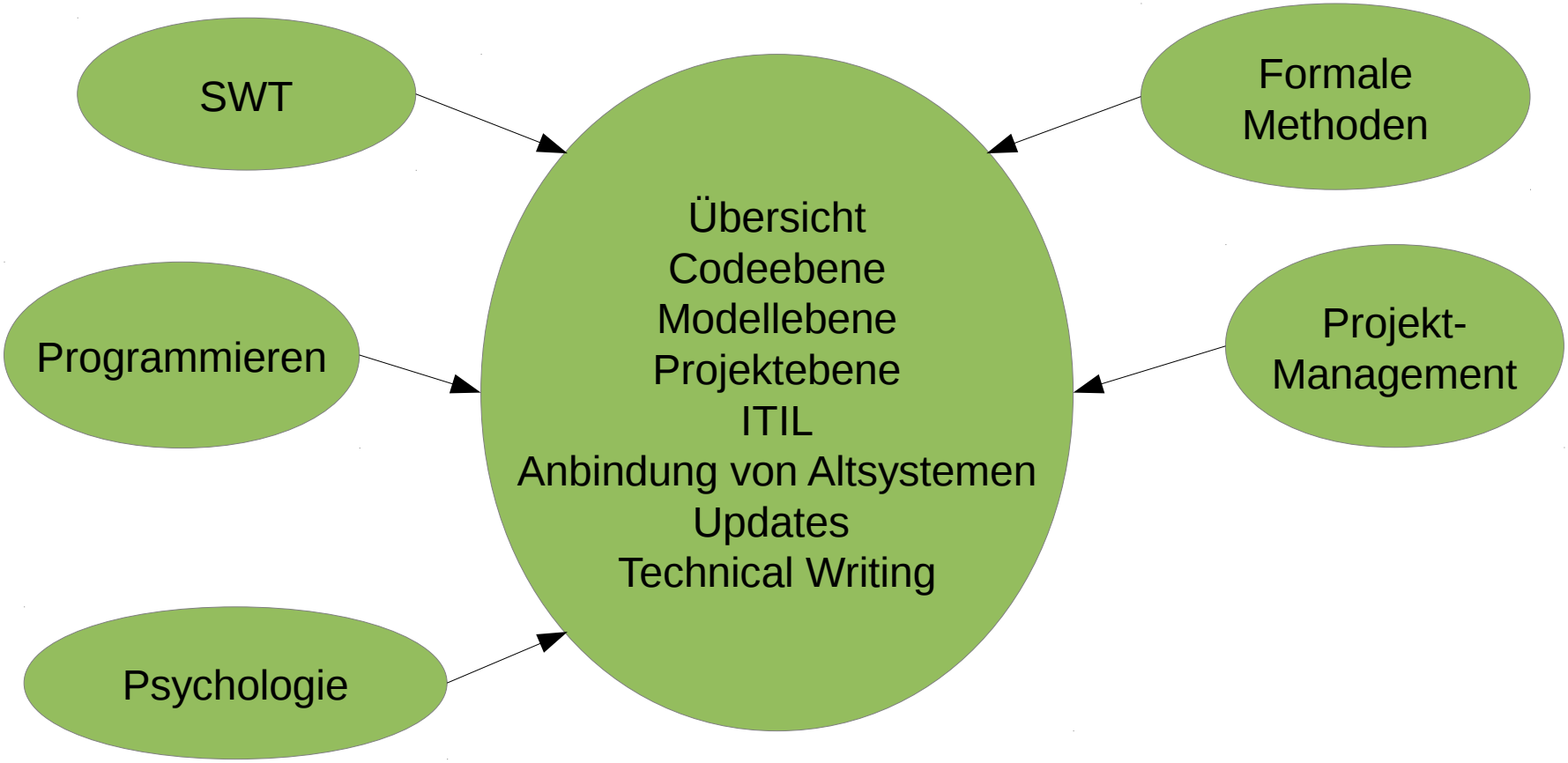
Software- Engineering für langlebige Systeme

Thomas Ruhroth | Dortmund
SoSe 2014

VL11 WarpUp

- Nochmal zurückschauen und Verbindungen sehen
-

Was haben wir gesehen:



Softwareerosion

Quellen ...

Änderungen in der Systemumgebung

- Prozessoränderungen
  - Umstellung 8 → 16 → 32 → 64 Bit-Systeme (s. Übung)
  - Prozessorgeschwindigkeit (TurboPascal- Zählerüberlauf)
- Änderungen im Betriebssystem
  - Geänderte Zugriffsrechtesysteme
- Neue/Geänderte Schnittstellen

Verlust von Wissen

- Designentscheidungen können nicht mehr nachvollzogen werden
  - Wichtige Gründe werden bei Umbauten nicht bedacht
- Bedeutung von Magic Numbers geht verloren
- Funktionen werden außerhalb der vorgesehenen und geprüften Bereiche genutzt

Unterschiede in der Programmiererfahrung

- Code Anderer wird fehlinterpretiert
- „Verschlimmbessern“
- Unangemessene Verallgemeinerung/Vereinfachung

Fehler in der Arbeitsorganisation

- Programmierer wissen es besser als Designer
  - Design und Code passen nicht zusammen
  - Testcases passen nicht zur Code-Basis
- Schlecht designter Prototyp wird Produkt
- Häufiger Wechsel der Programmierer/ Designer
- Updatefähigkeit wird später ergänzt
  - Teilweise werden Veränderungen „zurückgebaut“
- Keine „deprecated“-Verwaltung
  - Viel alter Code bleibt im System
- Aufgabenteilung nicht gelebt

Schlechter Code

- Code nicht verständlich
- Wenn eine Stelle geändert wird, müssen viele (weit entfernte) Code-Stellen mit angepasst werden.
- Wenn der Code geändert wird, ergeben sich viele Folgefehler

Achtung:

- Schlechter Code ist Auswirkung und Quelle zugleich!
- Hier beginnt ein Teufelskreis

Schlechtes Design

- Das Design ist zu komplex/zu einfach
- Es existieren viele (unübersichtliche) Abhängigkeiten (s. Findbugs-Bsp)
- Entscheidungen sind nicht dokumentiert
- Verschiedene widersprüchliche Spezifikationen
- Design für Zielsprache nicht angemessen/passend

Achtung:

- Schlechtes Design ist Auswirkung und Quelle zugleich!
- Hier beginnt ein Teufelskreis

Inkonsistenzen zwischen Artefakten

- Code
- Datenbank
- Design/Modell
- Dokumentation
- Tests

Achtung:

- Inkonsistenzen zwischen Artefakten sind Auswirkung und Quelle zugleich!
- Hier beginnt ein Teufelskreis

Beyond One-Shot Security:

Keeping Information Systems Secure through Environment-Driven Knowledge Evolution

Abschließende Bemerkungen

Mein Rat für weiteres Lernen

- Englisch
  - Alle wichtige Literatur kommt im Deutschen erst mit erheblicher Verspätung
  - Dokumentation
- Viel Programmieren in verschiedenen Sprachen
  - Es gibt nicht eine gute Sprache
  - Verschiedene Paradigmen
- Sozial- und Psychologie-Grundlagen
  - Verstehen der Effekte in einer Gruppe
- Interkulturelles Wissen
- Projekt-Management

Bachelorarbeiten in unserer Arbeitsgruppe

- Themen zu Sicherheit und Compliance
  - Sicherheit in der Cloud
  - Risiko-Management
- Softwareevolution und langlebige Systeme

- Webseite:

  http://www-secse.cs.tu-dortmund.de/secse/pages/teaching/thesis/index_de.shtml

- Regelmäßige Vor-Ort-Arbeitszeiten
  - Kurzfristige Treffen mit den Betreuern möglich
  - Austausch mit anderen Studenten
- Begleitendes Merkblatt verfügbar

Lehre nächstes Semester

-
- Vorlesung Softwarekonstruktion (2+1 SWS, INF-BSc-211)
- Vorlesung Sicherheit: Fragen und Lösungsansätze (2+1 SWS, INF-BSc-302)
- Fachprojekt „Werkzeuge für modellbasierte Sicherheitsanalysen für sicherheitskritische IT-Architekturen“
- Proseminar Werkzeugunterstützung für sichere Software
- Seminar Sicherheit und Softwareengineering (2 SWS)
- DiDo-Seminar Oberseminar (2 SWS)

Morgen: Prüfungsvorbereitung