Softwareengineering und Komplexität

Christian Silberbauer

Einführung

Christian Silberbauer

Motivation

- Wir leben in einer komplexen Welt.
- Verständnis von Einfachheit kann Orientierung schaffen.

- Gemeinsame Theoriebasis erleichtert die Nutzung von Synergieeffekten verschiedener Denkrichtungen, Schulen, Disziplinen.

Motivation

- Die Komplexitätsbewältigung im Softwareengineering ist in weiten Teilen unsystematisch.
- Sie orientiert sich an Best-Practices.
- Es fehlt and systematischen Entscheidungsgrundlagen für gute Softwareentwicklung.

- Komplexitätsbewältigung ist nicht softwarespezifisch, aber hochgradig relevant.

Beispiel: Morgenroutine

- Was ist zu tun?
- In welcher Reihenfolge?

- Ist das denn immer gleich?
- Überhaupt darüber nachdenken?

Das Problem der Wissenschaften

- Die wissenschaftlichen Disziplinen haben sich seit geraumer Zeit mehr und mehr ausdifferenziert.
- Gemeinsamkeiten verlieren an Bedeutung.
- Welche Wissenschaft referenziert heute noch auf die Philosophie?
- Fachdisziplinen verkommen zu Inseln.
- Dies ist begründet durch die Komplexität des Wissens.

Das Problem der Wissenschaften

- Überschneidungen sind offensichtlich:
  - Psychologie und Soziologie
  - Philosophie und Quantenphysik
  - Linguistik und Psychologie

Das Problem der Wissenschaften

- Man verheddert sich in Paradoxien:
  - Russellsche Antinomie (Menge aller Mengen, Barbier-Paradox)
  - Quantenverschränkung
  - Heißenbergsche Unschärferelation (Grenzen der Messgenauigkeit)
  - Gödelsche Unvollständigkeitssätze (Grenzen der Beweisbarkeit)
  - Entscheidungsproblem (Leibniz, Hilbert, Church, Turing)

Das Problem der Wissenschaften

- Man verliert sich in übermäßiger Komplexität, bspw. String-Theorie:
  - Man bindet gewaltige Ressourcen an hochkarätigen Physikern.
  - Sie ist bisher wenig hilfreich.
  - Und: zur Weltformel (Vereinigung aller Kräfte) führt sie ohnehin nicht, weil Gravitation außen vor bleibt.

Problem „Teilchen“

- Die Suche nach kleinsten Teilchen hat in der Physik Tradition.
- In der Mathematik gelten Zahlen als elementar.

- Allerdings: aus einer Dynamik lässt sich eine relative Statik ableiten. Umgekehrt aber nicht. Unbewegliches wird nicht relativ beweglich.

- Daraus müsste aber folgen, dass nicht Teilchen, Zustand oder Zahlen elementar sind, sondern Bewegung, Dynamik, Energie bzw. **Kommunikation**.

Komplexität

- Dinge weniger komplex handhaben zu müssen, ist ein erstrebenswertes Ziel.
- Die Informatik ist hierbei interessant.
- Sie wurde begründet von Mathematikern und Ingenieuren.
- Software wird recht schnell sehr komplex.
- Die Komplexität des Programmierens maß man anhand der Komplexität des Programms.
- Man wollte aus der Programmkomplexität die Programmierkomplexität ableiten.
- Dazu wurde eine Vielzahl von Softwaremetriken erfunden (LoC, Zyklomatische Komplexität, Halstead-Metrik…)
- Aber wenn komplexe Anforderungen zu komplexen Programmen führen, ist daran doch erstmal nichts verkehrt…?

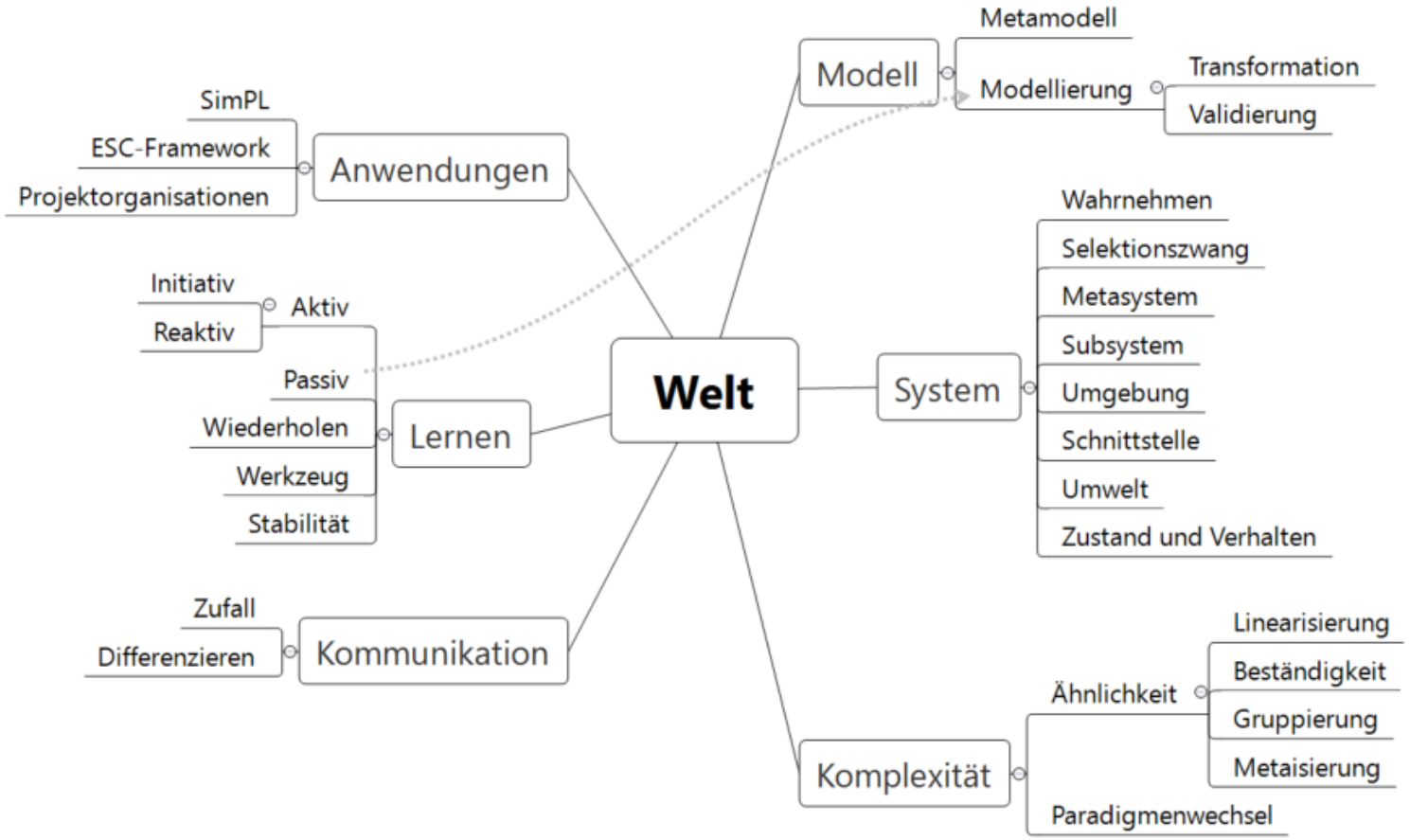
Ansatz

- Angenommen wird eine Welt basierend auf Kommunikation.
- Ein Theoriegebilde zum Umgang mit Komplexität wird vorgestellt mit dem Anspruch diese möglichst einfach und möglichst vollständig in seiner Grundstruktur darzustellen.

Die kurze Antwort:

Orientiere dich meist an Ähnlichem
und manchmal an Unähnlichem.

Überblick


Welt
Modell
Metamodell
Modellierung
Transformation
Validierung
Anwendungen
SimPL
ESC-Framework
Projektorganisationen
Lernen
Initiativ
Aktiv
Reaktiv
Passiv
Wiederholen
Werkzeug
Stabilität
System
Wahrnehmen
Selektionszwang
Metasystem
Subsystem
Umgebung
Schnittstelle
Umwelt
Zustand und Verhalten
Kommunikation
Zufall
Differenzieren
Komplexität
Ähnlichkeit
Linearisierung
Beständigkeit
Gruppierung
Metaisierung
Paradigmenwechsel

Überblick

Theorie:

- 1. Einführung
- 3. System
- 4. Abstraktion
- 6. Lernen
- 7. Modell
- 9. Komplexität
- 10. Grenzen
- 11. System Lernen

Software:

- 2. Softwaremetriken
- 5. Frameworks
- 8. Toolchain
- 12. SimPL
- 13. ESC-Framework
- 14. Muster
- 15. UML
- 16. Projektorganisation