Agenda

1. FAQ zur Programmierung
2. Übersicht der Themen
3. Grundlagen Softwaretheorie
4. Grundlagen Programmiersprachen
5. **Software-Entwicklungsprozesse**
6. Software-Entwurftools

> Danach: Praktisches Programmieren (Python)

5. Software-Entwicklungsprozesse

Übung zu Vorgehensmodellen

1. Trefft euch mit euren Gruppenmitgliedern in den Nebenräumen (Entsprechend der Gruppennummer, z.B. Gruppe A -> Nebenraum A)
2. Einigt euch auf ein (fiktives) Projekt:
   - Muss kein Software-Projekt sein!
   - Ideen:
     - Etwas, was ein Gruppenmitglied vor Kurzem durchgeführt hat
     - Entwicklung eines neuen Features für eine Webapp
     - …
3. Wählt eines der beiden Modelle (Wasserfall / V) und bearbeitet folgende Aufgaben:
   a) Warum habt ihr euch für das Modell entschieden?
   b) Wie lässt sich das Projekt mit dem gewählten Modell erfolgreich umsetzen?
   c) Was sind mögliche Fallstricke?
   d) BONUS: Was müsste sich am Modell ändern, damit euer Projekt noch erfolgreicher wird?

5. Software-Entwicklungsprozesse

Gegenentwurf: Extreme Programming (XP)

- Wenig Formalitäten (agil), daher kein einheitliches Modell
- Stellt die Softwareentwicklung in den Vordergrund
- Ablauf: Kein *großes Projekt* sondern iterative Prozesse (Zyklen)
  - Am Anfang eines Zyklus (oft 2 Wochen) werden Anforderungen gesetzt
  - Am Ende wird überprüft und besprochen ob die Ziele erreicht worden sind
- Werte: Kommunikation, Einfachheit, Feedback, Mut, Respekt
- Rollen: Kunde, Manager, Entwickler und Coach
- Vorteile: Flexibilität, Kunde ist stark involviert, Viel Tests
- Nachteile: viele Meetings, Kunde ist stark involviert, Selbstdisziplin

5. Software-Entwicklungsprozesse

Gegenentwurf: Extreme Programming (XP)

- Durch mangelnde Formalität kann XP zusätzliche Belastung für Entwickler bedeuten!
- Deutlich machen:
  - Was sind die Anforderungen eines Zyklus? (Überprüfbarkeit!)
  - Wann gilt der Zyklus als erfolgreich?
  - Sich nicht übernehmen!
- Eure Meinung: Wäre XP für euer Projekt *besser* als Wasserfall/V?

5. Software-Entwicklungsprozesse

Das beste aus beiden Welten: SCRUM

- „angeordnetes Gedränge“ (aus dem Rugby)
- Agile Entwicklung aber klare Rollen, Meetings, Werkzeuge

Rollen:

- Product Owner (1 Person):
  - Interessen (Entwickler/Kunde) balancieren, Ziele formulieren & kommunizieren
- Entwickler (<10 Personen, eher 3-8):
  - Setzt Aufgaben um, kommuniziert frühzeitig Probleme
- Scrum Master (1 Person):
  - Coach: Achtet darauf, dass *Spielregeln* eingehalten werden, beseitigt (formelle) Hindernisse

5. Software-Entwicklungsprozesse

Das beste aus beiden Welten: SCRUM

- Sprint: Entwicklungszyklus mit eindeutig definierten Aufgaben und bestimmter Dauer (ca. 1 – 4 Wochen)
- Jeder Sprint besteht aus:
  - (Refinement)
  - Planning (1x pro Sprint)
  - Daily (Jeden Tag)
  - Review (1x)
  - Retro (1x)

5. Software-Entwicklungsprozesse

Das beste aus beiden Welten: SCRUM

Meetings:

- Refinement (optional)
  Ziel: Synchronisation der Anforderungen vor dem Planning (Pre-Planning)
  Teilnehmer: PO, Entwickler (optional Scrum Master)
- Planning
  Ziel: Planung des Sprints (zu Beginn des Sprints)
  Ablauf: Schätzen des Aufwands der Aufgaben und Ziele eines Sprints
  Entwickler schätzen, wie viele Stunden bestimmte (Teil-)aufgaben dauern
  Teilnehmer: Alle

5. Software-Entwicklungsprozesse

Das beste aus beiden Welten: SCRUM

Meetings:

- Daily
  Ziel: Abstimmen über erledigte & anstehende Aufgaben (bis nächster Daily)
  Kurz halten! (ca. 15 min)
  Jeder Teilnehmer (Entwickler, Scrum Master, evtl. PO) muss was sagen
- Review
  Am Ende eines Sprints, Präsentation der Ergebnisse
  Ausreichend Zeit nehmen (ca. 4 Stunden)
  Teilnehmer: Alle, zusätzlich Kunden
- Retrospektive
  Reflektion der Zusammenarbeit: Was lief gut/nicht gut? Wie verbessern?
  Teilnehmer: Alle

5. Software-Entwicklungsprozesse

Das beste aus beiden Welten: SCRUM

SCRUM-Werkzeuge:

- Product Backlog
  Übersicht aller anstehenden Aufgaben
  Wird von PO gepflegt
  Von allen einsehbar, inkl. Kunden
- Product Backlog Item (User Stories)
  Eine einzelne Aufgabe
  von kleinen Bugfixes bis *Epics* (Anforderungen mit großen Auswirkungen)
  Enthalten Akzeptanzkriterien

5. Software-Entwicklungsprozesse

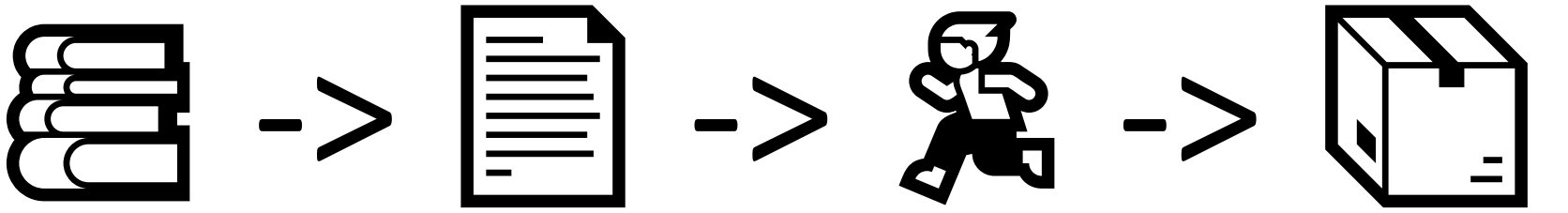
Das beste aus beiden Welten: SCRUM

SCRUM-Werkzeuge:

- Sprint Backlog
  Alle Aufgaben des aktuellen Sprints
  Entwickler *entscheiden* über den Inhalt
- Story Points
  Aufwand einer Aufgabe (mehr Points -> Aufgabe komplexer/aufwendiger)
- Planning Poker
  Spielerisches Ermitteln des Aufwands einer Aufgabe:
  Zu jeder Aufgabe werden Story Points vergeben
  Mögliche Werte: Fibonacci Reihe (1, 2, 3, 5, 8, 13, 21, …)
  Alle decken zeitgleich auf (-> Verhindern von Beeinflussung/*Group think*)

5. Software-Entwicklungsprozesse

Das beste aus beiden Welten: SCRUM



5. Software-Entwicklungsprozesse

Das beste aus beiden Welten: SCRUM

- Vorteile:
  - Kombiniert Struktur (Wasserfall-/V-Modell) mit Flexibilität (XP)
  - Aufteilung in Teilziele -> Verbesserung der Übersicht
  - Einbindung des Entwicklers in die Planung
- Nachteile:
  - Viel „drumherum“: Meetings können nichtssagend bis Zeitverschwendung sein
  - Setzt gute Kommunikation und Vertrauen voraus
  - Entwickler machen mehr als nur Entwicklung

5. Software-Entwicklungsprozesse

FAQ zu Vorgehensmodellen

Frage: *„Für welches Modell sollte ich mich denn jetzt entscheiden?“*

Antwort: *„Entwickler haben meist keinen Einfluss auf Auswahl. Generell gilt: Typsache!*
Wer lieber mehr Freiheiten genießt -> XP, SCRUM
Wer lieber auf klare Strukturen zurückgreift -> Wasserfall, V“

Frage: *„Gibt es Abwandlungen von den vorgestellten Modellen?“*

Antwort: *„Ja, im Grunde hängt es sehr stark von der Unternehmensphilosophie ab (Konzern vs. Startup).“*