Freie Universität Berlin

Lutz Prechelt

Softwaretechnik, SoSe21

Übung 12

TutorIn: Samuel Domiks
Tutorium 02
Materialien: Latex, Skript

Jonny Lam & Thore Brehmer

5. Juli 2021

---

1 Auswahl von Prozessmodellen

(a) *Fassen Sie noch einmal kurz die Unterschiede zwischen den drei Prozessmodellarten wasserfallartig, evolutionär, und inkrementell zusammen.*

- **wasserfallartig:** Aktivitäten werden nach Reihe durchlaufen, wobei jede Aktivität nur einmal durchgelaufen wird und Aktivität N erst nach N-1 Aktivitäten beginnt.
- **evolutionär:** System wird Schrittweise gebaut. In jedem Schritt werden neue Aktivitäten hinzugefügt und im Gegensatz zum inkrementellen Modell auch existierende Sachen verändert.
- **inkrementell:** System wird Schrittweise gebaut. In jedem Schritt werden nur neue Aktivitäten hinzugefügt, theoretisch wird nie etwas Existierendes verändert.

(b) *Wählen Sie für jedes der folgenden zu bauenden Systeme die am besten geeignete Prozessmodellart (aus den in a) genannten) und begründen Sie Ihre Wahl.*

1 **Ein Terminal als digitaler, interaktiver Ersatz für Papierfahrpläne in größeren Bahnhöfen (für Ankunft- und Abfahrtszeiten).**
Hier eignet sich das evolutionäre Modell ganz gut, da man hier Fahrpläne hinzufügen kann und im Gegensatz zu den anderen Modellen auch existierende Fahrpläne ändern kann. Was aufjedenfall nötig ist, da das Terminal ja interaktiv sein soll und neue wie auch alte Fahrpläne anzeigen soll.

2 **Eine Steuerungseinheit für das Antiblockiersystem (ABS) in PKWs.**
Hier eignet sich das inkrementelle Modell, da die Abfolge für das ABS festgelegt ist. Hier ist es nicht nötig die Abfolge zu verändern, im Gegensatz zum Wasserfall Modell muss die Aktivität mehr als einmal ausgeführt werden (z.B. bei jeder Bremsung, da man das ABS öfter braucht).

3 **Ein System zur Verwaltung von Lehrveranstaltungen, wie etwa unser KVV.**
Hier eignet sich das Wasserfall Modell, da jede Lehrveranstaltung nur einmal stattfindet (bzw. nur einmal zu einem bestimmten Zeitpunkt) und Lehrveranstaltung N erst nach N-1 Lehrveranstaltungen stattfindet (z.B. findet die 3. Vorlesung erst nach 1 und 2 statt).

Quellen:

[1] Vorlesung 17

2 Wasserfall vs. Agile Prozesse

(a) *In den letzten fünfzehn Jahren wurde das Wasserfallmodell stark kritisiert und mehr „Agilität“ gefordert. Zuvor war jedoch ein wasserfallartiges Vorgehen (fast) immer als das ideale Vorgehen beschrieben worden. Erklärung Sie diese Entwicklung:*

1 Warum kann man das Wasserfallmodell einerseits durchaus als ideal bezeichnen?
Am Ende jeder Phase liegen alle Ergebnisse in Dokumenten vor und diese Dokumente werden gründlich geprüft und in die nächste Phase übergeben. Man nimmt an, dass die Mängel in Phase N spätestens in Phase N+1 aufgedeckt werden und dann leicht in den Dokumenten beider Phasen (N und N+1) korrigiert werden.

2 Und warum hat man sich andererseits dennoch davon gelöst? Nennen Sie mindestens zwei Punkte, an denen Wasserfall-Projekte scheitern können, und die bei agilen Projekten zumindest stark abgefedert werden.

- **Unklare Anforderungen**: Wenn die Anforderungen nicht verstanden wurden führt das im Wasserfallmodell zum Chaos, da späte Änderungen der Anforderungen das Prozessmodell durcheinander bringen. Zum Beispiel muss man dann viel in den Dokumenten ändern, was enorm teuer wird oder die Dokumente werden nicht mehr korrekt gepflegt.
- **Veränderliche Anforderungen**: Das gleiche wie bei den unklaren Anforderungen. Bei Agilen Prozessen werden die beiden Punkte stark abgefedert.

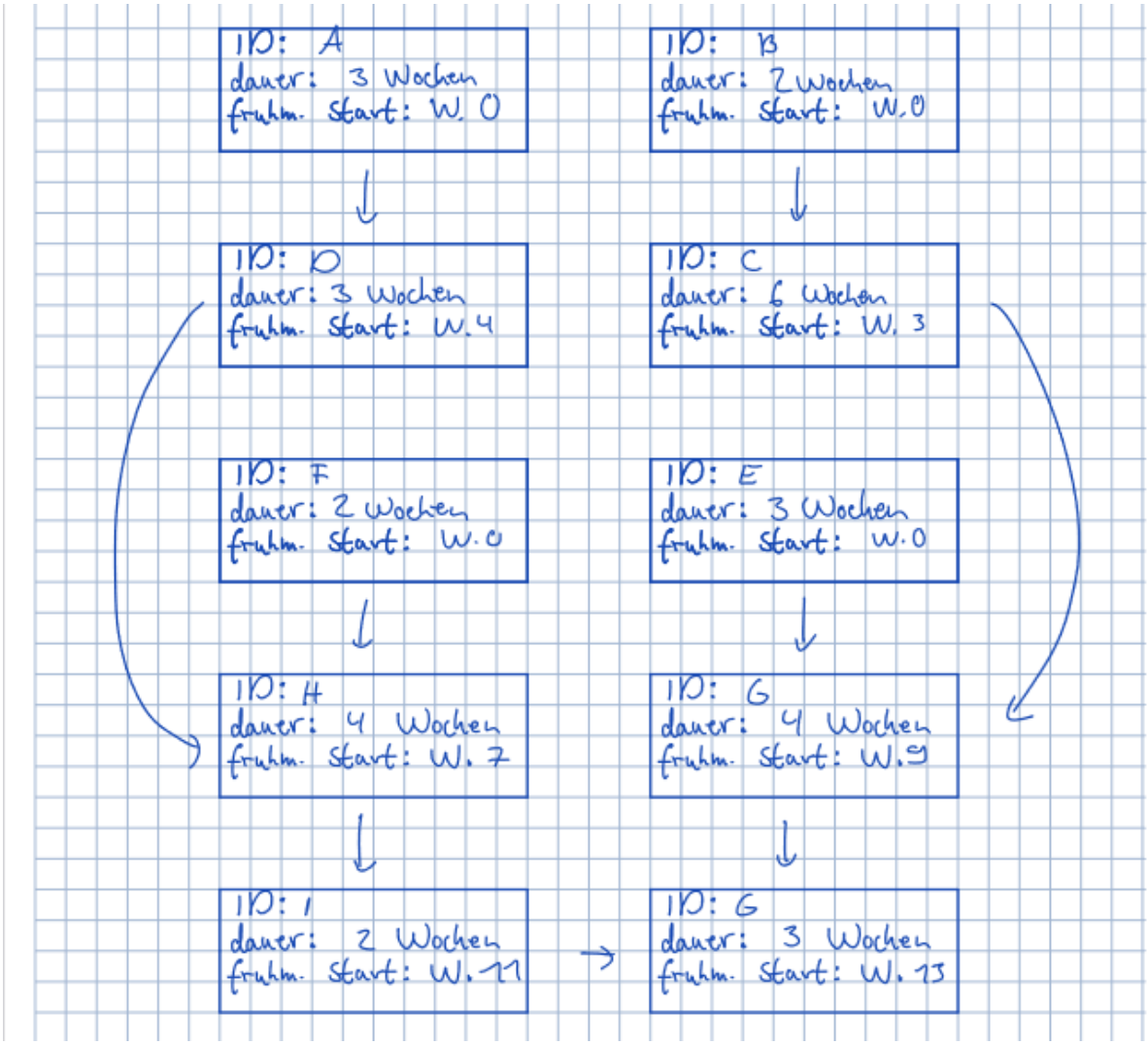
(b) *Füllen Sie den folgenden Lückentext mit passenden Begriffen. Falls nötig, nutzen Sie dazu die auf der Vorlesungswebseite angegebenen Quellen.*
Zeitlich ist ein Projekt, das mit Extreme Programming (Abk. **XP**) durchgeführt wird, in **Praktiken** eingeteilt. Diese enden jeweils mit einer neuen Version des Softwaresystems. Am **Anfang** einer jeden Iteration besprechen der Kunde und die Entwickler gemeinsam, welche Funktionalitäten realisiert werden sollen. Der Kunde formuliert dabei seine Wünsche auf den (engl.). Während der gesamten Entwicklung ist der Kunde **vor Ort** . Er definiert zudem **Anforderungen** , um am **Ende** jeder Iteration die Funktionalität testen zu können. Für die Entwickler gibt Extreme Programming zusätzlich eine Reihe von Praktiken vor, wie etwa **Short Releases**, **Testing** oder die gemeinsame Verantwortung.

Quellen:

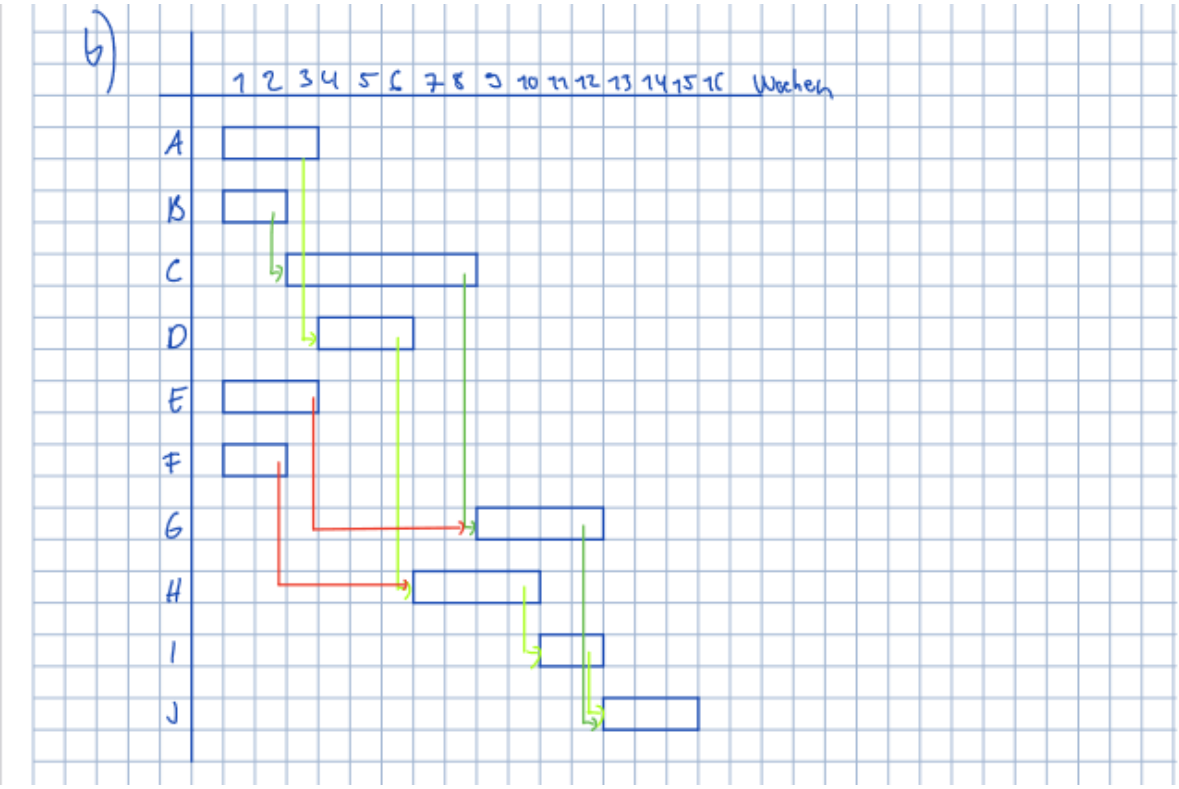
[1] Vorlesung 17

3 Projektplanung

(a) *Erstellen Sie einen Netzplan, der die logischen Abhängigkeiten der Aufgaben untereinander sichtbar macht. Stellen Sie jede der o.g. Aufgaben als Rechteck dar und tragen Sie die ID, die Aufgaben dauer und den frühst möglichen Start ein.*



(b) *Erstellen Sie ein Gantt-Chart, das die zeitlichen Abhängigkeiten der Aufgaben sichtbar macht.*



(c) *Ermitteln Sie den/die kritischen Pfad/e, die kürzeste Projektlaufzeit, und für alle Aktivitäten jeweils die Pufferzeit (slack time).*

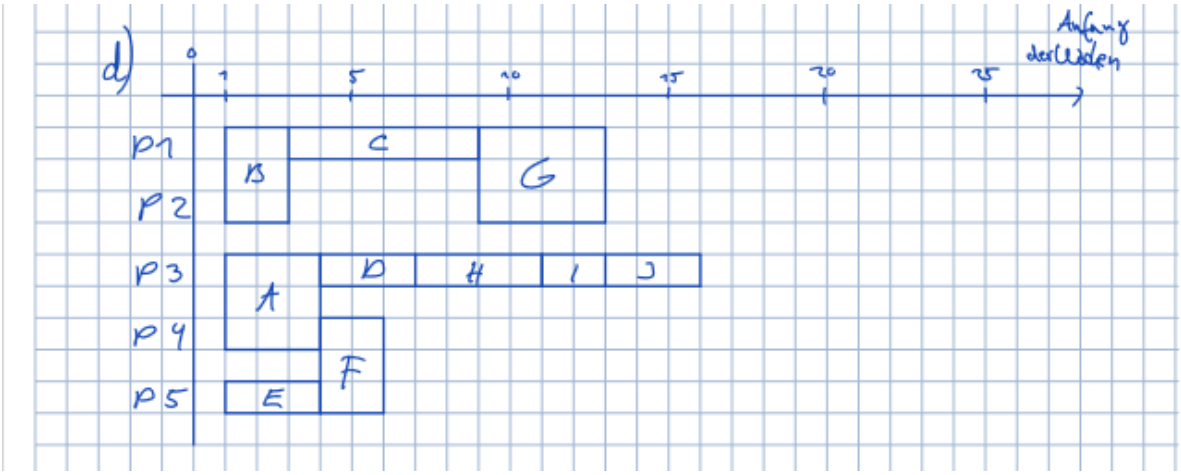
- **kritische Pfade** aus Aufgabe b) der hell grüne und dunkel grüne Pfad.
- **kürzeste Projektlaufzeit** aus Aufgabe b) ersichtlich: 15 Wochen

Aktivitäten	**Pufferzeit (slack time)** in Wochen
A	0
B	0
C	2
D	3
E	0
F	0
G	8
H	6
I	10
J	12

(d) *Nehmen Sie hypothetisch an, dass Sie den Projektablauf (bei gegebenen Abhängigkeiten) bestmöglich parallelisieren wollen, also die Projektlaufzeit minimieren wollen. Wie hoch wäre der Personalbedarf Pmax, also die größte sinnvolle Teamgröße ab der weitere Personen keine Beschleunigung mehr ergeben? Stellen Sie eine mögliche Aufgabenverteilung für n = Pmax graphisch dar (etwa wie in Abb. 1). Geben Sie für diese Verteilung auch jeweils die Pufferzeiten aller Aufgaben, sowie die Projektlaufzeit an.*

- **Pmax** = 5 = n
- **Projektlaufzeit**: 15 Wochen

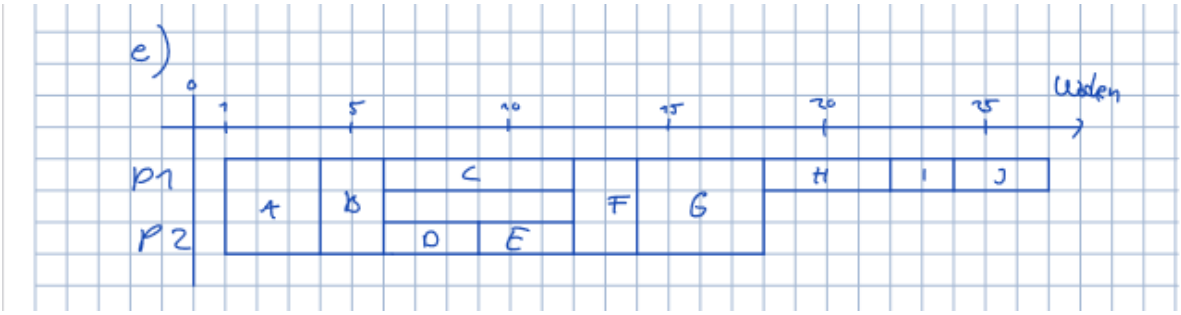
Aktivitäten	**Pufferzeit (slack time)** in Wochen
A	0
B	0
C	2
D	3
E	0
F	3 (einzige Änderung)
G	8
H	6
I	10
J	12



(e) *Nehmen Sie an, Sie könnten nur zwei Entwickler/innen für dieses Projekt abstellen. Was wäre hier die kürzeste Projektlaufzeit? Stellen Sie eine mögliche Aufgabenverteilung für n = 2 graphisch dar. Geben Sie auch für diese Verteilung die Pufferzeiten und die Gesamtlaufzeit an.*

- **n** = 2
- **Projektlaufzeit**: 26 Wochen

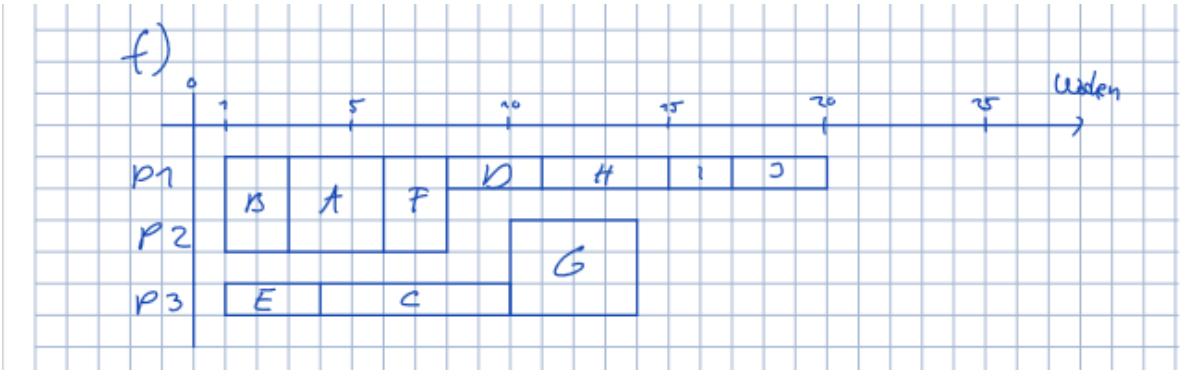
Aktivitäten	**Pufferzeit (slack time)** in Wochen
A	0
B	3
C	5
D	3
E	8
F	11
G	13
H	17
I	21
J	23



(f) *Stellen Sie Aufgabenverteilungen für alle weiteren möglichen Teamgrößen n (d.h. also 2 < n < Pmax) graphisch dar, sodass jeweils die Projektlaufzeit möglichst kurz ist. Geben Sie wiederum für jede Verteilung die Projektlaufzeit und die Pufferzeiten an.*

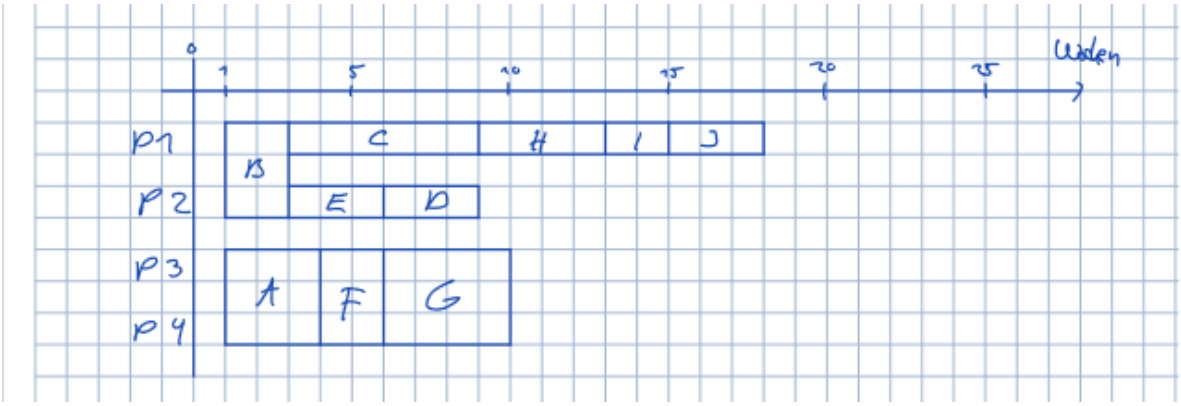
- **n** = 3
- **Projektlaufzeit**: 19 Wochen

Aktivitäten	**Pufferzeit (slack time)** in Wochen
A	0
B	2
C	9
D	10
E	3
F	7
G	13
H	14
I	16
J	19



- **n** = 4
- **Projektlaufzeit**: 17 Wochen

Aktivitäten	**Pufferzeit (slack time)** in Wochen
A	0
B	0
C	2
D	5
E	2
F	3
G	5
H	8
I	12
J	17



(g) *Betrachten Sie Ihre möglichen Projektabläufe (also für alle Teamgrößen von 2 bis einschließlich Pmax) und fällen Sie eine Entscheidung: Wie viele Entwickler/innen setzen Sie nun auf dieses Projekt an? Erläutern Sie Ihre Entscheidung: Vergleichen Sie Ihre Optionen explizit nach mindestens drei Gesichtspunkten.*

- Wir haben folgende Gesichtspunkte mit in Betracht gezogen:
  1. Teamgröße. (Weniger Mitarbeiter -> Weniger Einarbeitung, Kommunikation und Kosten.)
  2. Projektlaufzeit. (Je kürzer desto besser)
  3. slack time. (Auch von Mitarbeitern. Mitarbeiter sollen möglichst immer Arbeit haben)
- Aufgrund der oberen Gesichtspunkte haben wir uns für 3 Mitarbeiter entschieden. Hier finden wir das Verhältnis von Projektlaufzeit (19W) und slack time (der Mitarbeiter (12W)) am besten.
- Bei 2 Mitarbeitern wäre uns die Projektlaufzeit zu lang (26W anstatt nur 19W). Und bei 4 Mitarbeitern ist uns der Austausch von mehr slack time zu weniger Projektlaufzeit nicht wert. (Projektlaufzeit verkürzt sich nur um 2 Wochen. Dafür haben die Mitarbeiter nun insgesamt 25 Wochen keine Arbeit nur 12 Wochen)