Projektmanagement

Authors of slides:
Norbert Siegmund
Janet Siegmund
Oscar Nierstrasz
Sven Apel

Lernziele

- Aufgaben während Projektmanagement verstehen
- Nötiges Wissen über die Formalitäten von Projekten erfahren
- Zeitpläne für Projekte erstellen
- Grundlegendes Verständnis für Risikomanagement haben

Warum Projektmanagement?

- So gut wie jedes Software Produkt wurde innerhalb eines **Projektes** erstellt (im Gegensatz zum produzierenden Gewerbe)

  Projektherausforderung = **rechtzeitige Auslieferung** im **festgelegten Budget**

- Kernmerkmale eines Projektes
  - Zeitlich abgeschlossen
  - Definiertes Ziel
  - Einmaliges Unterfangen

Was ist Projektmanagement?

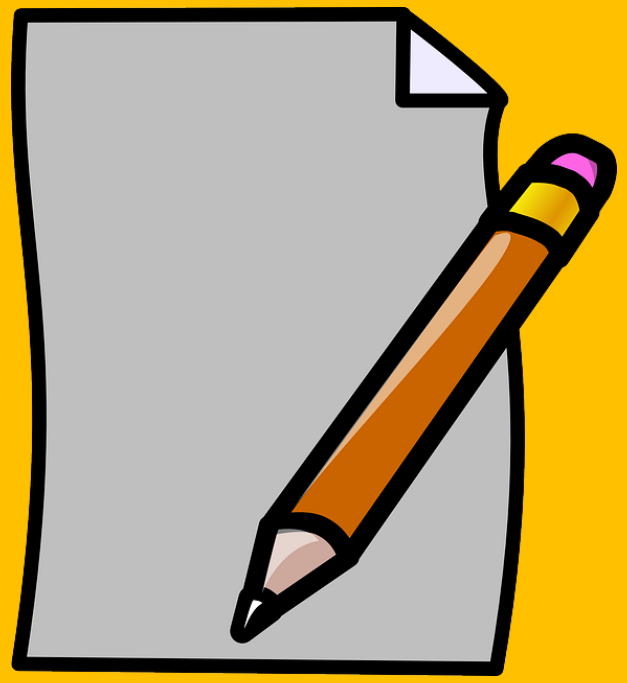
Project Management = **Plan the work** and **work the plan**

- Managementfunktionen:
  - **Planung**: Abschätzung und zeitl. Einteilung von Ressourcen
  - **Organisation**: Wer macht was?
  - **Mitarbeiter:innen**: Rekrutierung von motivierten Mitarbeiter:innen
  - **Dirigieren**: Sicherstellung, dass das Team zusammenarbeitet
  - **Monitoring** (Controlling): Erkenne Abweichungen im Plan und korrigiere Aktionen

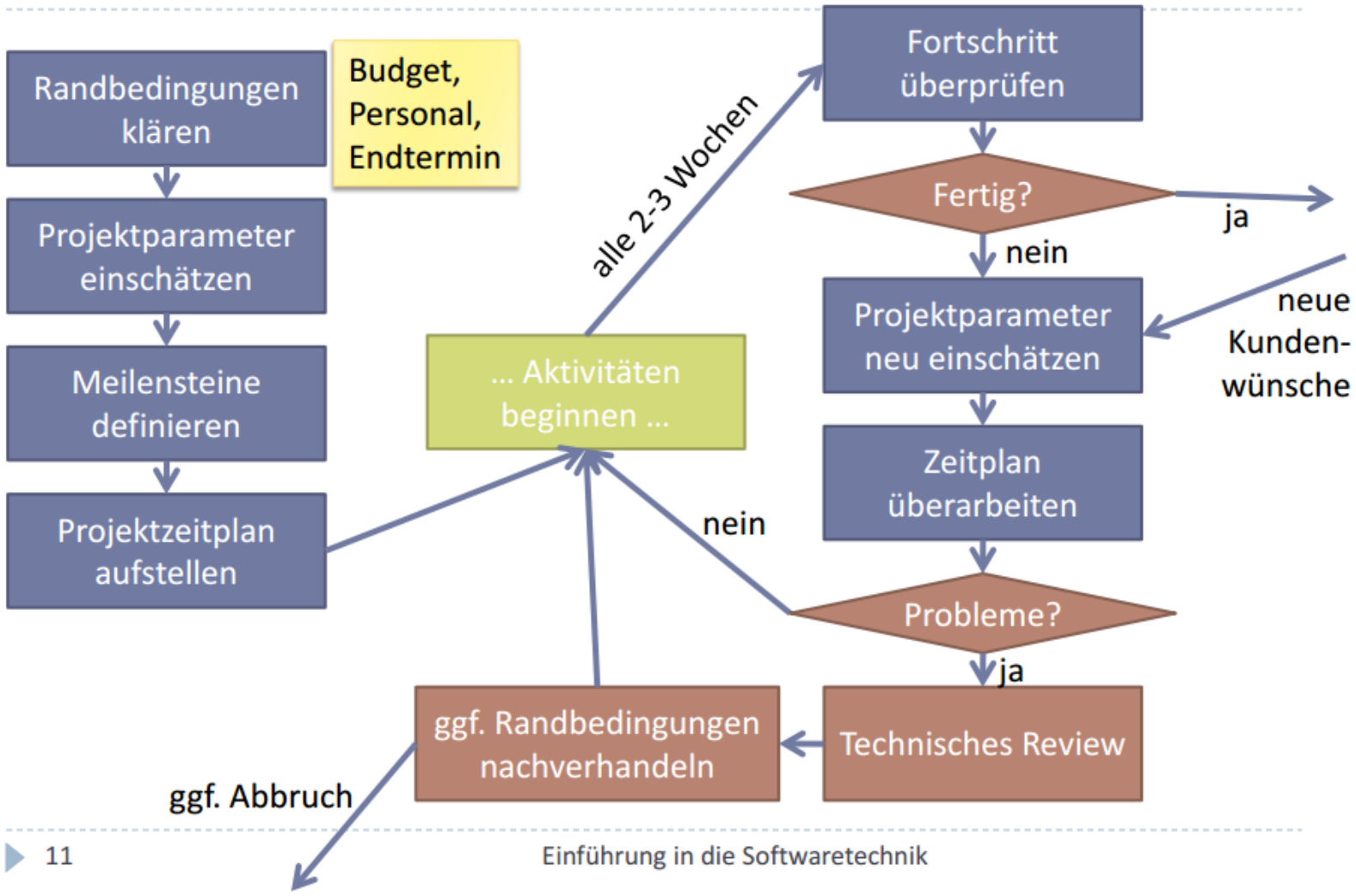
Aufgaben während Projektmanagement

- Googledoc

Projektplanung



Projektplanung



Projektplan

- Einführung: Ziele und Randbedingungen festlegen (Pflichtenheft)
- Projektorganisation: Personen, Rollen, Teams
- Risikoanalyse: Beschreibung und Bewertung von Risiken
- Arbeitsaufteilung, Verantwortlichkeiten, Weisungsbefugnisse
- Projektzeitplan: Wer, wann, was? Meilensteine, Lieferschritte
- Überwachungs- und Berichterstattungsinstrumente: Wann und wie wird geprüft und berichtet?

- Der Projektplan wird während des Projekts angepasst

Meilensteine

- Erkennbarer Endpunkt einer Teilaufgabe
- Für Projektmanager:in zur Überwachung/Überprüfung des Fortschritts
- Berichte, Prototypen, fertige Teilsysteme
- Überprüfbarkeit:
  - “Implementierung zu 80% abgeschlossen” kein geeigneter Meilenstein
  - Besser: Anforderung X erfüllt

Lieferschritte

- Projektresultat für Kund:innen
- Ähnlich Meilenstein
- Berichte, Prototypen, fertige Teilsysteme

- Sollten genau wie Meilensteine etwa alle 2-3 Wochen fällig sein

Lastenheft

- Spezifiziert durch Kunden / **Auftraggeber**
- Beschreibt Sicht des Auftraggebers
  - Was ist der IST-Zustand und was sind Gründe für das Projekt?
  - Was sind die Ziele des Projektes?
  - Welche Anforderungen gibt es (Katalog, Spezifikation)?

- Wird oft in Ausschreibungen verwendet
- Anforderungen sind sehr allgemein und wenig beschränkend formuliert

Möglicher Aufbau eines Lastenheftes

- Einführung
- Beschreibung des Ist-Zustands
- Beschreibung des Soll-Konzepts
- Beschreibung von Schnittstellen
- Funktionale Anforderungen
- Nichtfunktionale Anforderungen
  - Benutzbarkeit, Zuverlässigkeit, Effizienz, Änderbarkeit, etc.
- Risikoakzeptanz
- Skizze des Entwicklungszyklus und der Systemarchitektur oder auch ein Struktogramm
- Lieferumfang
- Abnahmekriterien

Pflichtenheft

- Spezifiziert durch **Auftragnehmer**
  - Fasst alle Anforderungen **konkret und vollständig** zusammen
  - Bildet Grundlage für vertraglich festgehaltene Leistungen
  - **Präzisiert** das Lastenheft und beschreibt **wie** die Anforderungen aus dem Lastenheft realisiert werden

- Folgende Punkte sind enthalten:
  - Funktionale Anforderungen (inkl. Datendefinitionen)
  - Nicht-funktionale Anforderungen (Performance, …)
  - Anforderungen an technische Realisierung (welche HW/OS,…)
  - Anforderungen an Projektablauf (Meilensteine, Risiko,…)
  - Benutzungsschnittstelle (Wie Präsentation)

Zeitplanung

Zeitplanung

- Zerlegt Projekt in Arbeitspakete (Dauer 1 bis 10 Wochen)
- Arbeitspakete klein genug wählen, dass realistische Kostenschätzung möglich ist
- Abhängigkeiten zwischen Arbeitspaketen definieren und minimieren
- Schätzt Zeiten und Ressourcen
- Erstellt sinnvolle Reihenfolge und Parallelität
- Zeitpuffer einplanen, eventuelle Probleme berücksichtigen
- Softwareunterstützung hilfreich, z.B. Microsoft Project, GanttProject, Kplato, uvm.

Was ist die minimale Projektdauer?

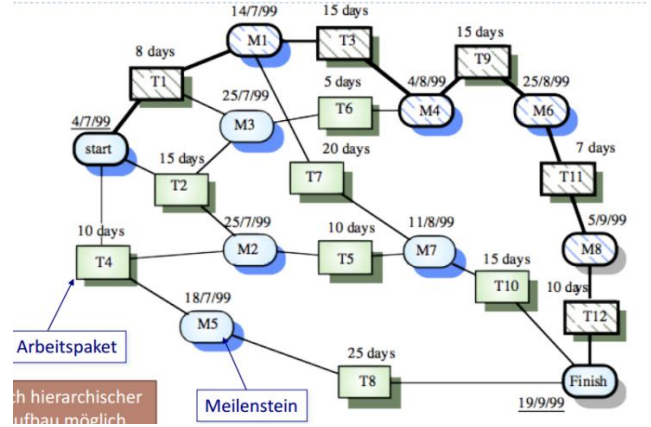
Arbeitspaket	Dauer in Tagen	Abhaengigkeiten
T1	8	
T2	15	
T3	15	T1
T4	10	
T5	10	T2, T4
T6	5	T1, T2
T7	20	T1
T8	25	T4
T9	15	T3, T6
T10	15	T5, T7
T11	7	T9
T12	10	T11

Netzplan

- Google-Zeichnung

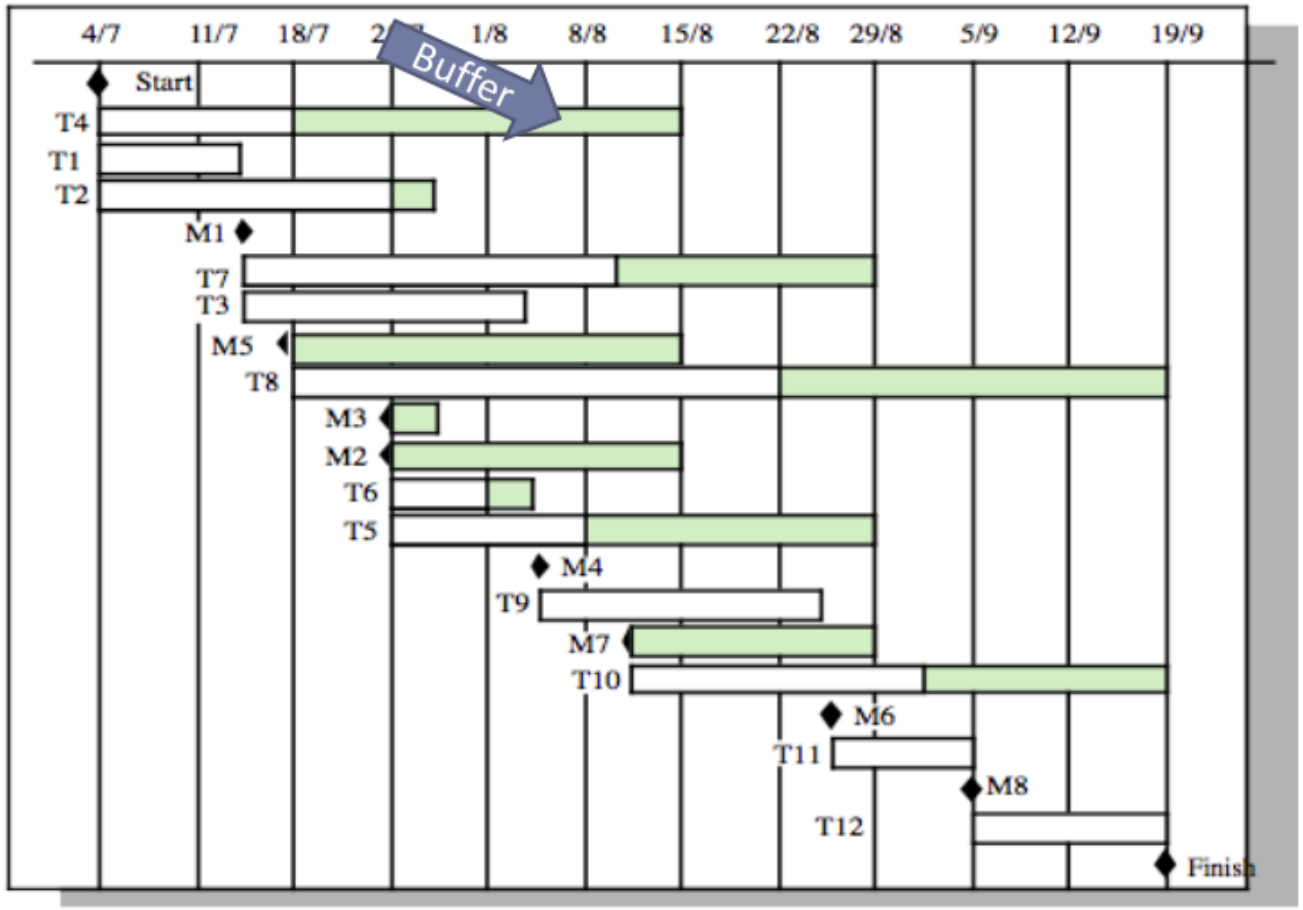
Kritischer Pfad

- Längster Pfad im Netzplan:
  - 55 Tage
  - Puffer T8: 20 Tage

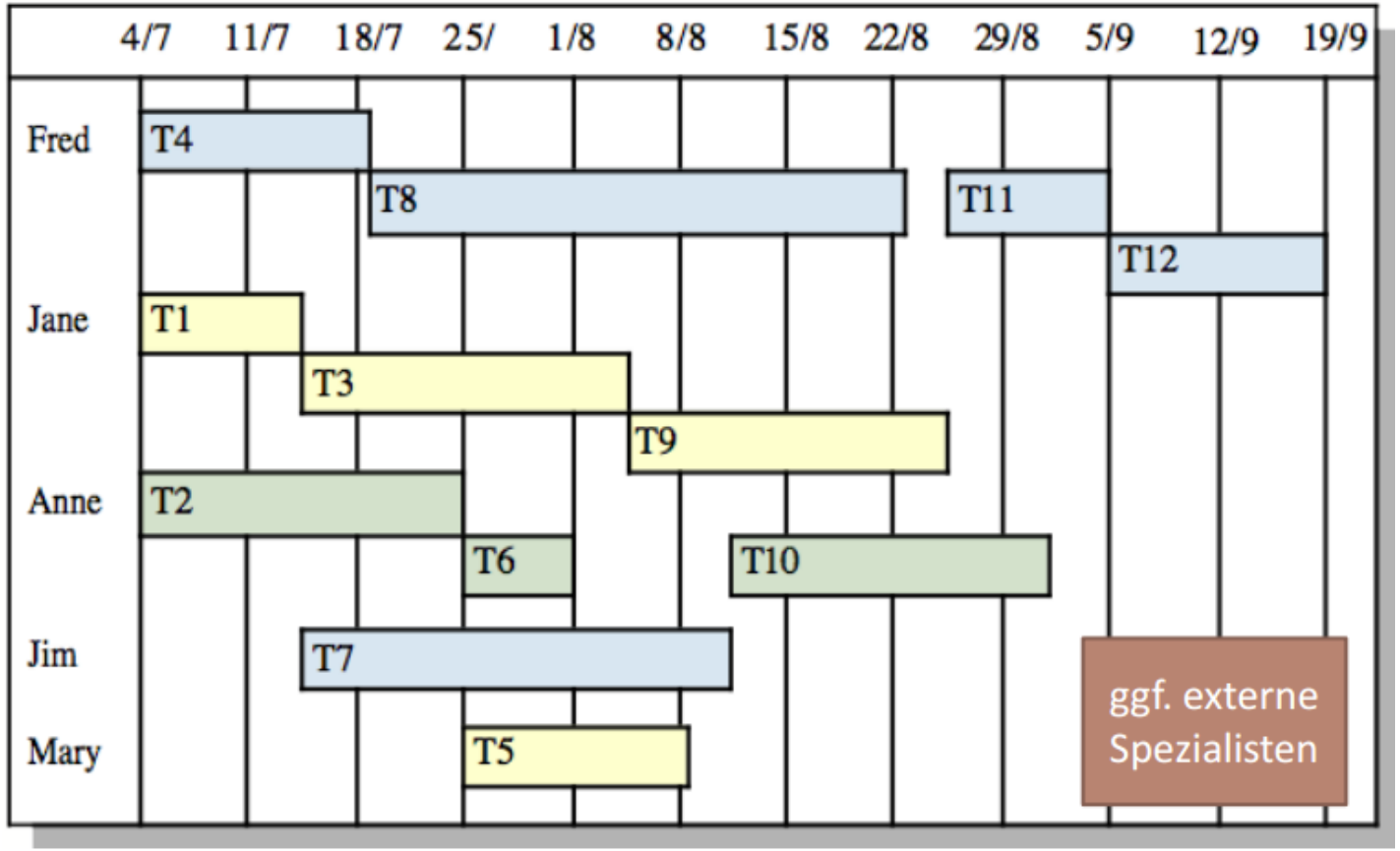


- Verzögerung vom Paketen auf kritischem Pfad -> Gesamtverzögerung
  - Dort besonders genau planen
  - Zeiten ggf. verkürzen durch Projektaufgaben umstrukturieren;
  - Pessimistisch planen
- Andere Pakete ggf. unkritisch, berechenbarer Puffer

Gantt-Diagramm



Gantt-Diagramm für Ressourcen



Zeitplanung

- Zeitplan ändert sich ständig
- Erfahrung zum Schätzen notwendig
- Trotzdem schwierig durch Neuartigkeit des Projekts und schnell wechselnde Technologie
- Vergleich mit ähnlichen Projekten zur besseren Zeitplanung (sinnvoll, diese in einer Datenbank zu speichern)

Reagieren auf Zeitprobleme

- ***Myth:***
  - *“If we get behind schedule, we can add more programmers and catch up.”*
- ***Reality:***
  - *Adding more people typically slows a project down.*

Zeitprobleme I

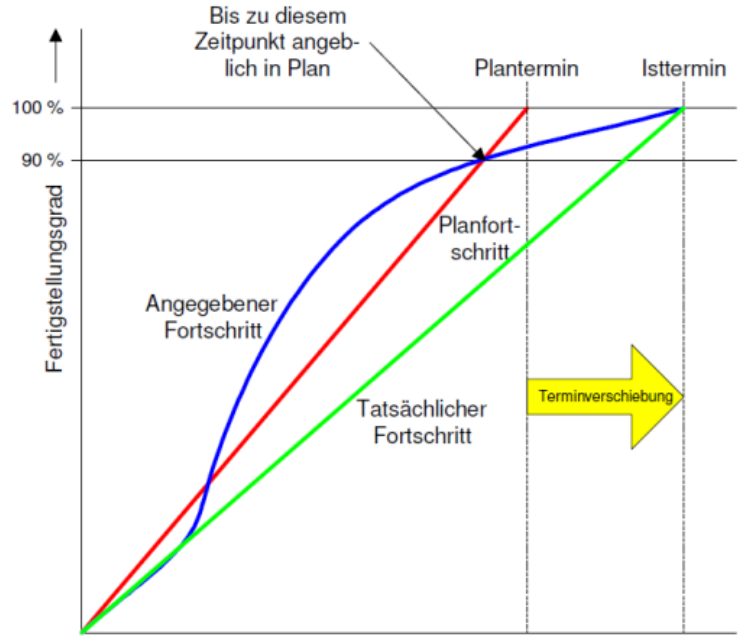
- **Abschätzung** der Schwierigkeit eines Problems und die Kosten für die Entwicklung einer Lösung **ist schwierig**
- Produktivität ist nicht proportional **zur Anzahl der Leute, die an einer Aufgabe arbeiten**
- **Hinzufügen von Leuten** in einer späten Projektphase **verlangsamt** das Projekt durch **Kommunikationsoverhead**
- Das **Unerwartete passiert immer**
- Das Herunterfahren von Testen und Reviews ist ein **Rezept für ein Desaster**
- **Nachts Arbeiten? Nur ein kurzfristiger Nutzen!**

Zeitprobleme II

- Personalmangel (Krankheit, Fluktuation, …)
- Fehlende Qualifikation
- Unvorhergesehene Schwierigkeiten
- Unrealistische Aufwandsabschätzungen
- Nicht bedachte Abhängigkeiten
- Zusätzliche Leistungsanforderungen
- Typisch bei studentischen Projekten:
  - Überraschende Prüfungszeit
  - Ungleichmäßige Arbeitsverteilung
  - Einarbeitungszeit unterschätzt

Fast-schon-fertig-Syndrom

- Letzten 10 % der Arbeit -> 40 % der Zeit
- Fortschritt messbar machen
- Nicht nur auf Schätzungen des Entwicklers verlassen



Umgehen mit Zeitproblemen

- Googledoc

Umgehen mit Zeitproblemen: Planungsphase

- Berichte eindeutig **was du weißt und was du nicht weißt und warum**!
- Berichte eindeutig **was du planst, um das Unwissen abzustellen**
- Stelle sicher, dass **alle frühen Meilensteine** erreicht werden können
- Zeitprobleme so **früh wie möglich** entdecken
- Plan to **replan**

Umgehen mit Zeitproblemen: Umsetzungsphase

- Einsatz von zusätzlichem Personal, insb. hochqualifiziertes Personal für spezielle Aufgaben
- Temporäres Erhöhen der Arbeitszeit (Überstunden, Urlaubssperre), aber nur kurzfristig möglich
- Verbesserter Tool- und Methodeneinsatz
- Optimierung der Arbeitsabläufe
- Verschiebung der Deadline
- Geringerer Leistungsumfang
  - Prioritäten vergeben, inkrementelles Ausliefern
  - Fertigstellungstermin verschieben

Kostenschätzung und Risiko

Risiken

- ***"If you don't actively attack risks, they will actively attack you."***

  – Tom Gilb

- Projektrisiken: Schedule, Ressourcen, Größe, Personal, Moral, ändernde Anforderungen, …

- Produktrisiken: Technologien (Implementierung, Sprachen), Verifikation, Wartung, …

- Businessrisiken: Markt, Verkäufe, Management, Standards, ...

Typische Risiken

Risiko	Art	Beschreibung
Personalveränderung	Projekt	Erfahrenes Personal verlässt das Projekt vorzeitig, Krankheit
Managementveränderung	Projekt	Neues Management mit anderen Prioritäten
Hardware/Software nicht verfügbar	Projekt/Produkt	Mehr Änderungen als erwartet
Verzögerung in der Spezifikation	Projekt/Produkt	Wichtige Schnittstellen nicht rechtzeitig bekannt
Unterschätzung des Umfangs	Projekt/Produkt	
Technologieveränderung	Business	Neue Technologie verdrängt benutze
Produktkonkurrenz	Business	Konkurrenzprodukt vorher auf dem Markt

Risikomanagementprozess

Risikoerkennung

- Teamarbeit, Ideensammlung, Checklisten
- Beispiele
  - Technologische Risiken: langsame Datenbank, fehlerhafte Komponente
  - Personenbezogene Risiken: Krankheit, unqualifiziertes Personal
  - Unternehmensbezogene Risiken: Managementwechsel
  - Risiken durch Werkzeuge: Code-Generator ineffizient
  - Anforderungsrisiken: Kund:innen versteht Konsequenzen von Anforderungsänderungen nicht
  - Schätzrisiken: Anzahl der Fehlerbehebungen wird unterschätzt

Risikoanalyse

- Schätzung von Wahrscheinlichkeit und Auswirkungen
- Erfahrung der Projektleitung nötig
- Grobe Skalen reichen
  - gering (<10%), niedrig (<25%), mittel (<50%), hoch (<75%), sehr hoch
  - katastrophal, ernst, tolerierbar, unbedeutsam
- Fokus auf die Top-10-Risiken

Risikoplanung

- Vermeidungsstrategien (Risiko vermeiden)
- Minimierungsstrategien (Konsequenzen minimieren)
- Notfallpläne
- **-> Erfahrung der Projektleitung nötig**
- Beispiele:
  - Kundenakzeptanz unklar: Prototyp entwickeln
  - Krankheit des Personals: Überschneidungen bei Arbeiten einplanen, Abhängigkeiten vermeiden
  - Datenbankleistung: Andere Datenbank kaufen
  - Finanzielle Probleme des Unternehmens: Zusammenfassung an Management, die Beitrag des Projekts erklärt

Typische Strategien im Risikomanagement

- Früh **Prototypen** entwickeln
- **Inkrementelle** Entwicklung
- **Gutes** Personal rekrutieren
- **Teambildende** Maßnahmen
- **Wiederverwendung**, Komponenten einkaufen

Software Teams

- Team Organisation
  - Teams sollten relativ klein sein (< 8 Mitglieder)
    - Minimiere Kommunikationsoverhead
    - Team-Qualitätsstandard kann entwickelt werden
    - Mitglieder können enger zusammenarbeiten
    - Mitglieder gehen in Team auf (haben kein “ego”)
    - Kontinuität kann selbst dann gewährleistet sein, wenn eine Person das Team verlässt
- Teile große Projekte in viele kleine Projekte auf
- Kleine Teams können informell und demokratisch organisiert sein
- *“Chief programmer teams” können das meiste aus ihren Skills und Expertisen heraus holen*

Chief Programmer Teams (Beispiel)

- Besteht aus einem Kern von Spezialist:innen, die von anderen unterstützt werden
  - Chefprogrammierer:in übernimmt **volle Verantwortung für Design, Programmierung, Testen und Installation** des Systems
  - Backup-Programmierer:in hält sich über den Stand der Arbeiten aktuell und **entwickelt Testfälle**
  - Bibliothekar:in verwaltet **sämtliche Information**
  - Andere Rollen: Projektadmin, Tool-Bauen, Doku-Schreiben, Sprach-/Systemexpert:in, Testen, und Programmieren, …
- Erfolgreich, aber mit Problemen:
  - Schwierig, talentierte Chefprogrammierer:innen zu finden
  - Kann normale Organisationsstrukturen stören
  - Kann demotivierend für Nicht-Chefprogrammierer:innen sein

Directing Teams

- **Projektmanagement unterstützen /dienen ihrem Team**
  - Management stellt sicher, dass das Team alle notwendigen Ressourcen und Informationen besitzt
  - *“The manager’s function is not to make people work, it is to make it possible for people to work”*

    – Tom DeMarco
- **Verantwortung erfordert Autorität**
  - Management muss **delegieren**: *Vertraue deinen eigenen Leuten und sie werden dir vertrauen*

Was Sie mitgenommen haben sollten:

- Nennen und erklären Sie die Aufgaben eines/r Projektmanager:in.
- Skizzieren Sie den Prozess zur Projektplanung
- Erklären Sie die Begriffe Meilenstein und Lieferschritt und nennen Sie je ein gutes und ein schlechtes Beispiel. Warum sind diese besonders bei Softwareprojekten notwendig?
- Nennen/Erklären Sie X typische Zeitprobleme und Techniken, mit diesen umzugehen.
- Nennen/Erklären Sie X typische Risiken und Techniken, mit diesen umzugehen.