Capgemini sd&m

IT-Projektmanagement
Unterstützende Prozesse

Kaiserslautern, WS 2008/2009

Dr. Gerhard Pews

ZUSAMMEN. FÜR NACHHALTIGEN ERFOLG.

Der Fahrplan durch die Vorlesung

Inhalte

- Einführung
- Das „Was“: Der Gegenstand von Softwareprojekten
- Das „Wie“: Die Tätigkeiten in einem Projekt und wie man sie ausführt
- Vorbereitung eines Projekts
- Projektplanung
- Durchführen eines Projekts
- **Unterstützende Tätigkeiten**
- Soft Factors
- Wirtschaftliche Aspekte

AGENDA

- **Meetings**
- Teamführung und Kommunikation
- Risikomanagement
- Change-Request-Verfahren
- Konfigurationsmanagement

Meetings

Lonely? Looking for company? Having trouble passing the time?
Find decision-making difficult?

Why not have a meeting!

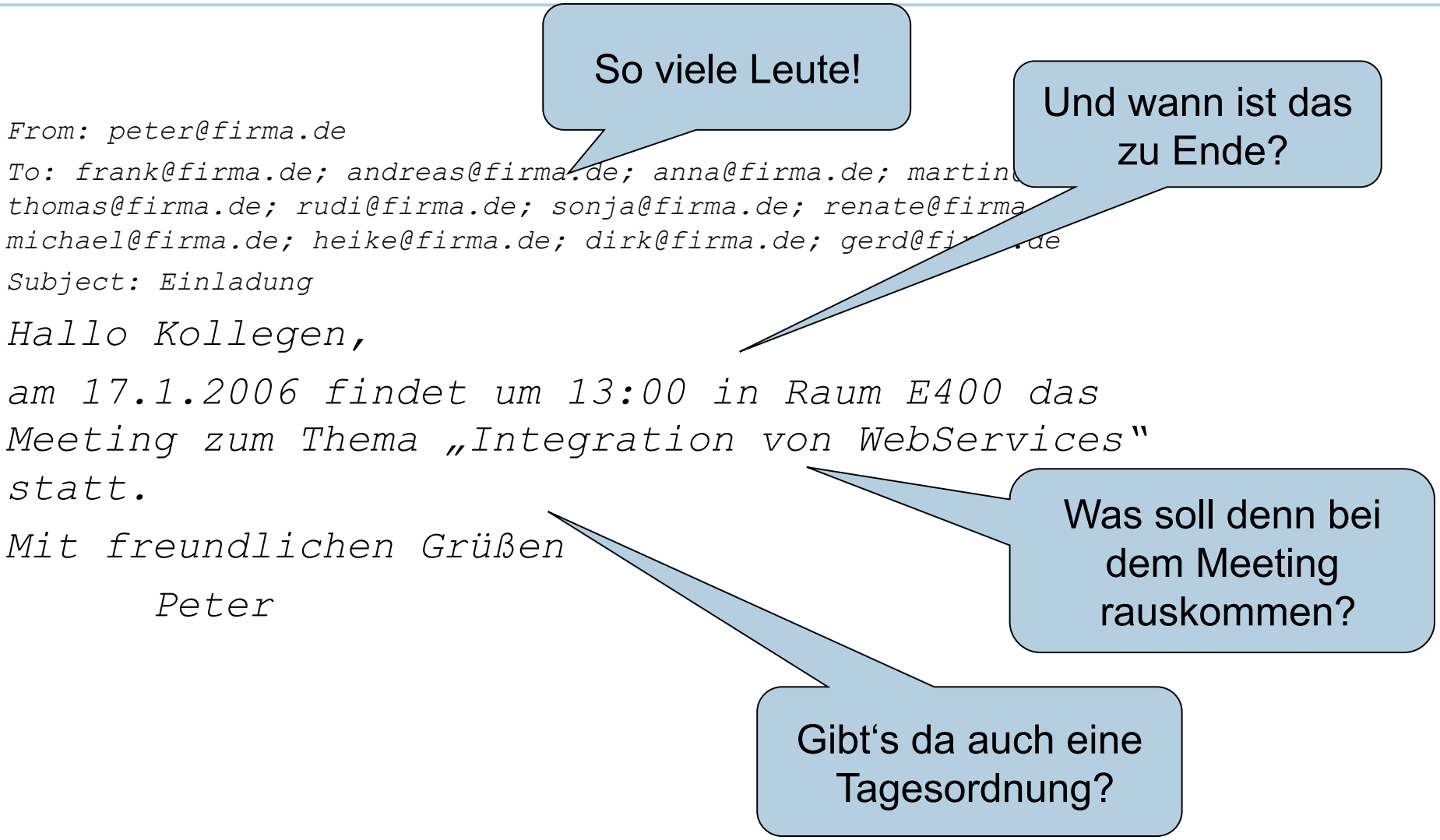
Meetings are the way to meet people, catch up on your sleep, avoid making decisions, feel important, and impress your colleagues.

And all in work time!

Meetings

The practical alternative to work.

Die Einladungsmail



Professionelle Meetings

- Ziel festlegen – und aufschreiben!
- Verantwortlichen festlegen
- Verantwortlicher verschickt Einladung
  - Termin, Ort, Zeitrahmen
  - Teilnehmer – Wer muss denn wirklich dabei sein?
  - Ziel
  - Agenda
- Wer moderiert, wer führt Protokoll?
- Technik vorher organisieren: Beamer, Flipchart, Getränke, …

Spielregeln für Meetings

- Handy aus!
- Protokoll führen, wenn die Ergebnisse dokumentiert werden müssen, weil z. B. weitere Personen unterrichtet werden müssen, weil die Ergebnisse Kosten verursachen, deren Ursache man dokumentieren will, etc.
- Die anderen ausreden lassen.
- Sich selbst kurz fassen.
- Für sich selbst reden.
- Zur Sache reden, nicht zur Person.
- Bei Telefonkonferenzen: Unbedingt ausführlich vorbereiten, straff moderieren.

Das Protokoll

- Protokoll vom 21.1.06
- Anwesend: Hr. Meier, Fr. Müller, Hr. Schulze, Fr. Schmidt
- TOP* 1: WebServices
  - Herr Müller stellt das Konzept zur Integration von WebServices vor.
  - Es muss geprüft werden, ob das auch mit MQ Series zusammen funktioniert.
  - Das XML-Schema muss angepasst werden. → Müller/Schulze
  - Hr. Schmidt klärt, ob die Services auch unter Windows genutzt werden sollen.
- TOP 2: Security
  - Herr Müller stellt das Security-Konzept vor.
- Anlage:
  - Präsentation WebServices
  - Präsentation Security

Verteiler fehlt

Schön, und wer macht das?

Wer ist denn da verantwortlich?

Und bis wann?

*TOP = Tagesordnungspunkt

Was gehört ins Protokoll?

- Datum
- Teilnehmer
- Verteiler (nicht teilgenommen, aber offiziell informiert)
- Vorgehen
  - Gleich nach dem Meeting aufschreiben, an Verteiler verschicken, ggf. Feedback einarbeiten.
- Tagesordnungspunkte
  - Information, Feststellung, Beschluss, Aufgabe
  - Aufgaben immer mit Termin und *einem* Verantwortlichen

Protokoll Lenkungskreis Projekt XY

- Protokoll vom 21.1.06
- Anwesend: Hr. Meier, Fr. Müller, Hr. Schulze, Fr. Schmidt.
- Verteiler: Teilnehmer, Fr. Wagner
- TOP 1: Teilprojekt Dialoge
  - Information: Herr Müller stellt den Status des Teilprojekts vor.
  - Festellung: Die Teilnehmer halten das Erreichen des Meilensteins 6 („Auslieferung Basis“) für unrealistisch
  - Beschluss: Die Dialoge zum erweiterten Suche werden aus dem Lieferumfang von Meilenstein 6 genommen und Meilenstein 8 zugeschlagen.
  - Aufgabe: Hr. Müller stellt die Auswirkungen dieser Änderung und die aktualisierte Planung vor → Müller, bis 28.1.06
- Anlage:
  - Präsentation Teilprojekt Dialoge

Tipps beim Protokollieren

- Derjenige, der protokolliert, muss das Thema des Meetings kennen und verstehen
  - In IT-Fragen kann das Sekretariat oder Projektoffice in der Regel nicht unterstützen
- Protokoll und Moderation sind schwierig gleichzeitig zu machen. Bei schwieriger Moderation sollte jemand anderes Protokoll führen.
- Wenn man selbst protokolliert
  - Nachhaken, wenn Punkte nicht klar sind: „Wie soll ich das jetzt festhalten?“
  - Als Verantwortlicher des Meetings dafür sorgen, dass Aufgaben klar und mit Termin verteilt werden. Ideal ist ein gutes Zusammenspiel mit dem Protokollführer.
- Protokoll gleich nach dem Meeting fertig schreiben – nicht erst Tage warten, weil es lästig ist.

AGENDA

- Meetings
- **Teamführung und Kommunikation**
- Risikomanagement
- Change-Request-Verfahren
- Konfigurationsmanagement

Teamführung: Kommunikation

- Kommunikation ist eine der wichtigsten Aufgaben des Projektleiters.
- Für Kommunikation muss man aktiv als Projektleiter sorgen, Kommunikation geschieht nicht automatisch von allein.
- Jeder Mitarbeiter sollte wissen, woran seine Kollegen arbeiten, z. B.
  - durch eine kurze Statusrunde im Teammeeting
  - Themen und Verantwortliche im Arbeitsraum als Poster aufhängen
- Als PL nicht davon ausgehen, dass das Team den gleichen Kenntnisstand hat wie man selber
  - Das eigene Bild vom Projekt und von Projektinhalten ändert sich schleichend und unbemerkt
  - Ruhig auch mal die „selbstverständlichen“ Dinge kommunizieren

Die „selbstverständlichen“ Dinge…

…geschehen seltsamerweise nicht von selbst

- Beispiele
  - Wenn ein Missstand entdeckt wird, muss *man* diesen beseitigen.
  - Wenn eine Maßnahme beschlossen wird, muss *man* diese auch durchführen.
- Häufige Ursache:
  - Im Team fühlt sich niemand verantwortlich
  - Der Projektleiter ist *immer* verantwortlich, er muss Mitarbeiter beauftragen und nachhaken

Praxisbeispiel: zwei Teams

- Führungsstil Team A
  - Informationen sind Herrschaftswissen.
  - Planung erfolgt heimlich und im „stillen Kämmerlein“
  - Jeder Mitarbeiter wird nur über die Dinge informiert, die er zur Erledigung seiner Aufgaben benötigt.
  - Projektleitung zieht alle Aufgaben an sich.

- Führungsstil Team B
  - Stetiger Informationsfluss ins Team (allerdings nicht über noch unsichere Dinge, die das Team verunsichern können)
  - Informationen aus Steuerungsgremien sind weitgehend transparent.
  - Projektleitung kommuniziert, vermittelt Bild für das Ganze
  - Projektleitung gibt Aufgaben ab.

DeMarco, Der Termin

- Vier Grundsätze guten Managements:
  - Wählen Sie die richtigen Leute aus.
  - Betrauen Sie die richtigen Mitarbeiter mit den richtigen Aufgaben
  - Motivieren Sie die Mitarbeiter
  - Helfen Sie den Teams, durchzustarten und abzuheben.

  - Alles andere sind Administrivialitäten

AGENDA

- Meetings
- Teamführung und Kommunikation
- **Risikomanagement**
- Change-Request-Verfahren
- Konfigurationsmanagement

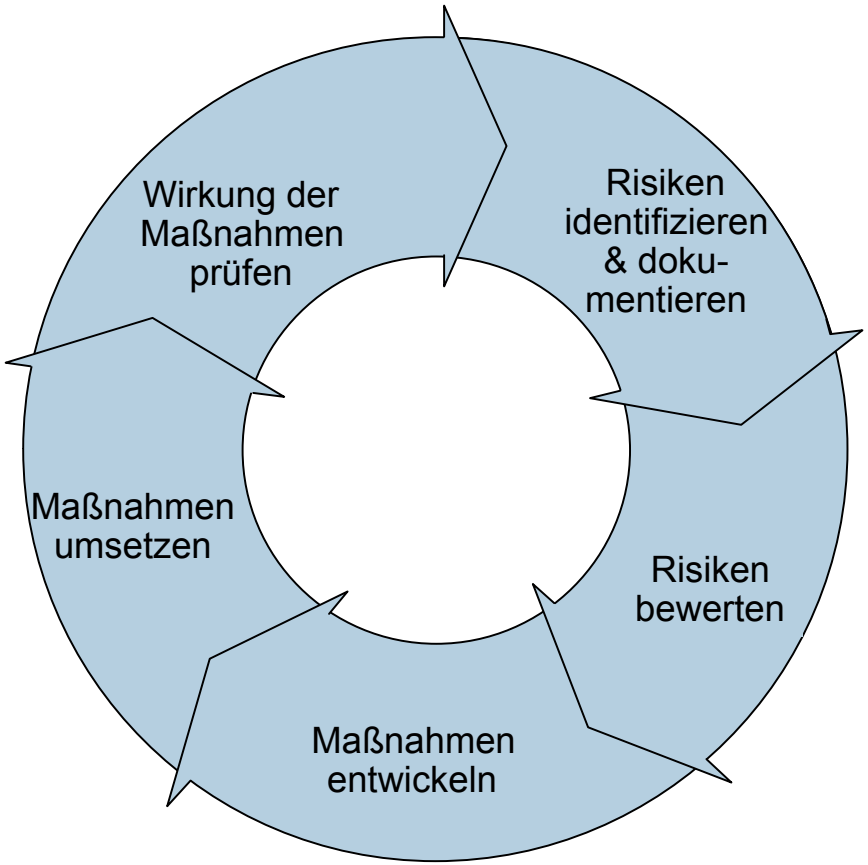
Motivation: Herr Müller und die Sommerreifen

Das hab ich kommen sehen!

- 27. November 2008: Herr Müller verlässt das Haus und steigt in seinen Wagen. Es hat über Nacht geschneit, doch der Wagen hat noch Sommerreifen. Herr Müller hat ein Problem: Der Weg führt über einen Hügel, aber mit Sommerreifen hat er keine Chance, über den Hügel zu kommen. Und jetzt sind die Werkstätten natürlich vollkommen überlaufen. Die Reifen können erst in zwei Wochen gewechselt werden.

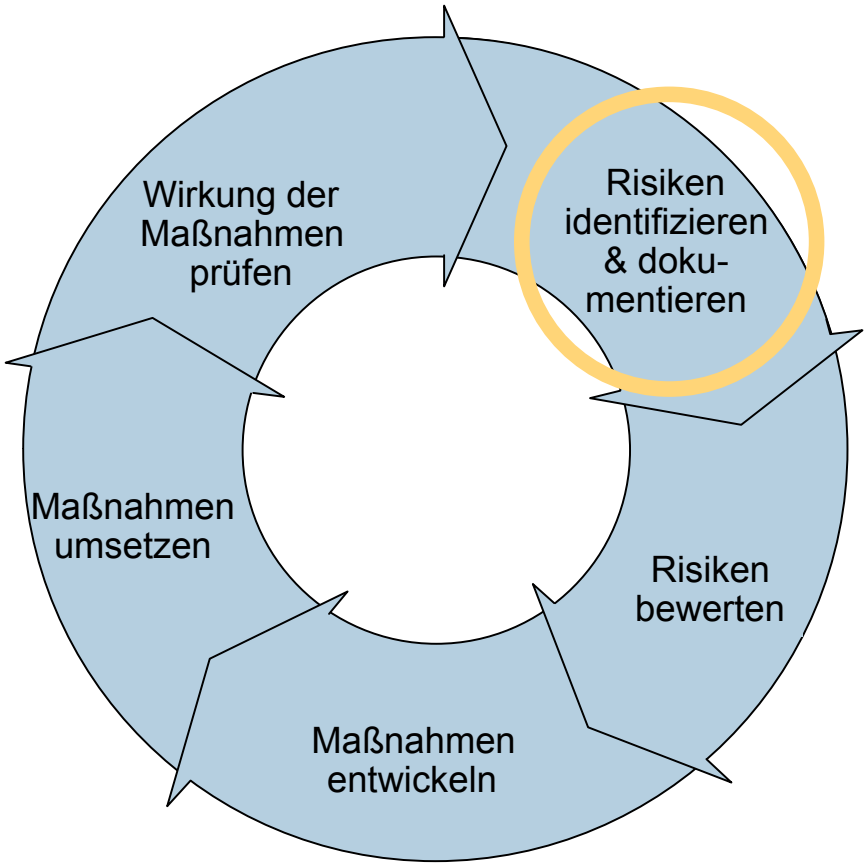
- Viele Probleme sind absehbar, wenn man sich nur früh genug Gedanken darüber macht.
- Ein Risiko ist ein potentielles Problem, das noch nicht eingetreten ist, das aber eintreten könnte.
- Risiken sind in der Regel einfacher zu bekämpfen als die Probleme!

Risikomanagement im Projekt



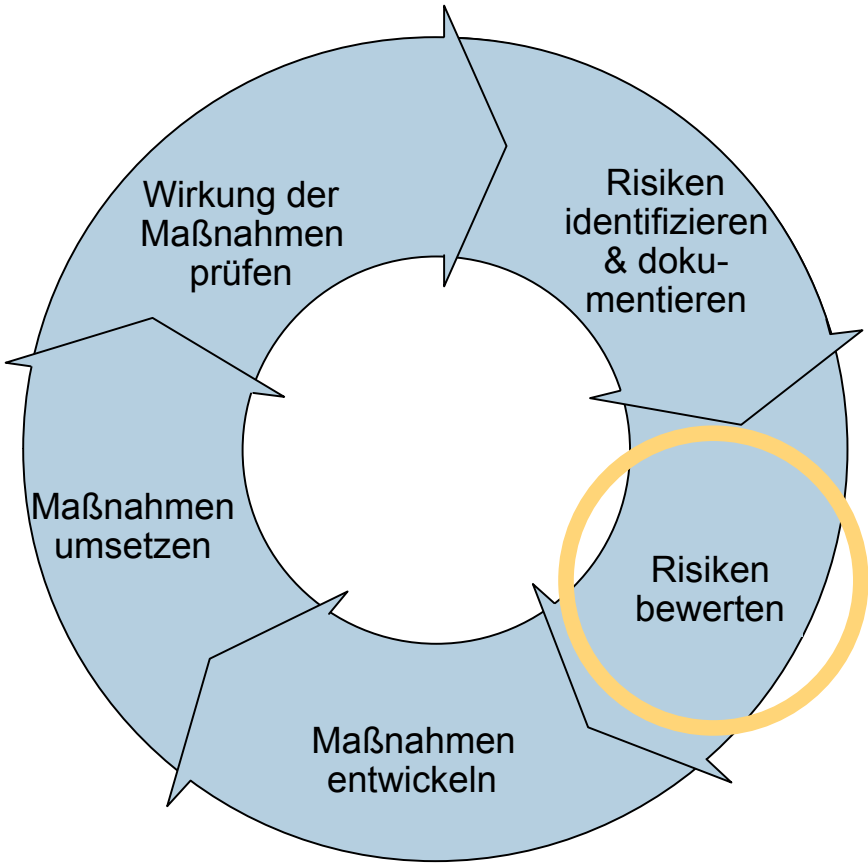
- Risikomanagement ist ein ständiger Prozess.
- Risikomanagement geht alle im Projekt an.
- Verantwortlich für Risikomanagement: Projektleiter, delegiert in der Regel an Qualitätssicherer oder Teammitglied.
- Risiken ändern sich ständig und müssen immer wieder neu bewertet werden.

Identifizieren von Risiken



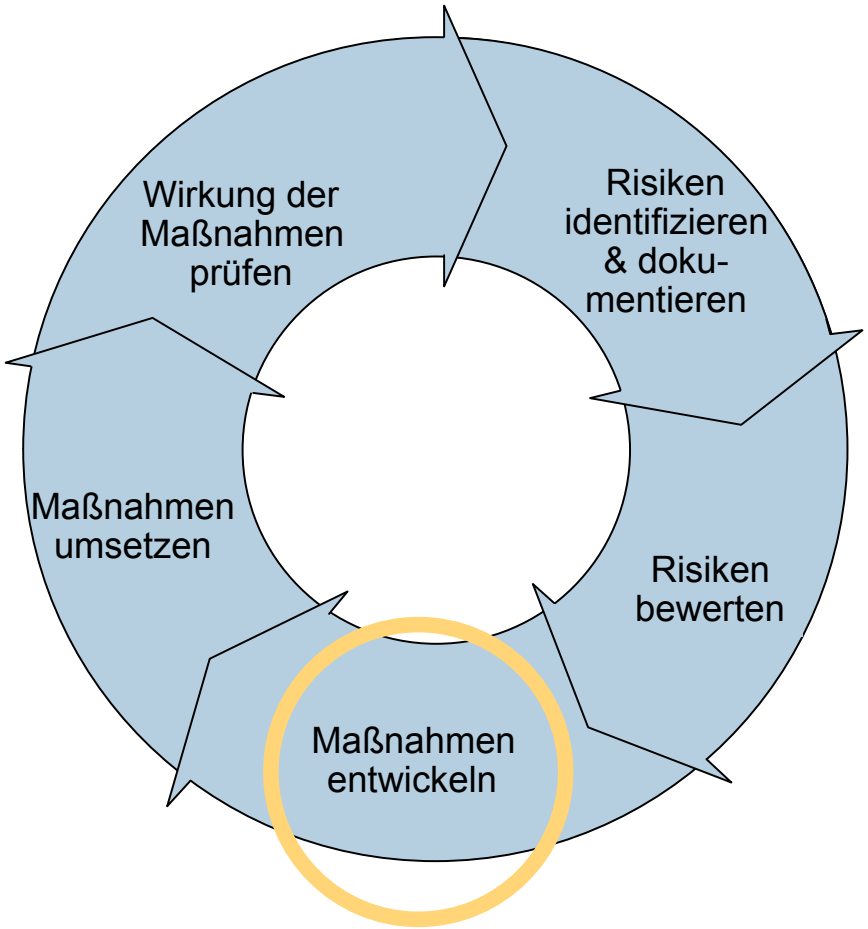
- Sammeln von Risiken
  - spezielle Risikorunde (PL, QS, CD, einzelne Mitarbeiter)
  - im regelmäßigen Team-Meeting: z. B. jedes 2. Mal werden die Risiken besprochen und neue Risiken und Maßnahmen gesammelt
- Bereiche für Risiken (als Ideengeber):
  - Technik
  - Fachlichkeit
  - Team
  - Vorgehen im Projekt, Planung
  - Auftraggeber

Bewertung von Risiken



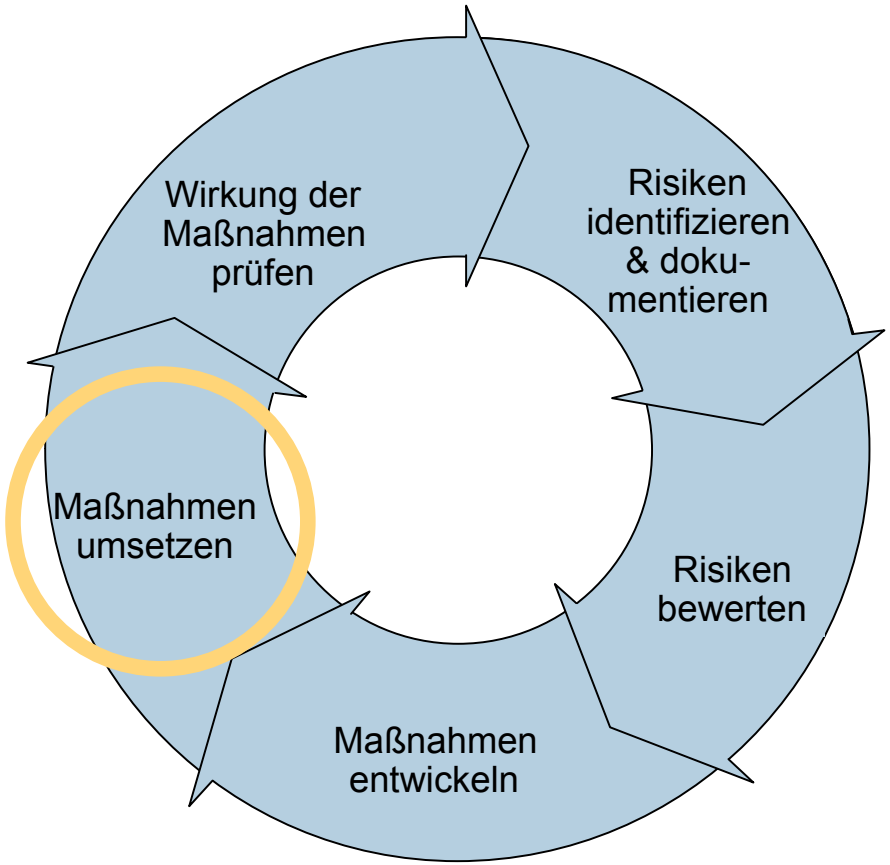
- Bewertung:
  - Wie schlimm sind die Auswirkungen, wenn das Risiko eintritt?
  - Wie hoch ist die Wahrscheinlichkeit, dass das Risiko tatsächlich eintritt?
- Daraus Gesamtbewertung: Wie groß ist das Risiko?

Maßnahmen entwickeln



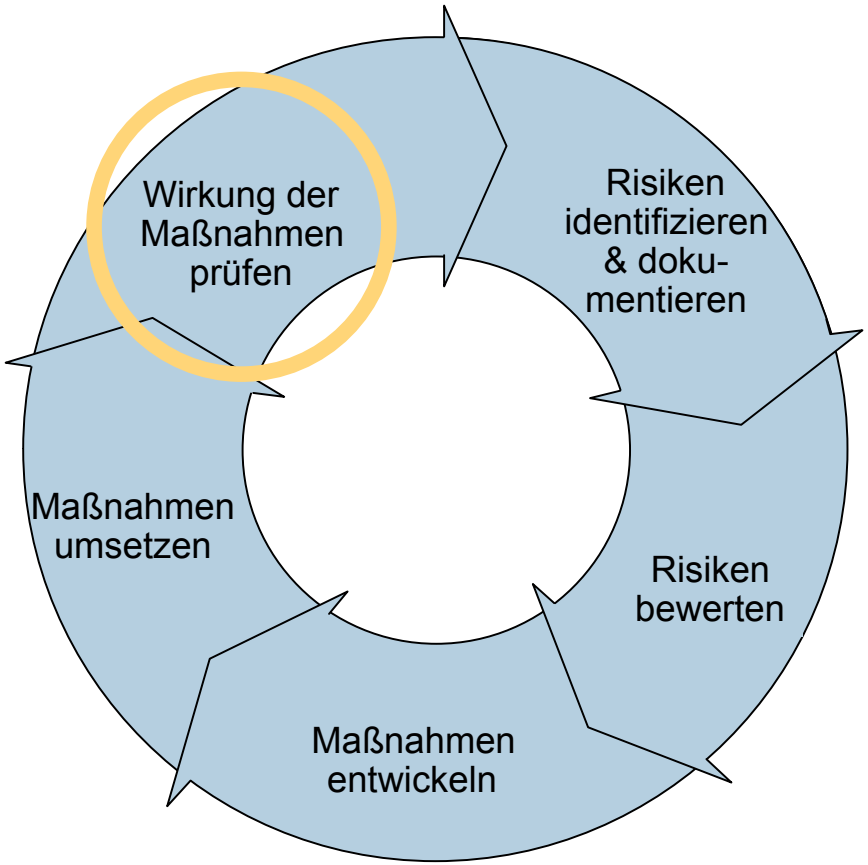
- Maßnahmen zielgerichtet entwickeln: für Risiken mit hohem Risiko sollte es Maßnahmen geben.
- Gegebenenfalls kann man sich auch explizit entschließen ein Risiko einzugehen oder sich schon Maßnahmen einfallen lassen, um die Auswirkungen bei Eintritt zu mildern.

Maßnahmen umsetzen



- Maßnahmen sind Aufgaben:
  - ein Verantwortlicher
  - ein Termin, bis zu dem die Maßnahme umzusetzen ist

Wirkung prüfen



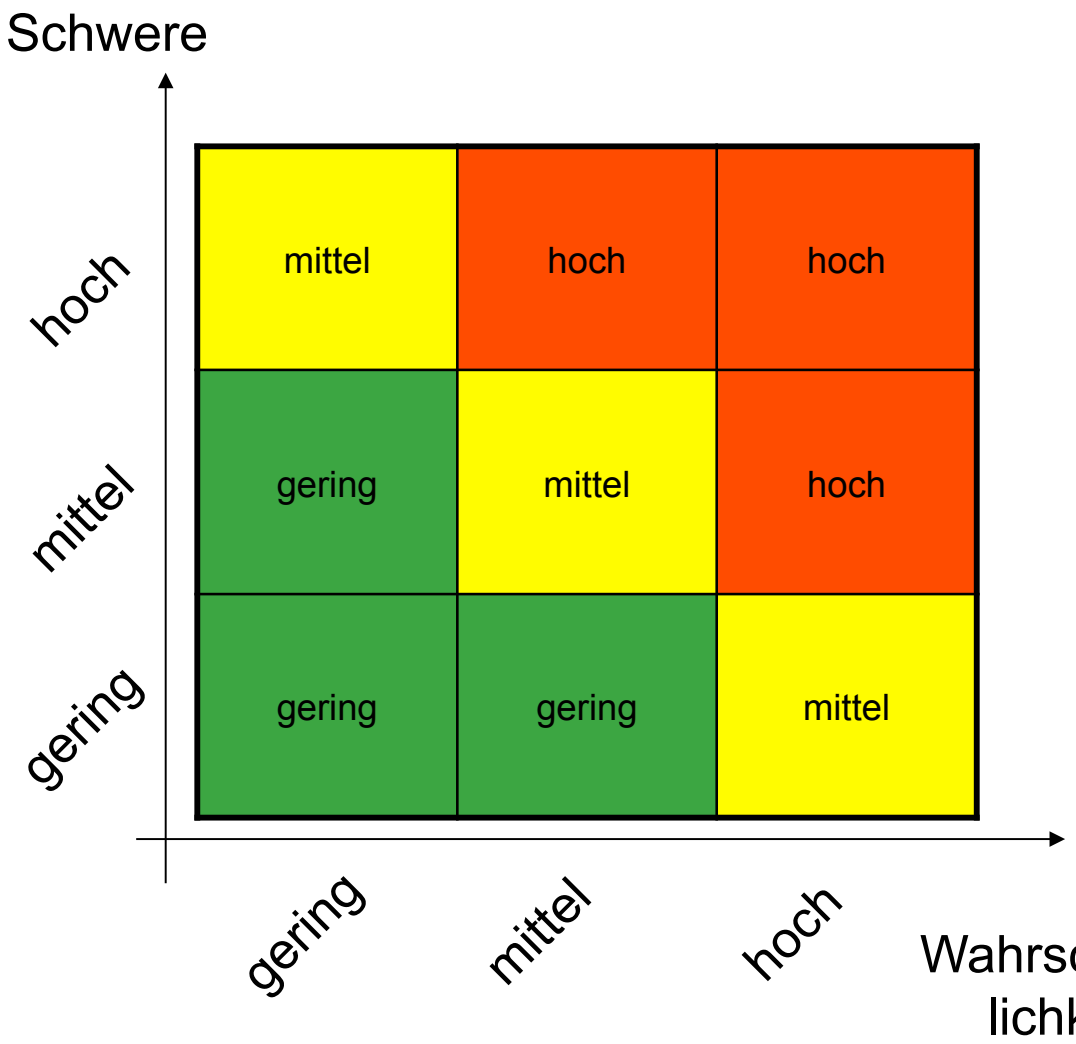
- Maßnahmen müssen wie jede andere Aufgabe auch nachgeprüft werden: wurde die Aufgabe wirklich erledigt?
- Maßnahmen können aber auch zu geringe Wirkung haben: neue Maßnahmen oder konsequentere Umsetzung.

Dokumentation von Risiken: die Risikoliste

Eine Risikoliste umfasst:

- Beschreibung des Risikos
  - Name, Bezeichnung
  - Was kann passieren?
  - Was sind die Konsequenzen?
- Bewertung
  - Wie groß ist der Schaden?
  - Wie groß ist die Eintrittswahrscheinlichkeit?
  - Gesamtbewertung
- Maßnahmen
  - Maßnahme, Aufgabe
  - Verantwortlicher
  - Termin
  - Status

Gesamtbewertung von Risiken

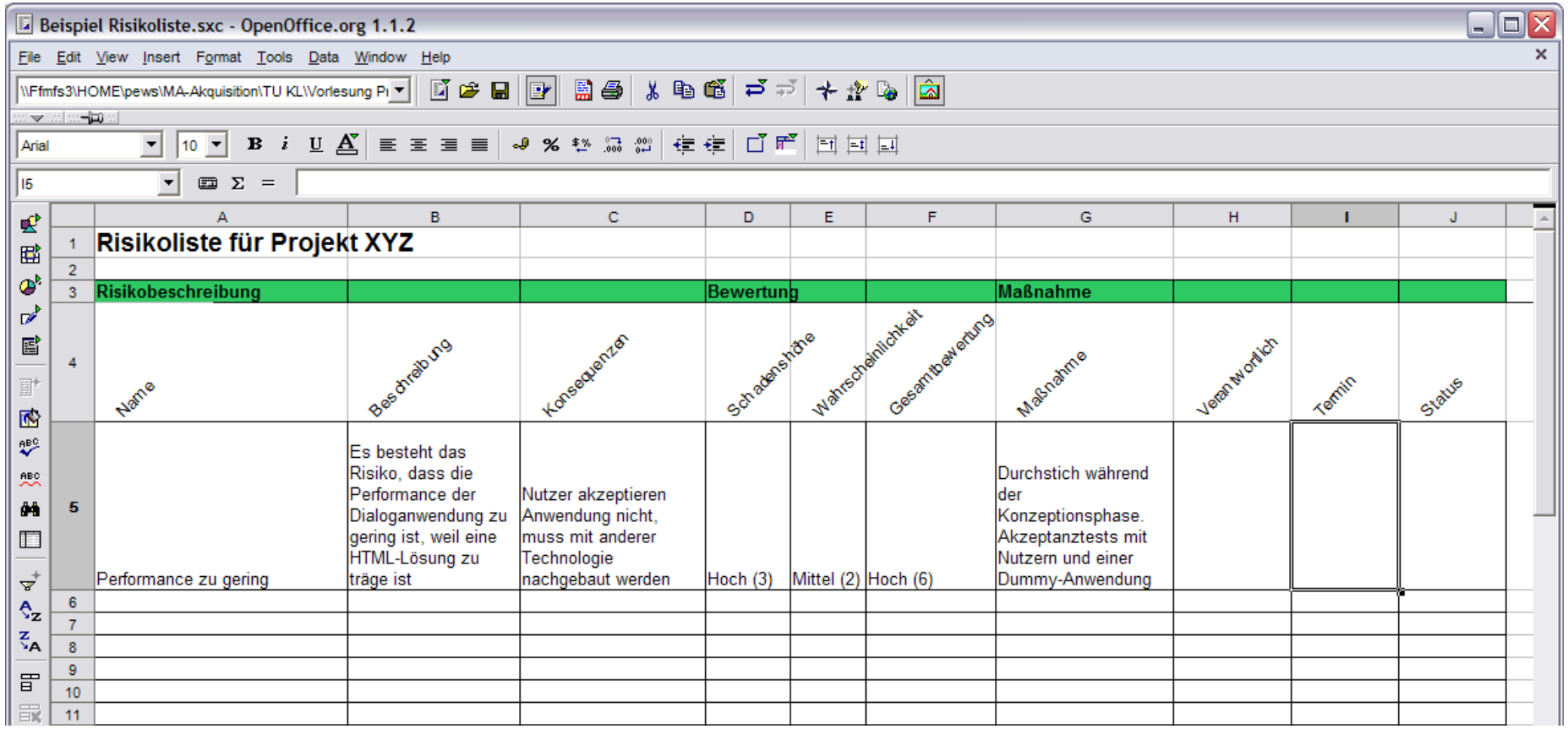


- Üblich: 3-stufige Bewertung gering, mittel, hoch.
- Verrechnung eigentlich nach Formel:

Wahrscheinlichkeit*Schwere

- Üblich: tabellarische Bewertung mit „Handjustierung“, d. h. Risiken können auch eine andere Kategorie haben als die der Tabelle

Beispiel Risikoliste



	A	B	C	D	E	F	G	H	I	J
1	**Risikoliste für Projekt XYZ**									
2										
3	**Risikobeschreibung**			**Bewertung**			**Maßnahme**			
4	Name	Beschreibung	Konsequenzen	Schadenshöhe	Wahrscheinlichkeit	Gesamtbewertung	Maßnahme	Verantwortlich	Termin	Status
5	Performance zu gering	Es besteht das Risiko, dass die Performance der Dialoganwendung zu gering ist, weil eine HTML-Lösung zu träge ist	Nutzer akzeptieren Anwendung nicht, muss mit anderer Technologie nachgebaut werden	Hoch (3)	Mittel (2)	Hoch (6)	Durchstich während der Konzeptionsphase. Akzeptanztests mit Nutzern und einer Dummy-Anwendung			
6										
7										
8										
9										
10										
11										

Weitere Beispiele für Risiken

- Technik
  - Risiko, dass Performance nicht stimmt
- Fachlichkeit
  - Risiko, dass Anwendung nicht nutzbar, weil bestimmte Daten im System fehlen und deshalb Berechnungen nicht durchgeführt werden.
- Team
  - Risiko, dass neue Mitarbeiter sich nicht schnell genug einlernen können
- Vorgehen
  - Risiko, dass Auftraggeber andere Zwischenergebnisse erwartet und deshalb die Abnahme der Zwischenergebnisse ablehnt
- Auftraggeber
  - Risiko, dass eine Mitwirkungsleistung nicht erbracht wird

Tipps zum Risikomanagement

- Risikomanagement darf kein leerer Formalismus sein – es muss gelebt werden.
- Risiken ernst nehmen, bei Erhebung nicht abblocken. Nicht sagen: „Das schaffen wir schon“
- Risiken gehen alle an: Mitarbeiter sollten alle die Top-5-Risiken kennen
- Risiken konkret formulieren – nicht abstrakt

AGENDA

- Meetings
- Teamführung und Kommunikation
- Risikomanagement
- **Change-Request-Verfahren**
- Konfigurationsmanagement

Ein Szenario…

- Anwender Schulze:
  - „In dem Fenster stört mich, dass ich nur zwischen den Anreden ‚Herr' und ‚Frau' auswählen kann. Eine zusätzliche Anrede ‚Familie' wäre praktisch."
  - „Ich kenne den Entwickler Meier, den rufe ich an und der macht das für mich."
- Meier:
  - „Klar, das mach ich doch so nebenbei. Im nächsten Release ist das drin."

Alles klar?

…und die Fortsetzung

- Wird das auch getestet? Irgendwo sonst im Code stand nämlich:

```
if (anrede == „herr“) then … else …
```

- Passt das zu allen Schnittstellen?
- Wie lange dauert das Programmieren? Was bleibt deshalb liegen?
- Wer bezahlt das denn? – Insbesondere bei Projekten zum Festpreis
- Ist das so wichtig wie der Änderungswunsch von Frau Schmidt?
- Ist das wirklich fachlich richtig? - Vielleicht war in der Anrede bisher das Geschlecht der Person codiert.
- Wer zieht das in der Nutzerdokumentation nach?

→ Änderungen in größeren Projekten erzeugen Unruhe, gefährden Termine, sorgen für sinkende Qualität

Der Konflikt

Stabilität im Projekt

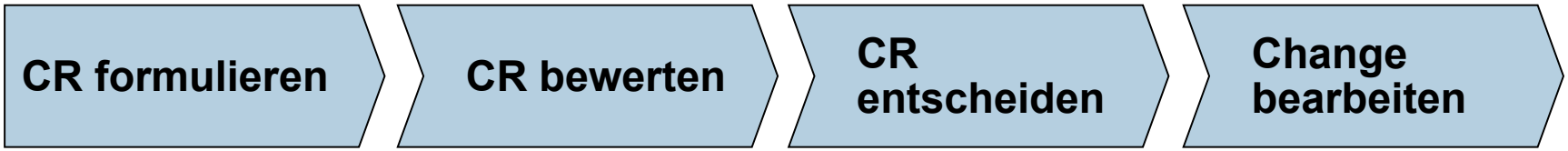
- *vergleiche Vorgehen „Wasserfall“*
- *Änderungen sind ein Schritt zurück*
- *Änderungen müssen umfangreich abgestimmt werden.*
- *Änderungen erzeugen Kosten, gefährden Termine*

Änderungen in der Welt

- *Seit Auftragserteilung ist die Welt nicht stehen geblieben*
- *Von außen ändern sich*
  - *Prozesse*
  - *Anforderungen*
  - *Gesetze*
  - *Prioritäten*
  - *beteiligte Personen*
- *Im Projekt lernt man dazu*

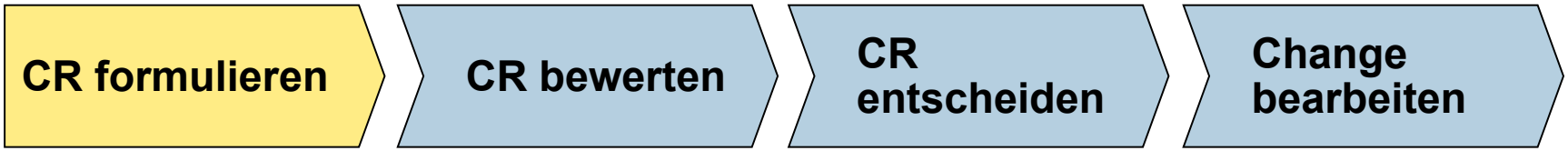
Man benötigt einen Filter, damit Änderungen nicht unkontrolliert im Projekt einschlägt und damit notwendige Änderungen kontrolliert und vollständig umgesetzt werden können

Change Request-Verfahren



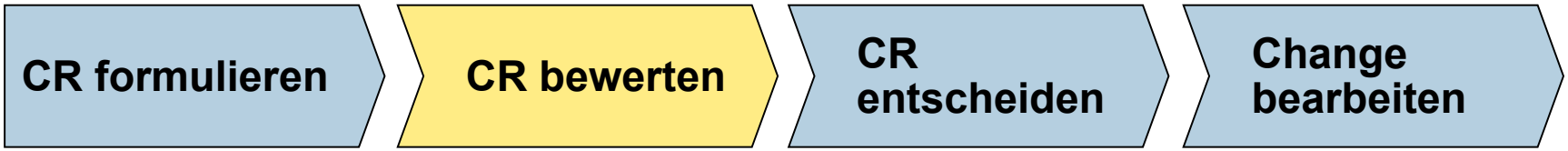
- Das Change Request-Verfahren sorgt dafür, dass Änderungen kontrolliert in das Projekt einfließen

Change Request formulieren



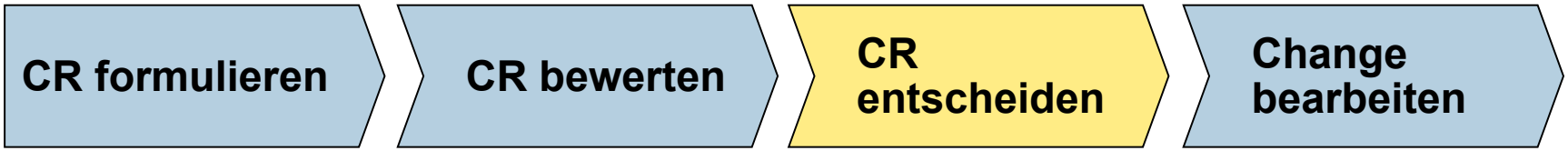
- Was genau ist das Problem?
- Was genau ist die Lösung?
- Wer will das, wer bezahlt das?
- Wer ist davon betroffen?
- Was ist die Konsequenz, wenn der CR nicht umgesetzt wird?
- Beim Hinschreiben merkt man oft schon, dass mit CR etwas nicht stimmt

Change Request bewerten



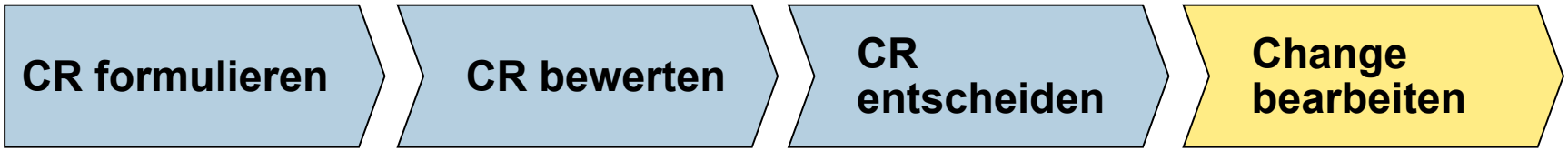
- Konsequenzen aus Umsetzung ermitteln:
  - Zeit
  - Budget
  - Fachliche, technische Auswirkungen
- Priorisierung des CR vornehmen: Wie wichtig ist er im Vergleich zu anderen CRs?
- Gemeinsam durch PL und Anforderer

Change Request entscheiden



- Möglichkeiten:
  - zulassen
  - ablehnen
  - zurückstellen
- Entscheidung durch
  - Projektleiter (kleinere CRs)
  - Lenkungskreis (größere CRs)

Change bearbeiten



- In Projektplanung einarbeiten
  - Planung ändern

Was ist ein CR?

- Jede Abweichung vom ursprünglich vereinbarten Leistungsumfang, vom Auftrag
  - Es gibt positive und negative CRs: Umfang kann steigen oder sinken
    → Regelfall: Umfang steigt
  - CRs können vom Auftraggeber und vom Auftragnehmer eingebracht werden
    → Regelfall: Auftraggeber
  - CRs können nicht nur die Software, sondern z. B. auch die Projektorganisation oder das Projektvorgehen betreffen
  - CRs können sich auf bereits erstellte oder auf noch zu erstellende Ergebnisse beziehen

Hinweise zum CR-Verfahren

- Bearbeiten von CR-Anträgen dauert und kostet Geld, egal, ob diese umgesetzt werden oder nicht
- Bei Projektbeginn schon die Spielregeln für CR-Verfahren bekannt machen, z. B. im Angebot bzw. im Projektauftrag beschreiben
- CRs technisch managen:
  - Tabellenkalkulation
  - Spezialsoftware, oft in Verbindung mit Konfig-Management und Anforderungen
  - Freeware: BugZilla
- Zurückgestellte CRs sind oft Basis für eine Folgestufe
- Überspitzt formuliert ist ein CR-Verfahren ein Change-Verhinderungs-Verfahren: Ein CR, der so ein Verfahren erfolgreich passiert, muss wirklich wichtig sein.
- Changes sind keine Fehler
  - oft schwierig auseinander zu halten, werden ähnlich gemanaged und eingeplant

Missbrauch des CR-Verfahrens

- Anbieter gibt auf Ausschreibung ein nicht kostendeckendes, niedriges Angebot ab → Anbieter bekommt Zuschlag.
- Projekt startet → Auftraggeber ist an Anbieter gebunden
  - Den Anbieter zu wechseln verzögert, das Projekt kostet mehr Geld
- Im Verlauf eines Projekts ergeben sich zwangsläufig CRs des Auftraggebers
  - Anbieter finanziert das Projekt über CRs
  - Anbieter optimiert seinen Gewinn über CRs

AGENDA

- Meetings
- Teamführung und Kommunikation
- Risikomanagement
- Change-Request-Verfahren
- **Konfigurationsmanagement**

Szenarien in der Software-Entwicklung

- Szenario 1: „Der Fehler war doch schon mal draußen!“
  - Wo kommt der Fehler denn jetzt wieder her, den habe ich doch schon vor Wochen beseitigt

→ Ursachen: Code von anderen übernommen, mit der falschen Datei weitergearbeitet, etc.

- Szenario 2: „Warum läuft das denn jetzt nicht mehr?“
  - Mehrere Personen entwickeln gemeinsam. Auf jedem einzelnen Entwicklungsrechner ist der Code lauffähig, bei der Integration der beiden Codeteile nicht mehr.
- Szenario 3: Wir müssen einen Fehler beheben. Und zwar in der Programmversion, die wir vor 6 Wochen ausgeliefert haben
  - Was war denn da genau für ein Code drin? Die neue Version ist noch nicht fertig und kann nicht genutzt werden.

Szenarien in der Software-Entwicklung

- Szenario 4: Was habe ich denn eigentlich alles geändert?
  - Der Code auf dem Entwicklerrechner ist lauffähig, aber es ist nicht klar, welche Dateien jetzt wirklich geändert wurden und welche nicht.
- Szenario 5: In der Online-Hilfe ist das neue Fenster ja gar nicht beschrieben
  - Der Text ist aber irgendwo erstellt worden. Aber leider passen auch die GIFs nicht mehr zum Hilfetext in HTML.

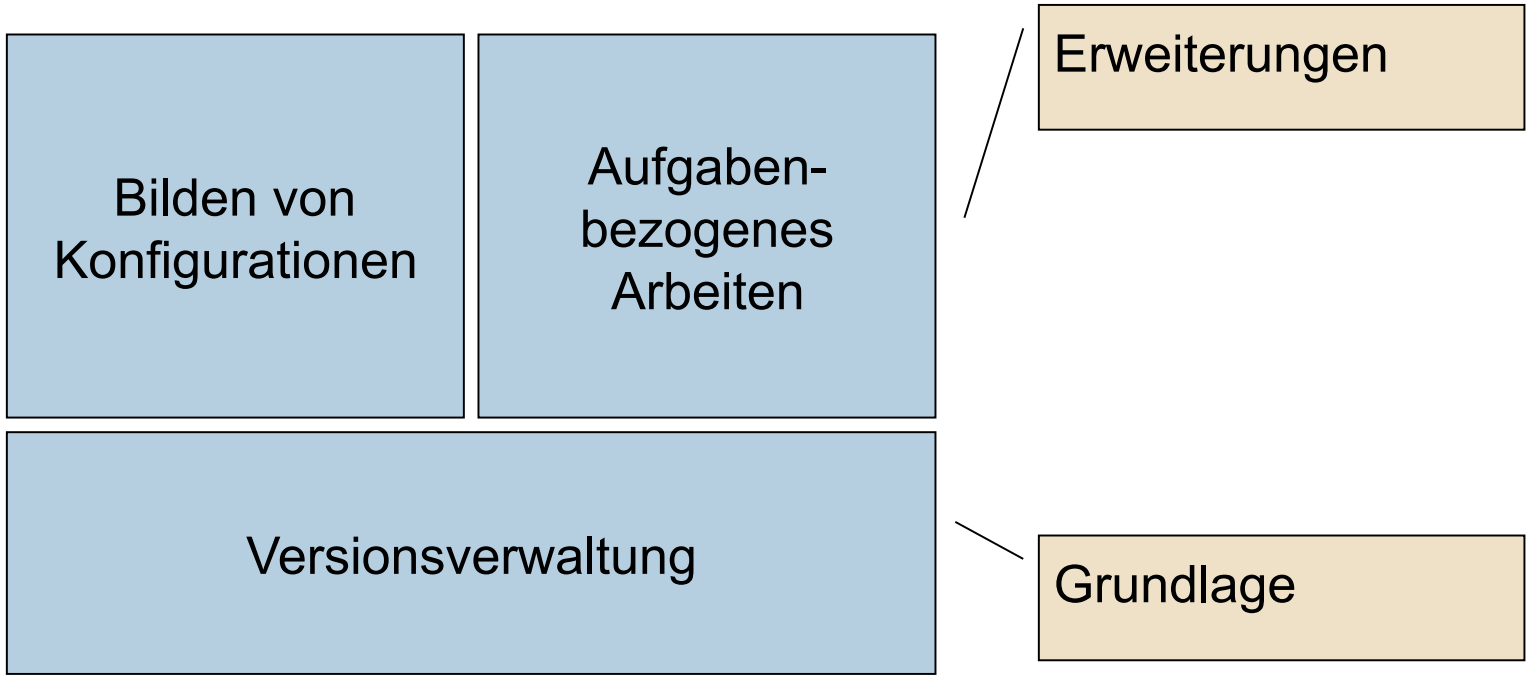
➔ ***Professionelle Software-Entwicklung benötigt professionelles Management der erstellten Ergebnisse/des Codes –*** *Konfigurationsmanagement*

Einfachster Fall: Konfigurationsmanagement durch gemeinsame Ablage

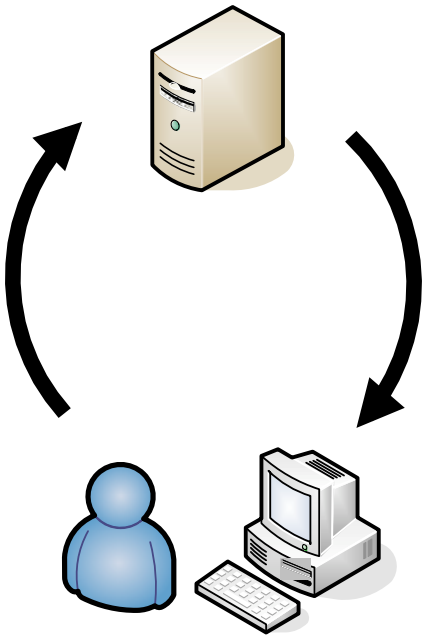
- Idee: alle Mitarbeiter arbeiten auf einer gemeinsamen Ablage, z. B. einem zentralen Repository oder einem gemeinsam genutzten Dateisystem.
- Alle Änderungen sind damit für alle sofort sichtbar.
- Dateien sind gesperrt, so lange sie bearbeitet werden.

- Häufig bei gemeinsam genutzten Modellierungstools (z. B. UML-Werkzeugen) anzutreffen.
- Funktioniert nur bei kleinen Teams und wenn die Mitarbeiter meistens auf disjunkten Dateibeständen arbeiten.
- Löst die meisten Probleme in den Eingangsszenarien nicht

Leistungsfähigere Ansätze

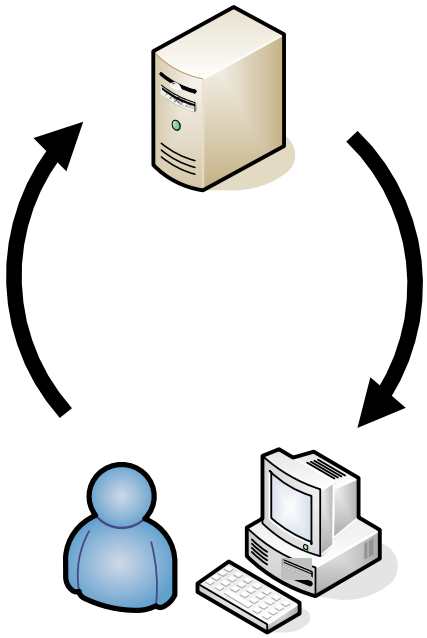


Versionsverwaltung



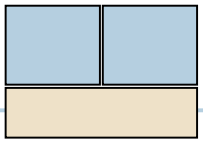
- Idee: zentrales Repository (Datenbank)
- Verwaltet alle Dateien in allen jemals erstellten Versionen
- Entwickler kopiert Dateien auf seinen Rechner (zum Lesen)
- Falls er eine Datei ändern will: check-out. Eine neue Version wird erstellt.
- Falls er die geänderte Datei in das Repository einstellen will: check-in. Neue Version ist für alle anderen Entwickler verfügbar. Man kann aber auch auf alte Versionen zurückgreifen.

Versionsverwaltung - Konflikte



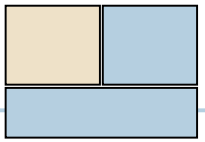
- Ein zweiter Nutzer will die Datei ändern:
  - System kann dies verhindern (Normalfall)
- oder
  - System warnt, dass Datei zwischenzeitlich verändert wurde
  - Nutzer führt eigene Änderungen mit fremden Änderungen zusammen

Versionsverwaltung



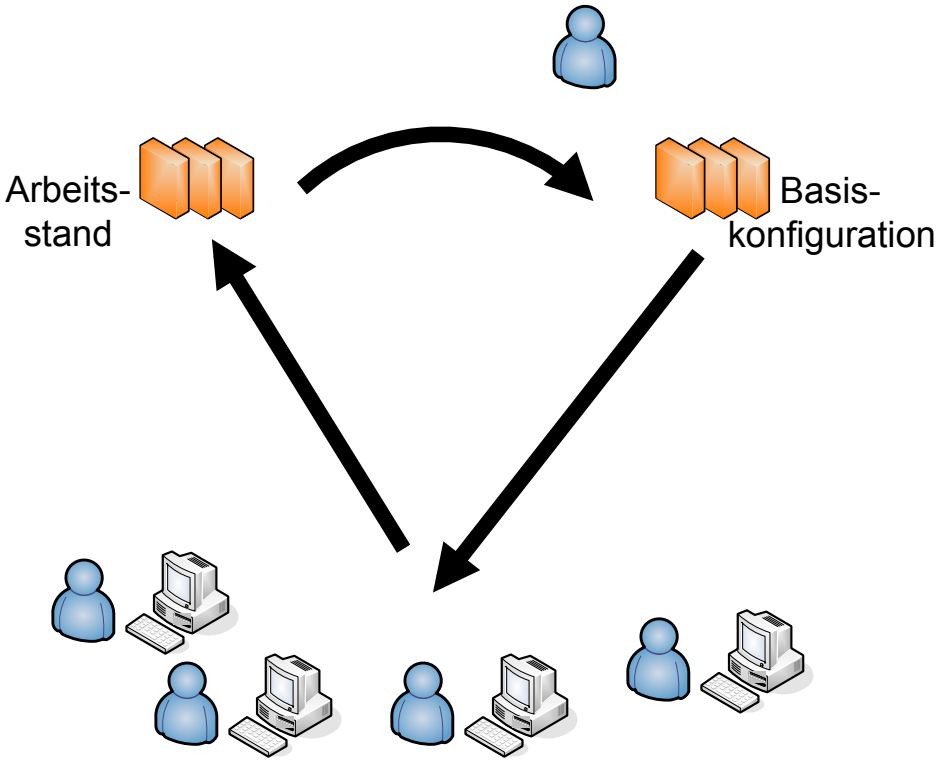
- Der aktuelle Stand der Gesamtsoftware ist derjenige, der aktuell im Repository eingecheckt ist.
- Typisches Vorgehen: nightly builds
  - Alle Mitarbeiter müssen bis zum Abend ihren Code wieder in das System eingecheckt haben, damit kein Code gesperrt ist.
  - Nachts wird der gesamte Code compiliert.
  - Wessen Code einen Compile-Fehler erzeugt, gibt eine Runde aus ☺
  - Der nächtlich erzeugte Stand ist die Grundlage für die Arbeit am nächsten Tag
- Probleme:
  - Änderungen, die mehrere Tage in Anspruch nehmen, werden nicht unterstützt
  - Wie kommt man an den Stand vom letzten Februar?

Erweiterung: Bilden von Konfigurationen



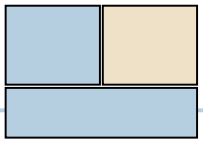
- Eine Konfiguration/Release legt zu jeder Datei fest
  - Ist die Datei Teil der Konfiguration?
  - In welcher Version ist sie Teil der Konfiguration?
- Damit lassen sich beliebige Softwarestände konservieren und wieder herstellen. Das Laden einer Konfiguration ist möglich.
- Einsatzszenarien:
  - Als Ergänzung zum bisher geschilderten Vorgehen, nur mit mächtigerer Historie
  - Statt nightly builds: ein Mitarbeiter übernimmt die Rolle des Konfigurationsmanagers

Erweiterung: Bilden von Konfigurationen



- Eine Basiskonfiguration ist der Ausgangspunkt der Entwicklung.
- Die Entwickler stellen Code in das Repository ein, dort bildet dieser den aktuellen Arbeitsstand.
- Sobald dieser Arbeitsstand die nötige Reife erreicht, kann der Konfigurationsmanager eine neues Basiskonfiguration bilden.
- Es kann auch für unterschiedliche Teilteams der Entwickler unterschiedliche Arbeitsstände geben.

Erweiterung: Aufgabenbezogenes Arbeiten



- Problem: um eine Änderung durchzuführen, müssen mehrere Dateien verändert werden. Diese Dateien sollen gemeinsam gemanaged werden.
- Aufgabenbezogenes Arbeiten
  - In das Konfigurationsmanagement wird der Begriff der Aufgabe „Task“ eingeführt. Ein Beispiel für eine Task ist: „Bearbeitung CR 234“ oder „Behebung Fehler XYZ“
  - Der Entwickler meldet im Konfigurationsmanagement an, dass er eine bestimmte Task bearbeiten will. Alle jetzt bearbeiteten Dateien werden dieser Task zugeordnet.
  - Diese Task lässt sich dann als eine Einheit verwalten, z. B. laden oder alle bereits gemachten Änderungen verwerfen.

Festlegungen für das Konfigurationsmanagement

- Was soll alles verwaltet werden?
  - Code
  - Dokumentation?
  - Testfälle?
  - Executables?
  - …
- Wie ist das KM aufgesetzt?
  - Eingesetztes Produkt
  - Prozesse: wann darf wer den Code verändern?
  - Verantwortliche: wer darf Releases und Versionen erstellen?

Capgemini sd&m

Zusammen. Für nachhaltigen Erfolg.