SOFTWARE ENGINEERING

Elite Graduate Program

# Projektmanagement: Risiko-, Änderungs- und Konfigurationsmanagement

**Martin Wirsing**

**Institut für Informatik**
**Ludwig-Maximilians-Universität München**

**WS 2006/07**

Ludwig Maximilians Universität München

# Ziele

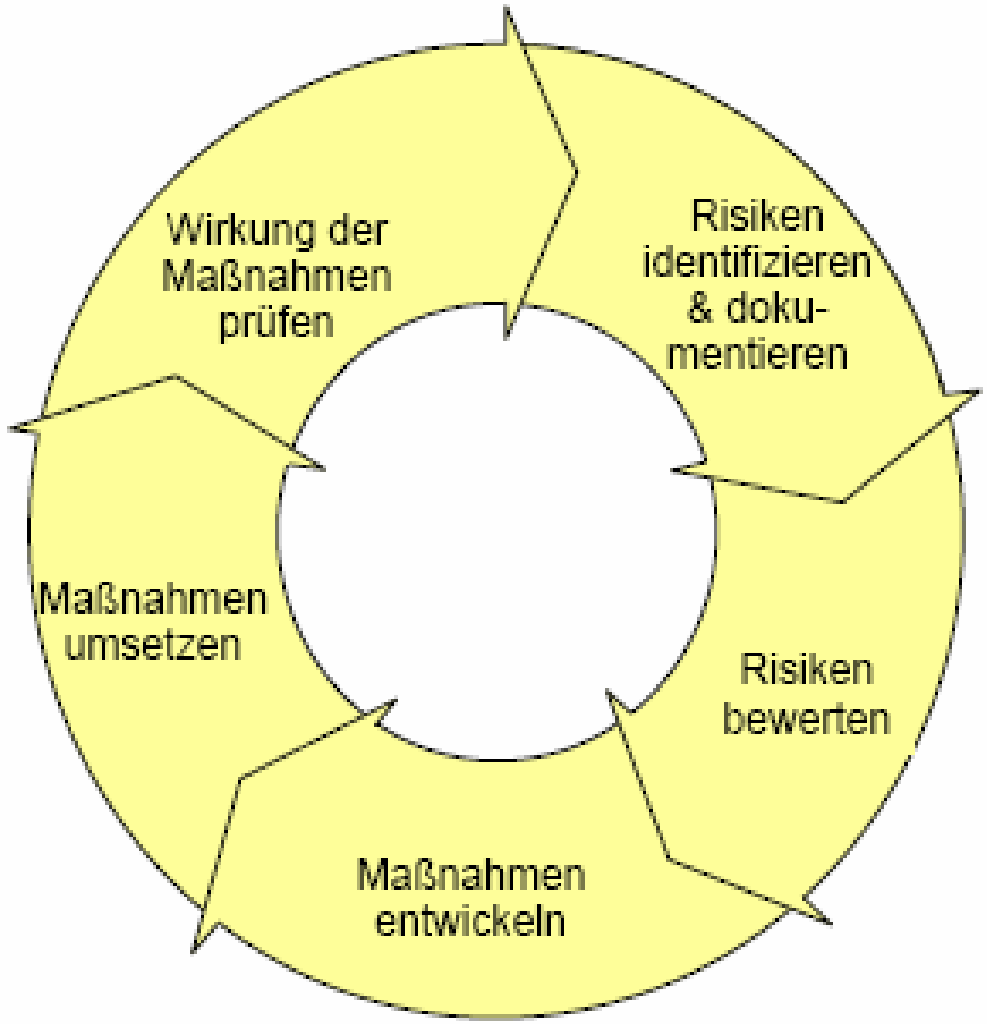
- **Projektrisiken einschätzen und vermeiden/verringern lernen**
- **Änderungsmanagment mit Change Requests kennen lernen**
- **Techniken des Konfigurationsmanagments kennen lernen**

# Motivation: Herr Müller und die Sommerreifen

- Das hab ich kommen sehen!
  - **15. November 2005: 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!

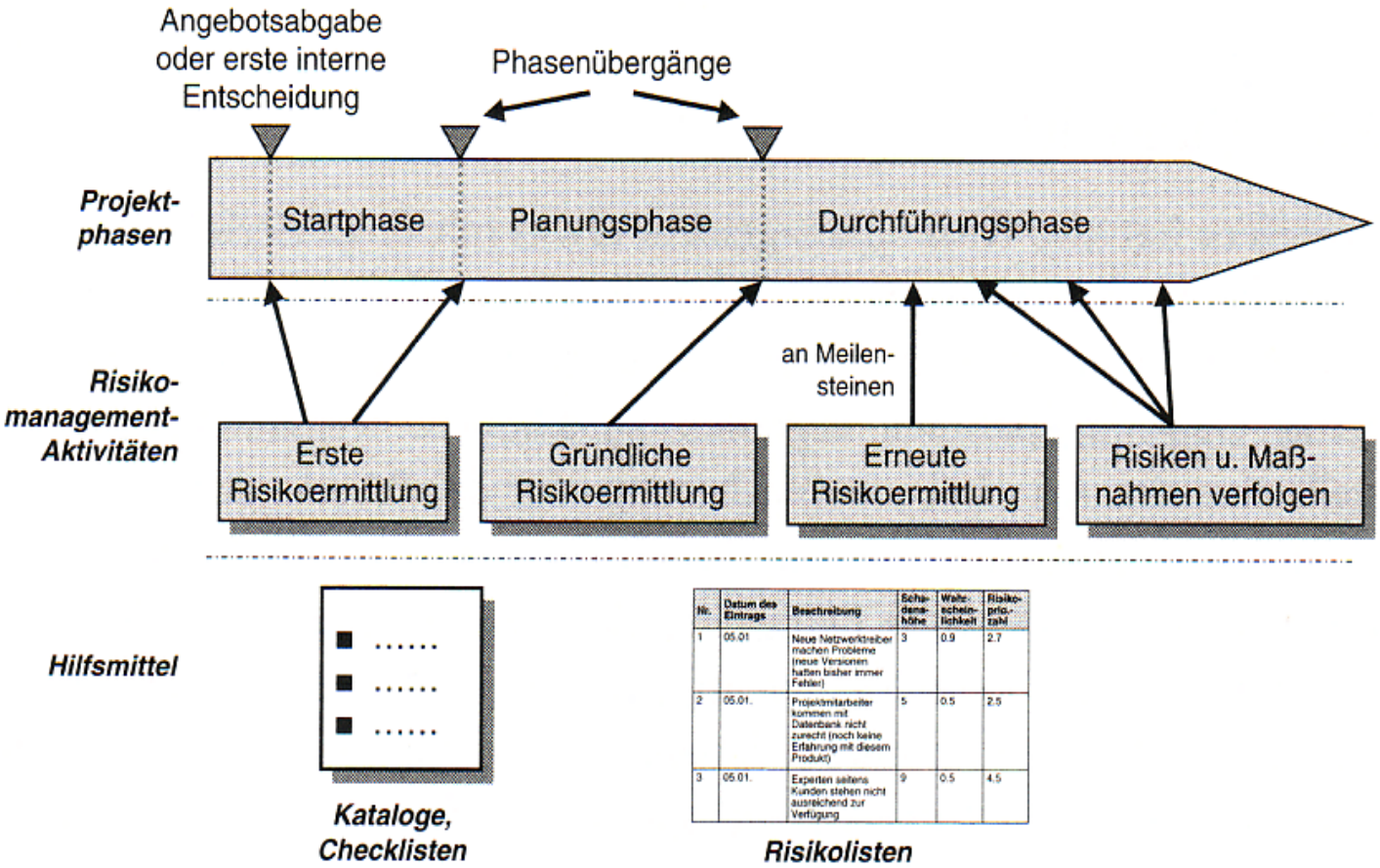
# Risikomanagement im Projekt

- 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.

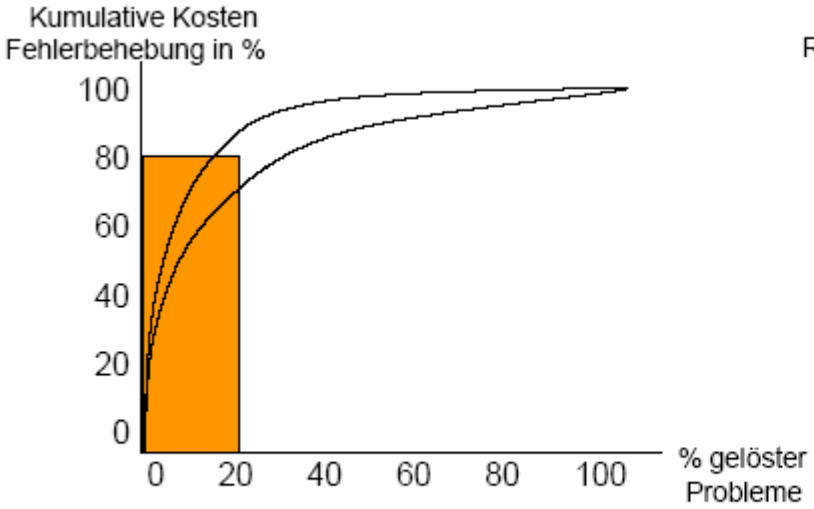


# Risikomanagement und Projektphasen

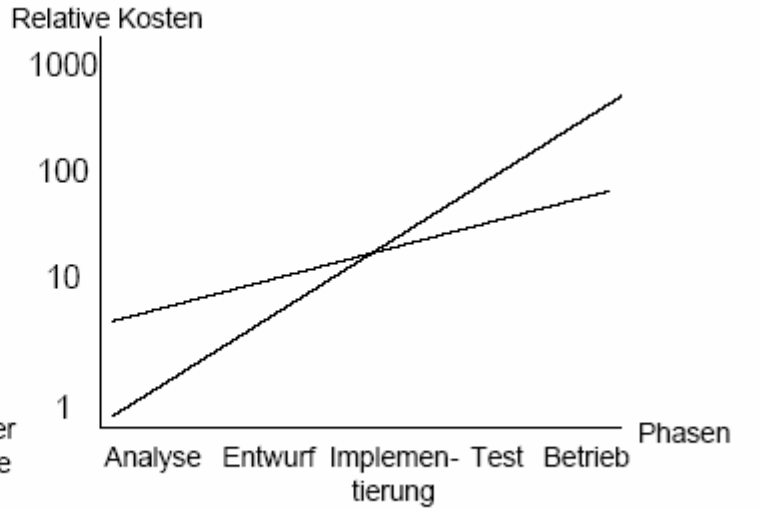
## Risikomanagementaktivitäten während der Projektdurchführung



# Pareto-Regel




**Boehm: Relativer Anteil Überarbeitungskosten**



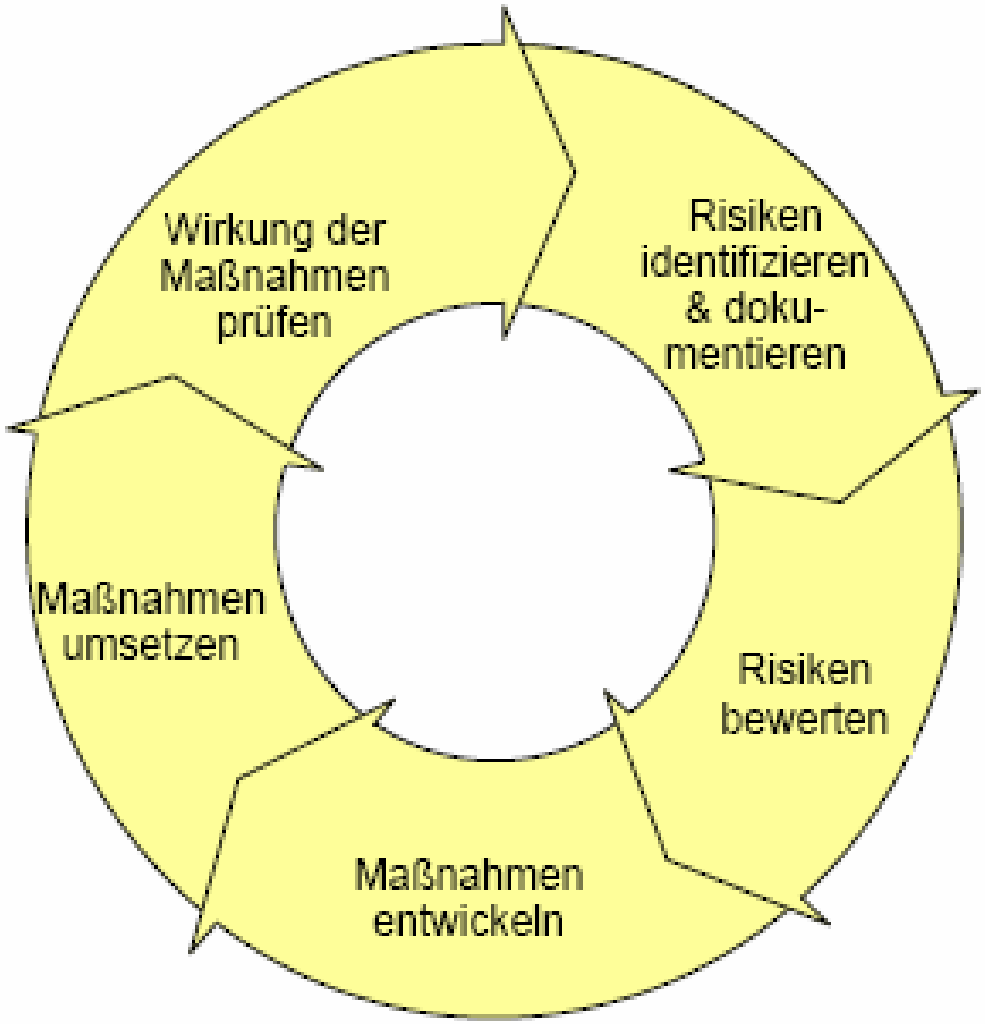

**Boehm: Relative Kostenerhöhung Fehlerverschleppung**

- **Beobachtungen:**
  - Pareto-Regel: 20% der Probleme verursachen 80% der Kosten
  - Behebungsverzögerung: je nach Phase bis zu 1000-fache Kosten
- **Konsequenzen:**
  - Fokussierung auf wichtige Risiken
  - Frühzeitige Erkennung und Vermeidung

**Vilfredo Federico Pareto** (gebürtig *Wilfried Fritz Pareto*; 1848 - 1923) italienischer Ingenieur, Ökonom und Soziologe

# 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 (zur Klassifikation):
  - **Technik**
  - **Fachlichkeit**
  - **Team**
  - **Vorgehen im Projekt, Planung**
  - **Auftraggeber**



# Beispiel SENSORIA

| Risk |
| --- |
| Scientific development fails |
| Tool development/integration fails |
| Scalability problems |
| Results incompatible with future technologies |
| Quick obsolescence of results |
| Fragmentation of results |
| Internal communication difficulties |
| Key researcher leaves project |
| Partner leaves project |
| Delayed Deliverables because of vacation period |
| Low Quality of Deliverables |

**Wissenschaftliche Risiken**

**Technologische Risiken**

**Organisatorische Risiken**

# Risikoquellen

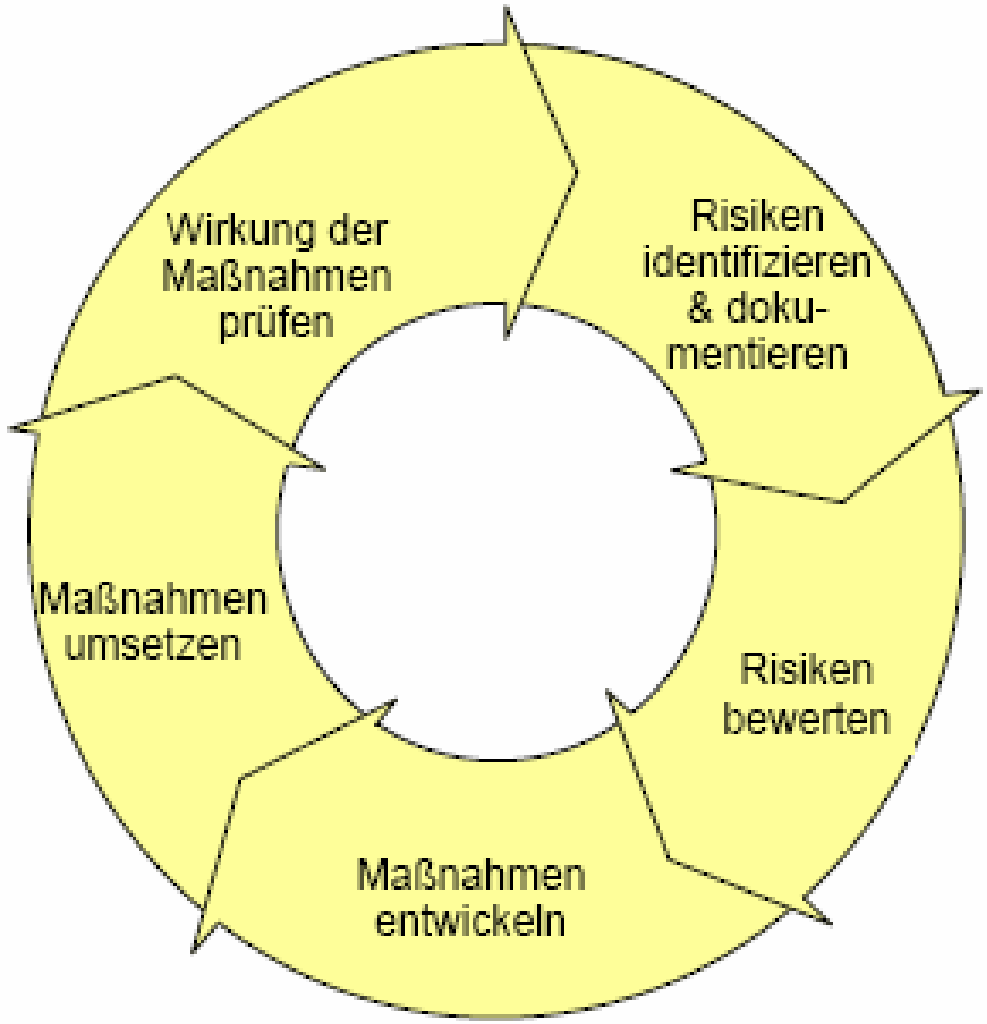
| Anforderungen |
| --- |
| ▪ Anforderungen insgesamt unausgereift oder in Teilen noch unklar |
| ▪ Voraussehbar zahlreiche Änderungen |
| ▪ Änderungsfreudigkeit des Kunden/Auftraggebers |
| **Technik** |
| ▪ Unrealistisches Design |
| ▪ Einzusetzende Technologie wird erst im Laufe des Projekts am Markt verfügbar |
| ▪ Unklar, ob geforderte Performanz erfüllt werden kann |
| ▪ Erstmaliger Einsatz neuer Tools, Soft- oder Hardwarekomponenten |
| **Anwendung** |
| ▪ Hohe Komplexität, viele ungelöste Probleme |
| ▪ Fehlende Erfahrung mit Teilaufgaben |
| ▪ Keine Erfahrung in neuer Anwendungsdomäne |
| **Kunde** |
| ▪ Konkurrierende Interessengruppen, Widerstände (z.B. von Anwendern) |
| ▪ Keine Erfahrung mit Auftragsvergabe nach außen |
| ▪ Mangelnde Kooperationsfähigkeit oder –bereitschaft |
| ▪ Integrierte Entwicklung Auftraggeber/Auftragnehmer |

# Risikoquellen

| |
|---|
| **Projektauftrag, -abwicklung** |
| ▪ Unrealistische Termine |
| ▪ Unrealistische Budgets |
| **Projektorganisation** |
| ▪ Einsatz von Unterlieferanten (generell) |
| ▪ Bekannte Probleme von Unterlieferanten |
| ▪ Zulieferungen von anderen Projekten oder internen Stellen |
| **Ressourcen** |
| ▪ Drohender Ressourcenentzug wegen anderer, wichtigerer Projekte |
| ▪ Ressourcenengpässe oder unzuverlässige Ressourcenzusagen |
| **Personelle Mängel** |
| ▪ Nicht der richtige Projektleiter (unerfahren, ungeeignet, nicht anerkannt etc.) |
| ▪ Nicht die richtigen Mitarbeiter (mangelnde Qualifikation oder Erfahrung) |
| ▪ Drohender Ausfall von Mitarbeitern (Krankheit, Kündigung) |

# Bewertung von Risiken

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

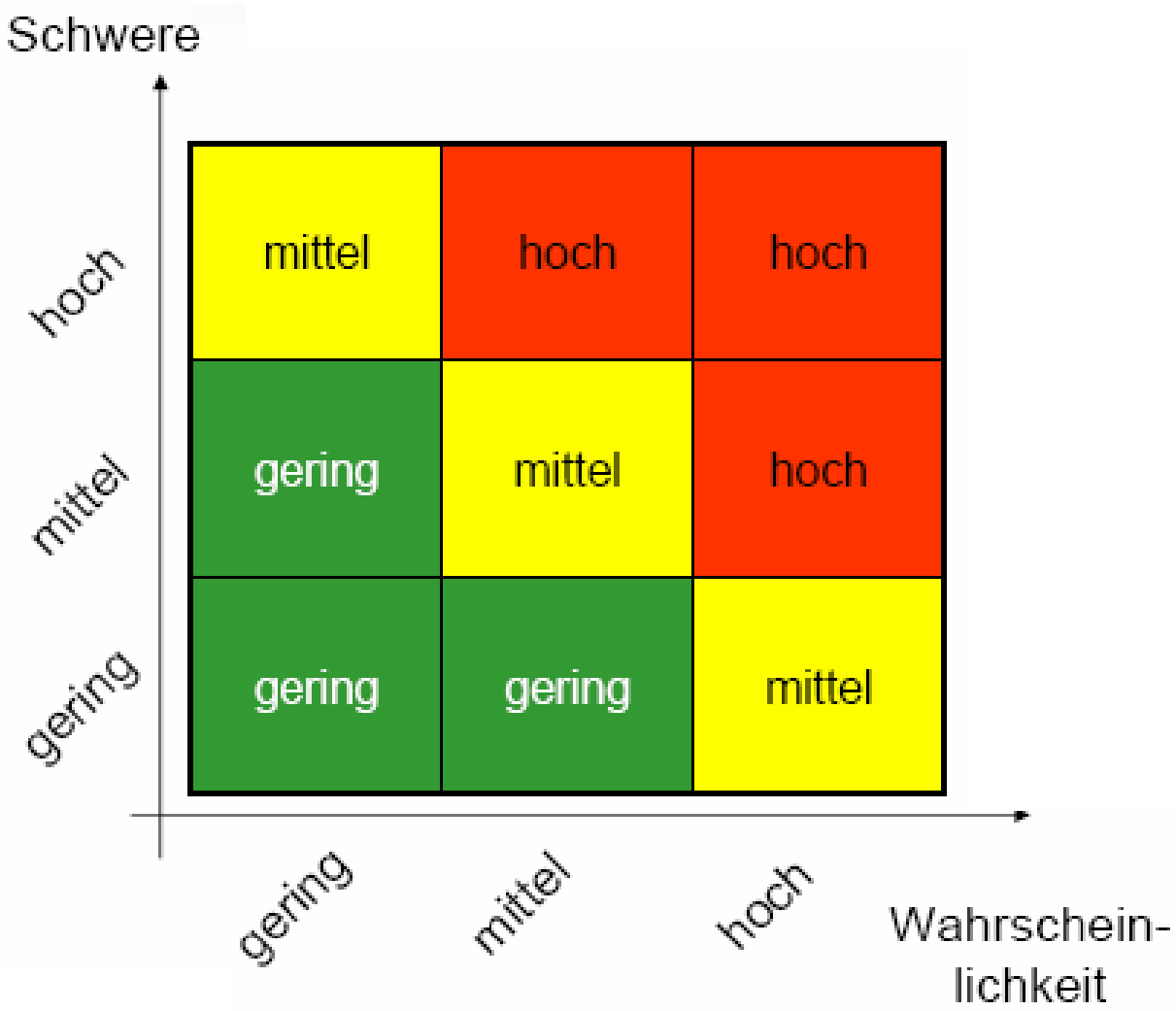


# Gesamtbewertung von Risiken

- 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 Risikobewertung SENSORIA

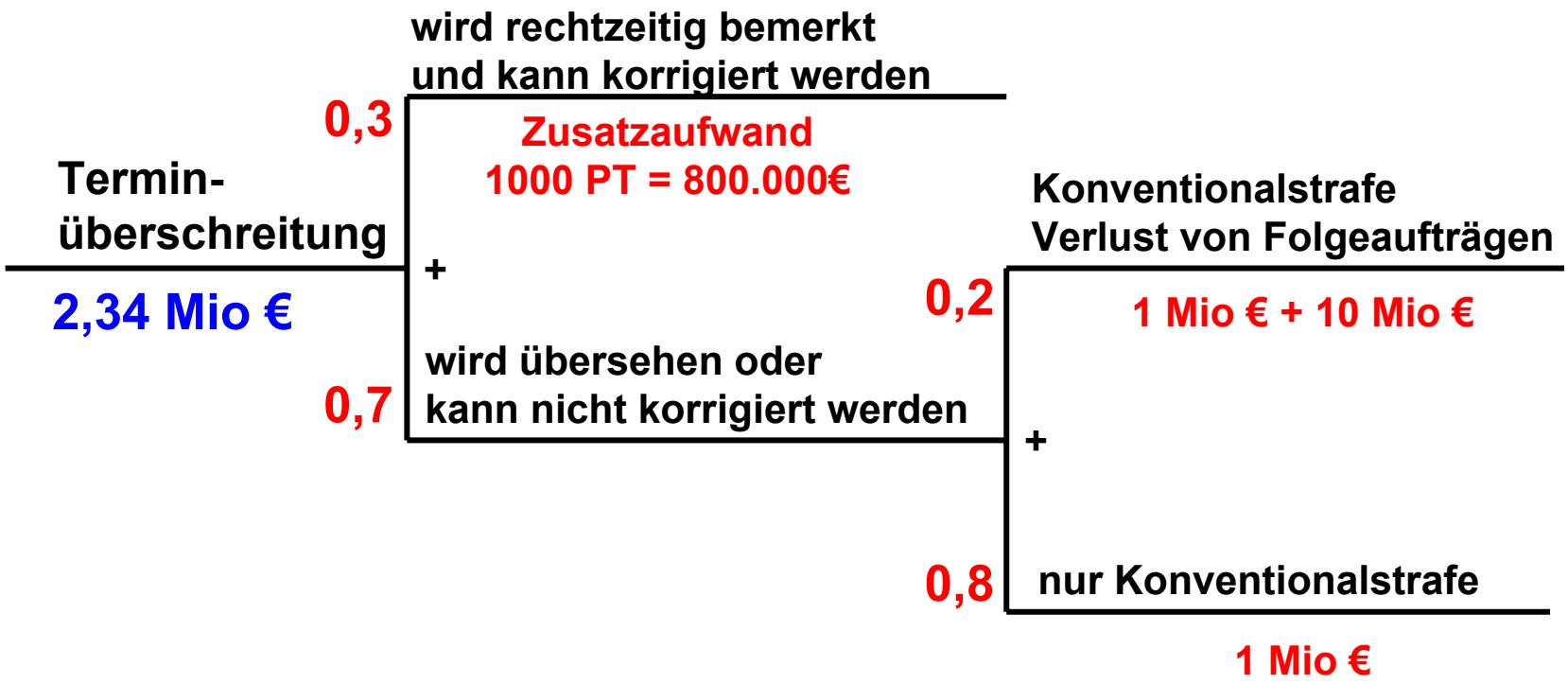
- Each risk is assigned a *likeliness (L)* between 0 (impossible) and 5 (almost certain).
- Each risk is assigned a measure of its possible *impact (I)* on the project between 0 (irrelevant) to 5 (catastrophic).
- We call the product of likeliness and impact the *risk factor (RF)* of the risk.
- Risks are classified into three *categories (Cat)*:
  - *Category A* are risks with a risk factor between 16 and 25. These are the risks that present the greatest danger to the project and on which we should therefore expend the largest amount of effort.
  - *Category B* are risks with a risk factor between 9 and 15. While those risks still could affect the project progress they are unlikely to have major impact on project results.
  - *Category C* are risks with a risk factor of 8 or less. These are risks with limited impact on project results and low likelyhood of happening. Only a small amount of resources should be dedicated to these risks.

# Beispiel Risikobewertung SENSORIA

| Risk | L | I | RF | Cat |
|---|---|---|---|---|
| Scientific development fails | 1 | 5 | 5 | C |
| Tool development/integration fails | 3 | 5 | 15 | B |
| Scalability problems | 3 | 3 | 9 | C |
| Results incompatible with future technologies | 1 | 4 | 4 | C |
| Quick obsolescence of results | 2 | 3 | 6 | C |
| Fragmentation of results | 3 | 5 | 15 | B |
| Internal communication difficulties | 4 | 4 | 16 | A |
| Key researcher leaves project | 5 | 2 | 10 | B |
| Partner leaves project | 4 | 3 | 12 | B |
| Delayed Deliverables because of vacation period | 2 | 3 | 6 | C |
| Low Quality of Deliverables | 1 | 4 | 4 | C |

# Beispiel: Risikoanalyse

## Auswirkungen mit Ereignisbaum abschätzen

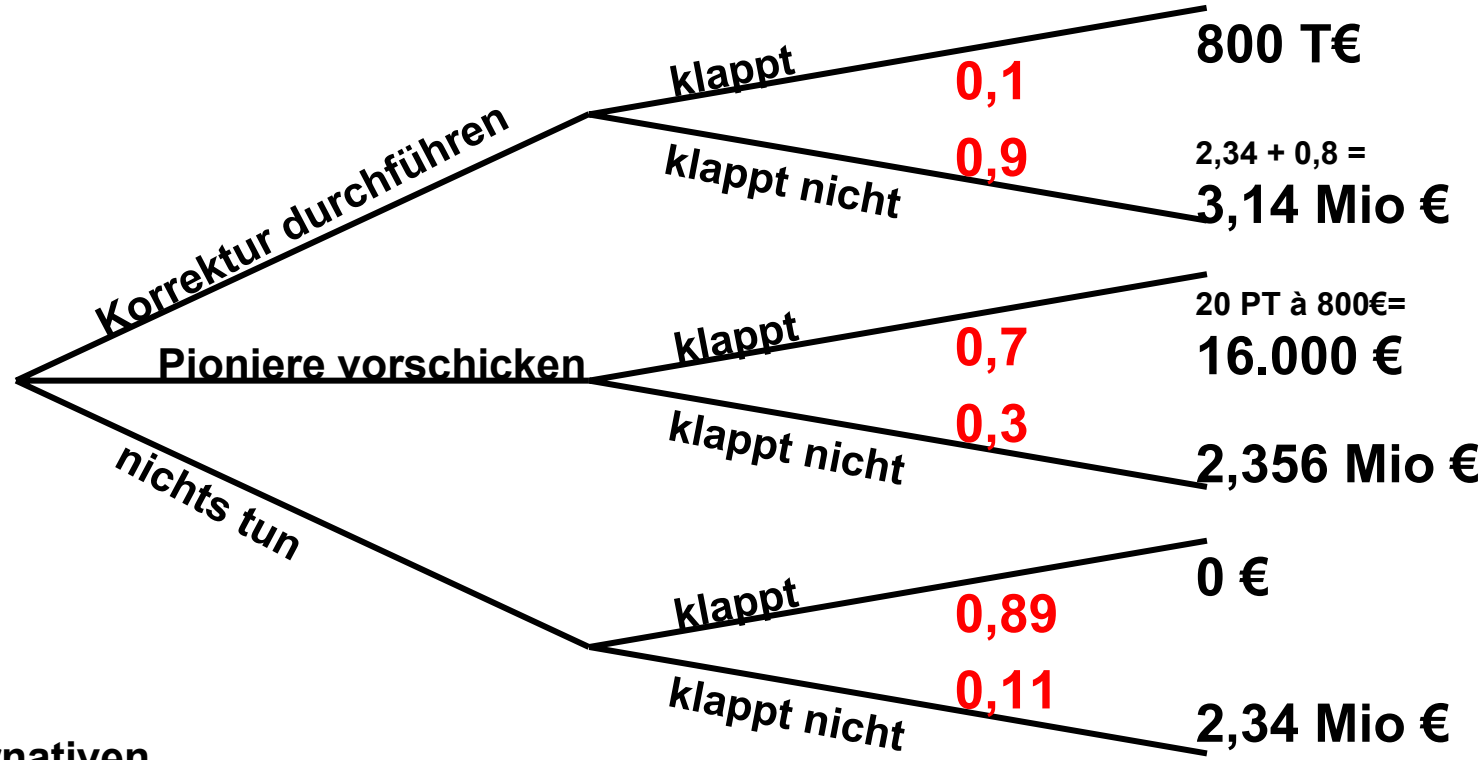



0,3 * 800.000 € + 0,7 * (0,2 * 11 Mio € + 0,8 * 1 Mio €)
= 2,34 Mio €

errechnet
geschätzt

# Beispiel: Risikoprioritätenbildung

## Maßnahmen durch Entscheidungsbaum vergleichen

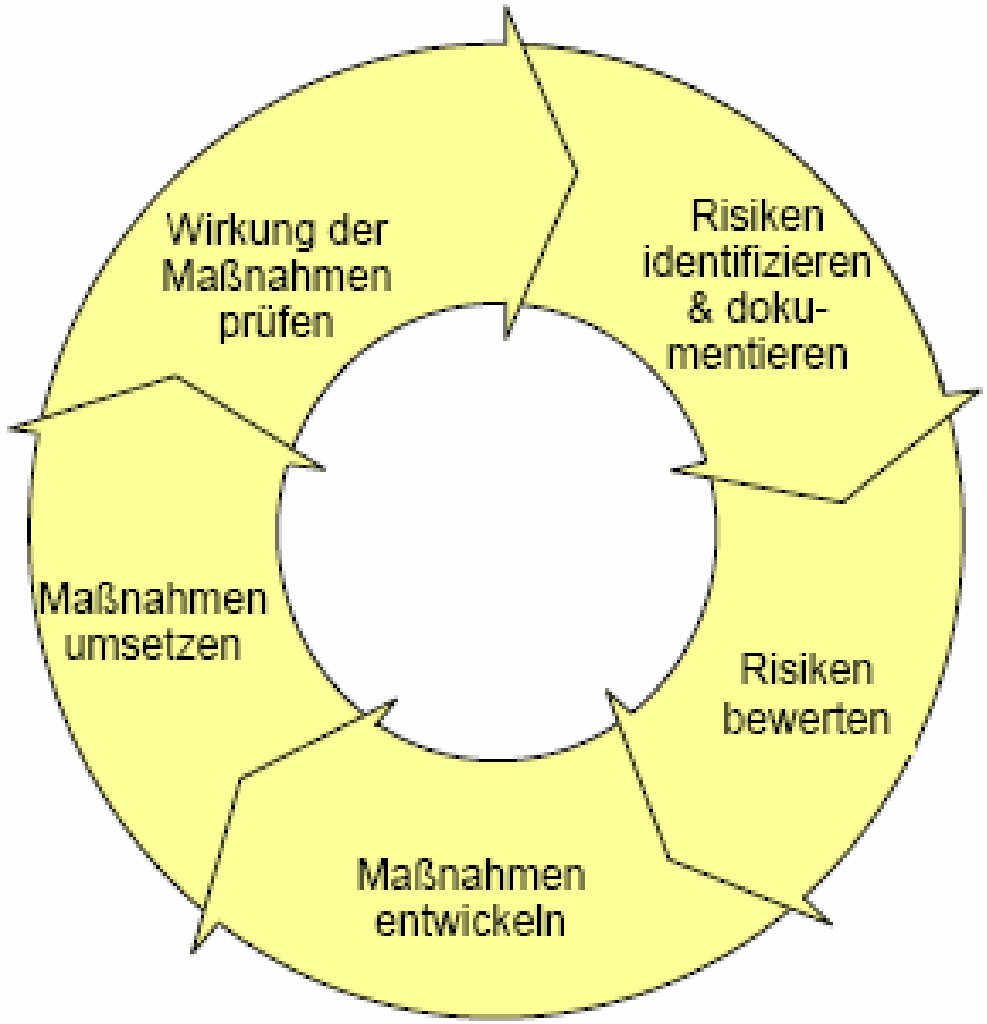



**Vergleich der Alternativen**

| | | |
|---|---|---|
| **nichts tun:** | **0,11 * 2,34 Mio€** | **= 257.400 €** |
| **Pioniere:** | **0,7*16,000 € + 0,3*2,356 Mio€** | **= 718.000 €** |
| **Korrektur:** | **0,1*0,8 Mio € + 0,9* 3,14 Mio€** | **= 2,906 Mio €** |

# 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.



# Beispiel SENSORIA

- **Vorgehensweise**
  - **Beobachtung von „Symptomen“**
  - **Planung von Korrektivmaßnahmen**

# Beispiel SENSORIA

**Risk: Internal Communication Difficulties (Cat. A)**

**Symptoms**

- Low traffic on project mailing lists.
- Little use of knowledge base and other communication infrastructure.
- Few informal meetings between partners.
- Low participation in workshops and project meetings.

**Corrective Measures**

- Encourage use of SENSORIA project communication infrastructure and in particular the mailing lists.
- Provide a whitepaper and scenarios describing the project vision.
- Provide FAQs and summaries of important discussions on the Wiki so that researchers can easily obtain an overview of the progress of Tasks in which they are not directly involved.
- Define a set of core tools and methods which should be used by all project partners to enable easier communication and exchange of work results.
- Facilitate exchange of researchers between partners; to this end the instrument of "Cooperation Grants" was introduced by the Steering Committee.

# Typische Präventivmaßnahmen

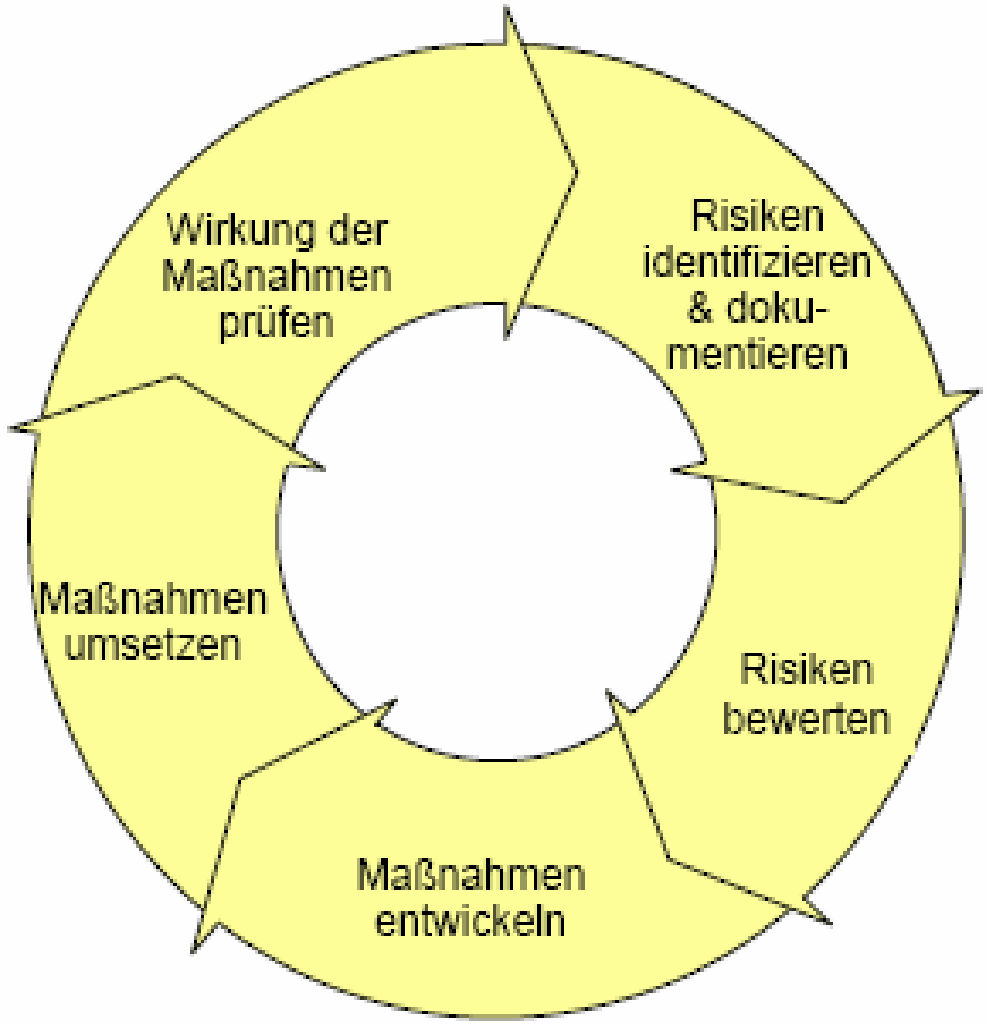
| Bei technischen Risiken: |
| --- |
| ▪ Simulation oder Prototyping von Systemen |
| ▪ Beratung, Reviews durch Unabhängige |
| ▪ Anbieten alternativer Funktionen/Konzepte (bei denen das Risiko nicht auftritt) |
| ▪ Vorherige Erprobung von neuen Technologien |
| **Bei Risiken bezüglich Anforderungen/Kunde:** |
| ▪ Verstärktes Bemühen um den Kunden, Kontaktpflege |
| ▪ Kundenworkshops |
| ▪ (Frühzeitige) Abnahme von Zwischenergebnissen vereinbaren |
| ▪ Pflichten des Kunden vertraglich absichern |

# Weitere Beispiele für Präventivmaßnahmen

| Bei Ressourcenproblemen: |
| --- |
| ▪ Schulung, Coaching von Mitarbeitern/Projektleiter |
| ▪ Einsetzen von Mitarbeitern in ähnliche Projekte im Vorfeld |
| ▪ Abwanderungsgefährdete Mitarbeiter zu halten versuchen |
| ▪ Prioritäten für die Ressourcenzusage mit Management diskutieren |
| Sonstiges |
| ▪ Preisaufschläge einkalkulieren |
| ▪ „Outsourcen“ von Risiken an Subunternehmer |

# Maßnahmen umsetzen und Wirkung prüfen

- Maßnahmen sind Aufgaben:
  - **ein Verantwortlicher**
  - **ein Termin, bis zu dem die Maßnahme umzusetzen ist**
- 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 Eintrittswahrscheinlichekit?**
  - **Gesamtbewertung**
- Maßnahmen
  - **Maßnahme, Aufgabe**
  - **Verantwortlicher**
  - **Termin**
  - **Status**

# Beispiel: Risikobewertung

## Risikoliste zur Priorisierung von Risiken

| lfd.Nr. | Beschreibung | Wahrschein. | Auswirkung | Gewicht | Gegenmaßnahme | verantwortlich |
|---|---|---|---|---|---|---|
| 1 | Zulieferer fällt aus | 30% | 100 T | 30 T | 2. Zulieferer | Störrle |
| 2 | Terminüber-schreitung | 11% | 2,34 M€ | 257 T€ | Enge Kontrolle | Wirsing |
| | | | | | | |

# 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

# Änderungsmanagement: 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."**
- PL: „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 umfang-reich 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

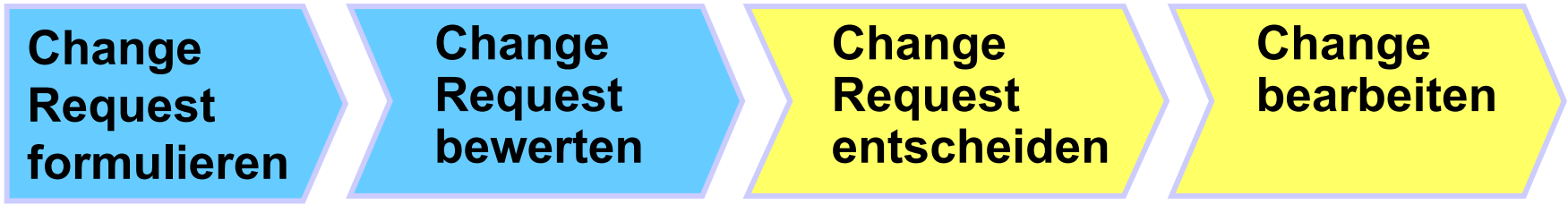
# Änderungsmanagement

- Änderungen an Entwicklungsergebnissen sind unvermeidlich
  - **Änderungen der Aufgabenstellung**
  - **Neue Erkenntnisse bei der Produktentwicklung**
  - **Fehler bei der Produktentwicklung**
- Ziel der formalisierten Behandlung von Änderungen (**Change Request**) ist es, die Konsistenz der Entwicklungsergebnisse zu erhalten
- Änderungen beziehen sich auf definierte Entwicklungsergebnisse (Baselines) und haben Auswirkungen auf
  - **Teilprozesse (Meilensteine)**
  - **Davorliegende Entwicklungsergebnisse (Backtracking)**
  - **Nachgelagerte Entwicklungsergebnisse**
- Änderungen werden einzeln oder als "Versionsentwicklung" durchgeführt

# Change Request

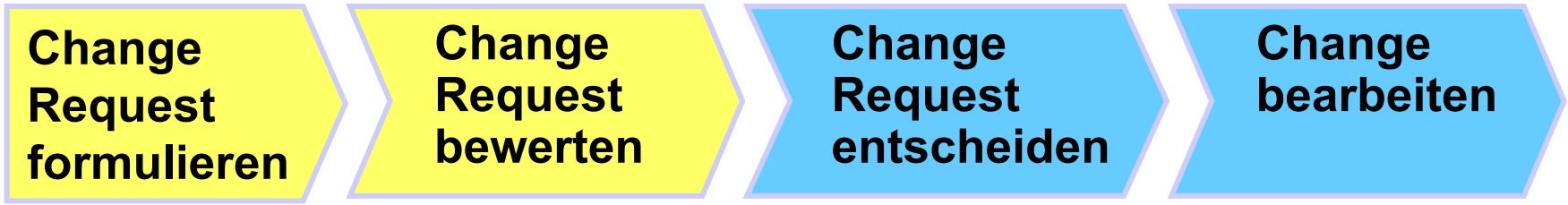
- Ein **Change Request** ist eine Forderung nach einer Abweichung vom ursprünglich vereinbarten Leistungsumfang, vom Auftrag
  - **Es gibt positive und negative Change Requests: Umfang kann steigen oder sinken**
  - **Regelfall: Umfang steigt**
- Change Requests können **vom Auftraggeber und vom Auftragnehmer eingebracht** werden
  - **Regelfall: Auftraggeber**
- Change Requests können nicht nur die Software, sondern z. B. auch die Projektorganisation oder das Projektvorgehen betreffen
- Change Requests können sich auf **bereits erstellte** oder auf **noch zu erstellende** Ergebnisse beziehen

# Change Request-Verfahren




- Das Change Request-Verfahren sorgt dafür, dass Änderungen kontrolliert in das Projekt einfließen
- Ein Change Request sollte folgende Fragen beantworten bzw. stellen:
  - **Was genau ist das Problem, was genau ist die Lösung?**
  - **Wer will das, wer bezahlt das, wer ist davon betroffen?**
  - **Wie wichtig ist der Change Request im Vergleich zu anderen? Was ist die Konsequenz, wenn der Change Request nicht umgesetzt wird?**
  - **Welche Konsequenzen für Zeit, Budget, Projektinhalte hat der Change Request? (Zu ermitteln gemeinsam mit PL und Anforderer)**
- Beim Hinschreiben merkt man oft schon, dass mit Change Request etwas nicht stimmt.

# Change Request-Verfahren




- Möglichkeiten der Entscheidung über Change Request:
  - **zulassen**
  - **ablehnen**
  - **zurückstellen**
- Entscheidung durch
  - **Projektleiter (kleinere Change Requests)**
  - **Lenkungskreis (größere Change Requests)**
- Die Änderung in die Projektplanung einarbeiten
  - **Planung ändern**

# Hinweise zum Change Request -Verfahren

- **Bearbeiten von Change Request-Anträgen** dauert und **kostet Geld**, egal, ob diese umgesetzt werden oder nicht
- Bei **Projektbeginn** schon die Spielregeln für **Change Request-Verfahren bekannt machen**, z. B. im Angebot bzw. im Projektauftrag beschreiben
- Change Requests technisch managen:
  - Tabellenkalkulation
  - Spezialsoftware, oft in Verbindung mit Konfig-Management und Anforderungen
  - Freeware: BugZilla
- Zurückgestellte Change Requests sind oft Basis für eine Folgestufe des Projekts
- Überspitzt formuliert ist ein Change Request-Verfahren ein **Change-Verhinderungs-Verfahren**:
  - **Ein Change Request, 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 Change Request-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 Change Requests des Auftraggebers
  - **Anbieter finanziert das Projekt über Change Requests**
  - **Anbieter optimiert seinen Gewinn über Change Requests**

# 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.**

# Konfigurationsmanagement : 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**

## Konfigurationsmanagement

- Ziel:

  Produkt ist hinsichtlich seiner funktionellen (Code) und äußeren (zugehörige Information) Merkmale jederzeit identifizierbar
- Funktion:
  - Sicherstellung der Identifizierbarkeit, Verfolgbarkeit, Kontrollierbarkeit, Status
  - Darstellung der Zusammenhänge und Unterschiede zwischen verschiedenen Konfigurationen
  - Sicherstellung der Verfügbarkeit früherer Konfigurationen
  - Sicherstellen der Integrität (Gültigkeit, Reifegrad) von Produkten
- Ansatz: Explizite Verwaltung von Produkten und ihrer Versionen

## Konfiguration

- Eine **Konfiguration** ist eine (konsistente) Anordnung von Produkten einschließlich ihrer Beziehungen

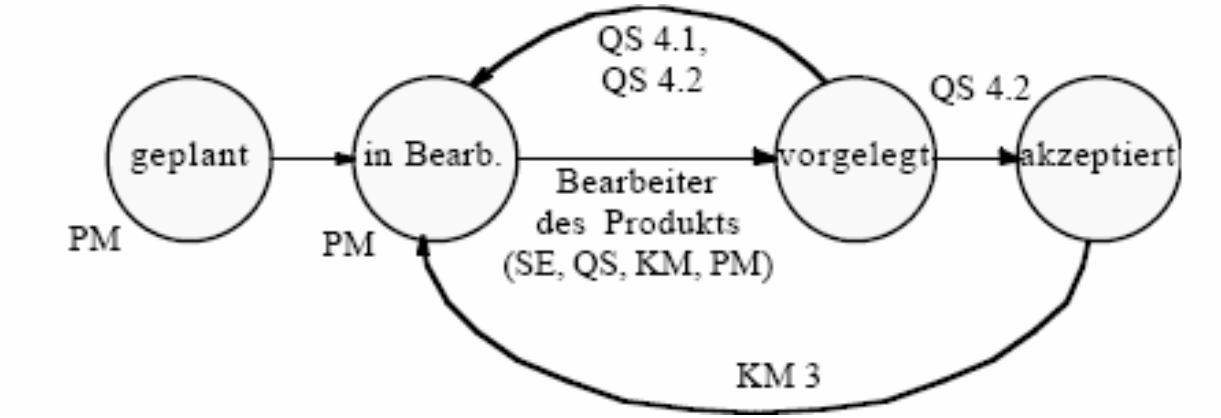
# Produkte und Aktivitäten

## Produkt und Aktivität

| von | | Produkt | nach | |
|---|---|---|---|---|
| Aktivität | Zustand | | Aktivität | Zustand |
| SE 1 | akzeptiert | Anwenderforderungen | — | — |
| SE 2.1 | in Bearb. | Systemarchitektur | — | — |
| — | — | Systemarchitektur. *Anforderungszuordnung* | SE 1, SE 2.5, SE 2.6, SE 3, SE 4-SW, PM 4, PM 5 | vorgelegt |

- Aktivitäten:
  - Benötigen, erzeugen, modifizieren Produkte
- Produkte:
  - Synchronisieren, beeinflussen Aktivitäten

# Konfigurationsmanagement: Produktzustände




- Produktzustände:
  - Definiert auf allen Produkten des V-Modells
  - Verbinden KM mit allen weiteren Modulen
  - Ermöglichen Produktverwaltung
- Zusammenhänge:
  - KM: Erfassung Produkt, Produktstatus, Produktänderung
  - QS. Überprüfung/Abnahme Produkt
  - PM: Planung Produkt

## Aufgaben Konfigurationsmanagement

- Planung Konfigurationsmanagement
- Bereitstellen Produkt-/Konfigurationsmanagement
- Bereitstellung Änderungsmanagement
- Bereitstellung von elementaren Diensten
  - Daten administrieren:
    - Zentrale, projektübergreifende Haltung aller Daten
    - Konsistente, vereinheitlichte Datendefinitionen
  - SW-/HW-Produkte katalogisieren: Vorbereitung Wiederverwendung
  - Schnittstellen koordinieren: Sicherung kompatibler Schnittstellen
  - Ergebnisse sichern: Wahrung des erreichten Projektstands
  - KM-Dokumentation führen zur Erstellung von Detailunterlagen und Übersichten verschiedenster KM-Belange,
  - Release-Management: Kontrollierte Konfigurationsfreigabe- und verteilung
  - Projekthistorie führen: Nachvollziehbare Dokumentation über den Projektverlauf

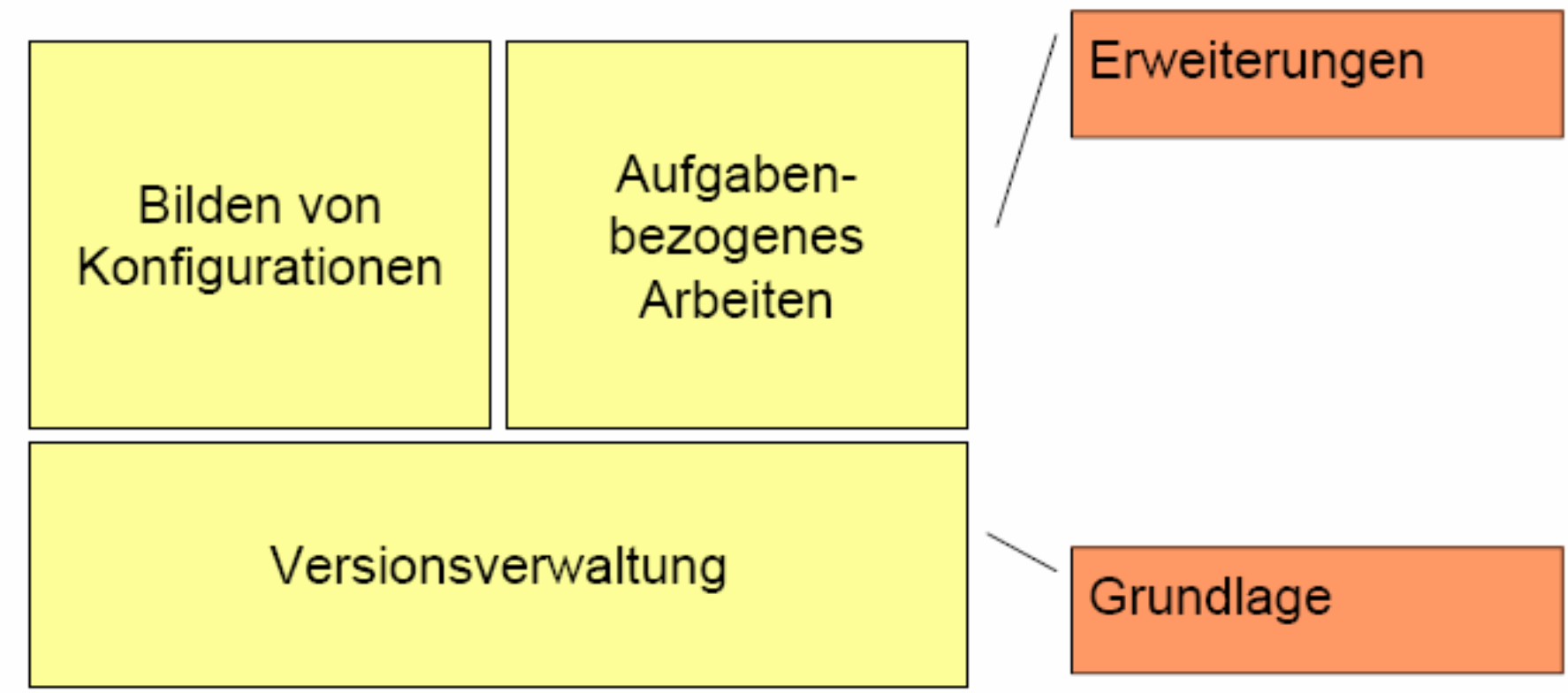
# Techniken des Konfigurationsmanagements

- Einfachster Fall: Konfigurationsmanagement durch gemeinsame Ablage
- Versionsverwaltung
- Bilden von Konfigurationen
- Aufgabenbezogenes Arbeiten

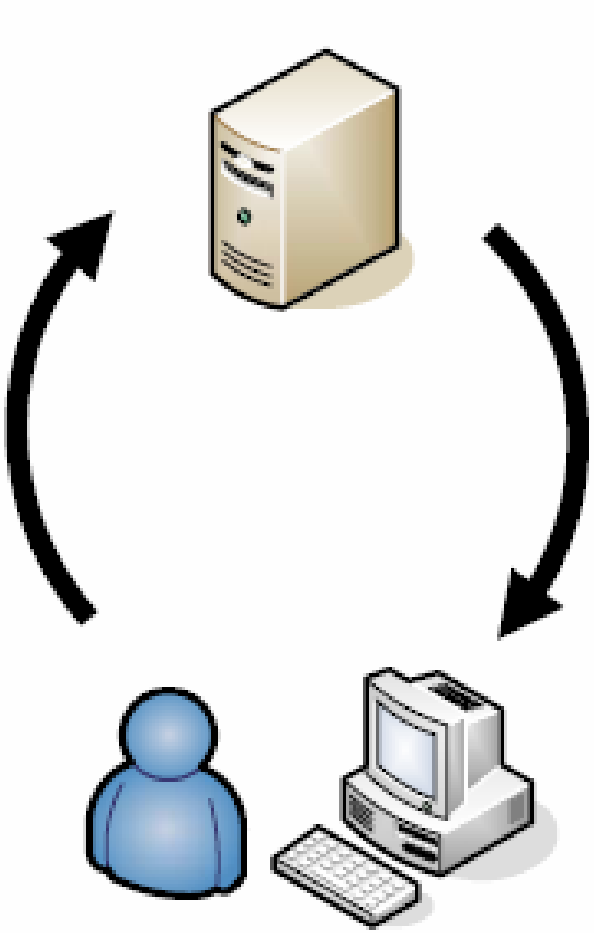
# Einfachster Fall: Konfigurationsmanagement durch gemeinsame Ablage

- 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

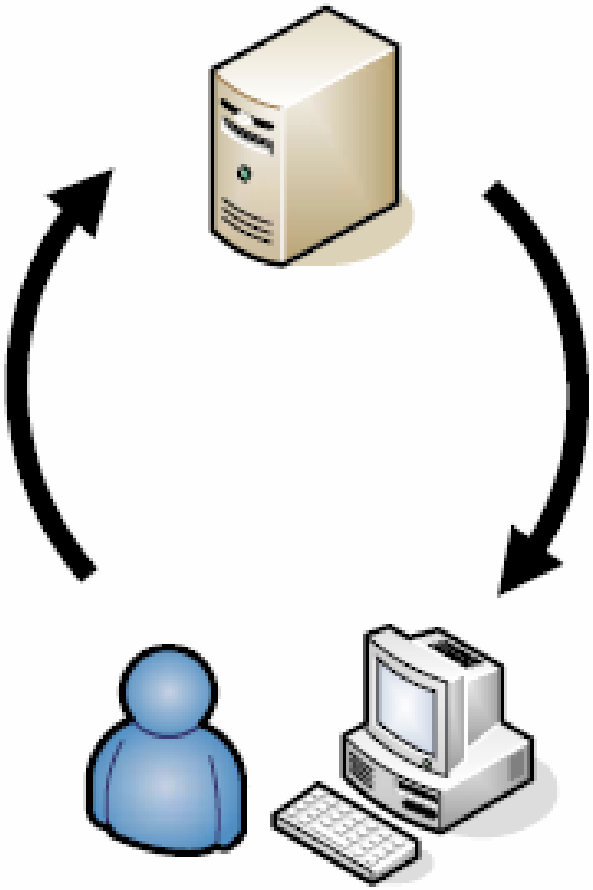


# Versionsverwaltung



- 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: checkout.**
  - **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: 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?**

# Zusammenfassung

- **Risikomanagement** umfasst sowohl die **Identifikation und Bewertung von Risiken** als auch die **Planung, Durchführung und Prüfung der Wirkung von Korrektivmaßnahmen**.
- **Risiken** werden nach ihrer **Eintrittswahrscheinlichkeit und ihren Auswirkungen auf das Projekt (Schwere)** bewertet und der nach Formel verrechnet:

  ***Wahrscheinlichkeit * Schwere***
- **Änderungsmanagement**: Ziel der formalisierten Behandlung von Änderungen (**Change Request**) ist es, die Konsistenz der Entwicklungsergebnisse zu erhalten
- Ein **Change Request** ist eine Forderung nach einer Abweichung vom ursprünglich vereinbarten Leistungsumfang, bezieht sich typischerweise auf definierte Entwicklungsergebnisse oder die Organisation des Projekts und hat Auswirkungen auf
  - **Teilprozesse (Meilensteine),**
  - **davorliegende und nachgelagerte Entwicklungsergebnisse**
- Eine **Konfiguration** ist eine (konsistente) Anordnung von Produkten einschließlich ihrer Beziehungen. Aufgabe des Konfigurationsmanagement ist die Verwaltung von Produkten und Konfigurationen und deren Änderungen.
- Die Techniken des Konfigurationsmanagement umfassen
  - **gemeinsame Ablage, Versionsverwaltung, Bilden von Konfigurationen und**
  - **aufgabenbezogenes Arbeiten**