

Achtung:  
Deutsch/Englisch-Gemisch

## Vorlesung "Softwaretechnik" Buchkapitel 16 **Software-Prozessmodelle (d.h. Softwareprozess-Modelle)**

L. Prechelt, B. Brügge, A. Dutoit, K. Schneider  
Freie Universität Berlin, Institut für Informatik  
<http://www.inf.fu-berlin.de/inst/ag-se/>

- Elemente von Prozessmodellen
  - Rollen, Artefakte, Aktivitäten
- Wasserfallmodell
- Reparatur 1: Iteration
  - Prototypmodell, Evolutionäre Modelle
- Reparatur 2: Flexiblere Planung
  - Agile Methoden
- Prozessmodell-Auswahlkriterien
- Anpassbare Prozessmodelle
  - Rational Unified Process
  - V-Modell XT
- Was bedeutet Agile Methode?
  - Beispiel:  
eXtreme Programming (XP)

# Wo sind wir?: Taxonomie "Die Welt der Softwaretechnik"

## Welt der Problemstellungen:

- Produkt (Komplexitätsprob.)
  - Anforderungen (Problemraum)
  - Entwurf (Lösungsraum)
- Prozess (psycho-soziale P.)
  - Kognitive Beschränkungen
  - **Mängel der Urteilskraft**
  - **Kommunikation, Koordination**
  - Gruppendynamik
  - Verborgene Ziele
  - **Fehler**

## Welt der Lösungsansätze:

- Technische Ansätze ("hart")
  - Abstraktion
  - Wiederverwendung
  - Automatisierung
- Methodische Ansätze ("weich")
  - Anforderungsermittlung
  - Entwurf
  - **Qualitätssicherung**
  - **Projektmanagement**

- Einsicht: Man sollte die Gesamt-Vorgehensweise nicht in jedem Projekt neu erfinden
  - sondern sich auf vorhandene Erfahrungen abstützen
- Prinzipien:
  - **Planung und Koordination:** Es senkt das Risiko, wenn alle Beteiligten im Voraus erkennen können, was wann getan werden muss
    - Es ist schwierig, dies im Einzelfall korrekt abzuschätzen
  - **Korrekturen:** Versuche, den Prozess so zu gestalten, dass die unvermeidlich auftretenden Fehler gut ausgeglichen werden können
  - **Iteration:** Es senkt das Risiko, wenn das Projekt in kurzen Abständen evaluierbare Versionen der Software hervorbringt

# Die wichtigsten Entscheidungen für ein SW-Projekt

---

- Projektziele
- Zeitplan und Budget
- Projektorganisation
- **Verwendetes Prozessmodell**
- Verwendete Technologie, Werkzeuge und Methoden
- Teammitglieder

## Software-Prozess:

- Die Abläufe, die in einem Softwareprojekt geschehen
  - entweder *deskriptiv* gemeint  
(also beschreibend, was tatsächlich geschieht)
  - oder *präskriptiv* (also als Vorschrift, wie es abzulaufen hat)
- auch Teile des Gesamtprozesses werden oft Prozess genannt
  - z.B. der Testprozess

## Softwareprozess-Modell (Software-Prozessmodell):

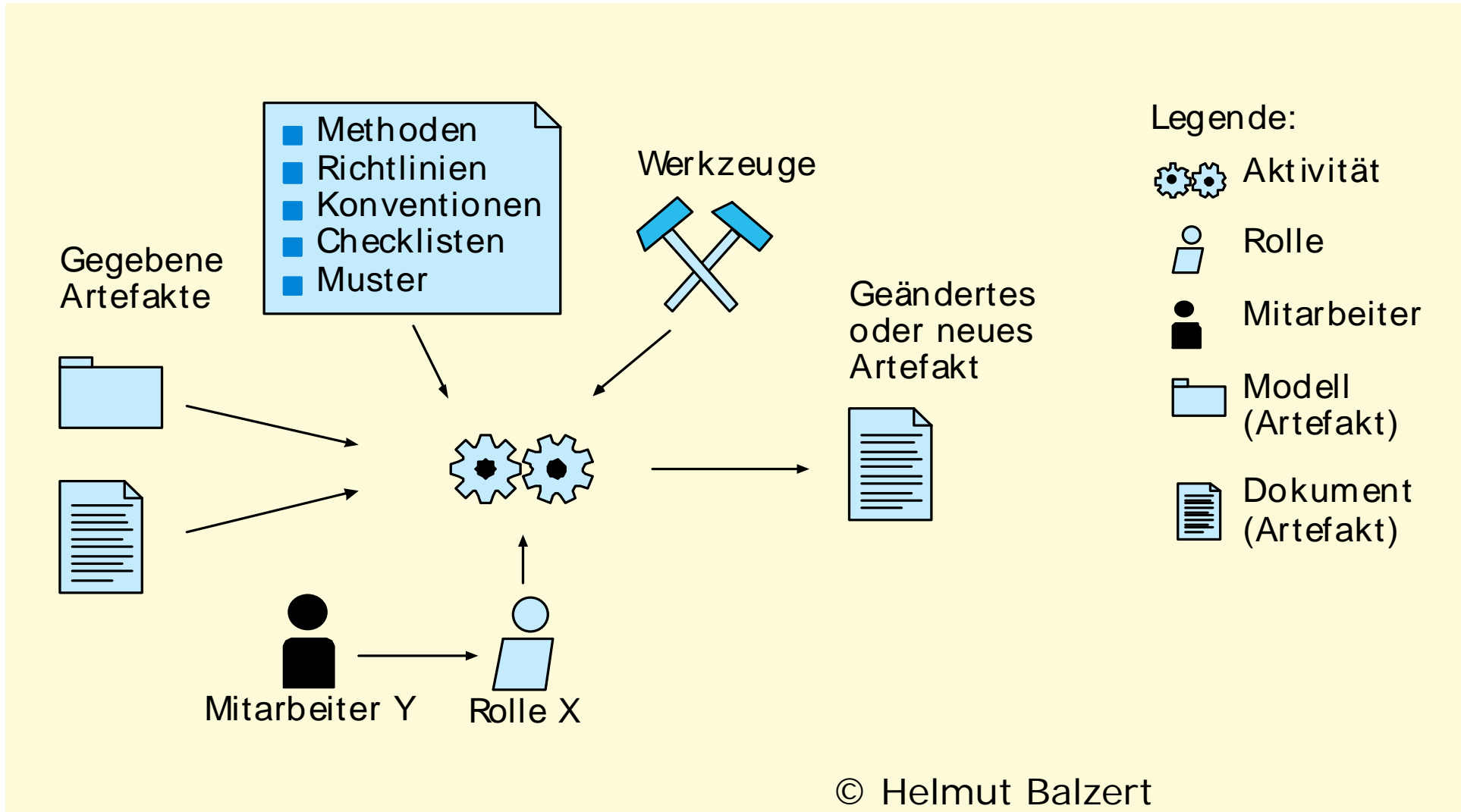
- Eine *Schablone* die die Gemeinsamkeiten der Abläufe in vielen ganz verschiedenen Projekten erfasst
  - meistens zumindest halbwegs präskriptiv gemeint

# Definitionen: Rollen, Aktivitäten, Artefakte

Prozesse werden beschrieben mittels folgender Hauptkonzepte:

- **Rolle:**
  - Abstraktion der Aufgabe einer Person in einer bestimmten Situation
  - z.B. "Entwerferin", "Entwickler", "Tester", "Projektleiterin" usw.
    - die selbe Person hat in einem Projekt oft mehrere Rollen
- **Aktivität:**
  - Abstraktion für eine Sorte zielgerichteten Handelns in einem Projekt
  - z.B. "Anforderungvalidierung", "Architekturentwurf", "Modulentwurf", "Kodieren eines Moduls", "Test eines Moduls" usw.
- **Artefakt:**
  - Abstraktion für eine Sorte von Arbeitsergebnis einer Aktivität
  - z.B. "UML-Klassendiagramm", "Programmcode", "Testfall", "Testbericht" usw.

# Rollen, Aktivitäten, Artefakte (2)



- Ein Prozess (deskriptiv) ist im Prinzip ungeheuer komplex
  - viel komplexer als das entstehende Produkt, da ja für jedes Detail im Produkt zahlreiche Abläufe vor sich gegangen sind
- Man meint deshalb mit "Prozess" in den meisten Fällen nur eine sehr starke Vergrößerung des tatsächlich Ablaufenden

Aber wie grob?

- Vergrößert man zu wenig,  
hat jedes Projekt einen einmaligen Prozess
  - der zudem nicht vorher definiert ist
- Vergrößert man zu stark,  
haben alle Projekte den selben Prozess
  - und man lernt nichts mehr
- Eine sinnvolle Vergrößerung ist so, dass sich zwar viele Projekte ähneln, aber andere noch verschieden aussehen
  - und dass die Unterschiede dazwischen interpretierbar sind



- Rollen
  - Welche gibt es?  
Wie genau sind sie definiert ("Arbeitsbeschreibungen")?
- Aktivitäten
  - Welche sind vorgesehen? Wie genau sind sie definiert?
  - Entscheidungsspielraum über Einsatz oder Art des Einsatzes?
  - Gibt es Checklisten?  
Vorgaben über Werkzeuge, Methoden, Richtlinien?
- Artefakte
  - Welche sind vorgesehen? Wie genau sind sie definiert?  
Wie verbindlich ist das?
  - Gibt es Vorlagen (Schablonen)?
- Steuerung der Aktivitäten
  - Gibt es einen festen Ablaufplan?  
Oder weichere Kriterien für die Abfolge von Aktivitäten?
  - Sind Eintritts- und Austrittsbedingungen definiert?

- Es gibt eine kleine Zahl verschiedener Aktivitäten, z.B.
    1. Planung
    2. Anforderungsbestimmung
    3. Architekturentwurf
    4. Feinentwurf
    5. Implementierung
    6. Integration
    7. Validierung
    8. Inbetriebnahme
  - Diese Aktivitäten werden der Reihe nach durchlaufen ("Phase")
    - Jede Phase nur einmal
    - Phase N beginnt erst nach Abschluss von Phase N-1
  - Dokumenten-getriebener Prozess
    - alle Ergebnisse jeder Phase liegen in Dokumenten vor
  - Am Ende jeder Phase erfolgt eine gründliche Prüfung der Ergebnisdokumente
    - und dann die Übergabe in die nächste Phase ("Meilenstein")
    - eventuell mit anderem Personal!
- Annahme:
    - Mängel in Phase N werden spätestens in Phase N+1 aufgedeckt
    - und können dann leicht in den Dokumenten beider Phasen korrigiert werden

## 1. Bei unklaren Anforderungen:

- Wenn Anforderungen in der Anforderungsbestimmung nicht gut verstanden werden, braucht man als Hilfe den Entwurf, die Implementierung und die Validierung
- Im Wasserfallmodell führt das zu Chaos, weil späte Änderungen der Anforderungen total das Prozessmodell durchbrechen
  - Entweder die Arbeitsweise mit gründlich ausgearbeiteten Dokumenten wird enorm teuer
  - oder die Dokumente werden nicht mehr korrekt gepflegt

## 2. Bei veränderlichen Anford.:

- Das gleiche gilt, wenn sich Anforderungen irgendwann im Projektverlauf plötzlich von außen ändern können

## 3. Bei nicht beherrschten Architekturen:

- Ähnlich wie bei unklaren Anf.

## 4. Durch "Über die Mauer werfen":

- Kommunikation nur über Dokumente
  - Desaster, wenn Dokumente nicht gelesen werden
- Verschiedenes Personal
- Verständnis für Phänomene von Phase N ist in N+1 weitgehend verloren

Eigentlich ist das Wasserfallmodell nur eine Legende (W.W. Royce 1970).

# 1. Reparatur: Iteration

- Modernere Prozessmodelle empfehlen ein iteratives Vorgehen
  - Projektergebnis nicht "in einem Rutsch" anfertigen, sondern sich in mehreren Schritten "herantasten"

## Vorteile:

- Senkt Komplexität in einzeltem Schritt (→ beherrschbarer)
- Kann mit unklaren oder veränderlichen Anforderungen etc. umgehen
- Verlangt engere Kommunikation der Beteiligten
  - und senkt dadurch die Neigung zum "Über die Mauer werfen"

## Nachteile: (das sind aber beides in Wirklichkeit keine)

- Bewirkt eine gewisse Doppelarbeit (wg. "Zwischenlösungen") und hat deshalb theoretisch höheren Aufwand
- Verlangt engere Kommunikation der Beteiligten

- Prototypmodell:
  - Baue anfangs ein (Teil)System "zum Wegwerfen", um kritische Anforderungen besser zu verstehen
  - Danach folgt noch ein Prototyp oder Wasserfallmodell
- Inkrementelles Modell:
  - Baue das Gesamtsystem schrittweise
  - In jedem Schritt werden nur neue Teile hinzugebaut, aber es wird (theoretisch!) nie etwas Existierendes verändert
- Iteratives Modell (evolutionäres Modell):
  - Baue das Gesamtsystem schrittweise
  - In jedem Schritt werden neue Teile hinzugebaut und wo nötig auch existierende verändert
- Spiralmodell (Risikomodell):
  - Tue in jeder Iteration das, was am stärksten zur Verringerung des kritischsten Projektrisikos beiträgt
  - Das hat alle obigen Modelle als Spezialfälle

## 2. Reparatur: Allgemeine Ziele anstatt konkreter Pläne

- Manche Prozessmodelle planen nicht gleich den Inhalt aller Iterationen von Anfang an
  - sondern geben nur grobe Ziele der Entwicklung vor
- Dies verbessert insbesondere die Bereitschaft, Anforderungsänderungen zu akzeptieren

Bezeichnung solcher Prozessmodelle:

- Agile Prozesse (oder agile Methoden)

**Agile Methoden verlangen sehr enge Kommunikation!**

# Hauptunterschied zwischen Prozessmodellen: Wieviel Planung?

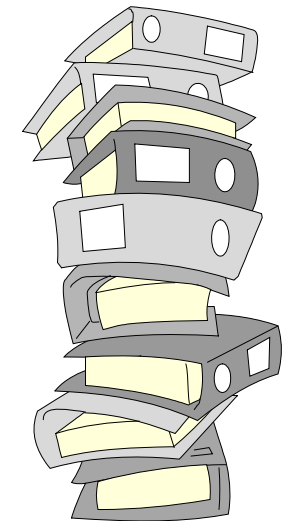
- Konkrete Prozessmodelle unterscheiden sich in vielerlei Hinsicht
- Wichtigste Dimension von Unterschieden: Wie präzise/strikt/weit\_voraus wird geplant?
- Sehr: Wasserfallmodell
  - möglichst präzise und strikt, für das gesamte Projekt im Voraus
- Mittel: Iterative Modelle
  - Planen so weit und so präzise wie möglich
  - nicht strikt (nötige Veränderungen werden akzeptiert)
  - nur für wenige Iterationen im Voraus
- Wenig: Agile Prozesse
  - Nur so viel Planung wie unbedingt nötig
  - Lieber Ziele als Pläne (wg. Flexibilität)

# Wieviel Planung?

nach Barry Boehm

## feingranulare Verträge

- + + klare Arbeitsgrundlage
- + finanzielle Sicherheit
- - enorm aufwändig
- - Änderungen auch für  
AG schwer durchsetzbar





# Wieviel Planung?

nach Barry Boehm

## ad hoc

- + wenig Planungsaufwand
- + individuelle Freiheit
- - Ergebnis unvorhersehbar
- - abhängig von "Helden"

feingranulare  
Verträge



# Wieviel Planung?

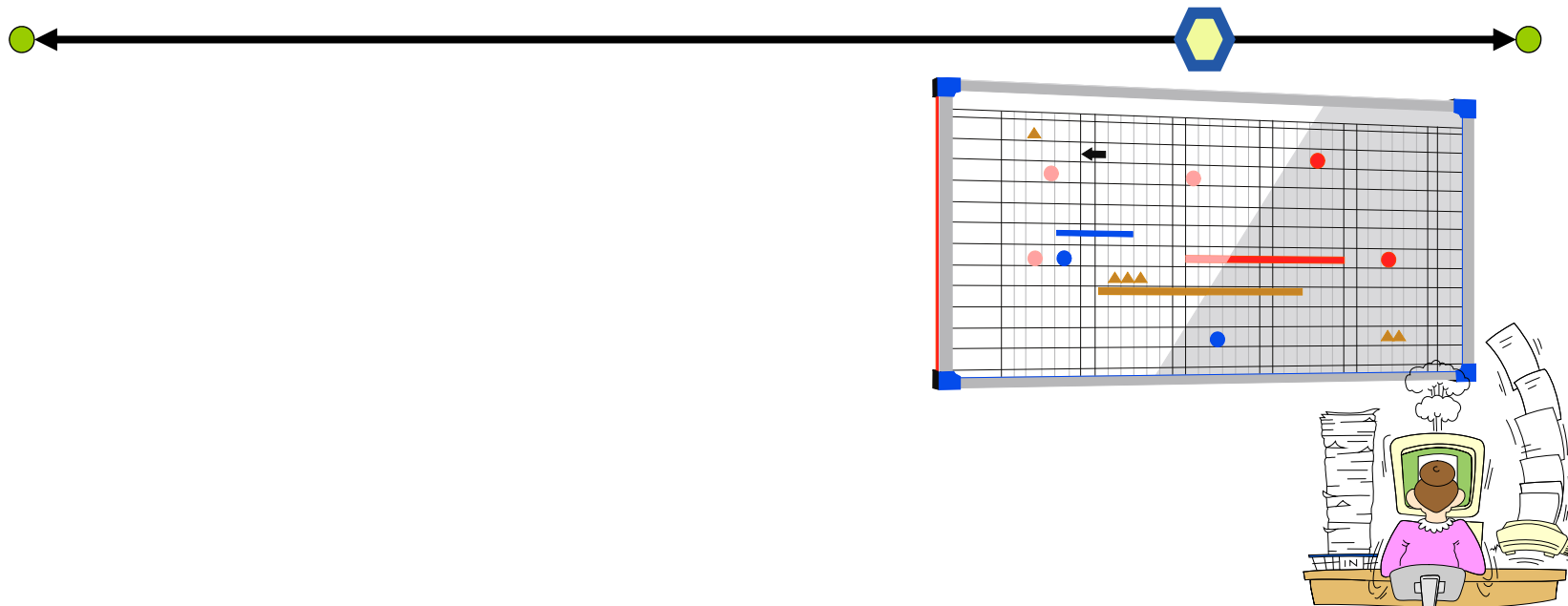
nach Barry Boehm

## Meilenstein- u. Plangetrieben

- + langfristige Vorhersagen
- + gute Zustandskontrolle
- - Änderungen aufwändig
- - unrealistische Annahmen  
schwer zu eliminieren

**feingranulare  
Verträge**

ad hoc



# Wieviel Planung?

nach Barry Boehm

## Meilenstein- u. Risikogesteuert

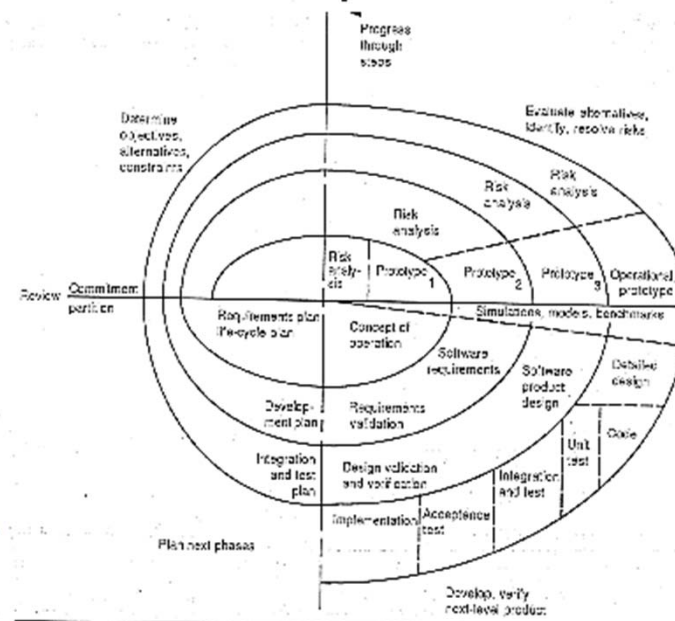
- + + Risiken aktiv ausgeräumt
- + Teilergebnisse früh
- kaum langfristig planbar
- relativ aufwändig

## Meilenstein- u. feingranulare Plangetriebene Verträge

ad hoc



Spiral model



# Wieviel Planung?

nach Barry Boehm

## eXtreme Programming

- + früh Kernergebnisse
- + flexibel für Änderungen
- in sehr großen Projekten  
Zusatzplanung nötig
- viel Selbstdisziplin nötig

ad hoc

Meilenstein- u. Meilenstein- u. feingranulare  
Risikogesteuert Plangetrieben Verträge

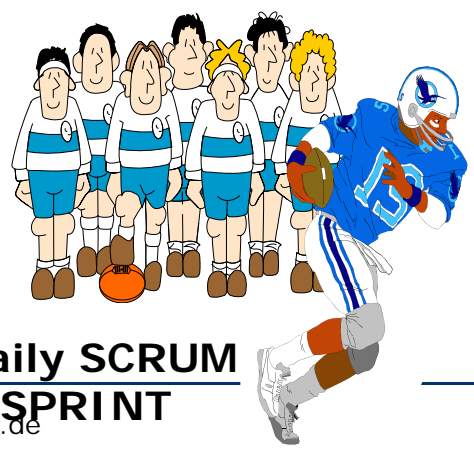
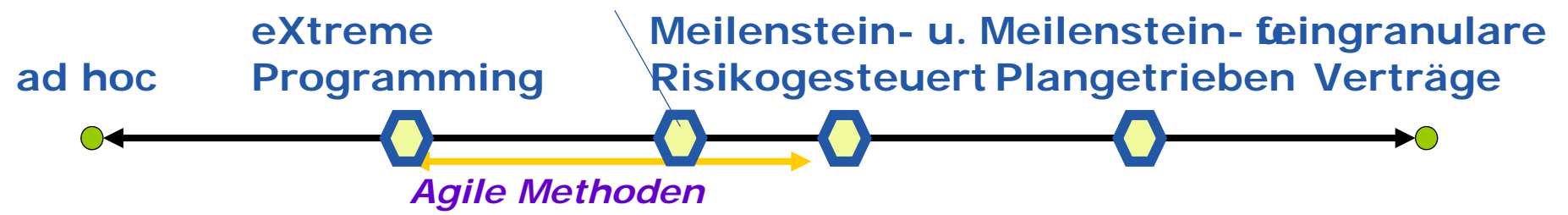


# Was bedeutet agil?

## Planungshorizont nach Maß n. Barry Boehm

### Agiles Beispiel: SCRUM

- + mittelfristig geplant
- + reagiert schnell
- hängt an MA-Qualifikation
- Endprodukt nicht spezifiziert



Daily SCRUM  
+ SPRINT

# Prinzip und Denkweise von Agilen Methoden

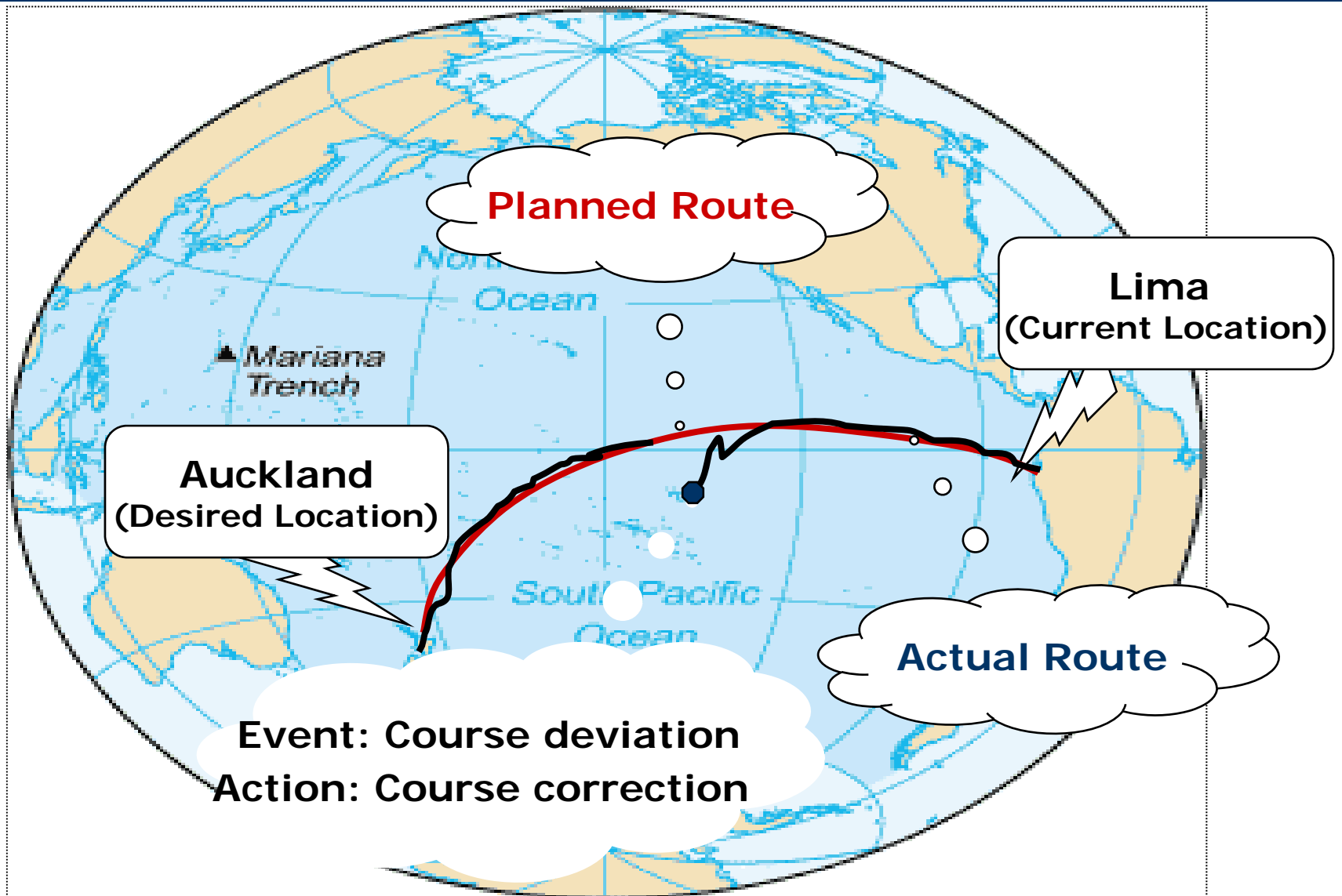
*nach Frühauf,  
Conquest 2001*

	<b>Bisheriger Ansatz</b>	<b>Agiler Ansatz</b>
<b>Ständige Mitwirkg. des Kunden</b>	unwahrscheinlich	kritischer Erfolgsfaktor
<b>Etwas Nützliches wird geliefert</b>	erst nach einiger (längerer) Zeit	mindestens alle sechs Wochen
<b>Das Richtige entwickeln durch</b>	langes Spezifizieren, Vorausdenken	Kern entwickeln, zeigen, verbessern
<b>Nötige Disziplin</b>	formal, wenig	informell, viel
<b>Änderungen</b>	erzeugen Widerstand	werden erwartet und toleriert
<b>Kommunikation</b>	über Dokumente	zwischen Menschen
<b>Vorsorge für Änderungen</b>	durch Versuch der Vorausplanung	durch "flexibel bleiben"

- Viel Planung und wenig Planung ist nicht nur ein Verfahrensunterschied, sondern ein Kulturunterschied!

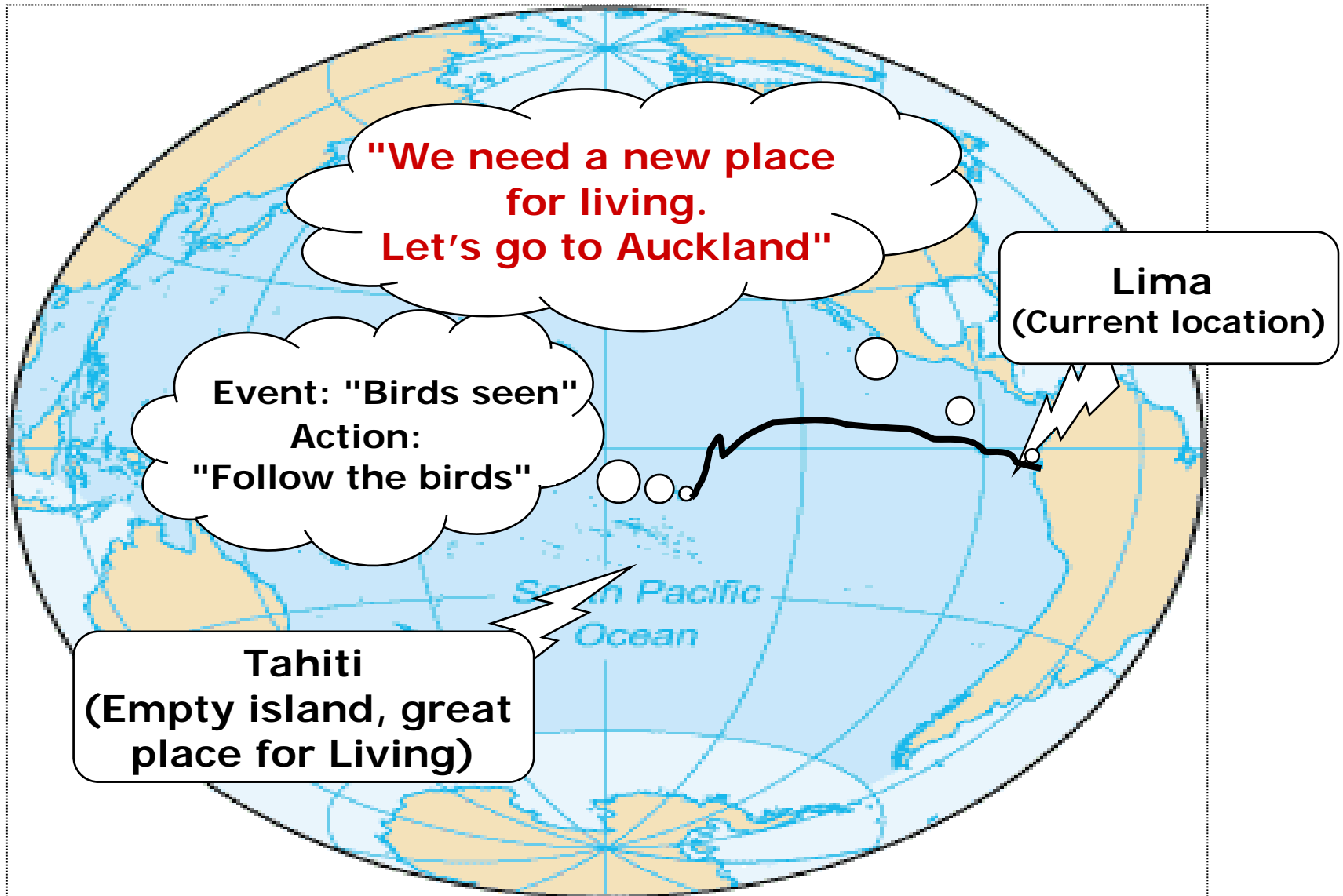
Zwei Stile von Navigation in der Seefahrt [Gladwin 1964]:

- Europäische Navigation
  - basiert auf aktuellem Ort und gewünschtem Zielort
  - ermittelt eine geplante Route
  - verfolgt genau diese Route und korrigiert jegliche Abweichung
- Polynesische Navigation
  - basiert auf gewünschten Ziel-Eigenschaften
  - sucht nach Gelegenheiten zum Erreichen dieser Eigenschaften





# Polynesische Navigation



# Europäische vs. polynesische Navigation: Auckland-Projektplan

- Desired Outcome: Auckland is found
  - Tools: Compass, speed meter, map
  - Methods: Determine planned course, write planned course before departure.
    - Example: Start in Lima. Sail West, keep the compass constantly at 97 degrees.
  - Process:
    - Check direction hourly. If there is a deviation, bring the ship back on the planned course.
  - Schedule: 50 days, if doldrums are encountered, 85 days.
- Desired Outcome: A new place for living is found
  - Tools: Use stars for navigation
  - Methods: Set up a set of event-action rules.
    - When an event occurs, determine the action to be executed in the given context.
  - Process:
    - Start with a certain course.
    - Regularly look out for signs of land (birds, stationary clouds).
    - Change to a different course if that appears promising.
  - Schedule: Probably about 1 or 2 months
    - We will stop when satisfied

- Situated action [Suchman 1990]
  - Selection of action should depend on context:  
type of event, situation and skill of the developer

## Examples of navigation events:

- European Navigation is context independent:
  - Event: "Course deviation in the morning"
    - Action: "Course correction towards planned route"
  - Event: "Course deviation in the evening"
    - Action: "Course correction towards planned route"
- Polynesian Navigation is context dependent:
  - Event: "Birds seen in the morning"
    - Action: "Sail to where the birds come from"
  - Event: "Birds seen in the evening"
    - Action: "Sail to where the birds fly to"

# Wann sind welche Modelle angemessen?

## Strikte, stark planende Modelle:

- Wenn ein wohldefiniertes Resultat in definierter Zeit erreicht werden muss
- Wenn sehr große (insbesondere verteilte) Projektgruppen koordiniert werden müssen
  - Denn dann sind die Pläne und Dokumente zur Koordination schwer verzichtbar

## Agile Modelle:

- Wenn hohe Unsicherheit über die Anforderungen besteht
  - Inhalt, Prioritäten
- Wenn Änderungen von außen häufig sind
  - Anforderungen, Zeitplan, Budget, Qualitätsziele etc.

# Andere Kriterien für die Auswahl des Prozessmodells

Kräfte in Richtung auf stark planende Modelle:

- Parallele Entwicklung von Hardware (z.B. Auto) und SW
- Örtlich verteilte Entwicklung (auch: in mehreren Firmen)

Kräfte in Richtung auf agile Modelle:

- Hat überhaupt keinen Zeitplan (z.B. Hobbyprojekt)
- Arbeit mit unausgereifter und unbeherrschter Technologie
  - und andere Arten stark radikalen Vorgehens

Sonstige:

- Wie viel u. welche Wiederverwendung ist geplant/gewünscht?
- Wie viel Reverse Engineering ist nötig?
- Welche sonstigen großen Risiken gibt es?

# Universal-Prozessmodelle: Die eierlegende Wollmilchsau

Es gibt Prozessmodelle, die den Anspruch erheben, universell geeignet zu sein, z.B.:

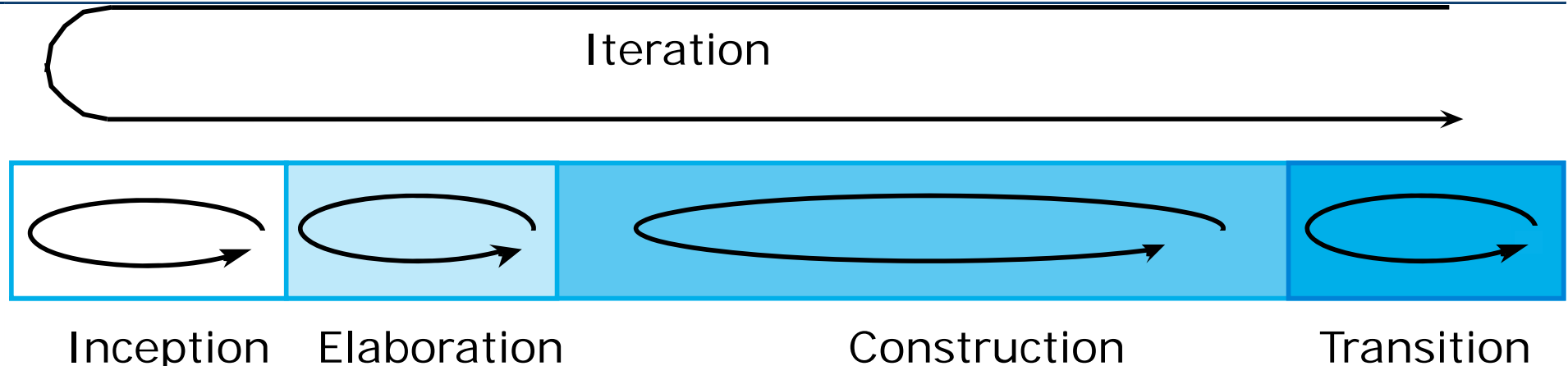
- Rational Unified Process (RUP) + Royce's Unified PM Approach
  - Fortentwicklung von Objectory (Ivar Jacobson) und Spiralmodell
  - RUP wird heute als Produkt vertrieben von IBM/Rational
  - Walker Royce's Methode ist ein Buch: "Software Project Management: A Unified Framework", Addison Wesley 1998.
- V-Modell XT
  - Entwickelt im Auftrag der deutschen Regierung
- Beide Modelle sind stark an Projektbedingungen anpassbar
- Beide Modelle sind sehr umfangreich, nicht leicht zu verstehen
  - Die meisten Leute, die behaupten, sich damit auszukennen, haben nur Grundkenntnisse

# The Rational Unified Process (RUP)

---

- A modern evolutionary (iterative) process model
- Normally described from 3 perspectives
  - 1. A dynamic perspective that shows phases over time
  - 2. A static perspective that shows process activities
  - 3. A practice perspective that suggests good practice
- RUP is the interaction of all three perspectives
- In a concrete project, RUP is tailored as appropriate
  - only parts of the overall RUP definition will be used
  - (For instance, the full RUP defines about 30 roles)

# Perspective 1: RUP phases

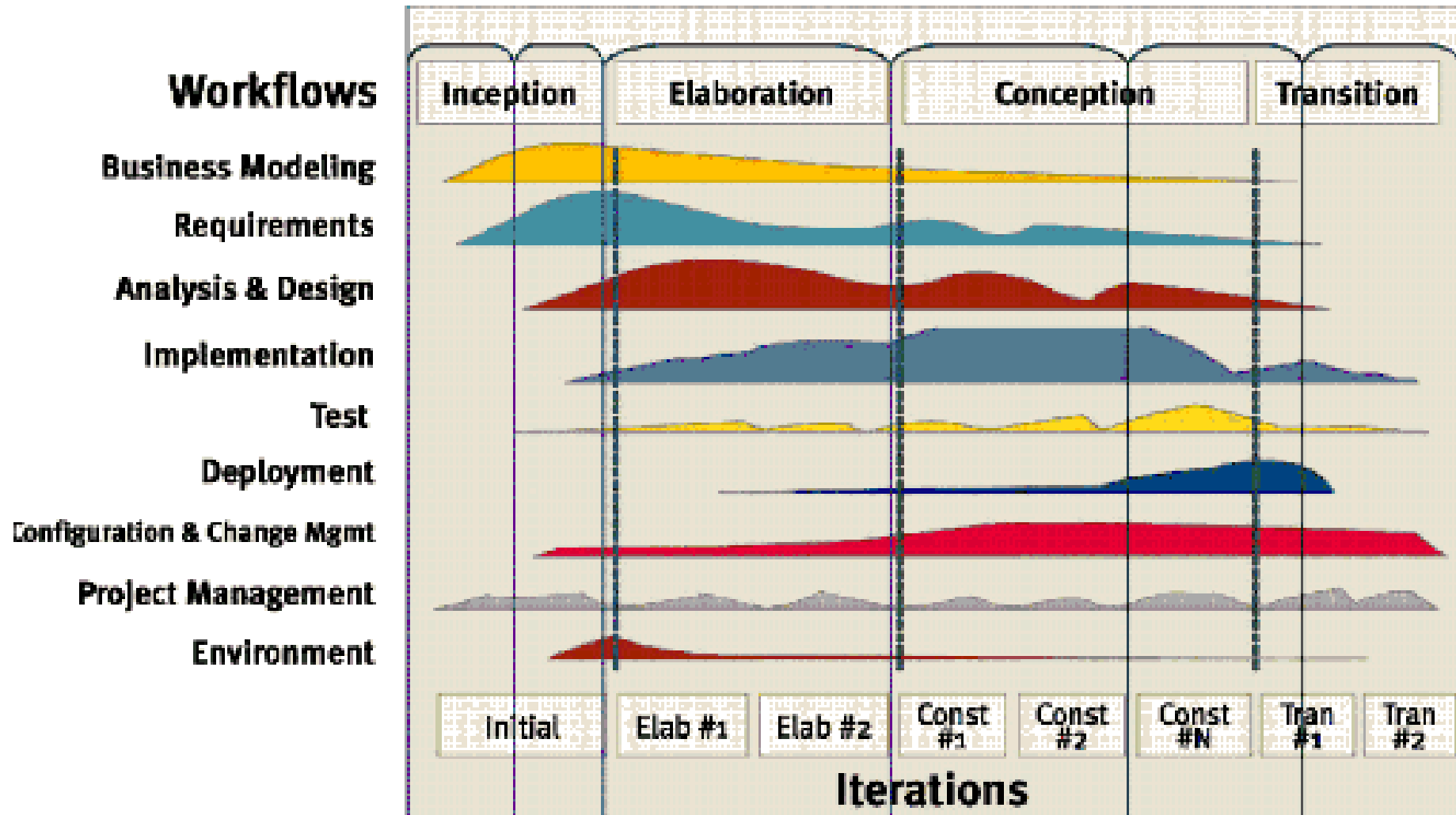


- Inception
  - Establish the business case for the system
- Elaboration
  - Develop an understanding of the problem domain and the system architecture
- Construction
  - System design, programming and testing
- Transition
  - Deploy the system in its operating environment

iterate over as well as within phases



# Intensity of workflows in different phases and iterations (simplistic, but popular)



## Perspective 3: RUP good practice

---

- Architecture-first approach
  - Focus on critical use cases, architecture, and life-cycle plans before starting. Create architecture and plan together
    - a bit of Waterfall Model
- Iterative life-cycle process
  - Each iteration should focus on one risk and move requirements, architecture, and planning in balanced manner
    - a la Spiral Model
- Demonstration-based approach
  - Identify performance issues early, assess intermediate artifacts
    - a la Agile Models
- Component-based development
  - Minimize human-generated lines of code. Use COTS.
- Visual modeling languages
  - to support modeling and documentation
- Round-trip engineering
  - Use automation to couple models and source code
- Change management environment
  - Automate change processes to deal with changes
- Objective quality control
  - Assess progress by automated metrics and quality indicators

## RUP: How much Planning?

- **The project plan is developed iteratively** (along the SW)
  - The Plan is detailed and refined as the stakeholders increase their knowledge in the application and solution domain
- Planning errors are treated like software defects
  - the earlier they are resolved, the less impact they have
- Work breakdown structure is organized around software life cycle activities
  - Level 1: life cycle workflows (management, requirements, design, implementation, assessment, and deployment)
  - Level 2: phases (inception, elaboration, construction, transition)
  - Level 3: the artifacts produced during each phase
- Estimation:
  - Compute the initial estimate with a model, such as COCOMO II
  - Refine it with the project manager, developers, and testers

# RUP: How much Process?

- Smaller Projects (say, 1-10 participants)
  - Require much less management overhead
  - Performance depends on technical skills and tools
  - Focus on the technical artifacts, few milestones, no formal processes
- Large Projects
  - Management skills become primary performance bottleneck
  - Well-defined milestones, focus on change management artifacts

## Principles:

- Stakeholder cohesion
  - Cooperating set of stakeholders: flexible plan, informal agreements
  - Conflict among stakeholders: formal agreements & processes
- Process flexibility
  - Rigor of the process definition impacted by rigor of contract
- Architectural risk
  - Demonstrate feasibility of the architecture before full-scale commitment
- Domain experience
  - Domain expertise shortens the earlier phases of the life cycle

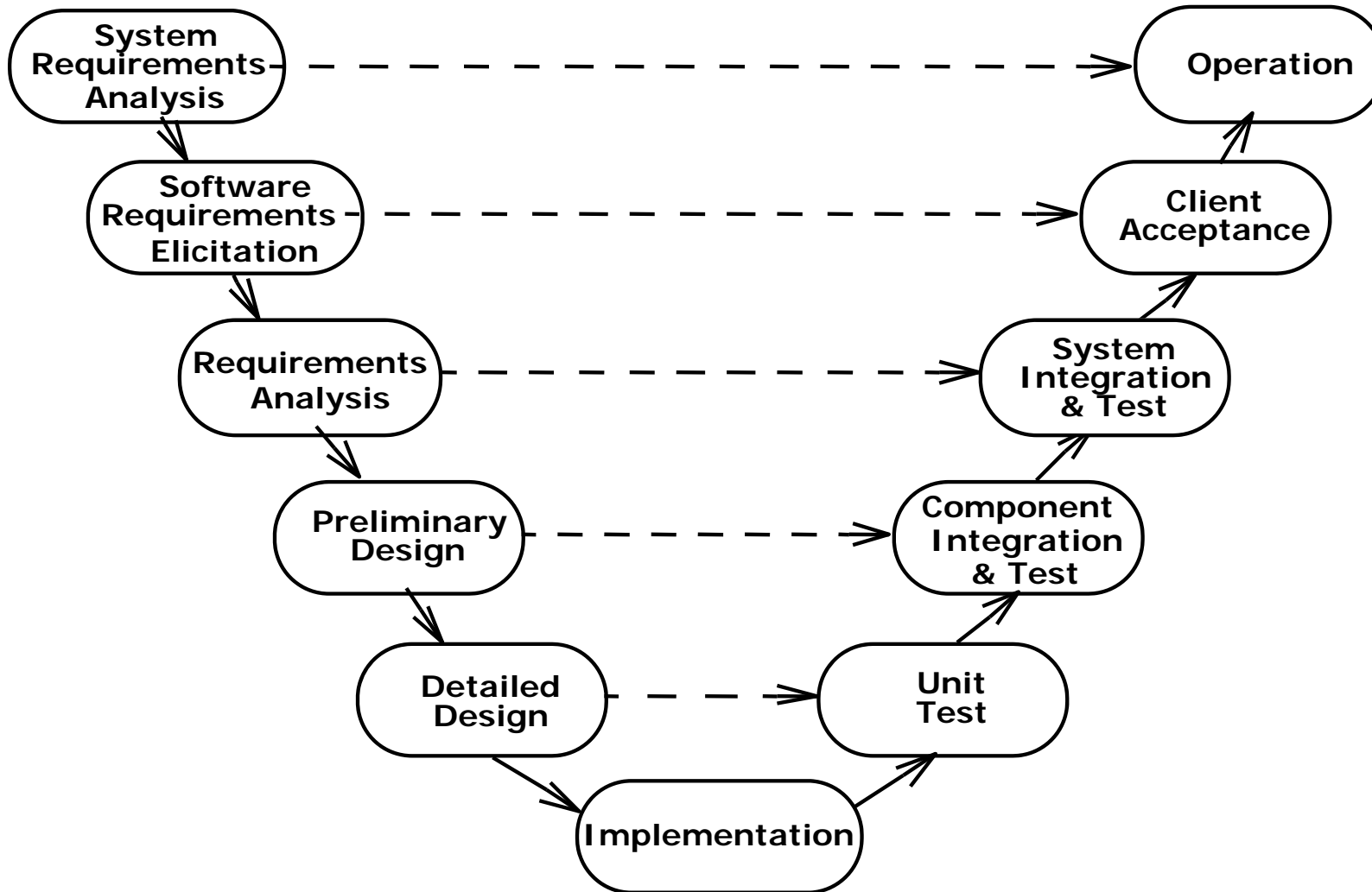
- Entwickelt ca. 1987–1992 im Auftrag des BMVg und BMI als "**Vorgehensmodell** zur Softwareentwicklung"
  - Überarbeitet 1997 als V-Modell 97 (Objektorientierung)
  - Überarbeitet 2005 als V-Modell XT ("extreme tailoring")
- Sehr umfangreiches Modell (V-Modell XT hat 9 Teile):
  - **1: Grundlagen des V-Modells** (Einführung, Konzepte, ...)
  - **2: Eine Tour durch das V-Modell** (Kleines Beispielprojekt)
  - **3: Referenz Tailoring** (Anpassung an mein Projekt)
  - **4: Referenz Rollen** (Wen brauche ich?)
  - **5: Referenz Produkte** (Wie sollen Projektergebnisse aussehen, was enthalten sie?)
  - **6: Referenz Aktivitäten** (Wie erstelle ich Produkte?)
  - **7: Referenz Konventionsabbildungen** (Ich kenne schon VM97/CMMI/... Wo im V-Modell XT finde ich Entsprechendes?)
  - **8: Anhang** (Werkzeuge, Methoden, Glossar...)
  - **9: Vorlagen** (Anwendungsbeschreibung)

## V-Modell Kern

- Das V-Modell wurde vor allem für sehr große Projekte konzipiert. V-Modell XT ist auch an kleinere anpassbar.
- Das V-Modell XT kennt 4 verbindliche (und 16 optionale) so genannte Vorgehensbausteine:
  - PM: Projektmanagement
  - QS: Qualitätssicherung
  - KM: Konfigurationsmanagement
  - PROB: Problem- und Änderungsmanagement
- Diese fassen zusammengehörige Aktivitäten zusammen
- Jedes Produkt (Dokument) durchläuft vier Zustände: geplant, in Bearbeitung, vorgelegt, akzeptiert
  - Qualitätssicherung ist also ein zentraler Bestandteil des V-Modells

# V-Modell: Das andere V

Nicht namensgebende Korrespondenz zw. Entwicklungs- und Validierungsaktivitäten



## Nicht Fisch und nicht Fleisch?

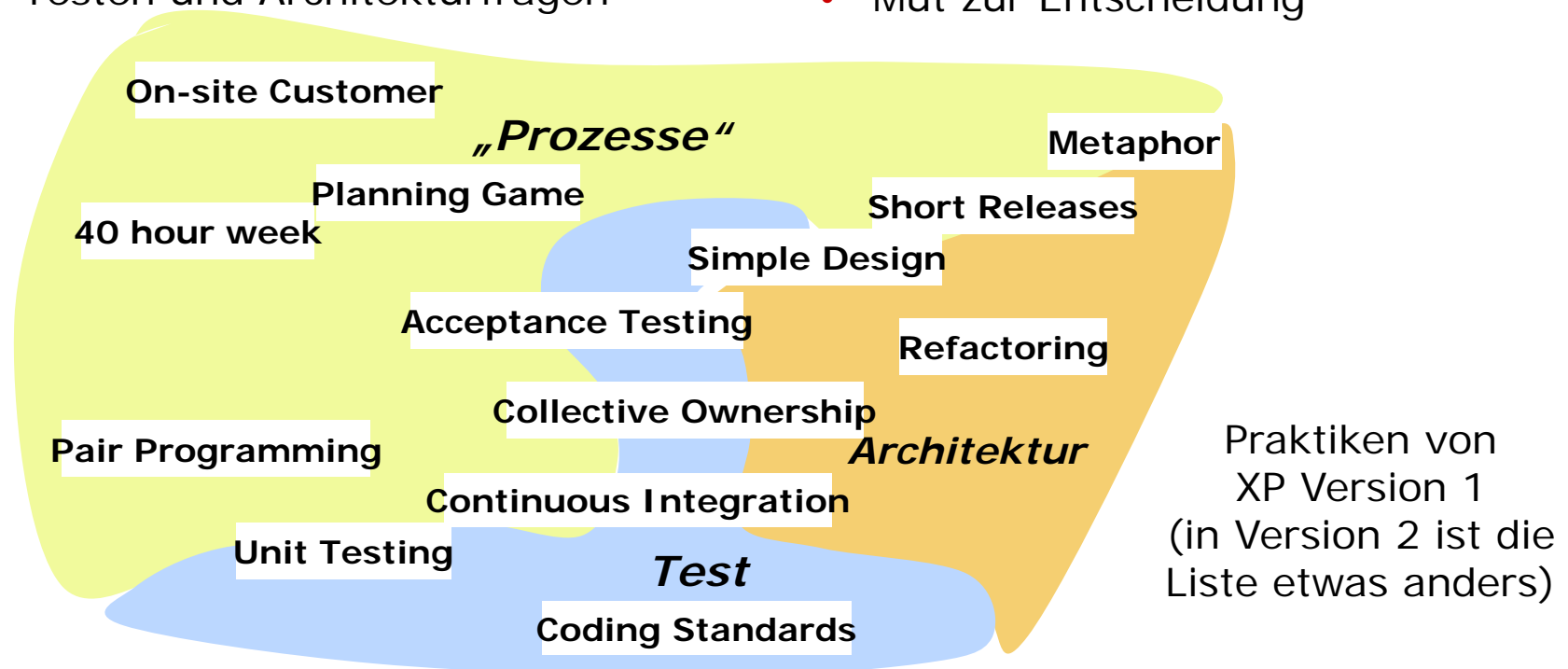
---

- RUP und V-Modell können je nach Bedarf eher wasserfallartigen Charakter oder eher agilen Charakter entfalten
- Dadurch werden sie aber auch schwieriger zu verstehen
  - und es gibt leicht Streit über sie, weil sie gar nicht als so flexibel wahrgenommen werden
- Es fehlt uns also noch ein "reinrassiges" Gegenstück zum Wasserfallmodell
- Es gibt diverse ausgearbeitete Agile Methoden
  - z.B. für eher kleine Projekte:
    - **XP** ("**eXtreme Programming**", Kent Beck)
    - SCRUM [nur Projektmanagement] (Jeff Sutherland, Ken Schwaber)
    - Crystal Clear (Alistair Cockburn)
  - z.B. für mittelgroße Projekte:
    - Scrum of Scrums (verschachteltes SCRUM)



# eXtreme Programming (XP)

- XP ist ein Satz von Praktiken
  - Müssen alle genau eingehalten werden
    - verlangt sehr hohe Disziplin
  - Verstärken sich gegenseitig
  - Zusammenspiel von Prozess, Testen und Architekturfragen
- Werte und Prinzipien
  - Kommunikation u. Kundeneinbindung
  - Feedback und inkrementelle Entwicklung
  - Einfachheit
  - Mut zur Entscheidung



## Was heißt hier extrem?

- "eXtreme Programming" bedeutet nicht, nur zu hacken
  - sondern erfordert ganz im Gegenteil **extreme Disziplin**
- Der Name kommt daher, dass XP bewährte Praktiken aus der SW-Entwicklung ins Extrem führt
  - Beispiel: Codedurchsichten haben sich bewährt, daher verlangt XP ständige Code-Reviews (durch Pair Programming)
- XP ist die vielleicht bekannteste Agile Methode
  - die verbreitetste ist aber inzwischen Scrum
- XP ist eine sehr radikale Agile Methode
- XP ist nicht die einzige, beste oder einfachste Agile Methode
  - Eine beste gibt es auch nicht: Muss zur Situation passen!

- Short Releases (Kurze Freigabezyklen):
  - alle paar Wochen wird eine benutzbare Version an den Kunden ausgeliefert
  - → kompletter Projektfehlschlag ist fast unmöglich
  - → Anforderungsmängel werden früh erkannt
- On-site customer (Kunde vor Ort):
  - entscheidungsbefugter Vertreter des Auftraggebers ist stets beim Projektteam
  - beschreibt Anforderungen, priorisiert, entscheidet, nimmt Ergebnisse ab
  - → Unklarheiten können *umgehend* ausgeräumt werden
- Pair programming (Paarprogrammierung):
  - sämtlicher Code wird von je zwei Programmierern gemeinsam geschrieben
  - → jede Klasse ist mindestens zwei Personen vertraut
  - → mehr Flexibilität bei "wer macht was wann?"
- Unit Testing (Modultests):
  - zu jedem Modul gibt es gründliche automatisierte Tests
  - → erhöhte Bereitschaft, neue Anforderungen zu akzeptieren
    - Leider müssen die Tests auch geändert werden...
  - → weniger Angst vor Änderungen am Code
- u.a.m.

- Wasserfallmodell:
  - <http://c2.com/cgi/wiki?WaterFall>
- Spiralmodell:
  - Barry Boehm: "A Spiral Model of Software Development and Enhancement", IEEE Computer 21(5):61-72, 1988.
- Royce's Method:
  - Walker Royce: "Software project management: A unified framework", Addison-Wesley Longman 1998
  - <http://www.ibm.com/software/awdtools/rmc/index.html>
- V-Modell XT:
  - <http://www.v-modell-xt.de>
- XP:
  - Kent Beck, Cynthia Andres: "Extreme Programming Explained: Embrace Change", 2nd ed., Addison-Wesley 2004
    - 2. Ausgabe: hat andere Liste von Praktiken als die oben vorgestellten
  - [www.extremeprogramming.org](http://www.extremeprogramming.org), [www.xprogramming.com](http://www.xprogramming.com)

# Werden Prozessmodelle eingehalten? Häufigkeit von Entwicklungspraktiken

- Cusumano, MacCormack, Kemerer, Crandall: "Software Development Worldwide: The State of the Practice", IEEE Software Nov. 2003

	India	Japan	US	Europe and other	Total
Number of projects	24	27	31	22	104
Architectural specifications (%)	83.3	70.4	54.8	72.7	69.2
Functional specifications (%)	95.8	92.6	74.2	81.8	85.6
Detailed designs (%)	100.0	85.2	32.3	68.2	69.2
Code generation (%)	62.5	40.7	51.6	54.5	51.9
Design reviews (%)	100.0	100.0	77.4	77.3	88.5
Code reviews (%)	95.8	74.1	71.0	81.8	79.8
Subcycles (%)	79.2	44.4	54.8	86.4	64.4
Beta testing (% > 1)	66.7	66.7	77.4	81.8	73.1
Pair testing (%)	54.2	44.4	35.5	31.8	41.3
Pair programming (%)	58.3	22.2	35.5	27.2	35.3
Daily builds (%)					
At the start	16.7	22.2	35.5	9.1	22.1
In the middle	12.5	25.9	29.0	27.3	24.0
At the end	29.2	37.0	35.5	40.9	35.6
Regression testing on each build (%)	91.7	96.3	71.0	77.3	83.7

Based on 104 projects surveyed

- Ein Prozessmodell beschreibt Aktivitäten, Rollen und Produkte im Softwareprozess
- Es gibt ein weites Spektrum verschiedener Prozessmodelle
  - von stark planungsorientierten (Extremfall: Wasserfallmodell)
    - eignen sich für Projekte, die sehr gut geplant werden können
    - und solche, die sehr gut geplant werden müssen (z.B. sehr große)
  - bis zu stark kommunikationsorientierten (z.B. eXtreme Programming)
    - eignen sich für Projekte mit hohem Änderungsbedarf unterwegs
- Es gibt recht universelle, für viele Fälle anpassbare Modelle
  - Rational Unified Process, V-Modell XT
  - Diese sind aber umfangreich und nicht einfach zu verstehen
- Prozessmodelle sind nicht gut oder schlecht, sondern zum Projekt passend oder unpassend

**Danke!**

# Additional information on RUP: Perspective 2: Static workflows (1)

- Business modelling
  - The business processes are modelled as use cases
- Requirements
  - Actors who interact with the system are identified
  - Use cases are developed to model the system requirements
- Analysis and design
  - A design model is created and documented using architectural models, component models, object models and sequence models
- Implementation
  - The components in the system are implemented and structured into implementation sub-systems
  - Automatic code generation from design models helps accelerate this process
- Test
  - Testing is an iterative process that is carried out in conjunction with implementation
  - System testing follows the completion of the implementation



## Perspective 2: Static workflows (2)

- Deployment
  - The product is installed in users' workplaces
- Configuration and change management
  - Coordinate and protocol changes to the system
- Project management
  - Plans, coordinates, and manages resources
- Environment
  - This workflow is concerned with making appropriate software tools available to the software development team

## RUP: How much Reuse?

- Make-or-buy decisions are treated as risks that should be confronted early in the life cycle (e.g., in the first iterations of the elaboration phase)
  - When components are reused in more than one project, the return on investment can be further increased
- Key principle: Minimize the amount of human-generated source code
  - Reuse commercial components (COTS)
  - Use code generation tools
  - Use high-level visual and programming languages
- Reuse is treated as a return on investment decision which decreases development time
  - Mature components and tools also reduce time to repair defects
  - Note: Immature components and tools increase quality problems drastically which off-sets any economic benefit

# RUP: How much Modeling?

## Modeling artifacts:

- Management Set (planning, monitoring):
  - Ad-hoc notations are used to capture the "contracts" among the stakeholders
  - Specific artifacts: Problem statement, software project management plan, configuration management plan, and status descriptions
- Requirements set
  - Visionary scenarios
  - Prototypes for user interfaces
  - Requirements analysis model
- Design set
  - Software architecture
  - Interface specifications

- Implementation set
  - Source code, components and executables needed for testing the system
- Deployment set
  - All the deliverables negotiated between the project manager and the client
  - In general it contains the executable code, the user manual and the administrator manual

Test artifacts are part of each of the above sets:

- Management set includes test plan and procedures
- Requirements set includes test specifications

## RUP: How much Control?

- Royce's methodology focuses on three management metrics and four quality metrics
- Management metrics:
  - Work. Number of tasks completed (compared to the plan)
  - Cost. Amount of resources consumed (compared to the budget)
  - Team dynamics. Number of participants that leave the project prematurely or that are added
- Quality indicators:
  - Change traffic. How many change requests are issued over time?
  - Breakage. How much source code is reworked per change?
  - Rework. How much effort is needed to implement a change?
  - Mean time between failures. How many defects are discovered per hour of testing?