

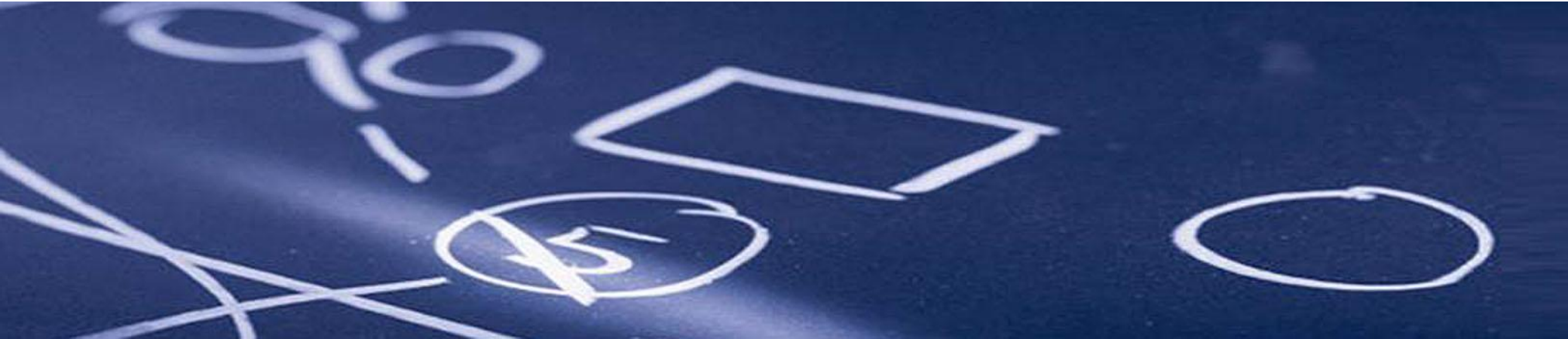


# eXtreme Programming

16.06.2008

---

Florian Conrad, Sven Kirchgäßner, Melanie Kleppsch





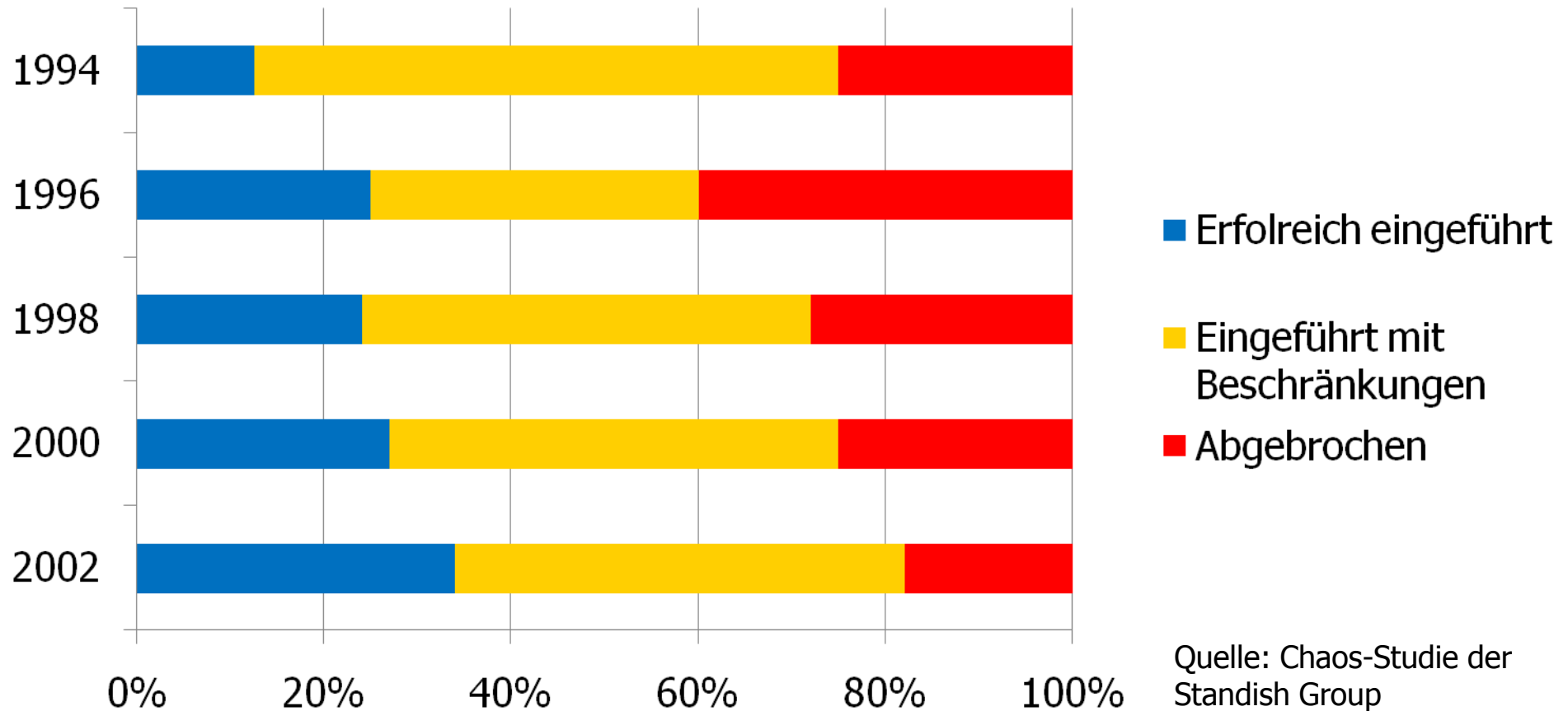
# Agenda

---

- Einführung
- Agile Softwareentwicklung
- Werte & Prinzipien
- Techniken & Rollen
- Prozess
- Nutzen & Kritik
- Fazit & Ausblick

# Einführung

## *Erfolg von IT-Projekten*





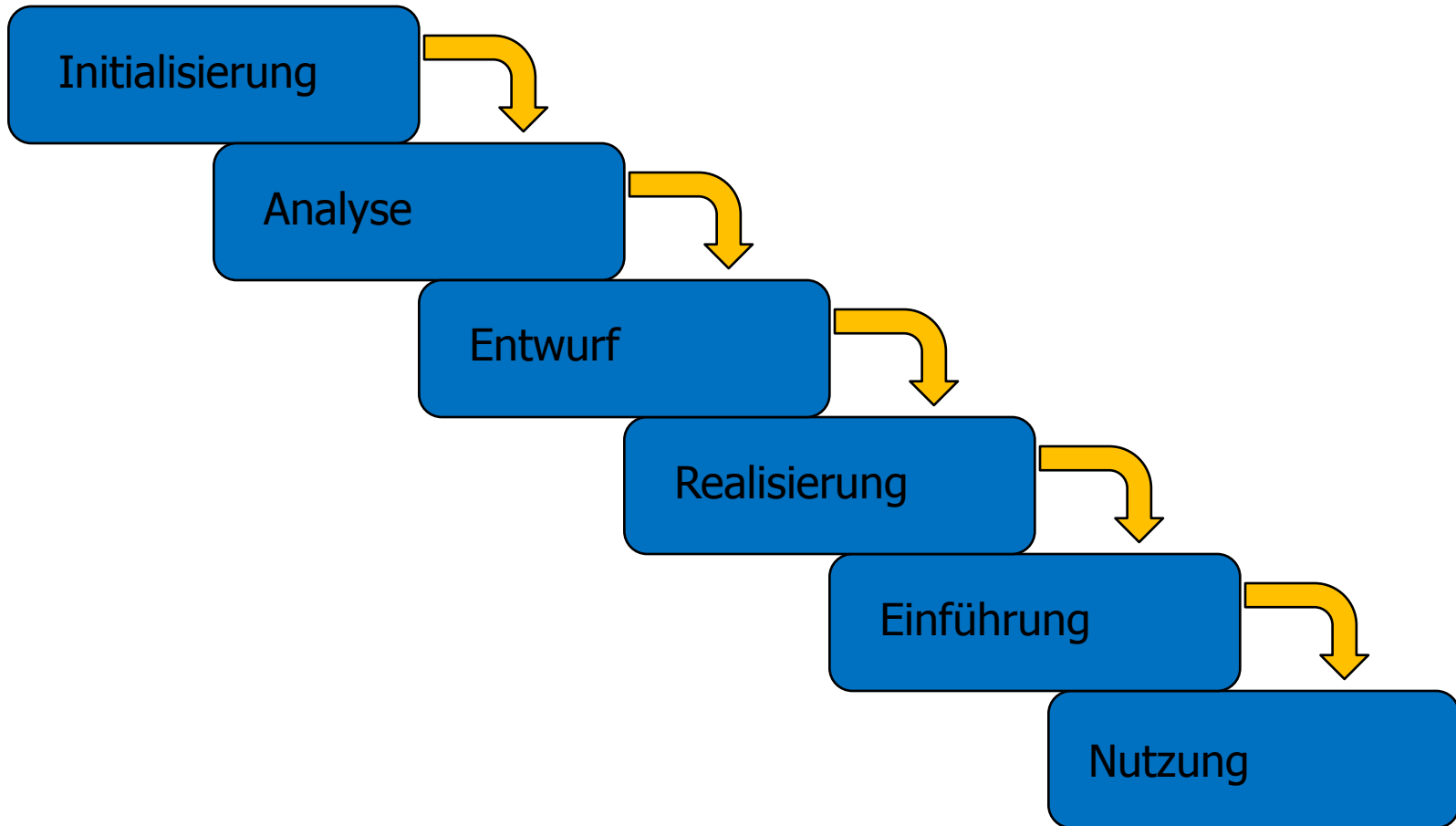
# Einführung

## *Erfolgsfaktoren*

---

- **Einbeziehen der Anwender** (User Involvement)
- **Management-Unterstützung** (Executive Management Support)
- **Erfahrener Projektmanager** (Experienced Project Manager)
- **Klare Geschäftsziele** (Clear Business Objectives)
- **Umfang und Inhalt minimieren** (Minimizing Scope)

# Einführung - *Herkömmliche Vorgehensweise (Wasserfallmodell)*



# Einführung

## *Umsetzung der Erfolgsfaktoren*

<b>Erfolgsfaktoren</b>	<b>Probleme der bisherigen Vorgehensweise</b>
Management-Unterstützung	Fehlende Kommunikation und Transparenz
Einbeziehen der Anwender	Fehlende Kommunikation Missverständnisse
Erfahrener Projektmanager	Unabhängig von der Vorgehensweise
Klare Geschäftsziele	Fehlende Kommunikation Missverständnisse
Umfang und Inhalt minimieren	Komplexität



# Einführung

## *weitere Probleme*

---

- unflexibel bei Änderungen
  - Vernachlässigung des Faktors Mensch
  - spätes Testen
- Softwareentwicklung soll flexibler, transparenter und kundenbezogener werden



# Agile Softwareentwicklung

## *Einführung*

---

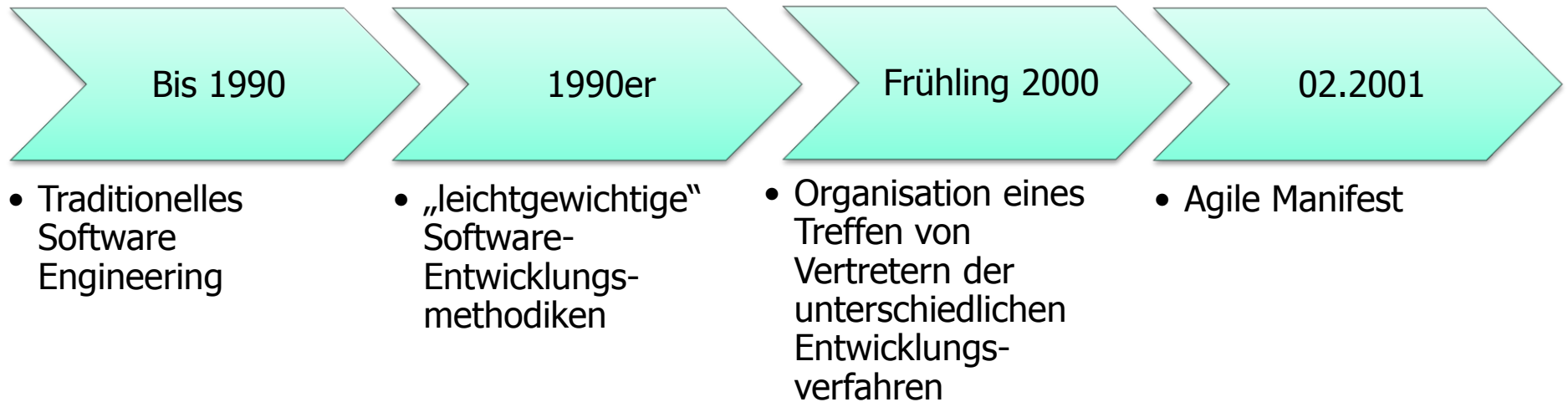
*"Einsatz von Agilität bei der Softwareentwicklung"*

### **Kennzeichen:**

- wenig Bürokratie
- wenige Regeln
- flexible Regeln

# Agile Softwareentwicklung

## *Entstehung*





# Agile Softwareentwicklung

## *Das Agile Manifest*

---

**1. Individuen und Interaktion** *sind wichtiger als Prozesse und Werkzeuge.*

**2. Funktionierende Software** *ist wichtiger als umfassende Dokumentation*

**3. Zusammenarbeit mit dem Kunden** *ist wichtiger als Vertragsverhandlungen*

**4. Sich auf Änderungen einzustellen,** *ist wichtiger als einem Plan zu folgen*



# Agile Softwareentwicklung

## *Methodiken*

---

- Scrum

- DSDM

- EDD

# eXtreme Programming

- ASD

- Pragmatic Programming

- Crystal



# Extreme Programming

## *Überblick*

---

- agile iterative Methodik der Softwareentwicklung
- entwickelt während des C3-Projektes
- Begründer Kent Beck
- flexible Rahmenbedingungen

# Extreme Programming

## *Werte*

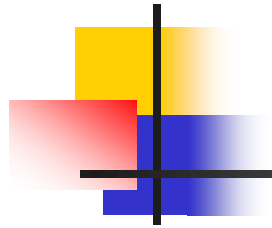
---

1. Kommunikation
2. Einfachheit
3. Mut
4. Feedback
5. Respekt

# Extreme Programming

## *Prinzipien*

➤ Menschlichkeit	➤ Wirtschaftlichkeit
➤ Beidseitiger Vorteil	➤ Selbstgleichheit
➤ Verbesserungen	➤ Vielfältigkeit
➤ Reflexion	➤ Lauf
➤ Gelegenheiten wahrnehmen	➤ Redundanzen vermeiden
➤ Fehlschläge hinnehmen	➤ Qualität
➤ Kleine Schritte	➤ Akzeptierte Verantwortung



# Techniken & Rollen



# Techniken

---

- Kunde vor Ort
  - Aktive Teilnahme am Entwicklungsprozess
  - User-Stories
  - Ansprechpartner
- Planungsspiel
  - Release / Iteration
  - Entwickler schätzen, Kunden priorisieren
  - Transparenz



# Techniken

---

- Metapher
  - Einfache, bildliche Vorstellungen
  - Muss von allen Projektbeteiligten geteilt werden
  - Sind handlungsleitend
- Kurze Releasezyklen
  - Für Kunden erkennbarer Fortschritt
  - Feedback
  - Risikominimierung



# Techniken

---

- Testen

- Komponententests

- Schneller Wechsel testen/programmieren
    - Qualitätssteigerung

- Akzeptanztests

- Aus Sicht des Anwenders
    - Automatisiert & funktionell

- Erfolgt in Paaren



# Techniken

---

- Einfaches Design
  - Aufs Nötigste beschränkt (Quick Design Sessions)
  - Leichte & schnelle Änderung
  - Keine Vorratsprogrammierung
- Refactoring
  - Umstrukturierung des Systems
  - Mit Anwenderanforderungen in Verbindung

- Programmieren in Paaren
  - 2 Entwickler, 1 Rechner, 1 Tastatur, 1 Maus
  - Gleichberechtigt
  - Vier-Augen-Prinzip
  - „Pilot und Co-Pilot“
  - Keine Spezialisierung auf ein Gebiet
  - Ständiges Wechseln
  - Effektiver



# Techniken

---

- **Gemeinsame Verantwortlichkeit**
  - Quelltexte & Dokumente gehören dem gesamten Team
  - Jeder kann alles ändern (erwünscht)
- **Fortlaufende Integration**
  - Geänderter Quelltext für alle „sichtbar“
  - Integrationsrechner (5-10 Min.)
  - Ermunterung zu kleinen Refactorings



# Techniken

---

- Programmierstandards
  - Äußere Form des Quelltextes
  - Einheitliche Quelltexte, schnelle Zurechtfindung
- 40-Stunden-Woche
  - Bei Überstunden leidet Qualität
  - Zusammenhang mit anderen Techniken
  - Arbeitszeit extrem verdichtet



# Rollen

---

## ■ Kunde

- Bestimmt, was zu programmieren ist
- Definiert Reihenfolge der Entwicklung
- Auftraggeber
  - Stellt finanzielle Mittel
  - Definiert Projektziele (geschäftspolitische Überlegungen)
- Anwender
  - Anwendungsfachliches Wissen
  - Story-Cards



# Rollen

---

- Programmierer

- Kodieren, entwerfen, dokumentieren & testen
- Schätzen Story-Cards
- Kommunikations- & Konfliktfähigkeit
- Gute soziale Beziehungsfähigkeiten

- Tester

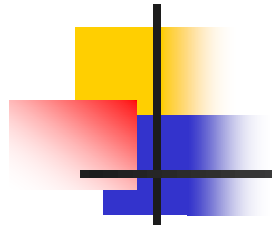
- Formulierung Akzeptanztests
- Meist gleichzeitig Programmierer



# Rollen

---

- Verfolger
  - „Gewissen des Teams“
  - Beobachtet Entwicklungsprozess
  - Sammelt Daten zur Projektsteuerung
- Gesamte Rollenverteilung ist flexibel!
- Auch noch zusätzliche Rollen möglich
  
- Jedes Teammitglied soll das Beste leisten



# **Prozess**

## **Nutzen & Kritik**

## **Fazit & Ausblick**



# Idealer Lebenszyklus eines XP-Projektes *nach Kent Beck*

---

Erforschung

Planung

Iteration bis zur ersten Version

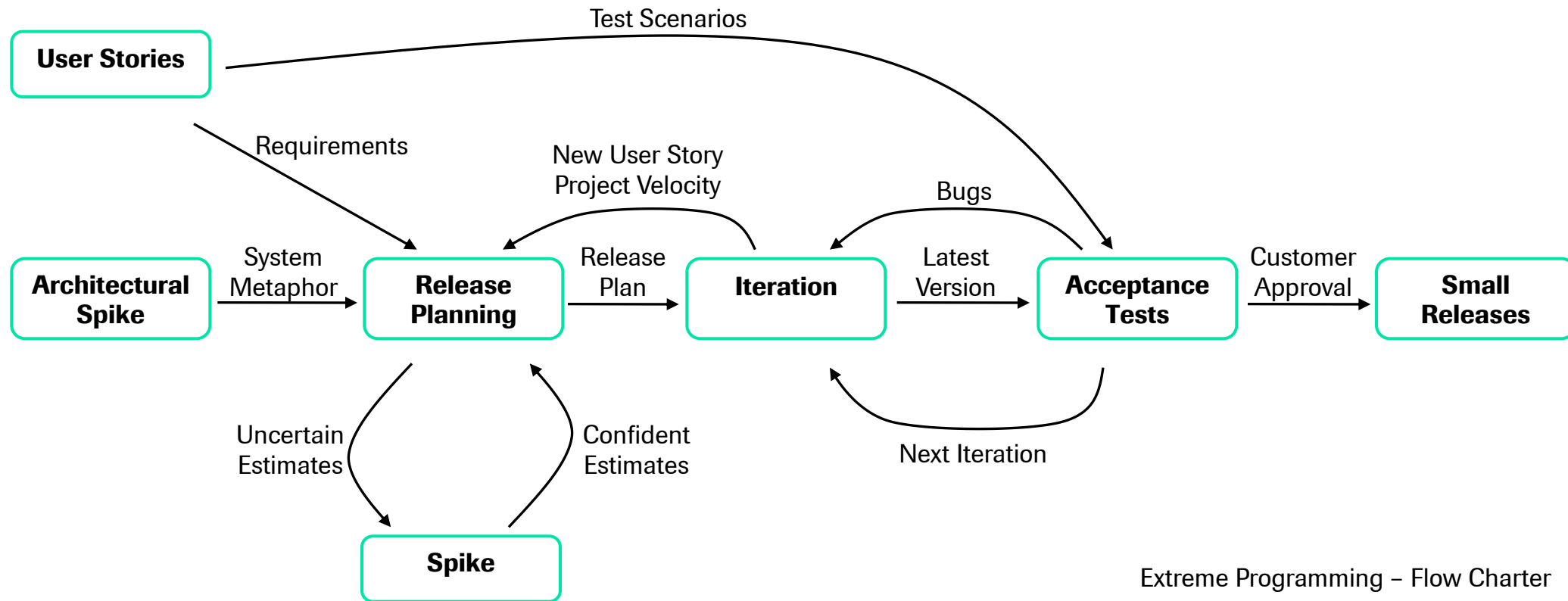
In Produktion gehen

Wartung

Tod

# eXtreme Programming Projekt

## Flow Chart



Extreme Programming – Flow Charter

Quelle: Eigenerstellung in Anlehnung an Don Wells



# eXtreme Programming in der Praxis

---

## **Forrester Research (2005):**

- 14% nordamerikanische und europäische Unternehmen
- weitere 19% in Planung
- Encyclopaedia Britannica, Symantec, Blizzard Entertainment
- Sabre Airline Solution: 42% Produktivitätssteigerung durch XP



# Nutzen durch eXtreme Programming

---

## **Programmierer:**

- Projektüberblick
- Entlastung des Einzelnen
- Spaß an der Arbeit

## **Kunde:**

- Kurzer Entwicklungszyklus
- Eingriff jederzeit möglich
- Anpassung jederzeit möglich
- Funktionsfähiges Programm nach kurzer Zeit
- Aktueller Informationsstand



# Nutzen durch eXtreme Programming

---

## **Projekt:**

- Minimierung von Risiken
- Schutz vor Krankheiten Einzelner
- Minimierung von Angst
- Zuverlässigkeit der Aufwandsschätzungen



# Kritik an eXtreme Programming

---

- „idealer Kunde“ → räumliche Trennung?
- „idealer Programmierer“ → Kommunikationsfähigkeit?
- Produktivität von Pair-Programming → doppelter Personalaufwand?
- Gemeinsamer Code → Konkurrenz zw. Teammitgliedern?
- Bezahlung der Programmierer → Teamgedanke?
- Wartungsproblematik → ohne Dokumentation?



# Fazit

---

- Vorgehensweise der Softwareentwicklung
  - “Effiziente Entwicklung qualitativ hochwertiger Software unter Einbehaltung von Zeit- und Kostenbudgets”
  - Paradigmenwechsel
  - „Es wird in der Praxis für die Praxis angepasst“
- Gelungenes Vorgehensmodell, das aus der Praxis entstanden ist, auf „best practices“ basiert und dem aktuellen Stand der Zeit entspricht!



# Ausblick

---

- Je mehr Projekte gemäß XP umgesetzt werden, desto mehr Erfahrungen fließen in die Entwicklung von XP.
- Aktuelle Lehrmeinung in Europa:
  - „systemematische und teilweise bürokratische Vorgehensweise mit großer Dokumentationsleistung“
- Praxisrelevanz wird XP am Leben erhalten
- „Extreme Modelling“
  - noch schnellere und kürzere Zyklen
  - Code- und Testgeneratoren