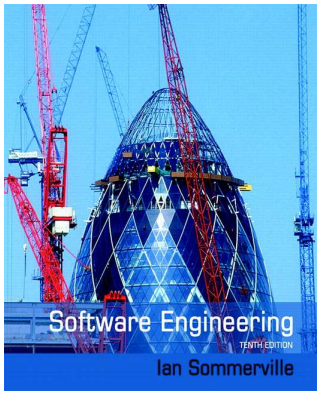
# 706.088 INFORMATIK 1

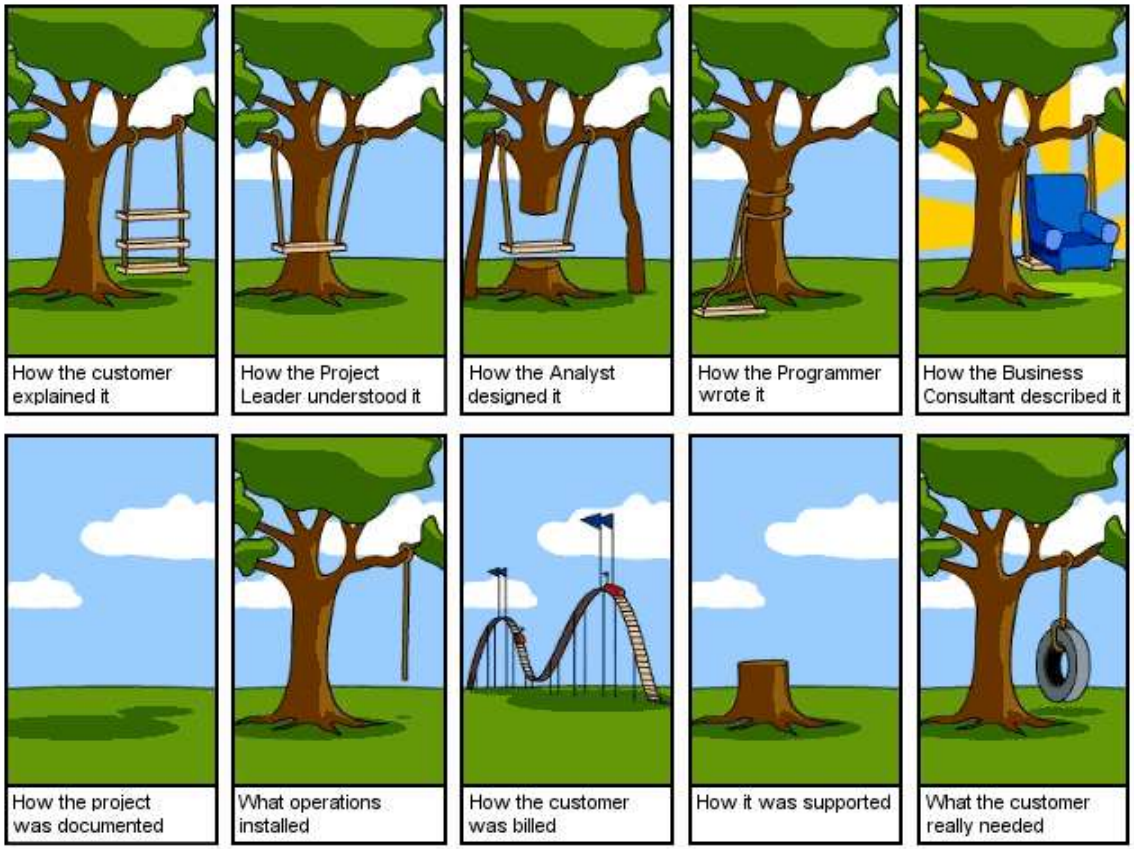
Softwareentwicklungsprozesse

› DI Johanna Pirker

- Buchempfehlung
  - **Software Engineering 10**
  - **Ian Sommerville**



# MOTIVATION


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

# SOFTWARE-ENTWICKLUNG

# DEFINITION SOFTWAREENTWICKLUNG

› **Software** = Computerprogramm + Dokumentation eventuell entwickelt für einen speziellen Markt

› **Gute Software** = Funktionalität, Performance, Benutzbarkeit und Wartbarkeit

# SOFTWARE-ENTWICKLUNG VS. INFORMATIK

› **Softwareentwicklung** = Disziplin, welche alle Aspekte der Softwareproduktion beinhaltet

› **Informatik** = Beschäftigt sich mit der Theorie und Grundlagen, Softwareentwicklung beschäftigt sich mit den Arten der Entwicklung und Lieferung von benutzbarer Software

# SYSTEMENTWICKLUNG

› **System Engineering** = beinhaltet zusätzlich auch alle computer-basierten Systeme (Hardware, Sofware, Prozesse)

# ATTRIBUTE GUTER SOFTWARE

- Wartbarkeit
- Zuverlässigkeit
- Sicherheit
- Effizienz und Performance
- Angemessenheit, Benutzerfreundlichkeit, Kompatibilität, Verständlich

# ARTEN VON ANWENDUNGEN

- Eigenständige Anwendungen
- Web-Anwendungen
- Embedded Control Systems
- Batch Processing
- Entertainment
- Modellierungen und Simulationen
- Datenerfassung und -sammlung
- ...

# SOFTWARE-ENTWICKLUNGS-PROZESS

# PROZESS (1/2)

› (1) **Spezifikation**: Funktionalitäten und Auflagen identifizieren und definieren

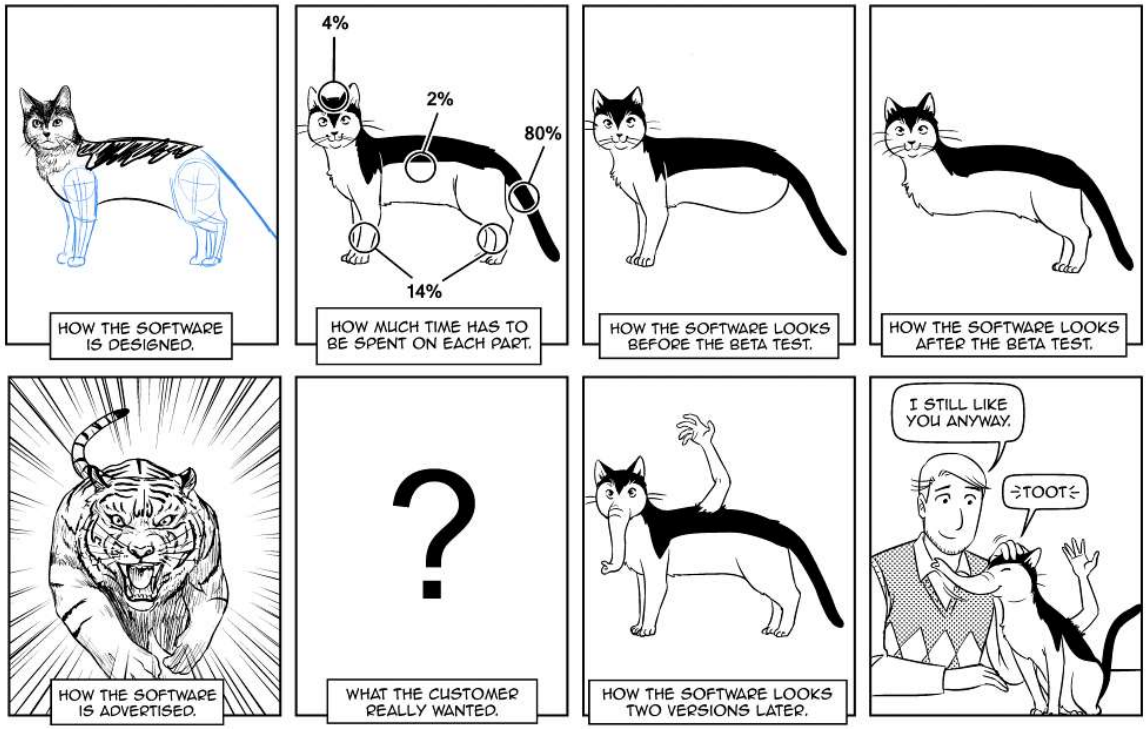
› (2) **Design und Implementierung**: Software, welche den Spezifikationen entspricht wird entwickelt

# PROZESS (2/2)

› (3) **Verifikation & Validierung**: Die Software wird getestet, ob die den Spezifikationen der Stakeholder (e.g. Kunden) entspricht

› (4) **Weiterentwicklung**: Anpassung an sich ändernde Kundenwünsche

# Richard's guide to software development




***Sandra and Woo*** by Oliver Knörzer (writer) and Powree (artist) – www.sandraandwoo.com

# VORGEHENSMODELLE

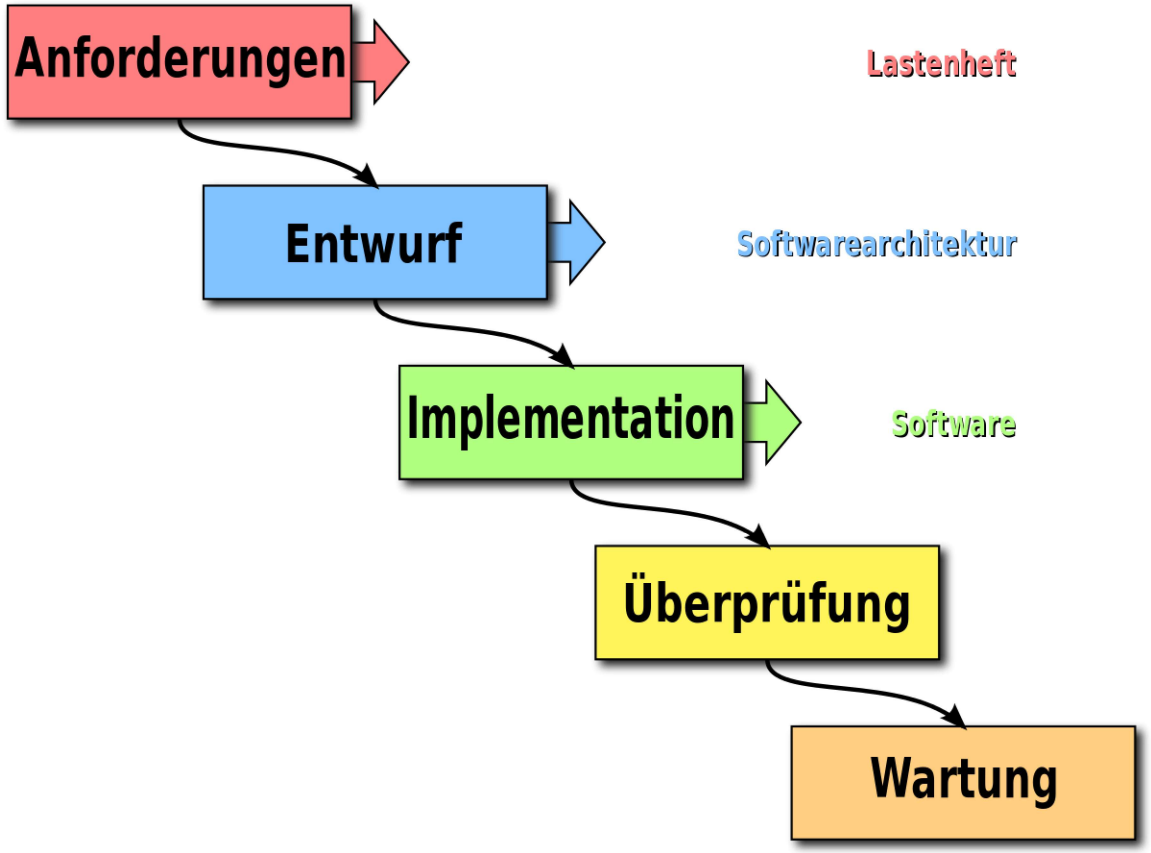
# VORGEHENSMODELLE

- abstrakte Repräsentation vom Softwareentwicklungsprozess
- für große Projekte oft Mischformen

# VORGEHENSMODELLE

- Wasserfall Modell (Trennung der Prozesse)
- Evolutionäre Entwicklung / Iterative Entwicklung
- Komponentenbasierte Entwicklung mit Wiederverwertung
- Agile Softwareentwicklung
  - Extreme Programming
  - SCRUM
- ...

# WASSERFALLMODELL




Von Paul Hoadley, Paul Smith and Shmuel Csaba Otto Traian, CC BY-SA 3.0, Link

# WASSERFALLMODELL

- Lineares Vorgehensmodell
- Klar definierte Start- und Endpunkte der Phasen
- Klar definierte Ergebnisse
- Meilensteinsitzungen
- Lastenheft (Anforderungsspezifikation)
- Pflichtenheft (Implementierungskonzept, Feature Specification)

# WASSERFALLMODELL

- Typische Phasen:
  - (1) Planung, (Resultat: Lastenheft)
  - (2) Systemdesign- und spezifikation (Resultat: Softwarearchitektur)
  - (3) Implementierung und Modultests (Resultat: Software)
  - (4) Integrationstest
  - (5) Auslieferung und Wartung

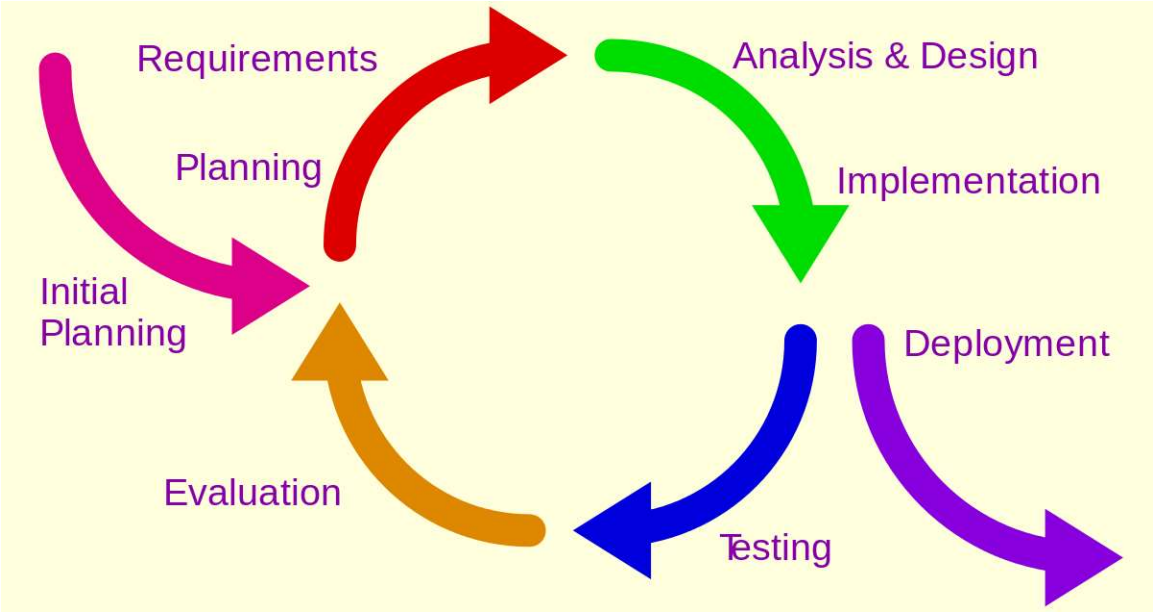
# WASSERFALLMODELL

- Vorteile:
  - klar getrennte Phasen
  - definierte Zeitpunkte für Prozessschritte
- Nachteile:
  - frühe Festlegung der gesamten Funktionalität notwendig
  - spätere Änderungen kostspielig

# ITERATIVE UND INKREMENTELLE ENTWICKLUNG

- Iterativ: Schrittweise Annäherung an die Lösung (Verändern)
- Inkrementell: Kleine Stufen der Zunahme (Hinzufügen)

# ITERATIVE UND INKREMENTELLE ENTWICKLUNG




Von Aflafla1 - Iterative development model V2.jpg , User:Westerhoff, CC0, Link

# ITERATIVE UND INKREMENTELLE ENTWICKLUNG

- Anwendung bei kleinen bis mittleren Projekten
- Vorteile:
  - Schrittweise Spezifikation
  - Kosten und Aufwand bei Anforderungsänderung geringer
  - Zunehmendes Verständnis des Kunden
  - Schnelleres Deployment von funktionstüchtiger Software an den Kunden (mit Grundfunktionalitäten)

# ITERATIVE UND INKREMENTELLE ENTWICKLUNG

- Nachteile:
  - Projektfortschritt schwer messbar (Manager benötigen Deliverables um Fortschritt zu messen)
  - Dokumentation muss ständig aktualisiert werden
  - Ohne Refactoring zwischen den Iterationen wird die Systemstruktur kompliziert und zukünftige Änderungen werden teurer

# LIVE-BEISPIEL

- Jetzt zu 2. oder zu 3. "Schere Stein Papier" spielen
  - Wie gut funktioniert es? Macht es Spass?
  - Eine Regel ändern oder eine neue Regel hinzufügen
  - noch einmal spielen
  - Wie gut funktioniert es? Macht es Spass?

# AGILE SOFTWARE-ENTWICKLUNG

# AGILE SOFTWAREENTWICKLUNG

- Agilität in der Softwareentwicklung
  - gesamten Softwareentwicklungsprozess (Extreme Progrogramming) oder Teilbereiche
- Ziel: Flexibilität

# AGILE MANIFEST (AGILE MANIFESTO)

Agile Manifesto DE

# AGILE MANIFEST (AGILE MANIFESTO)

› Individuen und Interaktionen mehr als Prozesse und Werkzeuge

› Funktionierende Software mehr als umfassende Dokumentation

› Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung

› Reagieren auf Veränderung mehr als das Befolgen eines Plans

# AGILE MANIFEST (12 PRINZIPIEN)

› Kundenzufriedenheit

› Anforderungsänderungen auch spät willkommen

› regelmäßige Lieferung funktionierender Software

› Tägliche Zusammenarbeit von Fachexperten und Entwicklern während des Projektes

› Umfeld und Unterstützung für motivierte Individuen

› Face-to-face Gespräche als primäre Informationsübertragung

» 12 Prinzipien

# AGILE MANIFEST (12 PRINZIPIEN)

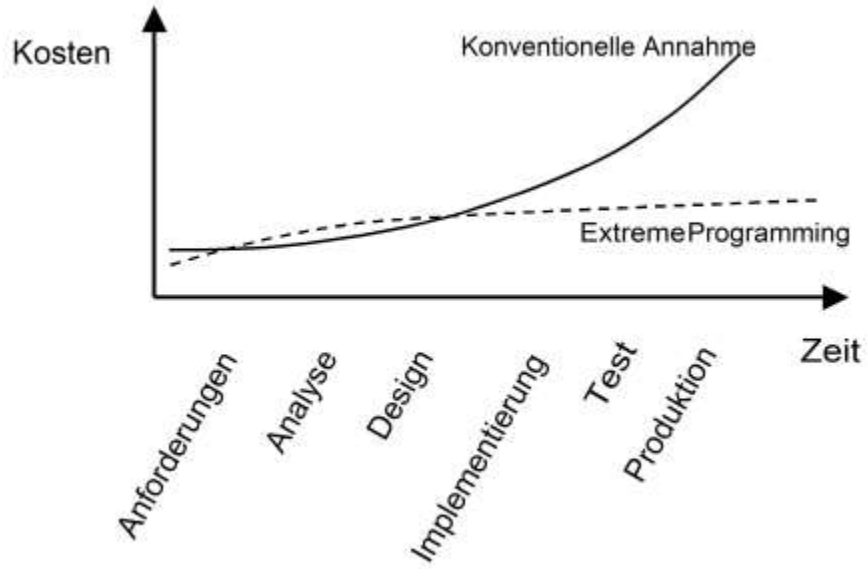
- Fortschrittmaß ist funktionierende Software
- Gleichmäßiges Tempo auf unbegrenzte Zeit
- Technische Exzellenz und gutes Design
- Einfachheit im Fordergrund
- Selbstorganisiertes Team
- Regelmäßige Reflektionen im Team
  - 12 Prinzipien

# EXTREME PROGRAMMING

- Agile Methode von Beck (2000)
  - Schrittweise Planung, kurze Iterationen
  - Simples Design
  - Refactoring (ständige Codeverbesserung)
  - Test-First Entwicklung (Unit-Tests zuerst geschrieben)
  - Pair Programming (2 Programmierer pro Computer)
  - “Kunde” bei Entwicklungsteam
  - Mini-Releases, Ständige Integration (lauffähiges Gesamtsystem)

# EXTREME PROGRAMMING

› Änderungskostenkurve in Abhängigkeit vom Zeitpunkt der Änderung



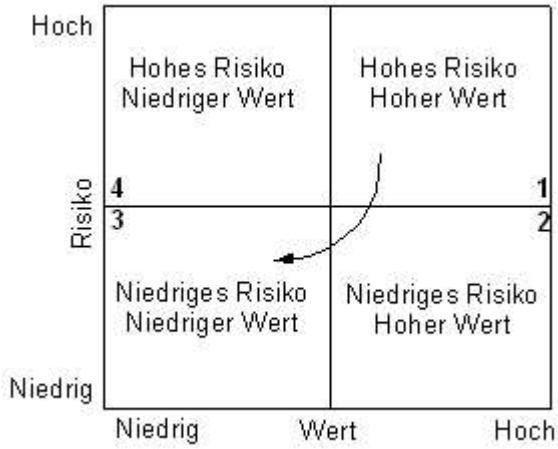

Von German Wikipedia Benutzer GuidoZockoll, CC BY-SA 3.0, Link

# EXTREME PROGRAMMING

- Rollen
  - Product Owner - Verantwortung, setzt Prioritäten
  - Kunde - Auftraggeber
  - Entwickler
  - (Projektmanager) - Teamführung
  - Benutzer - Nutzer des Produktes

# EXTREME PROGRAMMING

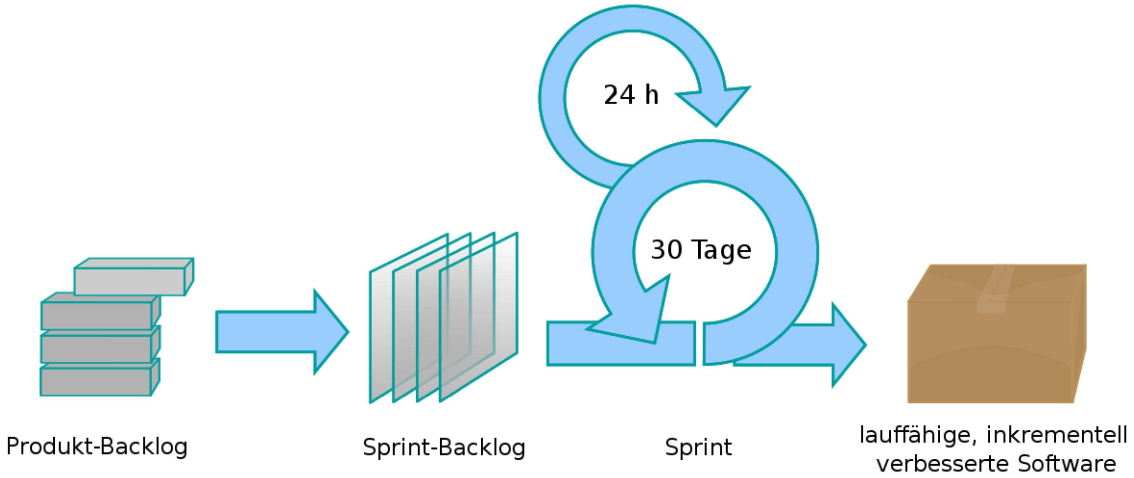
› User Stories




Von Michael Hüttermann - Eigenes Werk, Gemeinfrei, Link

# SCRUM

› Agiles Projektmanagement




Von Scrum_process.svg: Lakeworks derivative work: Sebastian Wallroth (talk) - Scrum_process.svg, CC BY-SA 3.0, Link

# SCRUM

- Sprint (2-4 Wochen)
- Arbeitsabschnitt um bestimmte Funktionalität zu entwickeln
- Sprint Planning (max. 2 Stunden)
  - Was wird entwickelt?
  - Wie wird es erledigt?

# SCRUM

› Product-Backlog: Liste von Aufgaben für das Projekt als Planungsstartpunkt

› Sprint-Backlog: Detaillierter Plan für nächsten Sprint

› Selection Phase: Features/Funktionalitäten für Sprint zusammen mit Kunde gewählt

› Development: mit “Daily Scrum Meetings” / Scrum Master

› Review (Code Review): Ende von Sprint

# SCRUM

- Vorteile:
  - Produkt ist unterteilt in verschiedene überschaubare und verständliche Teile
  - Hohe Fexibilität
  - unrealisierbare Anforderungen schnell identifiziert
  - Transparenz und Vertrauen durch regelmäßige Kommunikation
  - Vertrauen zwischen Kunde und Entwickler,
  - Kurze Kommunikationswege

# SCRUM

- Nachteile:
  - Gesamtüberblick schwierig
  - Hoher Kommunikationsaufwand
  - Zeitverlust bei schlechten Sprintplanungen
  - Schwieriger Umsetzbar bei Großprojekten (Höherer Koordinationsaufwand erfordert)

# PROJECT SUCCESS

**MODERN RESOLUTION FOR ALL PROJECTS**

| | 2011 | 2012 | 2013 | 2014 | 2015 |
|---|---|---|---|---|---|
| SUCCESSFUL | 29% | 27% | 31% | 28% | 29% |
| CHALLENGED | 49% | 56% | 50% | 55% | 52% |
| FAILED | 22% | 17% | 19% | 17% | 19% |

*The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.*

Source: Standish Group 2015: Chaos-Studie

# PROJECT SUCCESS

**CHAOS RESOLUTION BY PROJECT SIZE**

| | SUCCESSFUL | CHALLENGED | FAILED |
|---|---|---|---|
| Grand | 2% | 7% | 17% |
| Large | 6% | 17% | 24% |
| Medium | 9% | 26% | 31% |
| Moderate | 21% | 32% | 17% |
| Small | 62% | 16% | 11% |
| **TOTAL** | **100%** | **100%** | **100%** |

*The resolution of all software projects by size from FY2011–2015 within the new CHAOS database.*

Source: Standish Group 2015: Chaos-Studie

# PROJECT SUCCESS

**CHAOS RESOLUTION BY AGILE VERSUS WATERFALL**

| SIZE | METHOD | SUCCESSFUL | CHALLENGED | FAILED |
|---|---|---|---|---|
| All Size Projects | Agile | 39% | 52% | 9% |
| | Waterfall | 11% | 60% | 29% |
| Large Size Projects | Agile | 18% | 59% | 23% |
| | Waterfall | 3% | 55% | 42% |
| Medium Size Projects | Agile | 27% | 62% | 11% |
| | Waterfall | 7% | 68% | 25% |
| Small Size Projects | Agile | 58% | 38% | 4% |
| | Waterfall | 44% | 45% | 11% |

The resolution of all software projects from FY2011–2015 within the new CHAOS database, segmented by the agile process and waterfall method. The total number of software projects is over 10,000

Source: Standish Group 2015: Chaos-Studie

# LITERATUR


› Extreme Programming Explained. (Kent Beck, Addison-Wesley, 2000.)

› Running an Agile Software Development Project. (M. Holcombe, John Wiley and Sons, 2008.)

› Get Ready for Agile Methods, With Care. (B. Boehm, IEEE Computer, January 2002.) http://doi.ieeecomputersociety.org/10.1109/2.976920.

# (1) SPEZIFIKATION

# SOFTWAREANFORDERUNGEN

- Lastenheft oder Product Backlog
  - Customer Requirement, Anforderungen beschreiben)
- Pflichtenheft -> Design
  - Development Requirement, wie lösen

# SOFTWAREANFORDERUNGEN

- Funktionale Anforderungen
  - „Das Produkt soll den Saldo des Kontos am ersten des Monats berechnen.“
- Nichtfunktionale Anforderungen
  - „Das Produkt soll dem Anwender innerhalb von einer Sekunde antworten.“
  - Benutzbarkeit, Zuverlässigkeit, Effizienz, Änderbarkeit, Wartbarkeit, ..

# ANFORDERUNGSSPEZIFIKATION LAUT IEEE

- korrekt
- unzweideutig (eindeutig)
- vollständig
- widerspruchsfrei
- bewertet nach Wichtigkeit und/oder Stabilität
- verifizierbar
- modifizierbar
- verfolgbar (traceable)

# (2) DESIGN UND IMPLEMENTIERUNG

# SYSTEMDESIGN

› Softwarearchitektur (Komponentendarstellung, Systemdesign)

› Objektorientierte Analyse und Design (Anforderungsanalyse und Systementwurf, UML)

› Datenmodellierung, Entity-Relationship-Modell (e.g. Datenbankschema)

# (3) VERIFIKATION & VALIDIERUNG

# VERIFIKATION & VALIDIERUNG

- **Validierung**: “Entwickeln wir das richtige Produkt?”
  - wird den Kundenerwartungen entsprochen?
- **Verifikation**: “Entwickeln wir das Produkt korrekt?”
  - entspricht die Software den Anforderungen?

# VERIFIKATION & VALIDIERUNG

- Software Inspektion
  - Review von Spezifikation, Design und Code
  - Programmcode wird überprüft (Statischer Ansatz)
- Software Test
  - Programm „läuft“ mit Testdaten (Dynamischer Ansatz)
  - Ausgaben und Laufverhalten wird überprüft

# (4)
# WEITERENTWICKLUNG

# SOFTWAREWEITERENTWICKLUNG

› Neue und veränderte Anforderungen machen Weiterentwicklung notwendig

# SOFTWAREWEITERENTWICKLUNG

- Fehlerbehebung
  - Sourcecode-Fehler (geringster Aufwand)
  - Desing-Fehler (mittlerer Aufwand)
  - Requirement-Fehler (größter Aufwand)
- Softwareanpassung an
  - diverse Plattformen
  - neue Systemumgebung
- Funktionsanpassungen u. Erweiterungen (z.B. Gesetzesänderung)

# SONDERFALL: LEGACY SYSTEME (ALTSYSTEM)

- In der Vergangenheit entwickelt
- historisch gewachsten
- oft in veralteter Technologien und Programmiersprachen
- meist aufwendige und kostenintensive Systeme
- oft in Unternehmensabläufe verwoben
- Zunehmender Aufwand b. Weiterentwicklung

# DEBUGGING UND FEHLERSUCHE

# DEBUGGER

- Diagnose und Auffindung von Fehlern und Problemen
- Funktionen
  - Haltepunkte setzen (Einzelschritte überprüfen)
  - Daten untersuchen (Daten in flüchtigem Speicher)
  - Speicher modifizieren

# FEHLERTYPEN

- Laufzeitfehler
  - Arten von Fehler, die erst auftregen, während das Programm exekutiert wird
  - e.g. durch falsche Eingabe
- Programmierfehler (verhindern das Kompilieren)
  - Lexikalischer Fehler (undefinierte Variablen)
  - Syntaxfehler (fehlende Klammer)

# FEHLERTYPEN

› Semantische Fehler (inhaltlich falsch: e.g. Verwechslung vom Befehlcode)

```
* a = 0 vs a ==0
```

# FEHLERTYPEN

- Artithmetische Fehler
  - Division durch null
  - Precision (schlechtes Runden, e.g. float to int)
  - Arithmetischer Überlauf

32 Bit (Signed) Integer:

```
2.147.483.647 + 1 =  -2.147.483.648

( 01111111 11111111 11111111 11111111
                                  + 1 =
= 10000000 00000000 00000000 00000000 )
```

# FEHLERTYPEN

- Logikfehler
  - falscher Lösungsansatz
  - Endlosschleifen
  - Endlose Rekursionen
  - Off-by.one error in Arrays (OBOE)

# FEHLERTYPEN

- Designfehler
  - Fehler im Grundkonzept des Designs (e.g. durch mangelnde Erfahrung oder falsche Anforderungsspezifikation)

# FEHLERTYPEN

- Ressourcen
  - Null pointer dereferece
  - Falscher Datentyp
- Interface Errors
  - Interface Misure
  - Interface Misunderstanding
  - Timing Errors

# FEHLERTYPEN

- Performance
  - Runtime Errors / Multi-Threading
  - Deadlock
  - Race Condition
  - Mutual Exclusion,...

# FRAGEN?