Vorgehen im Softwareentwicklungsprozess

Der Softwareentwicklungsprozess

Für die Entwicklung von Software, namentlich für große Projekte, ist ein systematisches Vorgehen notwendig. Dieses Vorgehen, der Softwareentwicklungprozess, wird strukturiert durch Vorgehensmodelle, die wir in dieser Vorlesung betrachten.

Die Vorgehensmodelle für den Softwareentwicklungsprozess beinhalten

- Organisatorischer Rahmen für die Softwareentwicklung
- Strategie für die Durchführung eines Projekts

Dazu gehören

- Definition der Aktivitäten und Reihenfolge des Vorgehens
- Festlegung der Teilprodukte – Inhalt und Form
- Fertigstellungs- und Abnahmekriterien
- Rollen, Verantwortlichkeiten, Kompetenzen
- Standards, Richtlinien, Methoden & Werkzeuge

Produkte, Leistungen, Vorgehen

Die einzelnen Vorgehensmodelle definieren die Produkte und Leistungen, die im Laufe der Entwicklung zu erstellen oder zu erbringen sind und sie geben das Vorgehen dazu vor. Die in der Entwicklung im engeren Sinne vorkommenden Produkte, werden im Abschnitt über den Software-Life-Cycle kurz dargestellt. Darüber hinaus sind die Planung, das Qualitätsmanagement, das Risikomanagement, die Planung des Personaleinsatzes oder auch das Teammanagement Gegenstand des Softwareentwicklungsprozesses. Diese Aspekte werden wir nur streifen.

Häufig ist der Softwareprozess (oder Teile davon) auch Gegenstand des Vertrags zwischen Auftraggeber und Auftragnehmer.

Man findet oft als vertragliche Bestandteile:

Lastenheft
: - Anforderungen an die Leistungen und Lieferungen des Auftragsnehmers

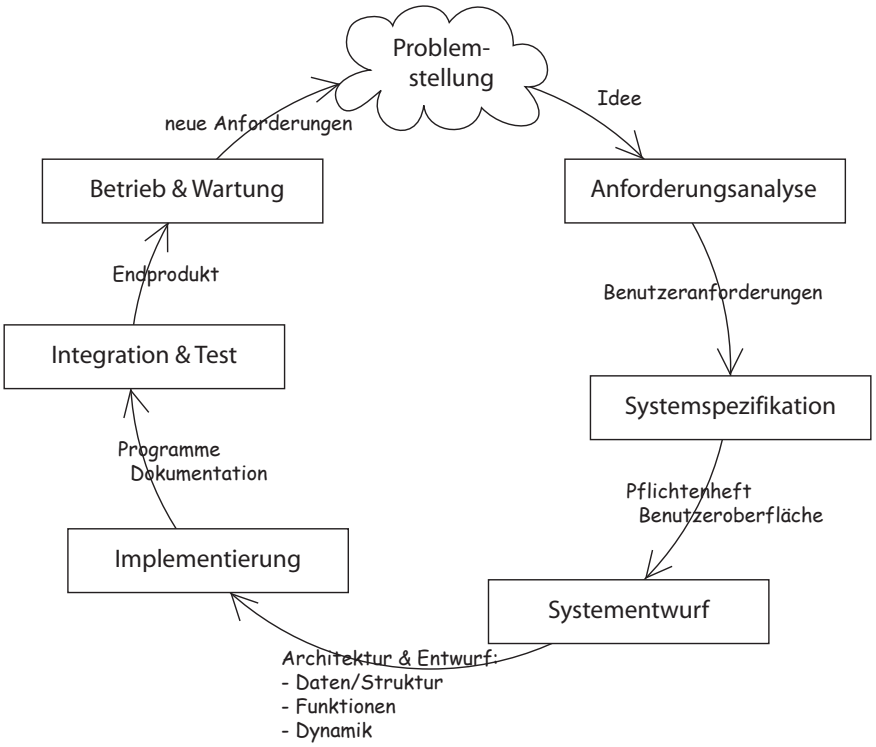
- Funktionale und qualitative Anforderungen in überprüfbarer Form
- Definiert, welche Aufgabe vorliegt und wofür sie zu lösen ist

Pflichtenheft

- Realisierungsvorgabe zur Umsetzung des Lastenhefts
- Umsetzung der Anforderungen in konkrete Lösungen
- Definiert, wie und wodurch die Anforderungen zu erfüllen sind.

Der Software-Life-Cycle

Der Software-Life-Cycle beschreibt, was im Zuge der Entwicklung von Software getan werden muss. Er stellt kein Vorgehensmodell dar, sondern eher eine Aufstellung all der Schritte, die in irgendeiner Form getan werden müssen, um Software zu erstellen, in Betrieb zu nehmen und zu warten.



Anforderungsanalyse Die Anforderungsanalyse beschreibt die Ziele des zu entwickelnden Systems, sie formuliert die funktionalen und nicht-funktionalen Anforderungen. Dafür werden oft Anwendungsfälle verwendet.

Ergebnis der Anforderungsanalyse ist die Kenntnis der Benutzeranforderungen.

Systemspezifikation Hierzu gehört die systematische Untersuchungen der Gegebenheiten des Anwendungsgebiets, die oft mittels Analysetechniken (heute insbesondere die objektorientierte Analyse mit UML als Notation) festgehalten werden. Erfasst werden die domänenspezifischen Begriffe, Konzepte und Vorgaben.

Aus dieser Darstellung der Gegebenheiten des Anwendungsgebiets als Fachmodell oder Geschäftsmodell wird darüberhinaus im Detail festgelegt, was das System leisten soll. In der Regel wird auch ein Entwurf der Benutzeroberfläche gemacht.

Im objektorientierten Vorgehen wird aus den Anwendungsfällen eine essentielle Lösung für das System gebildet, die von einer bestimmten Technologie unabhängig ist. Leitlinie für das objektorientierte Analysemodell ist die Szenarioanalyse, in der für die Anwendungsfälle die Schnittstellen-, Daten- und Kontrollerklassen definiert werden.

Als Ergebnis wird oft ein Pflichtenheft formuliert.

Systementwurf Der Entwurf spezifiziert das zu konstruierende System, indem eine Systemarchitektur (die Strukturierung in Systemteile) gefunden wird und die Komponenten dieser Architektur im Detail festgelegt werden.

Der Entwurf ist die Basis für die arbeitsteilige Entwicklung der Komponenten des Systems. Für den Entwurf müssen festgelegt werden: die Struktur des Systems, wie es seine Funktionen erfüllt und wie es sich zur Laufzeit dynamisch verhält.

Implementierung Die Implementierung ist die (in aller Regel arbeitsteilige) Umsetzung des Systementwurf in konkrete Software, also die Quelldateien, die Dokumentation, die Konfigurationsdateien usw.

Ergebnis der Implementierung sind Programme und Dokumentation.

Integration & Test Die konstruierten Komponenten werden zu einem Gesamtsystem zusammengeführt und dieses wird gegen die Systemspezifikation überprüft.

Daraus entsteht das auslieferfähige Endprodukt.

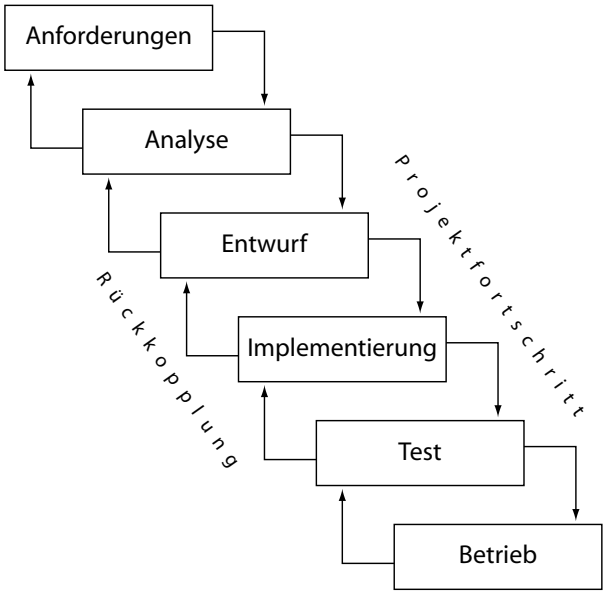
Betrieb & Wartung Aus Betrieb und Wartung, in denen sich das System (hoffentlich) bewährt, entstehen dann neue Anforderungen und der Kreislauf beginnt wieder.

Nochmals zur Warnung: gemeint sind mit dieser Darstellung nur sehr schematisch Aufgaben, die bei der Softwareentwicklung notwendig sind

– aber kein bestimmtes Vorgehen, geschweige denn in einer bestimmten zeitlichen Abfolge. Dazu gibt es die Vorgehensmodelle, die im Folgenden diskutiert werden.

Vorgehensmodelle

Wasserfall-Modell



Merkmale des Wasserfall-Modells

- Sequenzielles Vorgehen
- Vollständiger Abschluss einer Phase ist Voraussetzung für die nächste Phase
- Wenig Rückkopplung und wenig Vorausblick
- Oft wird eine Phase formell durch die Abnahme eines Dokuments (Artefakts) abgeschlossen
- Benutzerbeteiligung oft nur in den ersten Phasen

V-Modell

Das V-Modell ist das Vorgehensmodell des Bundes für die Entwicklung von IT-Systemen, auch von eingebetteten Systemen.

Charakteristik:

- Leitfaden zum Planen und Durchführen von Entwicklungsprojekten
- berücksichtigt den gesamten Lebenszyklus eines Systems
- Definition der zu erstellenden Arbeitsergebnisse
- Beschreibung konkreter Vorgehensweisen
- Festlegung der Verantwortlichkeiten: „wer, was, wann?“
- Festlegung der Kooperationen zwischen Auftraggeber und Auftragnehmer
- anpassbar für spezielle Projekte

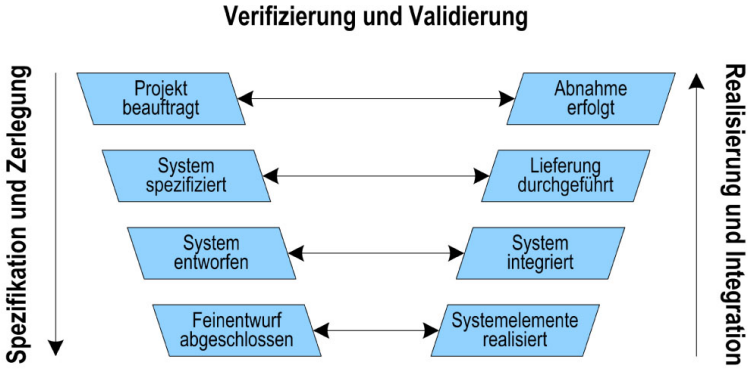
Geschichte:

- 1997 V-Modell als Entwicklungsstandard für IT-Systeme des Bundes
- 2005 V-Modell XT Weiterentwicklung inbesondere in Bezug auf komponentenbasierte Techniken und „Testfirst-Ansatz“
  XZ = „Extreme Tailoring“

Zielsetzung:

- Minimierung der Projektrisiken
- Verbesserung und Gewährleistung der Qualität
- Eindämmung der Gesamtkosten über den gesamten Projekt- und Systemlebenszyklus
- Verbesserung der Kommunikation zwischen allen Beteiligten

Folgende Graphik zeigt den Ausschnitt der eigentlichen Entwicklung aus dem Prozessmodell - den Ausschnitt, aus dem sich der Name „V-Modell“ ergab:



Merkmale des V-Modells:

- Unterteilung in Submodule:
  - Systemerstellung SE
  - Qualitätssicherung QS
  - Konfigurationsmanagement KM
  - Projektmanagement PM
- Unterscheidung von:
  - Aktivitäten — Arbeitsanleitungen
  - Produkte — Produktmuster
  - Rollen — Definierte Fähigkeiten/Kenntnisse
- Tailoring = „Maßschneidern“, d.h. Anpassen des Vorgehensmodells an das konkrete Projektvorhaben

Quelle: http://www.kbst.bund.de

Spiralmodell

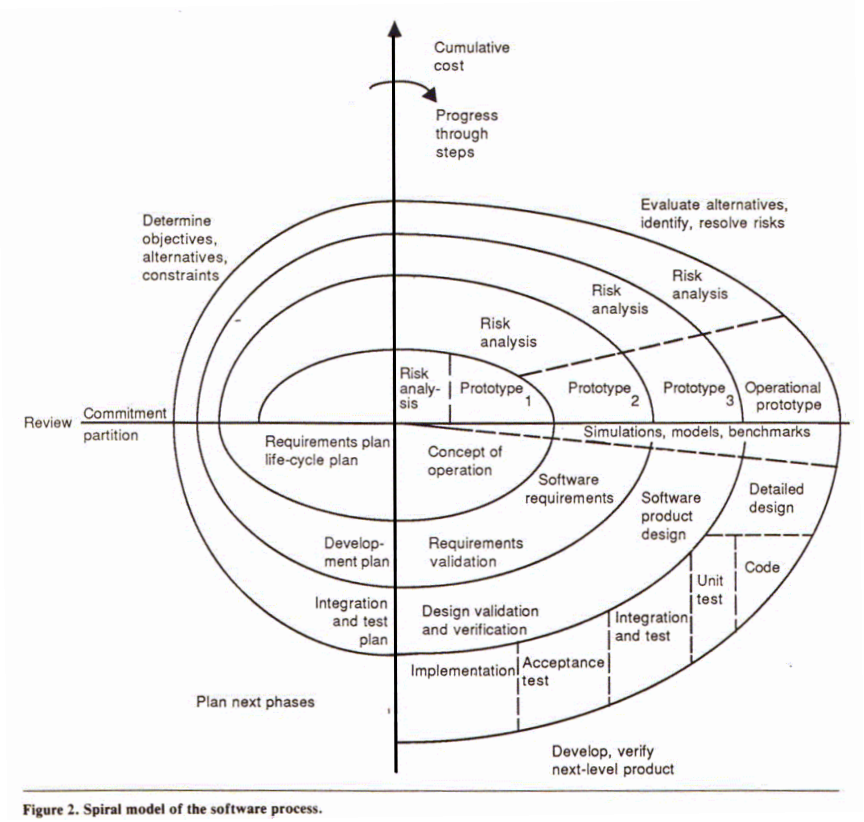


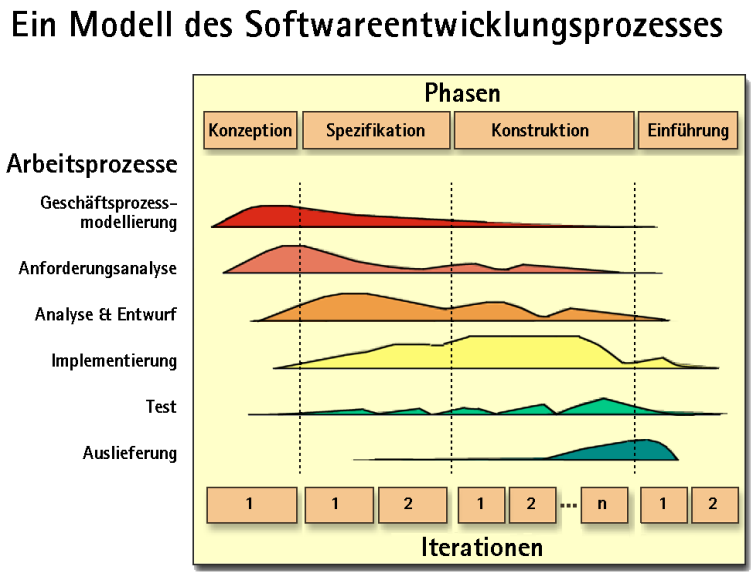
Figure 2. Spiral model of the software process.

Merkmale des Spiralmodells:

- Iterationen in 4 Schritten
  - Ziele, Alternativen, Randbedingungen identifizieren
  - Evaluierung der Alternativen in Bezug auf Risiken (auch mittels Prototypen)
  - Entwicklung und Überprüfung des Produkts im Iterationsschritt
  - Planung der nächsten Phase
- Meta-Modell: Innerhalb der Iteration wird über das Vorgehensmodell des nächsten Schrittes entschieden
- Risikogetrieben: Minimierung des (technischen und geschäftlichen) Risikios als wichtigstes Ziel
- Keine Trennung von Entwicklung und Wartung

Quelle: B.W.Boehm, A Spiral Modell of Software Development and Enhancement, in: IEEE Computer, May 1988

(Rational) Unified Process



Merkmale des Rational Unified Process:

- Charakteritik des (Rational) Unified Process
  - Getrieben durch Anwendungsfälle (*use-case driven*)
  - Architektur im Zentrum (*architecture-centric*)
  - iterativ und inkrementell
- Erprobte Vorgehensweisen (*best practises*)
  - Iterativ entwickeln
  - Anforderungen bewältigen
  - Komponenten(-architekturen) einsetzen
  - Visuell modellieren (UML)
  - Qualität überprüfen und sichern
  - Änderungen steuern
  - Werkzeuge einsetzen

  - Werkzeuge unterstützen das Vorgehensmodell
  - Werkzeuge steuern durch den Prozess
  - Prozess konfigurieren
- Schlüsselkonzepte
  - Rollen
  - Aktivität
  - Artefakte
  - Workflows als Integration von Rollen, die in Aktivitäten zu Artefakten gelangen

Lektüre: http://en.wikipedia.org/wiki/Rational_Unified_Process

Extreme Programming

Manager mögen Prozesse, weil dann

- alles planbar, berechenbar ist;
- alles steuerbar wird.

Der Irrglaube, Prozess stifte Qualität, ist weitverbreitet. (Natürlich ist in der Softwareentwicklung ein systematisches, diszipliniertes und auch reproduzierbares Vorgehen notwendig – darüber geht diese Vorlesung. Doch: dieses Vorgehen muss dem Ziel dienen.)

Überorganisierte Entwicklungsprozesse führen zu

- „Schrankware“ statt Software
- Entwicklern verharren in starr definierten Abläufen
- Kunden, Anwender wird aus dem Auge verloren
- Prozess verhindert rasches Reagieren auf Änderungen

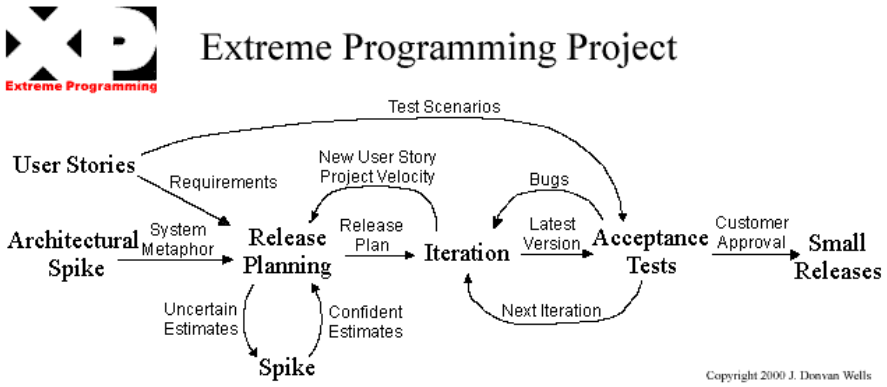
Aus dieser Erfahrung ist die Vorgehensweise des Extreme Programming XP entstanden.

Die Grundregeln von XP sind (siehe www.extremeprogramming.org):

- Konzentration auf das, worauf es wirklich ankommt, Tag für Tag
- Anwender (und Manager) sieht konkrete Fortschritte

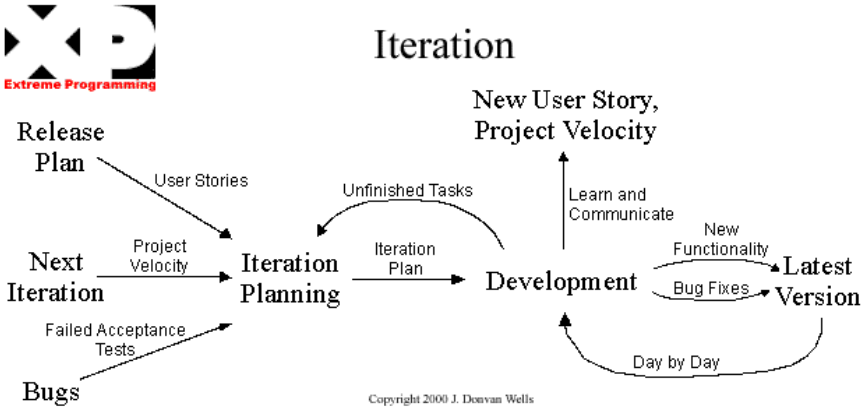
- Wie das: „XP takes commonsense principles and practices to extreme levels“

Das XP-Projekt:



- Release Planning Meeting: erstellt Release Plan
- Festlegung der zu implementierenden User Stories
- Jede Iteration erzeugt ein auslieferbares System
- Im Akzeptanztest wird die Implementierung der User Stories geprüft

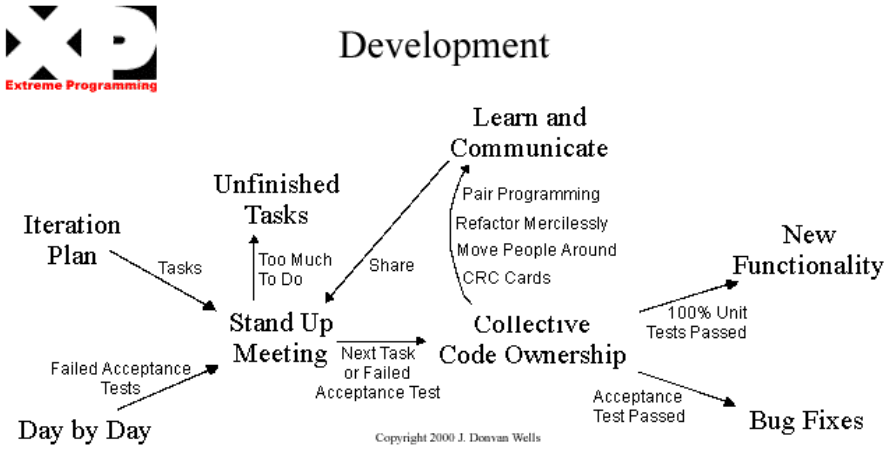
Iterationen im XP-Projekt:



- Iterationen von 1 - 3 Wochen Länge
- Keine Vorplanung, sondern erst zu Beginn der Iteration

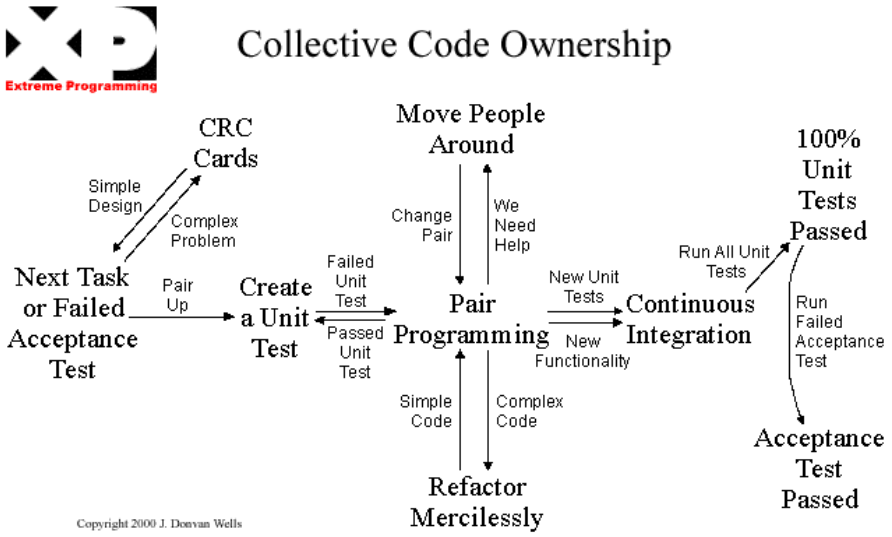
- Nur benötigte Funktionalität hinzufügen

Release entwickeln im XP-Projekt:



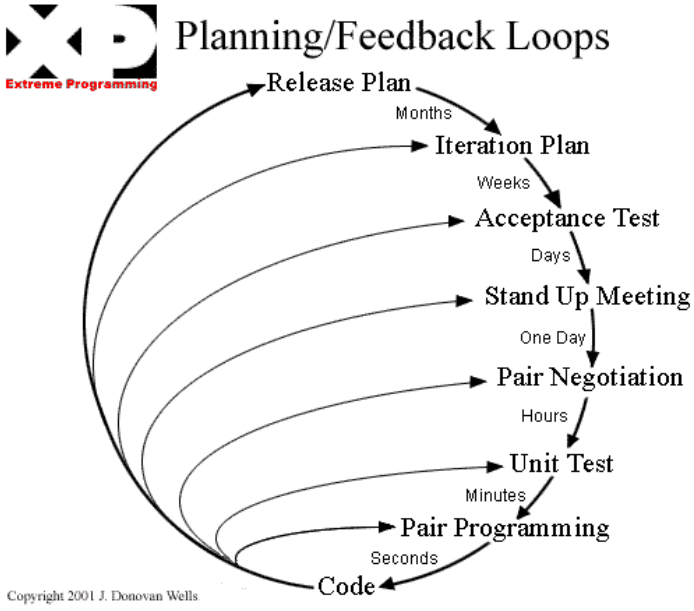
- Gemeinsame Verantwortung für den Code
- Generalisten, die einspringen können
- und voneinander lernen
- Tägliches Stand Up Meeting

Gemeinsames Codieren:



- Pair Programming - zwei sitzen gemeinsam vor dem Computer
- Refactoring
- Laufende Integration – Daily Build
- Unit Test – Test vor Code!!

Entwicklungsfluss in XP:



Und wo bleibt das Design?

- Explorative Prototypen
- System Metapher
- Ständige Diskussion und Verbesserung im Stand Up Meeting
- Refactoring

Agile Methoden

Aus dem XP-Ansatz haben sich mittlerweile eine ganze Reihe von sogenannten agilen Methoden entwickelt. Sie haben ihren gemeinsamen Ursprung im Manifest für agile Entwicklung http://agilemanifesto.org/ vom Februar 2001.

Manifesto for Agile Software Development
We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan
That is, while there is value in the items on
the right, we value the items on the left more.

Erstunterzeichner:
Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

Zu den agilen Methoden zählen:

- Extreme Programming XP
- SCRUM
- DSDM Dynamic Systems Development Method
- Adaptive Software Development
- Crystal
- Feature-Driven Development
- Pragmatic Programming

Viele Firmen setzen heute SCRUM ein: https://de.wikipedia.org/wiki/Scrum.

Lektüre: Udo Kelter Vorgehensmodelle Skript zur Softwaretechnik
http://pi.informatik.uni-siegen.de/kelter/lehre/04w/lm/lm_vm_info.html

Burkhardt Renz
Technische Hochschule Mittelhessen
Fachbereich MNI
Wiesenstr. 14
D-35390 Gießen

Rev 3.0 – 16. April 2012