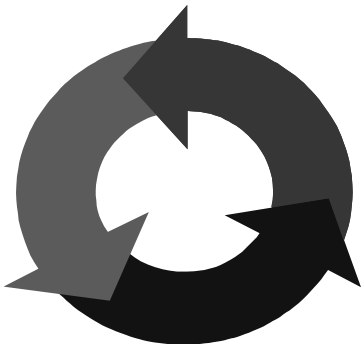
itelligence
Apcon Workplace Solutions
Member of itelligence

Extreme Programming (XP)

Henning Wolf, wolf@jwam.de
Martin Lippert, lippert@jwam.de
Stefan Roock, roock@jwam.de

Universität Hamburg &
APCON Workplace Solutions GmbH
Vogt-Kölln-Strasse 30
22527 Hamburg
Germany



itelligence
Apcon Workplace Solutions
Member of itelligence

Referenzen (1)

- Kent Beck: *Extreme Programming Explained* - Embrace Change. Addison-Wesley, 1999.
- Martin Fowler: *Refactoring : Improving the Design of Existing Code*. Addison-Wesley, 1999.
- **Lippert, Roock, Wolf: Software entwickeln mit eXtreme Programming - Erfahrungen aus der Praxis. dpunkt-Verlag, 2002.**
- Lippert, M., Roock, S., Wolf, H., Züllighoven, H.: *JWAM and XP - Using XP for framework development*, XP2000 conference. Cagliari, Sardinia, Italy (published in Extreme Programming Examined by Addison-Wesley, XP-Series, 2001).
- Lippert, M., Roock, S., Wolf, H., Züllighoven, H.: *XP in Complex Project Settings: Some Extensions*, XP2001 conference, Villasimius, Sardinia, Italy, 2001.

Referenzen (2)

itelligence
Apcon Workplace Solutions
Member of itelligence

- Williams, Laurie, Kessler, Robert R.: *Experimenting with Industry's "Pair-Programming" Model in the Computer Science Classroom*, Journal on Software Engineering Education, December 2000.
- Williams, Laurie, Kessler, Robert R.: Cunningham, Ward, and Jeffries, Ron, *Strengthening the Case for Pair-Programming*, IEEE Software, July/Aug 2000.
- Williams, Laurie and Kessler, Robert R.. *All I Really Need to Know about Pair Programming I Learned In Kindergarten*, Communications of the ACM, May 2000.
- Williams, Laurie and Upchurch, Richard. In Support of Student Pair Programming, 2001 SIGCSE Conference on Computer Science Education, Charlotte, NC, February 2001.
- Cockburn, Alistair and Williams, Laurie: *The Costs and Benefits of Pair Programming*, eXtreme Programming and Flexible Processes in Software Engineering XP2000.
- Williams, Laurie and Kessler, Robert R.: *The Effects of "Pair-Pressure" and "Pair-Learning" on Software Engineering Education*. Conference of Software Engineering Education and Training 2000.

XP-Ressourcen

itelligence
Apcon Workplace Solutions
Member of itelligence

- JUnit: http://www.junit.org
- WiKi-Web: http://c2.com/cgi/wiki.
- Laurie Williams: http://collaboration.csc.ncsu.edu/laurie/
- http://www.xprogramming.com
- http://www.ExtremeProgramming.org
- http://computer.org/seweb/dynabook/index.htm
- http://pairprogramming.com
- JWAM framework: http://www.jwam.org

Kontext

itelligence
Apcon Workplace Solutions
Member of itelligence

- XP ist noch jung und nimmt für sich in Anspruch, große Produktivitätspotenziale zu bieten.
- Erste Experimente mit XP an der Uni Hamburg Anfang 1999.
- Seit 2000 Einsatz von XP-Techniken im WPS-Firmenumfeld.

Erfahrungshintergrund

itelligence
Apcon Workplace Solutions
Member of itelligence

- Projekte in unterschiedlichen Domänen
- Breite Spanne an Projektgrößen: 4 PM in 11 Wochen bis mehrere 100 PM in mehreren Jahren)
- Multi-Channeling-Anwendungen
- Individual-Lösungen und Produktentwicklung
- Rahmenwerke (JWAM)

Hauptthese (1)

itelligence
Apcon Workplace Solutions
Member of itelligence

- XP *wirkt*
  - neu (eXtreme)
  - entwickler-zentriert (Programming)
  - ungeplant und chaotisch
  - unkontrolliert und unkontrollierbar
  - risikoreich
  - „Strukturierter Anarchismus“
- Das zieht Entwickler an und schreckt Manager ab.

Hauptthese (2)

itelligence
Apcon Workplace Solutions
Member of itelligence

- XP ist
  - bewährt (keine Technik ist neu)
  - anwendungsorientiert (anforderungsgetrieben)
  - geplant und diszipliniert
  - kontrolliert und verlässlich
  - risiko-minimierend
  - „IT Kommunismus“

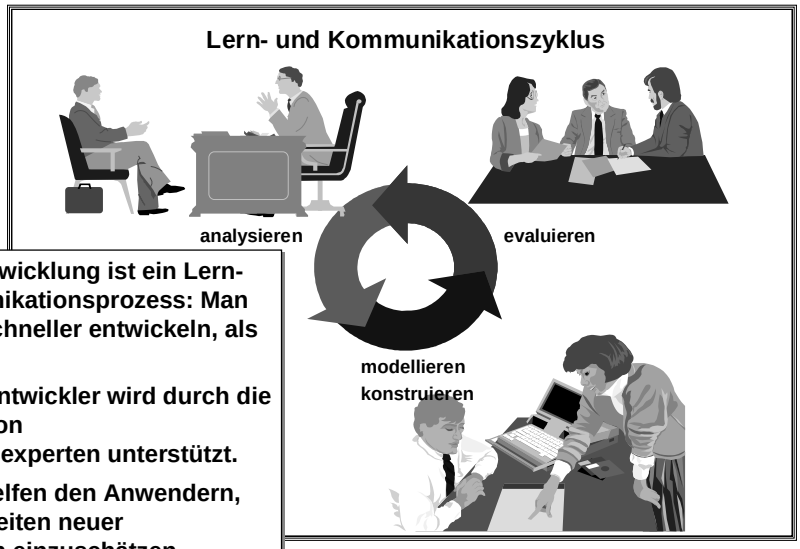
Hauptthese (3)

itelligence
Apcon Workplace Solutions
Member of itelligence

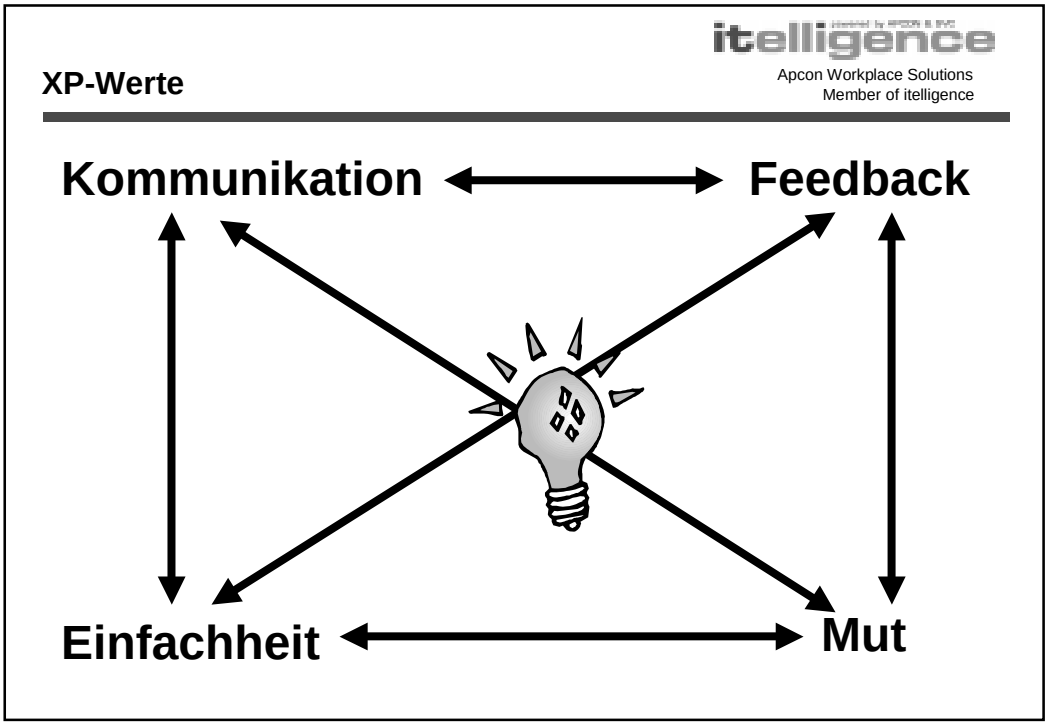
- XP hat zwei Seiten
  - für Entwickler: Freiheit, Flexibilität, Spaß
  - für Manager: Kontrollierbarkeit, Verlässlichkeit, Qualität

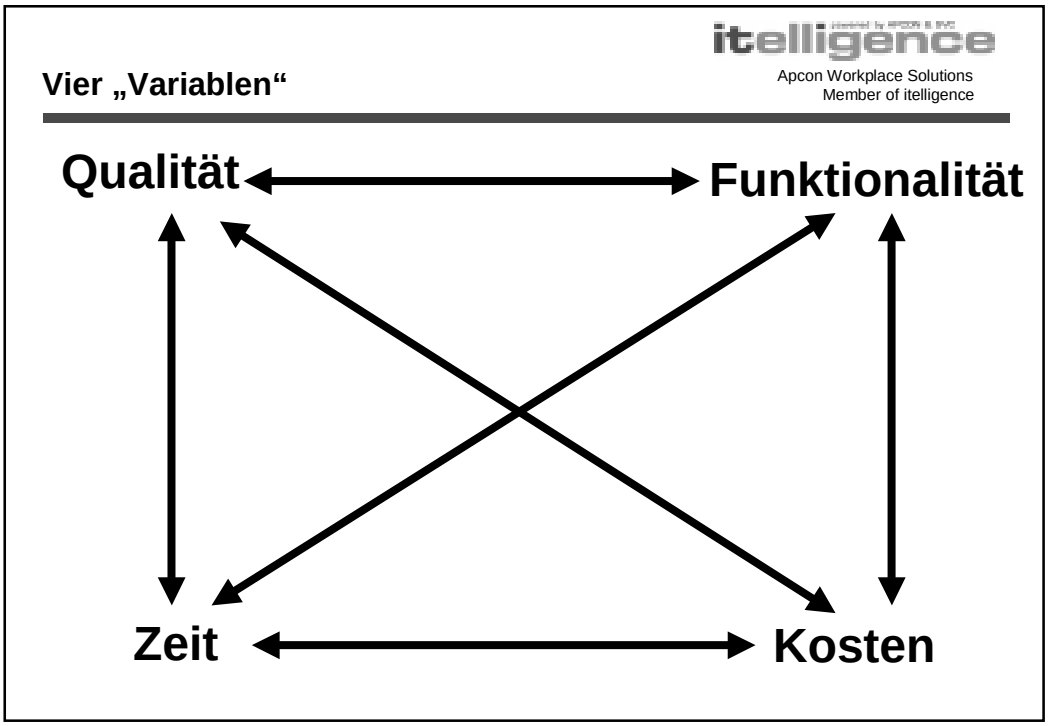
Zyklische und inkrementelle Entwicklung

itelligence
Apcon Workplace Solutions
Member of itelligence



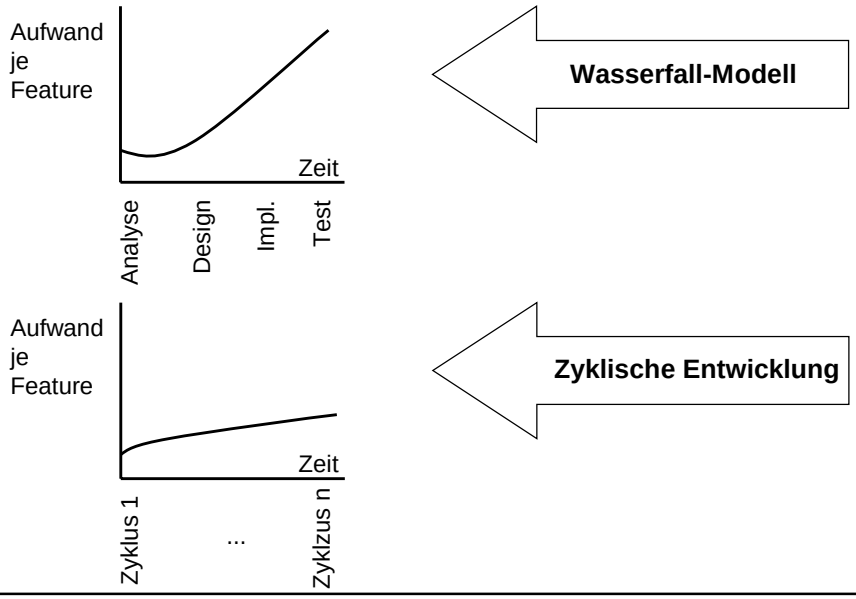
- **Software-Entwicklung ist ein Lern- und Kommunikationsprozess: Man kann nicht schneller entwickeln, als lernen.**
- **Lernen der Entwickler wird durch die Integration von Anwendungsexperten unterstützt.**
- **Entwickler helfen den Anwendern, die Möglichkeiten neuer Technologien einzuschätzen.**


XP-Werte
itelligence
Apcon Workplace Solutions
Member of itelligence
Kommunikation
Feedback
Einfachheit
Mut


Vier „Variablen“
itelligence
Apcon Workplace Solutions
Member of itelligence
Qualität
Funktionalität
Zeit
Kosten

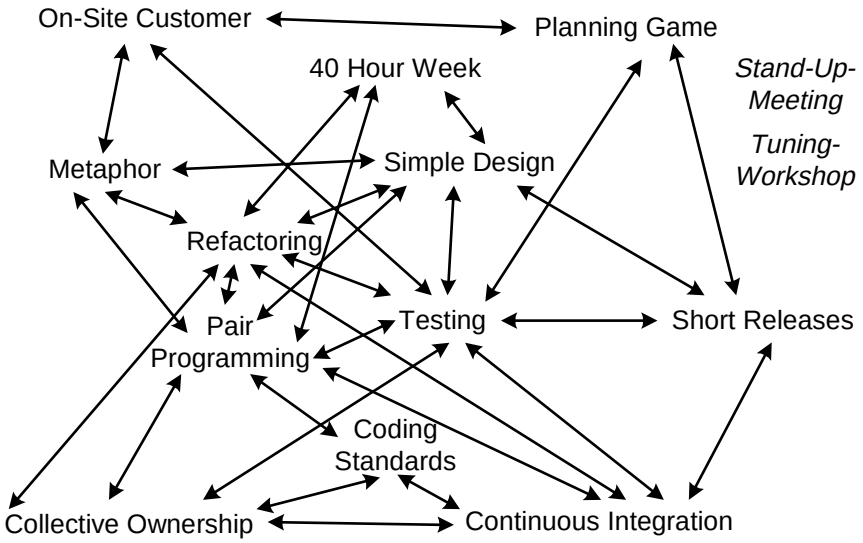
eXtreme Programming: Basisannahme

itelligence
Apcon Workplace Solutions
Member of itelligence



XP-Techniken: Überblick

itelligence
Apcon Workplace Solutions
Member of itelligence



XP-Technik: On-Site Customer (Kunde im Team)

itelligence
Apcon Workplace Solutions
Member of itelligence

- Anwender im Team.
- Fällt fachliche Entscheidungen.
- Beantwortet Fragen der Entwickler.
- Schreibt Anforderungen (User Stories) oder hilft beim Schreiben.

XP-Technik: User stories

itelligence
Apcon Workplace Solutions
Member of itelligence

- Definieren Anforderungen aus Anwendersicht.
- Kurz, keine vollständigen Spezifikationen.
- „Promise for conversation“
- Schätzbar.
- Testbar.
- Kann Akzeptanztests enthalten
- User Stories und Akzeptanztests unterstützen Fortschrittskontrolle im Projekt.

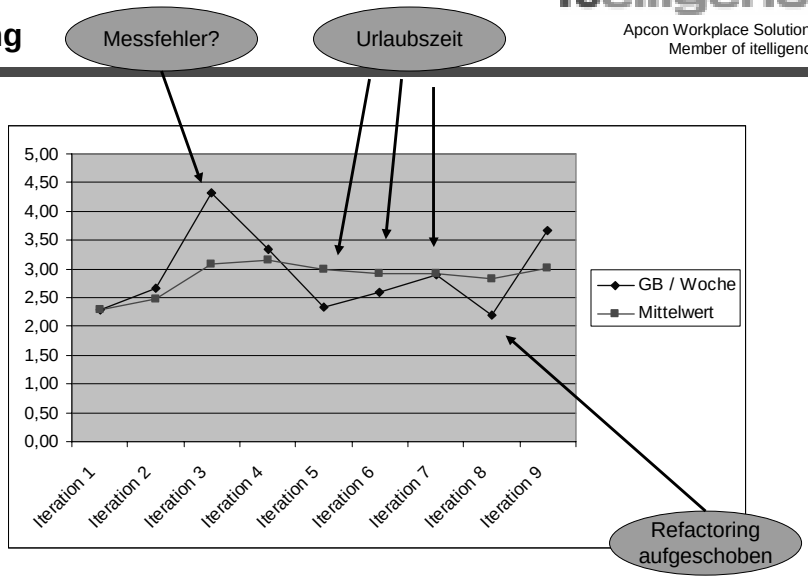
XP-Technik: Planning Game (Planungsspiel)

itelligence
Apcon Workplace Solutions
Member of itelligence

- Releaseplanung (1 bis 3 Monate).
- Iterationsplanung (max. 4 Wochen)
- Gemeinsame Aufgabe von Entwicklern und Anwender.
- Entwickler schätzen.
- Anwender priorisiert.

Tracking

itelligence
Apcon Workplace Solutions
Member of itelligence



XP-Technik: Short releases (Kurze Releasezyklen)

itelligence
Apcon Workplace Solutions
Member of itelligence

- Kleine, häufige Releases.
- Erstes Release nach 3 bis 6 Monaten.
- Danach neues Release alle 2-3 Monate.
- Erfahrungen mit dem produktiven Einsatz können schnell in die weitere Entwicklung einfließen.

XP-Technik: Tuning Workshop

itelligence
Apcon Workplace Solutions
Member of itelligence

- Ziel: Prozess verbessern.
- Zeitpunkt: Am Ende jeder Iteration.
- Teilnehmer: Alle.
- Vorgehen:
  - Was war gut?
  - Was ist verbesserungsfähig?
  - Wie können Verbesserungen erreicht werden?
- Ergebnis: ToDo-Liste.

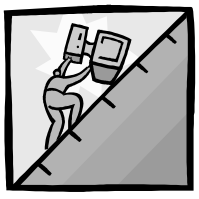
XP-Technik: Standup Meeting

itelligence
Apcon Workplace Solutions
Member of itelligence

- Ziel: Transparenz über Projektzustand und -fortschritt.
- Täglich 15 Minuten treffen.
- Alle stehen.
- Kurzer Statusbericht jedes Teammitgliedes:
  - Was habe ich in den letzten 24 Stunden getan?
  - Welchen Zustand habe ich erreicht?
  - Was werde ich in den nächsten 24 Stunden tun?
  - Welche Probleme habe ich im Moment?
- Inhaltliche Diskussionen außerhalb des Stand-Up-Meetings

XP-Technik: Metaphor

itelligence
Apcon Workplace Solutions
Member of itelligence



- Eine einfache, präzise Metapher gibt dem System seine innere und äußere Form.
- Konsistenz des Benutzungsmodells.
- Konsistenz der Architektur.

KISS: Keep It Simple, Stupid

itelligence
Apcon Workplace Solutions
Member of itelligence

- „Do the simplest thing that could possibly work“.
- Entfache Entwürfe sind
  - ➔ schneller zu implementieren
  - ➔ einfacher zu verstehen
  - ➔ schneller zu ändern
  - ➔ leichter zu testen

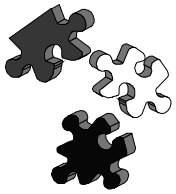
YAGNI: You Aren‘t Gonna Need It

itelligence
Apcon Workplace Solutions
Member of itelligence

- Keine Technology auf Vorrat bauen.
- Das bindet Ressourcen zur falschen Zeit.
- Ungenutzte Funktionalität wird in Akzeptanztests und im produktiven Einsatz nicht getestet.
- Ungenutzte Funktionalität altert sehr schnell.

Upfront Design will not succeed

itelligence
Apcon Workplace Solutions
Member of itelligence



- Design / Architektur ist sehr anspruchsvoll.
- Die Realisierbarkeit wird erst bei der Realisierung geprüft.
- Bewertung von Design ist kontextabhängig.
- Kontext kann und wird sich ändern:
  - ➔ Anwendungsbereich
  - ➔ Handhabung und Präsentation
  - ➔ Technologie.
- Was heute ein gutes Design ist, kann morgen ein schlechtes sein.

The „Truck-Factor“

itelligence
Apcon Workplace Solutions
Member of itelligence

- Der Truck-Faktor beschreibt die Wahrscheinlichkeit, mit der ein Projekt scheitert, wenn ein Projektmitglied von einem Truck überfahren wird.
- Der höchste Truck-Faktor (1,0) wird bei einem Spezialistenteam erreicht.
- Collective Ownership minimiert den Truck-Faktor.

XP-Technik: Collective Ownership (Gemeinsame Verantwortlichkeit)

itelligence
Apcon Workplace Solutions
Member of itelligence

- Der gesamte Code und alle erstellten Dokumente gehören dem Team.
- Jeder darf jederzeit alles ändern.
- Wenn etwas kaputt ist, ist das gesamte Team verantwortlich.
- Spezialwissen wird über Coaching und Schulungen ins Projekt transferiert.

XP-Technik: Continuous Integration (Kontinuierliche Integration)

itelligence
Apcon Workplace Solutions
Member of itelligence

- Mehrmals täglich integrieren.
- Es existiert ständig ein lauffähiges System auf dem Server.
  - ➔ Jeder kann sofort von neuem Code profitieren.
  - ➔ Neuer Code wird sofort von anderen Entwicklern getestet.
  - ➔ Risiko von Doppelentwicklungen wird reduziert.
  - ➔ Jederzeit kann ein lauffähiges System erstellt werden.
- Wenn sich Änderungen nicht mehr am selben Tag integrieren lassen, kann es sinnvoll sein, die Änderungen zu verwerfen und am nächsten Tag erneut zu beginnen.

XP-Technik: Testing (Testen)

itelligence
Apcon Workplace Solutions
Member of itelligence

- Unittests (Komponententests) und Akzeptanztests
- Ziel: 100% Anweisungsüberdeckung.
- Erster Ansatz: Je Klasse eine Testklasse, jede Operation einmal aufrufen.
- Verbesserter Ansatz: Test-First.
- Tests müssen automatisch ablaufen (JUnit).
- Tests werden ständig ausgeführt (Minutentakt).
- Der Serverbestand muss immer alle Tests bestehen.
- Wenn ein Problem im System auftritt, ist zunächst der Test zu korrigieren.

XP-Technik: Refactoring

itelligence
Apcon Workplace Solutions
Member of itelligence

- Refactoring ist die interne Umstrukturierung von Code, ohne die Funktionalität des Systems zu ändern.
- Es ist keine Schande, schlechten Code zu schreiben.
- Es ist allerdings schändlich, diesen nicht zu verbessern.
- Refactorings müssen in kleine Einzelschritte zerlegt werden (bei größeren Refactorings explizite Pläne notwendig)
- Jeder Refactoring-Schritt wird mit Tests abgesichert.
- Für die Rundablage: „Never change a running system“

XP-Technik: Pair-Programming (Programmieren in Paaren)

itelligence
Apcon Workplace Solutions
Member of itelligence

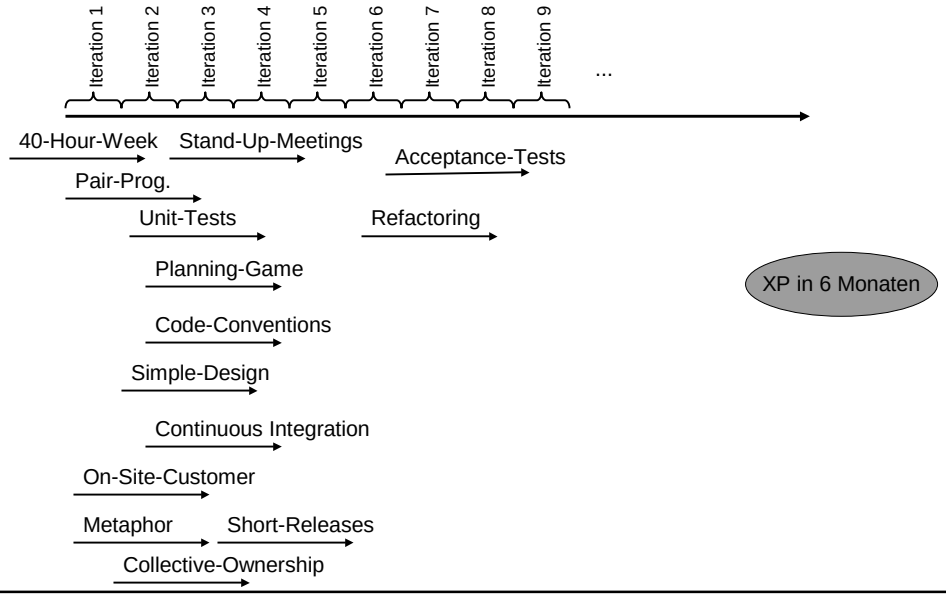
XP-Technik: Pair-Programming (Programmieren in Paaren)

itelligence
Apcon Workplace Solutions
Member of itelligence

- Immer zwei Entwickler an einem Rechner: Driver und Copilot.
- Ständiges Peer-Review.
- Höhere Code-Qualität:
  - ➔ weniger Bugs
  - ➔ bessere Schnittstellen
  - ➔ weniger Code (einfacheres Design)
  - ➔ besseres Design
  - ➔ weniger Redundanzen
- Wissen über System verteilt sich schnell im Team.
- Qualifikationen verteilen sich schnell im Team.
- Allerdings: Qualifikationsunterschied der Partner darf nicht zu groß werden.

Beispiel: Einführung der Techniken

itelligence
Apcon Workplace Solutions
Member of itelligence



Bewertung

itelligence
Apcon Workplace Solutions
Member of itelligence

Vorteile

- Eine zentrale Anlaufstelle für
  - Anforderungsdefinition
  - Entscheidungen
  - Bewertung des erstellten Systems

Nachteile

- Unterschiedliche Perspektiven von Geldgeber (Kunde) und verschiedenen Anwendergruppen.
- Multi-Channeling erfordert mehr als einen Kanal für die Anforderungen.
- Unklar, was zu tun ist, wenn Anwender keine brauchbaren Stories schreiben.
- On-Site Customer nicht immer realisierbar.