Extreme Programming (XP)

Yvonne Höller
Christian Lercher
Sebastian Stenger
Alexander Zrinyi

Universität Salzburg
FB Computerwissenschaften

16. Jänner 2009

Gliederung

Gliederung

Geschichte von XP

- Kent Beck, Ward Cunningham, Ron Jeffries.
- Projekt “Comprehensive Compensation System” (Daimler Chrysler).
- 1995 – 2000.
- $\longrightarrow$ dann Bücher (z. B. [Beck 2000]).

Einordnung

Schwer	Mittel	Leicht (“Agil”)
V-Modell	Spiralmodell	**XP**
Wasserfallmodell		ASD
Klass. Sequenzmodell		Scrum
...	...	...

- Kriterium: Formalisierungsgrad und Anzahl der Zwischenprodukte.

[POMBERGER und PREE 2004]

Besonderheiten

- “extrem”: treibt die besten Strategien der traditionellen Softwareentwicklung in den Extrembereich.

  [ANGIONI et al. 2006]

- “agile” Entwicklungsmethode: Welt ist nicht vorhersagbar
  $\longrightarrow$ Mit Veränderung arbeiten.

  [BECK 2000], [DYBÅ und DINGSØYR 2008]

- XP Programmierer denken mehr über Design nach.

  [DAMM et al. 2005]

- Zuerst Testen, dann Programmieren.

  [BECK 2001], [BECK 2003]

XP ist wie Autofahren: Es geht nicht darum, das Auto auf die richtige Bahn zu bringen. Auch wenn gerade alles perfekt läuft, darf man die Augen nicht von der Strasse lassen. Man muss ständig aufmerksam sein und kontinuierlich kleine Korrekturen anbringen.

[BECK 2000]

Gliederung

4 Zentrale Werte

- Kommunikation
  - Stetige Kommunikation im Team
    (Pair-Programming, Räumlichkeiten, Besprechungen, . . .)
  - Kommunikation mit dem Kunden
    (Storys, Interaktion, . . .)
- Feedback
  - Unit tests.
  - Feedback des Programmieres.
  - Feedback des Managers.
  - Feedback des Kunden.

[Beck 2000]

4 Zentrale Werte (Forts.)

- Einfachheit
  - So einfach wie möglich.
  - Redundanz vermeiden.
  - Lieber heute einfach und morgen etwas Komplexes einbauen, statt heute komplex, was morgen nicht in der antizipierten Form genutzt wird.
- Mut
  - Ehrliche und offene Kommunikation.
  - Fehler zugeben und Konsequenzen ziehen.
  - Veränderung auch von fremdem Code.

[BECK 2000]

Gliederung

Die vier Variablen

1. Kosten
   - Durch XP senken.
   - Marktparameter.

   [Beck 2000], [Voss 2005]
2. Zeit
   - Durch Testen Zeitaufwand senken.
   - Verluste minimieren durch kleine Schritte.

   [Beck 2000], [Damm et al. 2005]

Die vier Variablen (Forts.)

3. Qualität
   - Testgeleitetes Entwickeln ist der Wasserfallmethode überlegen.
   - Mehr Code.

[George und Williams 2004]

[Damm et al. 2005], [Dybå und Dingsøyr 2008]

4. Umfang
   - Am einfachsten anzupassende Variable.
   - Indem man nicht zu viel macht, ist man in der Lage, die geforderte Qualität zeitgerecht zu produzieren.

[Beck 2000]

Gliederung

Zusammenfassend: Wichtigste Prinzipien

- Schnelles Feedback.
- Einfachheit.
- Inkrementelle Veränderung.
- Qualitätsarbeit.

[BECK 2000]

Gliederung

Programmieren

- Programmierer produzieren Code, viel Code.
- Besser zu zweit: **Pair Programming!**

[Beck 2000]
[Dick und Zarnett 2002]
[Hazaan und Dubinsky 2003]
[Williams und Kessler 2003]

- Höhere Qualität.

[Bryant et al. 2008]

- Wissenschaftlich belegt.

[Erickson et al. 2005]

Testen

- Notwendiges Übel?

[BECK 2000]

- Testgeleitetes Entwickeln.

[BECK 2003]

- Schnelle, isolierte, automatisierte Tests.

[MAXIMILIEN und WILLIAMS 2003] [BECK 2000]

Testen (Forts.)

- Gleich viel Code für Programm und für Tests.

[HAYES 1995]

- Testen der Funktionalität und der Performanz.
- Arten:
  - Komponententests (Unit-Tests) von Programmierern,
  - Funktionalitätstests (bestimmt) von den Kunden.

[ANGIONI et al. 2006]

Design erstellen

- Testgeleitete Entwicklung ist Design-Technik.

[Damm et al. 2005]

- Schwierig.

[Dybå und Dingsøyr 2008]

- Anforderungen:
  - Erweiterbar,
  - modular,
  - eng geknüpft an Organisation der Daten,
  - vermeidet Redundanz,
  - ...

Zuhören

- Programmierer müssen den Kunden wirklich zuhören.
- nur die Kunden wissen, was sie brauchen
- Kommunikation allgemein:
  - Strukturierte, Regelmäßige Kommunikation.
  - Sicherstellung, dass das Mitgeteilte auch verstanden wird.

[Beck 2000]

Gliederung

Planen

- Spielerisch: The planning game.

[Beck 2000]

- Benutzergeschichten.

- Unterteilung in kleine und kleinere Schritte:
  - 1. Herausgabe
    - Wochenaufgabe
      - Tagesplan
        - Stunden, Minuten, ...

[Beck 2000]

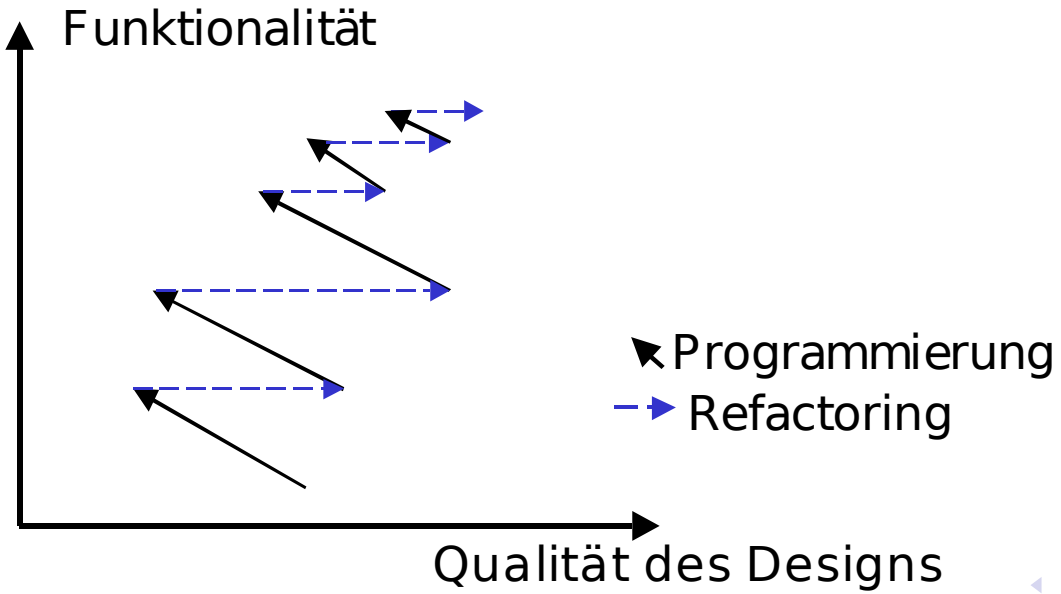
Testen. . . und dann Aufräumen

- Je einfacher das Design, um so einfacher Programmieren und Testen.

[Beck 2000]

- Tests bedeuten Verantwortung.
- Programmieren und Refactoring ergänzen sich.

[Rumpe 2001]



Gliederung

Beginn

- Exploration von
  - Technik, Architektur, Design;
  - Performance des Systems, Effizienz des Programmierens.
- Grober großer Plan.
- Definition erster naher Ziele.

[Beck 2000]

Produktion

- Früher Termin für Produktion.
- Enge Feedbackschleifen.
- Kurze Arbeitseinheiten.
- Kurze Zeitspanne zwischen Versionsherausgaben.

[Beck 2000]

Instandhaltung und Abschluss

- Normalzustand eines XP Projekts.
- Oft neue Versionen, damit Veränderungen nicht allzu groß sind.
- Das Ende eines Projekts ist erreicht, wenn ...
  - der Kunde keine Wünsche mehr hat und
  - die Software “fehlerfrei” läuft.
- Zum Abschluss: Dokumentation.

[BECK 2000]

Gliederung

Projekte

- Veränderung als “einzige Konstante” (bzw. häufig).

[BECK 2000], [POMBERGER und PREE 2004]

- Vage Anforderungen.

[POMBERGER und PREE 2004]

- Grüne Wiese.

[GITTINS et al. 2002]

- Optimaler Bereich: Kleinere, weniger kritische Projekte.

[MCCORMICK 2002]

Methode

- Als Trickkiste.

[McCormick 2002]

- Schrittweise Einführung.

[Gittins et al. 2002]

Gegenanzeigen

XP ist **nicht** gut einsetzbar, wenn:

- Entwicklung
  - verteilt,
  - langlebig → fehlende Dokumentation.

[GITTINS et al. 2002], [POMBERGER und PREE 2004]

- Unternehmensstruktur
  - starr traditionell,
  - mit wenig effektiver Kontrolle/Steuerung,
  - mit Fachleuten besetzt, die XP ablehnen.

[GITTINS et al. 2002], [MCCORMICK 2002]

Gliederung

Positiv

- Dokumentation durch Tests und Code → **stets automatisch aktuell**.
- Lauffähige Versionen frühzeitig vorhanden → Laufende Anpassung an Kundenwünsche.
- Qualität hat höchste Priorität (vor Kosten/Zeit/Umfang).

[GITTINS et al. 2002], [BECK 2000]

- Einfachheit als Ziel.

[SANZ 2002]

Belege?

- Oft *rein* akademisch.
- Wenige Daten aus echten Projekten.
- Wenige formale Studien.

[SANZ 2002]

- Glaubenssystem, "Kein Zweifel", Dogma.

[MCCORMICK 2002], [SANZ 2002]

Anwendbarkeit?

- Allgemein sind die extremen Ausprägungen der Arbeitsweisen schwer anwendbar.

[DYBÅ und DINGSØYR 2008]

- “Kunde vor Ort”-Prinzip eher Luxus.
- Metaphern zu finden ist schwierig.

[GITTINS et al. 2002]

- Multidisziplinäre Mitarbeiter zu finden ist schwierig.
- Einfaches Design: **Manchmal zu einfach**; zukünftige Erweiterungen sollten nicht immer ignoriert werden!

[SANZ 2002]


ANGIONI, M., D. CARBONI, S. PINNA, R. SANNA, N. SERRA und A. SORO (2006).
Integrating XP project management in development environments.
Journal of Systems Architecture, 52(11):619–626.


BECK, KENT (2000).
Extreme Programming Explained. Embrace Change.
Addison Wesley Longman, Inc., Massachusetts.


BECK, KENT (2001).
Aim, Fire [test-first coding].
IEEE Software, 18(5):87–89.


BECK, KENT (2003).
Test Driven Development - by example.
Addison Wesley.


BRYANT, SALLYANN, P. ROMERO und B. DU BOULAY (2008).

Pair programming and the mysterious role of the navigator.
International Journal of Human-Computer Studies, 66(7):519–529.


DAMM, LARS-OLA, L. LUNDBERG und D. OLSSON (2005).
Introducing Test Automation and Test-Driven Development: An Experience Report.
Electronic Notes in Theoretical Computer Science, 116:3–15.


DICK, A. und B. ZARNETT (2002).
Paired programming and personality traits..


DYBÅ, TORE und T. DINGSØYR (2008).

Empirical studies of agile software development: A systematic review.
Information and Software Technology, 50(9-10):833–859.


ERICKSON, J., K. LYYTINEN und K. SIAU (2005).
Agile Modeling, Agile software development, and extreme programming: the state of research.
Journal of Database Management, 16(4):88–100.


GEORGE, BOBY und L. WILLIAMS (2004).
A structured experiment of test-driven development.
Information and Software Technology, 46(5):337–342.


GITTINS, ROBERT, S. HOPE und I. WILLIAMS (2002).
Qualitative Studies of XP in a Medium Sized Business.
UPGRADE, The European Journal for the Informatics Professional, 3(2):18–21.


HAYES, L.G. (1995).
The Automated Testing Handbook.
Software Testing Institute.


HAZAAN, O. und Y. DUBINSKY (2003).
Bridging cognitive and social chasms in software development using extreme programming.


MAXIMILIEN, E.M. und L. WILLIAMS (2003).
Assessing test-driven development at IBM.
In: *Software Engigneering: Proceedings. 25th International Conference on*, S. 564–569. IEEE.


MCCORMICK, MICHAEL (2002).
Programming Extremism.
UPGRADE, The European Journal for the Informatics Professional, 3(2):9–10.


POMBERGER, GUSTAV und W. PREE (2004).
Software Engineering : Architektur-Design und Prozessorientierung.
Carl Hanser Verlag, München, Wien.


RUMPE, BERNHARD (2001).
Extreme Programming - Back to Basics?.
Technischer Bericht, Technische Universität München.


SANZ, LUIS FERNÀNDEZ (2002).
XP and Software Engineering: an opinion.
UPGRADE, The European Journal for the Informatics Professional, 3(2):23–25.


VOSS, RÖDIGER (2005).
BWL kompakt - Grundwissen Betriebswirtschaftslehre.
das Kompendium. Merkur Verlag, Rinteln.


WILLIAMS, L. und R. KESSLER (2003).
Pair Programming Illuminated.
Addison-Wesley, Boston.