Workshop "Praktiken des Programmierens"

Extreme Programming: Selbstorganisation durch gezielt ineinander greifende Praktiken

Lutz Prechelt
Freie Universität Berlin, Institut für Informatik
http://www.inf.fu-berlin.de/inst/ag-se/

- Software Engineering
- Begriff "Praktiken"
- (1) Einige Kernprobleme im Softwareprozess
- (2) Herkömmliche Antworten darauf
- Die Agile Bewegung
- Extreme Programming (XP)
- (3) Die Antworten bei XP: Ineinander greifende Praktiken

Kontext: Software Engineering (SE)

- Softwaretechnik (SWT) oder Software Engineering (SE): "Fachgebiet der Informatik, das sich mit der Bereitstellung und systematischen Verwendung von Methoden und Werkzeugen für die Herstellung und Anwendung von Software beschäftigt "
  - Hesse et al. 1984

- Wesentlicher Unterschied zum Programmieren liegt darin, dass Software Engineering die Kooperation mehrerer Personen verlangt
  - "Programmieren im Großen" (vs. "Programmieren im Kleinen")

- Wir nehmen im Folgenden an, die Anforderungen an das Softwareprodukt seien vorgegeben
  - und wir betrachten vor allem den Softwareprozess

Software Engineering in 2 Minuten

Freie Universität Berlin

Verkürzte Liste von Kernaufgaben im Software Engineering:

- Entwurf
  - Zerlegung des gedachten Ganzen in Einzelteile (Module)
  - Beschreibung jedes Moduls durch eine Schnittstelle
- Implementierung
  - Programmierung der einzelnen Module
- Test
  - einzelner Test von jeweils möglichst wenigen Modulen zusammen
    - auf Korrektheit und Vollständigkeit
  - Test des Systems im Ganzen
    - Korrektheit, Vollständigkeit, Angemessenheit, Benutzbarkeit, Robustheit, Geschwindigkeit, Speicherbedarf, Lastverhalten, Verfügbarkeit, Interoperabilität, Fehlertoleranz u.a.m.

Nebenher stets: Qualitätssicherung (QS)

- Prozess (konstruktive QS): Planung, Überwachung, Koordination
- Produkt (analytische QS): Durchsichten u.a.

Praktiken im Softwareprozess

- Herkömmliche Redeweise über den Softwareprozess:
  - Der Prozess besteht aus einzelnen Aktivitäten,
    die von Beteiligten in gewissen Rollen ausgeführt werden und
    aus Eingabedokumenten Ausgabedokumente (Artefakte)
    erzeugen
  - Es gibt eine Prozessdefinition (abstrakte Klasse von Prozessen)
    und eine konkrete Prozessdurchführung (Exemplare der Klasse)
- Definition und Durchführung werden oft kaum unterschieden

- **Praktik**: Eine Aktivität, die in Prozessdurchführungen regelmäßig auftaucht

Einige Kernprobleme beim SE

- (1) Es ist schwierig, beim Entwurf alles zugleich richtig zu bedenken und (2) Programmieren erfordert sehr hohe Konzentration.
- Deshalb (3) enthalten Programme oft Defekte und (4) müssen mehrfach wieder geändert werden.
- (5) Ein Programm zu verstehen ist schwierig, (6) zumal ein fremdes, und (7) es zu ändern ist riskant.
- (8) Es ist aufwändig, sich von der Richtigkeit einer Änderung zu überzeugen

Herkömmliche Antworten auf die Kernprobleme

- (1) Es ist schwierig, beim Entwurf alles zugleich richtig zu bedenken
  - Ausführliche Entwurfsphase
  - Modellierung des Entwurfs in verschiedenen Sichten
    - z.B. diverse UML-Diagramme
  - Systematische Entwurfsprüfung

Praktiken sind unterstrichen

Herkömmliche Antworten auf die Kernprobleme (2)

Freie Universität Berlin

- (2) Programmieren erfordert sehr hohe Konzentration.
  - Dieses Problem wird weitgehend ignoriert
  - DeMarco, Lister: "Programmer performance and the effects of the workplace", 1985

ENVIRONMENTAL FACTOR	TOP 25%	BOTTOM 25%
Dedicated floor space	78 sqft.	46 sqft.
Acceptably quiet workspace	57% yes	29% yes
Acceptably private workspace	62% yes	19% yes
Can you silence your phone?	52% yes	10% yes
Can you divert your calls?	76% yes	19% yes
Do people often interrupt you needlessly?	38% yes	76% yes
Does your workspace make you feel appreciated?	57% yes	29% yes

Herkömmliche Antworten auf die Kernprobleme (3)

Freie Universität Berlin

- Deshalb (3) enthalten Programme oft Defekte
  - Durchsichten
    - es mangelt jedoch oft an Disziplin, diese durchgehend durchzuhalten, zumal unter Zeitdruck
  - Gründliche Tests
- und (4) müssen mehrfach wieder geändert werden
  - Änderungshäufigkeit minimieren durch sorgfältigen Entwurf
  - Änderungen nach Defekten werden als unausweichlich akzeptiert
- (5) Ein Programm zu verstehen ist schwierig,
  - Unterstützung durch ausführliche Dokumentation der Ideen
  - Unterstützung durch Nutzung von Softwarewerkzeugen
- (6) zumal ein fremdes,
  - Änderungen durch Originalautor vornehmen lassen
- und (7) es zu ändern ist riskant
  - Gründlicher Entwurf der Änderung
  - Gründlicher Test der Änderung

Herkömmliche Antworten auf die Kernprobleme (4)

Freie Universität Berlin

- (8) Es ist aufwändig, sich von der Richtigkeit einer Änderung zu überzeugen
  - Auswahl der Tests, die wiederholt werden müssen

Beobachtungen:

- Optimaler Entwurf gilt als
  - a. extrem wichtig, aber zugleich
  - b. extrem schwierig
- Eine sehr große Rolle spielen unter anderem
  - Genaue Entwurfs*dokumentation*
  - *Separate* Aktivitäten zur Qualitätssicherung
- Viele verschiedene Maßnahmen

Ferner (nicht erwähnt):

- Intensives Projektmanagement zur Planung und Koordination nötig

Kulturrevolution: Die Agile Bewegung "Agile Manifesto"

Freie Universität Berlin

http://www.agilemanifesto.org

Manifesto for Agile Software Development (2001)

- "We are uncovering better ways of developing software by doing it and helping others do it.

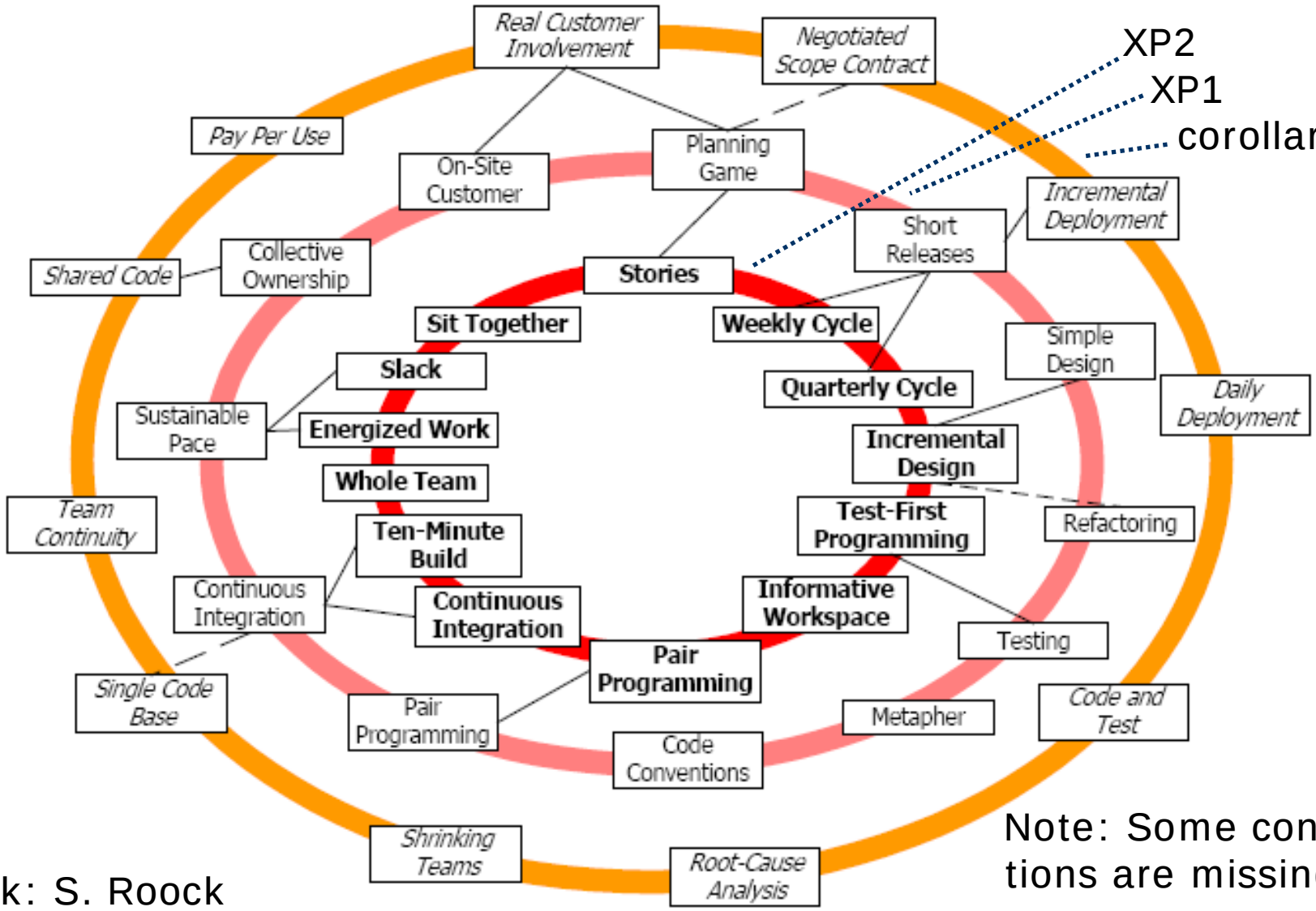
  Through this work we have come to value:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan

  That is, while there is value in the items on the right, we value the items on the left more."

Extreme Programming (XP): Ein Satz von Praktiken

- Inwiefern "extrem"?
  - Nimmt Vorgehensweisen, die bekanntermaßen hilfreich sind, "dreht den Knopf auf 10" und definiert sie als Praktiken
- Die einzelnen Praktiken sind nicht neu, neu sind aber
  - der Stil und Grad ihrer Anwendung und
  - ihre Kombination
- Originalquelle: Kent Beck: *"Extreme Programming Explained: Embrace Change"*, Addison-Wesley, 1999
  - 12 Praktiken
  - 2. Ausgabe 2004 mit Dirk Andres: 13 z.T. erheblich reformulierte Praktiken
  - Ich werde Begriffe aus XP1 und XP2 frei mischen

The XP practices, old and new



Note: Some connections are missing

Grafik: S. Roock

XPs Antworten auf die Kernprobleme

- (1) Es ist schwierig, beim Entwurf alles zugleich richtig zu bedenken
  - Simple Design: Versuche nicht, künftigen Bedarf im Entwurf vorherzusehen. Baue die einfachste Lösung, die *heute* ausreicht.
  - Short Releases: Baue das Gesamtprodukt schrittweise
  - Refactoring: Baue Deinen Entwurf stets sofort um, wenn Du eine Verbesserungsmöglichkeit entdeckst.
    Betrachte solche Änderungen am Entwurf als normal und wünschenswert.
  - Test-First Programming: Bevor man eine Klasse oder Methode implementiert, schreibt man zuerst automatisierte Tests dafür
  - Pair Programming: Entwurf und Implementierung sind eins und werden stets zu zweit ausgeführt

Praktiken sind unterstrichen

XPs Antworten auf die Kernprobleme (2)

- (2) Programmieren erfordert sehr hohe Konzentration.
  - XP1: Sustainable Pace (40-hour week), XP2: Energized work
    - "no extended stretches of working overtime"
  - XP2: Slack: Explizite Zeit für Spaß, Fortbildung, Experimente etc.
  - Pair Programming: Alle Programmierarbeit wird von zwei Personen gemeinsam erbracht



XPs Antworten auf die Kernprobleme (3)

- Deshalb (3) enthalten Programme oft Defekte
  - Pair Programming: Defekte gleiich beim Entstehen fangen
  - Simple Design: Komplexität minimieren, um Defekten vorzubeugen
  - Refactoring: Komplexitätszuwachs bei der Fortentwicklung kontinuierlich bekämpfen
  - Test-first programming: Vor Realisierung jedes Einzelteils stets genau die Anforderungen klären
- und (4) müssen mehrfach wieder geändert werden
  - Das ist der erwartete Normalfall: "We have come to value Responding to Change over Following a Plan"
  - Continuous Integration: Ständiges automatisiertes Zusammenbauen und Testen aller Teile stellt Passung sicher

XPs Antworten auf die Kernprobleme (4)

Freie Universität Berlin

- (5) Ein Programm zu verstehen ist schwierig,
  - Simple Design: Ersetzt weitgehend eine Entwurfsdokumentation
  - Refactoring: Beugt kontinuierlich schwerer Verständlichkeit vor
- (6) zumal ein fremdes,
  - Pair Programming: Es gibt zumindest zwei Leute, denen es nicht fremd ist
  - Code Conventions, Simple Design, Test-First Programming: Erleichtern das Verstehen auch fremder Teile
- und (7) es zu ändern ist riskant
  - Simple Design, Refactoring: Halten das Risiko niedrig
- (8) Es ist aufwändig, sich von der Richtigkeit einer Änderung zu überzeugen
  - Test-First Programming, Continuous Integration: Sorgen dafür, dass Probleme sofort und leicht erkannt werden

XPs Antworten auf die Kernprobleme (5)

Freie Universität Berlin

Beobachtungen:

- Die Praktiken funktionieren gut und ergänzen sich glänzend
  - Allerdings können konsequentes Refactoring und gründliches Test-First Programming sehr aufwändig werden
  - und das Verfahren skaliert nicht gut für große Projekte
- Entwurfsdokumentation und separate Qualitätssicherung werden weitgehend direkt in die Entwicklungspraktiken integriert
  → Verbesserte Selbstorganisation des Prozesses
- XP erfordert hohe Disziplin von allen Beteiligten!

Danke!