Agile Designprozesse

Jörg Cassens

Medieninformatik II
Contextual Design of Interactive Systems
SoSe 2016

Universität Hildesheim
medieninformatik
IMAI – Institut für Mathematik und Angewandte Informatik

1 Agile Ansätze

Agilität

- Unter dem Dachbegriff "Agile Prozesse" ist eine ganze Reihe verschiedener Entwicklungsmethoden entstanden
- Agile Prozesse stellen einen hochgradig interaktiven und leichtgewichtigen Ansatz dar
- Herkömmliche Prozesse häufig mit großem bürokratischen Aufwand verbunden
- Dies halten die Befürworter agiler Prozesse für eine Verschwendung von Arbeitskraft
  - Umfangreiche Dokumentation der Meilensteine
  - Umfassende Qualitätssicherung der erreichten Meilensteine
- Akzeptanz von Bürokratie bei Entwicklern oft gering
- Viele Aufgaben "lästige Pflicht"
- Dagegen:
  - Keine Dokumentation mehr schreiben
  - Keine Meilensteine im klassischen Sinn

Manifest

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.

☞ agilemanifesto.org

Principles I

1. Our highest priority is to **satisfy the customer** through early and continuous delivery of valuable software.
2. **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver **working software** frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must **work together daily** throughout the project.
5. Build projects around **motivated individuals**. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.

Principles II

7. **Working software** is the primary measure of progress.
8. Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to **technical excellence and good design** enhances agility.
10. **Simplicity** – the art of maximizing the amount of work not done – is essential.
11. The best architectures, requirements, and designs emerge from **self-organizing teams**.
12. At regular intervals, the **team reflects** on how to become more effective, then tunes and adjusts its behavior accordingly.

2 Modelle

2.1 XP

Beispiel XP

- **Planungsspiel**: Konstruktive Verhandlung zwischen Auftraggeber und Entwicklern zur Erreichung einer Balance von Kosten und Funktionalität
- **Kleine Versionsschritte**: Sehr kleine Schritte zunehmender Funktionalität an Auftraggeber ausliefern
- **Metapher**: eine das gesamte System umfassende Metapher, die einen ganzheitlichen Entwurf ermöglicht
- **Einfacher Entwurf**: Immer die einfachste Lösung bevorzugen, Komplexität nur einbauen, wenn notwendig
- **Testen**: Statt komplexer Spezifikationen vorab automatisierte Tests erstellen
- **Refaktorisierung**: Programmcode bei nachgewiesener Notwendigkeit (unter Zuhilfenahme von Entwurfsmustern) umschreiben, Tests stellen sicher, daß gleiche Funktionalität erreicht wird

Beispiel XP

- **Paar-Programmierung**: Eine Person programmiert, die andere beobachtet, korrigiert, plant
- **Kollektiver Code-Besitz**: Kein verantwortlicher Autor für Module
- **Fortlaufende Integration**: Keine Integration separat entwickelter Module zum Schluß, stattdessen soll es immer eine lauffähige Version geben (keine einchecken von Teilen, die die Tests nicht durchlaufen)
- **40-Stunden-Woche**: kontinuierliche, nachhaltige Arbeit hoher Qualität durch vernünftige Arbeitszeiten
- **Kunde vor Ort**: im Entwicklerteam ist ständig ein Vertreter des Kunden ansprechbar
- **Code-Standards**: einheitliche Vorschriften für Layout und Kommentierung

2.2 FDD

Feature Driven Development

- Fünf prinzipielle Aktivitäten
  - **Develop Overall Model** mit high- und low level walkthroughs, Optionen werden untersucht, ein Model gewählt
  - **Build Feature List** mittels funktioneller Dekomposition und Identifikation von Aktivitäten: <action> <result> <object>
  - **Plan By Feature** in dem ein Entwicklungsplan aufgestellt wird, Verantwortlichkeiten festgelegt, und Resourcen zugeteilt
  - **Design By Feature** wobei für jedes Feature Pakete geschnürt werden und eine detailliertere Modellierung erfolgt (UML)
  - **Build By Feature** die Pakete werden implementiert, getestet, und bei Erfolg integriert.
- Gliederung durch Milestones

2.3 SCRUM

SCRUM I

- Prozeßskelett mit Vorgehensweisen und Rollen
- Pigs:
  - Scrum Master
  - Product Owner
  - Team
- Chicken:
  - Stakeholders
    - Customers
    - Vendors
  - Managers

SCRUM II

- Organisiert in Sprints
- Strukturiert durch Meetings
  - Daily SCRUM
    - Was habe ich gemacht?
    - Was mache ich weiter?
    - Welche Probleme gibt es?
  - Sprint Planning Meeting
    - Vorher: Was ist zu tun?
  - Sprint Review Meeting
    - Danach: Was wurde geschafft?
    - Produkt- und Kundenorientiert
  - Sprint Retrospective
    - Danach: Wie wurde es geschafft?
    - Prozeß- und Teamorientiert

2.4 DSDM

Dynamic Systems Development Method I

- Rapid Application Development
- Drei Phasen
  - Pre-Project
  - Project life-cycle
  - Post-project
- Fünf Stufen im project life-cycle
  - Machbarkeitsstudie (sequentiell)
  - Business Study (sequentiell)
  - Functional Model (iterativ, prototypisch)
  - Design and Build (iterativ, integrativ)
  - Implementation (sequentiell, beim Kunden)

Dynamic Systems Development Method II

- Timeboxing
  - Split project into chunks with fixed budget, fixed time.
  - Pareto principle: 80% of effects stems from 20% of causes.
- MoSCoW
  - must have, should have, could have, would have.
- Prototyping
  - RAD background: Working prototypes which are delivered.
- Testing
  - Throughout each iteration.
- Workshop
  - Bringing stakeholders together.

3 Bewertung

Gemeinsamkeiten
Oft, aber nicht immer, finden sich folgende Gemeinsamkeiten bei den diversen Methoden und Vorgehensmodellen, wenn auch in unterschiedlicher Ausprägung:

- Iterativität
  - XP, FDD, SCRUM, DSDM
- Inkrementalität
  - XP, FDD, (DSDM)
- Feature-Bezug
  - (XP), FDD, SCRUM
- Umfangreiche Tests
  - XP, FDD, DSDM
- Code Gemeineigentum
  - XP, (SCRUM)

Kritik

- Vernachlässigung von Dokumentation
- Man braucht erfahrene, selbständige Entwickler
- Kostspielig durch viele Iterationen und Meetings
- Sind iterativ erstellte Implementationen Wegwerf-Prototypen oder Teil des Produktes?
- Aufwand und Dauer sind schwer abzuschätzen
- Wie können andere Qualitätsmerkmale als Features eingebunden werden? Usability!

Anwendbarkeit

- Agile home ground:
  - Low criticality
  - Senior developers
  - Requirements change very often
  - Small number of developers
  - Culture that thrives on chaos
- Plan-driven home ground:
  - High criticality
  - Junior developers
  - Requirements don't change too often
  - Large number of developers
  - Culture that demands order

Boehm and Turner [2003]

Challenges Usability

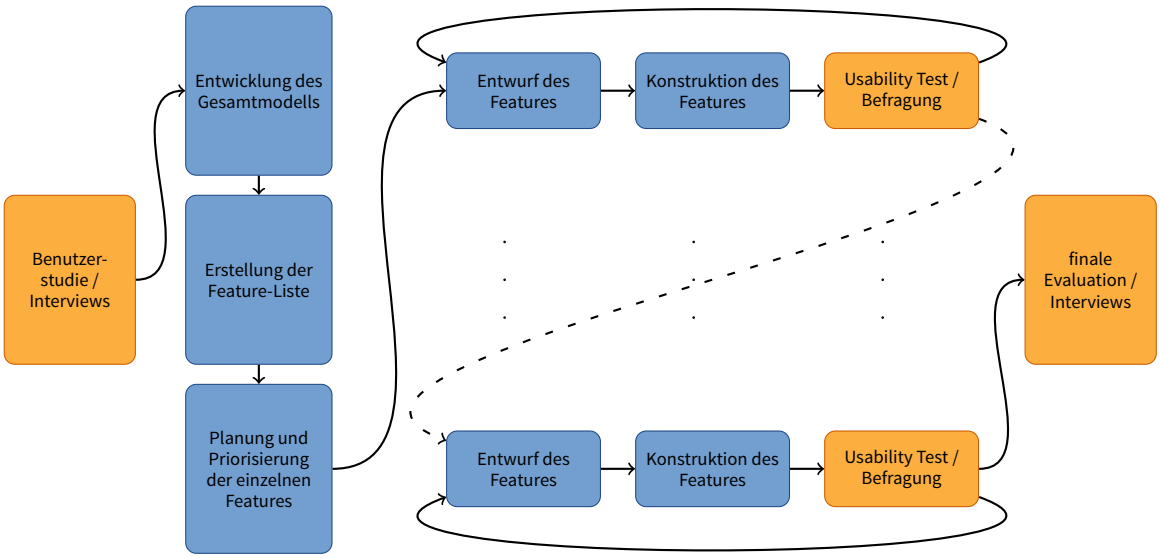
- **Different goals.** Technics-centered vs. human-centered.
- **Different approaches.** User-centered vs. stakeholder-centered.
- **Organizational challenges.** Collaboration of generalists vs. expert culture.
- **Process impedance mismatch.** "Big Design Up Front (BDUF)" vs. models on the go.
- Usability practitioners **struggle to be heard**.

Ambler [2008]

Herausforderungen Medieninformatik

- Vorteile agiler Prozesse
  - Entwicklung digitaler Medienprodukte häufig in kleinen Teams
  - Viele Gestaltungs- und Interaktionsaspekte schlecht abschließend definierbar
- Probleme agiler Prozesse
  - Vorproduktion aufwendig gestalteter Medienelemente (Hintergrundmusik, Produktvideos) steht im Widerspruch zu einem agilen, iterativen Prozeß
  - Automatisiertes Testen funktioniert hervorragend für Programmcode, bei dem Eingabevektoren und Ausgabevektoren gut formalisierbar sind
  - Das automatische Testen interaktiver Programme schwierig (Simulation von Eingabe und Nutzerreaktionen auf Ausgaben)
- Sind Kombinationen möglich, bei denen agile Prozesse in erster Linie für die Codeentwicklung eingesetzt werden, und menschzentrierte Prozesse für die Mediengestaltung?

Kombination



Kombination von FDD (Blau) und UCD (Orange), nach [Roenspieß, 2009].

4 Bildnachweis

Alle Abbildungen, wenn nicht anders angegeben, aus:
Malaka, Rainer; Butz, Andreas; Hussmann, Heinrich: *Medieninformatik – Eine Einführung*. ISBN 978-3-8273-7353-3, München: Pearson Studium, 2009.

References

Literatur

Scott W. Ambler. Tailoring usability into agile software development projects. In *Maturing Usability*, Human–Computer Interaction Series, pages 75–95, Heidelberg, 2008. Springer Verlag. doi: 10.1007/978-1-84628-941-5_4.

Barry W. Boehm and Richard Turner. Observations on balancing discipline and agility. In *Agile Development Conference/Australasian Database Conference*, pages 32–44, Los Alamitos, CA, USA, 2003. IEEE Computer Society. ISBN 0-7695-2013-8. doi: 10.1109/ADC.2003.1231450.

Amelie Roenspieß. Entwicklung einer mobilen Benutzungsschnittstelle für das Terminkoordinationssystem TeaCo. Bachelorarbeit, Institut für Multimediale und Interaktive Systeme, Universität zu Lübeck, 2009.