Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Agile Entwicklung und Architektur

Leon Fausten

Grundseminar WS 2014/15

12.12.2014

Agenda

- Scrum
- Motivation
- Refactoring
- Das Agile Manifesto und Architektur
- Agile Model Driven Development
- Architektur im Sprint Planning
- Die Rolle einer Softwarearchitekten
- Technische Unterstützung im Meeting
- Fazit
- Angestrebte Ziele

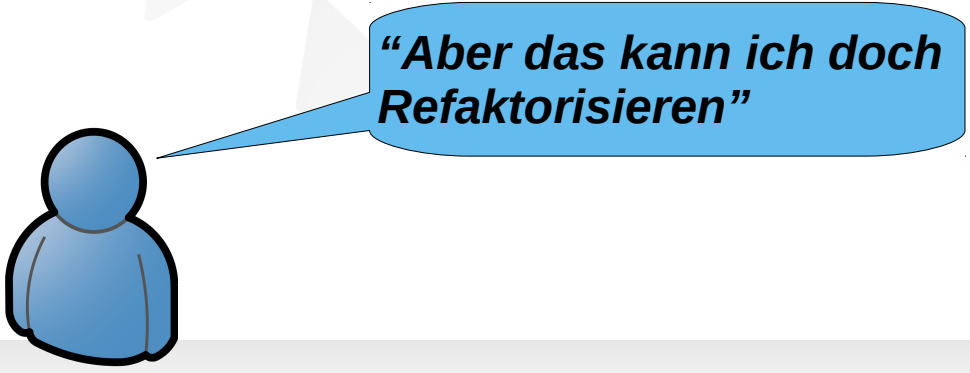
Scrum


Input from End-Users, Customers, Team and Other Stakeholders
Product Owner
1
2
3
4
5
6
7
8
9
10
11
12
FEATURES
Product Backlog
Team
Team Selects How Much To Commit To Do By Sprint's End
Sprint Planning Meeting
(Parts One and Two)
TASKS
Sprint Backlog
ScrumMaster
Product Backlog Refinement
Sprint
1-4 Weeks
No Changes
in Duration or Goal
Daily Scrum Meeting and Artifacts Update
Review
Potentially Shippable Product Increment
Retrospective

[1]

Motivation

- Architektur schnell vernachlässigt ! Hauptrisiko Punkt
- Bewusstsein schaffen, dass Architektur wichtig ist
- Architektur im Sprint Planning berücksichtigen
  - Wann? - Was? - Wer? - Wie?
- Eigenschaften wie Skalierbarkeit direkt unterstützen



Motivation - Fallstudie

- Während Entwicklungszeit
- Kapazität und Performance nicht mehr ausreichend
- Refaktorisierung beschlossen
- Managament zufrieden über Qualitätsanspruch
- Kosten und Zeit erheblich überschritten
- Unterschiedliches Verständnis der Änderungen

> Refactoring isn't limited to code detail but can range up to the larger scale of a system's software architecture.

[buschmann2011]

Refactoring

- Fehlerbehebung, Umstrukturierung, Auslagern, ...
- Wartbarkeit erhöhen, Komplexität verringern, …

- Schreitende Verbesserung des Codes
- Strg + a → Entf

Code Refactoring

[stefano13] [buschmann2011]

Refactoring

- Erreichen von nicht funktionalen Anforderungen
- Verhalten nach Außen unverändert
  - Skalierbarkeit – Erweiterbarkeit – Testbarkeit – Robustheit – Performance – …

- Bei Entwicklung
- Bei Wartung
- Reorganisieren von Code in Komponenten und logischen Schichten

Architektur Refactoring

[stefano13] [buschmann2011] [barbar14]

Scrum und Architektur – Die Balance finden – Agile Manifesto

The best architectures, requirements, and designs emerge from self-organizing teams.

Simplicity--the art of maximizing the amount of work not done--is essential.

Continuous attention to technical excellence and good design enhances agility.

Working software is the primary measure of progress.

[1]

[fowler01]

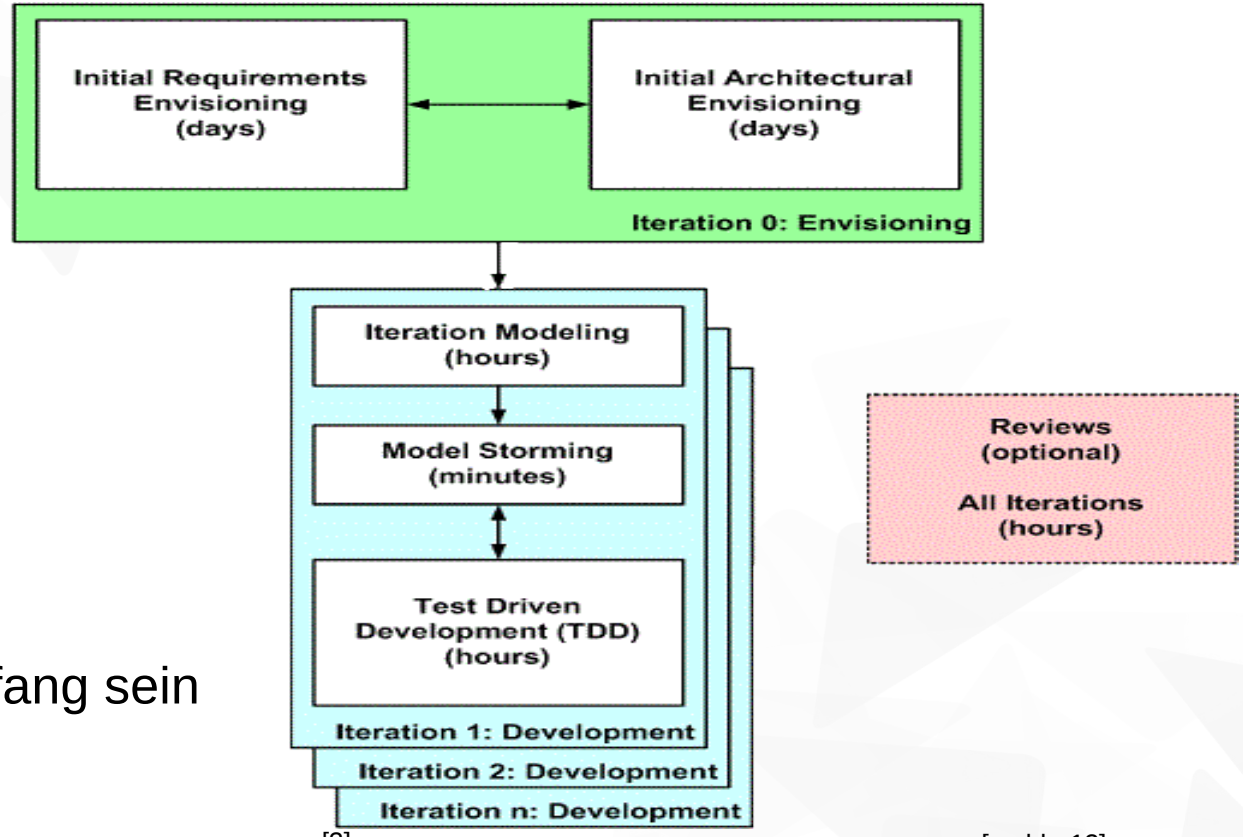
Wie kann man dem vorbeugen?

Agile Model Driven Development - Lifecycle

- Modellierung und Agilität vereinen
- Generierung von Code

- Einmalige initiale Phase

- Wiederholende Phase während Entwicklung
  - Muss nicht vollständig am Anfang sein



[2]

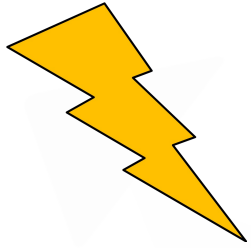
[ambler12]

Agile Model Driven Development

- Agile: Lauffähiger Code im Mittelpunkt
- MDD: Model im Mittelpunkt

- Generierung nur teilweise möglich
- Vernachlässigung von:
  - Wiederverwendbarkeit – Portierbarkeit – Wartbarkeit
  - Manueller Code zu Modell schwierig

Schnelligkeit



Modellbasiertheit

[matinnejad11] [zhang11]

Planung – So sollte es nicht laufen!

- Verwaltungssoftware für Krankenhäuser in den Niederlanden
- Wiederverwendbare Architektur
- Architekten und Management fällen Entscheidungen
- Entwicklern Feedback wird nicht angenommen

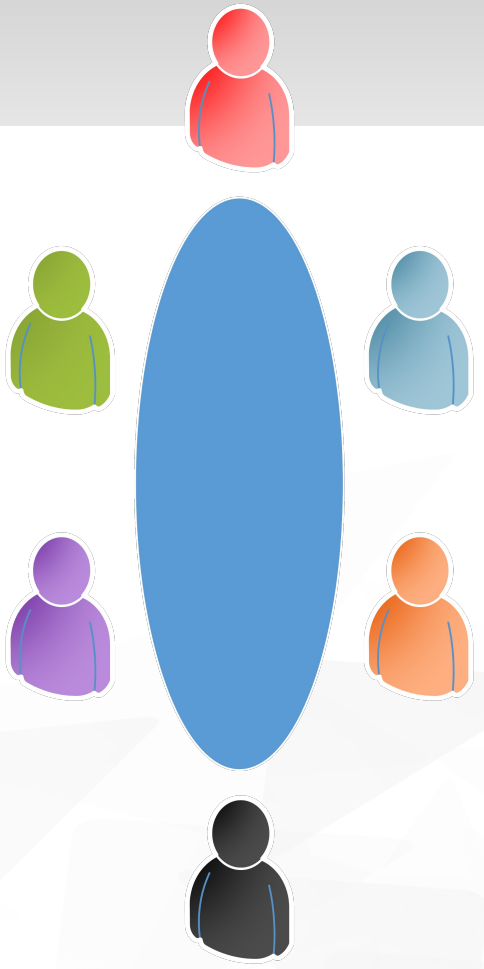
- Projektteam übernimmt Verantwortlichkeit
- Ignorieren vorherige Entscheidungen
- Softwarearchitekt wird informiert

[5]

[barbar14]

Sprint Planning – Wann? – Wie?

- Nur unmittelbar notwendige Entscheidungen treffen
- Eigenschaften priorisieren
- Stakeholder mit einbeziehen
- Dokumentation mit Begründung wichtig
- Entscheidungen schnell verifizieren
- Durch Externe begutachten lassen



[miychi11] [harrison11]

Wo ist dabei die Architekturplanung vorgesehen?

Rolle eines Softwarearchitekten

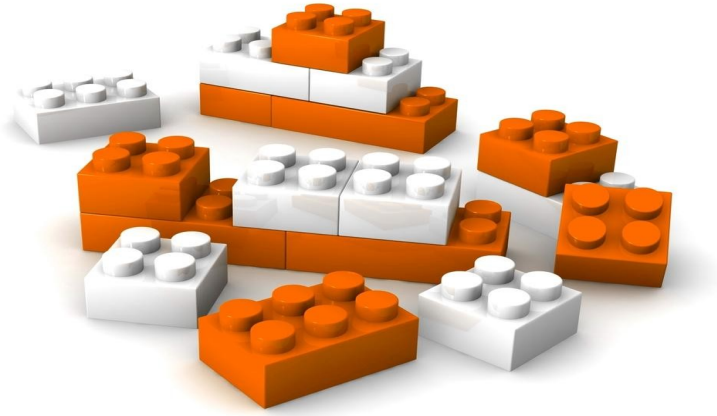
- Architektur Besitzer
- Normales Teammitglied
- Während gesamtem Projekt beteiligt
- Technischer Hintergrund mit Businesswissen

[4]

[ambler12] [barbar14]

Rolle eines Softwarearchitekten

- Aufgaben:
  - Initiale Anforderungsaufnahme und Architekturdesignung leiten
  - Erstellung und Evolution anleiten
  - Beraten und Coachen
  - Hilfestellung bei Einhaltung der Firmenrichtlinien
  - Maximal technische Einfachheit sicherstellen

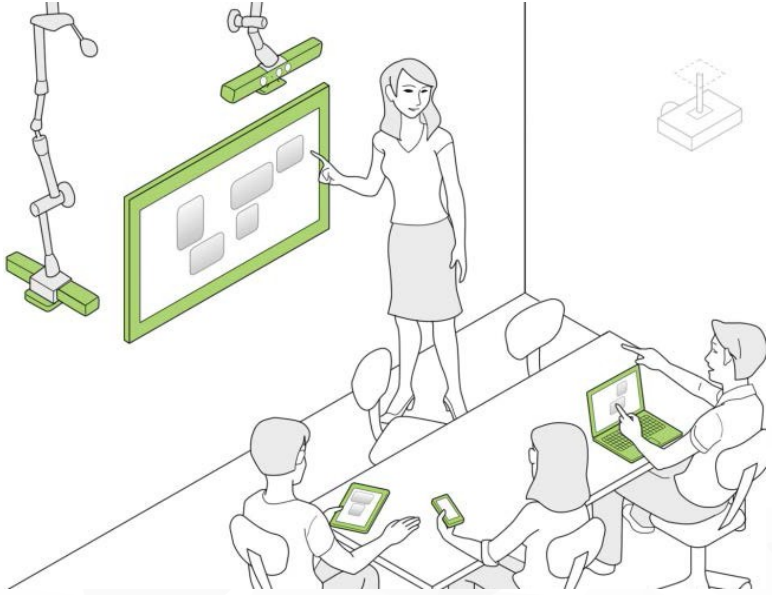


[3]

Technische Unterstützung im Meeting

- Interaktion mithilfe eines digitalen Taskboards

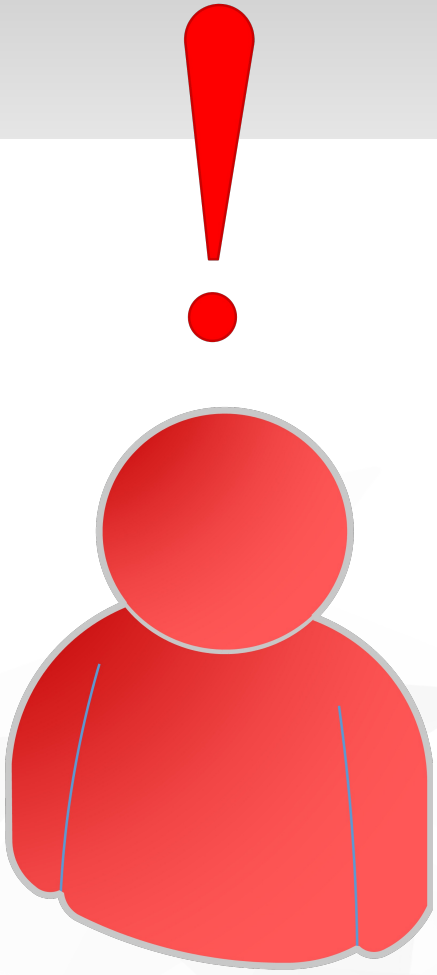
Design Principles
Principle 1 Everyone can interact with the shared display, from anywhere in the meeting space, with any device they bring.
Principle 2 Interactions should be socially acceptable and should not cause embarrassment or distraction.
Principle 3 Each modality should have a separable use.
Principle 4 Interactions should seamlessly span modalities and devices, forming *cross-device gestures.*
Principle 5 Interactions should be manifest to participants to create awareness of actions.
Principle 6 Cross-device interactions should use simple grammars to reduce the potential for error and learning hurdles.



[bragdon11]

Fazit

- Vorhandene Lösungungen nicht ausgereift
- nicht zu detailliert planen
- nicht zu weit vorraus planen
- Technische Unterstützung im Meeting hilfreich
- Regelmäßig Reviews
  - frühzeitig Refaktorisieren



Angestrebte Ziele

- Seminar
  - Lösungsmöglichkeiten verfeinern
  - Vorgehen zur Planung innerhalb des Meetings finden
  - Möglichkeiten zur technischen Unterstützung innerhalb des Sprint Planning Meetings erarbeiten
- Projekt
  - Praktischer Testlauf der erarbeiteten Lösung
  - Prototypische Realisierung der technischen Unterstützung
    - Evaluation im Unternehmen

Quellen

- [fowler01] Fowler, M., & Highsmith, J. (2001). The agile manifesto. Software Development, 9, 28–35. doi:10.1177/004057368303900411
- [abrahamsson10] Abrahamsson, P., Babar, M. A., & Kruchten, P. (2010). Agility and Architecture: Can They Coexist? IEEE Software, 27(2), 16–22. doi:10.1109/MS.2010.36
- [hadar12] Hadar, I., & Sherman, S. (2012). Agile vs. plan-driven perceptions of software architecture. In 2012 5th International Workshop on Co-operative and Human Aspects of Software Engineering, CHASE 2012 - Proceedings (pp. 50–55). doi:10.1109/CHASE.2012.6223022
- [stefano13] Di Stefano, M. (2013). Code Refactoring vs Architecture Refactoring | Refactoring Ideas, http://www.refactoringideas.com/code-refactoring-versus-architecture-refactoring/, Abgerufen 28.11.2014
- [buschmann2011] Buschmann, F. (2011). Gardening Your Architecture, Part 1: Refactoring. IEEE Software, 28(4), 92–94. doi:10.1109/MS.2011.76
- [ihme05] Ihme, T., & Abrahamsson, P. (2005). Agile architecting: The use of architectural patterns in mobile java applications. In International Journal of Agile Manufacturing (Vol. 8, pp. 97–112)
- [matinnejad11] Matinnejad, R. (2011). Agile Model Driven Development: An Intelligent Compromise. In 2011 Ninth International Conference on Software Engineering Research, Management and Applications (pp. 197–202). IEEE. doi:10.1109/SERA.2011.17
- [bragdon11] Bragdon, A., & DeLine, R. (2011). Code space: touch+ air gesture hybrid interactions for supporting developer meetings. In Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (pp. 212–221). doi:10.1145/2076354.2076393
- [ambler12] Ambler, S. W. (2012). Agile Model Driven Development The Key to Scaling Agile Software Development, http://www.agilemodeling.com/essays/amdd.htm, Abgerufen 29.11.2014
- [zhang11] Zhang, Y., & Patel, S. (2011). Agile model-driven development in practice. IEEE Software, 28, 84–91. doi:10.1109/MS.2010.85 #
- [keuler12] Keuler, T., Wagner, S., & Winkler, B. (2012). Architecture-aware Programming in Agile Environments. In 2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture (pp. 229–233). IEEE. doi:10.1109/WICSA-ECSA.212.35
- [miychi11] Miyachi, C. (2011). Agile software architecture. ACM SIGSOFT Software Engineering Notes, 36(2), 1. doi:10.1145/1943371.1943388
- [harrison11] Harrison, N., & Avgeriou, P. (2011). Pattern-Based Architecture Reviews. IEEE Software, 28(6), 66–71. doi:10.1109/MS.2010.156
- [ambler12] Ambler, S. W., & Lines, M. (2012). Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise. IBM Press (pp. 1–2)., http://books.google.com/books?hl=en&lr=&id=CwvBEKsCY2gC&oi=fnd&pg=PR11&dq=Disciplined+Agile+Delivery+:+A+Practitioner's+Guide+to+Agile+Software+Delivery+in+the+Enterprise&ots=lE_3XkN3JK&sig=1rgbs43vu5W6hzP-zayyM0VybaU, Abgerufen 30.11.14
- [barbar14] Barbar, M.A., & Brown A.W., & Mistrik, I. (2014). Agile Software Architecture. Aligning Agile Processes and Software Architectures, ISBN: 978-0-12-407772-0, Elsevier

Bildquellen

- [1] http://upload.wikimedia.org/wikipedia/commons/thumb/4/4c/Simple_balance_scales-01.jpg/1280px-Simple_balance_scales-01.jpg, Abgeufen 28.11.14
- [2] http://agilemodeling.com/images/AMDD.gif, Abgerufen 29.11.14
- [3] http://www.holonsolutions.com/wp-content/uploads/2013/06/blocks.jpg. Abgerufen 01.12.14
- [4] http://www.questteam.com/classroom/images/gear_team_420.png, Abgerufen 08.12.14
- [5] http://trocanada.com/wp-content/uploads/2014/05/Planning.jpg, Abgerufen 09.12.14

Danke für Ihre Aufmerksamkeit!

- Fragen?
- Anregungen?
- Kritik?