Practical Software Engineering

Agile Methods

FH JOANNEUM
Internettechnik
Software Design
WS 2014/15

Outline

Agile Methods

- Agile Software Development
  - Manifesto for Agile Software Development
  - Flavors of Agile Development
  - Agile Software Development Techniques
  - Agile & Contracts

A look at history…

The “beginning”:

- 60ties From “Hacking” to “Concurrency“
- 70ties “Structured methods” (Waterfall)
- Late 80ties RAD, CMM, Spiral, OO Methods

Paradigm shift 1:

“Software development is like production”

- 90ties Heavyweight Methods (e.g. RUP)

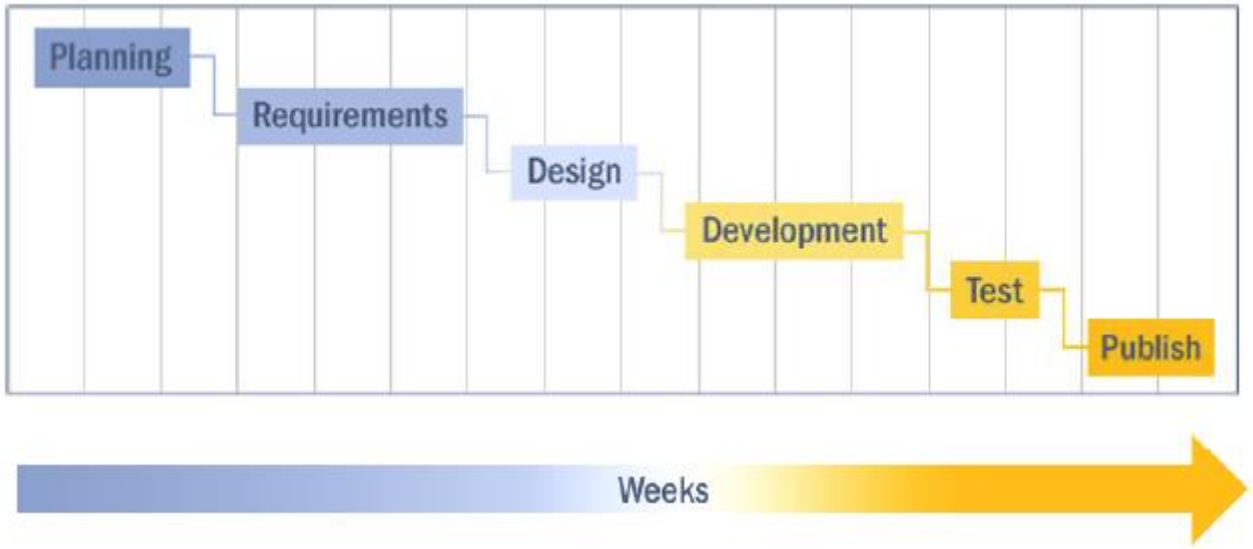
Paradigm shift 2:

“Software development is like developing new products (all at once)”

- Millenium change: Leightweight methods / Agile

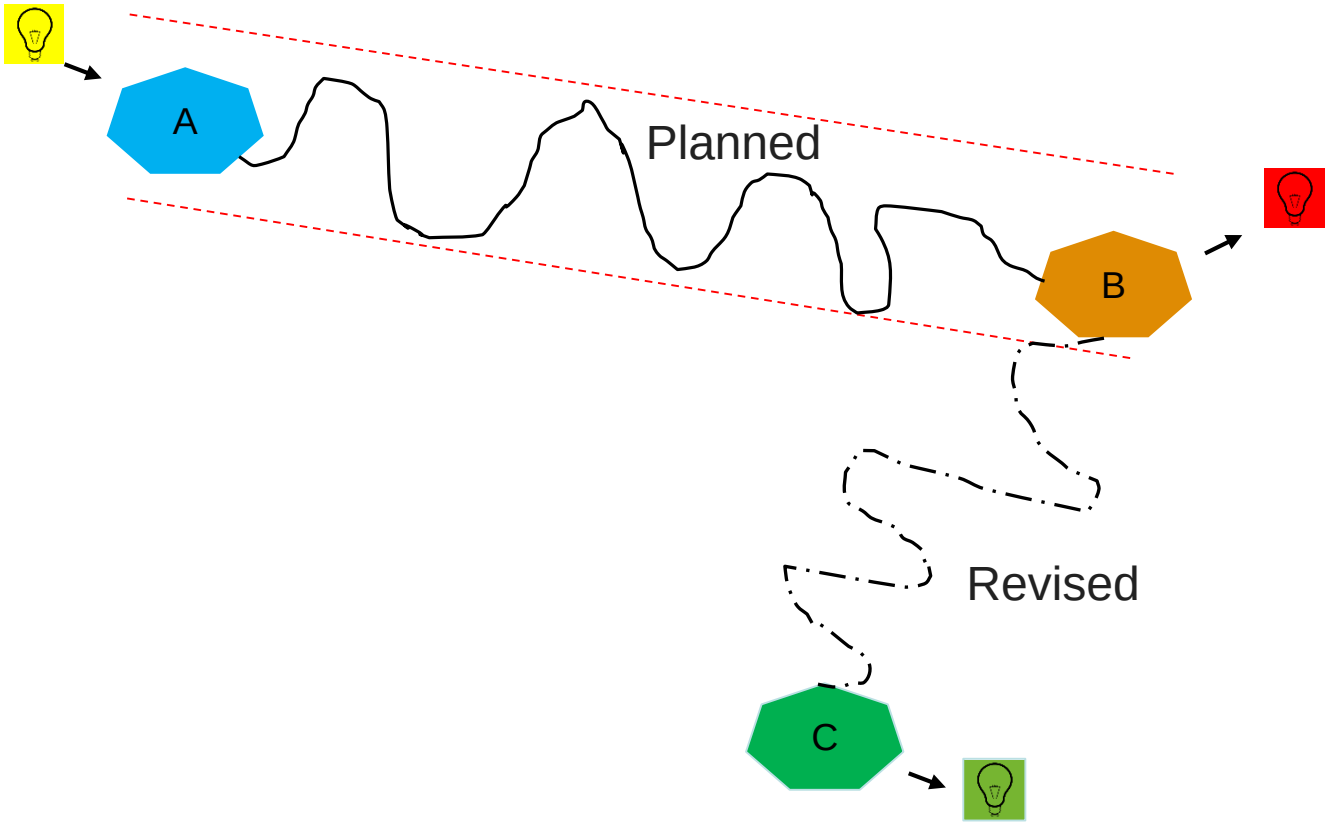
The waterfall model

- A “plan driven” approach

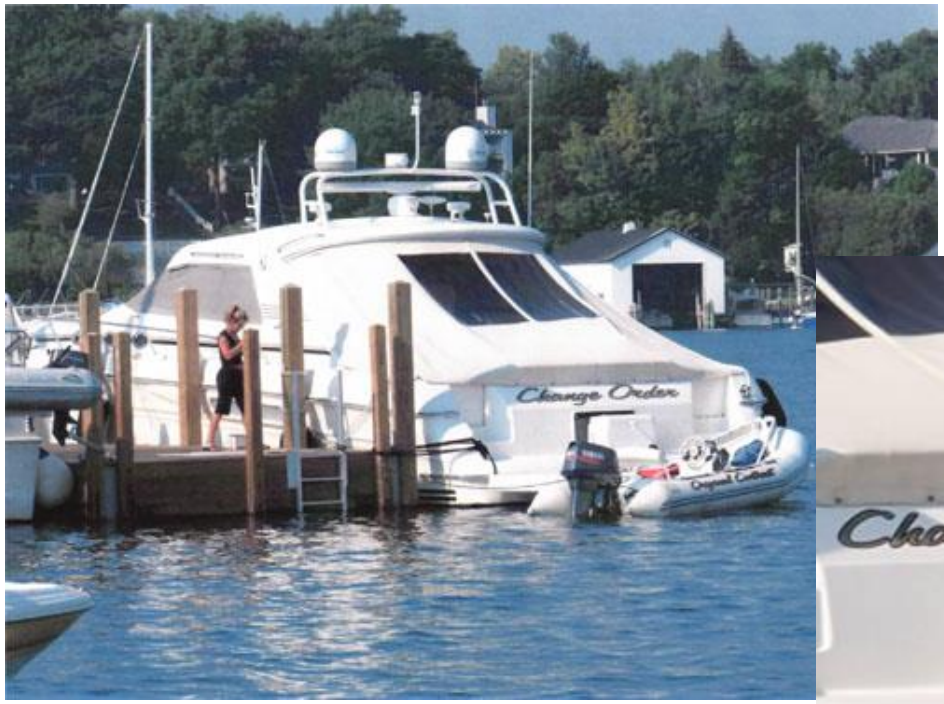


© by Mitch Lacey (taken from the „SCRUM for Managers.pdf“, Link: http://mitch.lacey.com)

A typical Software Project…



… the end



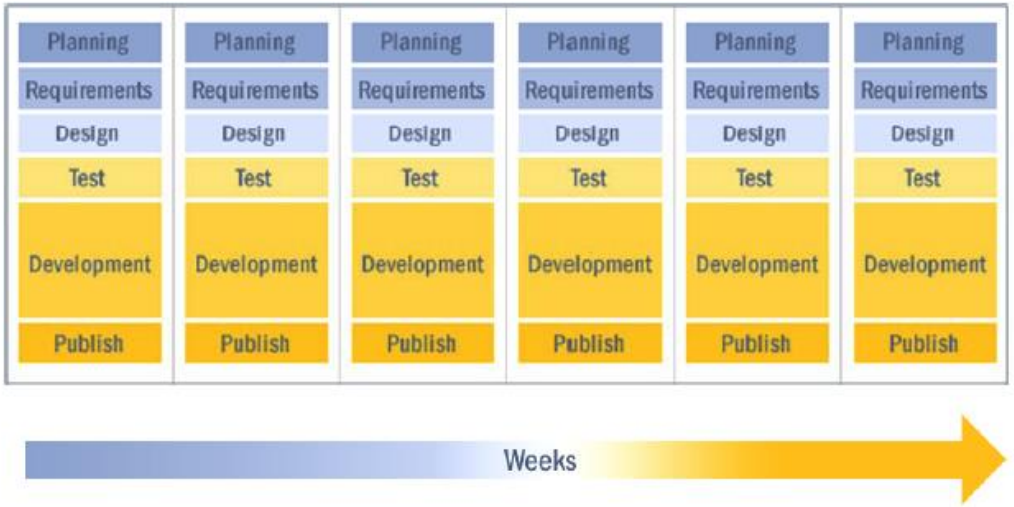
Why?

- Things Change…
  - Customers often do not exactly know what they want or need at the beginning of a project
- Late feedback
  - Customers
  - Design vs. Implementation
  - Long development phases
  - Quality (test phase)
  - Time to market
  - Budget

Evolutionary /Iterative Approach

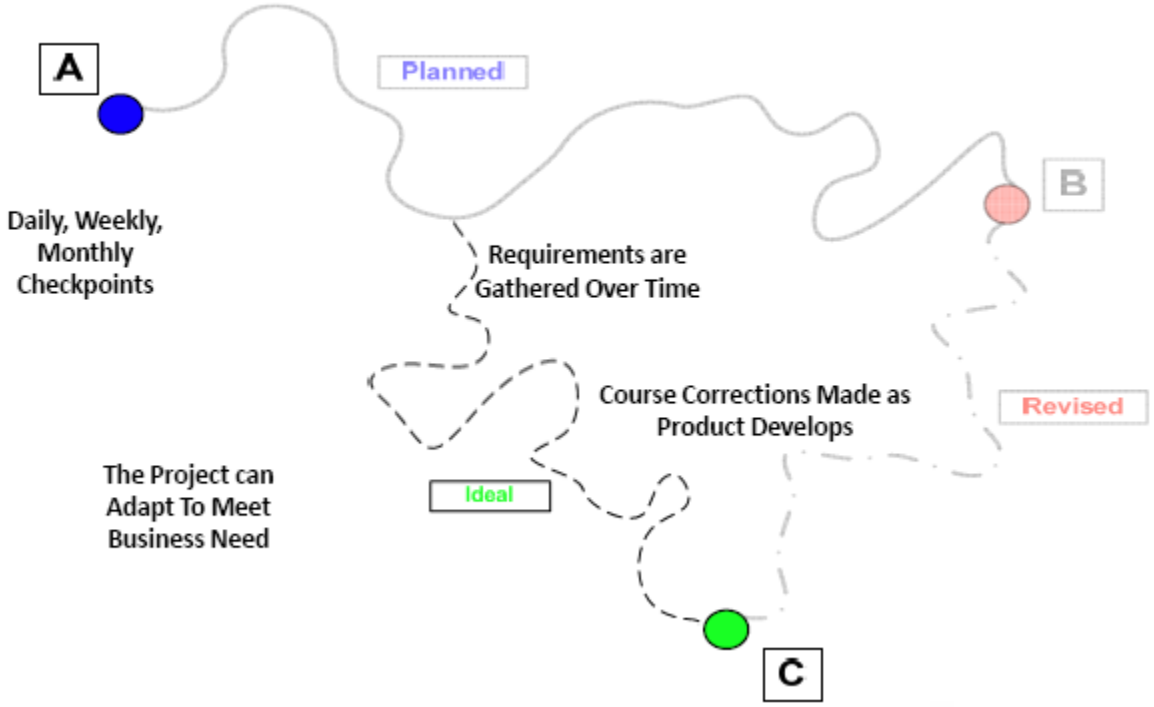
Iterative development breaks down a project by subsets of functionality. Each iteration produces tested, integrated code that is as close to production quality as possible.

„Do a bit of everything in each iteration“



© by Mitch Lacey (taken from the „SCRUM for Managers.pdf“, Link: http://mitch.lacey.com)

Goal of the Evolutionary / Iterative Approach



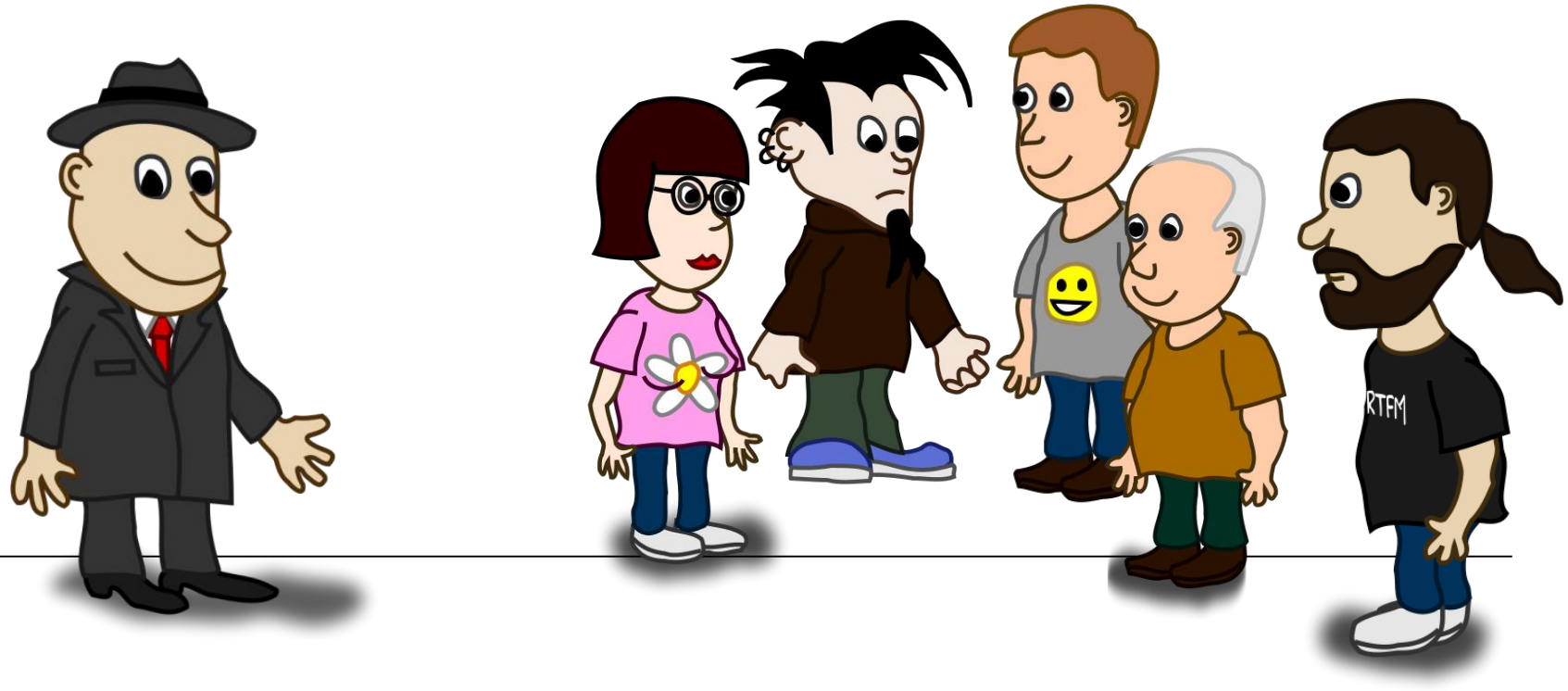
© by Mitch Lacey (taken from the „SCRUM for Managers.pdf“, Link: http://mitch.lacey.com)

DER PERMANENTE ENGPASS

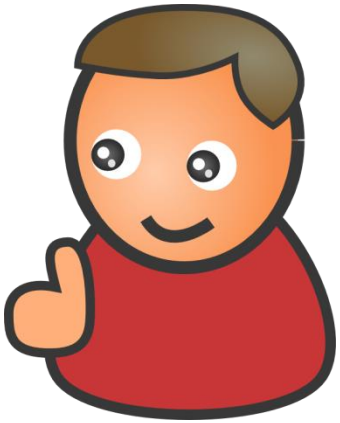
Das Projekt läuft nicht gut…

Ich tue eh schon was ich kann!

Wo sind die nächsten Tasks?



Meeting! Meeting! Meeting!



Ich bin der permanente Engpass!

Noch mehr Meetings…

Noch mehr Auswertungen…

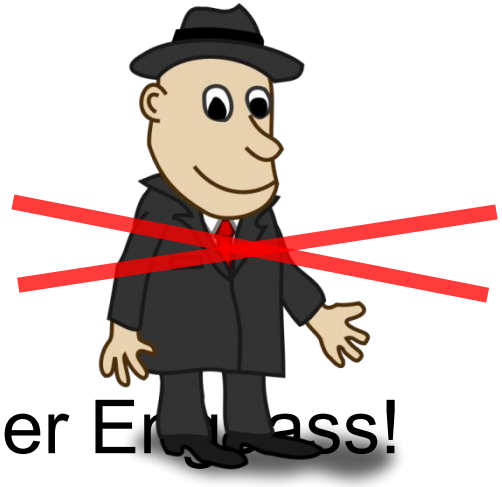
Bekannte Situation?

Task B fertig! Starte mit Task D

Hab mir Task E geschnappt!

Was kann ich tun für Euch?

Er ist weg…



…der Erlass!

Kernaspekte für das Team

- Kollektive Verantwortung
- Selbstorganisation des Einzelnen
- Kontinuierliche Verbesserung
- Effizienz

Agile Software Development

Manifesto for Agile Software Development (2001)

Individuals and interactions over process and tools.
Working software over comprehensive documentation.
Customer collaboration over contract negotiation.
Responding to change over following a plan.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

Manifesto for Agile Software Development

- **Individuals and interactions**.
  - People are the most important ingredient of success.
  - A team of average programmers who communicate well are more likely to succeed than a group of superstars who fail to interact as a team.
  - Building a team is more important than building an environment.

Manifesto for Agile Software Development

- **Working software**.
  - It is always a good idea for the team to write and maintain a rationale and structure document, but that document needs to be short.
  - Too much documentation is worse than too little.
  - Produce no document unless its need is immediate and significant.

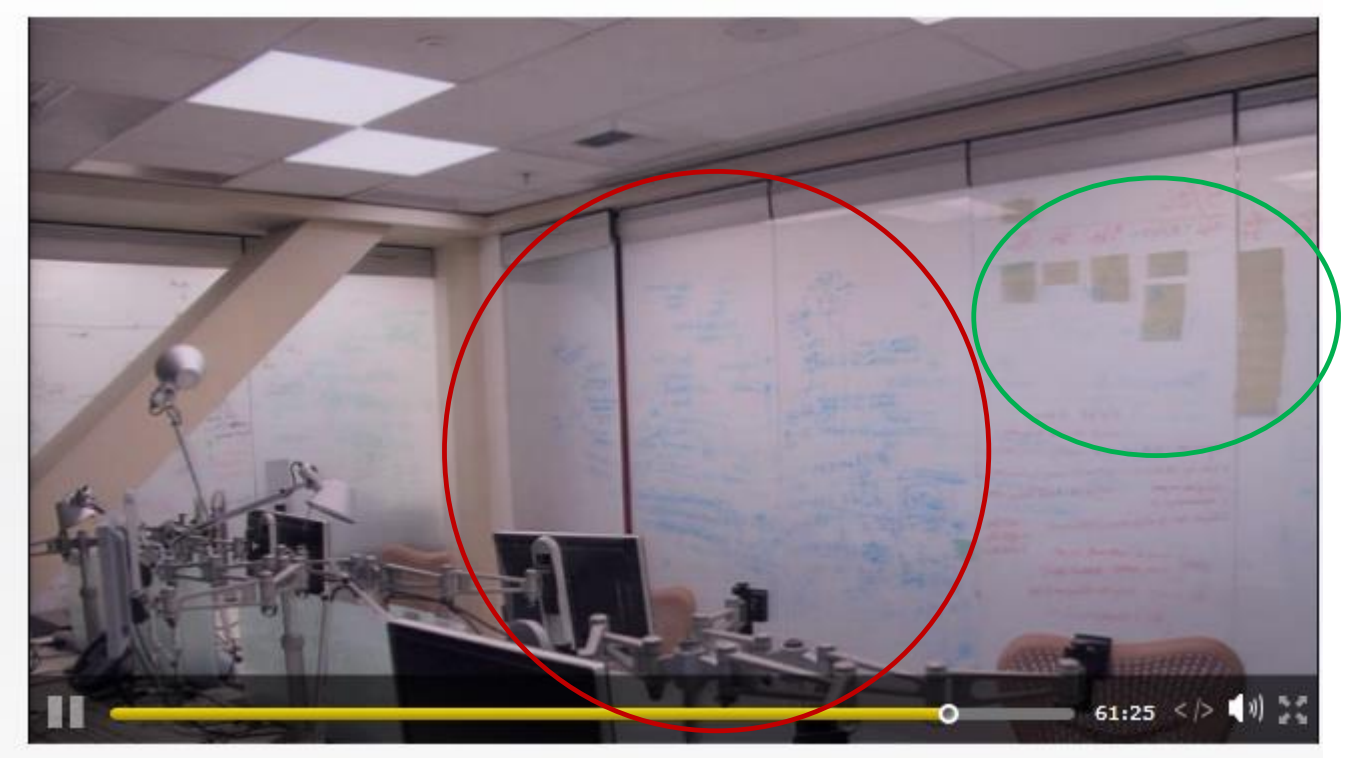
Manifesto for Agile Software Development

- **Customer collaboration**.
  - Successful projects involve customer feedback on a regular and frequent basis.
  - Rather than depending on a contract, the customer of the software works closely with the development team.

Manifesto for Agile Software Development

- **Responding to change**.
  - The business environment is likely to change, causing the software requirements to change.
  - Customers are likely to alter the requirements once they see the system start to function.

„Agile“ is…



Task Tracking
(SCRUM)

Architektur (Doku über Fotos)

… at Microsoft

„Agile“ is not…


WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.
THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.
I'M GLAD IT HAS A NAME.
THAT WAS YOUR TRAINING.
www.dilbert.com
© Scott Adams, Inc./Dist. by UFS, Inc.

Agile is a mind change…

- Impact on teams
  - Evolvement
  - Confidence
  - Self organization
  - Motivation
  - Communication
- Organization
  - Productivity
  - Adaptive
  - Culture (Enterprise 2.0)

Flavors of Agile Development

- **Extreme Programming**
  (Kent Beck)
  Many of XP's practices are old, tried and tested techniques. As well as resurrecting these techniques, XP weaves them into a synergistic whole where each one is reinforced by the others and given purpose by the values.

- **Scrum**
  (Ken Schwaber, Jeff Sutherland, and Mike Beedle)
  Scrum concentrates on the management aspects of software development, dividing development into thirty day iterations (called 'sprints') and applying closer monitoring and control with daily scrum meetings.

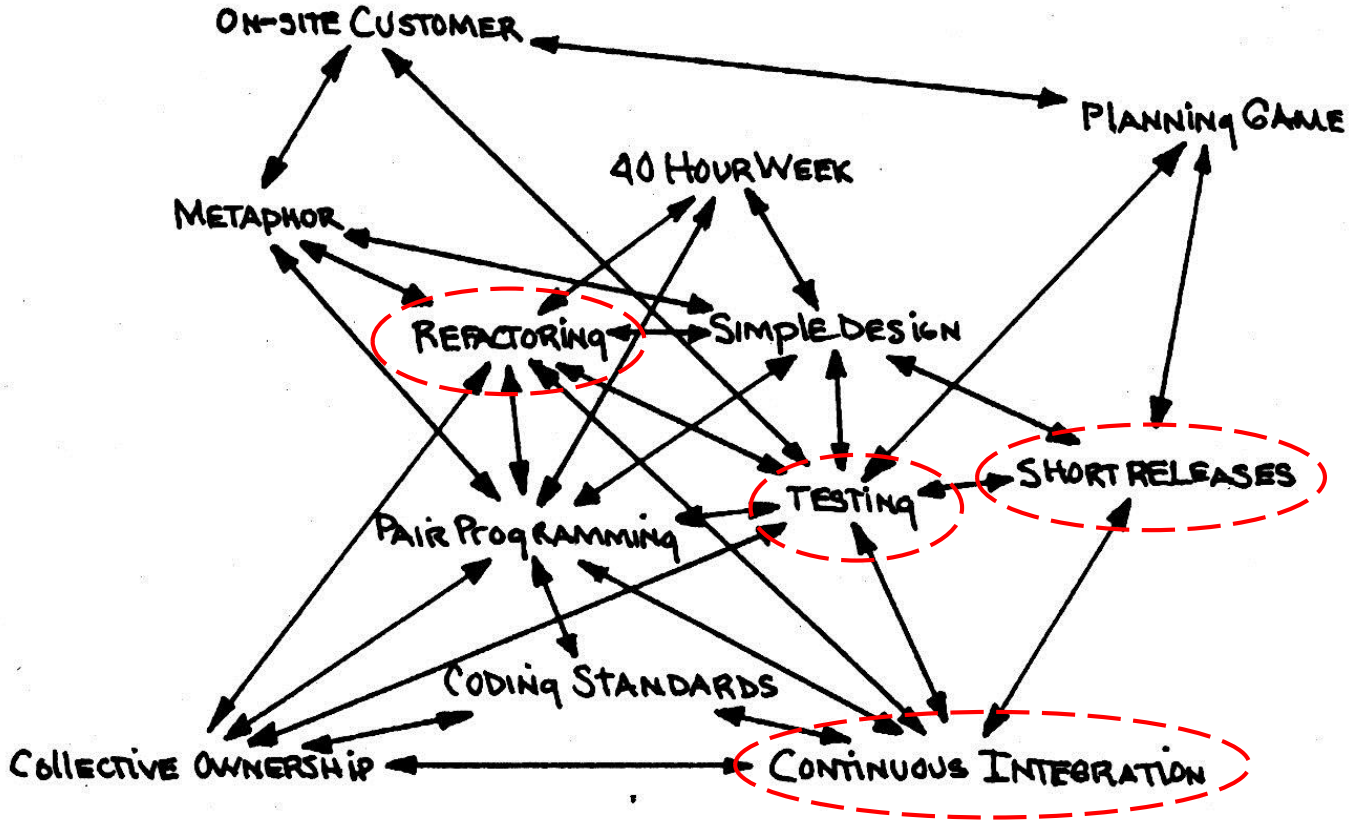
Agile Software Development

Flavors of Agile Development

- **Crystal**
  (Alistair Cockburn)
  (focusses on projects, family of processes with tailoring, similar ideas to agile Manifesto)
- **Context Driven Testing**
  (Brian Marick, Brett Pettichord, James Bach, Cem Karner)
  (main aspect: brings testers in process and not only developers)
- **Lean Development**
  (Mary and Tom Poppendieck)
  (Origin in automotive production [lean processes, flat hierarchies], influenced agile)
- **Kanban**
  (David Anderson)
  (Origin in automotive production, Reduced parallelity, enhanced time to market)
- **Rational Unified Process**
  (Phillippe Kruchten, Craig Larman)
  (large iterative framework, tailoring necessary, use case driven)

Agile Software Development

Extreme Programming (K Beck, W Cunnigham)



Agile Software Development Techniques

- **Automated regression tests** help by allowing you to quickly detect any defects that may have been introduced when you are changing things. A good rule of thumb is that the size of your unit test code should be the same size as your production code.
- **Continuous integration** keeps a team in sync to avoid painful integration cycles. At the heart of this lies a fully automated build process that can be kicked off automatically whenever any member of the team checks code into the code base.

Agile Software Development Techniques

- **Refactoring** is a disciplined technique for changing existing software. Refactoring works by using a series of small behavior-preserving transformations to the code base (incremental design).

- **Iterative development** breaks down a project by subsets of functionality. Each iteration produces tested, integrated code that is as close to production quality as possible.
  A common technique with iterations is to use time boxing. This forces an iteration to be a fixed length of time.

Entwicklung in kurzen Zyklen?

Basisanforderungen

Minimale Demo. „Walking Skeleton“. Nicht für produktiven Einsatz.

Beispiel: Formular mit Pflichtfeldern, keine Validierung

Mächtigkeit und Flexibilität

Variationen in der Anwendung. Zusätzliche Funktionen, Unterfunktionen.

Beispiel: Lookup-Control, optionale Felder

Sicherheit

Sicherheit für Benutzer und Stakeholder.

Beispiele: Eingabevalidierung, zusätzliche Regeln

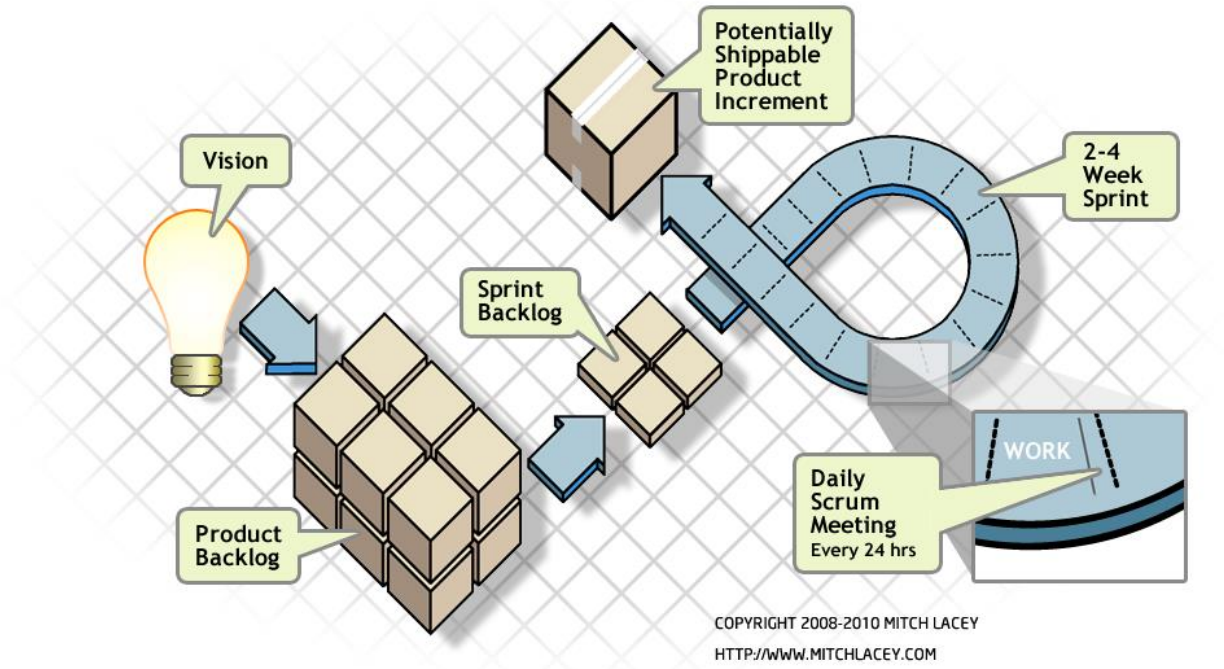
Benutzerbarkeit, Performance

Effzienter, einfacher, attraktiver in der Anwendung.

Beispiel: Auto-Complete, UI-design, Hotkeys

SCRUM

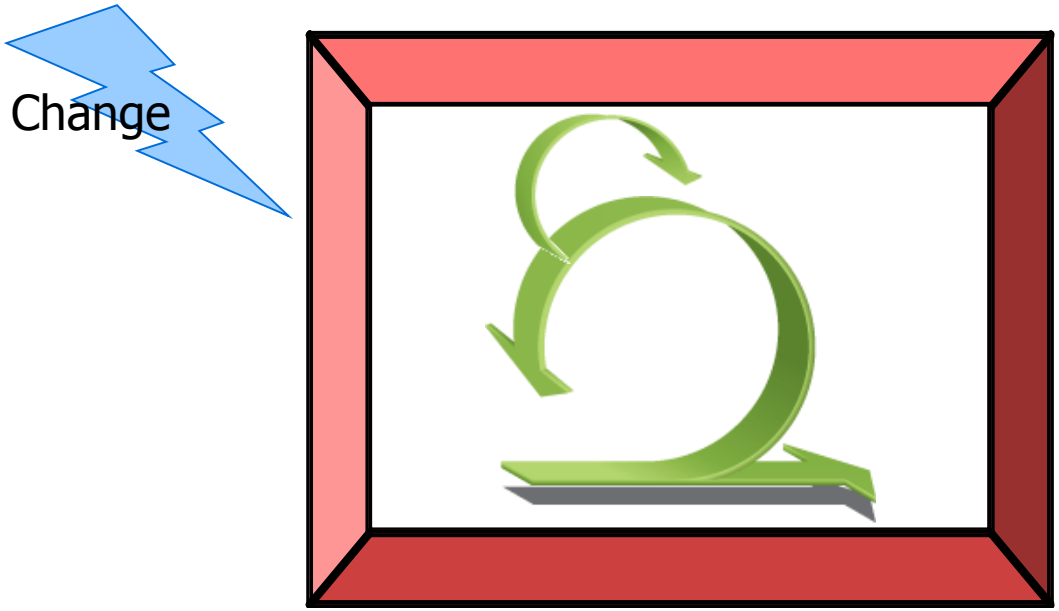
- Focuses on Project Management
- Based on agile Development-Techniques



SCRUM - Sprints

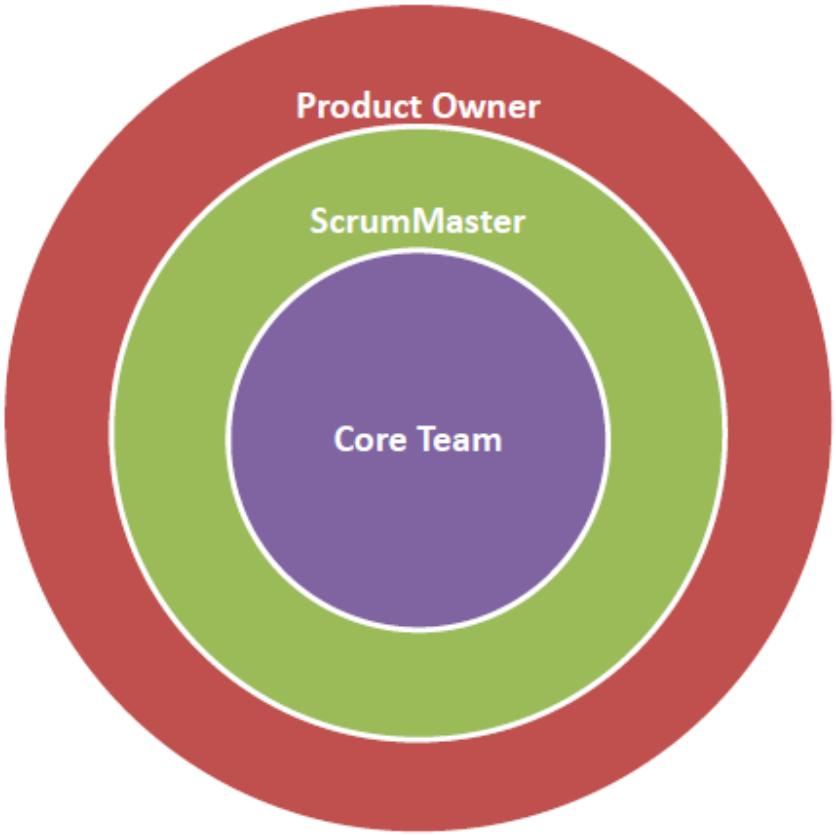
- Scrum projects make progress in a series of “sprints”
  - Analogous to Extreme Programming iterations
- Typical duration is 2–4 weeks or a calendar month at most (Most established: 2 weeks)
- A constant duration leads to a better rhythm
- Product is designed, coded, and tested during the sprint
- **NOGO**: Architecture Sprints etc

No changes during a sprint

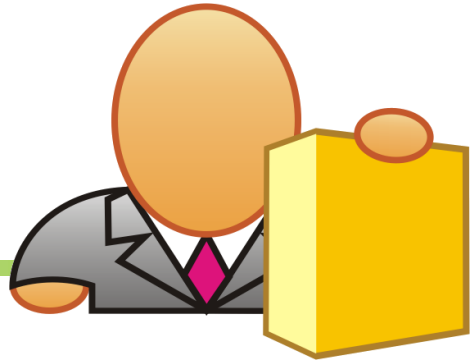


- Plan sprint durations around how long you can commit to keeping change out of the sprint

SCRUM Roles



Product owner

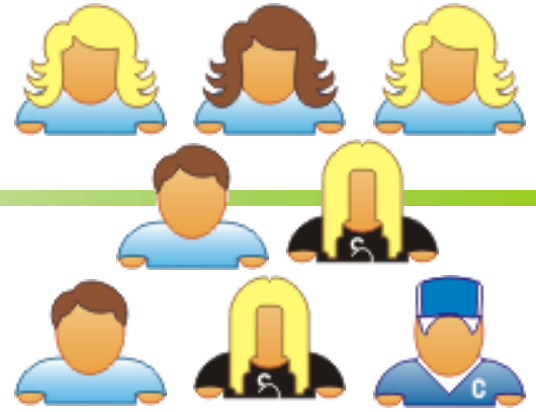


- Define the features of the product
- Decide on release date and content
- Be responsible for the profitability of the product
- Prioritize features according to market value
- Adjust features and priority every iteration, as needed
- Accept or reject work results
- An option: Product Owner Proxy (Inhouse)

The ScrumMaster

- Represents management to the project
- Responsible for enacting Scrum values and practices
- Removes impediments
- Ensure that the team is fully functional and productive
- Enable close cooperation across all roles and functions
- Shield the team from external interferences

The Team



- Typically 5-9 people
- Cross-functional:
  - Programmers, testers, user experience designers, etc.
- Members should be full-time
  - May be exceptions (e.g., database administrator)
- Teams are self-organizing
  - Ideally, no titles but rarely a possibility

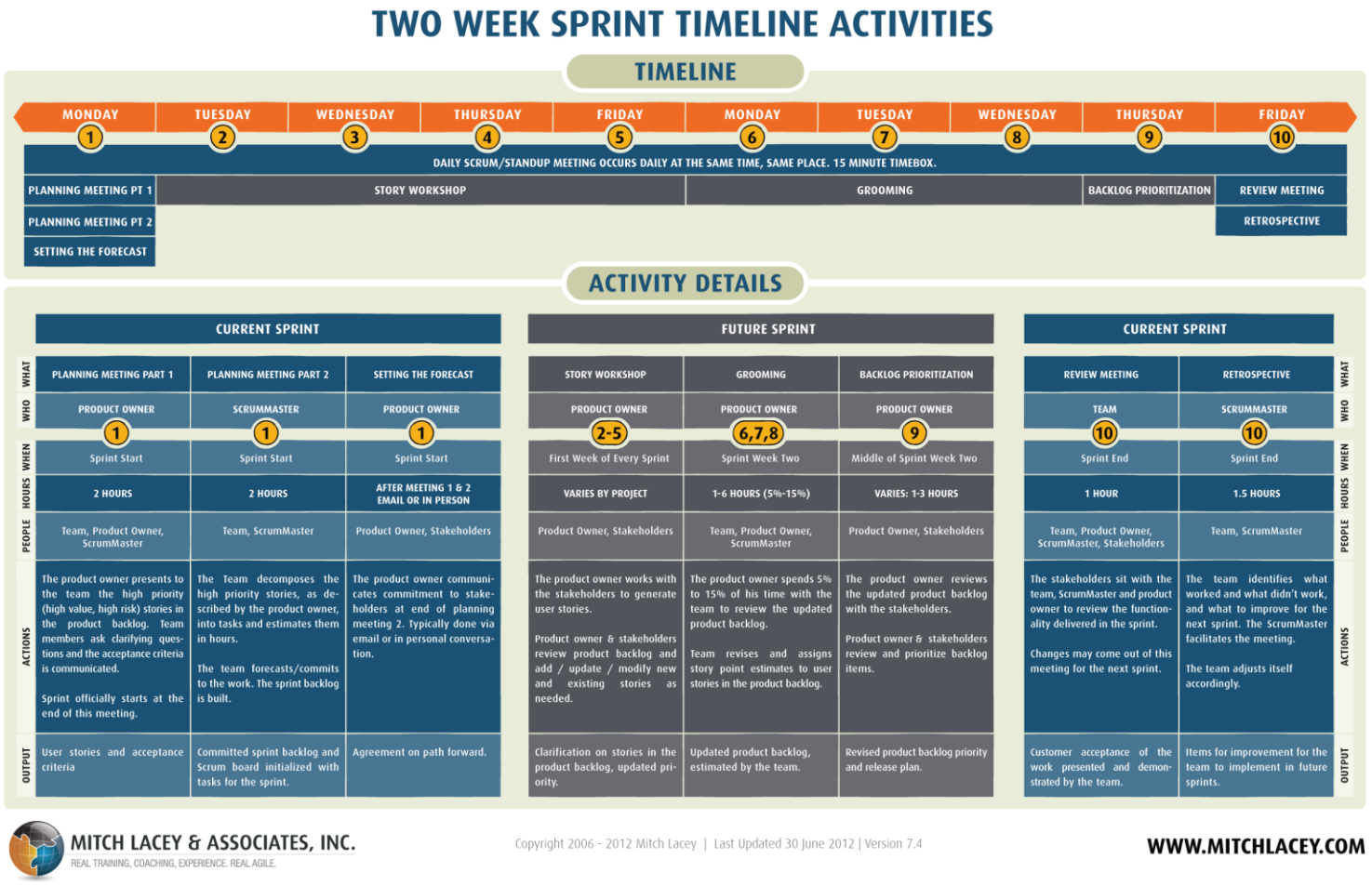
Scrum Meetings

- Sprint Planning
  - Planning, Estimation (by whole Team)
- Daily scrum meeting
  - Stand Up, 15 minutes, not for problem solving
- Sprint Review
  - What was accomplished, demo
- Sprint Retrospective
  - Periodically check what was not working
  - Not at each sprint

Scrum Artifacts

- Product Backlog
  - User Stories
- Sprint Backlog
  - Tasks
- Burn Down Charts
  - Management Reports

A Sprint – More in Detail



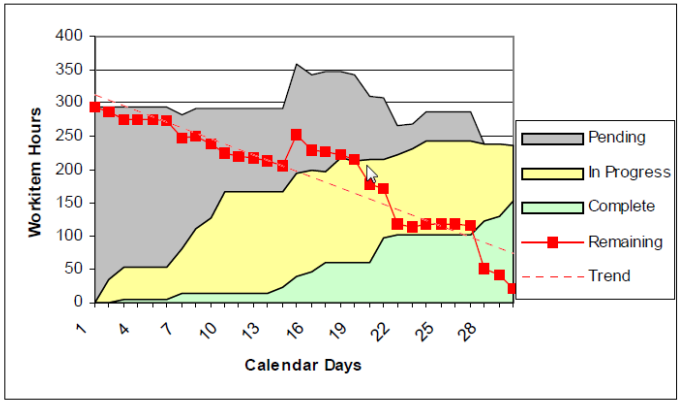
Agile –Tracking / Reporting

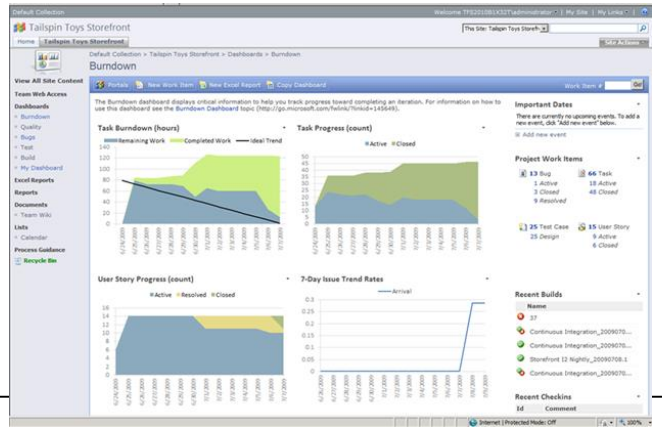
Taskboard (Tracking)

Best Friend: Whiteboard

Tool: Atlassian Grasshopper

Burn Down Chart (Reporting)

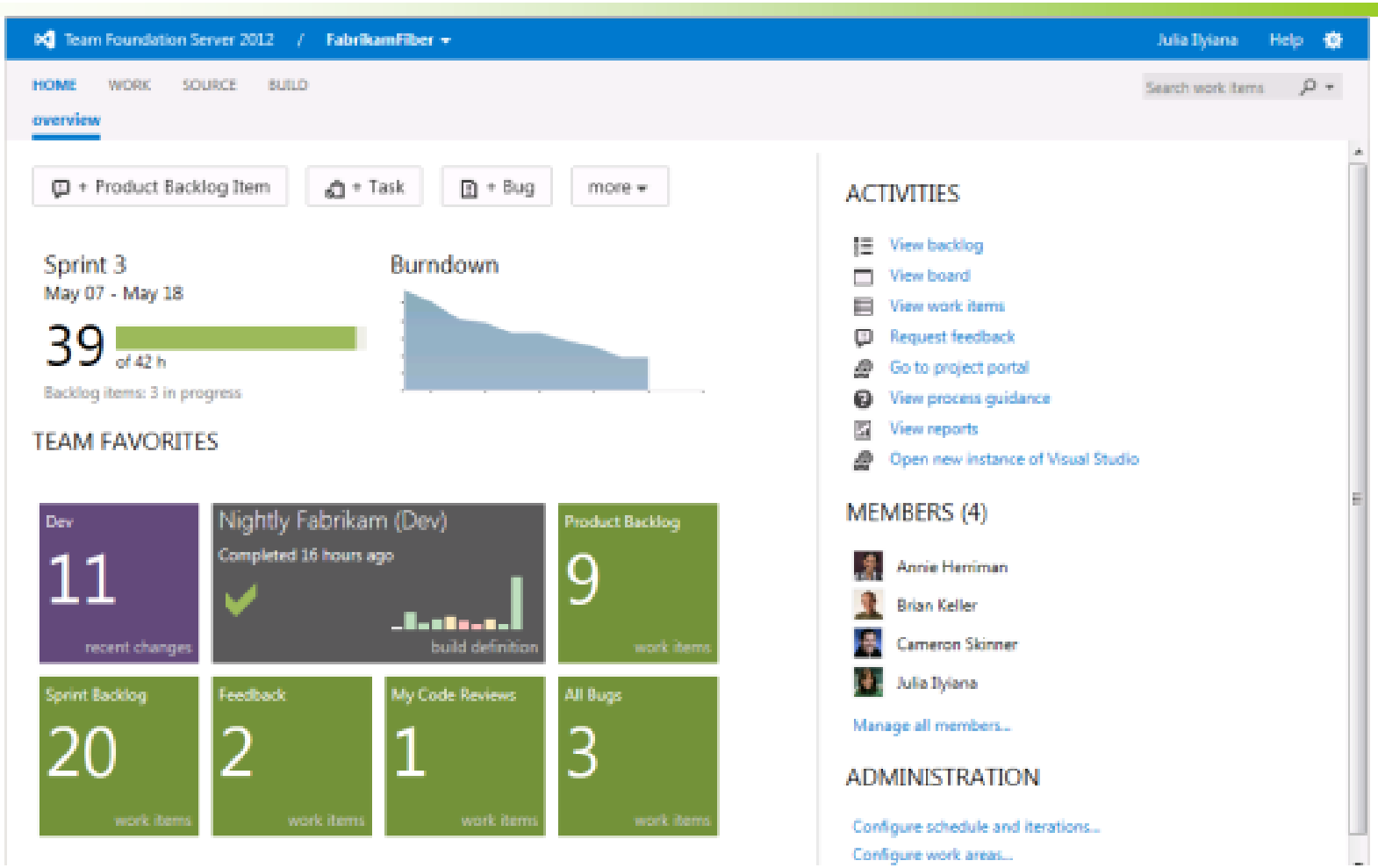




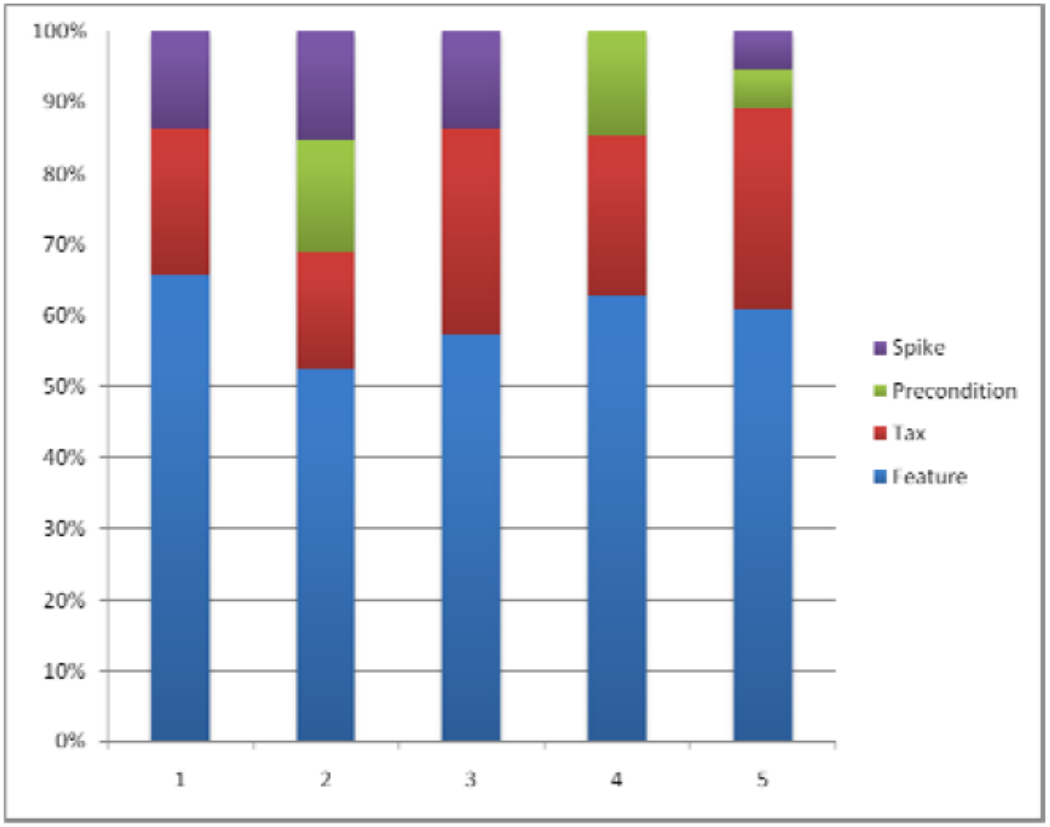
Tool: TFS2010 SCRUM Dashboard

- Simple
- Team velocity (Story Points / Features per iteration)
- Pipeline is full (Teammembers take tasks)

Burn Down Charts – Tool based Example (MS Team Foundation Server)



Reporting – More in Detail



Spike
- Investigation, Prototyping

Precondition
- Setup, CI, Deployment,…

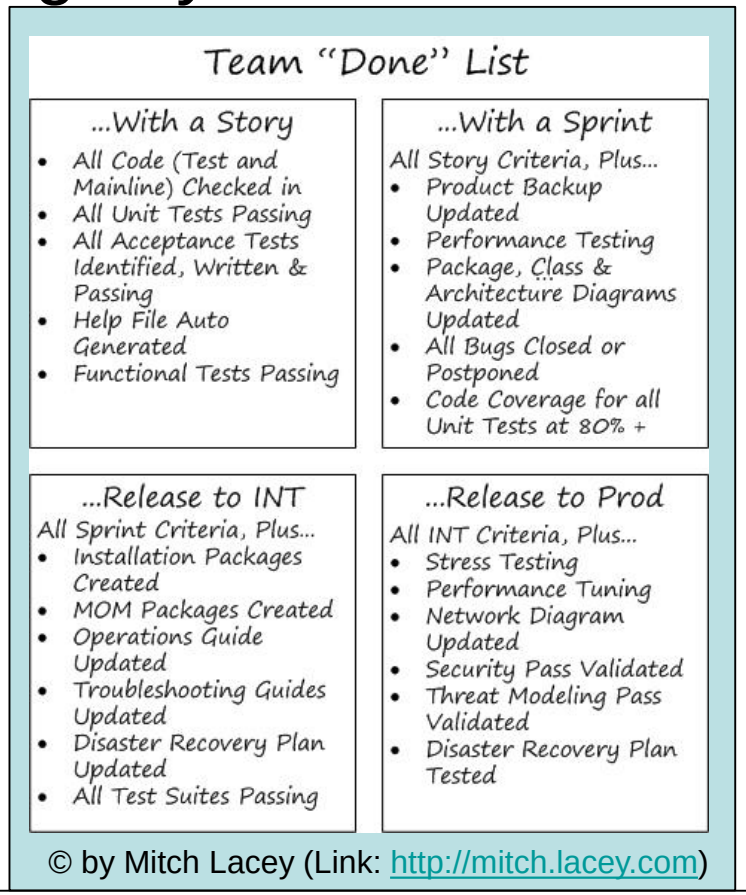
Tax
- Cost in Business (e.g. meetings and other things to be fulfilled within your organization)

Feature
- Customer value driven

© by Mitch Lacey (taken from the „SCRUM for Managers.pdf“, Link: http://mitch.lacey.com)

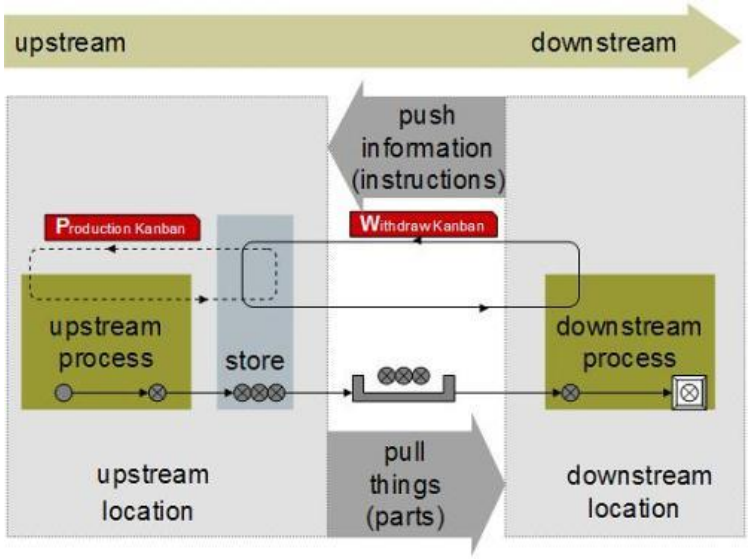
Digression: When are you done?

- Done is **NOT** after checking in your code!
- A sample „Done“ List taken from SCRUM
  (adopt it if necessary)



Kanban – the principle

Toyota Production System (Principle)



The downstream process pulls parts when needed, and provides production instructions, by exchanging Kanbans at the "store"

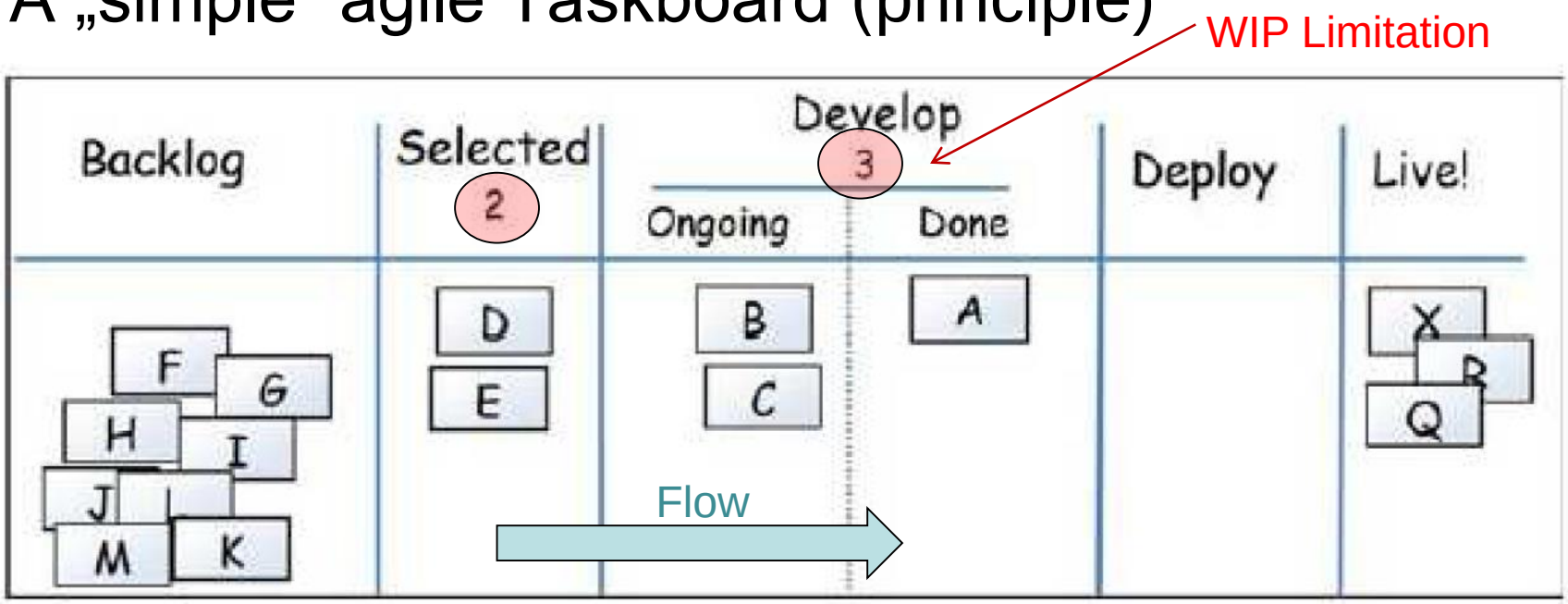
- Upstream produces parts to store
- Downstream pulls parts from store and finishes them
- Downstream pushes „needs“

Key ideas

- Flow principle (store is buffer)
- Limit Work in progress
- Optimize cycle time

Kanban in a SWE-Process

A „simple“ agile Taskboard (principle)



Kanban does not define the workflow itself
(even waterfall would be possible for each item)

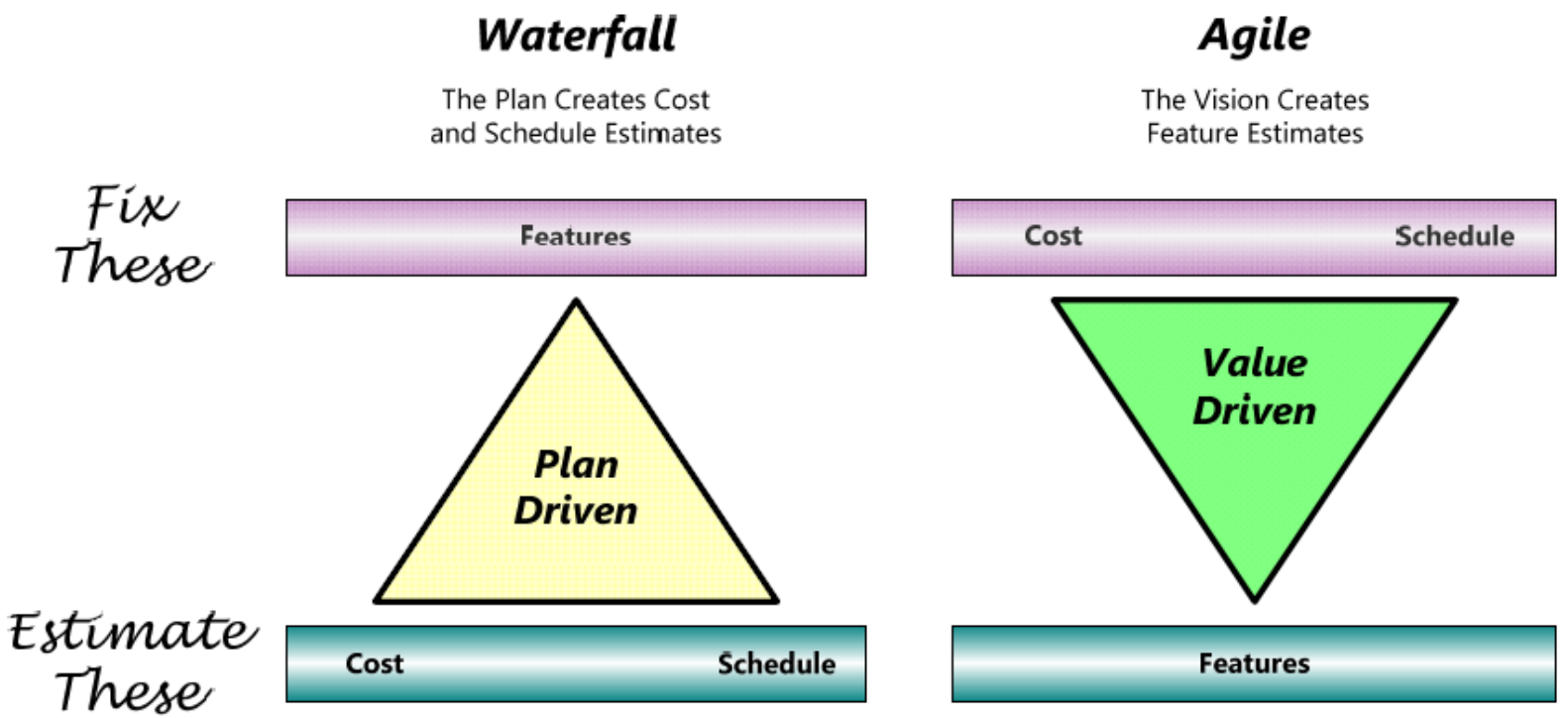
Scrum vs. Kanban - Differences

Scrum	Kanban
Iterationen mit festem Zeitschema vorgeschrieben.	**Iterationen mit festem Zeitschema sind optional.** Kann verschiedene Rhythmen (Kadenzen) für Planung, Release und Prozessverbesserung haben Kann ereignisgetrieben sein statt zeitgetrieben
Team verpflichtet sich zu einer bestimmten Menge an Abarbeitung für diese Iteration	**Verpflichtung optional**
Setzt **Geschwindigkeit** als Standardmetrik für Planung und Prozessverbesserung ein	Setzt **Durchlaufzeit** als Standardmetrik für Planung und Prozessverbesserung ein
Funktionsübergreifende Teams vorgeschrieben	Funktionsübergreifende Teams optional. **Spezialisierte Teams erlaubt.**
Arbeitspakete müssen aufgeteilt werden, so dass sie innerhalb eines Sprints abgearbeitet werden können.	Keine bestimmte Arbeitspaketgröße vorgeschrieben.
Burndown-Diagramm vorgeschrieben	Kein spezieller Diagrammtyp vorgeschrieben
WIP-Limit indirekt (durch Sprints)	**WIP-Limit direkt** (pro Zustand im Arbeitsablauf)
Schätzen vorgeschrieben	**Schätzen optional**
Kann keine Arbeitspakete in laufenden Iterationen hinzufügen	**Kann neue Arbeitspakete hinzufügen, wann immer Kapazität verfügbar ist**
Ein **Sprintbacklog gehört einem bestimmten Team**	Ein **Kanbanboard kann von mehreren Teams** oder Personen **geteilt werden**
Schreibt 3 Rollen vor (Product-Owner/ Scrummaster/Team)	**Schreibt gar keine Rollen vor**
Ein **Scrumboard wird** für jeden Sprint **neu eingesetzt**	Ein **Kanbanboard bleibt durchgehend bestehen**
Schreibt ein priorisiertes Produktbacklog vor	**Priorisierung ist optional**

Quelle: http://www.infoq.com/resource/news/2010/01/kanban-scrum-minibook/en/resources/KanbanAndScrum-German.pdf

Agile and Contracts

Agile vs. Plan Driven



© by Mitch Lacey (taken from the „SCRUM for Managers.pdf“, Link: http://mitch.lacey.com)

We need to rethink contracts

Agile Contracts must **contain** …

- Costs 
- Timeline 
- Vision and Feature Estimations 

… but must also **permit changes** 

Fixed everything does not work with Agile

Agile vs. Fixed Price Projects(1)

- Fixed price Contracts typically define
  - Fixed set of Features (Scope)
  - Fixed schedule
  - Scope change: is Change Request (extra costs)
- Agile paradigm
  - Scope change is expected, features may change (no extra costs)
  - Fixed schedule
  - budget limitation (i.e. time & material like contract)
  - But: does not fit with classical Fixed price contracts
- Issue: Customers are used to fixed price contracts

Agile vs. Fixed Price Projects(2)

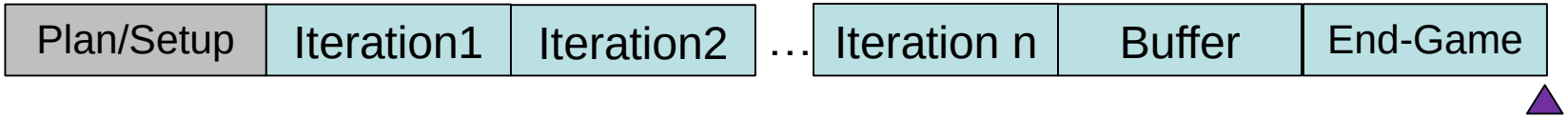
Possible Options

- Convince customer that agile works better (time & material with budget limitation)
  - Not always easy
    - Time & material issue: No warranty, no risk at supplier side
    - Comparison to other suppliers (“price pressure” does not work that easily)
    - Often does not fit to customers policies: Established RfP/RfQ processes
- Fixed price with change volume and feature changes at no cost
  - Define change volumes in contract (feature changes at no extra costs)
  - Define only a subset of fixed features with options to change
  - Customer needs to trust you!
- Develop agile internally in your organization
  - Some people say: don’t to it, I disagree
  - Why: Agile is a mindset and evolves your organization

Agile Fixed Price Contract Model

According to M Lacey / J Sutherland

- Product Backlog (with must have and optional features)
- Time Schedule (Iterations with Buffer)



- Change for Free
  - Define fixed price contract based on
    - # of iterations (internal: include “buffer iterations”)
    - Team velocity (Story Points / Features per Iteration)
    - Changes are included if other features are dropped
- Money for Nothing
  - Customer checks whether features have no more ROI
  - Customer aborts project at 20% of remaining contract volume at the end of an iteration
  - If team is faster, team gets a percentage of remaining contract volume as award for being faster

Summary

- Agile Methods
  - Focus on Teamwork
  - Development in small Iterations (running software)
  - Are based on fast feedback
  - Are a Mind Change
  - Are Effective and Adaptive
  - Even aspects of Agile help!
  - “Issue” with Customers: Contracts

References

- *Martin Fowler*
  "**The New Methodology"**
  http://martinfowler.com/articles, 13 Dec. 2005

- SCRUM (used in slide deck)
  Mike Cohn, Mountain Goat Software
  Mitch Lacey, MitchLacey.com

- Andreas Opelt, Boris Gloger, Wolfgang Pfarl, Ralf Mittermayr
  "Der Agile Festpreis", Sept. 2012