

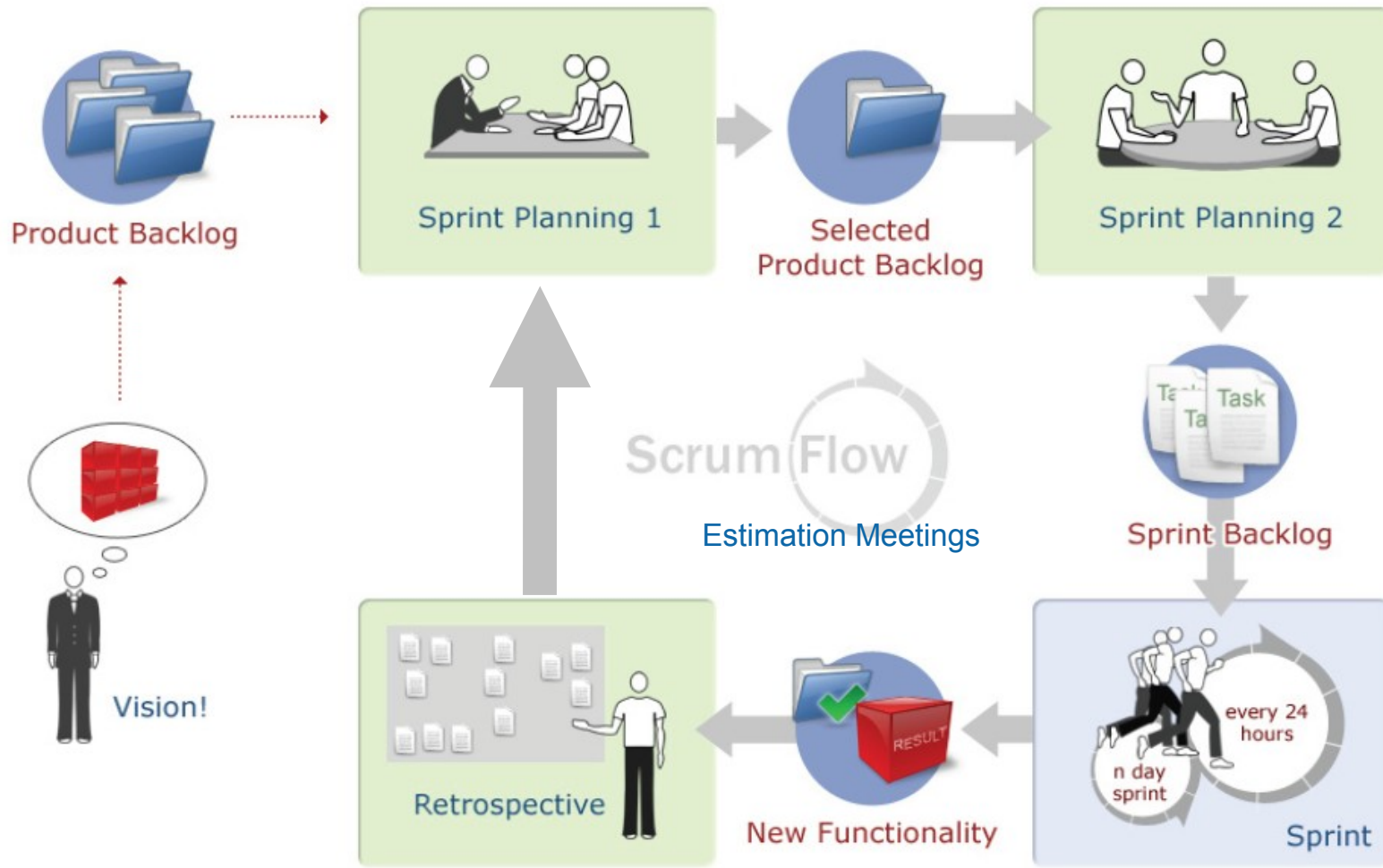
Effektiver Tool-Einsatz

für Scrum-Projekte im Java-Umfeld

Expected Knowledge

- Knowledge of Scrum principles
- General Enterprise Java Know How
- Basic Knowledge of Agile Development

Scrum Flow: Artifacts & Meetings



Background

- Our experience demonstrated here is a concrete example, other ways may also work
 - Startup company
 - 2 week iterations
 - Team size up to 14 (including product owner & scrum master)
- Disclaimer:
 - TNG is an Atlassian Partner (because we love their products)
 - So there is lot of 'Jira' and 'Confluence' 😊



Ingredients

- Wiki
- Issue Tracker
- Version Control Software
- Mind Mapping Software
- Artifact Management Software

Ingredients

- One Room
- Digital Camera
- DIN A6 File Cards
- Development Server
- Planning Poker Cards
- Task Board

Product Backlog

- Freemind, <http://freemind.sourceforge.net>



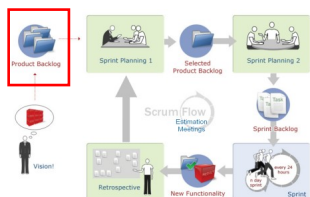
Product Backlog

- Mind map containing crude planning for future sprints
- Upcoming user stories

- Jira, <http://www.atlassian.com/software/jira/>



- Technical Product Backlog
- Bugs, improvements, ideas
- Technical debt (“works for now, but has to be revised”)
- Link from work items to source code

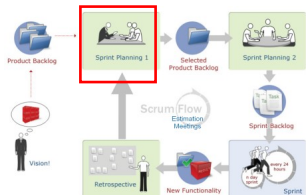


Product Backlog



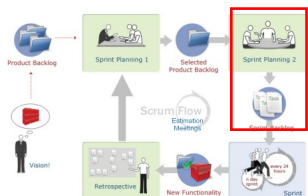
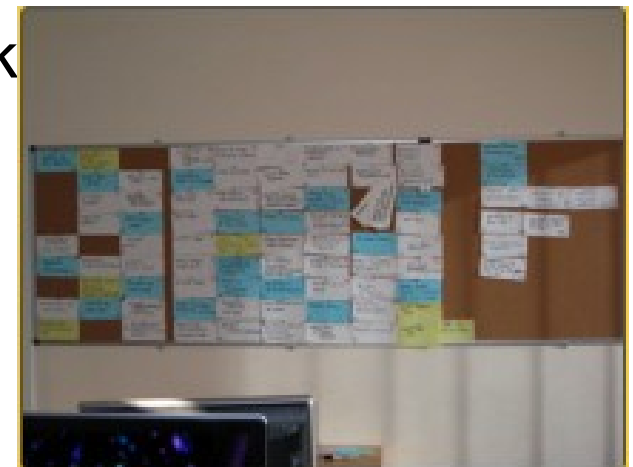
Sprint Planning 1

- Freemind “Online”
- Planning Poker
- Team Commitment
- **Result:** Mindmap containing “selected product backlog” is stored in Confluence
- **Who:** whole team plus Product Owner, about 3h



Sprint Planning 2

- Breakdown of User Stories into Tasks
- Per User Story one blue card
- Per User Story Card multiple white task cards
- Each User Story is one Jira issue of type “User Story”
- Each Task for a User Story is a sub-issue in Jira
- **Result:** Colorful task board, many task
- **Who:** whole team, about 2-3h
- Time effort for Jira: 1-2h (one person)



Selected Product Backlog in Jira



Selected
Product Backlog

Summary

- ☐ Project: [redacted]
- ☐ Components: [User Story](#)
- ☐ Resolutions: Unresolved
- ☐ Sorted by: Priority descending

Operations

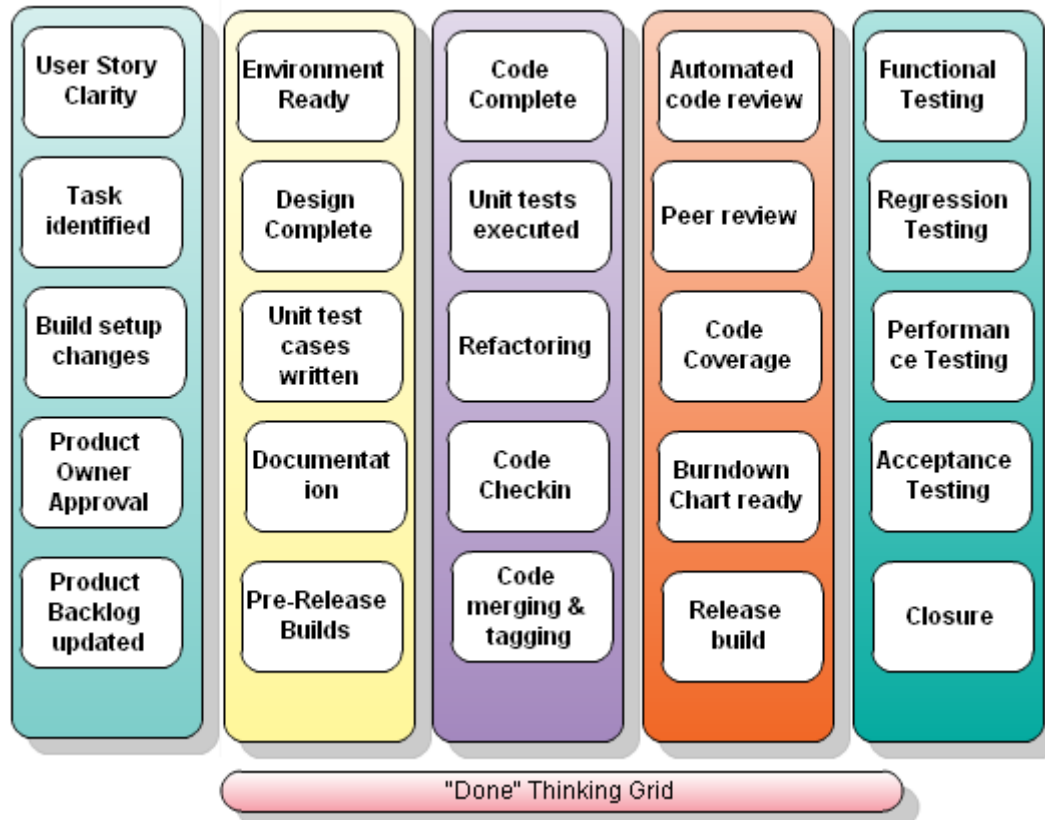
☐ [Save](#)

Browser ([Current Fields](#) | [Printable](#) | [Full Content](#)) | [XML](#) | RSS ([Issues](#) | [Comments](#)) | [Word](#) | Ex

T	Key	Summary
	TT-642	Miscellaneous Tasks Sprint 7
	TT-643	Designer Task Sprint 7
	TT-520	[redacted]
	TT-658	[redacted]
	TT-538	[redacted]
	TT-664	[redacted]
	TT-665	Prepare [redacted] beta version for testing
	TT-666	Finalize installation & upgrade process
	TT-667	Remaining Tasks from Sprints 1 to 6
	TT-644	Fix remaining bugs and tasks from Sprint 6



Definition of Done (DOD) thinking grid



Done may be different for each team

Definition of Done

A user story is done if *all of its identified tasks* are

- implemented
- reviewed
- tested
- documented
- deployed
- making the product owner happy.

The remainder of this page details the parts of "done".

Implemented

Implemented code keeps the following principles:

- adheres to the [coding styleguide](#)
- code is compiled and run against the current version in source control
- code is commented according to the [coding styleguide](#)
- all [FIXME/TODO tags](#) in the code have been attacked (this is checked by a Maven plugin). If they are not resolved yet, either the Product Owner or [\[redacted\]](#) had to agree.

It must also be possible to *automatically build* the implemented code. How this is done depends on the kind of task.

- Java code must be integrated into the central Maven-based build
- Native Linux applications, Firefox plugins etc. must be packageable as APT. The process of packaging must be automated and checked in.
- For development infrastructure, it is acceptable if an installation HOWTO is provided in Confluence.

Reviewed

Every task must be peer reviewed. There is no formal process how this should be done, just make sure that some colleague has a look at the changes you made and incorporate feedback as necessary.

For non-visual design issues, make sure [\[redacted\]](#) was involved somehow - he is in charge of keeping the whole system consistent.

For visual design issues, make sure [\[redacted\]](#) was involved somehow - he is in charge of keeping the whole system consistent.

Tested

Make sure that all implemented changes are thoroughly tested to keep overall quality consistently high.

For Java code this means:

- Write unit tests for all non-super-trivial code.
- Make sure that Cobertura reports a solid test coverage (Maven runs Cobertura for you).
- Write integration tests to automatically test the end-to-end scenario described by the user story (involves database etc.).
- Write Selenium-based test to check correct behavior of Web UIs.

Deployed

Code must be deployed on test machines.

Documented

The documentation must allow a reasonably knowledgeable person to understand the implemented changes. To check this level of documentation, reviews should usually be handed over just via a link to the entry point of the documentation in the Jira comment. Examples:

- Documentation: [http://\[redacted\]confluence/display/\[redacted\]](#)
- Have a look at Javadoc of package com.[redacted] persistence to get an overview of the design.

In addition, you have to clarify the licensing issues whenever you use 3rd party products. See [LicensingInformation](#) for more information.

Making the product owner happy

Ask the [product owner](#) for satisfaction - preferably before the Sprint review meeting.

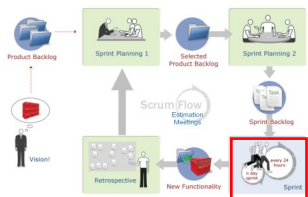
Please note that we are not developing for ourselves or even the product owner, but for end customers. Always keep in mind the personas representing them.



<http://www.atlassian.com/software/confluence/>

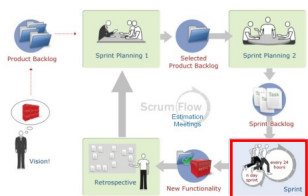
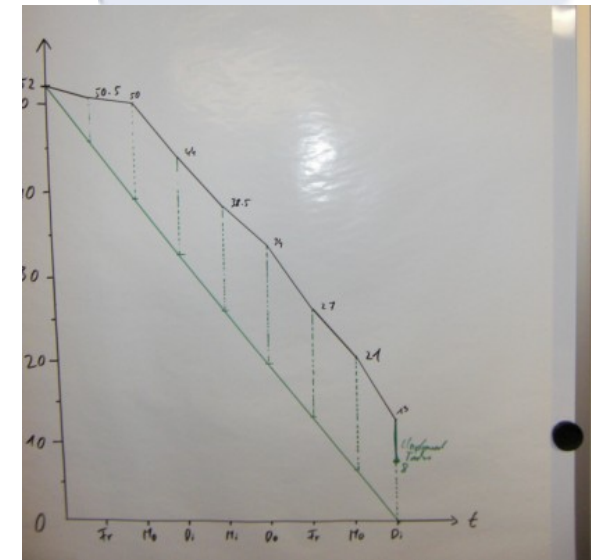
Testing

- It's all about feedback loops!
 - Some source code aspects (platform dependency, package dependencies, ...): AspectJ (included in maven build)
 - Test Framework: TestNG (categories!)
 - Test Types/Categories:
 - `sourcecode`, `unit` (Mockito), `integration`, `performance`, `external`, `ui`, `e2e` (client/server) for staged builds
 - UI Tests: Selenium RC
 - Java Script Tests via Selenium



Daily Scrum

- Tasks are moved on card board
 - States: *Open, In Progress, In Review, Done*
 - Jira issues are changed accordingly
- Take your task
 - Taking a card = „Assign to me“ in Jira
- Starting a user story → specify person responsible for story
- Story responsible coordinates final User Story test



Working with Tasks







As a user I want to log in into the system
Created: Today 08:26 AM Updated: Today 08:26 AM

Component/s: [component1](#)

Affects Version/s: [v1](#)

Fix Version/s: [v1](#)

Sub-Tasks: **All** [Open](#)

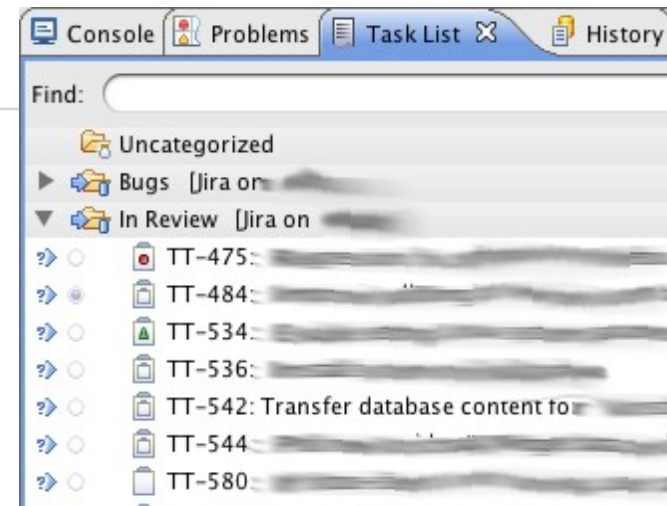
1. [Create Database-Structure](#)   Open
2. [Create Java-Services](#)   Open
3. [Create Login screen](#)   Open

[Add Sub-Task](#)

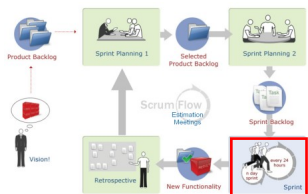


Tasks on Task Board

Tasks in Jira

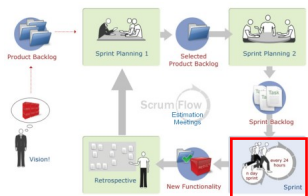


Tasks in Eclipse - Mylyn



Technology Stack (production code, excerpt)

- Java 6
- **Spring** (incl. auto wiring)
- Hibernate (with annotations)
- Apache-commons-*
- XStream
- **Tapestry 5**
- jQuery
- Joda-Time
- Jetty & Tomcat
- Log4j / Slf4j
- Quartz
- ...

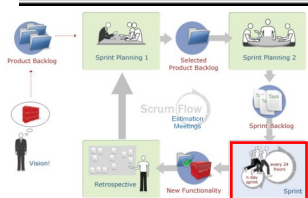


Build & Test Server: Hudson

- Hudson, <https://hudson.dev.java.net/>
 - Continuous Integration
 - Nightly Build
 - External Integration
 - 1-Click Release Job

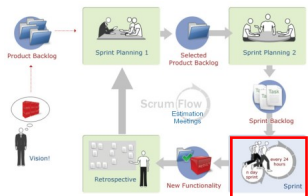
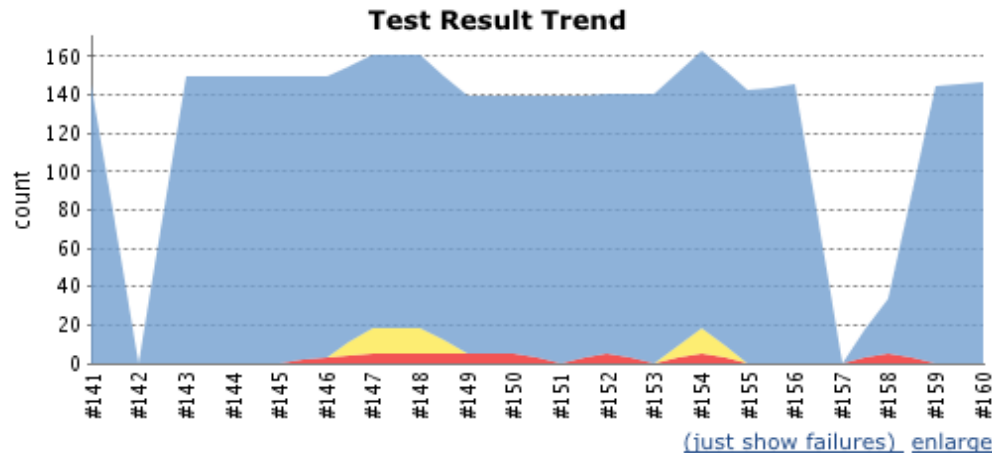


S	W	Job ↓	Last Success	Last Failure	Last Duration
		ContinuousIntegration	12 hr (#486)	14 hr (#478)	3 min 12 sec
		External Integration	5 hr 31 min (#59)	9 days 23 hr (#45)	2 min 9 sec
		Nightly Build	5 hr 48 min (#160)	18 hr (#158)	16 min
		Release	8 days 17 hr (#28)	13 days (#25)	29 min




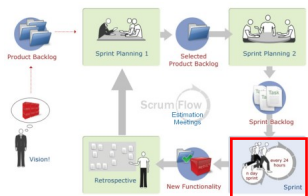
Reporting via Hudson

- Cobertura, FindBugs, Surefire, Javadoc, PMD, Checkstyle, Emma, VNC/Selenium (executed via Maven plugins) are part of nightly build
- E-Mails for broken builds
- The person breaking the build has to give free cookies to everyone ☺
 - Ice in the summer is a good substitute



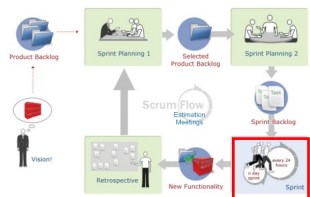
Artifact Management

- It's all about feedback loops!
 - Subversion: code commits only possible with JIRA Issue Number
 - Commit acceptance plug-in
 -  builds
 - Reproducible scripted builds, IDE independent
 - Enterprise Maven Repository



Custom Jira Workflow

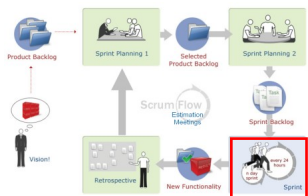
- Explicit peer review, with checklists from Wiki
 - Link to source code & documentation!
 - Task/Story fulfilled? Tests sufficient?
 - Code style? Documentation?
 - Configuration?
 - ...
- No explicit written development process, as tooling provides executable processes





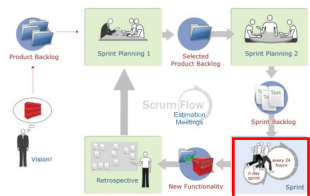
Documentation

- *Emergent documentation* with implicit updates and verification during review process
- JavaDoc for source code, Confluence for the rest:
 - Development:
 - Domain model, deployment, architecture, branching, team, guides
 - Install Guide, Admin Guide, User Guide
 - Separate Spaces in Confluence



Feedback from Live System

- Log4j done with care (logging statements, log levels, ...) from the beginning
- Log levels switchable at runtime
- Nagios monitoring
 - E-Mail and SMS notifications
- Error reporting from live system via detailed exception emails, GPG encrypted if necessary



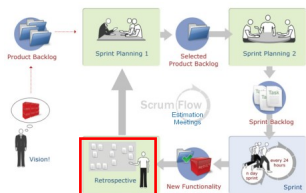
Sprint Review

- Version from stage repository (“release build”)
- Deployed on test system
- **Who:** Whole team plus Product Owner and Users
- **Result:**
 - Happy stakeholders
 - Some new product ideas
 - Jira issues (esp. ideas for improvement)
- 1-2 h, not much preparation (no ppt!)



Retrospective

- Team plus Product Owner if invited
 - And invited persons from other departments, if necessary
- About 2 h (at the beginning longer)
- Gadget-free environment – white boards are sufficient
- Result is documented in Confluence (pictures, action list)
 - And reviewed at the beginning of the next retrospective
- Action Items are pinned at the wall



Regular Estimation Meetings

- Once per week
- Different scopes (from whole product backlog to next sprint input)
- **Who:**
 - Product Owner plus representative team members (architecture, requirements, development, test, design, documentation,...)
 - Not necessary all the time: all team members
 - Too expensive? But only Team knows what will come...
- **Result:** Rough estimates of possible User Stories

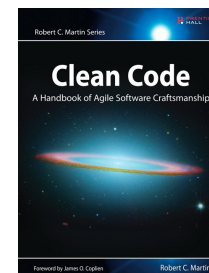
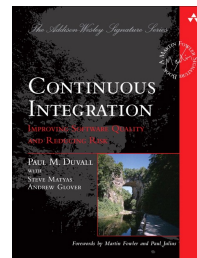
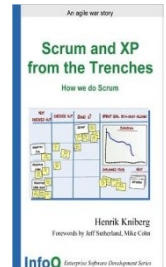


Further Topics

- Scrum Coaching, e.g.:
 - Conduction of Retrospectives
 - Agile Estimation and Planning
- Agile Development Practices, e.g.:
 - Test Driven Development, Continuous Integration, Build Management
 - Automation (Test, Environment setup, Rollout, ...)
- Reviews (development processes, architecture, software)
- Software Development
 - Especially Java! (JEE, Spring, Hibernate, TestNG, OSGi, Eclipse RCP, Maven2, Hudson, JavaScript, ...)

Books about Scrum

- Scrum. Produkte zuverlässig und schnell entwickeln
 - ISBN 978-3446414952 (Boris Gloger)
- Scrum and XP from the Trenches
 - ISBN 978-1430322641 (Henrik Kniberg)
 - Free PDF:
<http://www.infoq.com/minibooks/scrum-xp-from-the-trenches>
- The Art of Agile Development (James Shore, Shane Warden)
 - ISBN 0-596-52767-5
- Continuous Integration: Improving Software Quality and Reducing Risk (Paul Duvall, Steve Matyas, Andrew Glover)
 - ISBN 978-0321336385
- Clean Code: A Handbook of Agile Software Craftsmanship (Robert C. Martin)
 - ISBN 978-0132350884





Gerhard Müller
Diplom-Informatiker (Univ.)
Partner



TNG Technology Consulting GmbH
Betastr. 13a
85774 Unterföhring

Tel. +49 (0)89 2158 9960
Fax +49 (0)89 2158 9969
Mobil +49 (0)179 1338 060

gerhard.mueller@tngtech.com



Dr. Martin Wagner
Dipl.-Inf.
Senior Consultant



TNG Technology Consulting GmbH
Betastr. 13a
85774 Unterföhring

Tel. +49 (0)89 2158 9960
Fax +49 (0)89 2158 9969
Mobil +49 (0)176 2394 7429

martin.wagner@tngtech.com