Handout

Applied Scrum in a MedTech Project

Talk at Scrum Breakfast September 2nd 2009

Peter Rey

bbv Software Services AG

bbv behind things.

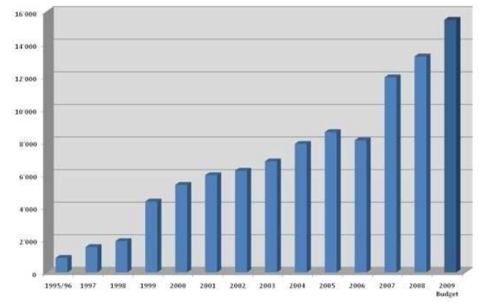
Content

bbv Facts and Figures

bbv Software Services AG is a pure software service provider specialized as outsourcing partner and main contractor for customer specific software products and systems.
bbv Software Services was founded in November 1995.

The main service areas are:

- System Services [.Net/Java]
- Apparatus Services [C++/.Net Mobile]
- Testing Services



Turnover 1995 to 2009

14 Years of Success

Since their foundation bbv Software Services AG has had continuous growth with satisfied customers and strong references. The main focus is on application knowledge and highly educated employees with above average technology skills. Most employees have a bachelor, master or even a post doctorate degree.

More than 80 Employees

By December 31st 2008, bbv Software Services AG had an employee count of over 80 at its three locations Lucerne, Zug and Berne. bbv Software Services uses a vast network of partners and self-employed specialists/freelancers to provide its high quality services. At present bbv Software Services AG trains three apprentices in applied computer science.

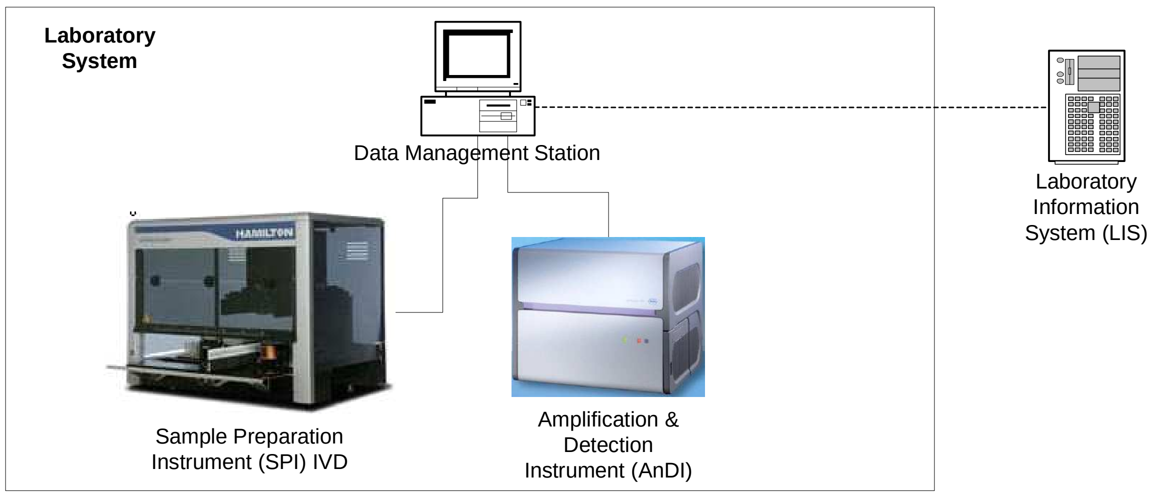
Project Overview

MedTech Project for a new Laboratory-System

The Laboratory-System is used by lab operators for the diagnosis of virus and bacteria in the In-Vitro Diagnostics (IVD) field. It offers fully automated sample preparation combined with real-time polymerase chain reaction (PCR) technology for amplification and detection (DNA replication). Together with easy-to-use software the system provides true walk-away automation for maximum lab efficiency.

At the moment the following tests (assays) are supported:

- HPV Human Papilloma Virus
  (responsible for cervical cancer "Gebärmutterhalskrebs")
- CT Chlamydia Trachomatis (sexually transmissible disease)
- NG Neisseria Gonorrhoeae (sexually transmissible disease)


Laboratory System
Data Management Station
Laboratory Information System (LIS)
HAMILTON
Sample Preparation Instrument (SPI) IVD
Amplification & Detection Instrument (AnDI)

Prerequisites

Business case	world-wide market, time to market
Duration	18 months (from inception to release – input formal validation to the FDA)
Quality standards	FDA regulatory compliance, In-Vitro Diagnostics (IVD)
Software team	up to 10 engineers
Complete team	close to 100 people
International team	USA, Switzerland, Germany
Requirements	instable or not complete
User Interface	prototype approach
Challenge	use new application framework (from customer)
Technology	.NET 3.0, C#, DevExpress, Windows XP, Oracle 11

• Iteration duration	2 weeks
• Number of iterations	30

Challenge - Scrum vs. Regulatory

Software Development Processes

On the customer side:

The customer has a V-Model and RUP based process, strongly waterfall oriented and artifacts driven. This process is derived from the Federal Drug Administration FDA certification constraints.

On the bbv side:

The preferred agile approaches at bbv are: Iteration Zero, Test- Driven Development (TDD), Scrum and some aspects of Extreme Programming (XP).

Departments and Teams on the Customer Side

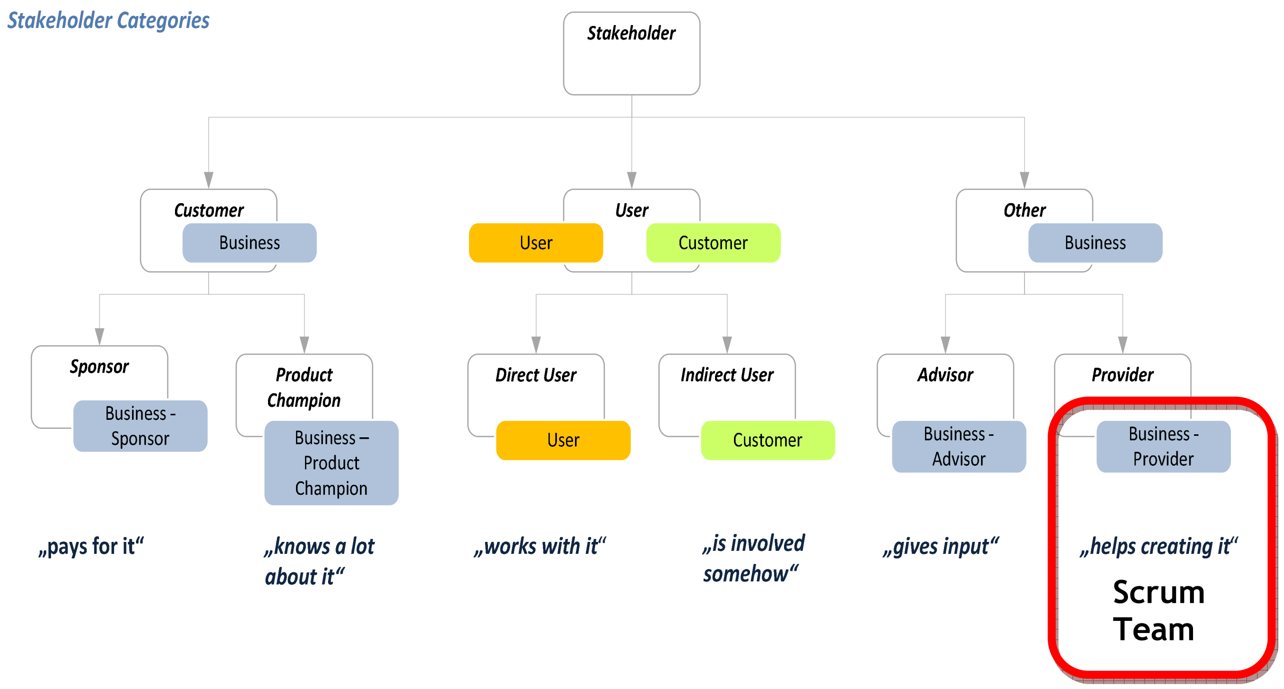
- Whole departments and teams are used to work in a regulatory environment.

  For example: software testers in the integration lab are not used to test software without having requirements and all test cases completed. There is also absolutely no comprehension why a software development team needs a tester in the team!
- Complete management team at customer was new.
- On the other hand bi-weekly releases on CD are way too much for an integration lab used to work in a regulatory environment.

Conversation – Information Exchange

Conversation between development team, integration team, business people and management must be very informal – best face-to-face – to be effective. Instead of writing long notes or documents, give a call or chat to solve the problem almost instantly if possible.

Organizational Structure



How we started

In spring 2007 the customer was setting-up a complete project (hardware and software), getting some ideas about product requirements, etc.

In August 2007 we started with software development. At this point we were only 3 people on our team: system architect/business analyst, product owner/scrum master and senior developer.

After one month of sprint zero - setting up the software development project, getting product backlog ready, evaluate major risks, etc. – we started our first two week sprint and delivered our first release on CD to the customer. At this point there were **no formal software requirements** written – all we had were a few stories. One issue was proof of concept for the integration of the new 3rd party instrument. Another issue was first usage and integration of new customer application framework.

The following months we ramped-up the team and on the customer side they started writing software requirement specifications (SRS).

We constantly improved our process: trained team in Test-Driven Development and Agile Design, automated build process and running unit tests with scripts, create structure of folders and files for CD creation, build up continuous integration server, etc.

Problems we faced doing Scrum

Convince Customer

One of the major issues at the beginning of a project is – how to convince a customer (or future customer) to work Agile.

Usually for the first agile project with a new customer, you need a hell of a lot of **trust**.

In our case, trust and a clear product case existed and a tremendous time pressure. Usually those kinds of projects have duration of 3 to 4 years. Since a competitor was almost ready on the market with another product, the project had to be done in less than 2 years. The only chance was – working Agile!

Team

Building up a good working team needs quite some time. Good engineers are a must!

With team size you have to experiment. We had once a team of ten (incl. Product Owner and Scrum Master) and found out, that the meetings were absolutely not efficient anymore…a size of 7 to 8 total works fine for us.

Once in a while we were struggling with the team "self-organizing" and "responsibility" part. The team had to learn to have the guts to decide (courage) and not just think – well somebody else will do it…

Other important features of a scrum team member are **openness**, **trust** and **appreciation**. There must be an atmosphere of trust in and around of the team. So that mistakes can be admitted – otherwise the **transparency** will suffer.

How we do Scrum

Pre-Planning Meeting

We introduced a Pre-Planning Meeting in the second half of the sprint. As information for the team what's coming up next and for the product owner to get a feeling about the time estimation for stories for the upcoming sprint.

We hold a one hour pre-planning meeting with the whole team.
Therefore the Sprint Planning Meeting at the beginning of a sprint will be shorter. Usually 3 to 4 hours for a 2 weeks sprint.

Sprint Planning Meeting – Part I – Part II

We split the Sprint Planning Meeting into two parts.

Part I: Product Owner (PO) presents his vision for the upcoming sprint. He explains the stories from the backlog with the highest priorities. The team may ask questions about the stories to get a good understanding. The team must be able to estimate the stories afterwards.
At the end of Part I all desired stories from the PO are defined with priority, description and how to demo. We use only cards for stories at this meeting – no PC – no Excel.
Part I is time-boxed to 1 hour.

Part II: The team estimates the stories for the upcoming sprint using planning poker. We start at the highest priority story, define the tasks and estimate the time in story points.
The team gathers a common understanding of the stories and tasks.
At the end of Part II the team tells the PO all the stories it will do this sprint – commit to!

Architecture Workshop

On the second day of the sprint, after the planning meeting, we hold an architecture workshop with the whole team for about an hour. Here we discuss and explain the latest news about the design and architecture of the software. It also includes tips how to make the software better, respectively how to get rid of design smells, etc.

Team Rules

In case of “really dumb” mistakes – e.g. check-in source just before leaving the office and therefore a build was broken – will be searched by a penalty – like bring in croissants “Gipfeli” next morning for the whole team!

We used to do it with beers – but…

Another rule is **Fist of Five**. This is used for team decisions: 5 means – I absolutely agree, 3 means – I can live with it, 1 means – I completely disagree.

In case of a meeting is dragging on, we can **squeeze the rat** or hold up our hand. If 3 and more hands are up, the discussion must stop.

Stories / Backlog Items

Additional to new stories, we map the software issues SWI (can be bugs and change requests) to stories and tasks.

As soon as the formal requirements are complete – we use them as agreement of a story (acceptance criteria).

Sprint Review (Demo)

Sprint Demo on Friday afternoon with adjacent drink “Apéro” at the end of each sprint.

All stakeholders are invited. Gives immediate feedback to team – advocates face-to-face conversation.

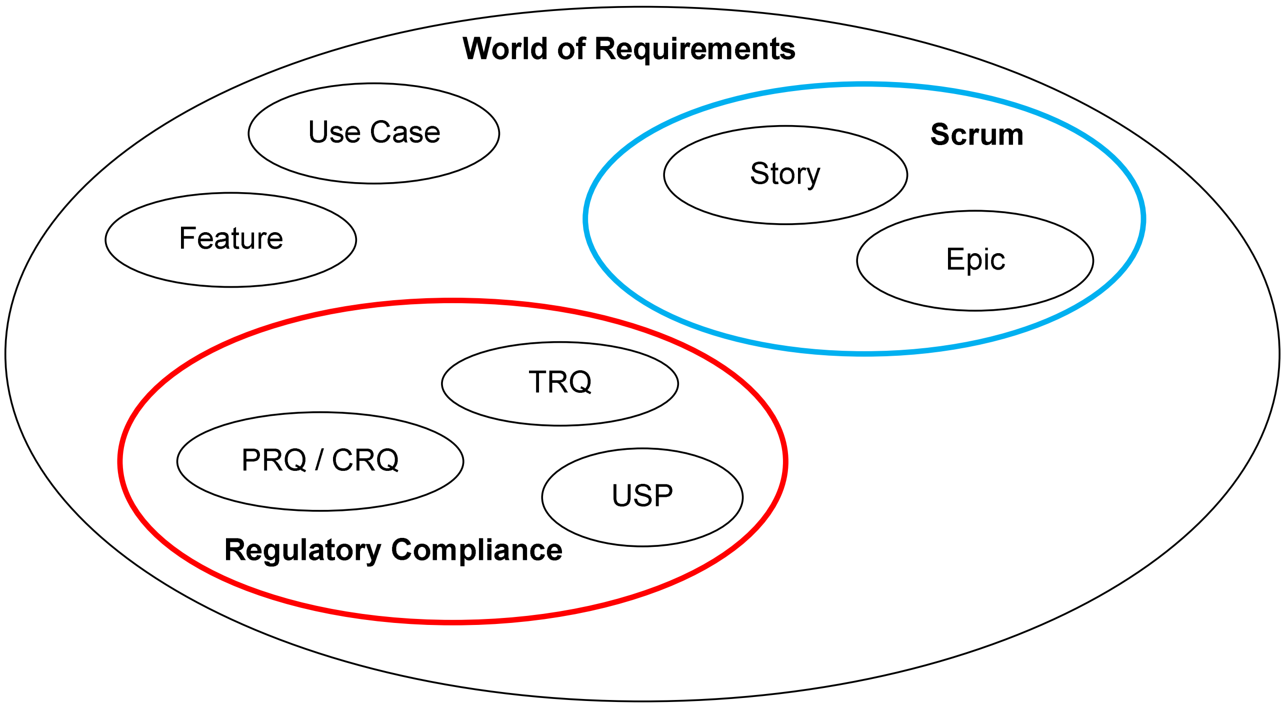
Mapping Formal Requirements to Stories and vice versa

There are three different levels of requirements on the customer side:

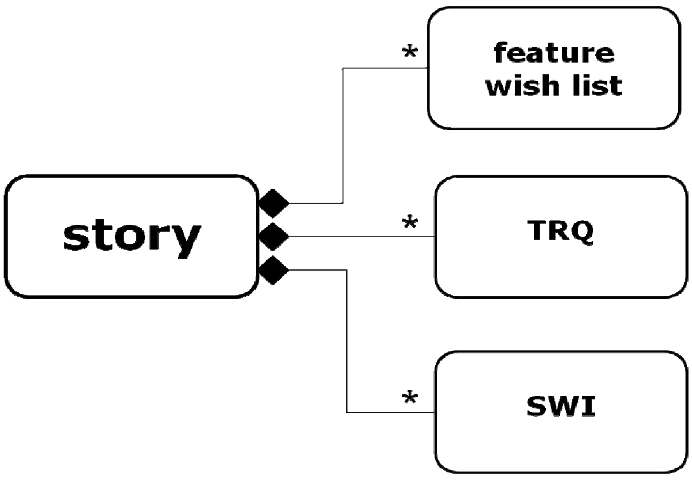
Customer / Product Requirements (PRS -> PRQ)

System Requirements (SRS -> TRQ) – hardware and software

Unit and Subsystem Requirements (USP)



We had to map the system requirements (TRQ) from the customer to stories.
At the beginning of the project there were no system requirements available so the input for the story was the wish list. Later in the project additionally to system requirements, we mapped also SWI's (software issues), could be bugs or change requests to stories.



Toolbox

Customer

- IBM Rational ClearQuest
  SW Change Management Tool (Change Request and Bug Tracking)
- IBM Rational RequisitePro
  Requirements Management Tool (Word, PDF, Excel documents)
- DVS SAP System R/3
  (Dokument Verwaltungssystem)
- Office 2003

bbv software development

We prefer open source components and also contribute to the open source community – please see NMock2 and bbv.Common.

.NET Framework 3.0	
MS Visual Studio	IDE Integrated Development Environment
ReSharper	Add-On to Visual Studio – increase productivity, refactoring
SVN	Subversion - version control system for source code, documents, tools
NAnt	Automated Build Tool
NUnit	Unit Testing Framework
NMock2	Mocking Framework (incl. Add-In to Resharper) free download at ***http://sourceforge.net/projects/nmock2***
PartCover	similar to NCover (but better) – code coverage (> 80%) lines of code (except Views)
NSIS	Nullsoft Scriptable Install System – scriptable installer
StyleCop	by Microsoft. StyleCop - enforces Coding Guidelines Analyzes C# source code to enforce a set of style and consistency rules. It can be run from inside of Visual Studio or integrated into an MSBuild project. (incl. ReSharper Plug-In)
FxCop	by Microsoft. FxCop - enforces Design Guidelines analyze .NET Assemblies about design, localization, performance, security. FxCop is an application that analyzes managed code assembl.
CC.NET	Cruise Control .NET – Continuous Integration Server
bbv.Common	library of components and functionality including: asynchronous notification component with thread switching, enum based state machine, programmatic context based rule engine, etc. free download at ***http://sourceforge.net/projects/bbvcommon***
Excel	for product backlog and project management

Generating Documents for Regulatory Compliance

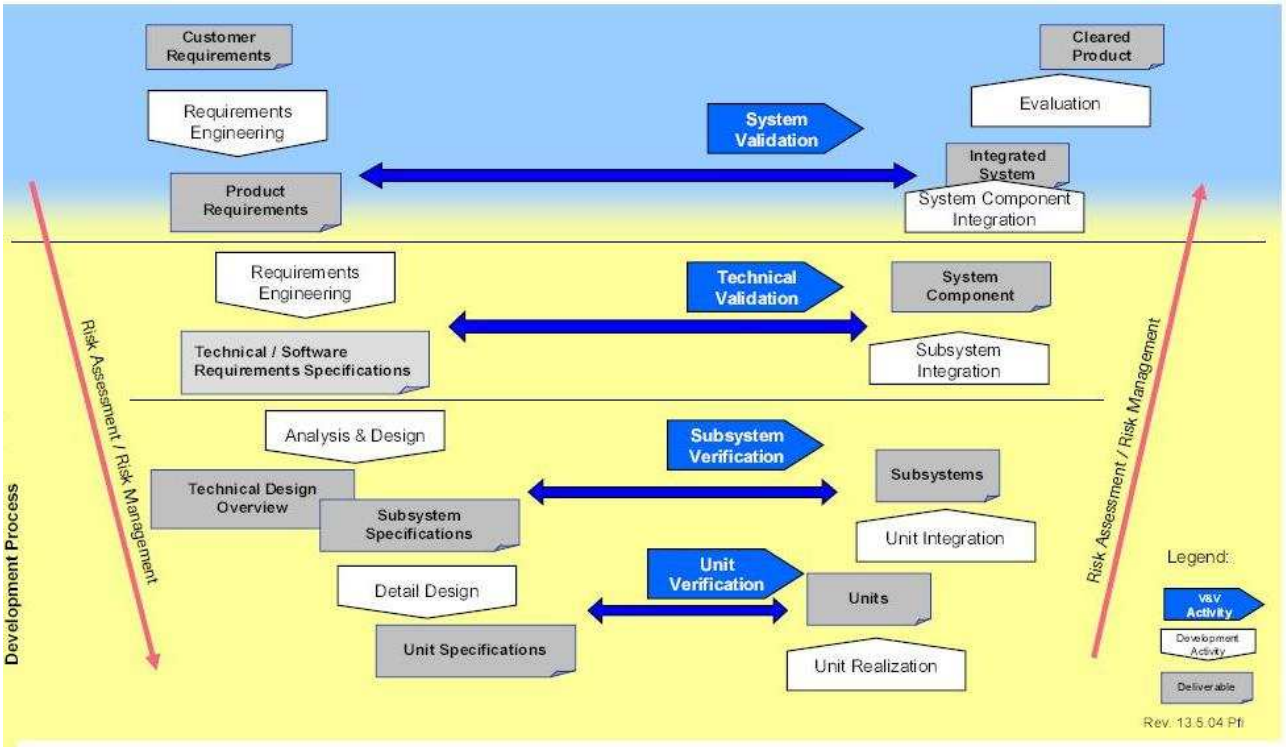
In a regulatory environment you must prove that all requirements are implemented.

Therefore all requirements must be traced from customer / product requirements (PRQ) over system requirements (TRQ) to unit requirements (USP). On all 3 levels the corresponding test scenarios must prove the implementation.

On the lowest level – Subsystem and Unit Specification – we as development team had to prove the proper implementation. Therefore we wrote unit tests on a bigger scale (boundary was a unit not a class) with expectations according to the USP. We call those unit tests "VV Test" – stands for Verification and Validation Test. The output was a .NET assembly with specific attributes for additional information per each unit.

A custom written tool (TestDocGen), which uses those .NET assemblies as input creates XML and PDF output containing all necessary information about the required test.

Then the XML output file could be imported in RequisitePro where all requirements and tests were traced.



The generated document (report) covers the automated tests for the project.
It contains the following artifacts:

- Verification Plan
- Test Case
- Test Result
- and Verification Report

The Verification Plan is part of the Verification Report and the Test Cases are part of Test Results.

Lessons Learned

- Good engineers are the asset, agile methods make engineers better
- Only co-located teams are effective
- Only three new technologies/processes per year (or time)
- Business users are extremely reluctant to embrace Agile approaches
- Iteration rhythm is often too fast for organizations
- It’s all about trust
- Retrospective Meeting – very important for the team to reflect on how to become more effective, then tune and adjust its behavior accordingly

bbs’s Agile Strategy

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

At bbv - Agile Projects Have

- A clear product case, but requirements are not completed
- An empowered team, management means coaching
- A product owner as part of the team
- Iterations are completed in 1 to 4 weeks
- Automatic unit test, build, release, and issue tracking

Bibliography

[1] Scrum and XP from the Trenches, Hernik Kniberg, ISBN: 978-1-4303-2264-1

[2] Test-Driven Development by Example, Kent Beck, Addison Wesley 2003

[3] Agile Principles, Patterns, and Practices in C#, Prentice Hall,
(Robert C. Martin Series)

[4] Agile Project Management with Scrum, Ken Schwaber, Microsoft Press 2004

[5] User Stories Applied: For Agile Software Development, Mike Cohn,
Addison Wesley 2004

[6] Refactoring: Improving the Design of Existing Code, Addison Wesley 1999