[Achtung: Verwenden Sie einen Sperrvermerk nur in sehr gut begründeten Fällen!]

[evtl. Sperrvermerk]

Auf Wunsch der Firma [FIRMA] ist die vorliegende Arbeit bis zum [DATUM] für die öffentliche Nutzung zu sperren.

Veröffentlichung, Vervielfältigung und Einsichtnahme sind ohne ausdrückliche Genehmigung der oben genannten Firma und der/dem Verfasser/in nicht gestattet. Der Titel der Arbeit sowie das Kurzreferat/Abstract dürfen jedoch veröffentlicht werden.

Dornbirn, Unterschrift der Verfasserin/des Verfassers

Firmenstempel

FH Vorarlberg
University of Applied Sciences

Qualitätsmanagement in Scrum-Teams

Untertitel

Masterarbeit
zur Erlangung des akademischen Grades

Master of Science (MSc)

Fachhochschule Vorarlberg
Informatik

Betreut von
Prof. Dr. Michael Felderer

Vorgelegt von
Daniel Grießer
Dornbirn, Juli 2018

[evtl. Widmung]

[Text der Widmung]

Kurzreferat

[Deutscher Titel Ihrer Arbeit]

[Text des Kurzreferats]

Abstract

[English Title of your thesis]

[text of the abstract]

[evtl. Vorwort]

[Text des Vorworts]

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

[evtl. Abkürzungsverzeichnis]

ETW Energietechnik und Energiewirtschaft

SQL Structured Query Language

Bash Bourne-again shell

1 Einleitung

(Mit Qualitäts-Analysetools, wie z.B. SonarQube[1], können ganze Softwaresysteme kontinuierlichen Qualitätstests unterzogen werden. Durch die ermittelten Kennzahlen können Aussagen zur Qualität des gesamten Systems, über einzelne Komponenten, bis hin zu einer einzelnen Quellcode-Datei getroffen werden.

Auch Scrum[2] wird als agiles Vorgehensmodell in der Softwareentwicklung immer beliebter. Dabei wird bei mehreren Teams, die auf vielen Systeme arbeiten, auf 2 Arten von Scrum Teams zurückgegriffen: Feature- oder Komponenten-Teams.

Diese Arbeit beschäftigt sich mit dem Qualitätsmanagement in Scrum-Teams. Das bedeutet, dass Kennzahlen zu Qualitätsmerkmalen nicht auf System-, sondern auf Komponentenebene gesammelt und aggregiert werden, um für jedes Team eine individuelle Sicht auf das Qualitätsmanagement bereitzustellen. Um das zu ermöglichen, wird erst eine Vorgehensweise zur Ermittlung von relevanten Kennzahlen entwickelt und diese an einer Beispiel-Organisation angewendet. Zur Sammlung, Auswertung und Darstellung dieser Kennzahlen wird eine Software entwickelt, die in eine bestehende Umgebung integriert werden kann.)

... schreibe ich ganz am Schluss neu

[1] *Continuous Code Quality | SonarQube.* URL: `https://www.sonarqube.org/` (besucht am 05. 01. 2018).

[2] *Scrum.* URL: `http://www.scrum.org` (besucht am 05. 01. 2018).

2 Situationsanalyse

2.1 Scrum in mehreren Teams[3]

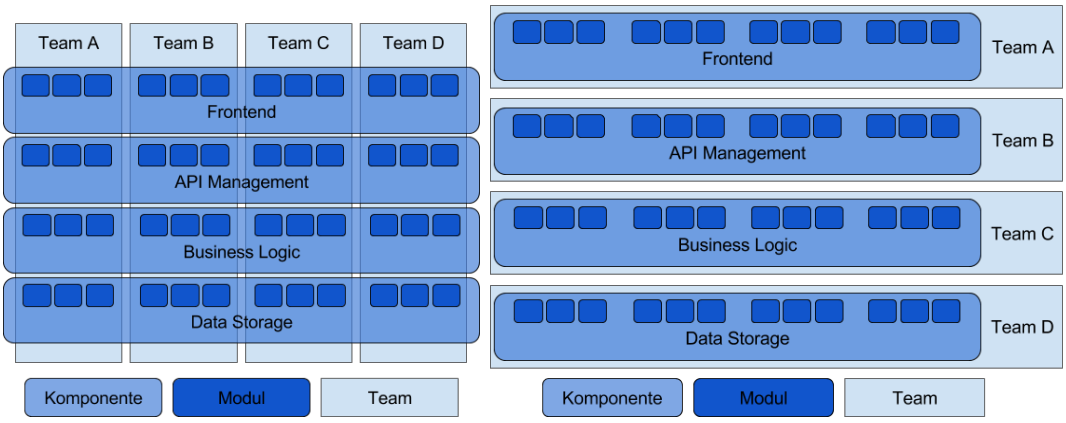
Das Scrum Framework beschreibt die agile Vorgehensweise für ein Team (ein Team entwickelt ein Produkt). In der Realität existieren aber oft mehrere Teams und/oder mehrere Produkte. Dahingehend muss die Organisation der unterschiedlichen Scrum Teams individuell angepasst werden. Für die Trennung der Teams gibt es unterschiedlche Ansätze:

Trennung nach Organisationseinheiten
: Die Teams werden entlang der Abteilungsstruktur einer Organisation getrennt. Aus Scrum-Sicht macht das nicht immer Sinn, da bei der Umsetzung eines Features Abhängigkeiten zu anderen Teams bestehen (keine cross-funktionalen Teams).

Trennung nach Komponenten (Komponenten-Teams)
: Die technischen Komponenten werden den Teams zugeteilt, was ebenfalls zu Abhängigkeiten zu anderen Teams führt und eine gute Abstimmung zwischen den Teams voraussetzt.

Trennung nach fachlichen Themen (Feature-Teams)
: Jedes Team entwickelt, unabhängig von den anderen Teams, eine fachliche Komponente. Diese Variante erfüllt die Forderung des Scrum Frameworks nach cross-funktionalen Teams, weshalb bei dieser Form die Abstimmung zwischen den Teams am geringsten ist.

[3] vgl. Rolf Dräther, Holger Koschek und Carsten Sahling. *Scrum: kurz & gut*. 1. Auflage. O'Reillys Taschenbibliothek. Beijing Cambridge Farnham Köln Sebastopol, Tokyo: O'Reilly, 2013. ISBN: 978-3-86899-833-7, S.172ff.



(a) Feature-Teams (b) Komponenten-Teams

Abbildung 2.1: Scrum Teams

In allen Varianten existieren aber pro Team unterschiedliche Software-Module und (agile) Prozesse, die unabhängig voneinander die Team-Qualität als gesamtes bestimmen.

2.2 Software-Qualität[4]

Eine mögliche Definition von Software-Qualität findet sich in der DIN-ISO-Norm 9126:

> Software-Qualität ist die Gesamtheit der Merkmale und Merkmalswerte eines Software-Produkts, die sich auf dessen Eignung beziehen, festgelegte Erfordernisse zu erfüllen.

Wie aus dieser Definition schon erkennbar ist, gibt es viele unterschiedliche Kriterien, um die Qualität von Software zu bewerten. Einige wesentliche Merkmale, um die Qualität von Software bewerten zu können, lassen sich in kunden- und herstellerorientierte Merkmale unterteilen:

Kundenorientierte Merkmale

Nach außen hin sichtbare Merkmale, die sich auf den kurzfristigen Erfolg der Software auswirken, da sie die Kaufentscheidung möglicher Kunden beeinflussen.

Funktionalität (Functionality, Capability)

Beschreibt die Umsetzung der funktionalen Anforderungen. Fehler sind hier häufig Implementierungsfehler (sogenannte Bugs), welche durch Qualitätssicherung bereits in der Entwicklung entdeckt oder vermieden werden können.

Laufzeit (Performance)

Beschreibt die Umsetzung der Laufzeitanforderungen. Besonderes Augenmerk muss in Echtzeitsystemen auf dieses Merkmal gelegt werden.

Zuverlässigkeit (Reliability)

Eine hohe Zuverlässigkeit ist in kritischen Bereichen, wie z.B. Medizintechnik oder Luftfahrt, unabdingbar. Erreicht werden kann diese aber nur durch die Optimierung einer Reihe anderer Kriterien.

Benutzbarkeit (Usability)

Betrifft alle Eigenschaften eines Systems, die mit der Benutzer-Interkation in Berührung kommen.

Herstellerorientierte Merkmale

Sinf die inneren Merkmale, die sich auf den langfristigen Erfolg der Software auswirken und somit als Investition in die Zukunft gesehen werden sollten.

Wartbarkeit (Maintainability)

Die Fähigkeit auch nach der Inbetriebnahme noch Änderungen an der Software vorzunehmen. Wird oft vernachlässigt, ist aber essentiell für langlebige Software und ein großer Vorteil gegenüber der Konkurrenz.

Transparenz (Transparency)

Beschreibt, wie die nach außen hin sichtbare Funktionalität intern umgesetzt wurde. Gerade bei alternder Software, kann es zu einer Unordung kommen, welche auch Software-Entropie (Grad der Unordnung) genannt wird.

[4] vgl. Dirk W. Hoffmann. *Software-Qualität*. eXamen.press. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. ISBN: 978-3-642-35699-5 978-3-642-35700-8, Kapitel 1.2.

Übertragbarkeit
: Wird auch Portierbarkeit genannt und beschreibt die Eigenschaft einer Software, in andere Umgeungen übertragen werden zu können (z.B. 32-Bit zu 64-Bit oder Desktop zu Mobile).

Testbarkeit (Testability)
: Testen stellt eine große Herausforderung dar, da oft auf interne Zustände zugegriffen werden muss oder die Komplexität die möglichen Eingangskombinationen vervielfacht. Aber gerade durch Tests können Fehler frühzeitig entdeckt und behoben werden.

Je nach Anwendungsgebiet und den Anforderungen der Software haben die Merkmale unterschiedliche Relevanz und einige können sich auch gegenseitig beeinflussen, wie aus der Korrelationsmatrix ersichtlich.

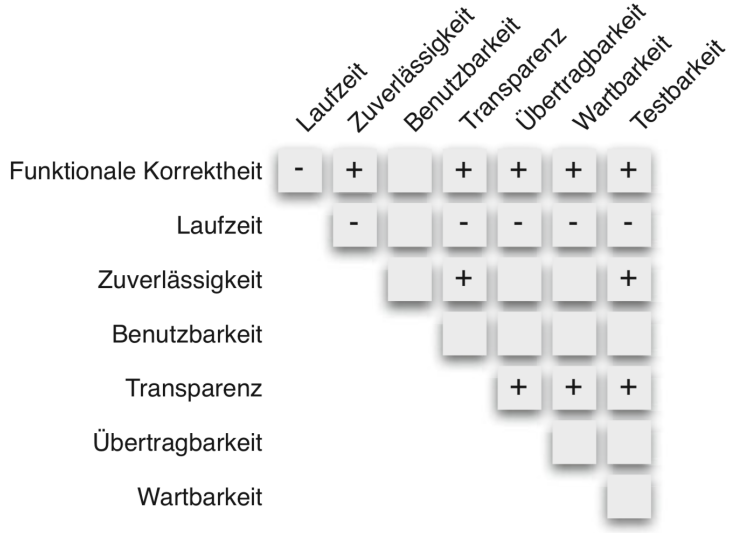


Abbildung 2.2: Korrelationsmatrix Qualitätskriterien[5]

2.3 Qualitätsmetriken

2.3.1 Testmetriken

2.3.2 Softwaremetriken

2.3.3 Agile-Metriken

[5]Dirk W. Hoffmann. *Software-Qualität*. eXamen.press. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. ISBN: 978-3-642-35699-5 978-3-642-35700-8, S. 11, Abb. 1.3.

3 Zielsetzung

3.1 Vorgehensmodell

Entwicklung eines Vorgehensmodells zur Bestimmung von relevanten Qualitätsmetriken von Teams.

3.2 Software

Entwicklung einer Software zur Darstellung von Qualitätsmetriken von Teams.

4 Methodik

4.1 Vorgehensmodell

Kriterien, auf was muss geachtet werden, etc.

4.2 Software

Technologien, Plattform, etc.

5 Ergebnisse

5.1 Vorgehensmodell

... Ergebnis der Ausarbeitung.

5.2 Einführung des Vorgehensmodells

... bei Gebrüder Weiss.

5.3 Inbetriebnahme der Software

... allgemein, Beschreibung der Connectoren, Darstellungsarten, etc.

5.4 Evaluierung des Vorgehensmodells

... wie wurde es angenommen? Welche Auswirkungen hatte es?

6 Schlussfolgerungen

Effektivität des Modells, Erkenntnisse aus der Einführung bei Gebrüder Weiss

7 Zusammenfassung

Erkenntnisse und Ausblick

Literatur

Continuous Code Quality | SonarQube. URL: `https://www.sonarqube.org/` (besucht am 05. 01. 2018).

Dräther, Rolf, Holger Koschek und Carsten Sahling. *Scrum: kurz & gut*. 1. Auflage. O'Reillys Taschenbibliothek. Beijing Cambridge Farnham Köln Sebastopol, Tokyo: O'Reilly, 2013. ISBN: 978-3-86899-833-7.

Hoffmann, Dirk W. *Software-Qualität*. eXamen.press. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. ISBN: 978-3-642-35699-5 978-3-642-35700-8.

Scrum. URL: `http://www.scrum.org` (besucht am 05. 01. 2018).

[evtl. Anhang]

Formatvorlage für den Fließtext.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Stellen sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Dornbirn, am [Tag. Monat Jahr anführen] [Vor- und Nachname Verfasser/in]