Manfred Oberlerchner

Einsatz von Testmetriken zur Risikoabschätzung und Qualitätsbeurteilung

Diplomarbeit

Technische Universität Graz

Institut für Softwaretechnologie
Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Franz Wotawa

Supervisor: Univ.-Prof. Dipl.-Ing. Dr.techn. Franz Wotawa

Graz, August 2018

This document is set in Palatino, compiled with pdfLaTeX2e and `Biber`.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, _____ _____

Date Signature

Eidesstattliche Erklärung[1]

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am _____ _____

Datum Unterschrift

[1] Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Danksagung

An dieser Stelle möchte ich all jenen danken, die durch ihre fachliche und persönliche Unterstützung zum Gelingen dieser Diplomarbeit beigetragen haben.

Mein Dank gilt Herrn Univ.-Prof. Dipl.-Ing. Dr.techn. Wotawa, der meine Arbeit durch seine fachliche und persönliche Unterstützung und seine freundliche Hilfsbereitschaft, die er mir entgegenbrachte, begleitet hat.
Danken möchte ich ebenfalls dem Unternehmen Hutchison Drei Austria, insbesondere Herrn Dipl.-Ing. Dr.techn. Günther Fischer und Head of Testcenter Andreas Kronlachner für die unterstützende Zusammenarbeit und Ermöglichung dieser Arbeit.

Mein besonderer Dank gilt meiner gesamten Familie, insbesondere meiner Mutter, die mir mein Studium ermöglicht und mich in all meinen Entscheidungen unterstützt haben.

Weißenstein, 2018
Manfred Oberlerchner

Abstrakt

In Softwareentwicklungsprozessen, beispielsweise bei der Verwendung des Wasserfall- bzw. V-Modells, kommt es vor, dass aufgrund eines unaufschiebbaren, vorgegebenen Releasetermins (Produktivstellung) und auftretender Verzögerungen in vorhergehenden Prozessschritten für die Phase(n) des Tests die ursprünglich dafür geplante Zeit nicht mehr zur Verfügung steht. Dies, in Kombination mit immer kürzeren Releasezyklen (Entwicklungsphasen, zwischen Releaseterminen), wirft die Frage auf, welche zusätzlichen Optionen es für die Testabteilung gibt, diese Herausforderung bestmöglich anzunehmen.

Eine mögliche Lösung wird darin gesucht, Metriken zu nutzen und einzusetzen, welche eine Qualitätsbeurteilung und Risikoabschätzung unterstützen. Der durchgeführte Softwaretest zur Qualitätssicherung und die daraus gewonnenen Testergebnisse sollen dadurch bestärkt und/oder bestätigt werden. Im Zuge dessen gilt es, anhand vorhandener Möglichkeiten (Daten) Testmetriken aufzustellen.

Ziel dieser Arbeit ist es, Testmetriken zu analysieren bzw. festzustellen, ob sich durch den Einsatz und die Verwendung dieser Metriken Aussagen bezüglich der Qualität einzusetzender Software tätigen lassen - falls ja, welche, falls nein, weshalb nicht.
Ebenfalls betrachtet wird die Möglichkeit einer Risikoabschätzung: Lassen sich im Zuge eines risikobasierten Testansatzes diese Testmetriken einsetzen, um für die Phase des Tests Optimierungen zu finden? Lassen sich anhand der Metrik Rückschlüsse auf die Auswahl notwendiger Tests, die Testdurchführung bzw. den Testaufwand ziehen?

Schlagwörter: Testmetriken, Softwarequalität, Softwaretest, Wasserfallmodell, risikobasiert, Blackboxtests

Abstract

During software development life cycles (SDLC) it sometimes occures, using the waterfall- or V-model for instance, that at the end initially planned time for tests run short. This happens due to occurring delays in prior process steps and scheduled dates for a go Live (release date).
Considering all this plus shorter timespans between releases, questions arise, how the test team is able challenging these situations and which additional options are given.

One option could be, using metrics, which support both quality assessment and risk estimation. Executed tests for quality assurance and given test results should be confirmed and/or acknowledged. In the course of that by using available options (data), test metrics have to be defined.

The goal of this paper is to analyze test metrics, or rather figuring out, if it is possible to make statements about quality of used software via these metrics - if yes, which ones, if not, why.
The options for risk estimation are observed too: Is it possible to use these test metrics via risk based test approach in order to improve them for the test phase? Is it possible drawing conclusions from metrics regarding selection of mandatory test cases, test execution and test effort?

Keywords: test metrics, software quality, software test, waterfall model, risk based, black box tests

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

*„Metriken liefern Zahlen als Grundlage für **KPIs!** (**KPIs!**)“*
(Stockerer, 2018)

Testmetriken - Qualitätsbeurteilung - Risikoabschätzung

Diese im Titel der Arbeit enthaltenen Begriffe umfassen ein sehr breites Spektrum und werden in verschiedensten Branchen, Sparten und Bereichen verwendet und erwähnt.
So auch in Softwareentwicklungsprozessen und das unabhängig vom gewählten Vorgehensmodell. Im Wasserfallmodell, V-Modell, Spiralmodell, Prototyping, Extreme Programming oder auch agilen Modellen (Scrum, Kanban) beispielsweise werden verschiedenste Metriken eingesetzt und verwendet. Um zu Qualität oder Risiken bei gewählten Modellen Aussagen treffen zu können, gelten Metriken als notwendiges und oft verwendetes Managementinstrument.

Das in dieser Arbeit betrachtete Unternehmen arbeitet nach dem Wasserfallmodell. Aus Unternehmenssicht, die der Geschäftsführung, den Abteilungen oder dem Projektmanagement haben Metriken in den verschiedenen Phasen des Wasserfallmodells unterschiedliche Bedeutung und sind unterschiedlich wichtig. Ebenso finden jeweils verschiedene Aspekte bei der Betrachtung der Qualität oder einer Risikoabschätzung Berücksichtigung.

Diese Arbeit beschäftigt sich ausschließlich mit der Betrachtung der Bereiche Metriken, Qualität und Risiko im Softwareentwicklungsprozess aus Test-Sicht. Aus Test-Sicht im Sinne

- der Möglichkeiten, welche es in der Testabteilung im Rahmen ihres Auftrags gibt

- von Daten, auf welche die Testabteilung zugreifen kann
- der Optionen, welche die Testabteilung im Rahmen ihres Auftrags hat
- des Bereiches, welchen die Testabteilung verantwortet und in dem sie Verantwortung wahrnehmen kann.

Testen ist eine der am häufigsten eingesetzten Vorgehensweisen zur Qualitätssicherung.

Verwendete Daten und Fakten dieser Arbeit stammen aus dem Unternehmen, und sie wurden während des täglichen Geschäftsbetriebs gesammelt, eingegeben, erstellt und gepflegt.

In den folgenden Abschnitten dieses Kapitels wird der theoretische Hintergrund zu den Titelbegriffen erläutert. Weiters soll der jeweilige Bezug hergestellt und ein Teilbereich daraus, der den Test betrifft, erklärt werden. Zudem gilt es, den Zusammenhang von Test und Theorie zu zeigen.

1.1 Metrik

„Softwaremetriken definieren, wie Kenngrößen der Software oder des Softwareentwicklungsprozesses gemessen werden."
(Taentzer, 2015)

1.1.1 Was ist Metrik

Das dem Griechischen entstammende Wort Metrik bedeutet allgemein Messsystem bzw. Verfahren zur Messung quantifizierbarer Einheiten oder Größen. (Vgl.[1])

Mathematisch wird die Metrik auch als Abstandsfunktion bezeichnet. Somit ist eine Funktion $d : M \times M \to \mathbb{R}_0^+$, welche jene gewissen Eigenschaften erfüllen muss, durch welche ein metrischer Raum M definiert ist:

- $\forall x, y \in M : d(x, y) = 0 \Leftrightarrow x = y$ und $d(x, y) \leq 0$ (Positive Definitheit)
- $\forall x, y \in M : d(x, y) = d(y, x)$ (Symmetrie)

[1] Workbedeutung.info, 2018.

- $\forall x, y, z \in M : d(x, z) \leq d(x, y) + d(y, z)$ (Dreiecksungleichung)

(Vgl.[2])

Beim Vergleich zweier Punkte eines metrischen Raumes, welche durch deren Koordinaten bestimmt sind, wird durch die Messung dieser Eigenschaften (Abstand) eine Maßbestimmung durchgeführt. Deshalb entspricht das Messen von Eigenschaften eines Systems einer Maßbestimmung im mathematischen Sinn und der Bestimmung von Maßen. Dennoch ist es üblich, in diesem Zusammenhang nicht von Maßen, sondern von Metriken zu sprechen. (Vgl.[3])

1.1.2 Wie werden Metriken unterteilt

Trotz einer Vielzahl an Möglichkeiten Metriken zu kategorisieren, herrschen meist Zweiteilungen vor.

Eine Möglichkeit besteht darin, nach Produktmetriken und Prozessmetriken zu unterscheiden.
Unter Produktmetriken versteht Georg Erwin Thaller (zitiert nach Ott, 2001) Metriken, die sich auf ein einzelnes Produkt beziehen. Dies sind Metriken, die sich auf Spezifikation, Design, Code, Tests oder Wartung einer Applikation beziehen.

Prozessmetriken sind Metriken, welche sich auf den Entwicklungsprozess beziehen, wie er etwa in einem Unternehmen umgesetzt wird. Beispiele dafür sind Messungen, die sich auf die Art der gemachten Fehler beziehen oder Messungen über die Dauer der Fehlerbehebungen.

Andere Möglichkeiten der Kategorisierung wären:

- *statische Metriken* (dokumentationsbezogene) und *dynamische Metriken* (laufzeitbezogene)
- *deskriptive Metriken* (aus Analyse gewonnene) und *prediktive Metriken* (prognostizierte, abgeschätzte)

[2]Wikibooks, 2018-05-19.
[3]Liggesmeyer, 2009.

- *white-box Metriken* (struktur-, konstruktionsbezogene) und *black-box Metriken* (verhaltensbezogene)

Eine weitere Möglichkeit besteht in einer Kategorisierung nach Softwareentwicklungsphasen, also je nachdem, in welcher Phase der Softwareentwicklung diese Verwendung finden.

- Anforderungsanalyse
  - Flesh-Kincaid-Index
  - Sachgerechtigkeits-Metrik
- Spezifikation
  - prediktive Metriken
  - Albrechts Metrik/Function-Point-Verfahren
- Design
  - Intra-modulare Metriken (Komplexität innerhalb der Module)
  - Inter-modulare Metriken (Komplexität zwischen Metriken)
  - Inter- und intra-modulare Metriken (vereinen beide Ansprüche)
- Codierung
  - statische Metriken (Längenmaß, Steuerflussmaß, Datenflussmaß)
  - LOC-Maß
  - Halstead-Metrik
  - McCabe-Metrik
- Test
  - Metriken zur Bewertung des Testniveaus
  - Metriken für die Softwarezuverlässigkeit (MeanTimeToFailure - MTTF)
  - Metriken zur Bestimmung der Testfälle (McCabe-Metrik)
- Wartung
  - aufwandsbezogene Metriken
  - änderungsbezogene Metriken
  - fehlerbezogene Metriken

(Vgl.[4])

[4]Ott, 2001.

1.1.3 Wofür, Welche und Warum

In der Informatik finden Metriken als Bewertungskriterium (numerisches Maß, Maßzahl) oder Methode zur Bewertung von Software durch das Schaffen von Vergleichsmöglichkeiten Verwendung. Bei dieser Form des Messens wird ein Vergleich angestellt. Es wird Objekten (z.B. einer Systemkomponente) ein Messwert zugeordnet und die Werte einer neuen Situation mit jenen einer bereits bekannten verglichen, um daraus Schlussfolgerungen ziehen zu können. (Vgl.[5])

Auf die oben erwähnten Metriken den Softwaretest betreffend (Testniveau, MeanTimeToFailure - MTTF, Testfallbestimmung) wird in der Arbeit nicht eingegangen. Solche stehen nicht zur Verfügung und sind nicht im Einsatz.

Es ist nicht Ziel der Arbeit, die Qualität des Testens zu beurteilen.
Anstelle dessen wird das Ziel verfolgt, anhand vorhandener Daten andere/neue Erkenntnisse und Schlüsse zu gewinnen/ziehen.

Bezugnehmend auf die erwähnten Optionen der Kategorisierung, lässt sich zu den in der Arbeit verwendeten Metriken folgende Einteilung treffen:

- Qualitätsbeurteilung
  - statische-
  - deskriptive-
  - black-box-
  - Produkt- Metriken
- Risikoabschätzung
  - dynamische-
  - prediktive-
  - black-box-
  - Prozess- Metriken

[5] Liggesmeyer, 2009.

1.2 Qualität

Die DIN-ISO-Norm 9126 definiert den Begriff Softwarequalität wie folgt:

„Softwarequalität ist die Gesamtheit der Merkmale und Merkmalswerte eines Softwareprodukts, die sich auf dessen Eignung beziehen, festgelegte Erfordernisse zu erfüllen."

1.2.1 Software und Qualität

Den Begriff Softwarequalität einfach zu fassen, erweist sich als schwierig. Fragen dazu sind:

- Was ist Qualität?
- Was ist Qualität in Zusammenhang mit Software?
- Kann Qualität gemessen werden?
- Wie kann Qualität gemessen werden?
- Kann Qualität beurteilt werden?
- Wie kann Qualität beurteilt werden?

Entwickler, Softwarearchitekten, Tester oder Management, um nur einige Bespiele zu nennen, haben jeweils unterschiedliche Meinungen dazu.

Deshalb macht es Sinn, auf bestehende Modelle für Qualität von Softwaresystemen zurück zu greifen, wie z.B. FURPS/FURPS+, DIN 66272, ISO 9126, ISO 25010, etc., um den Begriff zu erläutern. (Vgl. Gnoyke, 2016)

Da es mittlerweile zum Thema Qualitätsmanagement eine Fülle an Normreihen (z.B. ISO 900x bzw. ISO 14000) sowie allgemeine und spezielle Vorgehensmodelle für den Softwareentwicklungsprozess (z.B. EFQM, das deutsche V-Modell, CMM, SPICE, SPI) gibt, sei hier zur weiteren Erläuterung eine bestimmte Normreihe, die ISO 25010, herausgegriffen.

Abbildung 1.1: Qualitätsmerkmale ISO 25010 (Arendt, 2018)

1.2.2 Normreihe ISO 25010

Die ISO 25010 (siehe Abbildung 1.1) ist die Nachfolgenorm der ISO 9126 und ist eine nicht-harmonisierte[6] Norm, die Qualitätseigenschaften für Software klassifiziert.

Die ISO 9126 eignet sich als Checkliste, um die Vollständigkeit von Softwaresystemanforderungen und Softwaresystemtests zu überprüfen. Zu den Änderungen bei der ISO 25010 zählen zwei neue Hauptkategorien, die IT-Sicherheit und die Kompatibilität. (Vgl.[7])

Folgende Tabelle ISO 25010 (Vgl. Gnoyke, 2016) enthält Erläuterungen zu Abbildung 1.1:

[6]Definition „harmonisierte Norm“: „Eine nicht verbindliche technische Spezifikation, die von einer europäischen Normenorganisation, nämlich dem Europäischen Komitee für Normung (CEN), dem Europäischen Komitee für Elektrotechnische Normung (Cenelec) oder dem Europäischen Institut für Telekommunikationsnormen (ETSI), aufgrund eines Auftrags der Kommission nach den in der Richtlinie 98/34/EG des Europäischen Parlaments und des Rates vom 22. Juni 1998 über ein Informationsverfahren auf dem Gebiet der Normen und technischen Vorschriften und der Vorschriften für die Dienste der Informationsgesellschaft (1) [Anm. Fußnote: siehe Originaldokument] festgelegten Verfahren angenommen wurde. (CE-Wissen, 2018)“

[7]Institut, 2018-02-12.

Functional Suitability	Funktionalität	Die Art und Weise, wie die gewünschte Funktionalität geliefert wird	Vollständigkeit, Korrektheit, Angemessenheit
Performance Efficiency	Effizienz	Stellt die Leistung des Systems in Relation zum Ressourcenverbrauch dar	Zeitverhalten, Ressourcenverbrauch, Kapazität
Compatibility	Kompatibilität	Fähigkeit zum Austausch von Daten mit anderen Systemen oder Komponenten, die auf anderer oder der gleichen Hardware laufen	Ko-Existenz, Interoperabilität
Usability	Benutzbarkeit	Grad der Nutzbarkeit für definierte Benutzer, vorher bestimmte Ziele mit dem System effizient und zufrieden zu erreichen	Erlernbarkeit, Bedienbarkeit, Schutz vor Fehlern des Benutzers, Barrierefreiheit
Reliability	Zuverlässigkeit	Erbringung der Leistungen des Systems unter bestimmten Bedingungen über einen definierten Zeitraum	Reife, Verfügbarkeit, Fehlertoleranz, Wiederherstellbarkeit
Security	Sicherheit	Fähigkeit, die Daten und das System so zu schützen, dass sowohl beabsichtigte als auch unbeabsichtigte Zugriffe erkannt und abgewehrt werden	Vertraulichkeit, Integrität, Nachweisbarkeit, Ordnungsmäßigkeit, Authentizität

Maintainability	Wartbarkeit	Die Wirtschaftlichkeit, mit der ein System angepasst werden kann, um auf Veränderungen der Umwelt, Rahmenbedingungen oder Anforderungen zu reagieren sowie Fehler zu korrigieren	Modularität, Wiederverwendbarkeit, Analysierbarkeit, Änderbarkeit, Testbarkeit
Portability	Portabilität	Einfachheit, mit der ein System von einer Umgebung auf eine andere übertragen werden kann	Anpassbarkeit, Installierbarkeit, Austauschbarkeit

Tabelle 1.1: ISO 25010 (Vgl. Gnoyke, 2016)

1.2.3 Qualitätsbeurteilung

Je höher das Ergebnis bei der Überprüfung aller Produkt-Qualitätsmerkmale gemäß der Checkliste nach ISO 25010 ausfällt, desto leichter lässt sich die Qualität dieses Produkts beurteilen und abschätzen.
Unter der Voraussetzung, dass alle Kategorien abgedeckt sind, kann davon ausgegangen werden, dass alle Softwaresystemanforderungen und Softwaresystemtests vollständig abgedeckt sind.

Folgende Kategorien aus der Norm ISO 25010 finden bei der Qualitätsbeurteilung der Produktqualität in dieser Arbeit Berücksichtigung, da sie in der einen oder anderen Form in den Test einfließen:

- Funktionalität
  - Vollständigkeit
  - Korrektheit

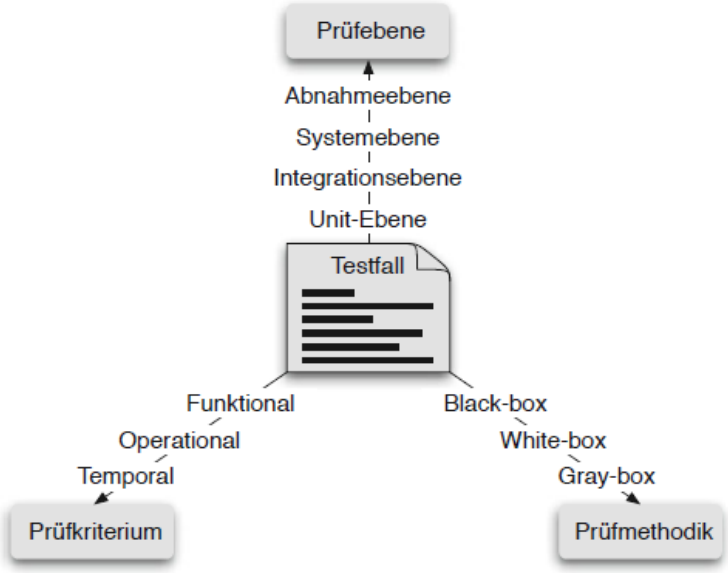

Abbildung 1.2: Die drei Merkmalsräume der Testklassifikation (siehe Hoffmann, 2013, S. 158)

- Kompatibilität
  - Interoperabilität
- Benutzbarkeit
  - Bedienbarkeit
  - Schutz vor Fehlern des Benutzers
- Wartbarkeit
  - Testbarkeit

1.3 Test

1.3.1 Softwaretests

Gemäß IEEE Standard Glossary of Software Engineering Terminology ist der Begriff Softwaretest wie folgt definiert:

„Test: (1) An activity in which a system or component is executed under specified coditions [sic!], the results are observed or recorded, and an evaluation is made of some aspect of the system or component."
(zitiert nach Hoffmann, 2013, S. 157)

Und R. D. Craig und S. P. Jaskiel beschreiben den Begriff Softwaretest folgendermaßen:

„Testing is a concurrent lifecycle process of engineering, using and maintaining testware in order to measure and improve the quality of the software being tested."
(zitiert nach Hoffmann, 2013, S. 157)

Diese beiden Zitate zeigen, dass die Meinungen zur Bedeutung des Begriffs Softwaretest stark variieren.
Wird bei der ersten Definition die Durchführung des konkreten Testfalls angesprochen, beschreibt die zweite Testen als Prozess.

Softwaretests werden zur analytischen Qualitätssicherung eingesetzt. Dabei ist im Regelfall ein Sollergebnis bekannt. Dann wird ein zu untersuchendes Produkt/Programm mit konkreten Eingabedaten ausgeführt und das Istergebnis ausgewertet und überprüft, ob es mit dem Sollergebnis übereinstimmt.

1.3.2 Testfälle

Die Durchführung von Softwaretests anhand von Testfällen findet in allen Phasen des Softwareentwicklungsprozesses statt.
Eine Einteilung/Klassifikation der Testfälle anhand von Merkmalen lässt sich wie folgt durchführen (siehe Abbildung 1.2) :

- Prüfebene
  - Unit-Ebene
  - Integrations-Ebene
  - System-Ebene
  - Abnahme-Ebene
- Prüfkriterium
  - Funktional
  - Operational
  - Temporal
- Prüfmethodik
  - Black-box
  - White-box
  - Gray-box

Prüfebene

Die Prüfebene beschreibt, in welcher Entwicklungsphase der Test durchgeführt wird.
Dabei geht es nicht darum, *Was* überprüft wird. Anstelle dessen werden die Testfälle eingeteilt in:

- Unit-Test
  - auf Modul-Ebene
  - programm-orientiert
- Integrations-Test
  - auf Integrations-Ebene
  - programm-orientiert

- System-Test
  - auf System-Ebene
  - funktions-orientiert
- Abnahme-Test
  - auf System-Ebene
  - funktions-orientiert

Prüfkriterium

Das Prüfkriterium legt fest, welche inhaltlichen Aspekte getestet werden. Die im Laufe eines Projekts durchgeführten Tests lassen sich wie folgt zuteilen:

- Funktional
  - Funktionstest
  - Trivialtest
  - Crashtest
  - Kompatibilitätstest
  - Zufallstest
- Operational
  - Installationstest
  - Ergonomietest
  - Sicherheitstest
- Temporal
  - Komplexitätstest
  - Laufzeittest
  - Lasttest
  - Stresstest

Prüfmethodik

Die Prüfmethodik beschreibt, wie die einzelnen Testfälle konstruiert werden. Diese lassen sich ebenfalls in 3 Kategorien einteilen:

- Black-box Test
  - Herleitung der Testfälle aus der Anforderungs- und Schnittstellenbeschreibung
  - Äquivalenzklassentests
  - Grenzwertbetrachtung
  - paarweises Testen
- White-box Test
  - Testfälle werden aus der inneren Programmstruktur systematisch hergeleitet
  - kontrollflussorientierte Tests
  - datenflussorientierte Tests
  - paarweises Testen
- Gray-box Test
  - Testfälle werden aus der Anforderungs- und Schnittstellenbeschreibung unter Kenntnis der inneren Programmstruktur hergeleitet

Jeder Testfall lässt sich somit anhand von 3 Merkmalen, je einem aus jedem Merkmalsraum, klassifizieren.

1.3.3 Softwareentwicklungsvorgehensmodelle und Test

Das Wasserfallmodell (siehe Abbildung 1.3) besteht aus folgenden Projektphasen:

- Systemanforderungen (Softwareanforderungen)
- Analyse
- Entwurf
- Implementierung
- Test (Überprüfung)
- Betrieb (Wartung)

Beim Wasserfallmodell ist der Test ein Glied in der sequentiellen Kette, welches zwischen Implementierung und Betrieb hängt.
Dadurch kommt es zu einer klaren Trennung von Implementierung und Test, was oft als Konstellation mit wenig Flexibilität gesehen wird.

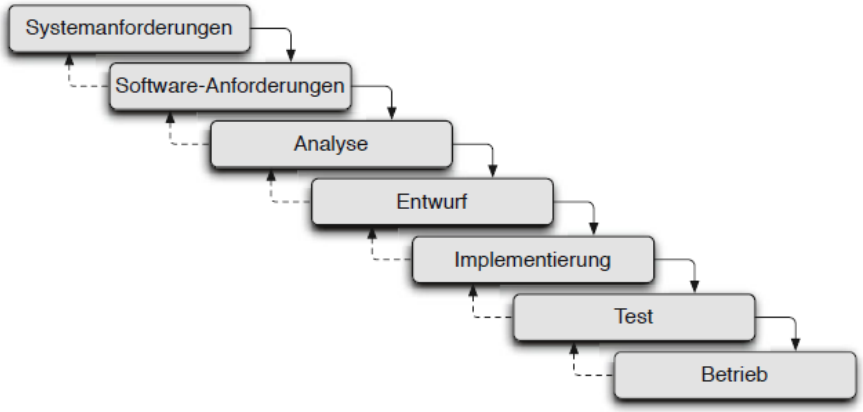

Abbildung 1.3: Das Wasserfallmodell von Royce, 1970 (Hoffmann, 2013)

Eine Weiterentwicklung und Erweiterung des Wasserfallmodells stellt das V-Modell (siehe Abbildung 1.4) dar. Während die Kernphasen des Wasserfallmodells bestehen bleiben, werden durch Hinzufügen eines zusätzlichen Astes wesentliche Aspekte der Softwarequalitätssicherung mit berücksichtigt und durch Tests abgedeckt.

Hinzu kommt beim V-Modell auch die Unterscheidung zwischen Validation und Verifikation.
Die Verifikation wird durch den praktischen Einsatz von Softwaretests und Testfällen abgewickelt, indem überprüft wird, ob das Produkt der Spezifikation entspricht.
Diese Spezifikation steht bei der Validierung nicht im Vordergrund. Bei der Validierung wird der Fokus darauf gelegt, ob das Endprodukt entsprechend dem geplanten Einsatzzweck den Anforderungen und Wünschen des Auftraggebers genügt beziehungsweise entspricht.

1.3.4 Test und Metriken

Testmetriken sind Metriken, welche im Bereich Softwaretest eingesetzt werden. Diese finden meist als Managementtool im Testmanagement Verwendung.

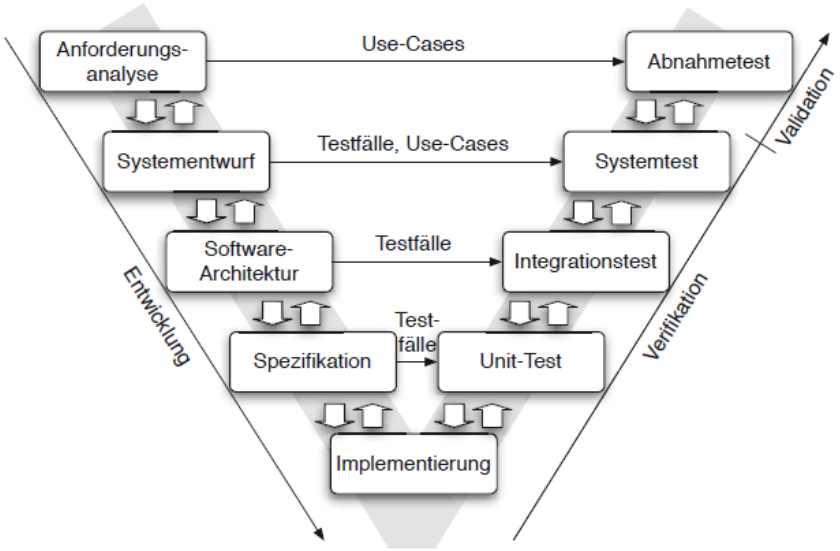

Abbildung 1.4: Das V-Modell (Hoffmann, 2013)

Dabei geht es darum, den Nutzen des Testens zu rechtfertigen, die Testkosten zu planen oder die Testleistung zu beurteilen.
In diesem Zusammenhang dienen sie beispielsweise dazu, die Qualität des Tests selbst, die Testdurchführung oder die Testfallauswahl darzustellen und in Zahlen zu fassen.

Solche Metriken können in 4 Kategorien unterteilt werden (Vgl.[8])

- jene zur Schätzung der Testkosten
- jene zur Bewertung der Testfälle
- jene zur Messung der Testfallüberdeckung
- jene zur Bewertung der Testeffektivität

und wesentlicher Faktoren, welche dabei eine Rolle spielen, sind beispielsweise bezüglich:

- Testkostenschätzung
  - Anzahl erforderlicher Testfälle (Testfallerstellung)

[8] Sneed, 2018.

  - Testbarkeit des Testobjekts selbst (Module, Datenbank, Schnittstellen, Pfade)
  - bisherige Testproduktivität (Aufwand Testdurchführung)
- Testfallbewertung
  - Komplexität der Testdaten, verschiedene Typen von Testdaten
  - Dichte der Testdaten, mögliche Kombinationen bei verschiedenen Typen
  - Anzahl der Testdaten, wie viele Parameter gibt es pro Testfall
  - wie viele Testfälle braucht man pro Funktion
- Testüberdeckung
  - Konzept (Regeln, UseCase, Bedingungen)
  - Benutzerhandbuch (Funktionen)
  - Code (Anweisung, Zweig)
  - Testfälle (Regressionstest, Sanitytest)
- Testeffektivität
  - Testqualität (welche Fehler werden wo von wem gefunden, Test/Produktion - Tester/Benutzer)
  - Vertrauen in ein System (wie viele Tests, wie oft, seit wann, mit welcher Fehlerrate)

Metriken zur Testkostenschätzung, Testfallbewertung und Testüberdeckung können vor Beginn der Tests bzw. zur Laufzeit erstellt und verwendet werden.
Im Gegensatz dazu sind Metriken zur Testeffektivität Metriken, welche erst nach der Durchführung der Tests, nachdem der Entwicklungsprozess für ein Produkt durchlaufen ist, zur Anwendung kommen.

Hier, in dieser Arbeit, finden Testmetriken Verwendung zur Interpretation in Kombination mit anderen Maßzahlen, welche aus dem Softwareentwicklungsprozess gewonnen werden. Diese:

- dienen nicht dazu, den Test zu beurteilen.
- lassen es nicht zu, Aussagen zum Testen selbst zu tätigen.
- unterstützen, basierend auf erstellten Maßzahlen, die Qualitätsbeurteilung
- unterstützen, basierend auf erstellten Maßzahlen, eine daraus mögliche Risikoabschätzung.

1.4 Risiko

1.4.1 Risikomanagement

Der Begriff Risikoabschätzung lässt sich in unmittelbaren Zusammenhang mit dem Begriff Risikomanagement bringen.

Ein Risikomanagement, das allgemein im Unternehmen aufgesetzt wird, bezieht Risiken, wie sie bei Softwareentwicklung und Softwareeinführung vorhanden sind, mit ein. Dazu gehören sowohl ein funktionierendes, risikobasiertes Softwarequalitätsmanagement, als auch ein solches Konfigurationsmanagement.

Typische Beispiele aus dem Softwareentwicklungsprozess, welche das Risiko beschreiben:

- spezielle Einstellungen oder Quellcode-Teile gehen verloren
- Dokumentation und Programmcode stimmen nicht überein
- Programmänderungen sind nur schwer nachvollziehbar
- mehrere Entwickler arbeiten an der gleichen Sache (redundante Wartung oder nicht abgestimmte, parallele Entwicklung)
- bereits korrigierte Fehler tauchen wieder auf
- ...

Testende Verfahren und davon Blackbox-Tests, als Teil von analytischen Qualitätssicherungsmaßnahmen (siehe Abbildung 1.5), spielen ebenfalls eine bedeutende Rolle.

1.4.2 Risikobasierter Ansatz

Werden Tests risikoorientiert geplant und durchgeführt, bringt dies den Vorteil, den Aufwand beim Testen sowohl wirtschaftlich, als auch zeitlich zu reduzieren. Gleichzeitig erhöht es die Wahrscheinlichkeit, kritische Fehler rechtzeitig zu entdecken.

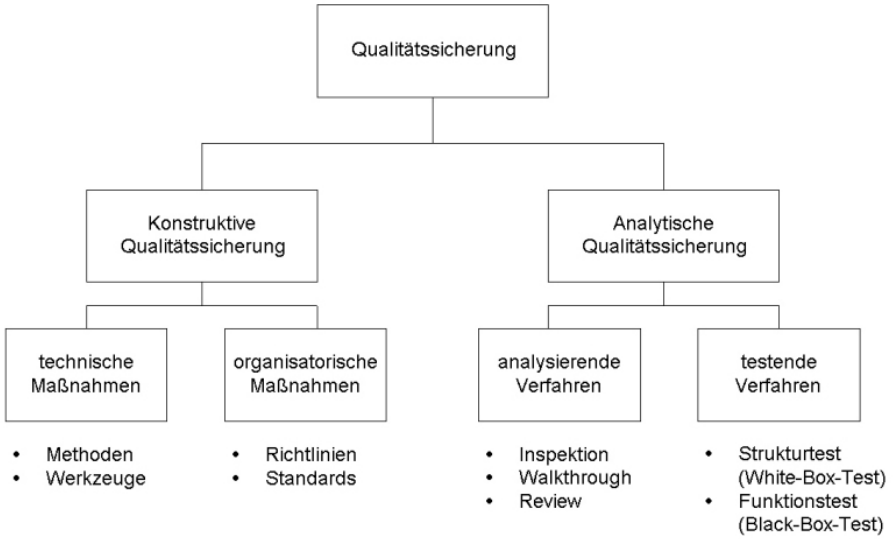

Abbildung 1.5: Qualitätssicherungsbaum (Gaulke, 2001)

Als ein solches risikoorientiertes Testkonzept beschreibt Markus Gaulke einen Managementkreislauf (siehe Abbildung 1.6), der folgende Phasen umfasst:

- Risikoerkennung
- Risikoanalyse
- Risikoverminderung
- Testkonzept
- Überwachung der Risiken während der Testdurchführung

Ein ähnliches Modell beschreibt Ståle Amland (siehe Abbildung 1.7), welches folgende Aktivitäten beinhaltet:

- Risikoerkennung - Testprodukt
- Risikoplanung - Testplan
- Risikobeurteilung - Kosten- und Eintrittswahrscheinlichkeits-Matrix
- Risikoreduzierung - Testen
- Risikoreport - Testmetriken
- Risikoprognose/Risikoabschätzung - Testmetriken

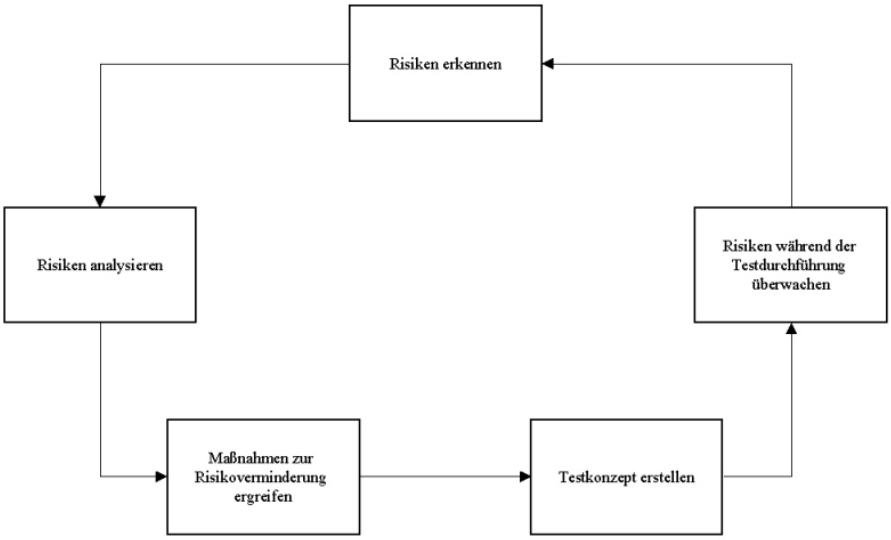

Abbildung 1.6: Risikoorientierte Testkonzeption (Gaulke, 2001)

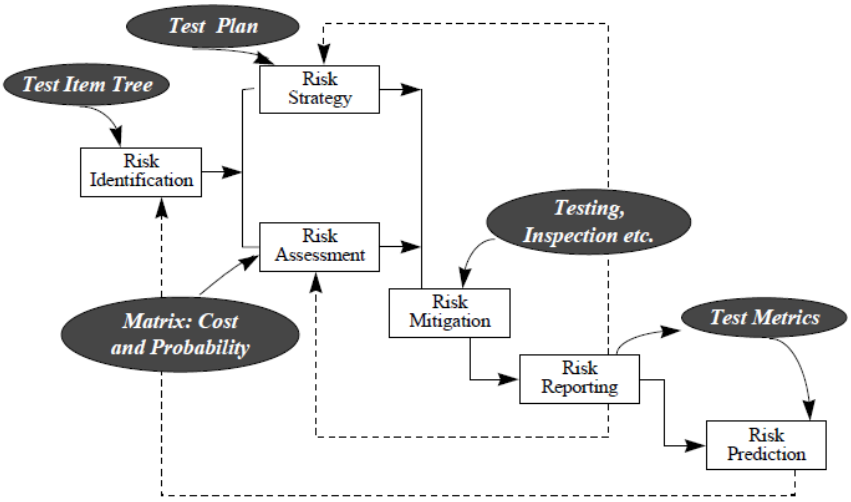

Abbildung 1.7: Risk analysis activity model (Amland, 8 - 12 November 1999)

2 Ausgangssituation

2.1 Unternehmen

Die Ausgangslage - auf welcher die Arbeit basiert

Das Unternehmen ist in Informations- und Kommunikationstechnologien (IKT) tätig, genauer gesagt, am Telekommunikationsmarkt.

Als Managementprozess ist das Wasserfallmodell als Vorgehensmodell im Einsatz. Planung und Durchführung der Softwareentwicklung erfolgen in enger Anlehnung an das V-Modell, wobei der Test als *Systemintegrationstest* (SIT [1]) ausgerichtet ist.

Anmerkung:
SIT wird im Unternehmen auf dreierlei Varianten verwendet

- SIT - die Durchführung des Systemintegrationstests
- SIT - die Umgebung, in welcher die Tests durchgeführt werden
- SIT - die Testmannschaft, welche die Tests ausführt

Der SIT (Durchführung) ist in ein eigenes Team ausgelagert, welches für alle Belange rund um das Testen zuständig ist.
Unit-Tests und grundlegende Integrationstests werden als erfolgreich durchgeführt vorausgesetzt, wenn neue Software geliefert wird.

Für die Durchführung der Tests stehen zwei getrennte Plattformen zur Verfügung:

[1] Definition „Systemintegrationstest“: „Testen der Integration von Systemen und Paketen, Testen der Schnittstellen zu einer externen Organisation (z.B. Electronic Data Interchange oder Internet).“ (German Testing Board, 2017-05-22)

- INT-Environment
  - Integration Testumgebung
  - Testumgebung verfügbar für die Entwicklung, zur Durchführung von Integrationstests
- SIT-Environment
  - Systemintegrationstest Testumgebung
  - Testumgebung verfügbar für das SIT-Team, zur Durchführung deren Tests

Zudem gibt noch zwei weitere Umgebungen - die DEV-Umgebung und die Pre-Prod-Umgebung.

Die DEV-Umgebung setzt sich aus den jeweils auf jedem Arbeitsplatz der Entwickler lokal installierten Entwicklungsinstanzen der Systeme zusammen. Auf diesen werden die Unit-Tests durchgeführt.
Die Pre-Prod-Umgebung ist eine StandAlone-Kopie der Produktion (Einzelplatzinstallation, ohne Anbindung an integrierte Systeme). Sie ist somit eine komplette Kopie aller Daten und Konfigurationen der Produktion, auf welcher Tests möglich sind, die sich in einer Testumgebung nicht vollständig nachstellen lassen (Datenbank-Performance, vergleichende Tests, ...).

Auf der INT-Umgebung ist es für die Entwicklung direkt und jederzeit möglich, neu eingesetzte Software für deren Integrationstest zu nutzen.
Beim Einsetzen neuer Software auf der SIT-Umgebung erfolgt dies durchgeführt von Operations nach vorheriger Freigabe durch das Testteam. Dabei werden fertig gelieferte, versionierte Softwarepakete verwendet und per *Statement of work* (SOW) durchgeführt.
Danach erfolgt auf der SIT-Umgebung per Systemintegrationstest (Systemtest) die Verifikation zur Qualitätssicherung. Weiters wird auch die Validation per UAT (User Acceptance Test - Abnahmetest) auf der SIT-Umgebung durchgeführt.

2.2 Administration, Tools

Im Bereich Test befindet sich das Tool *Hewlett Packard Application Lifecycle Management* (kurz HP ALM) Verwendung.

Es dient dazu,

- Dokumentation
- Planung
- Durchführung und
- Reports

der Tests und Testfälle abzuwickeln.

Die Informationsplattform des Unternehmens basiert auf der Softwareplattform *Microsoft Sharepoint*.
Dieses System wird unter anderem für folgende Aufgaben eingesetzt:

- Process Management Tool
- Demand Management Tool
- Effort Tracking Tool
- Defect Management Tool

Im operativen Bereich wird für das Applikations-, System-management und zum Verwalten und Überwachen der IT-Infrastruktur *Hewlett Packard OpenView* (kurz HP Openview) verwendet.
HP Openview ist somit das Change Management Tool und alle Systemänderungen an der Produktions-Landschaft werden darin administriert.

Sharepoint und HP Alm, sowie HP Openview dienen in dieser Arbeit daher als Hauptquelle jener Daten, welche für Auswertungen heran gezogen werden.

2.3 Begriffe

Bei den weiteren Betrachtungen in dieser Arbeit werden folgende englischsprachige Begriffe verwendet, die im Unternehmen ohne Übersetzung verwendet werden:

- *Demand* = Anforderung = Änderung = Wunsch eines Anforderers - benötigte neue Funktionalität - neues Produkt, ...
- *Workpackage* = Arbeitspaket

- *Task* = Aufgabe
- *Defect* = Fehler/Defekt (auch im Sinne von: fault, failure), gefundenes, dokumentiertes Fehlverhalten oder Abweichung von der Spezifikation
- *Testcase* = Testfall
- *Release* = Produktionsversion = Gesamtpaket aller Softwareversionen von Systemen, die in Produktion genommen werden, eine Release wird per Change in Betrieb genommen
- *Change* = Änderung an der Produktion = Inbetriebnahme von Produkten aus dem Softwareentwicklungsprozess
- *Operations* = operativ agierendes Team, welches die Produktivumgebung betreut und wartet
- *Dev* = Entwicklung

2.4 Ablauf, Organisation

Neue Änderungswünsche und Anforderungen, welche zur Umsetzung gebracht werden, werden als Demand erfasst. Somit beschreiben die Demands in Sharepoint Entwicklungsaufträge, welche den Softwareentwicklungsprozess durchlaufen.

Zur Abwicklung eines solchen Auftrags werden Workpackages für verschiedene Abteilungen oder Teams vergeben, diese in Tasks aufgeteilt und an jene Personen gebunden, welche für die Umsetzung verantwortlich sind.

Beim Demand werden sowohl die Spezifikation, das Design, Aufwände, Planung, . . . , als auch alle weiteren Dokumentationen erfasst.

Wird zu einem Demand ein Test angefordert, bekommt das SIT-Team ein Workpackage und alle involvierten Tester einen Task.

Die Tester erstellen dazu Testcases und speichern diese in HP Alm.

Während der Testdurchführung erfolgt die Dokumentation der Tests in HP Alm und die Dokumentation auftretender Defects in Sharepoint.
Defects sind somit typischerweise Demands eindeutig zugeordnet, jedoch lässt sich nicht unbedingt ein Zusammenhang zum zugehörigen Testcase herstellen.

Zur Planung eines Changes sind Demands zu Releases zusammengefasst, für welche es ein festgelegtes Datum der Produktivnahme gibt.

Test im Unternehmen

Der Test im Unternehmen umfasst, neben dem Systemintegrationstest (SIT) wie oben beschrieben, noch viele weitere Ausprägungen und findet ebenfalls in anderen Teams und Abteilungen statt.

Core Netzwerk Tests beispielsweise fallen hier ebenso darunter, wie Tests der Infrastruktur und des Netzwerks, als auch Tests von Terminals und Produkten, welche angeboten und verkauft werden, oder Tests von Data-Warehouse (DWH) oder des Enterprise-Resource-Planning-Systems (ERP-System).

Diese Tests sind organisatorisch eigenständig aufgestellt.

Das Kernaufgabengebiet von SIT und des SIT-Teams umfasst alle Business-Service-Support-Systems (BSS-Systeme). Solche sind unter anderen:

- Billing - Abrechnung , Fakturierung
- Customer Relationship Management (CRM) - Kundenpflege
- Middleware - allgemeine Dienste
- Shopsoftware
- ...

Sofern in dieser Arbeit in weiterer Folge von Test und Testen die Rede ist, bezieht sich dies nur auf den BSS-System Bereich und jene in diesem Bereich verwendete Daten.

3 Analyse

Für einen ausgewählten Zeitraum von 2 Jahren, den Jahren 2015 und 2016, stehen abgeschlossene Daten aus den Systemen HP Alm, Sharepoint und HP Openview zur Verfügung, welche als Quelle für weitere Auswertungen und Analysen dienen.
Diese Daten werden nicht mehr verändert und beziehen sich auf fertig gestellte und alle Prozessschritte bereits durchlaufene Änderungsanforderungen. Ebenso sind alle vorgenommenen Veränderungen an den Systemen bereits auf Produktion durchgeführt.

In diesem Kapitel werden anhand der eben genannten Daten Metriken beschrieben und definiert, sowie die daraus gewonnenen Kenngrößen erläutert.
Es gibt hierbei: Metriken, welche aus Testsicht kennzeichnend sind, um Rückschlüsse und Aussagen zu Qualität und Risiko im Sinne dieser Arbeit treffen zu können. Metriken, welche für sich selbst stehend Aussagekraft haben und welche in Kombination mit anderen eingesetzt werden. Metriken, die sich aus Kenngrößen in den eingesetzten Tools Sharepoint, HP Alm und HP Openview oder Parametern dieser Kenngrößen ableiten lassen.

In weiterer Folge verwendete Bezeichnungen, Schreibweisen oder Abkürzungen von Parametern sind im Original übernommen, wie sie in den erwähnten Tools definiert und eingesetzt sind.
Erläuterungen, Erklärungen und Übersetzungen finden an entsprechenden Stellen statt.

3.1 Kenngrößen

Für die weiteren Metriken werden folgende Begriffe als Kenngrößen definiert und in der Reihenfolge ihres Auftretens während der Softwareentwicklung gelistet:

- Demands
- Testcases
- Defects
- Changes

3.1.1 Demands

Die Kenngröße Demand ist für Tests von zentraler Bedeutung und beschreibt Änderungsanforderungen.
Diese werden vom SIT-Team während des SIT-Tests auf der SIT-Testumgebung getestet (siehe Absatz 2.4 Seite 25).

Es gibt verschiedene Typen und Größen von Demands. Beispielsweise kann es ein Analyse-Auftrag (Analysis Demand) oder ein Projekt (Project) und eine allgemeine Anforderung (General-Demand), nur eine kleine Änderung (Small-Demand) ebenso wie eine Anfrage (Standard-Request) sein. Eine Unterscheidung wird anhand des geschätzten Aufwands und der Anzahl betroffener Abteilungen getroffen, wodurch eine entsprechende Klassifizierung definiert ist.

Analysis-Demands und Project-Demands werden in dieser Analyse nicht berücksichtigt: Analysis-Demands haben keinen Testbezug und Project-Demands werden per Projektmanagement organisiert und abgewickelt.

Zeit- und Aufwandsplanung erfolgen pro Demand, involvierte Teams und/oder Systeme werden pro Demand erfasst.
Zur einfacheren organisatorischen Handhabung und Übersicht werden Demands in Workpackages und Tasks untergliedert. Anhand vorhandener Workpackages ist festgelegt, welche Teams involviert sind. Gesammelte Parameter von Tasks fließen automatisiert in die Parameter der Workpackages ein und finden deshalb keine gesonderte Berücksichtigung.

In diesem Zusammenhang ist Workpackage als Teilkenngröße von Demand zu betrachten. Workpackage ist mit seinen Parametern jedoch auch als eigene Kenngröße zu berücksichtigen.

Beispielhafte Eigenschaften dieser Kenngröße:

- zeitliche Abfolge
- Aufwandsschätzungen
- mitarbeitende Teams (Workpackages)
- involvierte Systeme
- Kategorisierung
- ...

3.1.2 Testcases

Die Kenngröße Testcases beschreibt vorhandene und dokumentierte Testfälle. Diese werden vom SIT-Team während des SIT-Tests auf der SIT-Testumgebung ausgeführt (siehe Absatz 2.4 Seite 25).

Testfälle werden individuell von den durchführenden Testern selbstständig erstellt. Dies erfolgt sowohl vor, während, als auch teilweise notwendig, erst nach der Testdurchführung.
Welche Testtechnik dabei zum Einsatz kommt, ist von Tester zu Tester verschieden. Testfälle sind nicht direkt miteinander vergleichbar, da keine einheitlichen Richtlinien zur Testfallerstellung und/oder Dokumentation vorgegeben sind.

Allen gemeinsam ist die Tatsache, dass es sich bei den Testfällen um funktionale (Kriterium), Black-box Tests (Methodik) auf Systemebene (Prüfebene) handelt.

Beispielhafte Eigenschaften dieser Kenngröße:

- welcher Demand wird getestet
- in welcher Testphase finden die Tests statt
- von wem (Dev, SIT, Business) werden die Tests durchgeführt
- welche Systeme stehen im Test
- welcher Geschäftsfall ist abgebildet
- ...

3.1.3 Defects

Die Kenngröße Defects beschreibt eingegebene und gehandhabte Fehler. Dies sind beispielsweise Fehler, welche vom SIT-Team während des SIT-Tests auf der SIT-Testumgebung im Zuge der Tests gefunden werden (siehe Absatz 2.4 Seite 25).

Oder es handelt sich um Fehler, welche auf Produktion aufgetreten sind und gelöst werden müssen. Ebenso zählen dazu Fehler, welche von Entwicklung im Zuge von deren Tests dokumentiert sind. Parametrisiert werden solche durch die Testphase (DEV, INT, SIT, PROD).

Ein wichtiger Parameter bei Defects beschreibt die Schwere des Fehlers (welche Auswirkungen der Fehler hat) und ist in 4 Stufen eingeteilt (*„Defect Severity“*):

- *critical* - der Fehler verhindert und blockiert die weitere Verwendung der Funktionalität und es ist zwingend erforderlich diesen Fehler zu beheben. Beispiel: Eingaben werden nicht verarbeitet
- *major* - der Fehler behindert die weitere Verwendung der Funktionalität - eine Behebung ist unbedingt notwendig. Beispiel: Eingaben werden nicht korrekt verarbeitet
- *minor* - die Funktionalität ist nicht wie angefordert gegeben, ist jedoch grundsätzlich gegeben und ein Arbeiten mit dem Fehler ist möglich. Beispiel: Eingaben werden nicht vollständig erarbeitet
- *cosmetical* - die Funktionalität ist gegeben, jedoch gibt es Unterschiede zur Spezifikation der Anforderung. Beispiel: Eingabefelder sind anders bezeichnet oder anders angeordnet

Ebenso haben zeitliche Abläufe bei der Handhabung und Behebung der Fehler im Zuge des Softwareentwicklungsprozesses Einfluss auf die geplante Testdurchführung.

Beispielhafte Eigenschaften dieser Kenngröße:

- zeitliche Abfolge
- betroffene Demands, Workpackes oder Produktionsprobleme (Service-Request = SR)
- an der Behebung beteiligte Teams

- in welcher Testumgebung oder Testphase entdeckter Defect ist (INT, SIT, Prod)
- Schwere, Dringlichkeit oder Bearbeitungsstatus des Fehlers
- ...

3.1.4 Changes

Die Kenngröße Changes beschreibt alle Änderungen, Anforderungen und Fehlerbehebungen, welche auf Produktion stattfinden. Für Teile davon (jene, zu denen ein SIT-Test angefordert ist), werden vom SIT-Team während des SIT-Tests auf der SIT-Testumgebung die Tests durchgeführt (siehe Absatz 2.4 Seite 25).

Changes beinhalten unter anderem ein vorgegebenes Datum, an welchem die Produktivnahme erfolgt. Bis zu diesem Datum muss der Softwareentwicklungsprozess gemäß Planung abgeschlossen sein und der Parameter Datum stellt meist einen unveränderlichen Fixwert dar. Welche Demands und Defects in welchen Changes Berücksichtigung finden, ist Teil der Releaseplanung und wird über Planungsmechanismen abgebildet.

Der Umfang und das Ausmaß dieser Kenngröße kann stark variieren.
So können Changes verhältnismäßig wenige Demands beinhalten, welche jedoch vom Aufwand her groß sind oder sich nur schwer abschätzen lassen. Es kann auch sein, dass Changes viele, kleinere Demands beinhalten, deren Aufwand gering und/oder auch relativ genau abschätzbar ist.

Daher richten sich Metriken speziell auf diese Kenngröße hin aus, um dafür die Qualität und das Risiko bewerten zu können.

Beispielhafte Eigenschaften dieser Kenngröße:

- Datum der Durchführung des Changes
- welche Systeme sind von der Änderung betroffen
- wie viele Systeme werden umgestellt
- geplante Demands
- inkludierte Defects
- ...

3.1.5 Kenngrößenparameter

Die folgenden Tabellen (siehe Tabelle 3.1, Tabelle 3.2, Tabelle 3.3, Tabelle 3.4 und Tabelle 3.5) listen die getroffene Auswahl an Parametern je Kenngröße, welche bei den Metriken Berücksichtigung finden.
Eine Liste aller gemessenen und verfügbaren Eigenschaften der Kenngrößen aus den jeweiligen Tools befindet sich im Anhang.

Created	Demand Category
Demand Number	Demand Phase
Demand Status	Effected Environments
Effort Estimation - Technology (MD)	ID
Modified	SIT required
Total effort Estimation (MD)	WP Number

Tabelle 3.1: Parameter der Kenngröße Demands

Created	Demand Category
Demand Number	ID
Modified	WP Actual Effort (MD)
WP Number	WP Status
WP Team Effort	

Tabelle 3.2: Parameter der Kenngröße Demands/Workpackages

Application	Criticality
Execution Status	Owner Group
Subject	Test Name
Test Phase	Test ID

Tabelle 3.3: Parameter der Kenngröße Testcases

Actual Effort	Affected Environment
Associated WP Number	Created
Defect Number	Defect Phase
Defect Severity	Defect Status
Demand Number	Environment Ressource
Found in Test Phase	Modified
No Fix Reason	Priority
Service Request Number	Team Ressource

Tabelle 3.4: Parameter der Kenngröße Defects

Affected Systems	Change ID
Demand Number	Defect Number
Risk Level	Summary

Tabelle 3.5: Parameter zur Kenngröße Changes

3.2 Metriken

Bezogen auf die zur Verfügung stehenden Daten, Kenngrößen und Parameter werden folgende Metriken definiert:

3.2.1 Demands pro Change

Inhalt:

$$\frac{\textit{Anzahl der Demands}}{\textit{Change}}$$

Beschreibung:

Inhalt dieser Metrik sind die Kenngrößen Demands und Changes.
Von der Kenngröße Demand finden dabei folgende Parameter Berücksichtigung:

- Demand Status
- Demand Phase
- Effected Environments
- SIT required

Die Relation wird über diese Parameter hergestellt:

- Demand Number

Interpretation:

Mit dieser Metrik wird erfasst,

- wie viele Demands,
- welche zur Implementierung freigegeben sind,
- sich in welcher Phase der Implementierung befindend
- und welche Systeme betreffend,
- für welche ein SIT-Test angefordert ist,

in einem Change involviert sind.

Der Parameter „Effected Environments“ beinhaltet dabei alle involvierten Systeme bzw. Entwicklungsteams (Collection, Billing, CRM, ...).

„Demand Status“ beschreibt die Managemententscheidung zur Anforderung (Registered, Declined, Deferred, . . .).
„Demand Phase“ zeigt, in welcher Entwicklungsstufe sich die Änderung befindet (New, Delivered, Gate1, . . .) und
„SIT required“ legt fest, ob für diese Änderung ein Test durch das SIT-Team festgelegt ist.

3.2.2 Defects pro Change

Inhalt:

$$\frac{\textit{Anzahl der Defects}}{\textit{Change}}$$

Beschreibung:

Inhalt dieser Metrik sind die Kenngrößen Defects und Changes.
Von der Kenngröße Defects finden dabei folgende Parameter Berücksichtigung:

- Defect Phase
- Defect Severity
- Defect Status
- Team Ressource
- Affected Environment

Die Relation wird über diese Parameter hergestellt:

- Defect Number

Interpretation:

Mit dieser Metrik wird erfasst,

- wie viele Defects,

- aus welchen Testphasen und
- von welchen Teams bearbeitet (=welche Systeme betreffend),

in einem Change involviert sind.

Weiters können durch die Verwendung der Parameter unterschiedliche Aussagen getroffen werden:
Der Parameter „Affected Environment“ dient der Unterscheidung, in welcher Testphase die Fehler gefunden werden (DEV, INT, SIT, PROD).
„Defect Phase“ beschreibt die Entwicklungsphase des Fehlers (New, In Progress, Fixed, No Fix Necessary, ...).
„Defect Severity“ legt die Schwere des Fehlers fest.
„Defect Status“ zeigt die Phase, in der sich der Defect befindet (Open, Verified, Closed, Documented).
„Team Ressource“ definiert, welches Entwicklungsteam mit dem Fehler befasst ist und somit ist indirekt auch bestimmt, welches System davon betroffen ist.

3.2.3 Defects pro Demand

Inhalt:

$$\frac{\textit{Anzahl der Defects}}{\textit{Demand}}$$

Beschreibung:

Inhalt dieser Metrik sind die Kenngrößen Defect und Demand.
Von der Kenngröße Defect finden dabei folgende Parameter Berücksichtigung:

- Defect Phase
- Defect Severity
- Defect Status
- Affected Environment

Die Relation wird über die folgenden Parameter hergestellt:

- Associated WP Number
- Service Request Number
- Demand Number

Interpretation:

Mit dieser Metrik wird erfasst,

- wie viele Fehler,
- bezogen auf die sich im Test befindlichen Demands,
- während der Tests

geöffnet wurden.

Weiters können durch die Verwendung der Parameter unterschiedliche Aussagen getroffen werden:
Der Parameter „Affected Environment“ dient der Unterscheidung, in welcher Testphase die Fehler gefunden werden (DEV, INT, SIT, PROD).
„Defect Phase“ beschreibt die Entwicklungsphase des Fehlers (New, InProgress, Fixed, No Fix Necessary, . . .).
„Defect Severity“ legt die Schwere des Fehlers fest und
„Defect Status“ zeigt die Phase, in welcher sich der Defect befindet (Open, Verified, Closed, Documented).

3.2.4 Testcases pro Demand

Inhalt:

$$\frac{\textit{Anzahl der Testcases}}{\textit{Demand}}$$

Beschreibung:

Inhalt dieser Metrik sind die Kenngrößen Testcases und Demands.
Von der Kenngröße Testcases finden dabei folgende Parameter Berücksichtigung:

- Application
- Criticality
- Execution Status
- Owner Group
- Subject

Die Relation wird über diese Parameter hergestellt:

- Test Name

Interpretation:

Mit dieser Metrik wird erfasst,

- wie viele Testcases,
- in welchem Ausführungsstatus und
- welche Systeme betreffend

zu einem Demand erzeugt wurden.

Weiters können durch die Verwendung der Parameter unterschiedliche Aussagen getroffen werden:
Der Parameter „Application“ bestimmt, welches System sich im Test befindet (CRM, Middleware, Billing, . . .).
„Criticality“ beschreibt die Dringlichkeit des Testcases (Important, Critical, Nice to have).
„Execution Status“ zeigt, in welchem Status der Ausführung sich der Testcase befindet (NoRun, Passed, Failed, Blocked, . . .).
„Owner Group“ legt fest, auf welcher Testumgebung der Test stattfindet (INT, SIT, . . .).
„Subject“ organisiert die Testcases durch hierarchische Untergliederung in

der Verzeichnisstruktur und es ist spezifiziert, dass dadurch die mit dem Demand beschäftigten Teams bezeichnet sind.

3.2.5 Teams pro Demand

Inhalt:

$$\frac{\textit{Anzahl der Teams}}{\textit{Demand}}$$

Beschreibung:

Inhalt dieser Metrik sind die Kenngrößen Workpackage und Demand.
Von der Kenngröße Workpackage finden dabei folgende Parameter Berücksichtigung:

- WP Team Effort
- WP Status

Die Relation wird über diese Parameter hergestellt:

- Demand Number

Interpretation:

Mit dieser Metrik wird erfasst,

- wie viele Workpackages,
- für welche ausführenden Teams (und somit System)
- sich in welcher Entwicklungsphase befindend,

es zu einem Demand gibt.

Weiters können durch die Verwendung der Parameter unterschiedliche Aussagen getroffen werden:
Der Parameter „WP Team Effort“ bestimmt, welches Team sich mit dem

Demand befasst und beauftragt ist (CRM, Middleware, Billing, Test ...).
„WP Status“ zeigt, in welchem Status der Ausführung sich das Workpackage befindet (New, InProgress, Delivered, Canceled).

3.2.6 Testaufwand zu Gesamtaufwand

Inhalt:

$$\frac{\textit{Aufwand für Test}}{\textit{Aufwand aller Beteiligten}}$$

Beschreibung:

Inhalt dieser Metrik sind die Kenngrößen Workpackage und Demand.
Von der Kenngröße Workpackage finden dabei folgende Parameter Berücksichtigung:

- WP Actual Effort (MD)

Die Relation wird über diese Parameter hergestellt:

- Demand Number

Interpretation:

Mit dieser Metrik wird erfasst,

- wie viele Workpackages,
- welcher Aufwand pro ausführendem Team angefallen ist,
- welcher Aufwand für den Test entstanden ist und
- wie groß der prozentuale Testaufwand ist, der

bei einem Demand entsteht.

3.2.7 Bearbeitungsdauer pro Demand

Inhalt:

$$\frac{Bearbeitungsdauer}{Demand}$$

Beschreibung:

Inhalt dieser Metrik sind die Kenngrößen Demand und Workpackage.
Von der Kenngröße Demand finden dabei folgende Parameter Berücksichtigung:

- Created
- Modified
- Demand Phase

Von der Kenngröße Workpackage finden dabei folgende Parameter Berücksichtigung:

- Created
- Modified
- WP Status
- WP Team Effort

Die Relation wird über diese Parameter hergestellt:

- Demand Number

Interpretation:

Mit dieser Metrik wird erfasst,

- wie lange ein Demand oder
- wie lange die zugehörigen Workpackages geöffnet sind,
- bei verschiedenen Systemen
- wie groß der prozentuale Testaufwand ist,

bis die Entwicklung abgeschlossen ist.

Weiters können durch die Verwendung der Parameter unterschiedliche Aussagen getroffen werden:
Der Parameter „WP Team Effort“ bestimmt, welches Team sich mit dem Demand befasst und beauftragt ist (CRM, Middleware, Billing, Test ...).
„WP Status“ zeigt, in welchem Status der Ausführung sich das Workpackage befindet (New, InProgress, Delivered, Canceled). „Demand Phase“ zeigt, in welcher Entwicklungsstufe sich die Änderung befindet (New, Delivered, Gate1, ...).

3.2.8 Bearbeitungsdauer pro Defect

Inhalt:

$$\frac{\textit{Bearbeitungsdauer Defect}}{\textit{Demand}}$$

Beschreibung:

Inhalt dieser Metrik sind die Kenngrößen Defect und Demand.
Von der Kenngröße Defect finden dabei folgende Parameter Berücksichtigung:

- Created
- Modified
- Defect Status
- Defect Severity
- Environment Ressource
- Found in Test Phase

Die Relation wird über diese Parameter hergestellt:

- Defect Number

Interpretation:

Mit dieser Metrik wird erfasst,

- wie lange ein Defect,
- von verschiedenen Teams bearbeitet,
- in verschiedenen Testphasen gefunden,

geöffnet ist, bis die Entwicklung abgeschlossen ist.

Weiters können durch die Verwendung der Parameter unterschiedliche Aussagen getroffen werden:
Der Parameter „Defect Status“ zeigt die Phase, in der sich der Defect befindet (Open, Verified, Closed, Documented).
„Defect Severity“ legt die Schwere des Fehlers fest.
„Environment Ressource“ bestimmt, welches System an dem Fehler arbeitet und somit, welches System involviert ist (CRM, Middleware, . . .).
Der Parameter „Found in TestPhase“ zeigt, in welcher Testphase die Fehler gefunden werden (DEV, INT, SIT, PROD).

4 Anwendung

Im Kapitel 3 Analyse (siehe Seite 27) wurden Daten aus den Tools Sharepoint, HP Alm und HP Openview, wie sie im Unternehmen vorhanden sind, analysiert und Metriken dazu vorgestellt.

In diesem Kapitel kommen die erwähnten Metriken zur Anwendung und werden eingesetzt.

Dafür wird aus den vorhandenen Daten ein Beispiel gewählt (eine komplette Release = Change), alle Parameter zu den Kenngrößen Demands, Defects und Testcase dieses Changes ermittelt sowie die Parameter des Changes selbst und daraus Kennzahlen zu den Metriken gewonnen.
Danach erfolgt eine Bewertung der gewonnenen Kennzahlen in Bezug auf Qualität und eine Diskussion zur Risikobeurteilung.

In die Bewertung und Diskussion werden vorhandene Kennzahlen mit einbezogen, welche im Unternehmen bereits während der Umsetzung im Einsatz waren. Diese sind im Abschnitt 4.3 Beispiele eingesetzter Metriken erläutert.

4.1 Definition Qualität

Qualität, im Zusammenhang mit SIT, wird im Unternehmen wie folgt definiert:

- alle Testcases des SIT sind gestartet und keiner ist geblockt
- es gibt keine Defects folgender severity:
  - critical
  - major

- zu allen Demands, bei denen SIT-Tests angefordert sind, gibt es Testergebnisse

Des Weiteren gibt es im Zuge des definierten Test Prozesses während des Softwareentwicklungsprozesses mehrere festgelegte QualityGates mit klaren Eintritts- und Austritts- Kriterien.

4.2 Beispiele ergänzender Metriken

Anhand eines gewählten Beispiels aus dem Pool analysierter Daten, werden die Werte der Kenngrößen und Parameter ermittelt und anschließend in die aufgestellten Metriken eingesetzt.

Als Beispiel dazu dient Release Schneeberg, Produktiv genommen am 27.01.2016.

4.2.1 Change CRQ000000011230

Die Daten dieses Changes wurden in HP Openview recherchiert. Jedem Change liegt ein Releaseletter bei, welcher alle Demands, Defects und betroffenen Systeme zusammenfasst sowie eine detaillierte Liste des geplanten Ablaufs zur Durchführung des Changes.

In der Zusammenfassung des Releaseletters sind 29 Demands und 139 Defects zu finden.
Gelistet beinhaltet dieser Change (CRQoooooo011230):
30 Demands, 139 Defects

Es gibt einen für diese Release geplanten Demand, welcher sich in der Demand Phase New befindet und im Demand Status declined.

Weitere entnommene Details zu dieser Release siehe Tabelle 4.1

Affected Systems	40
Change ID	CRQ000000011230
Demand Number	29
Defect Number	139
Risk Level	Risk Level 1
Summary	IT CRM Release Schneeberg - 27.01.16

Tabelle 4.1: Parameter zu Change CRQ000000011230

4.2.2 Beispiel Demands pro Change

Diese Metrik zeigt den Zusammenhang aller Demands innerhalb eines Changes in Bezug auf Systeme, welche eine Änderung erfahren, welche involviert sind und ob dazu Tests angefordert sind oder nicht.

In Abbildung 4.1 ist als Gesamtüberblick, ohne den Einsatz von Filtern, dargestellt, welche Systeme (sofern angegeben) von den Demands betroffen sind. Dabei kommt es auch zur Mehrfachnennung von Systemen beim selben Demand, was darauf hinweist, dass den Systemintegrationstests eine besondere Rolle zukommt.
Solche systemübergreifenden Tests sind im Entwicklungsprozess im SIT erstmalig vorgesehen.

Des Weiteren lässt sich erkennen, wie viele Demands in dem jeweiligen System geliefert werden und welche Systeme von den Demands bekanntermaßen betroffen sind.
Auffallend in der Abbildung ist die große Zahl von Demands (17 von 29), bei welchen keine solche Information angegeben ist. Dies kann sein, wenn sich keine Zuteilung treffen lässt oder die Zuteilung nicht getroffen wurde.

Bei 25 von 29 Demands ist ein Systemintegrationstest angefordert und ein Blick auf jene, für die kein SIT vorgesehen ist, zeigt, dass es sich dabei um Umstellungen ohne funktionale Änderung handelt.
Dadurch lässt sich aus Sicht des SIT beurteilen, dass bei diesem Change eine breite Testabdeckung gegeben ist.

Die nicht vorhandenen Informationen betreffend involvierter Systeme erschweren es, Zusammenhänge zwischen den Demands zu erkennen. Das

sollte dazu führen, dass diese fehlenden Informationen nachgefragt werden und nachgegangen wird, weshalb dies nicht geschehen ist.

Durch den Einsatz der weiteren Parameter in dieser Metrik (Demand Status, Demand Phase) lässt sich die Entwicklungssituation der Demands beobachten. Dies ist im angeführten Beispiel nicht zielführend, weil es sich um abgeschlossene Daten nach bereits erfolgter Inbetriebnahme handelt. Oder man lässt diese, nach einmaliger Beurteilung betreffend der Demands, zu denen kein SIT angefordert ist, bei den weiteren Betrachtungen weg. Es besteht die Möglichkeit, die Metrik gefiltert per Demands nach System zu verwenden.

Qualität

Die Anforderung eines Systemintegrationstests zu 25 von 29 Demands dieses Changes ergibt eine Testabdeckung von $\approx 86\,\%$ aller Änderungen an den Systemen. Tatsache ist, dass die weiteren 4 Demands nicht in den Bereich von SIT fallen.
Die Aussagekraft zur Qualität der Demands dieses Changes anhand von Testergebnissen aus der Testdurchführung sind entsprechend zu beurteilen und haben entsprechende Aussagekraft.
Es gibt für diese Metrik keine Klassifizierung der Qualität in Zahlen (prozentuelle Bewertung), siehe 4.4.

Risiko

Die fehlenden Daten betreffend involvierter Systeme (Effected Environments) bei den Demands erfordern eine manuelle, nachträgliche Abschätzung und Einteilung durch die Tester.
Eine solche Einteilung ist notwendig, da das Risiko beispielsweise gänzlich unterschiedlich zu bewerten ist, falls alle Demands ohne eingetragenem System das gleiche System betreffen oder sich alle Demands auf verschiedene Systeme aufteilen.

Dies erfolgt während der Testdurchführung und fließt somit nicht in die Metrik ein.

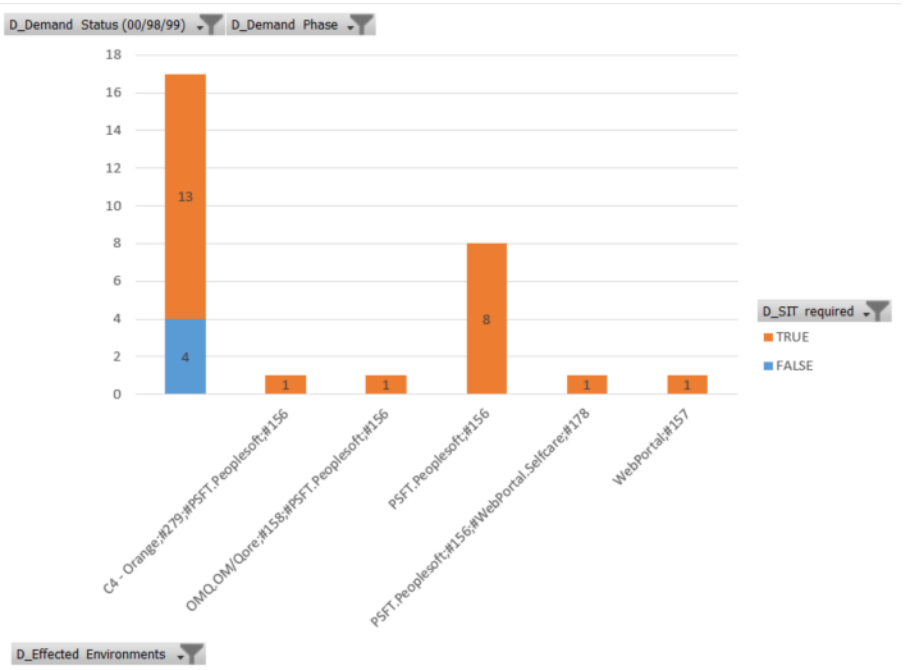

Abbildung 4.1: Chart Demands pro Change

Es gibt für diese Metrik keine Klassifizierung des Risikos in Zahlen (prozentuelle Bewertung), siehe 4.4.

4.2.3 Beispiel Defects pro Change

Diese Metrik zeigt den Zusammenhang aller Defects eines Changes, welche auch in Produktion auftreten oder aufgetreten sind, von welchen Teams diese bearbeitet werden und mit welchem Change diese behoben werden.

Abbildung 4.2 zeigt, von welcher Team Ressource wie viele Defects welcher Severity in Bearbeitung waren und gelöst wurden.
Affected Environment dieser Defects ist jeweils Prod, da diese aus vorhergehenden Releases stammen und im Zuge einer Risikobeurteilung anderer Releases produktiv genommen wurden und deshalb produktionsrelevant

sind. Service Requests, gemeldete Probleme, welche in Produktion gefunden werden, sind ebenfalls Teil dieser Metrik.
Nicht enthalten sind Defects, welche im Zuge der aktuellen Release (dieses Beispiels) während der Testdurchführung protokolliert wurden. Diese sind Teil der Metrik Defects pro Demand.

Zwischen dem Parameter Team Ressource dieser Metrik und dem Parameter Effected Environment der Metrik Demands pro Change besteht ein direkter Zusammenhang. Die Systeme sind bearbeitenden Teams zugeteilt. So werden beispielsweise Demands des Parameters Effected Environments PSFT.Peoplesoft vom selben Team bearbeitet, welches Defects des Parameters Team Ressource IT CRM - Development bearbeitet.

Von den 129 Defects dieses Changes sind 127 Defects SIT-relevant und zwei betreffen andere Abteilungen (Team Ressource IT BI und IT MP). Somit durchlaufen 127 der 129 Defects den Systemintegrationstest und werden vom SIT-Team verifiziert.

Durch den Einsatz der weiteren Parameter in dieser Metrik (Defect Status, Defect Phase) lässt sich die Bearbeitung der Defects beobachten.

Qualität

127 von den 129 produktiv genommenen Defects werden vom Systemintegrationstest verifiziert, das sind $\approx 98\,\%$. Tatsache ist, dass zwei davon nicht in den Bereich von SIT fallen.
Die Aussagekraft zur Qualität die Defects betreffend dieses Changes anhand von Testergebnissen aus der Testdurchführung sind entsprechend zu beurteilen und haben entsprechende Aussagekraft.

Es gibt für diese Metrik keine Klassifizierung der Qualität in Zahlen (prozentuelle Bewertung), siehe 4.4.

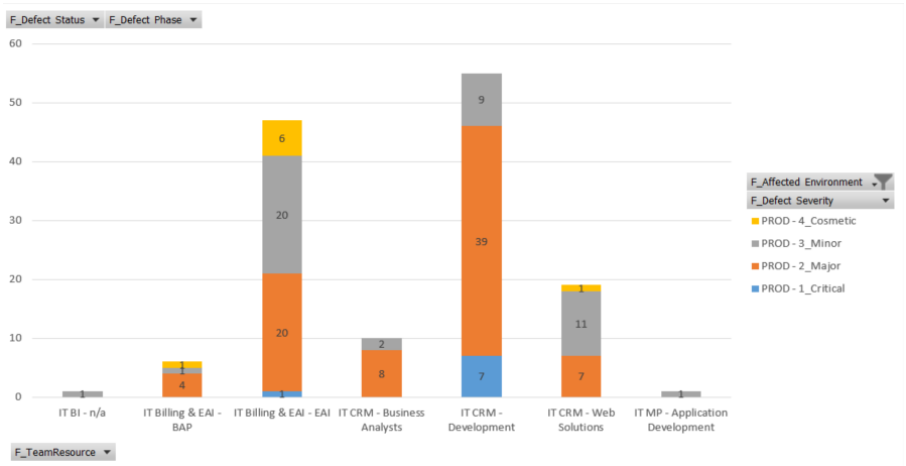

Abbildung 4.2: Chart Defects pro Change

Risiko

Von 7 bearbeitenden Teams dieser Defects werden Defects von 5 Teams dem Systemintegrationstest unterzogen, das sind $\approx 71\,\%$.
Zusätzlich zu berücksichtigen ist, dass von den zwei nicht SIT-relevanten Defects jeweils 1 Defect geliefert wird.
Die Aussagekraft zum Risiko die Defects betreffend bei diesem Change sind entsprechend zu beurteilen.

Es gibt für diese Metrik keine Klassifizierung des Risikos in Zahlen (prozentuelle Bewertung), siehe 4.4.

4.2.4 Beispiel Defects pro Demand

Diese Metrik zeigt im Zusammenhang alle Defects je Demand, unterteilt nach Workpackages, welche im Zuge des Systemintegrationstests für diesen Change bearbeitet wurden.
Des Weiteren ersichtlich sind die Anzahl und Schwere dokumentierter Fehler je System/Entwicklungsteam. Der Parameter ServiceRequestNumber

ist der besseren Übersichtlichkeit wegen als Filteroption ausgeführt. Dies wird so gelöst, da solche Service Requests als Sonderfall im Zuge des Testprozesses gesehen werden und dennoch mit gehandhabt und dokumentiert sind.

In Abbildung 4.3 zeigt sich, dass es zu den neuen Demands dieses Changes Änderungen gibt, welche von jeweils nur einem Team bearbeitet werden (Workpackage = Team) und ebenso welche, die von bis zu 4 Teams bearbeitet werden.
Erkennbar ist, dass die Anzahl gefundener Defects höher liegt, wenn mehrere Teams an einem Demand arbeiten.

Zu den 25 sich im SIT befindenden Demands wurden bei 19 Demands Defects protokolliert. Bezüglich der 6 weiteren Demands liegen keine Informationen vor.

Durch den Einsatz der weiteren Parameter in dieser Metrik (Defect Status, Defect Phase, Affected Environment, ServiceRequestNumber) lässt sich die Veränderung dieser Metrik während des Testens beobachten.

Qualität

Im Zuge der Testdurchführung von 25 Demands wurden bei 19 Demands und 25 daran arbeitenden Teams insgesamt 77 Defects bearbeitet.
Davon waren 5 critical, 44 major, 23 minor und 5 cosmetic. Auf zwei Demands, jene mit mehr als einem daran beteiligten Team, entfallen davon 39 Defects: 1 critical, 28 major, 6 minor und 2 cosmetic. Die Aussagekraft zur Qualität der Demands dieses Changes anhand dokumentierter Fehler durch die Testdurchführung ist entsprechend zu beurteilen.

Es gibt für diese Metrik keine Klassifizierung der Qualität in Zahlen (prozentuelle Aufteilung), siehe 4.4.

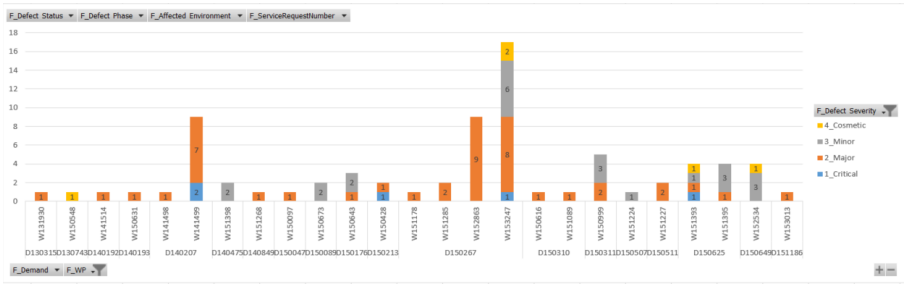

Abbildung 4.3: Chart Defects pro Demand

Risiko

Die Testdurchführung des Systemintegrationstests aller Demands, zu denen ein SIT angefordert wurde, ergab 5 critical und 44 major Defects, bei 19 Demands.
Alle Defects befinden sich im Defect Status Closed und der Defect Phase Fixed oder No Fix Necessary.

Es gilt, eine Abschätzung des Risikos anhand dieser Fakten durchzuführen.

Es gibt für diese Metrik keine Klassifizierung des Risikos in Zahlen (prozentuelle Aufteilung), siehe 4.4.

4.2.5 Beispiel Testcases pro Demand

Diese Metrik liefert einen Gesamtüberblick aller durchgeführten Testcases pro System/Entwicklungsteam, gruppiert nach Demands und Ausführungsstatus des Testfalls - siehe Abbildung 4.4.

In den analysierten Daten sind alle Testcases enthalten, welche zu Demands dieses Changes erstellt wurden:

- Integrationstests von Entwicklung
- Systemintegrationstests von SIT
- Useracceptancetests von Business

- Systemintegrationstests weiterer Teams (Products)

Das sind 1317 Testfälle, 1246 Passed, 31 N/A, 20 No Run, 16 Failed und 2 Blocked

Diese gefiltert auf SIT-Relevanz ergibt:
Zu 24 der 25 getesteten Demands gibt es Testfälle, insgesamt 1022. Davon sind 966 Passed, 29 N/A, 8 No Run, 15 Failed und 2 Blocked dokumentiert. 10 Systeme befanden sich im Test (qcApplication), eine große Gruppe davon ist nicht kategorisiert (240 von 1022).
Die größte Gruppe umfasst 470 der 1022 Testfälle, jene im Ausführungsstatus Blocked betreffen den selben Demand und das selbe System/Team.
Bei den Testfällen im Ausführungsstatus Failed sind 5 Demands gelistet, wobei 3 dieser 5 Demands das selbe System betreffen.

Zu dem fehlenden Demand (24 von 25 sind dokumentiert) liegen keine weiteren Informationen vor.
Durchgeführte Useracceptancetests ergaben folgendes Ergebnis: 25 Testfälle, 10 Passed, 2 N/A, 12 No Run, 1 Failed.

Qualität

24 der 25 getesteten Demands liefern Testfälle, insgesamt 1022, davon 966 Passed, 29 N/A, 8 No Run, 15 Failed und 2 Blocked. Bei 9 Systemen lässt sich die Zuteilung durchgeführter Tests erkennen. Diese Ergebnisse der Metrik fließen in die Qualitätsbeurteilung ein.

Es gibt für diese Metrik keine Klassifizierung der Qualität in Zahlen (prozentuelle Unterteilung), siehe 4.4.

Risiko

Von den 1022 Testfällen befinden sich 8 im Status No Run, das sind beispielsweise $\approx 0{,}8\,\%$. Von einem Demand gibt es keine Ergebnisse und durchgeführte Useracceptancetests ergaben gemischte Ergebnisse.

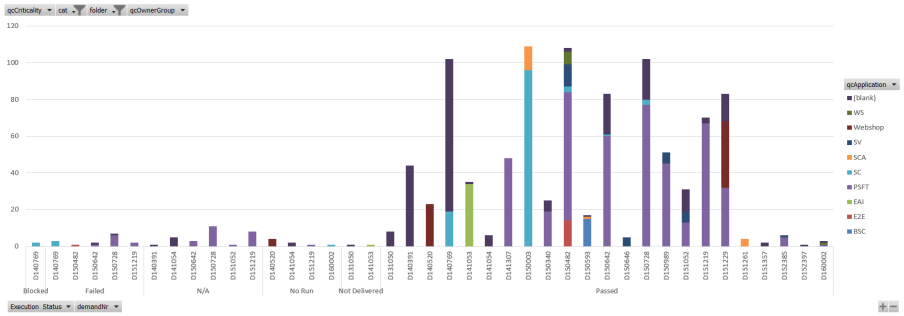

Abbildung 4.4: Chart Testcases pro Demand

Anhand dieser Fakten gilt es, eine Abschätzung des Risikos aus den gewonnenen Statistiken zur Anzahl erstellter und ausgeführter Testfälle zu machen.

Es gibt für diese Metrik keine Klassifizierung des Risikos in Zahlen (prozentuelle Unterteilung), siehe 4.4.

4.2.6 Beispiel Teams pro Demand

Diese Metrik liefert einen Überblick, wie viele und welche Teams bei diesem Change involviert sind, sowie welche Teams an welchen und wie vielen Demands arbeiten.

Abbildung 4.5 beinhaltet dabei alle Teams, auch jene, dessen Workpackages nicht durch Systemintegrationstests verifiziert werden, beispielsweise IT MP, IT ERP, ...
Zusammen sind das 13 Teams (inklusive SIT-Team), welche an diesem Change beteiligt sind und die 29 Demands sind auf insgesamt 106 Workpackages aufgeteilt.

Dabei kommt es vor, dass Teams zum gleichen Demand mehrere Workpackages bekommen, sei es aus zeitlichen, thematischen oder organisatorischen Gründen. Oft betrifft diese die dem Demand vorausgehende und/oder begleitende Businessanalyse. Beispielsweise hat ein Demand 8 Workpackages, während nur 5 Teams daran beteiligt sind.

Filtert man diese auf SIT-Relevanz, sind es 6 Teams (inklusive SIT-Team), die an 93 Workpackages arbeiten.
Ohne das Team der Businessanalyse sind es 5 Teams, die gemeinsam an 70 Workpackages arbeiten. Und ohne dem SIT-Team, welches zu jedem Demand, für welchen ein Systemintegrationstest angefordert ist, an wenigstens einem Workpackage arbeitet (insgesamt sind es 26 Workpackages zu 25 Demands), sind es 4 Teams mit 44 Workpackages.

Anhand dieser Metrik lässt sich erkennen, welche Auslastung die einzelnen Teams haben. Diese 4 Teams teilen sich die 44 Workpackages folgendermaßen auf: 16, 13, 8 und 7 Workpackages.

Qualität

Die Anzahl an Workpackages, welche pro Team innerhalb der vorgegebenen Entwicklungszeit zu erledigen sind, findet in der Qualitätsbeurteilung in Zusammenhang mit anderen Metriken Berücksichtigung. Es gibt für diese Metrik keine Klassifizierung der Qualität in Zahlen, siehe 4.4.

Risiko

Die Anzahl an Workpackages, welche pro Team innerhalb der vorgegebenen Entwicklungszeit zu erledigen sind, lässt sich bei der Risikoabschätzung berücksichtigen.
4 Teams teilen sich die 44 Workpackages zu den Teilen 16, 13, 8 und 7 auf, das entspricht $\approx 36\,\%$, $\approx 30\,\%$, $\approx 18\,\%$ und $\approx 16\,\%$.

Es gibt für diese Metrik keine Klassifizierung des Risikos in Zahlen (prozentuelle Aufteilung), siehe 4.4.

4.2.7 Beispiel Testaufwand zu Gesamtaufwand

Diese Metrik zeigt den prozentuellen Aufwand des Testteams (quantifiziert in Mann-Tagen = MD) in Relation zum Gesamtaufwand aller anderen, an jeweiligen Demands beteiligten Teams.

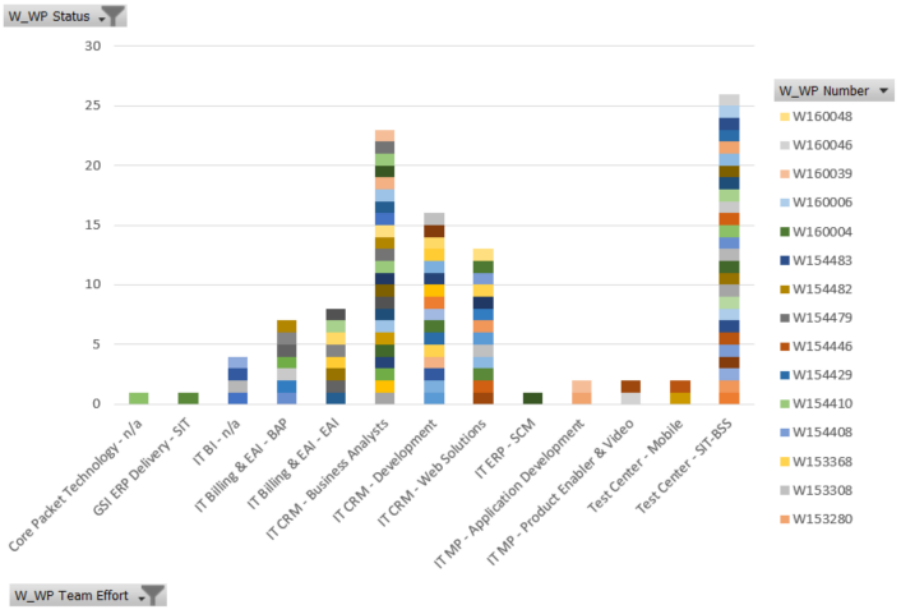

Abbildung 4.5: Chart Teams pro Demand

Für die Darstellung wurde eine logarithmische Skala gewählt und sie beinhaltet auch jene Demands, für welche kein SIT angefordert ist.

Der Testaufwand reicht dabei von 0,18 % bis zu 14,46 %, was 0,25MD bis zu 20,25MD entspricht.

Qualität

Diese Metrik hat für sich allein stehend wenig Relevanz zur Beurteilung der Qualität.
In Zusammenhang mit weiteren Metriken, zum Beispiel der Bearbeitungsdauer von Demands oder Defects und die Metrik Defects pro Demand dazu genommen, lassen sich aus Fakten der Vergangenheit Rückschlüsse für zukünftige Demands ziehen.

Risiko

Diese Metrik bietet für sich allein stehend wenig Mehrwert für eine Risikoabschätzung.
Durch eine ständige Beobachtung dieser Kenngröße und plötzlich auftretender Abnormität beispielsweise, stellt diese eine wertvolle Ergänzung dar.

4.2.8 Beispiel Bearbeitungsdauer pro Demand

Diese Metrik liefert den Zusammenhang der Bearbeitungsdauer des Workpackages für den Systemintegrationstest in Relation zur Bearbeitungsdauer, der zum gleichen Demand gehörenden Workpackages, der anderen Teams. Die Berechnung der Bearbeitungsdauer erfolgt in Tagen, wobei mithilfe von Zeitstempeln der Workpackages gearbeitet wird. Das wird erreicht, indem der Zeitraum zwischen Erstellung des Workpackages (created) und der letztmaligen Änderung (modified) errechnet wird.

In Abbildung 4.7 ist als Gesamtüberblick aller Demands dargestellt (alle 30), wie groß der Anteil des Workpackages für Test ist.

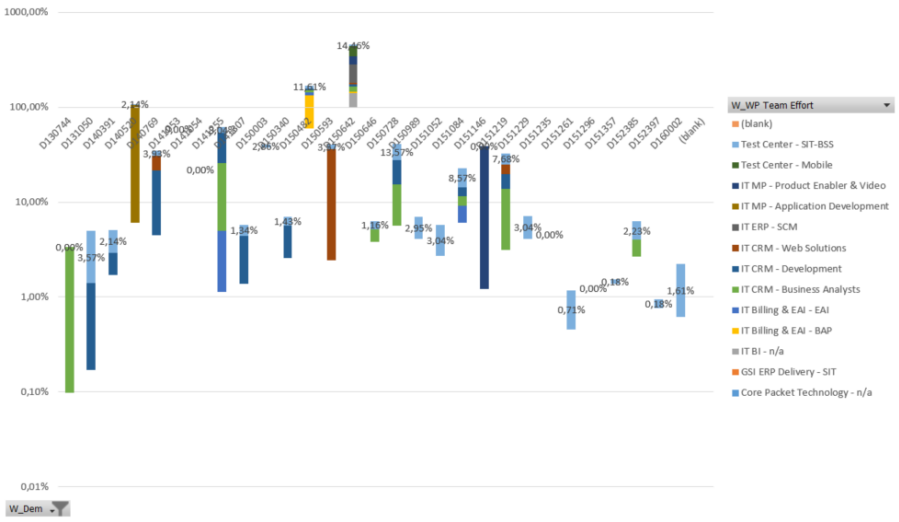

Abbildung 4.6: Chart Testaufwand zu Gesamtaufwand

Folgende Annahme: Alle Workpackages werden beim Anlegen eines Demands festgelegt und erst, wenn der Test abgeschlossen ist, schließen die Entwicklungsteams ihre Workpackages. Unter dieser Annahme sollte der prozentuale Anteil von Test, dem prozentualen Anteil der Anzahl von Test Workpackages zur Anzahl aller Workpackages entsprechen. Somit wären das bei 2 Workpackages $\approx 50\,\%$, bei 3 $\approx 33\,\%$ oder bei 5 Workpackages $\approx 20\,\%$.

Sollte es diesbezüglich Ausnahmen geben, braucht es dafür eine tiefergehende Analyse, wie es dazu gekommen ist.

Die Abbildung weist alle 30 Demands aus und pro Demand die daran arbeitenden Teams.
Filtert man diese nach SIT-Relevanz und ohne Businessanalyse, verbleiben wieder 4 Teams (5 inklusive Systemintegrationstest) mit Anteilen von 0 % bis 100 %.
100 % ergeben sich bei Demands, zu denen es nur ein Workpackage für SIT gibt (in dem Beispiel bei Small-Demands), 0 % bei jenen, zu denen kein SIT angefordert ist.

Eine Ausnahme ist erkennbar: Bei einem Demand mit 3 Workpackages

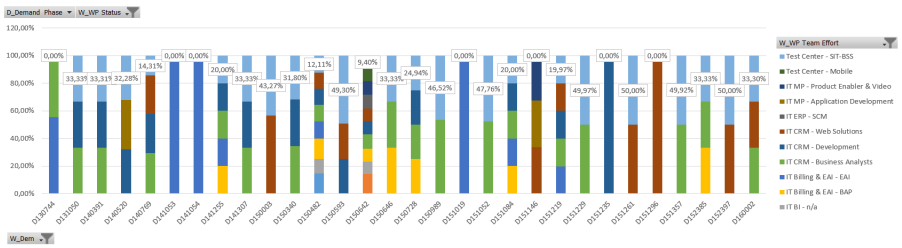

Abbildung 4.7: Chart Bearbeitungsdauer pro Demand

kommt es zu einer Aufteilung 40-40-20 (20 % Test). Dies ist in einer nachträglichen Anfrage für SIT begründet, wodurch dieses Workpackage erst zu einem späteren Zeitpunkt angelegt wurde.

Durch den Einsatz der weiteren Parameter in dieser Metrik (Demand Phase, WP Status) lässt sich die Übersichtlichkeit bei der Beobachtung der Metrik erhöhen.

Qualität

Diese Metrik hat für sich allein stehend wenig Relevanz zur Beurteilung der Qualität.
Obwohl sich Abnormität erkennen lässt, ist dies noch kein verwendbarer Indikator für Qualität. In Zusammenhang mit weiteren Metriken, zum Beispiel der Metrik Testaufwand zu Gesamtaufwand und die Metrik Defects pro Demand dazu genommen, lassen sich aus Fakten der Vergangenheit Rückschlüsse für zukünftige Demands ziehen.

Risiko

Diese Metrik bietet für sich allein stehend wenig Mehrwert für eine Risikoabschätzung.
Durch eine ständige Beobachtung dieser Kenngröße und plötzlich auftretende Abnormität beispielsweise, stellt diese eine wertvolle Ergänzung dar.

4.2.9 Beispiel Bearbeitungsdauer pro Defect

Diese Metrik stellt dar (in MD), wie lange Defects, welche während des Systemintegrationstests der Release geöffnet sind, von welchen Teams in Bearbeitung sind.
Die Berechnung der Bearbeitungsdauer erfolgt in Tagen, wobei mithilfe von Zeitstempeln der Defects gearbeitet wird. Das wird erreicht, indem der Zeitraum zwischen dem Öffnen des Defects (created) und der letztmaligen Änderung (modified) errechnet wird.

Abbildung 4.8 stellt einen Gesamtüberblick aller Defects dar und welche Teams an diesen arbeiten.
Dabei kommen erstmalig weitere Teams (= Environment Ressource) vor, da sich diese Metrik nicht nur auf getestete Demands der Release bezieht, sondern auf alle Defects, welche in der Zeit des SIT dieser Release gehandhabt wurden (= Found in TestPhase).

Die Bearbeitungszeiten von Defects beträgt oftmals auch Jahre. Um die Übersichtlichkeit des Graphen zu erhöhen, ist deshalb die Ordinate des Graphen auf 300 MD beschränkt, um kürzere Bearbeitungszeiten erkennbar darzustellen.
Defects mit längeren Bearbeitungszeiten sind daher auf 300 MD limitiert dargestellt.

Durch den Einsatz der weiteren Parameter in dieser Metrik (Defect Status, Defect Severity) lässt sich die Übersichtlichkeit bei der Beobachtung der Metrik erhöhen.

Qualität

Diese Metrik hat für sich allein stehend wenig Relevanz zur Beurteilung der Qualität.
In Zusammenhang mit weiteren Metriken, zum Beispiel der Metrik Testaufwand zu Gesamtaufwand und die Metrik Defects pro Demand dazu genommen, lassen sich aus Fakten der Vergangenheit Rückschlüsse für zukünftige Demands ziehen.

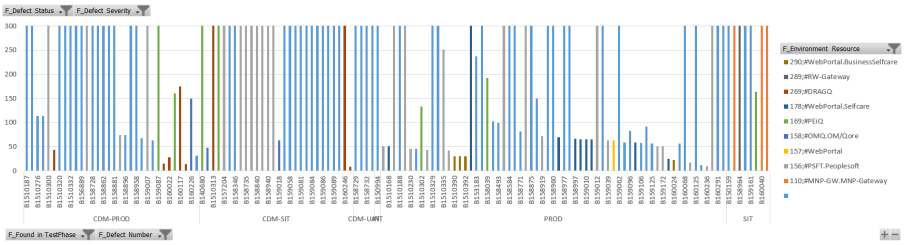

Abbildung 4.8: Chart Bearbeitungsdauer pro Defect

Risiko

Diese Metrik bietet für sich allein stehend wenig Mehrwert für eine Risikoabschätzung.
Durch eine ständige Beobachtung dieser Kenngröße und plötzlich auftretende Abnormität beispielsweise, stellt diese eine wertvolle Ergänzung dar.

4.3 Beispiele eingesetzter Metriken

Das Testmanagement hat eine Vielzahl von Metriken im Einsatz, mit welchen die Planung, Administration, Durchführung oder Reports von Tests unterstützt wird.

Aus den dazu aufgestellten Übersichten lässt sich fortlaufend ein Gesamtbild sowohl bezüglich des Tests, als auch dem dazu stattfindenden Entwicklungsfortschritt gewinnen. Des Weiteren werden umfangreiche Statistiken geführt, um diverse Trends und Entwicklungen zu beobachten.

4.3.1 Demand Dashboard

Ein Beispiel dazu stellt ein Demand test-lifecycle dashboard dar, über welches ein Überblick für Demands zu folgenden Eigenschaften vorhanden ist:

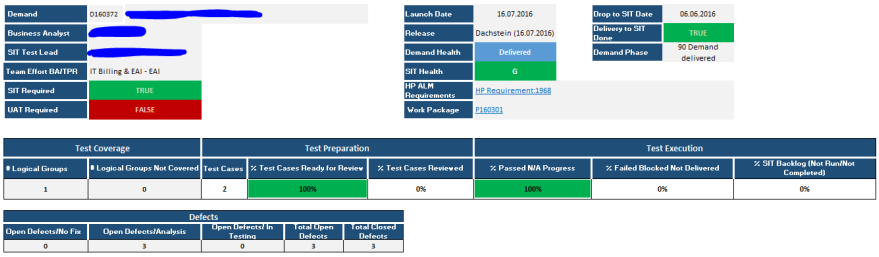

Abbildung 4.9: Demand Dashboard

- Test Coverage - basierend auf Anforderungen, welche in sogenannten LogicalGrp abgebildet sind
- Test Preparation - wie läuft die Testcase Vorbereitung (reviewed, ready, . . .)
- Test Execution - Status zur Testdurchführung (NoRun, Passed, InProgress, . . .)

Beispiel:
siehe Abbildung 4.9

4.3.2 Defect Report

Der Defect Report liefert eine statistische Trendbeobachtung aller Defects. Diese findet beispielsweise folgendermaßen statt:

- per Year
- per Severity
- per Team Ressource
- per Environment Ressource
- per Anzahl offener Defects per Severity

Beispiele dazu:
siehe Abbildung 4.10, siehe Abbildung 4.11 oder
siehe Abbildung 4.12

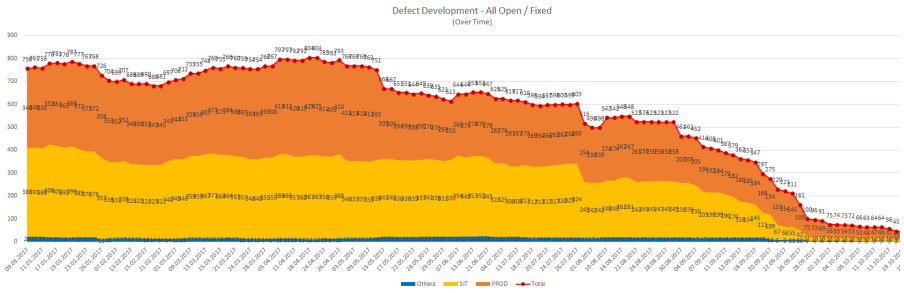

Abbildung 4.10: Defect Development All Open / Fixed, over Time

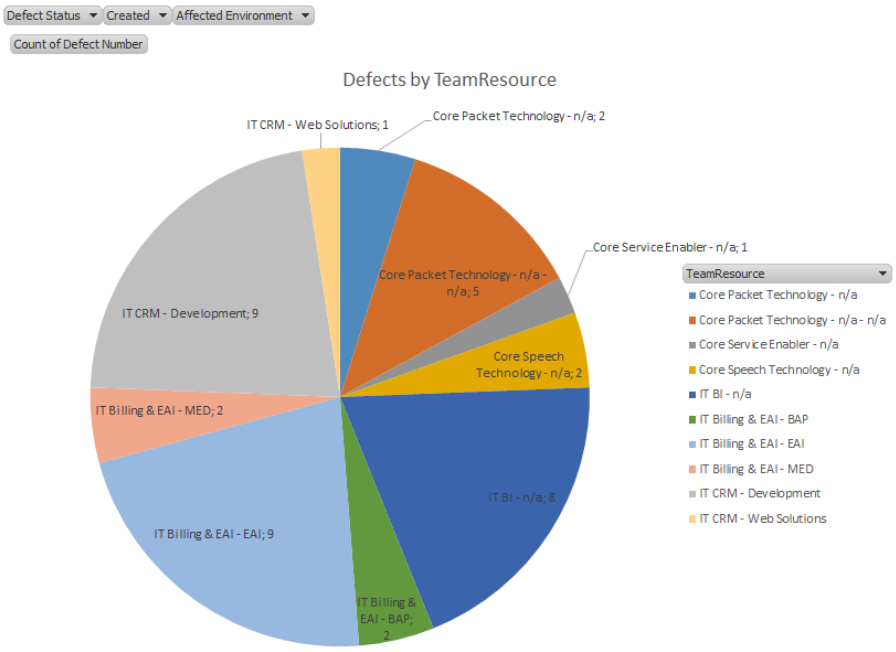

Abbildung 4.11: Defects by Team Ressource

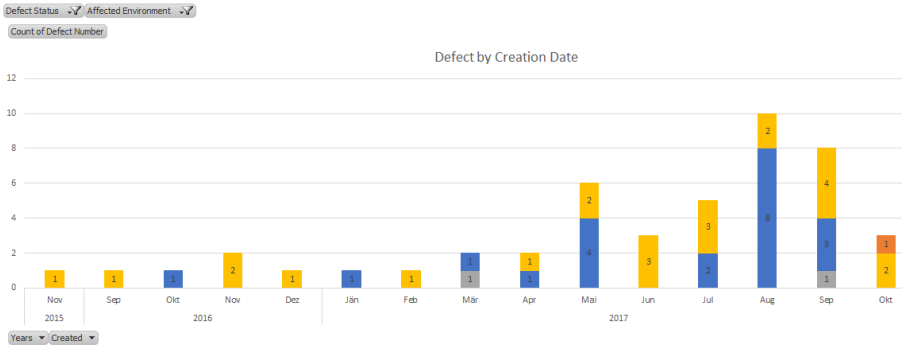

Abbildung 4.12: Defect by Creation Date

4.3.3 Defect Status

Der Defect Status liefert eine Auswertung zu Defects bezüglich:

- per project
- per Creation Date
- per Team Ressource
- Anzahl offener Defects per Severity per month

Beispiel:
siehe Abbildung 4.13

4.3.4 Ressourcen Planung

Zur Planung des Einsatzes verfügbarer Ressourcen (Tester) werden pro Release (Change) aktuelle Aufwände den verfügbaren Aufwänden gegenübergestellt. Der Aufwand wird in Mann-Tagen (MD) gezählt.

Dies erfolgt anhand folgende Parameter:

- Summe Actual Effort (MD)
- Summe Budget Effort (MD)
- per Teammember (Task)
- per Demand

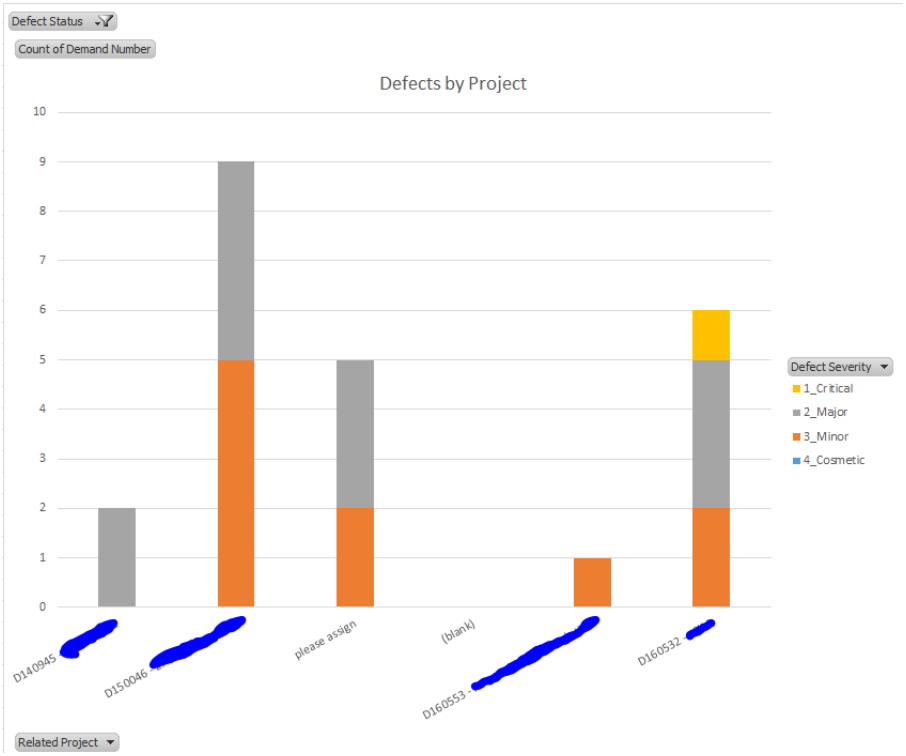

Abbildung 4.13: Defects by Project

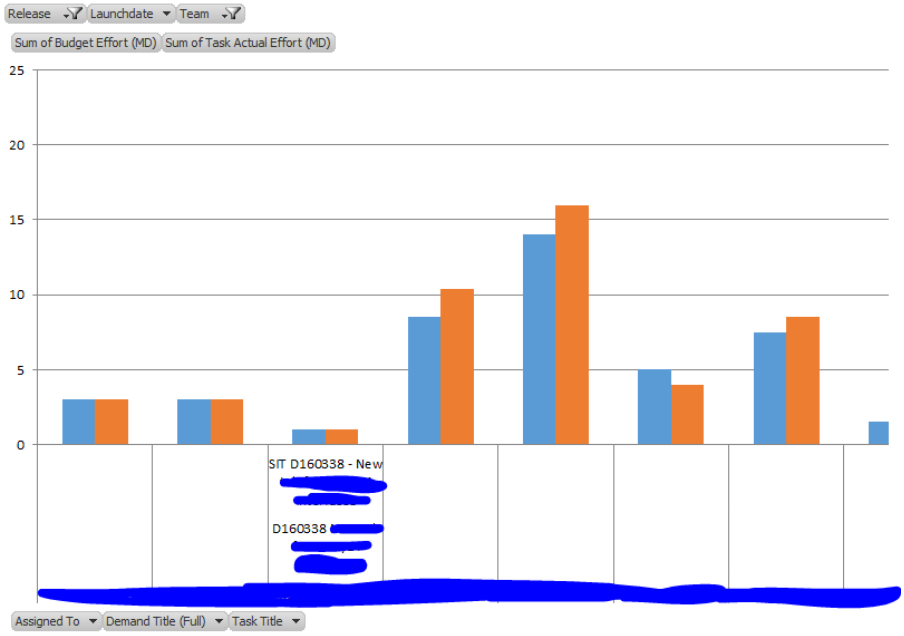

Abbildung 4.14: Efforts per Member

Beispiel:
siehe Abbildung 4.14

4.3.5 Testing Dashboard

Während der Testdurchführung wird sowohl pro Demand oder auch per Release der Testfortschritt gemessen und graphisch dargestellt.

- testing progress
- Summe Budget Effort (MD)
- per Demand
- per test execution status

Beispiele dazu:
siehe Abbildung 4.15, siehe Abbildung 4.16 oder
siehe Abbildung 4.17

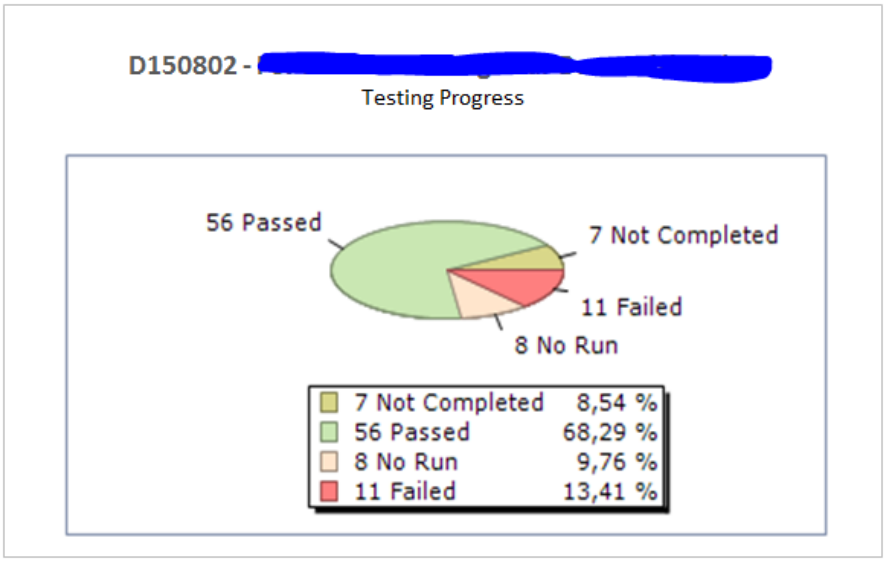

Abbildung 4.15: Testing Progress eines Demands

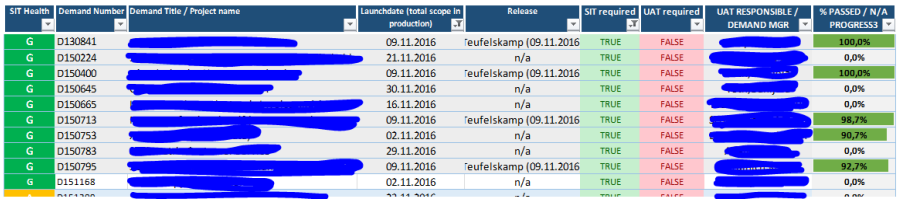

Abbildung 4.16: Release Monitoring General

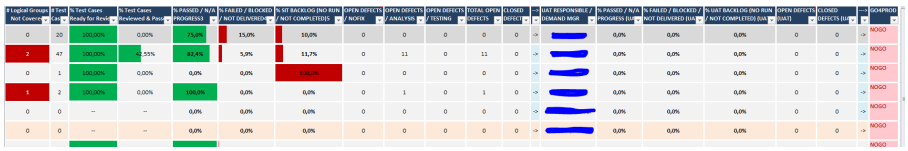

Abbildung 4.17: Release Monitoring Testing related

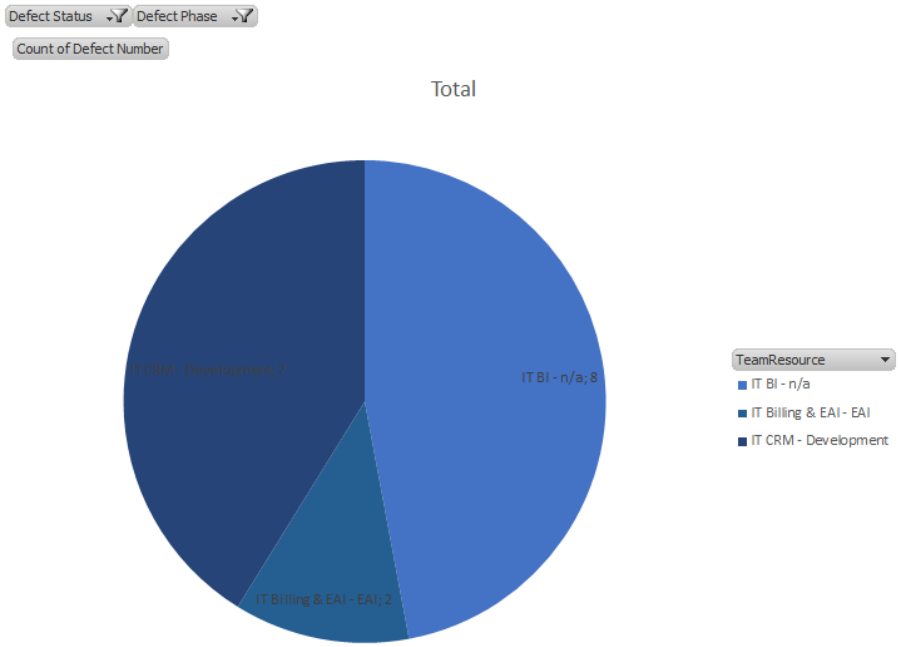

Abbildung 4.18: Defect per Team Ressource

4.3.6 Defect Dashboard

Eine weitere Messung erfolgt bei Defects bezüglich der Team Ressource. Welche Teams haben wie viele Defects in welchem Status und in welcher Phase in Bearbeitung?

- Anzahl Defects
- per Team Ressource
- per Defect status
- per Defect phase

Beispiel:
siehe Abbildung 4.18

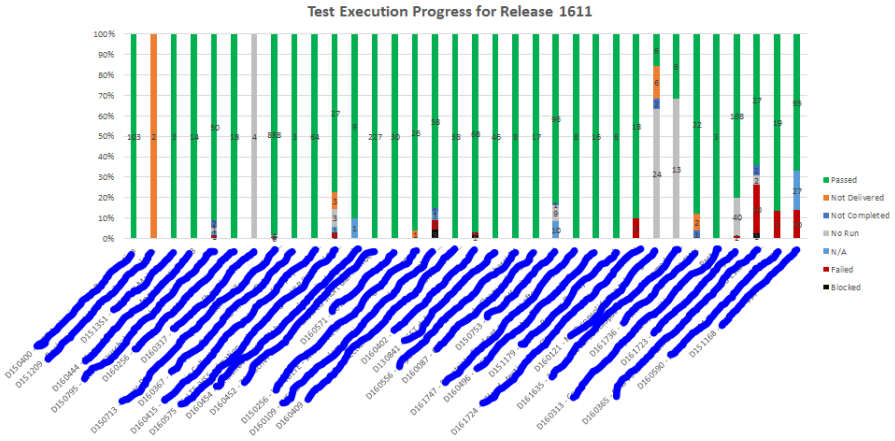

Abbildung 4.19: Testcase execution trend per Release

4.3.7 Testcase Report

Der Testcase Report liefert eine statistische Trendbeobachtung aller Testcases. Solche Beobachtungen finden statt:

- per Release
- per Demand

Beispiele dazu:
siehe Abbildung 4.21, siehe Abbildung 4.20

4.3.8 Testexecution Status

Der Testexecution Status liefert einen Überblick zu allen geplanten Demands einer Release:

- Anzahl der Testcases
- Ausführungsstatus der Testcases

Beispiel:
siehe Abbildung 4.21

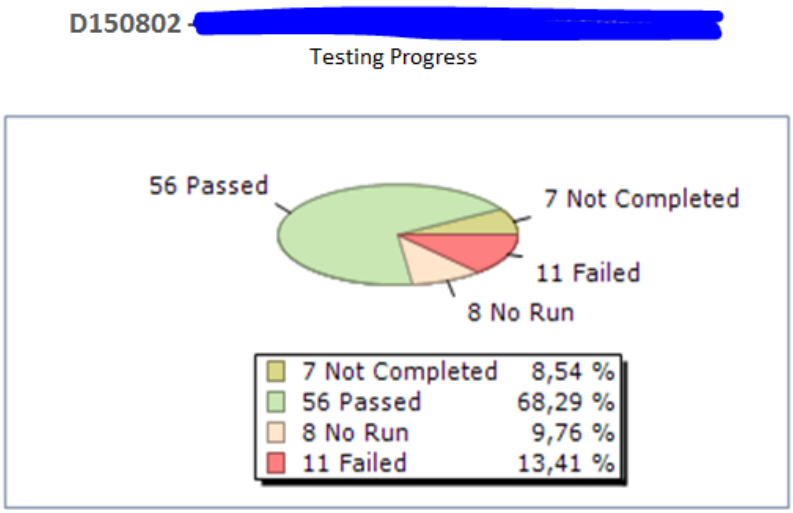

Abbildung 4.20: Testcase execution trend per Release

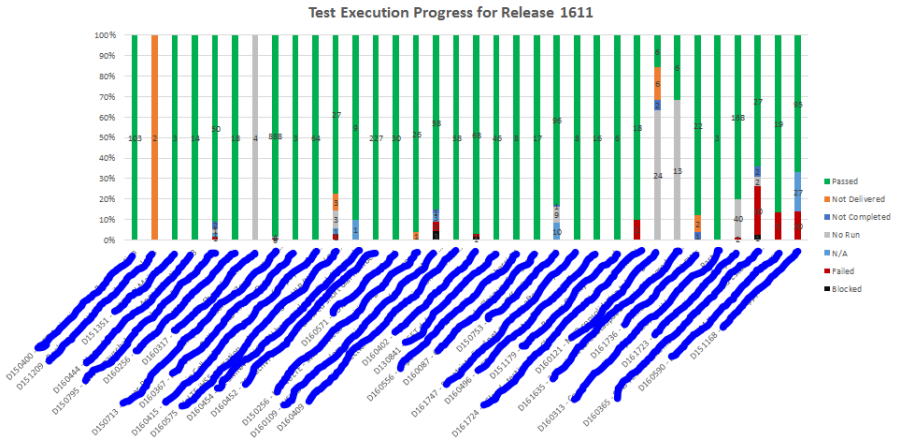

Abbildung 4.21: test execution progress per Release

4.4 Diskussion

In den beiden vorigen Abschnitten dieses Kapitels wurden sowohl von bereits verwendeten Metriken innerhalb des Unternehmens, als auch von den neu gefundenen Metriken im Zuge der Analyse, Beispiele angeführt und erläutert.
Grundsätzlich lassen sich bei beiden Ausführungen Gemeinsamkeiten, Überschneidungen und gegenseitig Ergänzendes erkennen.

Bei Metriken des Abschnittes 4.3 Beispiele eingesetzter Metriken liegt der Fokus darauf, möglichst zeitnah und jederzeit einen aktuellen Überblick zur laufenden Testdurchführung gewinnen zu können.
Dabei dient die Planung der Releases als Basis und die dynamischen Daten dieser Planung werden verarbeitet, um den Teststatus zur jeweils aktuell geplanten Release verfügbar zu haben.

Bei Metriken des Abschnittes 4.2 Beispiele ergänzender Metriken dienen abgeschlossene, statische Daten aus bereits entschiedenen, fertigen Umsetzungen als Basis.
Dabei besteht die Option, aus den Erfahrungen der Vergangenheit Rückschlüsse ziehen zu können und basierend darauf in der Lage zu sein, aktuelle Ergebnisse zu beurteilen und abzuschätzen.

Aufgrund der unterschiedlichen Einsatzvarianten lassen sich die beiden Arten (ergänzende beziehungsweise eingesetzte Metriken) nicht miteinander vergleichen. Beide finden entsprechend ihres Anwendungsfalles Verwendung.

Zu den ergänzenden Metriken erfolgt und gibt es in dieser Arbeit generell keine Bewertung und Einteilung in Kategorien, da sich solche anhand eines einzelnen Beispiels nicht treffen lassen.
Dadurch gibt es keine Klassifizierungen von Qualität und Risiko in Zahlen, Gruppen oder Werten.

Die Analyse und Anwendung bei den Beispielen schließt immer mit offenen Aussagen, um keine Interpretationen vorweg zu nehmen und die Metriken der jeweiligen Anwendung entsprechend einsetzen zu können.

4.4.1 Risiken von Testmetriken

Ein wesentlicher Aspekt bei der Verwendung von Metriken ist, mögliche Risiken der eingesetzten Metriken ebenfalls zu berücksichtigen und in die jeweilige Diskussion einfließen zu lassen.

In beiden betrachteten Beispielen, jenen Metriken, welche sich bereits im Einsatz befinden oder jenen, welche neu erarbeitet wurden, handelt es sich um Messungen und Darstellung von Leistungen.
Leistungen sind jene, welche von Teams und Personen erbracht werden und auf Daten basieren, die in Systemen von Personen erzeugt und gepflegt werden. Des Weiteren werden die Ergebnisse der Messungen oftmals in schön aufbereiteten Grafiken dargestellt und von Menschen subjektiv interpretiert.

Nicht immer werden Metriken als positiv aufgefasst oder führen zu positiven Betrachtungen.
Mögliche Ursachen dafür liegen oftmals darin, dass die Ergebnisse auf Teams oder handelnde Personen angewandt werden, anstatt diese auf die Prozesse oder das Produkt anzuwenden.

Metriken schaffen Transparenz und Transparenz kann, positiv wahrgenommen, schnelle und leicht erfassbare Einblicke schaffen, negativ wahrgenommen jedoch als Bloßstellung (des Ergebnisses eines Kollegen oder der handelnden Person) aufgefasst werden.
Ein sensibler Umgang mit der Darstellung und den Erläuterungen zu den Metriken, oder Ableitungen von Maßnahmen daraus, ist daher wesentlich, um die Akzeptanz von Metriken zu gewährleisten.

Ein weiteres Risiko besteht darin, Ergebnisse unzulässig zu verwenden oder zu deuten, beispielsweise Defectstatistiken bei Mitarbeitergesprächen.
Dadurch wird ein neutraler Umgang von Defects zur Dokumentation und als selbstverständlicher Teil des Prozesses untergraben. Der Test und das Erstellen von Defects würde als negativ aufgefasst werden und das Ziel von Entwicklung wäre es, Defects bestmöglich zu vermeiden.

Ähnliches kann im Test stattfinden, falls zum Beispiel für die Tester die Anzahl gefundener Fehler in einem Ranking geführt wird. Das verleitet

dazu, möglichst viele Defects zu reporten und aus einem gefundenen Fehler zu Folgefehlern ebenfalls Defects aufzumachen, um im Ranking höher zu kommen.

Der Test selbst und die Ergebnisse daraus sind ein Messinstrument im Softwareentwicklungsprozess und ein dazu vorhandener Testprozess bietet viele Chancen, Verbesserungen zu unterstützen.
Wesentlich dabei ist es daher, nicht nur Metriken und den Einsatz dieser zu definieren, sondern gleichzeitig auch organisatorische und kulturelle Rahmenbedingungen zu schaffen, um die Risiken zu berücksichtigen.

5 Zusammenfassung

Qualitätssicherungsmaßnahmen in Softwareentwicklungsprozessen unterliegen laufend der Notwendigkeit, flexibel auf neue Herausforderungen einzugehen und diese anzunehmen.

Solchen ist der Softwaretest, ein zentrales Instrument zur analytischen Qualitätssicherung, unterworfen.
Trotz bester terminlicher Planung und dem Versuch, alle Risiken vorab zu berücksichtigen, kommt es vor, dass die zur Verfügung stehende Zeit gegen Ende des Prozesses knapp wird. In diese Phase des Prozesses fällt der Softwaretest.

Dennoch ist es erforderlich, mithilfe der Softwaretests Ergebnisse zu liefern. Ergebnisse, welche als Grundlage dienen, den Prozess bezüglich des Risikos zu möglicherweise vorhandenen Problemen, oder der vorhandenen Qualität, zu bewerten. Neben dem Ermitteln von Testergebnissen durch Testdurchführung sind weitere Lösungsansätze von Interesse, beispielsweise die Verwendung von Metriken.
In dieser Arbeit wird dieser Lösungsansatz erläutert und Möglichkeiten dazu nachgegangen, sowie Metriken anhand von Beispielen eingesetzt und diskutiert. Als Grundlage werden dazu reale Daten und Fakten eines Unternehmens herangezogen, welches am Telekommunikationsmarkt tätig ist.

Testen, Metriken, Qualität und Risiko - Zusammenhänge und Theorie dazu stehen am Beginn der Arbeit. Anhand dieser wird zusammengefasst, in welchem Rahmen sich die Arbeit befindet:
Welche Arten von Metriken kommen zum Einsatz?
Welche Kategorien, beispielsweise der Norm ISO 25010, finden zur Qualitätsbeurteilung bei den Tests Berücksichtigung?

Wie werden Metriken in Zusammenhang mit Qualität bewertet?
Risikobasierter Testansatz und Risikomanagement.

In weiterer Folge ist die Situation des Tests im Unternehmen beschrieben. Der Test ist als eigenständiges Team in den Softwareentwicklungsprozess eingegliedert und die Testdurchführung folgt dem V-Modell in Form eines Systemintegrationstests. Für die Durchführung stehen verschiedene Testplattformen zur Verfügung und diese beziehen sich auf einen speziellen Bereich, aus dem gesamten Unternehmensspektrums, dem Business-Service-Support-System Bereich. Zur Dokumentation und Administration werden Hewlett Packard Application Lifecycle Management, Microsoft Sharepoint und Hewlett Packard OpenView verwendet.

Bevor es zum Einsatz der Testmetriken kommt, erfolgt eine Auswertung und Analyse der vorhandenen Daten.
Viele der im Unternehmen gesammelten und verfügbaren Daten sind für den Test nicht relevant und dienen dem allgemeinen Prozessmanagement, sowie der Unternehmensführung. Dazu werden aus dem gesamten Pool verfügbarer Daten die Kenngrößen Demands, Testcases, Defects und Changes identifiziert und Parameter dazu festgelegt. Anhand dieser sind, dem Lösungsansatz folgend, neue Metriken erstellt, beschrieben und erläutert worden.
Zu jeder Metrik finden sich Inhalt, Beschreibung und Interpretation. Der Inhalt definiert den Zusammenhang zwischen den Kenngrößen, dessen eingesetzte Parameter in der Beschreibung erläutert sind. Die Aussagekraft der jeweiligen Metrik ist in der Interpretation zusammengefasst.

Im letzten Teil der Arbeit werden die Metriken angewendet.
Dies erfolgt anhand konkreter Daten und Fakten eines gewählten Beispiels. Jede Metrik ist graphisch aufbereitet, um darzustellen, in welcher Form die Anwendung und Auswertung erfolgen kann. Weiters sind die Ergebnisse beschrieben und Ergänzungen angeführt, welche Möglichkeiten die Parameter bieten und diese betreffen.

Gestellte Fragen und Ziele dieser Arbeit, betreffend Qualitätsbeurteilung beziehungsweise Risikoabschätzung, werden hier zu jeder Metrik erläutert.

Die Ergebnisse lassen sich zusammenfassen:
Durch den Einsatz der neuen Metriken werden bezüglich Qualität der einzu-

setzenden Software ergänzende Informationen gewonnen. Diese ermöglichen es, Aussagen zu Qualität und Risiko des Changes als Gesamtes zu tätigen, in welchen nicht nur die Testergebnisse berücksichtigt sind. Weiters können diese in die Qualitätsbeurteilung, in Zusammenhang mit den Detailergebnissen der Tests und der Beurteilung der Tester, einfließen.
Optimierungen für den Test, Testfallauswahl, Aufwand bzw. Testdurchführung lassen sich aus den daraus gewonnenen Informationen gewinnen und für den risikobasierten Testansatz anwenden. Erfahrenheit von Testern kann diese Erkenntnisse ersetzen bzw. ergänzen.

Anschließend sind Metriken gelistet, wie sie im Unternehmen bereits verwendet werden. Diese sind beispielhaft den Berichten und Statistiken aus dem Pool der Daten entnommen und stehen in keinem unmittelbaren Zusammenhang mit dem zuvor gewählten Beispiel. Solche Metriken dienen dem Testmanagement zur Administration und Organisation des Testteams und der Testdurchführung.

Bei einer darauffolgenden Betrachtung der bereits im Unternehmen eingesetzten Metriken und den neu erstellten, ergänzenden Metriken finden sich sowohl Überschneidungen, als auch Ergänzungen. Auf Risiken und Gefahren von Metriken wird hier hingewiesen.

Zusammenfassend lässt sich sagen, dass sich der Einsatz von Testmetriken zur Risikoabschätzung und Qualitätsbeurteilung als eine sinnvolle Ergänzung erweisen kann. Wesentlich und wichtig dabei ist, mögliche Gefahren, welche beim Einsatz von Metriken vorhanden sind, unbedingt einfließen zu lassen und zu berücksichtigen.

Anhang

In Kapitel 3, Abschnitt Kenngrößen, wurde eine Auswahl an Parametern je Kenngröße getroffen, welche bei den Metriken Berücksichtigung finden.

Folgende Tabellen beinhalten alle gemessenen und verfügbaren Eigenschaften der Kenngrößen aus den jeweiligen Tools.

Alle Parametereigenschaften der Kenngröße Demands:

Tabelle .1: DemandsFullList

Analysis Demand 2	Planned Deployment
App Created By	Planned Start UAT
App Modified By	preliminary duration (months)
Assessment Effort	preliminary effort estimation - Business internal (MD)
Attachment Link (cp_attlink)	preliminary effort estimation - Technology (MD)
Budget Effort	preliminary external costs
CirResButton	preliminary total effort estimation (MD)
Classification	Project
CMB Notes	Project 2
CMB Representative	Project Assignment
Committed BRS ready	Project Close Down
Committed Delivery	Project CMB Representative
Committed DRS ready	Project Controller
Committed Start Testing / Deployment	Project Demand Manager / Project Manager
consequences „if not“	Project Demand Owner / Project Owner
Content Type	Project Number
Created	Project Proposal
Demand Business Reasons	Project Result achieved (Delivery Date)
Demand Category	Project Start
Demand Dashboard	Project Team Effort (All)
Demand Description	Ranking

Tabelle .1: DemandsFullList *(Fortsetzung)*

Demand Health	Records of Processing Activities (Verfahrensverzeichnis) approved
Demand Number	Related Project
Demand Owner / Project Owner	Reporting Cluster 1
Effected Environments	Reporting Cluster 2
Effort Estimation - Business internal (MD)	Roadmap reference
Effort Estimation - Technology (MD)	Score
External costs	Score Details
Folder Child Count	send notification
Follow up Date	set InitiatorOrg(Admin)
GCP Demand ID	set Team Effort Field
General Demand	SIT required
General Demand 2	Small Demand
Group Single Instance related	Small Demand 2
ID	Standard Request
Initiator	Standard Request 2
InitiatorOrgInfo	Strategic Direction
Involved business units	Supporting Business Analyst
Involved departments	Target Systems (H3G/Orange)
IT Delivery Notes	Team Effort (All)
Item Child Count	Team Effort of Business Analyst / TPR
Item Type	Team Effort of CMB Representative
Link to Attachments	Team Effort of Project Manager/ Demand Manager
Link to Group Demand	Team Effort of Supporting BA
Link to Workpackage	Team Effort of Test Lead
Link To WP Container	TechNotes
Modified	Test Lead
Modified By	Total effort Estimation MD
No ORS reason	UAT required
Objective & Constraints	Umlaufbeschluss Approval
Old Demand	Umlaufbeschluss Approval Starter

Tabelle .1: DemandsFullList *(Fortsetzung)*

ORS approved	WP Number
ORS required	xOLD_Assessment Effort
Path	xOLD_NotificationSent
PID 2	xOLD_setInitiatorOrg
PID Type	xTMP_Calculated_Hyperlink

Alle Parametereigenschaften der Kenngröße Workpackages:

Tabelle .2: WorkpackagesFullList

App Created By	temp
App Modified By	WorkPackage
Content Type	WorkPackage Management - Handle WP Changes
Created	WP Actual Effort (MD)
Created By	WP aggregated Budget effort Tasks
Demand Category	WP Budget Effort (MD)
Demand Name	WP Cost Center Detail
Demand Number	WP Delay Description
Demand Phase	WP Description
Demand Planned Delivery	WP End Date
Folder Child Count	WP End Date of the Last Task
ID	WP Health
InitiatorOrgInfo	WP Manager
is Part of WP Container (deprecated)	WP Months
Item Child Count	WP Name
Item Type	WP Number
Link to WP Container	WP Planned Drop to SIT
Modified	WP Reason for Delay
Modified By	WP Report Notes
NotificationSent	WP Start Date

Tabelle .2: WorkpackagesFullList *(Fortsetzung)*

Path	WP Status
Related Project	WP Strategic Goal
Start handle WP Changes_b	WP Team Effort
SW Config Version	WP Type Local/ Group

Alle Parametereigenschaften der Kenngröße Testcases:

Tabelle .3: TestcasesFullList

Application	Change Status
Comments	Creation Date
Criticality	Customer Type
Description	Designer
Estimated DevTime	Execution Status
Expected Results	Owner Group
Path	Post-Conditions
Pre-Conditions	Priority
Protocol Type	Scenario Group
Scenario Type	Status
Subject	Template
Test Data	Test ID
Test Name	Test Phase
Testing Mode	Type
Version Date	Version Number
Version Owner	Version Status

Alle Parametereigenschaften der Kenngröße Defects:

Tabelle .4: DefectsFullList

Actual Drop to Prod Date	Found in Version
Actual Drop to SIT Date	Found in Version Freetext
Actual Effort	ID
Affected Environment	Item Child Count
App Created By	Item Type
App Modified By	Link to Attachments
Associated WP Number	Modified
Associated WP Number Link	Modified By
Attachment Link	No Fix Reason
Content Type	OverallManager
Created	Path
Created By	planned Drop to Prod Date
Datafix	planned Drop to SIT Date
Defect Description	Priority
Defect Manager	Project Number
Defect Number	Project Number (deprecated)
Defect Phase	Related Project
Defect Severity	Responsible Engineer
Defect Status	Service Request Number
Demand FullName	Set Team Resource, WP Number, Project Number
Demand Number	Synopsis
Environment Resource	Target Release
Fixed in Version	Team Resource
Fixed in Version Freetext	Temp
Folder Child Count	Test Team Resource
Found in Test Phase	

Alle Parametereigenschaften der Kenngröße Changes:

Tabelle .5: ChangesFullList

Affected Systems	Flag
Business Analyst / Technical Project Responsible	Impact
Change ID	Manager Group
Change Location	planned/ committed
Change Reason	Risk Level
Class	Short Desciption
Comment	Status
Demand Manager	Summary
Demand Number/Defect Number	Urgency
Demandtitel/Defect Synopsis	

Literatur

Amland, Stale (8 - 12 November 1999). *Risk Based Testing and Metrics: Risk Analysis Fundamentals and Metrics for software testing including a Financial Application case study*. Barcelona, Spain. URL: https://pdfs.semanticscholar.org/5f59/191293150d9a06c88549f86c346048a0396e.pdf (siehe S. 20).

Arendt, Thorsten (2018). *Software-Evolution: Software-Qualität*. Hrsg. von Dr. Thorsten Arendt. Phillips Universität Marburg. URL: https://www.uni-marburg.de/fb12/arbeitsgruppen/swt/lehre/files/sevo1516/sevo_ws1516_08_qual.pdf (besucht am 13. 05. 2018) (siehe S. 7).

Gaulke, Markus (2001). *Risikomanagement bei Softwareentwicklung und -einführung*. Hrsg. von KES Zeitschrift für Kommunikation und EDV-Sicherheit. Frankfurt. URL: http://2014.kes.info/archiv/heft/abonnent/0104/40.htm (besucht am 04. 06. 2018) (siehe S. 19, 20).

German Testing Board (2017-05-22). *ISTQB Glossar - Systemintegrationstest - SIT*. Hrsg. von ISTQB - International Software Testing Qualifications Board. URL: http://glossar.german-testing-board.info/#s (besucht am 07. 06. 2018) (siehe S. 21).

Gnoyke, Harm (2016). *ISO, weshalb warum? Ist Software-Qualität Geschmackssache? - embarc*. Hrsg. von embarc Software Consulting GmbH. URL: https://www.embarc.de/software-qualitaet-iso-25010/ (besucht am 13. 05. 2018) (siehe S. 6, 7, 9).

Hoffmann, Dirk W. (2013). *Software-Qualität*. 2., aktualisierte u. korr. Aufl. 2013. eXamen.press. Berlin und Heidelberg: Springer. ISBN: 9783642356995. DOI: 10.1007/978-3-642-35700-8. URL: http://dx.doi.org/10.1007/978-3-642-35700-8 (siehe S. 10, 11, 15, 16).

Institut, Johner (2018-02-12). *ISO 25010 – ISO 9126 ~ Norm für Qualitätsmerkmale (Software)*. Hrsg. von Prof. Dr. Christian Johner. URL: https://www.johner-institut.de/blog/iec-62304-medizinische-software/iso-9126-und-iso-25010/#iso25010/ (besucht am 13. 05. 2018) (siehe S. 7).

Liggesmeyer, Peter (2009). *Software-Qualität: Testen, Analysieren und Verifizieren von Software*. 2. Aufl. Heidelberg: Spektrum Akademischer Verlag und SpringerLink (Online service). ISBN: 9783827422033 (siehe S. 3, 5).

Ott, Alexander (2001). *Qualitätsmetriken für Software: Pleiten, Pech und Pannen in der Informatik*. URL: http://www-ti.informatik.uni-tuebingen.de/~ruf/seminar0102/Software-Metriken.pdf (besucht am 17. 03. 2018) (siehe S. 3, 4).

Sneed, Harry M. (2018). „Testmetriken für die Kalkulation der Testkosten und die Bewertung der Testleistung: P3Sneed1TAV20.pdf". In: URL: http://pi.informatik.uni-siegen.de/stt/23_4/01_Fachgruppenberichte/TAV/ (besucht am 17. 03. 2018) (siehe S. 16).

Stockerer, Philip (2018). *Early Access – lassen wir den Kunden testen?!* ATB Expertentreff. URL: https://www.austriantestingboard.at/events/event/early-access-lassen-wir-den-kunden-testen/ (besucht am 12. 05. 2018) (siehe S. 1).

Taentzer, Gabriele (2015). *Softwaremetriken: Softwarequalität 2015, Einführung in die Softwaretechnik* (siehe S. 2).

Wikibooks (2018-05-19). *Mathematik: Analysis: Metrische Räume: Definition – Wikibooks, Sammlung freier Lehr-, Sach- und Fachbücher*. Hrsg. von Wikibooks. URL: https://de.wikibooks.org/wiki/Mathematik:_Analysis:_Metrische_R%C3%A4ume:_Definition (besucht am 03. 06. 2018) (siehe S. 3).

CE-Wissen (2018). *Definition „harmonisierte Norm" - CE-Wissen*. Hrsg. von CE-Wissen - Wissensplattform zur CE-Kennzeichnung im Maschinen-, Anlagen- und Steuerungsbau. URL: http://www.ce-wissen.de/?p=1556 (besucht am 13. 05. 2018) (siehe S. 7).

Workbedeutung.info (2018). *Metrik - Wortbedeutung.info*. Hrsg. von Wortbedeutung.info. URL: https://www.wortbedeutung.info/Metrik/ (besucht am 02. 06. 2018) (siehe S. 2).