

This page is intentionally left blank

This page is intentionally left blank

**Abschlussarbeit zur Erlangung des akademischen Grades**  
**Bachelor of Science**

Thema: Evaluation der Zuordnung von Werkzeug- und Software-  
qualitätsmaß Klassifizierungen zu Qualitätsmerkmalen  
der Norm ISO/IEC 25000

Vorgelegt von: Thorsten Ahlbrecht

geboren am: 15.08.1987  
in: Stadtoldendorf

eingereicht am: 04.08.2017

Erstprüfer: Professorin Dr.-Ing. Sabine Radomski

Zweitprüfer: Professor Dr.-Ing. Andreas Hartmann

# Vorwort

Die Ausarbeitung der Bachelorarbeit zur Erreichung des akademischen Grades Bachelor of Science erfolgte im Rahmen des berufsbegleitenden Studiengangs Wirtschaftsinformatik der Hochschule für Telekommunikation in Leipzig. Aufgrund meiner Tätigkeit als Entwicklungsleiter im Business Configuration (BC) Umfeld des Projekts Business Integration Platform (BIP) der T-Systems International (TSI) GmbH habe ich ein gesteigertes Interesse daran, wie der Begriff Software Qualität definiert ist, in welchen Normen dieser verwendet wird und durch welche Werkzeuge dieser in der Praxis gemessen werden kann. Für die Möglichkeit die Bachelorarbeit anzufertigen, die kontinuierliche Betreuung und das regelmäßige konstruktive Feedback bedanke ich mich außerordentlich bei Frau Prof. Dr.-Ing. Sabine Radomski. Meinen Arbeitskollegen und insbesondere Herrn Christian Kieme danke ich für die Unterstützung bei fachlichen Fragen und für die investierte Zeit zur Betreuung im Erstellungszeitraum der Thesis. Im besonderen Maße möchte ich meiner Verlobten Ivana Kreies für den alltäglichen Rückhalt, die Unterstützung und ihr Verständnis für zeitliche Restriktionen, die im Laufe der Studienzeit angefallen sind, meinen Dank aussprechen. Zusätzlich bedanke ich mich bei meiner Familie und Freunden für die Korrekturlesungen der Arbeit und für die geschaffenen Freiräume während der Studienzeit.

Thorsten Ahlbrecht,  
Wolfsburg, 24.07.2017

## Kurzfassung

In der Bachelorarbeit wurden Software Qualitätsmaße der Qualitätsmodelle aus der Normenreihe ISO/IEC 25000 und Werkzeuge zur Operationalisierung von Qualitätsmerkmalen untersucht und klassifiziert. Für ein generelles Verständnis der Thematik wurden die Begriffe Qualität, Software Qualität und Software Qualitätsmaß, Metrik mit Hilfe von Fachliteratur voneinander abgegrenzt, sowie der Aufbau von Qualitätsmodellen erläutert. Für die Thesis relevante Inhalte der Normenreihe ISO/IEC 25000 wurden identifiziert und Ansätze zur Bewertung und Visualisierung von Software Qualität vorgestellt. Dem theoretischen Teil folgt eine Klassifikation der in den Qualitätsmodellen als relevant identifizierten Software Qualitätsmaße und selektierten Werkzeuge auf Basis von Ansätzen der Literatur und Möglichkeiten, die aus der Norm extrahiert werden konnten. Anschließend wurden einzelne Werkzeuge der eruierten Werkzeugklassen vorgestellt. Die Werkzeuge wurden bezüglich ihrer Eignung zur Messung der Software Qualitätsmaße der ausgewählten Qualitätsmodelle bewertet und die Ergebnisse für eine Aussage über die Tauglichkeit der Werkzeugklassifikationen aggregiert. Als am Besten geeignete Werkzeugart wurden dynamische Testwerkzeuge identifiziert. Abschließend wurden die qualifiziertesten Werkzeuge ALM, SonarQube und Silk Test den Software Qualitätsmaß Klassifikationen gegenübergestellt und die Resultate mittels Kiviat-Diagrammen visuell aufbereitet. Den Kern dieser Arbeit bildet die Untersuchung von Hypothesen. Die definierte Gesamthypothese setzt sich aus mehreren Teilhypothesen zusammen. Jede Teilhypothese behandelt einen Teilbereich der Fragestellung, welcher zur Beantwortung der Gesamthypothese benötigt wird. Die Ergebnisse dieser Bachelorarbeit können als Grundlage zur Bestimmung eines minimalen Software Qualitätsniveaus eines Software Gütesiegels genutzt werden. Die Bachelorarbeit dient als Basis für weitere empirische Arbeiten.

## Abstract

This bachelor thesis examines software quality measures related to quality models defined in the standard ISO/IEC 25000 and classifies tools for operationalization of quality attributes. For a general understanding of the subject, the terms quality and software quality, software quality measure and metric were distinguished based on technical literature and the structure of quality models was exemplified. Relevant contents for the bachelor thesis of the standard ISO/IEC 25000 were identified and approaches for evaluation and visualization of software quality were introduced. Following the theoretical part software measures of the quality models and selected tools were classified based on literature approaches and extracted possibilities of the standard. Thereafter, tools of the elicited tool classifications were introduced. These tools were then rated based on their adequacy to determine software measures; the results were then aggregated to form a conclusion about the appropriateness of the tool classification. Dynamic test tools were identified as the most suitable category. Concluding the evaluation the tools which offer the best coverage of software measures ALM, SonarQube and Silk Test were opposed to the software measure classifications and the results were visualized in Kiviat diagrams. The essence of inspection is formed by an overriding hypothesis consisting of multiple part hypotheses that each examines a different subject. Parts of the results of the bachelor thesis could be used as a principle for a minimally acceptable quality level for a software quality seal. The bachelor thesis can be used as a foundation for further empirical compositions.

# Inhaltsverzeichnis

<b>Abkürzungsverzeichnis</b>	<b>VIII</b>
<b>Formelverzeichnis</b>	<b>IX</b>
<b>Abbildungsverzeichnis</b>	<b>X</b>
<b>Tabellenverzeichnis</b>	<b>XII</b>
<b>Quellcodeverzeichnis</b>	<b>XIII</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Problemstellung	1
1.2 Forschungsstand	2
1.3 Ziele und nicht-Ziele	3
1.4 Gang der Untersuchung	4
<b>2 Theoretische Grundlagen</b>	<b>5</b>
2.1 Qualität	5
2.2 Die ISO/IEC 25000 Normenreihe	6
2.2.1 Software Qualität	6
2.2.2 Qualitätsmodelle	7
2.2.3 Abgrenzung der Qualitätsmodelle der ISO/IEC 25000 Normenreihe	8
2.2.4 Das System- und Software Produkt Qualitätsmodell	9
2.2.5 Abgrenzung der Begriffe Software Qualitätsmaß und Metrik	11
2.2.6 Identifikation erklärungsbedürftiger Software Maße der ISO/IEC 25023	12
2.3 Kiviat-Diagramme	13
2.4 Werkzeuge	14
<b>3 Hypothesen</b>	<b>15</b>
3.1 Teilhypothese 1	15
3.2 Teilhypothese 2	15
3.3 Teilhypothese 3	15
<b>4 Auswertung der Hypothesen</b>	<b>16</b>
4.1 Auswertung der Teilhypothese 1	16
4.2 Auswertung der Teilhypothese 2	30
4.2.1 Dynamische Testwerkzeuge	30
4.2.1.1 Strukturorientierte Testwerkzeuge	30
4.2.1.2 Funktionsorientierte Testwerkzeuge	32
4.2.1.3 Regressionswerkzeuge	34
4.2.1.4 Leistungs- und Stresstestwerkzeuge	36
4.2.2 Statische Analysewerkzeuge	38
4.2.2.1 Messwerkzeuge	38
4.2.2.2 Stilanalysatoren	41

4.2.2.3	Slicing Werkzeuge	43
4.2.3	Formale Verifikationswerkzeuge	44
4.2.4	Modellierende und analysierende Werkzeuge	47
4.2.4.1	Markov Modellierungswerkzeuge	47
4.2.4.2	FMECA Werkzeuge	48
4.2.4.3	Fehlerbaumanalyse Werkzeuge	50
4.2.5	Auswertung Werkzeugklassen zur Operationalisierung von Merkmalen	51
4.3	Auswertung der Teilhypothese 3	54
4.4	Auswertung der Gesamthypothese	64
<b>5</b>	<b>Zusammenfassung der Ergebnisse</b>	<b>65</b>
<b>6</b>	<b>Weiterführender Forschungs- und Gestaltungsbedarf</b>	<b>70</b>
	<b>Literaturverzeichnis</b>	<b>71</b>
	<b>Selbständigkeitserklärung</b>	<b>76</b>
	<b>Anlagen</b>	<b>77</b>
A	Glossar	77
B	Qualitätsmerkmale, Maße und Metriken der ISO/IEC 25023	82
C	Übersicht der unterstützten Programmiersprachen	98
D	Zuordnung dynamischer und statischer Eigenschaften nach Balzert	99
E	Eignung Werkzeuge zur Bestimmung der Metrikparameter	102
F	Aggregation der Bewertungsergebnisse der Werkzeugklassen	109
G	Quantitative Analyse der System- und Software Produkt Maße	116
H	Kategorien messbarer Code Konventionen des Werkzeugs Checkstyle	123
I	Kategorien von Metriken des Werkzeugs SonarQube	124

## Abkürzungsverzeichnis

<b>Abb.</b>	Abbildung
<b>ALM</b>	Application Lifecycle Management
<b>API</b>	Application Programming Interface
<b>AUT</b>	Application Under Test
<b>BC</b>	Business Configuration
<b>BIP</b>	Business Integration Platform
<b>CDA</b>	Continuous Delivery Automation
<b>CPA</b>	Configurable Program Analysis
<b>CPU</b>	Central Processing Unit
<b>CISIAD</b>	Research Center on Intelligent Decision Support Systems
<b>DIN</b>	Deutsches Institut für Normung
<b>EN</b>	Europäische Norm
<b>FCM</b>	Factor-criteria-metric
<b>FMEA</b>	Fehlermöglichkeits- und Einflussanalysen
<b>FMECA</b>	Fehlermöglichkeits-, Einfluss- und Kritikalitätsanalyse
<b>FURPS</b>	Functionality, Usability, Reliability, Performance and Supportability
<b>GMBH</b>	Gesellschaft mit beschränkter Haftung
<b>HFTL</b>	Hochschule für Telekommunikation Leipzig
<b>HP</b>	Hewlett-Packard
<b>HR</b>	highly recommended
<b>ID</b>	identification
<b>IDE</b>	Integrated Development Environment
<b>IEC</b>	International Electrotechnical Commission
<b>ISO</b>	International Organization for Standardization
<b>JVM</b>	Java Virtual Machine
<b>LOC</b>	Lines of code
<b>MTBF</b>	Mean Time Between Failure
<b>N/A</b>	not available
<b>OWASP</b>	Open Web Application Security Project
<b>PGM</b>	Probabilistisches graphisches Modell
<b>R</b>	recommended
<b>RAM</b>	Random Access Memory
<b>RPZ</b>	Risiko-Prioritätszahl
<b>SCM</b>	Supply Chain Management
<b>SQUARE</b>	Systems and Software Quality Requirements and Evaluation
<b>Tab.</b>	Tabelle
<b>TSI</b>	T-Systems International
<b>UD</b>	user's discretion
<b>URL</b>	Uniform Resource Locator
<b>vgl.</b>	vergleiche
<b>VM</b>	Virtual Machine
<b>z.T.</b>	zum Teil



## Formelverzeichnis

(2.1) Der Qualitätsbegriff nach WINTERSTEIGER .....	5
(2.2) Die zyklomatische Komplexität nach MCCABE .....	12
(4.1) Code Coverage für Unit Tests durch Sonar Qube .....	31
(4.2) Risiko Prioritätskennzahl durch Xfmea.....	50

# Abbildungsverzeichnis

Abb. 1 – Gang der Untersuchung .....	4
Abb. 2 – Magisches Dreieck .....	5
Abb. 3 – Strukturierung der Standards der SQuaRE Serie .....	6
Abb. 4 – Das Teufelsquadrat von Harry M. Sneed .....	7
Abb. 5 – Struktur von Qualitätsmodellen.....	8
Abb. 6 – Anzahl Software Qualitätsmaße der Qualitätsmodelle im SQuaRE Framework.....	8
Abb. 7 – Abhängigkeiten der Qualitätsmodelle und Maße der SQuaRE Reihe.....	9
Abb. 8 – System- und Produkt Qualitätsmodell der SQuaRE Reihe .....	10
Abb. 9 – Beispielhafte Kiviat-Diagramme für Software Qualitätsmaße .....	13
Abb. 10 – Beispielhafte Darstellung von Messergebnissen .....	13
Abb. 11 – Klassifikation anhand statischer und dynamischer Eigenschaften .....	17
Abb. 12 – Klassifizierung anhand der Perspektive der Maße im SQuaRE Framework .....	18
Abb. 13 – Klassifizierung Maße anhand Validität der Messergebnisse .....	19
Abb. 14 – Klassifizierung Maße anhand korrelierender Qualitätsmerkmale .....	20
Abb. 15 – Klassifizierung Maße nach generischen und spezifischen Eigenschaften .....	20
Abb. 16 – Gegenüberstellung der Klassifikationen G/S und HR/R/UD .....	21
Abb. 17 – Eingangsgrößen von Metriken der ISO/IEC 25023 .....	23
Abb. 18 – Abhängigkeit der Funktionalität von externen Eingangsgrößen .....	24
Abb. 19 – Abhängigkeit der Performanz von externen Eingangsgrößen .....	25
Abb. 20 – Abhängigkeit der Kompatibilität von externen Eingangsgrößen .....	25
Abb. 21 – Abhängigkeit der Benutzbarkeit von externen Eingangsgrößen .....	26
Abb. 22 – Abhängigkeit der Zuverlässigkeit von externen Eingangsgrößen .....	27
Abb. 23 – Abhängigkeit der Sicherheit von externen Eingangsgrößen .....	27
Abb. 24 – Abhängigkeit der Wartbarkeit von externen Eingangsgrößen .....	28
Abb. 25 – Abhängigkeit der Portabilität von externen Eingangsgrößen.....	29
Abb. 26 – Codeüberdeckungsvisualisierung mit dem Werkzeug SonarQube .....	31
Abb. 27 – ALM GUI Struktur .....	33
Abb. 28 – Testabdeckung durch ALM .....	33
Abb. 29 – Silk Test Ergebnis eines visuellen Tests .....	35
Abb. 30 – Telemetrie Dashboard einer JVM mit dem Werkzeug JProfiler .....	36
Abb. 31 – Method Call Recording mit dem Werkzeug JProfiler .....	37
Abb. 32 – Dashboard des Security Fortify on Demand Werkzeugs .....	39
Abb. 33 – Messkategorien in Security Fortify on Demand.....	39
Abb. 34 – Dashboard des Werkzeugs CodeInspector .....	40
Abb. 35 – Auswertung der Einhaltung von Programmierkonventionen mit Checkstyle .....	42
Abb. 36 – Messung der Programmierkonventionen mit dem Checkstyle Eclipse Plugin .....	42
Abb. 37 – Call Graph des Werkzeugs CodeSurfer.....	44
Abb. 38 – Verifikation von Nutzer definierten Verhalten mit dem Jessie Plugin .....	46
Abb. 39 – Bayes'sches Netz mit dem Werkzeug Open Markov .....	47
Abb. 40 – Einflussdiagramm mit dem Werkzeug Open Markov .....	48
Abb. 41 – FMEA mit dem Werkzeug Xfmea .....	49
Abb. 42 – Fehlerbaumanalyse mit der Isograph Reliability Workbench .....	51
Abb. 43 – Evaluation von Werkzeugklassen zur Messung von Software Maßen .....	52
Abb. 44 – Abdeckung der Qualitätsmerkmale durch dasWerkzeug ALM .....	55
Abb. 45 – Abdeckung der generischen und spezifischen Maße durch ALM .....	56
Abb. 46 – Abdeckung der HR, R, UD Software Qualitätsmaße durch ALM.....	56
Abb. 47 – Abdeckung der internen, externen, hybriden Maße durch ALM .....	57
Abb. 48 – Abdeckung der statischen und dynamischen Maße durch ALM .....	57
Abb. 49 – Abdeckung der Qualitätsmerkmale durch dasWerkzeug Silk Test.....	58

Abb. 50 – Abdeckung der generischen und statischen Maße durch Silk Test .....	58
Abb. 51 – Abdeckung der HR, R, UD Software Qualitätsmaße durch Silk Test .....	59
Abb. 52 – Abdeckung der internen, externen, hybriden Maße durch Silk Test.....	59
Abb. 53 – Abdeckung der statischen und dynamischen Maße durch Silk Test .....	60
Abb. 54 – Abdeckung der Qualitätsmerkmale durch das Werkzeug SonarQube .....	60
Abb. 55 – Abdeckung der generischen und statischen Maße durch SonarQube .....	61
Abb. 56 – Abdeckung der HR, R, UD Software Qualitätsmaße durch SonarQube.....	62
Abb. 57 – Abdeckung der internen, externen, hybriden Maße durch SonarQube .....	62
Abb. 58 – Abdeckung der statischen und dynamischen Maße durch SonarQube .....	63
Abb. 59 – Zusammenfassung der Tauglichkeit der Werkzeugklassen .....	66
Abb. 60 – Übersicht qualifizierteste Werkzeuge zur Qualitätsmerkmal Operationalisierung .	67
Abb. 61 – Übersicht qualifizierteste Werkzeuge zur Abdeckung der T <sub>1</sub> -Klassifizierungen ...	67
Abb. 62 – Abdeckung Software Qualitätsmaße durch untersuchte Werkzeuge .....	68

## Tabellenverzeichnis

Tab. 1 – Qualitätsmerkmale des System– und Software Produkt Qualitätsmodells.....	10
Tab. 2 – Definition der zyklomatischen Komplexität auf Codebasis .....	12
Tab. 3 – Klassifizierungen von Werkzeugen nach LIGGESMEYER.....	14
Tab. 4 – Klassifikationsansätze in $T_1$ .....	16
Tab. 5 – Beschreibung externer Abhängigkeiten der Metriken in ISO/IEC 25023 .....	23
Tab. 6 – Unterstützte Programmiersprachen des Werkzeugs SonarQube .....	30
Tab. 7 – Funktionalitäten des Werkzeugs ALM .....	32
Tab. 8 – Unterstützte Programmiersprachen des Werkzeugs Silk Test .....	34
Tab. 9 – Funktionalitäten des Werkzeugs Silk Test.....	34
Tab. 10 – Unterstützte Programmiersprachen des Werkzeugs JProfiler.....	36
Tab. 11 – Funktionalitäten des Werkzeugs JProfiler .....	36
Tab. 12 – Unterstützte Programmiersprachen durch Security Fortify on Demand.....	38
Tab. 13 – Unterstützte Programmiersprachen des Werkzeugs CodeInspectors.....	39
Tab. 14 – Funktionalitäten des Werkzeugs CodeInspector.....	40
Tab. 15 – Unterstützte Programmiersprachen des Werkzeugs Checkstyle.....	41
Tab. 16 – Unterstützte Programmiersprachen des Werkzeugs CodeSurfer .....	43
Tab. 17 – Funktionalitäten des Werkzeugs CodeSurfer.....	43
Tab. 18 – Unterstützte Programmiersprachen durch das Jessie Plugin.....	44
Tab. 19 – Funktionalitäten des Werkzeugs Jessie Plugin .....	45
Tab. 20 – Semantik der Anweisungen des Jessie Plugins.....	45
Tab. 21 – Funktionalitäten des Werkzeugs Open Markov .....	47
Tab. 22 – Funktionalitäten des Werkzeugs Xfmea .....	49
Tab. 23 – Funktionalitäten des Werkzeugs Isograph .....	50
Tab. 24 – Ermittlung der qualifiziertesten Werkzeugklassen .....	52
Tab. 25 – Abdeckung der Software Qualitätsmaße durch dynamische Testwerkzeuge .....	54
Tab. 26 – Anzahl der Software Qualitätsmaß der $T_1$ -Kategorien .....	55
Tab. 27 – Zusammenfassung der Klassifizierungen von Software Qualitätsmaßen.....	65
Tab. 28 – Abdeckung der Qualitätsmerkmale durch Werkzeugklassen .....	69
Tab. 29 – Qualitätsmerkmale, Maße und Metriken der ISO/IEC 25023 .....	97
Tab. 30 – Übersicht der unterstützten Programmiersprachen der untersuchten Werkzeuge ...	98
Tab. 31 – Zuordnung dynamischer und statischer Eigenschaften nach BALZERT .....	101
Tab. 32 – Eignung Werkzeuge zur Bestimmung der Metrikparameter .....	108
Tab. 33 – Aggregation der Bewertungsergebnisse der Werkzeugklassen .....	115
Tab. 34 – Ergebnisse der quantitativen Analyse der System- und Software Produkt Maße..	122
Tab. 35 – Kategorien messbarer Code Konventionen des Werkzeugs Checkstyle .....	123
Tab. 36 – Kategorien von Metriken des Werkzeugs SonarQube.....	126

## **Quellcodeverzeichnis**

Codeausschnitt 1 – Keyword-driven testing mit dem Werkzeug Silk Test .....	35
Codeausschnitt 2 – Prüfung von Source Code mit dem Werkzeug Jessie .....	45

# 1 Einleitung

## 1.1 Problemstellung

*"When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind."*

– KELVIN WILLIAM THOMSON ([1], S. 73)

Software ist heutzutage in vielen Lebensbereichen ubiquitär und gehört zu den komplexesten Errungenschaften der Menschheitsgeschichte (vgl. [2], S. 37). Umso wichtiger ist es, dass Software Produkte fehlerfrei funktionieren und eine hohe Qualität gewährleisten. Aufgrund der Komplexität und der kontinuierlichen Modifikationen von Software ist es jedoch nicht möglich, diese vollständig zu testen und absolute Fehlerfreiheit zu garantieren (vgl. [2], S. 36). Erschwert wird die Operationalisierung von Software Qualität eines Software Produktes dadurch, dass dieses nicht nur einen statischen Zustand besitzt, sondern ebenfalls ein dynamisches Verhalten zur Laufzeit aufweist, welches differenzierte Testverfahren erfordert (vgl. [3], S. 5). Da Qualität in einem Spannungsfeld mit der Zeit und anfallenden Kosten steht (vgl. [4], S. 12), wird in der Praxis versucht einen möglichst großen Bereich der Software Funktionalitäten mit einer minimalen Anzahl von Testfällen abzudecken (vgl. [5], S. 85), damit die Software durch die Testaufwände und einhergehenden Kosten für die Hersteller nicht unwirtschaftlich wird. Die Software Tests weisen dabei ausschließlich die Anwesenheit von Fehlern nach, aber niemals deren Abwesenheit (vgl. [6], S. 35). Man kann daher von einer bewussten Entscheidung von Softwareherstellern sprechen ein Produkt am Markt zu platzieren, welches bisher nicht identifizierte Fehler oder Mängel beinhaltet. BALZERT verweist in diesem Kontext auf ein Defektniveau in ausgelieferter Software von 0,2 bis 0,05 pro 1000 Zeilen Code und verdeutlicht die Kritikalität anhand eines Vergleichs der Bedeutung eines äquivalenten Aufkommens an Fehlern für andere Branchen (vgl. [7], S. 190). Als zusätzliche Komplikation zu existierenden Fehlern innerhalb von verfügbarer Software besitzt diese die Eigenschaft, mit steigender Lebensdauer und Anzahl an Änderungen, zu degenerieren (vgl. [4], S. 373 ff.). Durch nachträgliche Modifikationen kommt es nicht etwa zu einer Verbesserung des Software Qualitätsniveaus, sondern zu einer kontinuierlichen Verschlechterung. Um ein adäquates Qualitätsniveau zu erreichen, dieses objektiv zu bewerten und möglichst konstant zu halten, kann das Messen und die dauerhafte Überwachung von Software Qualität ein probates Mittel sein. Es besteht jedoch das Problem, dass der Begriff der Software Qualität nicht mit dem allgemeinen Qualitätsbegriff vergleichbar und nicht mit den gleichen Methoden bestimmbar ist (vgl. [8], S. 711). Um Software Qualität zu operationalisieren werden Qualitätsmodelle aufgestellt, die einzelne Eigenschaften einer Software durch Qualitätsmerkmale beschreiben und diese soweit verfeinern, bis die Ausprägungen der einzelnen Merkmale mit Metriken berechnet und von Software Qualitätsmaßen ausgedrückt werden können (vgl. [5], S. 30 f.). Die Schwierigkeit besteht in der Auswahl der relevanten Metriken aus der Theorie für den spezifischen Anwendungsfall. Normen beinhalten Zusammenstellungen nutzbarer Software Qualitätsmaße und beschreiben die Abhängigkeiten zu korrelierenden Qualitätsmerkmalen. Die Normenreihe ISO/IEC 25000 stellt eine Definition des Begriffs Software Qualität bereit und beschreibt Qualitätsmodelle, um diesen zu typisieren. Es ist jedoch unklar, ob die Software Qualitätsmaße des Standards in der Praxis anwendbar sind, wie diese klassifiziert werden können und welche Werkzeuge sich zur Messung eignen.

## 1.2 Forschungsstand

Die Thematik der Quantifizierung von Software Qualität beschäftigt die Forschung im Bereich Software Engineering seit Jahrzehnten. Einer der ersten Ansätze aus dem Jahr 1977 zur Aufstellung eines Software Qualitätsmodells geht auf MCCALL zurück (vgl. [9], S. 262). Die Hewlett-Packard Company (HP) entwickelte 1985 ein eigenes Qualitätsmodell mit der Bezeichnung Functionality, Usability, Reliability, Performance and Supportability (FURPS), um die Qualität von Software Produkten zu messen und kontinuierlich zu verbessern (vgl. [9], S. 260). Im Jahr 1991 wurde ein erster proprietärer internationaler Standard zur Bewertung von Software Qualität in Form der ISO/IEC 9126 herausgegeben. Der Unterschied zu den vorherigen Qualitätsmodellen von MCCALL oder HP besteht darin, dass die ISO/IEC 9126 keine Überschneidungen der enthaltenen Qualitätsmerkmale besitzt und auf alle Ausprägungen von Software Produkten anwendbar ist (vgl. [10], S. 60). 2011 wurden die Normen ISO/IEC 9126 und ISO/IEC 14598 in die internationale Normenreihe ISO/IEC 25000 überführt (vgl. [11], S. 19 f.). Die aktuellste veröffentlichte Version der Normenreihe ISO/IEC 25000 zur Bereitstellung von Software Qualitätsmodellen zur quantitativen Messung von Software Qualität wurde im Jahr 2014 herausgegeben (vgl. [11], S. 1). Die Anwendung von Software Qualitätsmaßen und die Beurteilung ihrer Güte wurden in verschiedenen empirischen Arbeiten untersucht. In der Thesis von JAHNEL aus dem Jahr 2016 kommt dieser zu dem Ergebnis, dass sich die Software Qualitätsmaße der ISO/IEC 25000 nicht pauschal auf ein System- oder Software Produkt übertragen lassen (vgl. [12], S. 68). Die Anwendbarkeit der Qualitätsmodelle der Normensammlung ISO/IEC 25000 im Rahmen der agilen Entwicklungsmethode Scrum ist nach SIMBÜRGER möglich, solange die Prinzipien der agilen Entwicklung beachtet und Metriken mit Bedacht angewendet werden (vgl. [13], 1 ff.). In einer weiteren empirischen Untersuchung werden Software Qualitätsmaße auf Basis von Ansätzen aus der Literatur in die Kategorien Produkt-, Prozess- und hybride Maße eingeteilt und die so gebildeten Klassifikationen den Qualitätsmerkmalen der ISO/IEC 25000 Normenreihe tabellarisch zugeordnet (vgl. [14], S. 49). Es wird in dieser Thesis nicht berücksichtigt, dass die ISO/IEC 25000 Normenreihe selbst Metriken zur Operationalisierung von Qualitätsmerkmalen vorgibt. In dieser Bachelorarbeit sind Klassifikationen auszuarbeiten, der Bezug von Software Qualitätsmaßen der ISO/IEC 25000 zu Werkzeugen herzustellen und anschließend zu evaluieren. Im Rahmen der Untersuchung wird auf weitere empirische Arbeiten referenziert, falls diese im spezifischen Kontext relevant sind.

### **1.3 Ziele und nicht-Ziele**

Den Rahmen und das Ziel der Bachelorarbeit bildet die Untersuchung und Beantwortung einer Gesamthypothese und drei abgeleiteten Teilhypothesen, die einzelne Bereiche der Fragestellung thematisieren. Im Rahmen der Hypothesen werden die Software Qualitätsmaße der Qualitätsmodelle der Norm ISO/IEC 25010 und Werkzeuge zur Messung analysiert und klassifiziert. Es ist zu untersuchen, ob die definierten Software Qualitätsmaße explizit durch das Vorliegen von Quelltext bestimmt werden können oder diese abhängig von externen Einflüssen sind. Eine Aussage über die tauglichsten Werkzeugtypen zur Bestimmung der Software Qualitätsmaßausprägungen der relevanten Qualitätsmodelle ist anhand eines Bewertungsverfahrens abzuleiten. Die qualifiziertesten Werkzeuge und Werkzeugklassen sind den ermittelten Software Qualitätsmaß Klassifikationen gegenüberzustellen und die Ergebnisse zu bewerten, grafisch aufzubereiten und abschließend kritisch zu reflektieren. Kein Ziel der Bachelorarbeit ist es Qualitätsmodelle die subjektiven Einflüssen unterliegen zu analysieren, da diese ohne das Einbeziehen und die Befragungen von Personengruppen nicht ausgewertet werden können. Ebenfalls sind datenabhängige Qualitätsmodelle kein Teil der Untersuchung, da die Datenqualität durch die jeweiligen Anwendungsfälle variiert. Der Fokus der Ausarbeitung liegt auf Qualitätsmodellen, die einer Software oder dem Entwicklungsprozess direkt zugeordnet und objektiv bewertbar sind. Auf eine ausführliche Beschreibung jedes einzelnen Software Qualitätsmaßes eines Qualitätsmodells und der damit korrelierenden Berechnungsvorschrift wird im Rahmen der Thesis verzichtet, da diese in der zugehörigen Norm und in aufbereiteter Form im Anhang B dieser Bachelorarbeit nachgeschlagen werden können (vgl. Tab. 29). Hierzu wird auf die in der Normenreihe ISO/IEC 25000 definierte Identifikationsnummern (ID) der Software Qualitätsmaße verwiesen. Relevante Grundlagen zum Verständnis der in der Bachelorarbeit bearbeiteten Themen werden in zusammengefasster Form vorgestellt und im Fall der Notwendigkeit einer ausführlichen Beschreibung auf die Literatur oder das Glossar im Anhang A verwiesen. Das Ergebnis der Ausarbeitung kann für weitere Abschlussarbeiten im Bereich Software Engineering genutzt werden.



## 1.4 Gang der Untersuchung

Die Bachelorthesis mit dem Titel „*Evaluation der Zuordnung von Werkzeug- und Software qualitätsmaß Klassifizierungen zu Qualitätsmerkmalen der Norm ISO/IEC 25000*“ setzt sich aus sechs Kapiteln zusammen, die in Abb. 1 schematisch dargestellt sind. Im ersten Kapitel erfolgt die Einführung in das Thema der Arbeit. Es werden die Problemstellung und der aktuelle Forschungsstand beschrieben, Ziele und nicht- Ziele an die Arbeit präzisiert und der Gang der Untersuchung vorgestellt. Das zweite Kapitel beinhaltet die theoretischen Grundlagen, die zum Verständnis der Untersuchung notwendig sind. Neben den Bereichen Qualität und der Darstellung relevanter Qualitätsmodelle der Normenreihe ISO/IEC 25000, wird auf Werkzeuge, Grundlagen zur Messung von Software Qualität und deren Visualisierungsform durch Kiviat-Diagramme eingegangen. Im dritten Kapitel wird die Struktur des praktischen Teils der Thesis definiert, indem eine Hypothese aufgestellt und Teilhypothesen als Grundlage zur Beantwortung der Gesamthypothese abgeleitet werden. Im vierten Kapitel der Thesis werden die Hypothesen sequenziell untersucht, beantwortet und die erzeugten Ergebnisse kritisch reflektiert. Abschließend erfolgt die Beantwortung der übergeordneten Hypothese durch die Resultate der einzelnen Teilhypothesen. Die Ergebnisse der Forschungsarbeit werden im fünften Kapitel aufbereitet und zusammengefasst. Abschließend erfolgt im Kapitel sechs ein Ausblick bezüglich des weiterführenden Forschungs- und Gestaltungsbedarfs in Form von offenen Forschungsfragen.

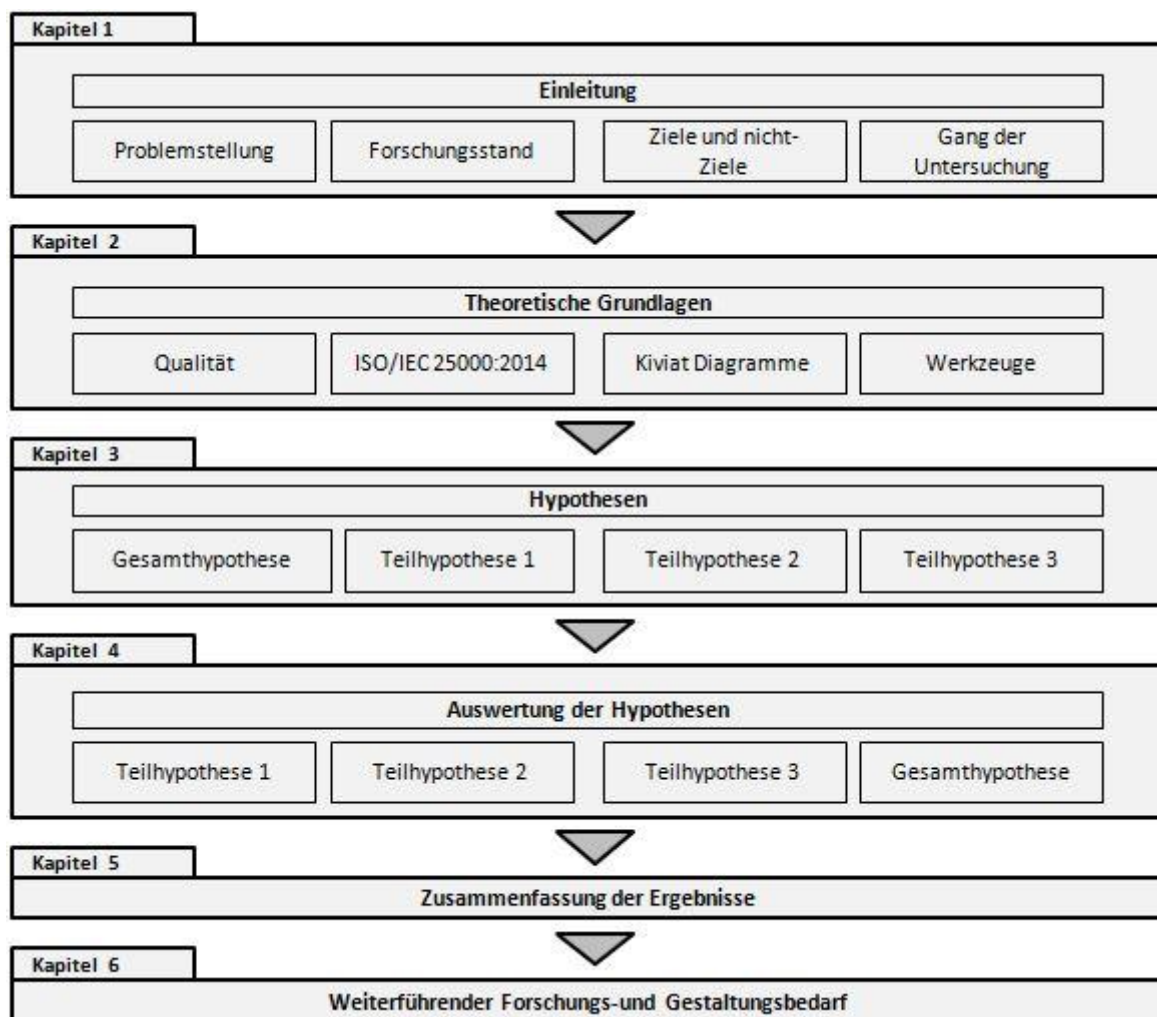


Abb. 1 – Gang der Untersuchung<sup>1</sup>

<sup>1</sup> Eigene Darstellung

## 2 Theoretische Grundlagen

### 2.1 Qualität

Ansätze der Definition des Qualitätsbegriffs unterliegen nach GARVIN der Perspektive in dem dieser relevant ist. Er unterscheidet die Perspektiven des transzendenten, produkt-, prozess-, benutzer-, kosten- und nutzerbezogenen Ansatzes zur Definition von Qualität (vgl. [15], S. 25, f.). BALZERT unterstellt dem Begriff der Qualität ähnliche Abhängigkeiten (vgl. [7], S. 460) und definiert diesen als gemessene Eigenschaften von Objekten anhand vorgegebener, vereinbarter oder erwarteter Merkmale, die eine prozess- oder ergebnisorientierte Charakteristik aufweisen (vgl. [7], S. 461). Die proprietäre internationale Norm DIN EN ISO 9000:2015 liefert einen konzeptionellen Rahmen, Grundsätze und Begriffsdefinitionen zur Erstellung von Qualitätsmanagementsystemen (vgl. [16], S. 7). In diesem Kontext wird der Begriff Qualität wie folgt definiert:

*„Grad, in dem ein Satz inhärenter Merkmale eines Objekts Anforderungen erfüllt.“*  
- DIN EN ISO 9000 ([16], S. 39)

In der nationalen Norm DIN 55350-11, die die Norm DIN EN ISO 9000 ergänzt, findet sich eine ähnliche Definition des Qualitätsbegriffs. Hier wird Qualität als Relation zwischen einer Anforderung und den umgesetzten Eigenschaften eines Objektes beschrieben.

*„Qualität ist die an der geforderten Beschaffenheit gemessene realisierte Beschaffenheit.“*  
- DIN 55350-11 ([17], S. 10)

TIEMEYER bezeichnet Qualität als „Fit for Purpose“ oder übersetzt als „Geeignet für den jeweiligen Zweck“ und verweist auf eine pseudomathematische Formulierung des Qualitätsbegriffs von WINTERSTEIGER (vgl. [18], S. 447), die in Formel (2.1) dargestellt ist.

$$\text{Qualität: } 1 = \frac{\text{Leistung}}{\text{Erfordernisse}} \quad (2.1)$$

Qualität befindet sich in einem kontinuierlichen Spannungsfeld mit anfallenden Kosten und verfügbarer Zeit. Erfolgt eine Änderung an einem der Parameter besteht, wie in Abb. 2 dargestellt, eine Wechselwirkung mit den korrelierenden Parametern (vgl. [4], 11 f.).

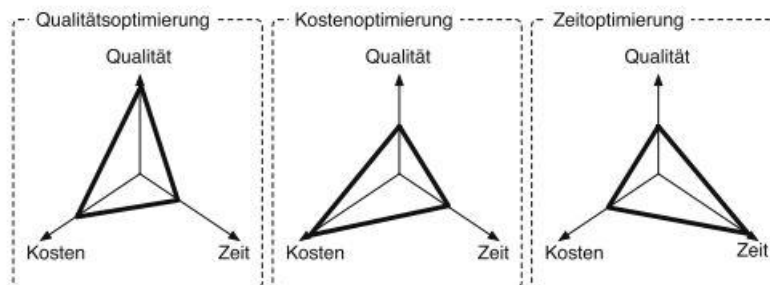


Abb. 2 – Magisches Dreieck<sup>2</sup>

<sup>2</sup> Abb. aus ([4], S. 12)

## 2.2 Die ISO/IEC 25000 Normenreihe

Der internationale Standard ISO/IEC 25000 mit dem Titel „*Systems and software engineering – System and software Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE*“ stellt Referenzmodelle, Definitionen, Bedingungen und einen Leitfaden zur Evaluation von Spezifikationen und zur Messung von Software Qualität bereit (vgl. [11], vi). Die Normen ISO/IEC 9126 und ISO/IEC 14598 wurden partiell in die ISO/IEC 25000 Normenreihe integriert und von der SQuaRE Serie ersetzt (vgl. [11], S. 20). Die Inhalte sind in 14 Dokumenten in fünf Themenbereiche und eine Erweiterung gegliedert (vgl. [11], S. 8). Der systematische Aufbau des SQuaRE Frameworks ist in Abb. 3 dargestellt.

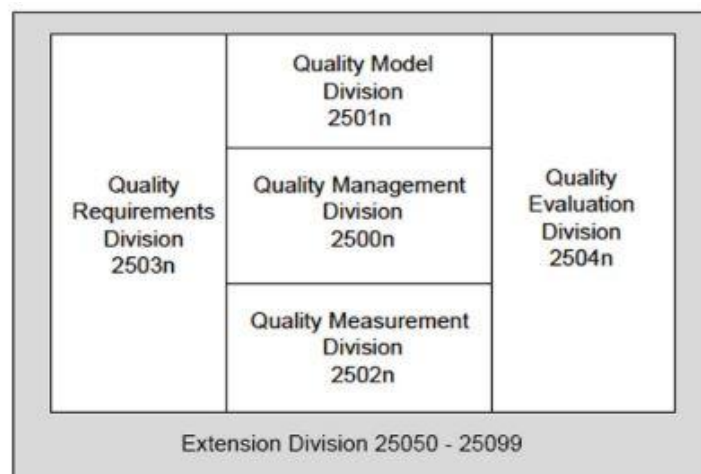


Abb. 3 – Strukturierung der Standards der SQuaRE Serie<sup>3</sup>

Der Normenbereich ISO/IEC 2500n definiert grundlegende Begriffe und referenziert allgemeine Modelle und Standards, die zum Verständnis der Norminhalte notwendig sind. Die Normen 2501n stellen Qualitätsmodelle auf und werden durch die Normen aus dem Bereich 2502n zur Quantifizierung von Software Qualität ergänzt. Der Bereich 2503n thematisiert und unterstützt die Erstellung von Spezifikationen. 2504n enthält Anforderungen, Empfehlungen und Richtlinien zur Bewertung von Software Produkten. 2505n-2509n beinhaltet eine Erweiterung der Normenreihe durch zusätzliche internationale Standards und technische Berichte (vgl. [11], S. 8).

### 2.2.1 Software Qualität

*You can't control what you can't measure.*

- DEMARCO, TOM ([19], S. 3)

Aus dem Zitat von DEMARCO geht hervor, dass die Messbarkeit eines Objekts die Grundlage für dessen Bewertung und Kontrolle bildet. Dieses betrifft ebenfalls den Begriff der Software Qualität, der mit dem allgemeinen Qualitätsbegriff nicht vergleichbar und nicht mit den selben Verfahren operationalisierbar ist (vgl. [8], S. 711). Um eine Messung des Software Qualitätsniveaus zu ermöglichen, wird der Software Qualitätsbegriff durch Qualitätsmodelle beschrieben (vgl. [7], S. 460 f.). Bei jedem Software Produkt variieren die Qualitätsanforderungen (vgl. [11], S. 25). Deshalb ist es notwendig Qualitätsziele und Qualitätsstufen festzulegen, aus denen Qualitätsanforderungen in Form von Qualitätsmerkmalen abgeleitet werden (vgl. [7], S. 474). Die Qualitätsanforderungen sind in funktionale und nichtfunktionale Anforderungen differenziert. Funktionale Anforderungen spezifizieren das Verhalten und die

<sup>3</sup> Abb. aus ([11], S. 7)

Funktionalität von Software Produkten unter spezifischen Bedingungen. Nichtfunktionale Anforderungen repräsentieren technische Eigenschaften der Software (vgl. [20], S. 108). Um eine Aussage über die Qualität und die Korrektheit einer Software zu generieren, wird der Grad der Befriedigung von Qualitätsmerkmalen gemessen. Zusätzlich sind in diesem Kontext die Erfüllung der geforderten Funktionalitäten und allen damit verbundenen Kriterien relevant (vgl. [2], S. 65). Die Ergebnisse der Messung von Software Qualität liefert eine Basis zur Einschätzung, inwiefern ein Software Produkt den definierten Qualitätszielen entspricht. Des Weiteren steigern Messungen von Software Qualität das Verständnis und die Vergleichbarkeit mit anderen Produkten, ermöglichen Vorhersagen bezüglich des Verhaltens einer Software (vgl. [3], S. 7) und stellen die Grundlage zur Steuerung für die Minimierung von Risiken durch proaktiv eingeleitete Maßnahmen bereit (vgl. [5], S. 52). In der ISO/IEC 25000 Reihe wird Software Qualität wie folgt beschrieben:

*“Capability of software product to satisfy stated and implied needs when used under specified conditions.”*

- ISO/IEC 25000 ([11], S. 6)

Die Definition des Software Qualitätsbegriffs der Normenreihe ISO/IEC 25000 unterscheidet sich vom Qualitätsbegriff der ISO 9000 durch die explizite Befriedigung von festgelegten und erwarteten Eigenschaften eines Software Produkts, anstelle der exklusiven Erfüllung von Anforderungen (vgl. [11], S. 6). Wie der allgemeine Qualitätsbegriff (vgl. Kapitel 2.1) befindet sich Software Qualität in einem Spannungsfeld mit den Einflussgrößen Quantität, Kosten und Entwicklungsdauer, die bei einer Änderung eines Parameters Auswirkung auf alle anderen besitzt. Den Unterschied zum Trilemma des Qualitätsbegriffs bildet die zusätzliche Dimension der Quantität. Mit steigender Quantität steigt die Komplexität einer Software, die HOFFMANN als zentrale Ursache für mangelhafte Qualität nennt (vgl. [4], S. 13). Das Teufelsquadrat von SNEED (vgl. Abb. 4) ist ein Modell zur Visualisierung der Wechselwirkung zwischen den Einflussgrößen Qualität, Quantität, Entwicklungsdauer und Kosten auf die verfügbare und begrenzte Produktivität von Entwicklungsressourcen (vgl. [7], S. 196 f.).

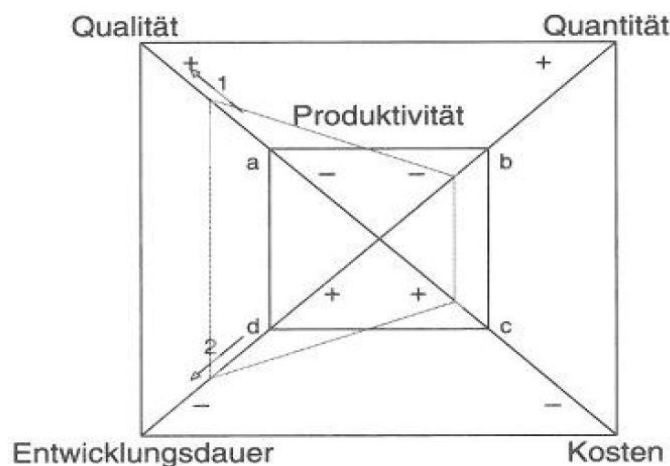


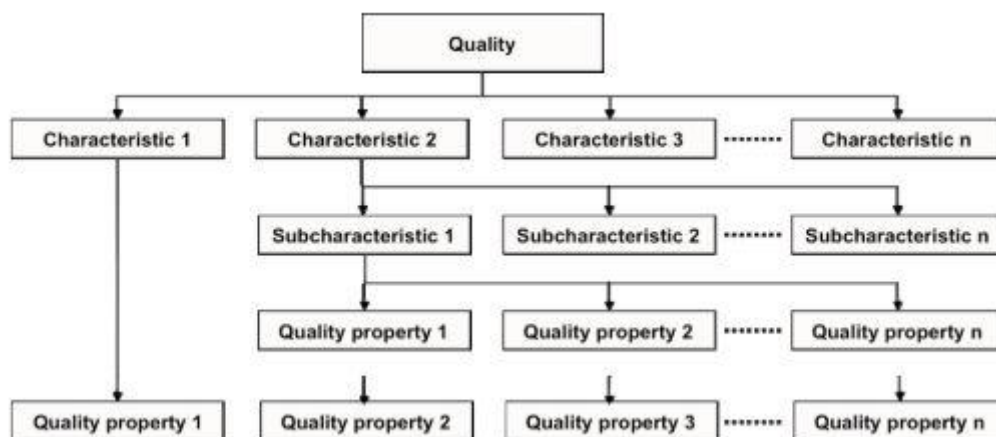
Abb. 4 – Das Teufelsquadrat von Harry M. Sneed<sup>4</sup>

### 2.2.2 Qualitätsmodelle

Ein Qualitätsmodell stellt die Verbindung zwischen allgemeinen Qualitätsbegriffen, Qualitätszielen und Metriken her, um Aussagen über Qualitätseigenschaften von Produkten, Maßnahmen und Prozessen zu ermöglichen (vgl. [5], S. 53 f.; [5], S. 35). Der Begriff Software

<sup>4</sup> Abb. aus ([18], S. 454)

Qualität wird in Qualitätsmodellen durch einzelne Qualitätsmerkmale präzisiert, die sich wiederum aus mehreren Teilmerkmalen zusammensetzen (vgl. [9], S. 258). Teilmerkmale repräsentieren Eigenschaften eines Produkts und werden soweit verfeinert, bis diese durch Qualitätsindikatoren beziehungsweise Metriken abgebildet werden können (vgl. [21], S. 2). Software Qualitätsmaße ermöglichen die Quantifizierung der Qualitätsindikatoren und bilden die Resultate auf einer Skala ab (vgl. [7], S. 462). Mehrere Qualitätsmerkmale können gemeinsame Teilmerkmale aufweisen und in einer hierarchischen Baumstruktur dargestellt werden. Dieses Schema wird als factor-criteria-metric-Modell (FCM) bezeichnet (vgl. [9], S. 258). Der systematische Aufbau eines FCM-Modells ist in Abb. 5 dargestellt.

Abb. 5 – Struktur von Qualitätsmodellen<sup>5</sup>

### 2.2.3 Abgrenzung der Qualitätsmodelle der ISO/IEC 25000 Normenreihe

Das SQuaRE Framework definiert in Norm ISO/IEC 25010 drei Qualitätsmodelle, die zusammengesetzt alle Merkmale von Software Qualität für spezifische Stakeholdergruppen abbilden (vgl. [21], S. 2). Das Qualitätsmodell der Datenqualität besteht aus 15 (vgl. [22], S. 4), das System- und Software Produkt Qualitätsmodell aus acht und das Nutzungsqualitätsmodell aus fünf verschiedenen Qualitätsmerkmalen (vgl. [21], S. 3), die sich wiederum aus mehreren Teilmerkmalen zusammensetzen. In der Normenreihe sind Metriken vorgegeben die definieren, wie einzelne Software Qualitätsmaße korrelierender Qualitätsmerkmale berechnet werden. Die Software Qualitätsmaße des Qualitätsmodells der Datenqualität sind in ISO/IEC 25024, die der Nutzungsqualität in ISO/IEC 25022 und die des System- und Software Produktqualität in ISO/IEC 25023 definiert. Die Summe der Software Qualitätsmaße innerhalb der zugehörigen Norm ist in Abb. 6 aufgeführt.

	Nutzungsqualitätsmaße	System- und Software Produkt Qualitätsmaße	Datenqualitätsmaße
ISO25023		86	
ISO25022	36		
ISO25024			63
Σ Qualitätsmaße		185	

Abb. 6 – Anzahl Software Qualitätsmaße der Qualitätsmodelle im SQuaRE Framework<sup>6</sup>

Die drei Qualitätsmodelle des SQuaRE Frameworks besitzen Abhängigkeiten zueinander (vgl. [21], S. 2). Eine schematische Darstellung der Wechselwirkungen der vorgegebenen Software Qualitätsbereiche der ISO/IEC 25000 Normenreihe ist in Abb. 7 visualisiert. Die Grundlage von Software Qualität bildet die Prozessqualität. Die Kombination von System-,

<sup>5</sup> Abb. aus ([21], S. 2)

<sup>6</sup> Eigene Darstellung

Software Produkt- und Datenqualität beeinflusst die Servicequalität, die wiederum mit der von Stakeholdern empfundenen Nutzungsqualität korreliert.

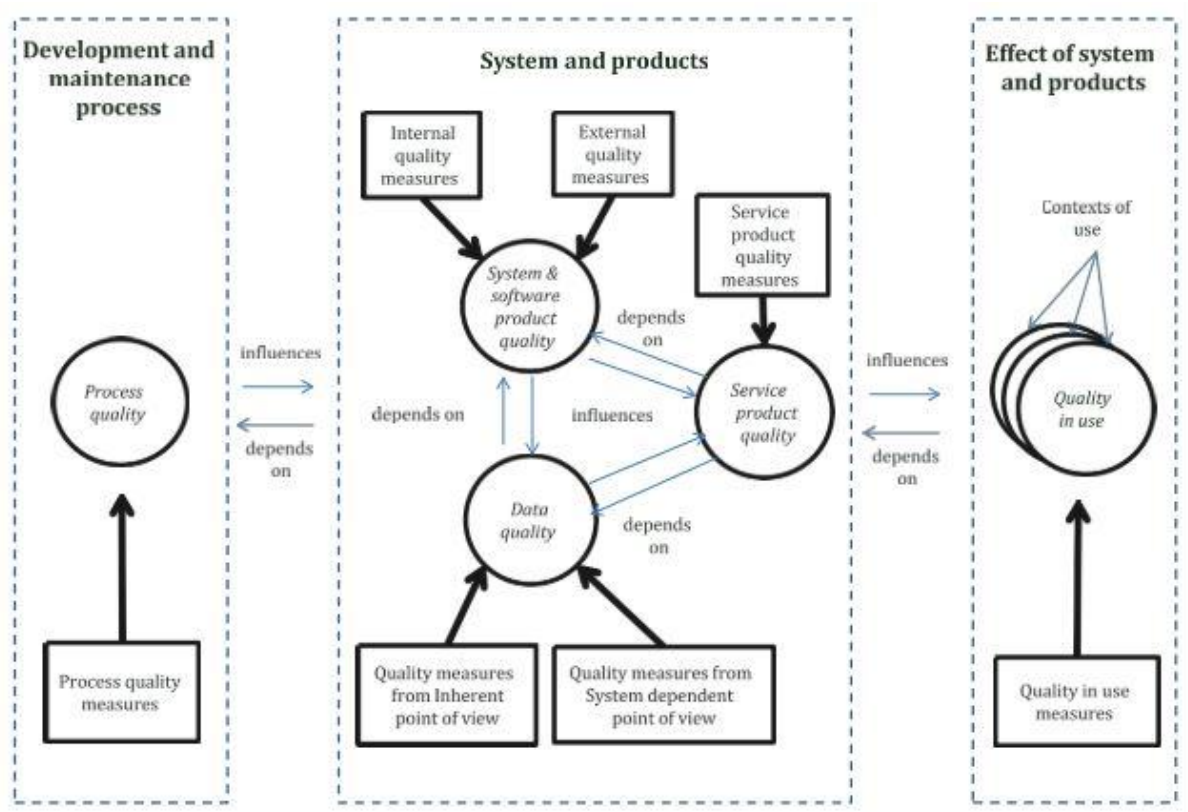


Abb. 7 – Abhängigkeiten der Qualitätsmodelle und Maße der SQuaRE Reihe<sup>7</sup>

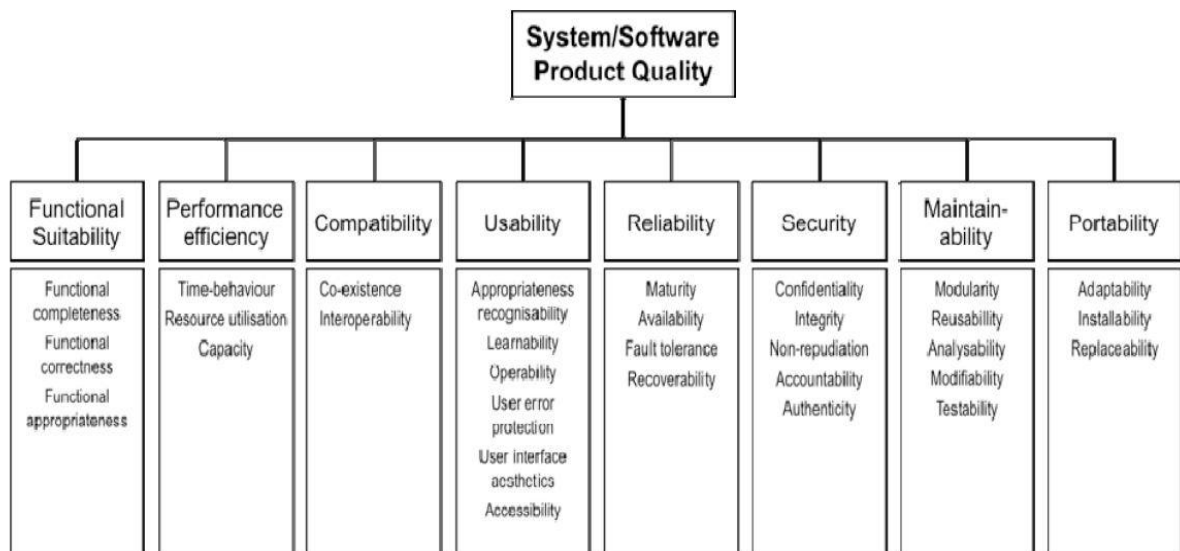
Da die Datenqualität von einem situationsbezogenen Datenbestand abhängig ist und dieser in keiner generischen Form im Rahmen der Thesis vorliegt, wird das Qualitätsmodell der Datenqualität und der zugehörigen 63 Software Qualitätsmaße aus ISO/IEC 25024 nicht weiter untersucht. Ebenfalls ist das Qualitätsmodell der Nutzungsqualität und der verknüpften 36 Software Qualitätsmaße aus ISO/IEC 25022 kein Untersuchungsgegenstand der Bachelorarbeit, da diese von einzelnen Individuen, deren subjektiven Empfindungen, kognitiven Fähigkeiten und Umgebungsbedingungen abhängig sind (vgl. [23], S. 35). Ausgeschlossen von der Untersuchung der Thesis sind außerdem die Bereiche der Norm 2503n, 2504n und die Ergänzungen ISO/IEC 25050 bis ISO/IEC 25099, da Leitlinien für Anforderungen, Evaluationsmodelle für Spezifikationen und technische Berichte für spezifische Bereiche keinen direkten Einfluss auf die Ausarbeitung besitzen. Der Fokus der Bachelorarbeit liegt auf dem System- und Software Produkt Qualitätsmodell der ISO/IEC 25010 und den korrelierenden 86 Software Qualitätsmaßen aus Norm ISO/IEC 25023.

#### 2.2.4 Das System- und Software Produkt Qualitätsmodell

Das Qualitätsmodell der System- und Software Produktqualität aus ISO/IEC 25010 definiert die acht Qualitätsmerkmale Funktionalität, Performanz, Kompatibilität, Benutzbarkeit, Zuverlässigkeit, Sicherheit, Wartbarkeit und Portabilität (vgl. [21], S. 3). Jedes dieser Merkmale setzt sich, wie in Abb. 8 dargestellt, aus mehreren Teilmerkmalen zusammen.

<sup>7</sup> Abb. aus ([23], S. 37)



Abb. 8 – System- und Produkt Qualitätsmodell der SQuaRE Reihe<sup>8</sup>

Die Bedeutung der einzelnen Qualitätsmerkmale wird in tabellarischer Form vorgestellt (vgl. Tab. 1), da auf diese im weiteren Verlauf der Bachelorarbeit referenziert wird und ein allgemeingültiges Verständnis elementar für die Nachvollziehbarkeit der Thesis ist.

Qualitätsmerkmal	Beschreibung
Funktionalität	Entspricht dem Grad der von einem System– oder einem Software Produkt bereitgestellten Funktionen im Vergleich zu Spezifikationen und impliziten Bedürfnissen von Stakeholdern.
Performanz	Effizienz eines Systems oder einer Software im Vergleich zu eingesetzten Ressourcen.
Kompatibilität	Grad von austauschbaren Informationen von Produkten, Systemen oder Komponenten mit anderen, die auf derselben Hardware oder Software Umgebung betrieben werden.
Benutzbarkeit	Grad der Effektivität, Effizienz und Zufriedenheit mit dem Stakeholder spezifische Ziele durch Einsatz eines Systems oder Produkts erreichen.
Zuverlässigkeit	Zeitraum, indem ein System, Produkt oder eine Komponente spezifizierte Funktionen unter vorgegebenen Bedingungen bereitstellt.
Sicherheit	Schutzniveau und Autorisierungsstufen für Zugriff durch Personen oder andere Produkte für sensible Daten.
Wartbarkeit	Grad der Effektivität und Effizienz mit dem ein Produkt oder ein System geändert und angepasst werden kann.
Portabilität	Grad der Effektivität und Effizienz mit dem ein System oder Produkt von einer Umgebung zu einer anderen transferiert werden kann.

Tab. 1 – Qualitätsmerkmale des System– und Software Produkt Qualitätsmodells<sup>9</sup>

Die Ausprägung jedes Teilmerkmals in Abb. 8 wird durch ein oder mehrere Software Qualitätsmaße bestimmt. Für jedes Software Qualitätsmaß ist eine Metrik und ein eindeutiger Identifikator (ID) im SQuaRE Framework definiert (vgl. Tab. 29). Die ID setzt sich aus dem ersten Zeichen des korrelierenden Qualitätsmerkmals, gefolgt von den ersten beiden des zugehörigen Teilmerkmals, einer Sequenznummer bezogen auf die Subcharakteristik und dem Buchstaben G für generisch oder S für spezifisch zusammen (vgl. [24], S. 7). Generische

<sup>8</sup> Abb. aus ([21], S. 4)<sup>9</sup> Eigene Darstellung nach ([21], S. 10 ff.)

Software Qualitätsmaße sind für jede selektierte Subcharakteristik von ISO/IEC 25023 explizit für die Anwendung vorgeschrieben. Falls ein generisches Software Qualitätsmaß nicht genutzt wird, ist dieses ausdrücklich zu begründen (vgl. [24], S. 2). Die Nutzung spezifischer Software Qualitätsmaße ist optional und nur in Situationen in denen sie relevant sind angemessen (vgl. [24], S. 7). ISO/IEC 25023 beinhaltet im Annex A eine Erweiterung der definierten Software Qualitätsmaße mit den Parametern highly recommended (HR), recommended (R) und user's discretion (UD), welche mit der Nutzungsempfehlung durch die ID mit den Attributen G und S korreliert und vereinzelt Antagonismen aufweist. Zusätzlich verweisen die Parameter HR, R und UD auf die Validität von Messergebnissen, die durch die Verwendung der Software Qualitätsmaße generiert werden. Die Kategorie HR beinhaltet Software Qualitätsmaße die Messresultate mit einer hohen Zuverlässigkeit und einer expliziten Nutzungsempfehlung durch die ISO/IEC 25023 generiert. Software Qualitätsmaße mit dem zugeordneten Parameter R stellen ebenfalls robuste Messergebnisse bereit, werden aber nur genutzt, wenn diese im spezifischen Kontext der Anwendung relevant sind. Das Segment der Software Qualitätsmaße mit dem Parameter UD ist aufgrund der unbekannten Validität der Messergebnisse nur als Referenz bei der Entwicklung neuer Software Qualitätsmaße zu nutzen (vgl. [24], S. 31). Zur Differenzierung werden im weiteren Verlauf der Thesis die Parameter G und S als Nutzungsempfehlung durch die Norm und die Parameter HR, R und UD als Validität von Messergebnissen bezeichnet. Das System- und Software Qualitätsmodell separiert interne und externe Eigenschaften von Software Qualitätsmaßen. Die Bewertung interner Eigenschaften eines Software Produktes erfordert eine interne Sicht und die Kenntnis des Aufbaus eines Software Produktes. Interne Software Qualitätsmaße erlauben die Messung von intrinsischen Eigenschaften in Form von statischen Attributen und ermöglichen Aussagen über den Grad der Erfüllung von Qualitätszielen der Architektur, Struktur und integrierten Komponenten (vgl. [11], S. 12 f.). Es ist vergleichbar mit dem White-Box Prinzip im Software Test Umfeld (vgl. [21], S. 31) und bildet die Voraussetzungen zur Befriedigung von Qualitätszielen mit einer Abhängigkeit zu externen Eigenschaften eines Software Produktes und der Nutzungsqualität (vgl. [21], S. 27). Externe Anforderungen beziehen sich auf externe Qualitätsziele in Form des Verhaltens von Software Produkten anhand der Validierung und Verifikation von Rückgabewerten zur Laufzeit. Der innere Aufbau der Software ist bei Nutzung externer Software Qualitätsmaße nicht relevant. Die Messung externer Eigenschaften entspricht dem Black-Box Prinzip eines Software Tests (vgl. [21], S. 31).

### **2.2.5 Abgrenzung der Begriffe Software Qualitätsmaß und Metrik**

Software Qualitätsmaße generieren quantitative Aussagen über Softwareentwicklungsprozesse und Software Produkte. Dieses schafft die Voraussetzungen zur Kontrolle der Software Qualität und ermöglicht es Vorhersagen abzuleiten, um Risiken vorzubeugen (vgl. [7], S. 377 f.). Die Grundlage von Software Qualitätsmaßen bilden heuristische Regeln die zu Normen erhoben wurden. Die Resultate von Messungen werden auf unterschiedlichen Skalen abgebildet (vgl. [3], S. 6). Die Messergebnisse dienen zur Identifikation von Anomalien und können durch Zählung, Berechnung oder Bewertung spezifischer Objekte und deren Attribute generiert werden (vgl. [25], S. 244). Sie weisen nach BALZERT entweder objektive oder subjektive, absolute oder relative, explizite oder abgeleitete, dynamische oder statische, vorhersagende oder erklärende, prozess- oder produktorientierte, globale oder spezielle Eigenschaften auf (vgl. [7], S. 385). Die Zuordnung von Objekten der realen Welt zu Messwerten wird als Maßbestimmung bezeichnet (vgl. [25], S. 233 f.). In der Literatur finden sich folgende Definitionen des Begriffs Maß:

*“A formal, precise, reproducible, objective mapping of a number or symbol to an empirical entity for characterizing a specific attribute.”*

- EBERT, CHRISTOF DUMKE ([26], S. 523)



*“Set of operations having the object of determining a value of a measure.”*

- ISO/IEC 25000 ([11], S. 4)

Der Begriff Metrik wird im Kontext des Vergleichs zweier Objekte der Realität verwendet (vgl. [25], S. 233 f.):

*A distance vector comparing two  $\rightarrow$  measurements. Aggregation of two or more measurements.“*

- EBERT, CHRISTOF DUMKE ([26], S. 523)

Metriken basieren auf der Erfüllung der Gütekriterien Objektivität, Vergleichbarkeit, Robustheit, Ökonomie, Korrelation und Verwertbarkeit (vgl. [4], S. 248), während Gütekriterien von Softwaremaßen als Objektivität, Zuverlässigkeit, Validität, Normierung, Vergleichbarkeit, Ökonomie und Nützlichkeit in der Literatur beschrieben werden (vgl. [7], S. 383).

### 2.2.6 Identifikation erklärungsbedürftiger Software Maße der ISO/IEC 25023

Das interne Software Qualitätsmaß zur statischen Analyse von Software Produkten mit der ID MMo-2-S und der Bezeichnung CYCLOMATIC COMPLEXITY ADEQUACY (vgl. Tab. 29) setzt die Kenntnis der Definition der zyklomatischen Komplexität voraus. Die Metrik zur Berechnung der zyklomatischen Komplexität wurde von Thomas J. MCCABE entwickelt, indem dieser die zyklomatische Zahl aus der Graphentheorie auf Kontrollflussgraphen anwendete und die Bedeutung für den Software-Test aufzeigte. Das Resultat der Metrik liefert eine Information über die Anzahl der Ablaufmöglichkeiten eines Programms und wird dadurch als Struktur- oder Komplexitätsmetrik bezeichnet (vgl. [4], S. 259 f.). Als Grenzwerte des Qualitätsmaßes definiert MCCABE einem Wert kleiner zehn als niedrig, einen Wert bis 20 als mittel, bis 50 als hoch und über 50 als undurchschaubar (vgl. [5], S. 61 ff.). Zwei unterschiedliche Definitionen der zyklomatische Komplexität sind in Formel (2.2), auf Grundlage von Kontrollflussgraphen und in Tab. 2, bezogen auf den Quellcode einer Software, angegeben (vgl. [5], S. 64).

$$V(G) = |E| - |N| + 2 \quad (2.2)$$

$V(G)$  gibt die zyklomatische Komplexität des Kontrollflussgraphen  $G$  an. Der Parameter  $E$  resultiert aus der Anzahl der Kanten und Parameter  $N$  aus der Anzahl der Knoten im untersuchten Graphen.

Bemerkung	Operand	Operator
	Zyklomatische Komplexität	=
if	Anzahl der Verzweigungen	
for, while, repeat...	Anzahl der Schleifen	+
case, switch	Je: Anzahl der Zweige -1	+
	1	+

Tab. 2 – Definition der zyklomatischen Komplexität auf Codebasis<sup>10</sup>

Die sinnvolle Anwendung des Software Qualitätsmaßes der zyklomatischen Komplexität ist vom spezifischen Messobjekt abhängig. Exemplarisch ist für objektorientierte Programme die zyklomatische Komplexität nur eingeschränkt anwendbar (vgl. [9], S. 497). Ebenso liefert die Metrik keinen Aufschluss über die Komplexität im Bereich von Daten (vgl. [25], S. 267).

<sup>10</sup> Eigene Darstellung (vgl. [5], S. 64)

## 2.3 Kiviat-Diagramme

Kiviat-Diagramme bieten eine Möglichkeit zur Visualisierung von mehreren voneinander unabhängigen Maßzahlen und deren Grenzwerten (vgl. Abb. 9). Sie werden aufgrund ihrer kreisförmigen Form auch als Radar- oder Netzdiagramme bezeichnet. Die Anzahl der zu darstellenden Maße gibt die Anzahl der Bereiche vor, die durch Achsen getrennt werden. Jede Achse wird in drei Segmente unterteilt. Das mittlere Element definiert den Normbereich. Die Größe der eingeschlossenen Fläche wird als kumuliertes Qualitätsmaß bezeichnet. Alle Messwerte haben als kumuliertes Qualitätsmaß, durch die gleichmäßigen Achsenverteilung, eine äquivalente Gewichtung (vgl. [4], S. 270 ff.).

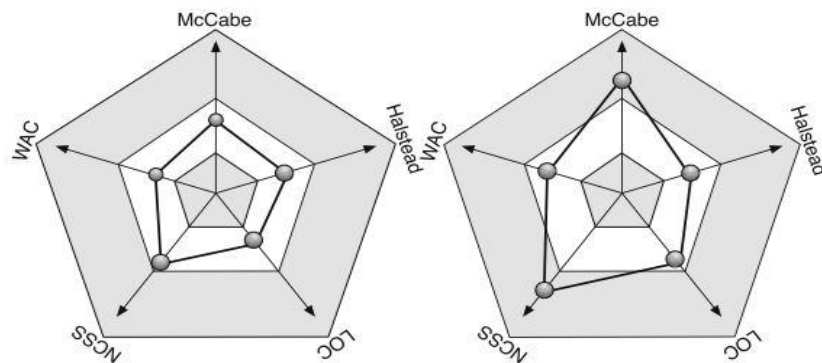


Abb. 9 – Beispielhafte Kiviat-Diagramme für Software Qualitätsmaße<sup>11</sup>

Zur Auswertung der Ergebnisse werden im Rahmen dieser Thesis modifizierte Formen von Kiviat-Diagrammen verwendet (vgl. Abb. 10). Die Unterteilung der Achsen in verschiedene Segmente entfällt, da die Grenzwerte der Akzeptanz abgestimmt und in Anforderungsdokumenten festgelegt werden, die dieser Arbeit nicht zugrunde liegen. Die Messwerte stellen die prozentuale Abdeckung der Bezeichnungen am äußeren Rand der Achse dar. Je größer die Ausprägung des gemessenen Wertes, desto qualifizierter ist das Untersuchungsobjekt für den jeweiligen Anwendungsfall.

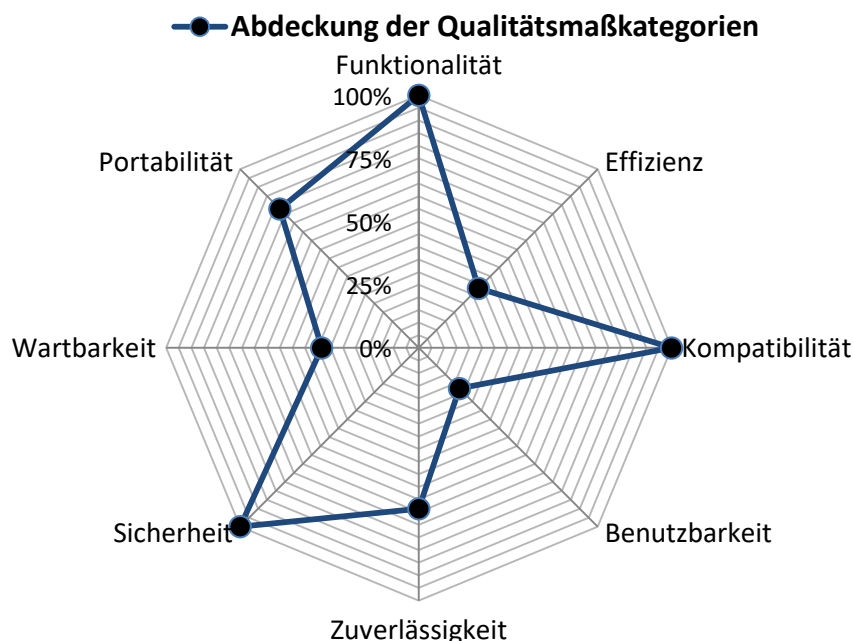


Abb. 10 – Beispielhafte Darstellung von Messergebnissen<sup>12</sup>

<sup>11</sup> Abb. aus ([4], S. 271)

<sup>12</sup> Eigene Darstellung

## 2.4 Werkzeuge

In der Software Entwicklung werden Werkzeuge zur Verbesserung der Projekt- und Produktqualität eingesetzt. Sie steigern die Produktivität, indem manuelle Tätigkeiten automatisiert werden (vgl. [2], S. 346 f.). LIGGESMEYER beschreibt sechs Ziele die durch den Einsatz von Werkzeugen befriedigt werden. Neben der Durchführung von Messungen und Abwicklung von komplexen und monotonen Tätigkeiten, stellen sie Informationen über Prüfstände bereit, erzeugen Protokolldateien zur Dokumentation und erhöhen die Effizienz (vgl. [25], S. 392). Anhand ihrer Funktionalität lassen sich Werkzeuge klassifizieren. Tab. 3 zeigt einen Klassifizierungsansatz auf den im weiteren Verlauf der Bachelorarbeit referenziert wird.

Werkzeugklasse	Werkzeugtyp	Beschreibung
Dynamische Testwerkzeuge	Strukturorientierte Testwerkzeuge	Liefern Informationen über Teststände und dokumentieren anhand Testüberdeckungsprotokolls durchgeführte Tests. Besitzen z.T. integrierte statische Analysefunktionen.
	Funktionsorientierte Testwerkzeuge	Erzeugen und planen Testfälle. Dokumentieren Testdaten und erwartete Ergebnisse.
	Regressionswerkzeuge	Automatisierung und Wiederholung der durchzuführenden Tests. Aufzeichnung der Testdaten und Testergebnisse.
	Leistungs- und Stresstestwerkzeuge	Belastung eines Testlings. Leistungstest evaluiert die Schwellwerte des Testlings und Stresstest simuliert Verhalten bei Überlast.
Statische Analysewerkzeuge	Messwerkzeuge	Extrahierung von Informationen aus Quellcode und Aufbereitung der Ergebnisse.
	Stilanalysatoren	Bewerten den Quellcode anhand vordefinierter semantischer und syntaktischer Programmierregeln.
	Werkzeuge zur Erzeugung von Grafiken und Tabellen	Aufbereitung von komplexen Informationen zur verständlichen Darstellung in Graphen, Diagrammen und Tabellen.
	Slicing Werkzeuge (statisch, dynamisch)	Generierung von Informationen über Wirkzusammenhänge von Anweisungen und Werten.
	Datenflussanomalieanalyse Werkzeuge	Statische Analyse des Datenflusses zur Identifikation von Abweichungen.
Formale Verifikationswerkzeuge	-	Nachweis von Eigenschaften einer Software durch algebraische Verfahren.
Modellierende und analysierende Werkzeuge	FMECA-Werkzeuge	Werkzeuge zur Nutzung von FMECA-Formblättern zur Bestimmung von Risiken und damit verbundenen Auswirkungen.
	Fehlerbaumanalyse-Werkzeuge	Unterstützung bei qualitativer und quantitativer Auswertung von Fehlerbäumen bei sicherheitskritischer Systementwicklung.
	Markov-Modellierungswerkzeuge	Quantitative Auswertung von Markov-Modellen. Stochastische Zuverlässigkeitsanalyse von Software.

Tab. 3 – Klassifizierungen von Werkzeugen nach LIGGESMEYER<sup>13</sup>

<sup>13</sup> Eigene Darstellung nach ([25], S. 393 ff.)

### 3 Hypothesen

In diesem Kapitel werden Hypothesen aufgestellt, die im weiteren Verlauf der Bachelorthesis zu untersuchen sind. Den übergeordneten Rahmen bildet dabei die Gesamthypothese  $H_0$ . Diese baut auf einzelnen Teilhypothesen  $T_1$  bis  $T_n$  auf, die spezifische Themenbereiche behandeln und für die Beantwortung der Gesamthypothese relevant sind. Die Teilhypothese  $T_3$  basiert auf den Ergebnissen der Teilhypothesen  $T_1$  und  $T_2$ .

*$H_0$ : Klassifikationen von Werkzeugen und Software Qualitätsmaßen der Norm ISO/IEC 25023 können auf Basis der Eignung zur Operationalisierung zugeordnet, evaluiert und visuell dargestellt werden.*

#### 3.1 Teilhypothese 1

Eine Aussage zur ersten Teilhypothese erfordert die Analyse der im SQuaRE Framework beschriebenen Software Qualitätsmaße. Es wird untersucht, ob Klassifizierungsansätze aus der Literatur auf das System- und Software Produkt Qualitätsmodell adaptiert werden können. Anschließend wird eruiert, welche Klassifikationsmöglichkeiten aus der Norm direkt abgeleitet werden können und welche Software Qualitätsmaße von externen Eingangsgrößen abhängen.

*$T_1$ : Die Software Qualitätsmaße aus ISO/IEC 25023 des System- und Software Produkt Qualitätsmodells lassen sich klassifizieren.*

#### 3.2 Teilhypothese 2

Zur Beantwortung der zweiten Teilhypothese werden ausschließlich Klassifikationsansätze von Werkzeugen zur Messung von Software Qualität aus der Literatur genutzt, da in der Normenreihe ISO/IEC 25000 keine Werkzeuge thematisiert werden.

*$T_2$ : Den Software Qualitätsmaßen aus ISO/IEC 25023 des System- und Software Produkt Qualitätsmodells lassen sich Werkzeugklassen gemäß ihrer Eignung zuordnen und bewerten.*

#### 3.3 Teilhypothese 3

Die Ergebnisse der Untersuchung der dritten Teilhypothese sind Diagramme die aufzeigen, inwiefern spezifische Werkzeuge zur Messung von Software Qualitätsmaß-Klassifizierungen der Teilhypothese  $T_1$  geeignet sind. Die Auswahl der Werkzeuge erfolgt auf Grundlage der in  $T_2$  evaluierten Werkzeugklassen und deren Eignung zur Messung der Software Qualitätsmaße. Die Visualisierung der Resultate der Untersuchung erfolgt in Form von Kiviat-Diagrammen.

*$T_3$ : Einzelne Werkzeuge können anhand der Tauglichkeit zur Messung der Klassifizierungen von Software Qualitätsmaßen aus Norm ISO/IEC 25023 bewertet und visuell dargestellt werden.*

## 4 Auswertung der Hypothesen

### 4.1 Auswertung der Teilhypothese 1

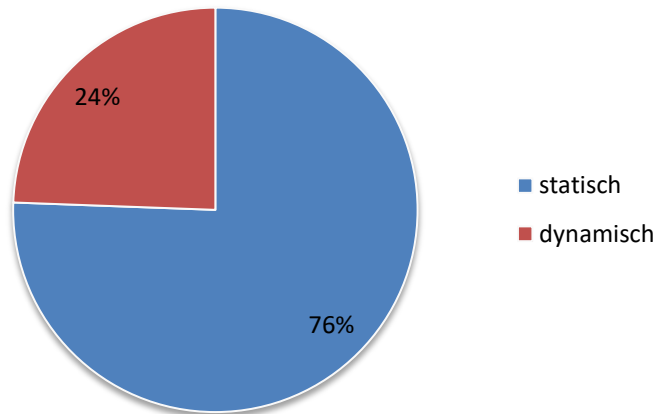
In der Literatur finden sich verschiedene Ansätze um Software Qualitätsmaße zu klassifizieren. Ein Ansatz der Einteilung in unterschiedliche Kategorien besteht anhand der in Kapitel 2.2.5 dargestellten Eigenschaften von Maßen. Exemplarisch wurde eine Klassifikation nach dem Kriterium der statischen und dynamischen Eigenschaften der Software Qualitätsmaße des System- und Software Produkt Qualitätsmodells vorgenommen. Neben den Ansätzen der Klassifikation aus der Literatur, werden im Rahmen von T<sub>1</sub> Klassifikationen für die Software Qualitätsmaße des System- und Software Produkt Qualitätsmodells aus Norm ISO/IEC 25023 abgeleitet und analysiert. Eine tabellarische Übersicht der untersuchten Klassifikationen befindet sich in Tab. 4.

Klassifizierung	Bezugsquelle
Statische und dynamische Software Qualitätsmaße	Literatur
Interne, externe und hybride Software Qualitätsmaße	ISO/IEC 25023
HR, R und UD Software Qualitätsmaße	ISO/IEC 25023
Korrelierende Qualitätsmerkmale von Software Qualitätsmaßen	ISO/IEC 25023
Generische und spezifische Software Qualitätsmaße	ISO/IEC 25023
Eingangsgrößen für Software Qualitätsmaße	ISO/IEC 25023

Tab. 4 – Klassifikationsansätze in T<sub>1</sub><sup>14</sup>

Die Darstellungsform der Klassifikationen von Software Qualitätsmaßen entspricht Kreisdiagrammen. Die prozentualen Anteile ergeben sich aus der absoluten Häufigkeit des Auftretens der im Kontext spezifischen Kriterien, im Verhältnis zur Summe aller Software Qualitätsmaße des System- und Software Produkt Qualitätsmodells. Eine Ausnahme besteht in der Darstellung der Klassifikation der Abhängigkeiten von externen Eingangsgrößen, da ein Software Qualitätsmaß von mehr als einer Eingangsgröße abhängen kann. Zur Darstellung wird die Anzahl der identifizierten Abhängigkeiten von Software Qualitätsmaßen in Kiviat-Diagrammen beschrieben, da diese die Möglichkeit der Visualisierung der Abhängigkeit eines Maßes von mehreren Eingangsgrößen ermöglichen. Um den Detaillierungsgrad zu erhöhen werden anschließend die Abhängigkeiten von Eingangsgrößen der Software Qualitätsmaße auf der Ebene korrelierender Qualitätsmerkmale verfeinert. Jede Ausprägung einer Klassifikation wird exemplarisch durch die Beschreibung eines Software Qualitätsmaßes begründet. Wurde die Definition des Software Qualitätsmaßes bereits bei einer vorherigen Klassifikation beschrieben, wird diese nicht wiederholt, sondern auf die ID des Software Qualitätsmaßes verwiesen. Details weiterer Berechnungsvorschriften und Definitionen der analysierten Software Qualitätsmaße sind Tab. 29 im Anhang B zu entnehmen. Die Ergebnisse der Zuordnung von Software Qualitätsmaßen des System- und Software Produkt Qualitätsmodells zu statischen und dynamischen Charakteristiken ist in Tab. 32 im Anhang D aufgeführt. Die aggregierten Resultate der Analyse sind in Abb. 11 aufgeführt.

<sup>14</sup> Eigene Darstellung

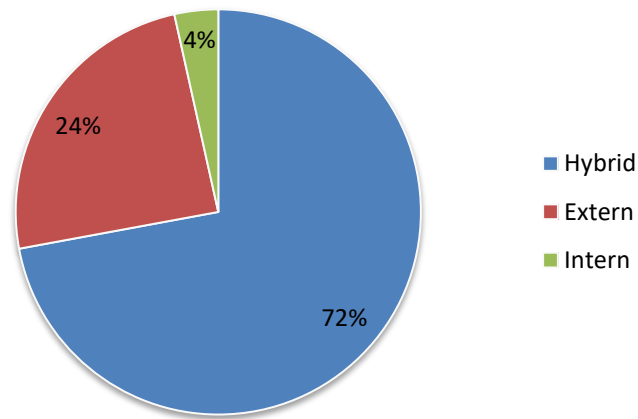


**Abb. 11 – Klassifikation anhand statischer und dynamischer Eigenschaften<sup>15</sup>**

Die betrachteten Software Qualitätsmaße weisen mit 76 Prozent verstärkt statische Eigenschaften, ohne einen Bezug zum Zeitpunkt der Messung, auf. Die Anzahl der dynamischen Software Qualitätsmaße, mit einer Abhängigkeit zum Zeitraum der Messung, ist mit 24 Prozent geringer als die Häufigkeit der statischen Software Qualitätsmaße im System- und Software Produkt Qualitätsmodell. Durch einen Vergleich der Software Qualitätsmaße mit den IDs RMa-2-G und MMo-2-S wird exemplarisch aufgezeigt, wie sich ein Software Qualitätsmaß mit dynamischen von einem Software Qualitätsmaß mit statischen Eigenschaften unterscheidet. RMa-2-G ist ein aus dem Qualitätsmerkmal Zuverlässigkeit abgeleitetes Software Qualitätsmaß mit der Bezeichnung MEAN TIME BETWEEN FAILURE (MTBF). Es wird berechnet aus der Zeit in der eine Software zur Verfügung steht im Verhältnis zu den Fehlern, die in diesem Zeitraum aufgetreten sind. Das Software Qualitätsmaß weist durch den Bezug zum Zeitraum der Verfügbarkeit des Software Produktes eine dynamische Charakteristik auf. MMo-2-S ist ein Software Qualitätsmaß des korrelierenden Qualitätsmerkmals Wartbarkeit mit dem Namen CYCLOMATIC COMPLEXITY ADEQUACY und stellt einen Indikator bereit der angibt, wie viele Module einer Software eine akzeptable zyklomatische Komplexität überschreiten. Das Software Qualitätsmaß wird berechnet, indem das Verhältnis der Software Module die den Schwellwert der zyklomatischen Komplexität überschreiten zur Summe der implementierten Software Module gebildet und das Ergebnis vom Wert 1 subtrahiert wird. Das Software Qualitätsmaß mit der ID MMo-2-S besitzt keinen zeitlichen Bezug und weist ausschließlich statische Eigenschaften auf. Mittels der durchgeführten Auswertung der Häufigkeit des Auftretens von Software Qualitätsmaßen mit statischen und dynamischen Eigenschaften wurde aufgezeigt, dass Ansätze aus der Literatur zur Klassifikation auf das System- und Software Produkt Qualitätsmodell der Norm ISO/IEC 25010 anwendbar sind. Zusätzliche Gegenüberstellungen weiterer Klassifizierungsansätze der Literatur zum Nachweis der Anwendbarkeit sind im Rahmen von T<sub>1</sub> nicht vorgesehen und werden deshalb ohne die Durchführung einer Zuordnung zu den Software Qualitätsmaßen nur noch genannt. Nach BALZERT können Software Qualitätsmaße außerdem durch objektive oder subjektive, absolute oder relative, explizite oder abgeleitete, vorhersagende oder erklärende, prozess- oder produktorientierte und globale oder spezielle Eigenschaften klassifiziert werden (vgl. Kapitel 2.2.5). Anderere Ansätze zur Bildung von Software Qualitätsmaßkategorien besteht in der Zuordnung anhand Phasen der relevanten Vorgehensmodelle (vgl. [9], S. 226) oder in der Einordnung der Software Qualitätsmaße in den Kontext, in dem diese erhoben werden. Bei letzterem lassen sich Produkt-, Prozess- und Projektmaße unterscheiden (vgl. [5], S. 55 f.). Ohne die Durchführung einer

<sup>15</sup> Eigene Darstellung nach Tab. 31

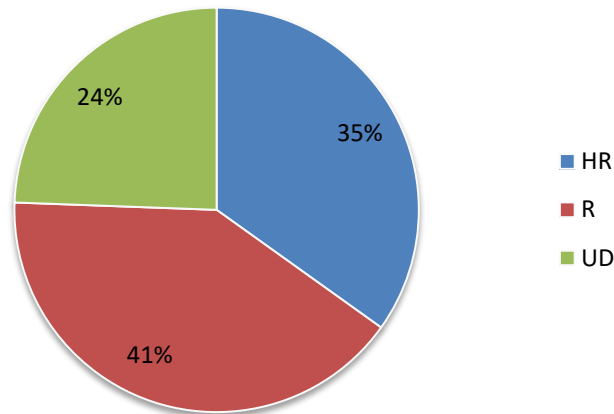
Untersuchung der einzelnen Software Qualitätsmaße kann aus der Definition des System- und Software Qualitätsmodells die Aussage abgeleitet werden, dass die Gesamtheit der Software Qualitätsmaße der Kategorie der Produktmaße angehören. Ein Ansatz zur Klassifikation von Software Qualitätsmaßen auf Grundlage der Definitionen in der Norm ISO/IEC 25023 ergibt sich aus dem vorgegebenen Perspektive, der Validität der Messresultate, der generischen und spezifischen Eigenschaften, der Abhängigkeit von externen Eingangsgrößen und der Zugehörigkeit zu übergeordneten Qualitätsmerkmalen.



**Abb. 12 – Klassifizierung anhand der Perspektive der Maße im SQuaRE Framework<sup>16</sup>**

Abb. 12 beschreibt den prozentualen Anteil der absoluten Häufigkeit von internen, externen und hybriden Software Qualitätsmaßen in der Norm ISO/IEC 25023 (vgl. Kapitel 2.2.4). Die internen, externen und hybriden Eigenschaften bilden die Klassifikation der Software Qualitätsmaße auf Basis der Zuordnung im SQuaRE Framework. Die Häufigkeit der exklusiv externen Software Qualitätsmaße übersteigt mit 24 Prozent signifikant die Anzahl der ausschließlich internen Software Qualitätsmaße, die zu 4 Prozent im Qualitätsmodell enthalten sind. Den Schwerpunkt bilden mit 72 Prozent die hybriden Software Qualitätsmaße, die Kenntnis des inneren Aufbaus der Software voraussetzen und zusätzlich einen Bezug zu äußerlich sichtbarem Verhalten aufweisen. Daraus folgt, dass überwiegend die von Außen sichtbaren Eigenschaften eines Software Produktes durch die Metriken des System- und Software Produkt Qualitätsmodells fokussiert werden. Die Anzahl der exklusiv internen messbaren Attribute ist minimal. Exemplarisch wird das Software Qualitätsmaß des korrelierenden Qualitätsmerkmals Zuverlässigkeit mit ID RAv-2-G und der Bezeichnung MEAN DOWN TIME mit dem, bereits für die dynamische und statische Klassifikation beschriebenen, Software Qualitätsmaß MMo-2-S verglichen. RAv-2-G gibt die durchschnittliche Ausfallzeit eines Software Produktes an und ist ein exklusiv externes Software Qualitätsmaß. Es wird durch das Verhältnis der Zeit in dem das Software System nicht zur Verfügung steht zur Anzahl der beobachteten Abstürze berechnet. Der Zeitraum in dem die Software nicht zur Verfügung steht und die Anzahl der Abstürze sind messbar, ohne die innere Struktur des Aufbaus der Software zu kennen. Im Gegensatz dazu bezieht sich die zyklomatische Komplexität (vgl. Kapitel 2.2.6) des internen Software Qualitätsmaßes MMo-2-S auf die Architektur des Quellcodes der Software.

<sup>16</sup> Eigene Darstellung nach Tab. 29

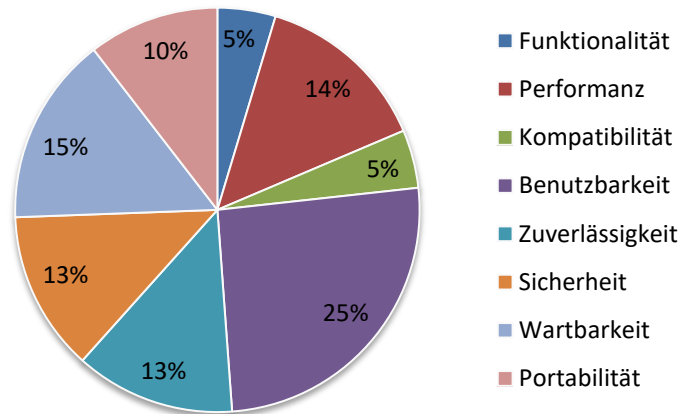


**Abb. 13 – Klassifizierung Maße anhand Validität der Messergebnisse<sup>17</sup>**

In Abb. 13 werden die prozentualen Anteile der absoluten Häufigkeit der in ISO/IEC 25023 enthaltenen Software Qualitätsmaße mit den Eigenschaften HR, R und UD dargestellt. Die Kategorien HR, R und UD repräsentieren die Klassifikation der Software Qualitätsmaße nach, der im Qualitätsmodell definierten, Nutzungsempfehlung und der Validität der Messergebnisse einer Metrik (vgl. Kapitel 2.2.4). Am häufigsten sind mit 41 Prozent die Software Qualitätsmaße des Qualitätsmodells der Kategorie R zugeordnet. Mit 35 Prozent sind die Software Qualitätsmaße der Kategorie HR am zweithäufigsten im untersuchten Qualitätsmodell enthalten. Die Gruppierung HR und R generieren Messergebnisse mit einer hohen Validität. Die kleinste absolute Häufigkeit im Qualitätsmodell bilden mit 24 Prozent Software Qualitätsmaße der Kategorie UD, deren Zuverlässigkeit der Messresultate von der Norm als unbekannt eingestuft sind und die nur als Referenz bei der Entwicklung neuer Software Qualitätsmaße genutzt werden sollten. Der Schwerpunkt liegt folglich bei Software Qualitätsmaßen deren Metriken valide Indikatoren bereitstellen und laut der Norm immer genutzt werden sollten, wenn diese im jeweiligen Kontext der Nutzung anwendbar sind. Das bereits für die Klassifikation der Perspektive verwendete Software Qualitätsmaß mit der ID RAv-2-G zur Messung der mittleren Ausfallzeit einer Software ist der Kategorie R zugeordnet. Das Software Qualitätsmaß zur Messung der Funktionsabdeckung mit der ID FCp-1-G und der Bezeichnung FUNCTIONAL COVERAGE ist der Klasse HR zugewiesen. Die Metrik ist definiert als das Verhältnis von fehlenden Funktionen einer Software zu Funktionen, die in der zugehörigen Spezifikation enthalten sind, subtrahiert vom Wert 1. Die Messresultate der Software Qualitätsmaße mit den IDs RAv-2-G und FCp-1-G sind laut Definition der Norm zuverlässige Indikatoren, um Aussagen über die Qualität von Software Produkten zu treffen. Das bereits zur Klassifikation der statischen und dynamischen Eigenschaften und der Klassifikation der Perspektive verwendete Software Qualitätsmaß mit der ID MMo-2-S ist der Kategorie UD zugeordnet und generiert Messresultate mit unbekannter Validität.

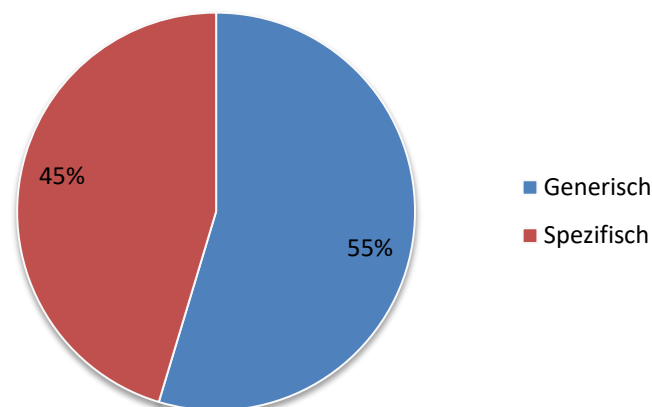
<sup>17</sup> Eigene Darstellung nach Tab. 29





**Abb. 14 – Klassifizierung Maße anhand korrelierender Qualitätsmerkmale<sup>18</sup>**

Die Klassifikation der Software Qualitätsmaße des System- und Software Produkt Qualitätsmodells der SQuaRE Reihe durch korrelierende Qualitätsmerkmale ist in Abb. 14 visualisiert. Die prozentualen Anteile berechnen sich aus der absoluten Häufigkeit der Software Qualitätsmaße die den Qualitätsmerkmalen Funktionalität, Performanz, Kompatibilität, Benutzbarkeit, Zuverlässigkeit, Sicherheit, Wartbarkeit und Portabilität zugeordnet sind. 25 Prozent der Software Qualitätsmaße des Qualitätsmodells stehen in Relation zum Qualitätsmerkmal Benutzbarkeit und bilden so die Kategorie mit der numerisch größten Anzahl der Zuordnung. Software Qualitätsmaße des Qualitätsmerkmals Wartbarkeit sind mit 15 Prozent am zweithäufigsten im Qualitätsmodell definiert. Anschließend folgt die Häufigkeit der Zuordnung von Software Qualitätsmaßen zu den Qualitätsmerkmalen der Effizienz mit 14 Prozent, der Zuverlässigkeit und Sicherheit mit jeweils 13 Prozent und der Portabilität mit 10 Prozent. Die geringste Anzahl der Software Qualitätsmaße sind mit jeweils 5 Prozent die der Qualitätsmerkmale Funktionalität und Kompatibilität. Eine Wertung, ob es positiv oder negativ zu interpretieren ist, wenn einem Qualitätsmerkmal viele Software Qualitätsmaße zur Operationalisierung von Software Qualität zugeordnet sind wird im SQuaRE Framework nicht vorgenommen und ebenfalls im weiteren Verlauf der Arbeit nicht thematisiert, da dieses nicht relevant für die Untersuchung der in Kapitel 3 gestellten Hypothesen ist.



**Abb. 15 – Klassifizierung Maße nach generischen und spezifischen Eigenschaften<sup>19</sup>**

<sup>18</sup> Eigene Darstellung nach Tab. 29

<sup>19</sup> Eigene Darstellung nach Tab. 29

Abb. 15 visualisiert die prozentualen Anteile der absoluten Häufigkeiten von generischen und statischen Eigenschaften (vgl. Kapitel 2.2.4) im System- und Software Produkt Qualitätsmodell. Die Anzahl der generischen Software Qualitätsmaße, die laut dem SQuaRE Framework standardmäßig bei jeder Evaluation von Qualität einer Software genutzt werden sollten, überwiegen mit 55 Prozent die Anzahl der mit 45 Prozent vorhandenen optionalen spezifischen Software Qualitätsmaße, deren Nutzung von der Normenreihe nur als sinnvoll eingestuft wird, wenn diese im jeweiligen Kontext relevant sind. Das bereits bei der Klassifikation nach statischen und dynamischen Eigenschaften, der Perspektive und der Validität von Messergebnissen verwendete Software Qualitätsmaß mit ID MMo-2-S ist der Klasse der spezifischen Software Qualitätsmaße zugeordnet. Das ebenfalls bereits bei der Klassifikation anhand der Validität von Messergebnissen beschriebene Software Qualitätsmaß mit der ID FCp-1-G zur Messung der Funktionsabdeckung ist der Kategorie der generischen Maße zugeordnet und laut Norm, bei der Operationalisierung des Qualitätsmerkmals Funktionalität, immer zu berücksichtigen. Die Klassifikation auf Basis generischer und spezifischer Charakteristika der Software Qualitätsmaße weist aufgrund der impliziten Nutzungsempfehlung durch das System- und Software Produkt Qualitätsmodell Ähnlichkeiten zur Klassifizierung anhand der Validität der Messresultate HR, R, UD und der damit korrelierenden Nutzungsempfehlung auf. Die Ergebnisse der Gegenüberstellung beider Klassifikationen wurden in Abb. 16 aufbereitet.

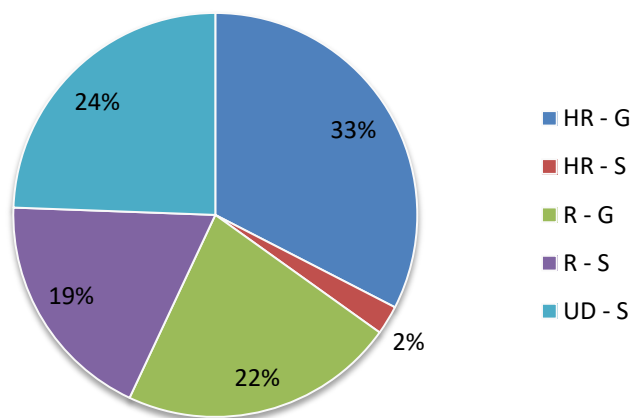


Abb. 16 – Gegenüberstellung der Klassifikationen G/S und HR/R/UD<sup>20</sup>

Die Grafik beschreibt welche generischen und spezifischen Software Qualitätsmaße gleichzeitig die Eigenschaften HR, R und UD aufweisen. Generische Software Qualitätsmaße der Kategorie HR besitzen zu 33 Prozent valide Messresultate und eine hohe Nutzungsempfehlung durch das Qualitätsmodell. Spezifische Software Qualitätsmaße haben zu 2 Prozent die Eigenschaft HR zugewiesen und sind demzufolge optional mit unbekannter Zuverlässigkeit von Messresultaten. Die mit dem Parameter HR einhergehende Nutzungsempfehlung antagonisiert mit der Definition der spezifischen Eigenschaft des Software Qualitätsmaßes. Die Maße sind dementsprechend zu nutzen, wenn dieses der Kontext der Messung erlaubt. Ist die Messung nicht möglich, erfordert die fehlende Nutzung des Software Qualitätsmaßes keine spezifische Begründung. 22 Prozent der generischen Software Qualitätsmaße mit einer hohen Nutzungsempfehlung generieren valide Messergebnisse und sind anzuwenden, wenn das Umfeld der Messung dieses erlaubt. Sind diese im jeweiligen Szenario nicht nutzbar, erfordert die fehlende Anwendung eine explizite Begründung. Die 19 und 24 Prozent der spezifischen Software Qualitätsmaße mit korrelierender Kategorie R und UD haben Messresultate mit unbekannter

<sup>20</sup> Eigene Darstellung nach Tab. 29

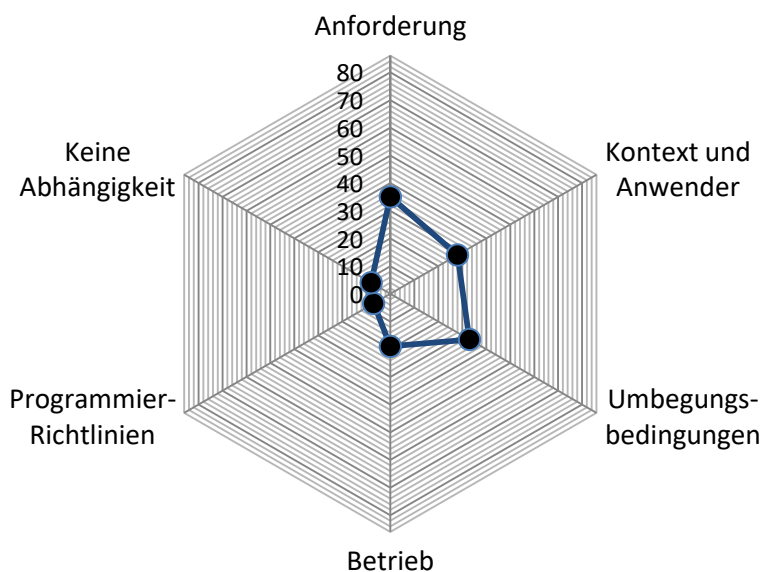
Zuverlässigkeit und sind ausschließlich im Ermessen der Stakeholder und bei der Entwicklung neuer Software Qualitätsmaße anzuwenden. Die letzte Klassifizierung im Rahmen der Bachelorthesis basiert auf einer quantitativen Analyse der 86 Software Qualitätsmaße des System- und Software Produkt Qualitätsmodells. Für jedes Software Qualitätsmaß wurde bewertet, ob die Parameter der Berechnungsvorschrift direkt messbar sind oder eine Abhängigkeit zu externen Eingangsgrößen besitzen. Um eine Aussage über die Abhängigkeiten der Parameter abzuleiten, werden die in der Norm definierten Metriken quantitativ ausgewertet. Die Definitionen der Metriken werden anhand von Schlagworten analysiert. Eine Abhängigkeit besteht, sobald mindestens einem Parameter der Berechnungsvorschrift ein Schlagwort der quantitativen Analyse zugeordnet werden kann. Die Zuordnung von definierten Schlagworten zu Eingangsgrößen ist in Tab. 5 aufgeführt. Eine qualitative Analyse der Semantik der Berechnungsvorschriften zur Identifikation von Abhängigkeiten der Software Qualitätsmaße ist nicht vorgesehen.

Schlagwort	Eingangsgröße	Beschreibung
appropriate	Anforderung	Definiert in Lastenpflicht, Pflichtenheft, Change Request, User Stories, Design Dokumenten
modified		
modification		
require		
requiring		
specific		
specified		
operation	Betrieb	Definiert in Bedienungsanleitungen und Betriebshandbüchern
user document		
job		
installation		
coding rules	Programmierrichtlinien	Festgelegte Regeln der Entwicklungsabteilung.
default value		
diagnostic		
error message		
monitoring		
threshold		
benefit	Kontext und Anwender	Im bestimmten Umfeld eines Anwenders beabsichtigte Handlung durch Nutzung von Software. Kognitive Fähigkeiten des Nutzers.
desire		
expect		
explain		
objective		
scenario		
understand		
user		
capacity	Umgebungsbedingungen	Datenverarbeitendes System zum Betrieb von Software oder mit denen Software interagiert.
component		
device		
environment		
hardware		
memory		
module		
other		
processor		
software		

Schlagwort	Eingangsgröße	Beschreibung
system	Umgebungsbedingungen	Datenverarbeitendes System zum Betrieb von Software oder mit denen Software interagiert.

Tab. 5 – Beschreibung externer Abhängigkeiten der Metriken in ISO/IEC 25023<sup>21</sup>

Das Software Qualitätsmaß mit der Bezeichnung DATA FORMATS EXCHANGEABILITY und der ID CIn-1-G wird berechnet aus der Anzahl der unterstützten und mit anderen Software Systemen austauschbaren Datenformaten im Verhältnis zur Gesamtzahl der spezifizierten Datenformate. Die Schlagworte *specified*, *system*, *software* und *other* weisen auf Abhängigkeiten zu den Eingangsgrößen Anforderung und Umgebungsbedingungen hin. Die Metrik des Software Qualitätsmaßes mit der ID UEp-3-S und der Bezeichnung USER ERROR RECOVERABILITY resultiert aus der Anzahl von Anwenderfehler, die von der Software identifiziert und behoben werden zur Anzahl von möglichen Anwenderfehlern im Betrieb. Die Stichworte *operation*, *system* und *user* deuten auf eine Abhängigkeit vom Kontext oder Anwender, den Umgebungsbedingungen und dem Betrieb hin. Die Metrik des Software Qualitätsmaßes mit der ID MRe-2-S und der Bezeichnung CODING RULES CONFORMITY ist definiert als Verhältnis der Anzahl der Software Module die Programmierkonventionen erfüllen zur Summe implementierter Software Module. Den Formulierungen *specific*, *software*, *module*, *coding rules* und *system* werden Abhängigkeiten zu einer Anforderung, Umgebungsbedingungen und Programmierrichtlinien unterstellt. Die Ergebnisse der quantitativen Analyse zeigen, dass einzelne Software Qualitätsmaße mehreren Kategorien von Einflussgrößen zugeordnet werden können. Die Ergebnisse der vollständigen quantitativen Analyse sind in Abb. 17 durch ein Kiviati-Diagramm aggregiert. Der maximale Wert der Achsen entspricht der Anzahl der 86 Software Qualitätsmaße im System- und Software Produkt Qualitätsmodell.

Abb. 17 – Eingangsgrößen von Metriken der ISO/IEC 25023<sup>22</sup>

Das Kiviati-Diagramm in Abb. 17 zeigt, dass 35 der untersuchten Software Qualitätsmaße eine Abhängigkeit zu einer Anforderung besitzen, ohne deren Vorliegen der Wert des Maßes nicht erhoben werden kann. Die Abhängigkeit zu Anforderungen stellt die höchste Anzahl an Abhängigkeiten von Software Qualitätsmaßen zu Eingangsgrößen dar. Die zweitgrößte Ausprägung an Relationen besteht für 33 Software Qualitätsmaße zu Umgebungsbedingungen. 28

<sup>21</sup> Eigene Darstellung<sup>22</sup> Eigene Darstellung nach Tab. 34

ermittelte Abhängigkeiten von Software Qualitätsmaßen beziehen sich auf den Kontext der Nutzung der Software oder auf die kognitiven Fähigkeiten des Nutzers. Eine Abhängigkeit zu Betriebshandbüchern oder Bedienungsanleitungen weisen 18 der betrachteten Software Qualitätsmaße im System- und Software Produkt Qualitätsmodell auf. Die schwächste Ausprägung von Abhängigkeiten stellen die Programmierkonventionen dar. Die Grafik stellt heraus, dass lediglich acht Software Qualitätsmaße direkt messbar sind und keine Abhängigkeiten zu zusätzlichen Einflussgrößen aufweisen. Im weiteren Verlauf der Bachelorthesis wird untersucht, ob die Abhängigkeiten von Eingangsgrößen für Qualitätsmerkmale aus Abb. 14 gleichmäßig verteilt sind oder spezifische Qualitätsmerkmale verstärkt mit bestimmten Eingangsgrößen korrelieren. Die Darstellungen der Ergebnisse erfolgen weiterhin als Kiviat-Diagramme. Die Skala der Kiviat-Diagramme resultiert aus der absoluten Häufigkeit der den Qualitätsmerkmalen zugeordneten Software Qualitätsmaße im Qualitätsmodell der System- und Software Produktqualität. Dies bewirkt eine unterschiedliche Ausprägung des Maximums der Achsen jedes Kiviat-Diagramms. Für jedes Qualitätsmerkmal wird exemplarisch ein Software Qualitätsmaß mit einer Abhängigkeit zu einer Eingangsgröße aus Tab. 5 beschrieben.

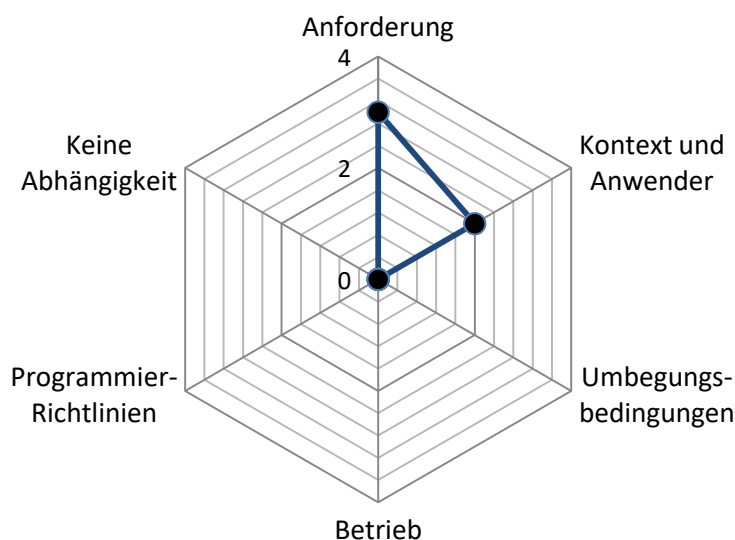


Abb. 18 – Abhängigkeit der Funktionalität von externen Eingangsgrößen<sup>23</sup>

Eine grafische Aufbereitung von Abhängigkeiten der vier untergeordneten Software Qualitätsmaße des Qualitätsmerkmals Funktionalität zeigt für drei und zwei Maße einen Konnex zu den Eingangsgrößen Anforderung, Kontext der Messung und dem Nutzer (vgl. Abb. 18). Die bereits in Kapitel 4.1 beschriebene Metrik mit ID FCp-1-G besteht aus dem Verhältnis von fehlenden zu spezifizierten Funktionen. Spezifizierte Funktionen sind von Lasten- und Pflichtenheften, Change Requests und User Stories abhängig und wurden anhand des Schlagwortes *specified* in der Metrikdefinition des Maßes FCp-1-G der Eingangsgröße Anforderung zugeordnet. Ohne die erfüllte Voraussetzung einer vorliegenden Anforderung ist die Messung des Qualitätsmerkmals Funktionalität nur eingeschränkt möglich.

<sup>23</sup> Eigene Darstellung nach Tab. 34

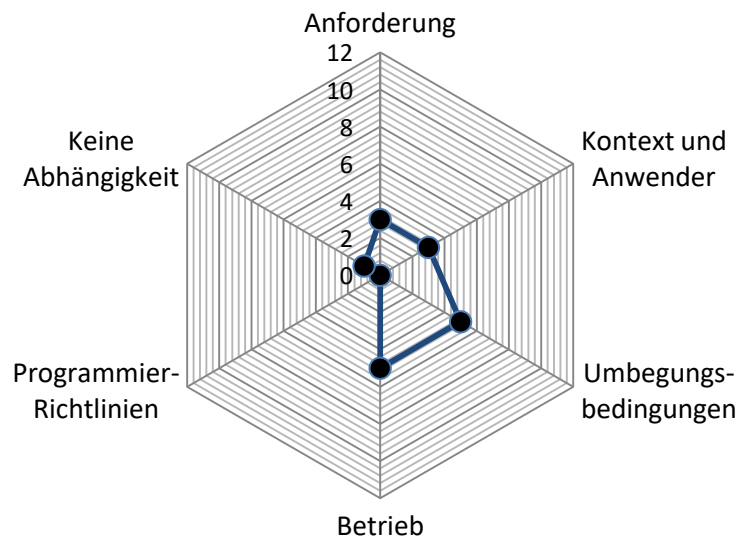


Abb. 19 – Abhängigkeit der Performanz von externen Eingangsgrößen<sup>24</sup>

Die zwölf Metriken des Software Qualitätsmerkmals Performanz sind laut der aufbereiteten Ergebnisse der quantitativen Analyse in Abb. 19 mit jeweils fünf Maßen von den Eingangsgrößen Betrieb und Umgebungsbedingungen abhängig. Jeweils drei Maße besitzen eine Relation zu den Eingangsgrößen Anforderungen, Kontext und Anwender. Das Kapazitätsmaß mit der ID PCa-1-G und der Bezeichnung TRANSACTION PROCESSING CAPACITY ist von keiner Eingangsgröße abhängig. Es wird aus der Anzahl der abgeschlossenen Transaktionen im Verhältnis zur Dauer der Beobachtung berechnet. Das Software Qualitätsmaß mit der ID PRu-4-S und der Bezeichnung BANDWIDTH UTILIZATION wird durch die über einen Zeitraum gemessene Bandbreite einer Übertragung zur verfügbaren Bandbreite gebildet. Das Schlagwort *capacity* der Definition der Parameter deutet auf eine Abhängigkeit zu den Umgebungsbedingungen hin.

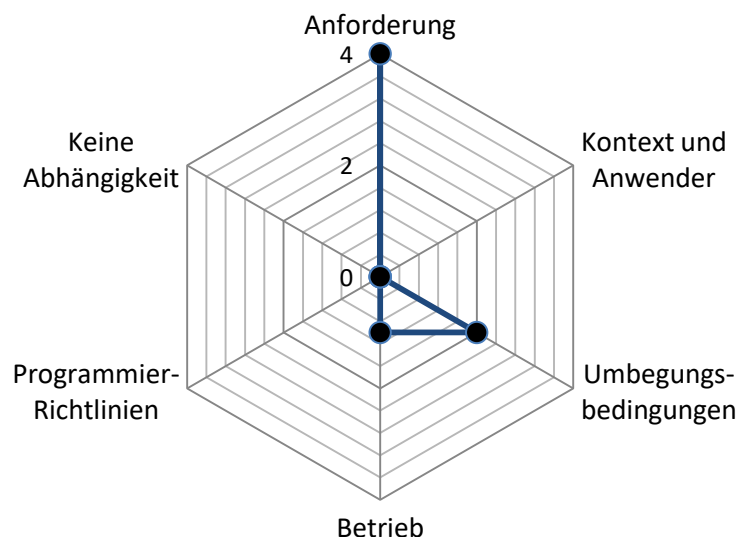


Abb. 20 – Abhängigkeit der Kompatibilität von externen Eingangsgrößen<sup>25</sup>

In Abb. 20 ist ersichtlich, dass die vier Software Qualitätsmaße des Qualitätsmerkmals Kompatibilität vollständig von der Eingangsgröße Anforderung abhängen. Die Existenz von Anforderungsdokumenten bildet folglich die Voraussetzung für die Messung der Kompatibilität.

<sup>24</sup> Eigene Darstellung nach Tab. 34

<sup>25</sup> Eigene Darstellung nach Tab. 34

Zusätzlich besteht für zwei Software Qualitätsmaße eine Abhängigkeit zu Umgebungsbedingungen und für eines eine Abhängigkeit zur Eingangsgröße Betrieb. Das aus dem Qualitätsmerkmal Kompatibilität abgeleitete Software Qualitätsmaß mit der ID CIn-2-G und der Bezeichnung DATA EXCHANGE PROTOCOL SUFFICIENCY wird aus dem Verhältnis der Anzahl der unterstützten zu spezifizierten Protokolle berechnet. Die Abhängigkeit zur Eingangsgröße Anforderung geht aus der Verwendung des Stichwortes *specified* in der Definition der Metrik hervor.

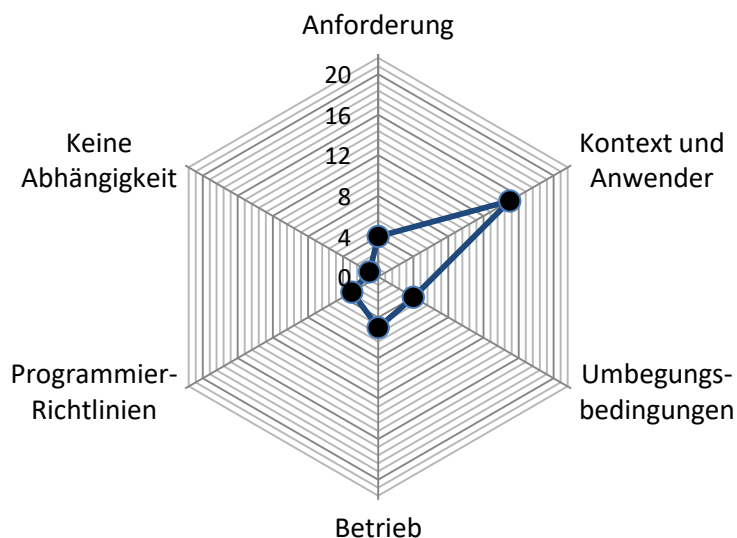
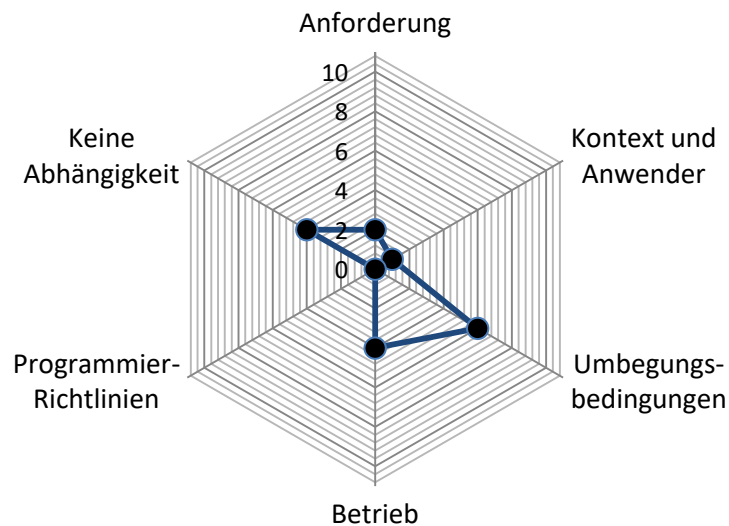


Abb. 21 – Abhängigkeit der Benutzbarkeit von externen Eingangsgrößen<sup>26</sup>

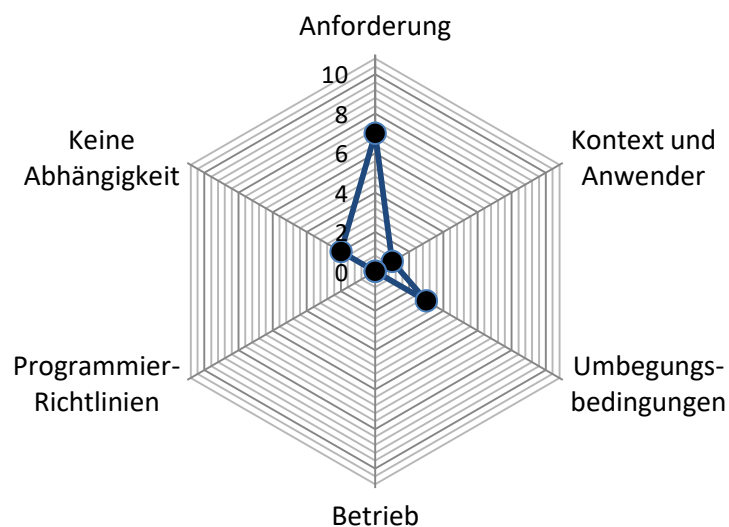
Das Kiviat-Diagramm in Abb. 21 beschreibt die Abhängigkeiten der Parameter der Berechnungsvorschriften von Eingangsgrößen für das Qualitätsmerkmal Benutzbarkeit und den zugehörigen 22 Software Qualitätsmaßen. Die Parameter der Metriken des Qualitätsmerkmals Benutzbarkeit weisen mit 15 abhängigen Software Qualitätsmaßen einen hohen Konnex zum Kontext der Nutzung und dem Anwender auf. Das Software Qualitätsmaß der Benutzbarkeit mit ID ULe-4-S und dem Namen SELF-EXPLANATORY USER INTERFACE bildet sich aus dem Verhältnis von verständlichen Informationselementen zu Elementen, die für die erstmalige Erfüllung einer Aufgabe benötigt werden. Die Schlagworte *user* und *understand* bilden die Abhängigkeit zur Eingangsgröße des Anwenders und dem Kontext der Nutzung. Desweiteren besitzen fünf Metriken der Benutzbarkeit eine Relation zu Umgebungsbedingungen, jeweils vier zu Betriebsdokumenten oder Anforderungen und drei zu Programmierrichtlinien. Eine Metrik der Benutzbarkeit ist laut den Ergebnissen der quantitativen Analyse direkt messbar und weist keine Abhängigkeit zu Eingangsgrößen auf. Die Berechnungsvorschrift des unabhängigen Software Qualitätsmaßes mit ID UAc-2-S und der Bezeichnung SUPPORTED LANGUAGES ADEQUACY besteht aus der Anzahl der von der Software unterstützten Sprachen im Verhältnis zu benötigten Sprachen.

<sup>26</sup> Eigene Darstellung nach Tab. 34



Abb. 22 – Abhängigkeit der Zuverlässigkeit von externen Eingangsgrößen<sup>27</sup>

Die Abb. 22 visualisiert die Abhängigkeit der Parameter der elf Berechnungsvorschriften des Qualitätsmerkmals Zuverlässigkeit zu Eingangsgrößen aus Tab. 5. Die Abhängigkeit mit der stärksten Ausprägung an Abhängigkeiten ist mit sechs identifizierten Metriken die zu Umgebungsbedingungen, gefolgt von vier zu Betriebsdokumenten, zwei zu Anforderungen und einem zum Kontext oder Anwender. Die Programmierrichtlinien weisen keine Abhängigkeit zu Software Qualitätsmaßen des Qualitätsmerkmals Zuverlässigkeit auf. Das Software Qualitätsmaß mit der ID RAV-1-G und der Bezeichnung SYSTEM AVAILABILITY wird berechnet durch die aktuelle Betriebszeit der Software im Verhältnis zur spezifizierten Zeit im Betriebs-handbuch. Durch die in der Definition genutzten Schlagworte *system*, *operation* und *specified* wird der Berechnungsvorschrift eine Abhängigkeit zu den Eingangsgrößen der Umgebungsbedingungen, Anforderung und dem Betrieb unterstellt. Vier Software Qualitätsmaße sind auf Basis der Ergebnisse der quantitativen Analyse direkt messbar und frei von Abhängigkeiten. Die bereits in Kapitel 3.1 beschriebene Metrik mit der ID RAV-2-G und dem Namen MEAN DOWN TIME wird gebildet durch die Zeit in der die Software nicht verfügbar ist dividiert zur Anzahl beobachteter Abstürze. RAV-2-G repräsentiert eines der Software Qualitätsmaße ohne Abhängigkeit zu einer der definierten Eingangsgrößen.

Abb. 23 – Abhängigkeit der Sicherheit von externen Eingangsgrößen<sup>28</sup><sup>27</sup> Eigene Darstellung nach Tab. 34<sup>28</sup> Eigene Darstellung nach Tab. 34



Die Abhängigkeit der elf Software Qualitätsmaße des Qualitätsmerkmals Sicherheit von Eingangsgrößen aus Tab. 5 ist in Abb. 23 dargestellt. Die Eingangsgröße der Anforderung ist mit sieben abhängigen Software Qualitätsmaßen die Gruppe mit der höchsten Anzahl identifizierter Relationen, gefolgt von den drei Abhängigkeiten von Umgebungsbedingungen und einer Abhängigkeit zum Kontext oder Anwender. Das Software Qualitätsmaß mit der ID SCo-1-G und der Bezeichnung ACCESS CONTROLLABILITY besitzt eine Abhängigkeit zur Eingangsgröße Anforderung, welche durch die Verwendung des Stichworts *require* abgeleitet wurde. Die Metrik wird berechnet durch den Wert 1, von dem der Quotient der Anzahl von vertraulichen Daten, die ohne Autorisierungsverfahren zugänglich sind, zur Anzahl von Daten die Zugriffskontrolle benötigen subtrahiert wird. Die mit Autorisierungsmechanismen zu sichernden Daten werden in Anforderungsdokumenten spezifiziert. Die Metrik mit ID SCo-3-S und dem Namen STRENGTH OF CRYPTOGRAPHIC ALGORITHMS weist keinen Konnex zu den definierten Eingangsgrößen auf. Die Parameter der Metrik sind definiert als Verhältnis der Anzahl der genutzten, riskanten kryptografischen Algorithmen zur Summe der genutzten kryptografischen Algorithmen subtrahiert vom Wert 1.

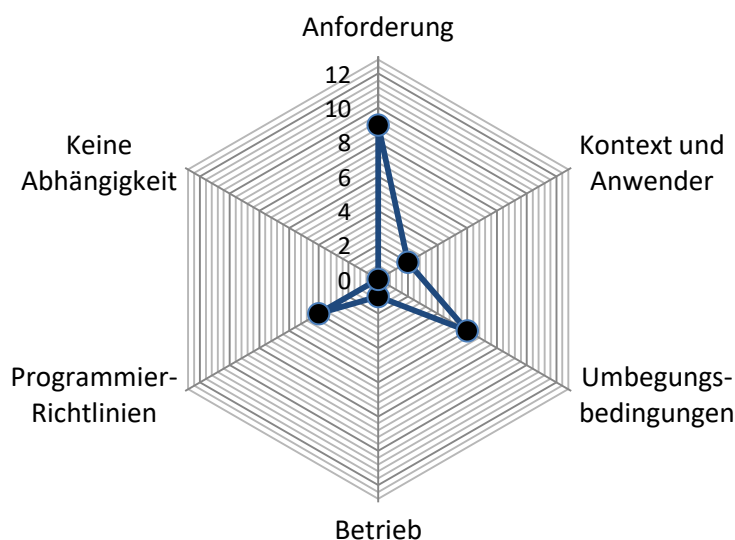


Abb. 24 – Abhängigkeit der Wartbarkeit von externen Eingangsgrößen <sup>29</sup>

Der Zusammenhang zwischen den Metriken des zugehörigen Qualitätsmerkmals Wartbarkeit und den definierten Einflussgrößen ist in Abb. 24 aufgezeigt. Die Klasse der Anforderung ist mit neun Abhängigkeiten zu Software Qualitätsmaßen der Wartbarkeit die mit der größten Ausprägung. Weitere Abhängigkeiten der Software Qualitätsmaße des korrelierenden Qualitätsmerkmals Wartbarkeit bestehen zu den Eingangsgrößen der Umgebungsbedingungen mit sechs, Programmierrichtlinien mit vier, dem Kontext oder Anwender mit zwei und Betriebsdokumenten mit einer identifizierten Relation. Das Software Qualitätsmaß mit der ID MAN-1-G und dem Namen SYSTEM LOG COMPLETENESS wird aus dem Verhältnis der Anzahl der geschriebenen Logfiles zur Anzahl benötigter Protokollierungen einer Software gebildet. Durch die in der Definition verwendeten Schlagworte *system*, *require* und *operation* wird dem Software Qualitätsmaß eine Abhängigkeit zu den Eingangsgrößen Umgebungsbedingungen, Programmierrichtlinien und Anforderungen unterstellt. Durch die quantitative Analyse konnte kein Software Qualitätsmaß des Qualitätsmerkmals Wartbarkeit identifiziert werden, welches direkt messbar ist und keine Abhängigkeit von Eingangsgrößen besitzt.

<sup>29</sup> Eigene Darstellung nach Tab. 34

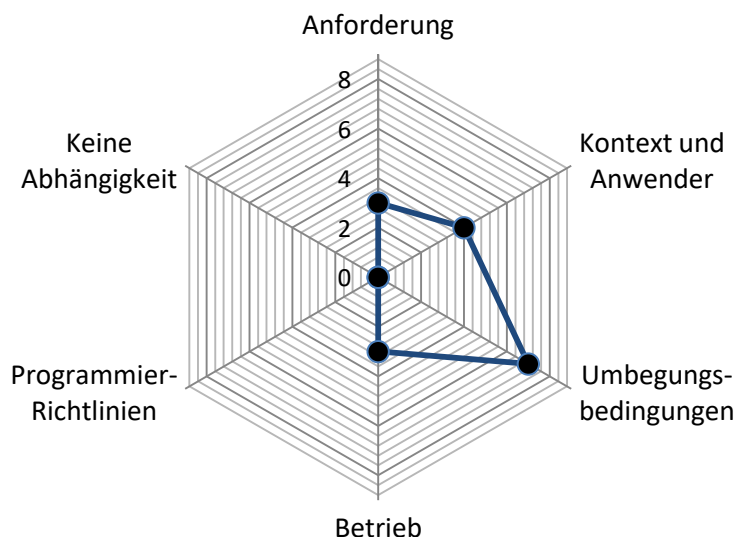


Abb. 25 – Abhängigkeit der Portabilität von externen Eingangsgrößen<sup>30</sup>

Der Abhängigkeiten der Software Qualitätsmaße des Qualitätsmerkmals Portabilität zu definierten Eingangsgrößen ist in Abb. 25 dargestellt. Die Klasse mit der höchsten Anzahl an Abhängigkeiten ist mit sieben identifizierten Relationen die der Eingangsgröße Umgebungsbedingungen. Zusätzlich wurden vier Abhängigkeiten zum Kontext oder Anwender und jeweils drei zu den Eingangsgrößen Betrieb und Anforderung festgestellt. Das Software Qualitätsmaß PRE-1-G mit dem Namen USAGE SIMILARITY wird aus der Anzahl von Nutzungsszenarien, die ohne zusätzliche Schulungen durchführbar sind im Verhältnis zur Anzahl der Funktionen im Substitutionsprodukt gebildet. Die Metrik des Qualitätsmaßes ist aufgrund der Verwendung der Schlagworte *user* und *software* abhängig vom Anwender und den Umgebungsbedingungen. Anhand der quantitativen Analyse konnte kein Software Qualitätsmaß der Portabilität identifiziert werden, welches frei von Abhängigkeiten zu Einflussgrößen und direkt messbar ist. Die Software Qualitätsmaße des System- und Software Produkt Qualitätsmodells wurden auf Basis von fünf abgeleiteten Ansätzen aus der Norm ISO/IEC 25023 klassifiziert. Eine Kategorisierung anhand externer Abhängigkeiten wurde durch eine quantitative Analyse durchgeführt und auf Basis von Qualitätsmerkmalen konkretisiert. Während der Analyse wurde festgestellt, dass die Resultate der quantitativen Analyse eine qualitative Bewertung erfordern, um valide Aussagen über die Abhängigkeiten der Metriken von externen Einflussgrößen zu erstellen. Exemplarisch weist das Software Qualitätsmaß mit der ID UAc-2-S des zugehörigen Qualitätsmerkmals Benutzbarkeit laut quantitativer Analyse keine Relation zu einer Eingangsgröße auf, obwohl die unterstützten Sprachen einer Software in Anforderungen vorgegeben werden. Die qualitative Analyse wird im Rahmen dieser Thesis nicht durchgeführt, da diese den Umfang der Arbeit übersteigt. Die untersuchten Qualitätsmaße lassen sich ebenfalls den in der Literatur beschriebenen Ansätzen zuordnen. T<sub>1</sub> zeigt, dass die betrachteten Software Qualitätsmaße klassifiziert und eingeschränkt bestimmt werden können.

<sup>30</sup> Eigene Darstellung nach Tab. 34

## 4.2 Auswertung der Teilhypothese 2

Die Bewertungsergebnisse der Tauglichkeit von Werkzeugklassen aus Tab. 3 zur Operationalisierung von Qualitätsmerkmalen des System- und Software Produkt Qualitätsmodells der Norm ISO/IEC 25010 ist im Anhang F in Tab. 33 aufgeführt. Im Rahmen der Analyse wurden die Funktionalitäten und integrierten Metriken einzelner Werkzeuge mit den Berechnungsvorschriften der Software Qualitätsmaße aus Tab. 29 im Anhang B verglichen und zensiert. Die Grundlage der Bewertung bildet die Analyse der Funktionalitäten aus den Produktbeschreibungen und Dokumentationen der Werkzeuge, welche die Werkzeugtypen Markov Modellierungswerkzeuge, strukturorientierte Testwerkzeuge, funktionsorientierte Testwerkzeuge, Regressionswerkzeuge, Leistungs- und Stresstestwerkzeuge, Stilanalysatoren, Messwerkzeuge, formale Verifikationswerkzeuge, Slicingwerkzeuge, FMECA-Werkzeuge und Fehlerbaumanalyse Werkzeuge repräsentieren. Datenflussanomalienanalyse-Werkzeuge und Werkzeuge zur Erzeugung von Grafiken und Tabellen sind nicht Bestandteil der Untersuchung, da kein Werkzeug des Werkzeugtyps identifiziert werden konnte oder diese in anderen Werkzeugtypen integriert sind. Als Bewertungsverfahren für die selektierten Werkzeuge wurden die Werte 1 für geeignet, 0,5 für partiell geeignet, sowie 0 für ungeeignet oder falls keine Aussage zur Tauglichkeit der integrierten Funktionen des Werkzeugs zur Bestimmung des Software Qualitätsmaßes getroffen werden konnte, zugeordnet. Der Wert 0,5 wurde ebenfalls vergeben, wenn ein Werkzeug keine direkte Messfunktion der definierten Metrik bereitstellt, aber die Werte der Parameter der Berechnungsvorschrift aus den Daten der Software abgeleitet werden können oder die Bestimmung der Software Qualitätsmaßausprägung erst durch Kopplung mit anderen Werkzeugtypen ermöglicht wird.

### 4.2.1 Dynamische Testwerkzeuge

Dynamische Testwerkzeuge analysieren die Eigenschaften und das Verhalten von Software zur Laufzeit. Sie liefern Informationen darüber, wie sich der Betrieb von Software auf die vorhandenen Ressourcen auswirkt, Algorithmen ablaufen oder in welchem Umfang der Quellcode durch Tests abgedeckt ist (vgl. [27], S. 24 f.).

#### 4.2.1.1 Strukturorientierte Testwerkzeuge

Das Open Source Werkzeug SonarQube ist eine vom Unternehmen SonarSource entwickelte Software für die Betriebssysteme Microsoft Windows und Linux zur Analyse von Quellcode auf Basis der Qualitätsbereiche Komplexität, Sicherheit, Wartbarkeit, Modultests, Programmierkonventionen, Kommentare und Duplikate (vgl. [28]). Eine Liste aller unterstützten Programmiersprachen ist in Tab. 6 aufgeführt.

Programmiersprachen			
ABAP	C	C++	Objective-C
COBOL	C#	Flex	Java
JavaScript	PHP	VB.NET	VB6
Python	RPG	Swift	PL/SQL
Web	XML	PL/I	

Tab. 6 – Unterstützte Programmiersprachen des Werkzeugs SonarQube<sup>31</sup>

Die Funktionalitäten von SonarQube Version 6.4 (vgl. [30]) lassen sich durch die Integration von zusätzlichen Analysewerkzeugen erweitern (vgl. [31]). Die Ausgabe der Messergebnisse erfolgt in der Komponente Code Viewer (vgl. [32]). Die vom Werkzeug bereitgestellten Metriken werden nicht im Detail erläutert sondern auf die Dokumentation verwiesen, falls diese

<sup>31</sup> Eigene Darstellung nach ([29])

im jeweiligen Kontext relevant sind. Eine Übersicht, der für die Analyse erforderlichen, im Werkzeug integrierten Metriken und deren Definitionen ist in Tab. 36 im Anhang I aufgeführt.



Abb. 26 – Codeüberdeckungsvisualisierung mit dem Werkzeug SonarQube<sup>32</sup>

Aufgrund der Funktionalität zur Messung der Code Überdeckung (vgl. Abb. 26) als Indikator für die Abdeckung von Quellcode durch Modultests repräsentiert SonarQube ein Werkzeug der Klasse strukturorientierte Testwerkzeuge. Die Berechnungsvorschrift von SonarQube zur Bestimmung der Code Überdeckung ist in Formel (4.1) dargestellt.

$$coverage = \frac{(CT + CF + LC)}{(2 * B + EL)} \quad (4.1)$$

Die Parameter CT und CF sind boolsche Ausdrücke die mindestens einmal die Werte true und false annehmen, um alle Ablaufmöglichkeiten im Programm abzudecken. LC gibt die Anzahl der relevanten Code Zeilen subtrahiert von den unberücksichtigten Zeilen an. B definiert die Anzahl der Bedingungen im Quellcode und EL die der ausführbaren Zeilen. Während der Analyse der Metriken in Tab. 36 wurde festgestellt, dass SonarQube Funktionalitäten weiterer Werkzeugtypen aufweist und nicht nur den strukturorientierten Werkzeugtyp zugeordnet werden kann. SonarQube besitzt zusätzlich Funktionalitäten der Werkzeugtypen Messwerkzeuge und Stilanalysatoren. Das Software Qualitätsmaß aus der Norm ISO/IEC 25023 mit der ID RMa-1-G und der Bezeichnung FAULT CORRECTION ist definiert als die Anzahl von korrigierten Fehlern mit einer Abhängigkeit vom Qualitätsmerkmal Zuverlässigkeit im Verhältnis zur Gesamtzahl an Zuverlässigkeitsfehlern in den Phasen Design, Entwicklung und Test des Software Lebenszyklus. SonarQube identifiziert durch die Metriken *Programmfehler* und *Neue Programmfehler* zuverlässigkeitsrelevante Probleme im Quelltext und stellt Detailinformationen durch die Metriken *Zuverlässigkeitsbewertung*, *Zuverlässigkeitsspezifische Sanierungskosten* und *Zuverlässigkeitsspezifische Sanierungskosten neuen Codes* zur Verfügung (vgl. Tab. 36). Die Zuordnung zu spezifischen Phasen des Softwarelebenszyklus ist in SonarQube nicht möglich. Die Bewertung der Messbarkeit des Software Qualitätsmaßes RMa-1-G durch SonarQube wurde mit dem Wert 0,5 dokumentiert, da die Anzahl der Fehler mit einem Bezug zum Qualitätsmerkmal Zuverlässigkeit einer Software aus den Informationen im Werkzeug abgeleitet werden kann, aber dieses keine Funktion zur direkten Messwertbestimmung besitzt. Das in Kapitel 4.1 beschriebene Software Qualitätsmaß FCp-1-G berechnet sich aus dem Verhältnis von fehlenden Funktionen einer Software zu spezifizierten Funktionen. Bei FCp-1-G handelt es sich um eine Überdeckungsmetrik, jedoch ist die Formel (4.1) zur Abdeckung einer Software durch Modultests und die weiteren verfügbaren Überdeckungsmetriken von SonarQube nicht auf die Messung von Funktionen, wie sie FCp-

<sup>32</sup> Darstellung aus ([33])

1-G erfordert, ausgelegt. Der Eignung von SonarQube zur Messung des Software Qualitätsmaßes FCp-1-G wurde der Wert 0 zugewiesen. Die 84 weiteren Software Qualitätsmaße des System- und Software Produkt Qualitätsmodells aus ISO/IEC 25010 wurden nach dem Schema wie RMa-1-G und FCp-1-G bewertet (vgl. Tab. 32). Im Rahmen der Untersuchung konnte kein Software Qualitätsmaß des untersuchten Qualitätsmodells identifiziert werden, welches direkt von SonarQube bestimmt werden kann.

#### 4.2.1.2 Funktionsorientierte Testwerkzeuge

Application Lifecycle Management (ALM) ist ein proprietäres Werkzeug des Unternehmens HP zur Verwaltung des Anwendungslebenszyklus von der Anforderungsplanung bis zur Bereitstellung der Implementierung (vgl. [34], S. 22). Eine Übersicht des Funktionsumfangs von HP ALM der Version 12.00 ist in Tab. 7 aufgeführt.

<b>Funktionalitäten</b>	<b>Beschreibung</b>
Management-funktionen	Dashboards, Erstellung von projektübergreifenden Berichten, Projektplanung und Überwachung, Excel Schnittstelle
Verwaltung von Anforderungen	Dokumentenverwaltung, Anforderungsmanagement, Anforderungsüberwachung, Versionskontrolle, Business Prozessmodelle, Baselineing, risikobasiertes Qualitätsmanagement, Wiederverwendung/Teilen von Anforderungen auf Basis des Quellcodes oder Builds, Bereitstellung und Verwaltung von Anforderungstemplates, Shared Customization
Verwaltung von Tests	Testplanverwaltung, Test Lab Verwaltung, Test Ressourcenmanagement, Test Konfiguration, Versionskontrolle, Baselineing, Application under test (AUT) Topology, Continuous Delivery Automation (CDA) Integration, Wiederverwendung/Teilen von Tests, Testplanung basierend auf Build Health oder Risiko
Verwaltung von Defects	Defect Verwaltung auf Basis von Quellcode und Builds, Defectverwaltung, Workflow Anpassung, Teilen von Defects, Unterstützung mobiler Applikationen, Defectverwaltung durch Web Client
Erweiterungen	Supply Chain Management (SCM) Integration, Build System Integration, Statusverfolgung von Modultests, Testabdeckung, Bewertung der Auswirkung von Quellcodeänderungen, Change Impact Bericht, Build Success Report, IDE Integration

Tab. 7 – Funktionalitäten des Werkzeugs ALM<sup>33</sup>

Durch die von ALM bereitgestellten Funktionen zur Erzeugung, Planung, Ausführung, Dokumentation von Tests und deren Ergebnissen ist das Werkzeug der Gruppe der funktionsorientierten Testwerkzeuge zugeordnet. Die Struktur des Graphical User Interface (GUI) ist in Abb. 27 visualisiert. ALM ist primär ein Werkzeug zur Verwaltung von Daten mit Bezug zu einer Software und dadurch keinen speziellen Programmiersprachen zugeordnet.

<sup>33</sup> Eigene Darstellung nach ([35], S. 2 ff.)

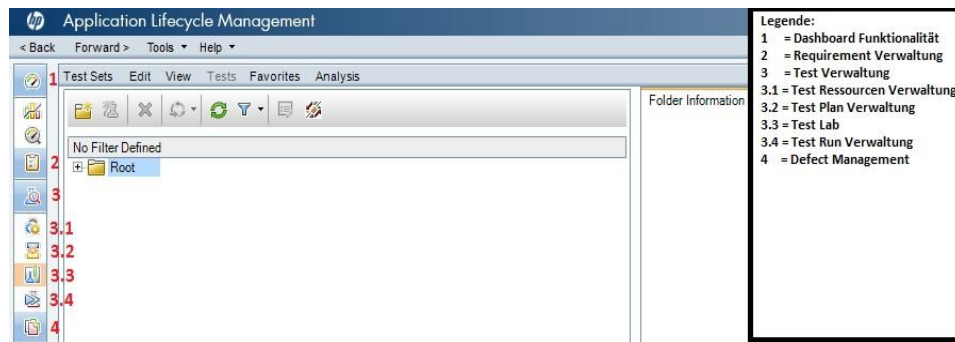
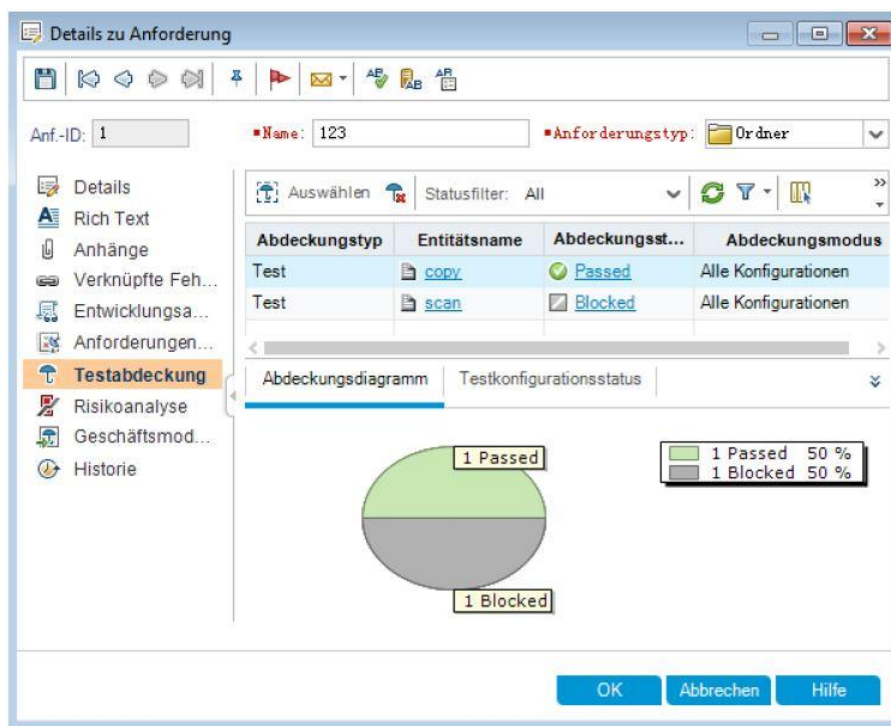
Abb. 27 – ALM GUI Struktur<sup>34</sup>

Abb. 28 zeigt die Testabdeckung einer Anforderung im Werkzeug HP ALM und die zugehörigen spezifischen Testfälle, die den Status *Open*, *Failed*, *Blocked*, *not Completed* und *passed* annehmen können.

Abb. 28 – Testabdeckung durch ALM<sup>35</sup>

Die Bewertung der Eignung von ALM als Repräsentant der Kategorie der funktionsorientierten Testwerkzeuge zur Messung der Software Qualitätsmaße des System- und Software Produkt Qualitätsmodells erfolgt analog zur Untersuchung der strukturorientierten Testwerkzeuge (vgl. Kapitel 4.2.1.1). Bei der Analyse der Funktionalitäten von ALM und der Möglichkeit andere Werkzeuge in ALM zu integrieren wurde festgestellt, dass das Werkzeug ebenfalls Charakteristiken der Werkzeugtypen der Werkzeuge zur Erzeugung von Grafiken und Tabellen, Regressionswerkzeuge und Leistungs- und Stresstestwerkzeuge aufweist und nicht nur dem Werkzeugtyp der funktionsorientierten Werkzeuge zuzuordnen ist. Die Berechnungsvorschrift des in Kapitel 4.1 beschriebenen Software Qualitätsmaßes der FUNCTIONAL COVERAGE mit ID FCp-1-G ist mittels der Dashboardfunktion in ALM grafisch abbildbar, da eine Funktion ein Teil einer Anforderung ist (vgl. Abb. 28). Der Werkzeugtyp erhält die Bewertung 1 zur Bestimmung der Parameter der Metrik der Funktionsabdeckung. Die Metrik des Software

<sup>34</sup> Eigene Darstellung

<sup>35</sup> Abb. aus ([34], S. 644)



Qualitätsmaßes der Performanz mit der ID PRu-1-G und der Bezeichnung MEAN PROCESSOR UTILIZATION berechnet sich aus der Summe der Verhältnisse der aktuellen Prozessorauslastung während der Ausführung von Aufgaben zur benötigten Zeit, dividiert durch Anzahl der beobachteten Szenarien. In der Dokumentation von HP ALM wird zwar darauf verwiesen, dass durch Kopplung von ALM mit weiteren Testwerkzeugen Last- und Stresstests durchgeführt werden können (vgl. [34], S. 600), allerdings werden diese Funktionen nicht von ALM exklusiv bereitgestellt. Die Bewertung der Eignung der Messung von dem Software Qualitätsmaß erfolgt mit dem Wert 0,5. Das in Kapitel 4.1 beschriebene Software Qualitätsmaß des SELF-EXPLANATORY USER INTERFACE mit ID ULe-4-S kann nicht durch ALM gemessen werden, da weder die für einen Nutzer verständlichen Informationselemente, noch die Informationselemente, die ein Nutzer benötigt um eine Aufgabe bei erstmaliger Durchführung erfolgreich abzuschließen, durch ALM bestimmt werden können. Die Definition eines verständlichen Informationselementes und die erfolgreiche Ausführung unterliegen den kognitiven Fähigkeiten des Anwenders. Das Werkzeug ALM erhält für die Tauglichkeit der Bestimmung des Software Qualitätsmaßes die Bewertung 0.

#### 4.2.1.3 Regressionswerkzeuge

Das kommerzielle Werkzeug Silk Test des Unternehmens Micro Focus dient zur Automatisierung und Beschleunigung von funktionalen Tests für klassische Legacy Anwendungen und nativen, webbasierten oder hybriden Mobilanwendungen (vgl. [36]). Silk Test bietet die Möglichkeit der Kopplung mit Build Werkzeugen, Quellcodeverwaltungs- und Testmanagementsystemen. Die unterstützten Betriebssysteme von Silk Test Version 17.5 sind Microsoft Windows, iPhone OS (iOS) und Android (vgl. [37]). Das Werkzeug besteht aus den Komponenten Silk Test Client und Silk Test Agent. Der Silk Test Workbench host ist eine IDE zur Entwicklung, Editierung, Kompilierung, Ausführung und zum Debuggen von visuellen Tests und Skripten (vgl. [38]). Der Silk Test Agent interagiert mit der Benutzerschnittstelle der Applikation und übersetzt VB.NET Kommandos für Tests durch Skripte in GUI spezifische Befehle. Die von Silk Test unterstützten Programmiersprachen sind in Tab. 8 aufgelistet.

Programmierersprachen			
AJAX	JAVA	.NET	HTML5
DHTML	Flex		

Tab. 8 – Unterstützte Programmiersprachen des Werkzeugs Silk Test<sup>36</sup>

Eine Übersicht der Funktionalitäten des Werkzeugs Silk Test mit der Version 17.5 befindet sich in Tab. 9.

Funktionalitäten			
Keyword-driven Tests	Visuelle Tests	Cross-browser Tests	Mobile testing
Debugging Tests	Testweiterentwicklung		

Tab. 9 – Funktionalitäten des Werkzeugs Silk Test<sup>37</sup>

Durch die Möglichkeit der Integration von Silk Test als Plugin in die genutzte IDE können Tests in der eigenen Entwicklungsumgebung generiert werden. Das Plugin Silk4J ist für die Java IDE Eclipse und Silk4NET für .NET IDE Visual Studio verfügbar (vgl. [39], S. 2). In Abb. 29 ist ein Protokoll eines erfolgreichen automatisierten visuellen Testdurchlaufs mit dem Werkzeug Silk Test durch den Zugriff und den Login auf die Website der HFTL mit dem Webbrowser Mozilla Firefox aufgeführt.

<sup>36</sup> Eigene Darstellung nach ([39], S. 2)

<sup>37</sup> Eigene Darstellung nach ([38])



Summary	Passed	Failed	Flags	Details (8 steps)
Overall result for run 2 :				 Passed
Browser:	 Firefox 53			
Reason for abort:	Not applicable			
Latest run number:	2			
Recent runs:	2   1			
Visual test executed:	Login			
Visual test description:	Login on HFT Leipzig Homepage			
Result description:				
Scripts (number of times each ran):	Login(1)			
Verifications passed:	0 / 0			
Verifications failed:	0 / 0			
Flags:	0			
Start time:	01.06.2017 20:27:49			
End time:	01.06.2017 20:28:00			
Total time:	11 s			
Steps run:	8			
Playback setting:	System Defaults			

Abb. 29 – Silk Test Ergebnis eines visuellen Tests<sup>38</sup>

Die skriptbasierte Definition eines Schlüsselwortes im Rahmen des Test Designs mit .NET, für das automatisierte Keyword-driven testing, ist im Codeausschnitt 1 dargestellt.

```
Public Module Main
    Dim _desktop As Desktop = Agent.Desktop

    Public Sub Main()
        ' method implementation
    End Sub

    'VB .NET code
    <Keyword("Login")>
    Public Sub Login()
        ' method implementation
    End Sub
End Module
```

Codeausschnitt 1 – Keyword-driven testing mit dem Werkzeug Silk Test<sup>39</sup>

Silk Test ist durch die Funktionen der Automatisierung von Tests prädestiniert als Repräsentant des Werkzeugtyps Regressionswerkzeuge. Das Werkzeug weist zusätzlich Merkmale des Werkzeugtyps funktionsorientierte Werkzeuge aufgrund der Dokumentationsfunktion von Testdaten auf. Die Analyse der Eignung von Regressionswerkzeugen erfolgt analog zu den bisher durchgeführten Untersuchungen. Das Software Qualitätsmaß mit der ID FCr-1-G und dem Namen FUNCTIONAL CORRECTNESS ergibt sich aus der Anzahl von fehlerhaften Funktionen im Verhältnis zu untersuchten Funktionen. Beide Parameter der Metrik werden in der Berichtsfunktion des Werkzeugs Silk Test abgebildet und sind direkt ablesbar. Die Tauglichkeit von Silk Test zur Erfassung des Software Qualitätsmaßes wird mit dem Wert 1 dokumentiert. Das Software Qualitätsmaß mit der ID MTe-3-S und der Bezeichnung TEST RESTARTABILITY des System- und Software Produkt Qualitätsmodells ist definiert als die Anzahl der Testdurchläufe die unterbrochen, pausiert und schrittweise ausgeführt werden können, im Verhältnis zur Anzahl an Testdurchläufen die pausierbar sind. Beide Werte der Para-

<sup>38</sup> Eigene Darstellung

<sup>39</sup> Eigene Darstellung



meter der Metrik können aus den Daten im Werkzeug Silk Test für visuelle Tests, Keyword-driven Tests, Debugging-Tests und Cross-browser Tests erhoben werden, jedoch ist eine direkte Bestimmung des Messwertes nicht möglich. Für die Eignung des Werkzeugs zur Messung wird der Wert 0,5 vergeben. Das Software Qualitätsmaß des Qualitätsmerkmals Benutzbarkeit mit der ID ULe-2-S und der Bezeichnung ENTRY FIELDS DEFAULTS wird aus dem Verhältnis der Anzahl der Eingabefelder mit Default Werten zur Anzahl der Eingabefelder mit möglichen Default Werten gebildet. Beide Parameter der Metrik können mit den Funktionen von Silk Test nicht ermittelt werden. Der Eignung des Werkzeugs zur Messung des Software Qualitätsmaßes wird der Wert 0 zugewiesen.

#### 4.2.1.4 Leistungs- und Stresstestwerkzeuge

JProfiler ist ein proprietäres Werkzeug des Unternehmens EJ Technologies (vgl. [40], S. 9) zur Erstellung von Leistungsprofilen von Komponenten beim Betrieb von Java Applikationen für die Plattformen Windows, Linux, MacOS, Solaris, AIX, FreeBSD und HP-UX (vgl. [40], S. 116 f.). Die unterstützten Programmiersprachen sind in Tab. 10 aufgeführt.

Programmierersprachen	
Java	

Tab. 10 – Unterstützte Programmiersprachen des Werkzeugs JProfiler<sup>40</sup>

Die analysierte Version des Werkzeugs entspricht der Revision 10.0.1. Das Werkzeug kann als eigenständige Applikation oder als Plugin in den IDEs Eclipse, IntelliJ IDEA, NetBeans und Oracle JDeveloper betrieben werden (vgl. [41]). Die Funktionalitäten des JProfilers sind in Tab. 11 aufgeführt.

Funktionalitäten			
Memory Profiling	Heap Walker	CPU Profiling	Thread Profiling
Monitor Profiling	Virtual Machine (VM) Telemetry	Aufrufgraphen Analyse	

Tab. 11 – Funktionalitäten des Werkzeugs JProfiler<sup>41</sup>

Zur Generierung von Profilen wird das Sampling genutzt (vgl. [40], S. 91 f.).

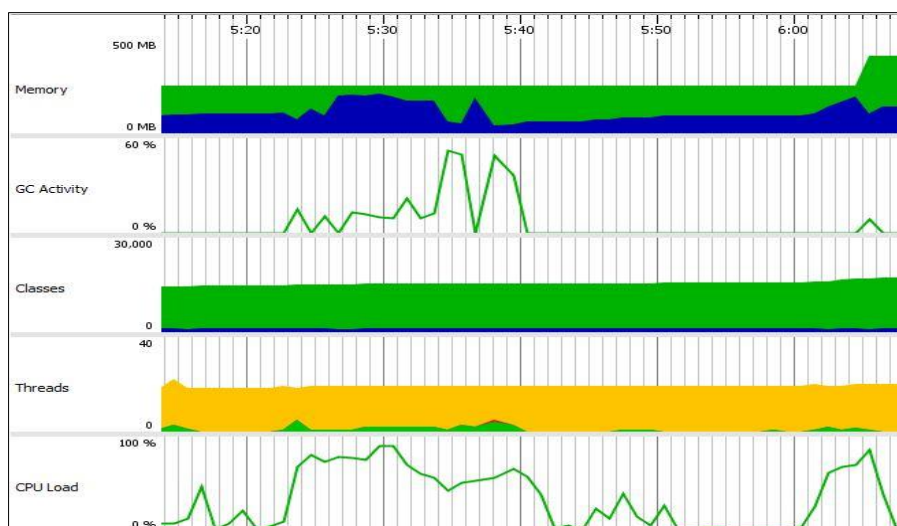


Abb. 30 – Telemetrie Dashboard einer JVM mit dem Werkzeug JProfiler<sup>42</sup>

<sup>40</sup> Eigene Darstellung nach ([40], S. 10 ff.)

<sup>41</sup> Eigene Darstellung nach ([40], S. 16 ff.)

<sup>42</sup> Eigene Darstellung

Abb. 30 zeigt das Telemetrie Dashboard des JProfilers, welches Messdaten autonomer Bereiche aufbereitet. Das Werkzeug aggregiert Informationen über Speicher-, Thread- und CPU Auslastung während der Ausführung einer Java Applikation. Die Bereiche Threads und Classes repräsentieren die Anzahl der Threads und Klassen der Applikation gruppiert nach dem Status in dem diese sich befinden. GC Activity beschreibt die Aktivitäten des garbage collectors für den Zeitraum des Profilings.

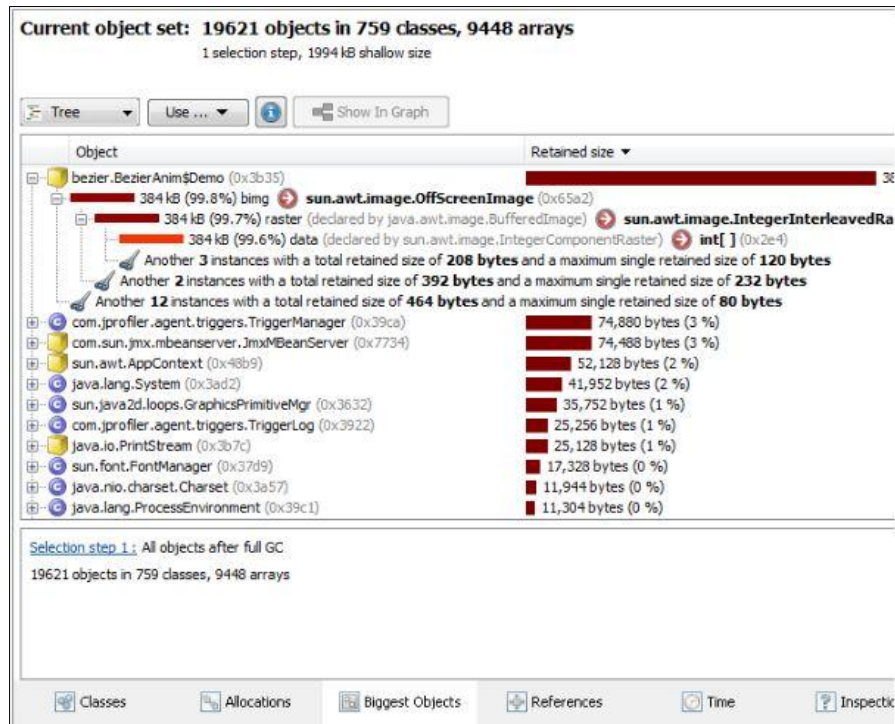


Abb. 31 – Method Call Recording mit dem Werkzeug JProfiler<sup>43</sup>

Eine Übersicht mittels einer Baumstruktur zur Visualisierung der erzeugten Speicherlast von Paketen und Klassen ist in Abb. 31 dargestellt. Die dargestellten Objekte sind der Größe absteigend sortiert. Große Objekte sind ein Indikator für bestehende Memory Leaks, die auf Grundlage der aufbereiteten Detailinformationen näher analysiert werden können. JProfiler weist zusätzlich zu den Eigenschaften eines Leistungs- und Stresstest Werkzeugs Merkmale von statischen Analysewerkzeugen des Typs Messwerkzeuge auf, da Informationen über die Anzahl von Klassen, Objekten und Arrays ermittelt werden können (vgl. Abb. 31). Als Repräsentant des Werkzeugtyps Leistungs- und Stresstestwerkzeuge eignet sich der JProfiler, aufgrund der fehlenden Funktionen Überlastszenarien für einen Testling zu simulieren, nur partiell. Das Werkzeug ist folglich als Leistungs-, aber nicht als Stresstestwerkzeug einzuordnen. Dieses trifft ebenfalls für andere untersuchte Werkzeuge wie Heavy Load der JAM Software GmbH und LoadGen des Unternehmens LoadGen BV zu. Aufgrund des Mangels an identifizierten Alternativen wurde die Bewertung der Tauglichkeit der Messung von Software Qualitätsmaßen des System- und Software Produkt Qualitätsmodells durch Leistungs- und Stresstestwerkzeuge mit dem Werkzeug JProfiler durchgeführt. Das Software Qualitätsmaß des Qualitätsmerkmals Performanz mit der ID PRu-2-G und der Bezeichnung MEAN MEMORY UTILIZATION wird durch die Summe der Quotienten von genutzten zu verfügbaren Speicher im Verhältnis zur Anzahl der Stichproben berechnet. Da das Werkzeug JProfiler keine Stichproben ermittelt, sondern das Verhalten des Speichers über einen Zeitraum beschreibt und folglich der Messwert der Metrik nicht direkt mit dem Werkzeug bestimmt werden kann, wurde die Eignung mit dem Wert 0,5 dokumentiert. Das bereits in Kapitel 4.2.1.1 verwendete

<sup>43</sup> Abb. aus ([40], S. 36)

Software Qualitätsmaß der Funktionsabdeckung FCp-1-G ist nicht mit dem JProfiler messbar, da spezifikationsabhängige Parameter vom Werkzeug nicht ausgewertet werden können. Für die Bewertung der Tauglichkeit des Werkzeugs zur Bestimmung der Parameter der Berechnungsvorschrift von FCp-1-G wurde der Wert 0 vergeben. Im Rahmen der Untersuchung wurde kein Software Qualitätsmaß des System- und Software Produkt Qualitätsmodells identifiziert, welches vollständig durch das Werkzeug JProfiler bestimmt werden konnte.

#### 4.2.2 Statische Analysewerkzeuge

Statische Analysewerkzeuge werden zur Extrahierung von Informationen aus vorliegenden Quellcode genutzt, ohne diesen auszuführen oder einen formalen Korrektheitsbeweis zu erbringen. Zur Untersuchung der Struktur von Quelltext werden von statischen Analysewerkzeugen lexikalische-, syntaktische- und semantische Analysen durchgeführt, welches der Arbeitsweise von Compilern entspricht (vgl. [27], S. 2 f.).

##### 4.2.2.1 Messwerkzeuge

Security Fortify on Demand ist ein Werkzeug des Unternehmens HP zur Identifikation von sicherheitsrelevanten Schwachstellen im Quellcode von Softwareprodukten durch Anwendung integrierter statischer und dynamischer Analysefunktionen (vgl. [42], S. 3 ff.). Das Werkzeug wird als Security as a Service in Form einer plattformunabhängigen Webanwendung bereitgestellt. Die unterstützten Programmiersprachen sind in Tab. 12 aufgeführt.

Programmierersprachen			
ABAP	ActionScript/MXML	ASP.NET	Apex
VB.NET	C	C++	ClassicASP
C#	ColdFusion CFML	XML	HTML
COBOL	JavaScript/AJAX	VBScript	JSP
Java	PHP	PL/SQL	Python
Objective-C/C++	Ruby	Swift	Visual Basic
T-SQL			

Tab. 12 – Unterstützte Programmiersprachen durch Security Fortify on Demand <sup>44</sup>

Die dynamische Analyse zur Identifikation von Sicherheitsmängeln ist für Web- und Mobile Applikationen mittels automatisierter Simulation von Attacken und Ausnutzung bekannter Schwachstellen durchführbar (vgl. [42], S. 4 f.). Die Funktionalitäten der statischen und dynamischen Analyse erfordern den Upload der Software oder die Angabe des Uniform Resource Locators (URL) über die die Applikation erreichbar ist. Security Fortify on Demand bewertet Software anhand der Taxonomie von Software Sicherheitsmängeln (vgl. [43]; [44]).

<sup>44</sup> Eigene Darstellung nach ([42], S. 3)

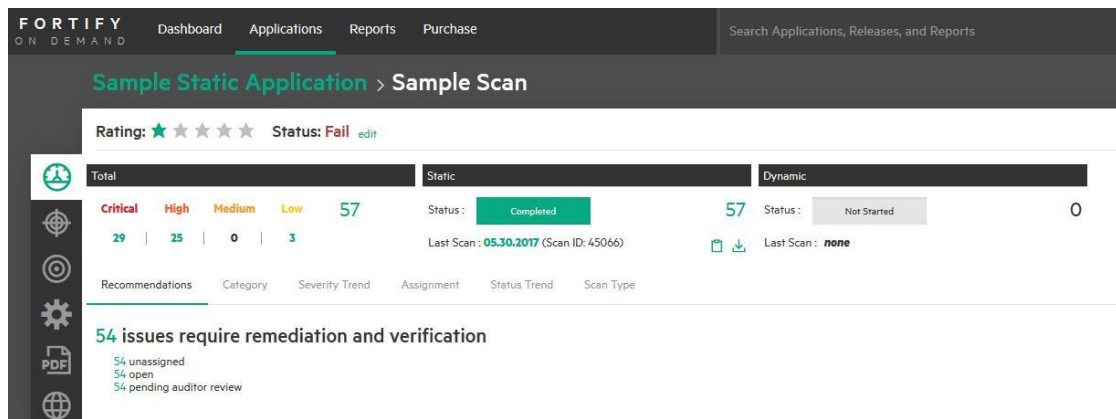
Abb. 32 – Dashboard des Security Fortify on Demand Werkzeugs<sup>45</sup>

Abb. 32 zeigt die Messergebnisse einer Webapplikation im Dashboard des Werkzeugs Security Fortify on Demand. Die Bewertung von Critical, High, Medium und Low ergibt sich aus der Wahrscheinlichkeit der Ausnutzung der identifizierten Schwachstellen und dem potenziellen Schaden durch einen Angriff.

Rating	Category	Test Type	
Critical	Cross-Site Scripting: DOM	Static	4
Critical	Cross-Site Scripting: Reflected	Static	8
Critical	Dangerous File Inclusion	Static	2
Critical	Path Manipulation	Static	2
Critical	Privacy Violation	Static	5
Critical	SQL Injection: Hibernate	Static	4
Critical	SQL Injection	Static	4
High	Open Redirect	Static	1
High	Password Management: Password in Configuration File	Static	1
High	Portability Flaw: File Separator	Static	1
High	Unreleased Resource: Database	Static	13
High	Unreleased Resource: Sockets	Static	3
High	Unreleased Resource: Streams	Static	6
Low	Cookie Security: Cookie not Sent Over SSL	Static	3

Abb. 33 – Messkategorien in Security Fortify on Demand<sup>46</sup>

Eine Auflistung aller verwendeten Metriken des Werkzeugs Security Fortify on Demand ist in der Dokumentation nicht enthalten. Aus diesem Grund führte eine Gegenüberstellung von Funktionen und Software Qualitätsmaßen der ISO/IEC 25023 zu keinem Ergebnis. Als Alternative wurde das Messwerkzeug Code Inspector der HFT Leipzig untersucht. Das Werkzeug unterstützt die Betriebssysteme Microsoft Windows, Linux und OS X. Der CodeInspector dient zur statischen Analyse von Quellcode anhand vorgegebener Qualitätskriterien. Die unterstützten Programmiersprachen sind in Tab. 13 aufgelistet.

Programmiersprachen	
Java	C

Tab. 13 – Unterstützte Programmiersprachen des Werkzeugs CodeInspectors<sup>47</sup>

Eine Übersicht der Funktionalitäten des CodeInspectors gruppiert nach Programmiersprache und Klassifikation des Schweregrads ist in Tab. 14 aufgeführt.

<sup>45</sup> Eigene Darstellung<sup>46</sup> Eigene Darstellung<sup>47</sup> Eigene Darstellung nach ([45], S. 5)

Programmiersprache	Kategorie	Funktionalitäten
Java	Minor	Prüfung der Einhaltung von JAVA-Konventionen durch Methoden, Klassen und Pakete. Prüfung Länge der Variablennamen.
Java	Major	Keine Nutzung des Default Packages und des Symbols * im Namen der Importdeklaration. Prüfung Methoden- und Klassenlänge und mindestens eine Typdeklaration pro Datei. Prüfung, ob Klassenname Dateinamen entspricht.
Java	Critical	Prüfung der Anzahl von Übergabeparametern an Funktionen, Variablendeklarationen von Klassen und Public Variablen in Projekten.
C	Minor	Prüfung der Einhaltung von C-Konventionen durch globale Variablen, globaler STRUCT, globaler Konstante, Funktionsnamen und Variablennamen.
C	Major	Prüfung Anzahl Funktionen, doppelte Definition von Includes, Länge der Quelltextdatei und Funktionen.
C	Critical	Prüfung Anzahl der Übergabeparameter an Funktion und globaler Variablen.

Tab. 14 – Funktionalitäten des Werkzeugs CodeInspector<sup>48</sup>

Das Werkzeug CodeInspector analysiert Quellcode ohne diesen auszuführen, extrahiert Informationen aus diesem und bereitet die Resultate der Messung visuell auf. Das Werkzeug ist aufgrund dieser Merkmale dem Werkzeugtyp Messwerkzeuge, Stilanalysatoren und den Werkzeugen zur Erzeugung von Grafiken und Tabellen zuzuordnen. Die Ergebnisse einer Messung werden als Datei abgelegt und können im Webbrowser dargestellt werden (vgl. [45], S. 6). Die Dashboard Funktion des CodeInspectors stellt Übersichten über die gemessenen Eigenschaften des Quellcodes bereit und ordnet die identifizierten Probleme den Fehlerklassen Minor, Major und Critical zu (vgl. Abb. 34).

Abb. 34 – Dashboard des Werkzeugs CodeInspector<sup>49</sup>

Die Informationen im Dashboard des CodeInspectors können in den Ansichten Packages und Klassen weiter verfeinert werden (vgl. [45], S. 7 ff.). Das Software Qualitätsmaß des Quali-

<sup>48</sup> Eigene Darstellung nach ([45], S. 10 ff.)

<sup>49</sup> Eigene Darstellung

tätsmerkmals Portabilität mit der Bezeichnung PRODUCT QUALITY EQUIVALENCE und der ID PRe-2-S wird aus dem Quotienten der Anzahl an Maße eines Substitutionsproduktes, welche äquivalente oder bessere Messergebnisse als die der ursprünglichen Software erzielen, zur Anzahl der relevanten Maße des neuen Software Produktes gebildet. Beide Parameter der Metrik können aus den Daten im CodeInspector manuell erhoben werden. Die relevanten Software Qualitätsmaße des Substitutionsproduktes werden vom Werkzeug Code Inspector durch den Funktionsumfang vorgegeben. Eine Erweiterung der Funktionen des Werkzeugs durch zusätzliche Metriken ist nicht möglich. Der Eignung des CodeInspectors zur Bestimmung des Software Qualitätsmaßes PRe-2-S wurde der Wert 0,5 zugewiesen, da die Werte der Parameter der Berechnungsvorschrift lediglich aus den Daten im CodeInspector ableitbar sind. Das Software Qualitätsmaß des Qualitätsmerkmals Portabilität mit der ID PIn-1-G und dem Namen INSTALLATION TIME EFFICIENCY berechnet sich aus der Summe der Verhältnisse von benötigter zu erwarteter Installationszeit dividiert durch die Anzahl beobachteter Installationsszenarien. Die Parameter der Metrik können mit dem Werkzeug CodeInspector nicht ausgewertet werden. Die Tauglichkeit des CodeInspectors zur Messung des Software Qualitätsmaßes PIn-1-G wurde mit dem Wert 0 dokumentiert. Während der Analyse der Software Qualitätsmaße wurde im System- und Software Produkt Qualitätsmodell kein Software Qualitätsmaß identifiziert, dessen Ausprägung direkt mit dem CodeInspector bestimmt werden kann.

#### 4.2.2.2 Stilanalysatoren

Checkstyle ist ein Open Source Werkzeug zur automatisierten Überprüfung von Software auf Grundlage vorgegebener Programmierstandards in Form von Methoden-, Klassendesign und Quellcode Formatierungsvorgaben (vgl. [46]).

Programmierersprachen	
Java	

Tab. 15 – Unterstützte Programmiersprachen des Werkzeugs Checkstyle<sup>50</sup>

Die untersuchten Funktionalitäten des Werkzeugs Checkstyle entsprechen der Version 7.8. Das Werkzeug kann in einer JVM als Apache Ant Task oder als Plugin in einer integrierten Entwicklungsumgebung genutzt werden (vgl. [48]). Checkstyle stellt standardmäßig 152 verschiedene Prüfungen von Programmierrichtlinien für Quelltext bereit. Die in der Dokumentation beschriebenen Code Konventionen werden in 14 verschiedene Kategorien eingeteilt (vgl. Tab. 35). Für detaillierte Beschreibungen einzelner Konventionen wird auf die Checkstyle Dokumentation verwiesen. Durch die syntaktische Untersuchung von Quellcode auf Grundlage von Programmierkonventionen ist Checkstyle dem Werkzeugtyp der Stilanalysatoren zugeordnet. Checkstyle weist zusätzlich Charakteristika des Werkzeugtyps Messwerkzeuge durch die Möglichkeit der Extrahierung von Informationen aus vorliegendem Quellcode auf. Der Funktionsumfang von Checkstyle ist aufgrund der Unterstützung weiterer Programmierstandards, wie die Sun Code Conventions oder Google Java Style, skalierbar (vgl. [46]). Für die Gegenüberstellung der Funktionalitäten von Checkstyle und den Software Qualitätsmaßen des System- und Software Produkt Qualitätsmodells sind ausschließlich die standardmäßig integrierten Konventionen des Werkzeugs Checkstyle relevant.

<sup>50</sup> Eigene Darstellung nach ([47])



## Checkstyle Results

The following document contains the results of Checkstyle 7.8 with /google\_checks.xml ruleset: [Link](#)

## Summary

Files	Info	Warnings	Errors
897	0	2807	0

## Rules

Category	Rule	Violations	Severity
blocks	EmptyCatchBlock <ul style="list-style-type: none"> <li>exceptionVariableName: "expected"</li> </ul>	82	Warning
	LeftCurly <ul style="list-style-type: none"> <li>maxLength: "100"</li> </ul>	9	Warning
	NeedBraces	26	Warning
	RightCurly <ul style="list-style-type: none"> <li>tokens: "CLASS_DEF, METHOD_DEF, CTOR_DEF, LITERAL_FOR, LITERAL_WHILE, STATIC_INIT, INSTANCE_INIT"</li> <li>id: "RightCurlyAlone"</li> <li>option: "alone"</li> </ul>	17	Warning
	RightCurly <ul style="list-style-type: none"> <li>tokens: "LITERAL_TRY, LITERAL_CATCH, LITERAL_FINALLY, LITERAL_IF, LITERAL_ELSE, LITERAL_DO"</li> <li>id: "RightCurlySame"</li> </ul>	17	Warning
coding	IllegalTokenText <ul style="list-style-type: none"> <li>format: "\\u00(09 0(a A) 0(c C) 0(d D) 22 27 5(C c)) \\(0(10 11 12 14 15 42 47) 134)"</li> <li>tokens: "STRING_LITERAL, CHAR_LITERAL"</li> <li>message: "Consider using special escape sequence instead of octal value or Unicode escaped value."</li> </ul>	1	Warning

Abb. 35 – Auswertung der Einhaltung von Programmierkonventionen mit Checkstyle<sup>51</sup>

Die Ergebnisse einer statischen Messung von Quellcode auf der Grundlage von vorgegebenen Programmierkonventionen werden von Checkstyle in einem Bericht dokumentiert. Bei dem exemplarischen Bericht in Abb. 35 handelt es sich um einen Ausschnitt und nicht um ein vollständiges Ergebnis. Die Übersicht verdichtet die Resultate einer Messung und beinhaltet Details zur Anzahl der von Checkstyle analysierten Dateien und identifizierten Auffälligkeiten gruppiert nach Kategorien, Informationen, Warnungen und Fehlern.

Problems 13 Progress				
0 errors, 152,128 warnings, 0 others (Filter matched 100 of 152,128 items)				
Description	Resource	Path	Location	Type
Other (100 of 152,128 items)				
class def rcurl' hat eine unerwartete Einrückungstiefe von 4 (erwartet: 2).	XMLSchemaValidator.java	/xerces-2.11.0/src/org/apache/xerces/impl/xs	line 4601	Checkstyle Problem
method def rcurl' hat eine unerwartete Einrückungstiefe von 8 (erwartet: 4).	XMLSchemaValidator.java	/xerces-2.11.0/src/org/apache/xerces/impl/xs	line 4600	Checkstyle Problem
'if rcurl' hat eine unerwartete Einrückungstiefe von 12 (erwartet: 6).	XMLSchemaValidator.java	/xerces-2.11.0/src/org/apache/xerces/impl/xs	line 4599	Checkstyle Problem
Das Untererelement von 'if hat eine unerwartete Einrückungstiefe von 16 (erwartet: 8).	XMLSchemaValidator.java	/xerces-2.11.0/src/org/apache/xerces/impl/xs	line 4598	Checkstyle Problem
Das Untererelement von 'if hat eine unerwartete Einrückungstiefe von 16 (erwartet: 8).	XMLSchemaValidator.java	/xerces-2.11.0/src/org/apache/xerces/impl/xs	line 4597	Checkstyle Problem
Das Untererelement von 'if hat eine unerwartete Einrückungstiefe von 16 (erwartet: 8).	XMLSchemaValidator.java	/xerces-2.11.0/src/org/apache/xerces/impl/xs	line 4596	Checkstyle Problem
'else' hat eine unerwartete Einrückungstiefe von 12 (erwartet: 6).	XMLSchemaValidator.java	/xerces-2.11.0/src/org/apache/xerces/impl/xs	line 4595	Checkstyle Problem
'J' an Position 13 sollte auf der gleichen Zeile stehen wie der nächste Teil der Multi-Block-Anweisung	XMLSchemaValidator.java	/xerces-2.11.0/src/org/apache/xerces/impl/xs	line 4594	Checkstyle Problem
'if rcurl' hat eine unerwartete Einrückungstiefe von 12 (erwartet: 6).	XMLSchemaValidator.java	/xerces-2.11.0/src/org/apache/xerces/impl/xs	line 4594	Checkstyle Problem
Das Untererelement von 'if hat eine unerwartete Einrückungstiefe von 16 (erwartet: 8).	XMLSchemaValidator.java	/xerces-2.11.0/src/org/apache/xerces/impl/xs	line 4593	Checkstyle Problem
'if' hat eine unerwartete Einrückungstiefe von 12 (erwartet: 6).	XMLSchemaValidator.java	/xerces-2.11.0/src/org/apache/xerces/impl/xs	line 4592	Checkstyle Problem
'method def modifier' hat eine unerwartete Einrückungstiefe von 8 (erwartet: 4).	XMLSchemaValidator.java	/xerces-2.11.0/src/org/apache/xerces/impl/xs	line 4591	Checkstyle Problem
'method def rcurl' hat eine unerwartete Einrückungstiefe von 8 (erwartet: 4).	XMLSchemaValidator.java	/xerces-2.11.0/src/org/apache/xerces/impl/xs	line 4584	Checkstyle Problem
Das Untererelement von 'method def' hat eine unerwartete Einrückungstiefe von 12 (erwartet: 6).	XMLSchemaValidator.java	/xerces-2.11.0/src/org/apache/xerces/impl/xs	line 4583	Checkstyle Problem
'for rcurl' hat eine unerwartete Einrückungstiefe von 12 (erwartet: 6).	XMLSchemaValidator.java	/xerces-2.11.0/src/org/apache/xerces/impl/xs	line 4582	Checkstyle Problem
'if rcurl' hat eine unerwartete Einrückungstiefe von 16 (erwartet: 8).	XMLSchemaValidator.java	/xerces-2.11.0/src/org/apache/xerces/impl/xs	line 4581	Checkstyle Problem
Das Untererelement von 'if hat eine unerwartete Einrückungstiefe von 20 (erwartet: 10).	XMLSchemaValidator.java	/xerces-2.11.0/src/org/apache/xerces/impl/xs	line 4580	Checkstyle Problem
'if' hat eine unerwartete Einrückungstiefe von 16 (erwartet: 8).	XMLSchemaValidator.java	/xerces-2.11.0/src/org/apache/xerces/impl/xs	line 4579	Checkstyle Problem

Abb. 36 – Messung der Programmierkonventionen mit dem Checkstyle Eclipse Plugin<sup>52</sup>

Eine andere Darstellungsform der Messergebnisse kann mit dem Checkstyle Plugin in der IDE Eclipse erzeugt werden. Die in Abb. 36 dargestellte Verletzung von Codekonventionen resultiert aus einer Messung des, bereits in Kapitel 4.2.2.1 mit dem Werkzeug CodeInspector analysierten, Open Source Projekts des Java Parsers Xerces der Version 2.11.0. Die generierten Ergebnisse von Checkstyle sind mit denen des CodeInspectors aufgrund abweichender Metriken nicht vergleichbar. Die in Checkstyle integrierten Metriken zur Erfüllung von Programmierstandards können zur Bestimmung der Parameter der Berechnungsvorschrift des in Kapitel 4.1 beschriebenen Software Qualitätsmaßes mit der ID MRe-2-S und der Bezeichnung CODING RULES CONFORMITY genutzt werden, da die Anzahl der Verletzungen von

<sup>51</sup> Darstellung aus ([49])

<sup>52</sup> Eigene Darstellung

Codekonventionen und die Anzahl von Modulen aus den Daten im Werkzeug abgeleitet werden können. Die Bewertung des Werkzeugs zur Bestimmung der Maßausprägung entspricht dem Wert 0,5, da die Bildung des Quotienten von Checkstyle nicht automatisiert erfolgt. Das Software Qualitätsmaß mit der ID ULe-3-S und der Bezeichnung ERROR MESSAGE UNDERSTANDABILITY, welches laut quantitativer Analyse in Kapitel 4.1 eine Abhängigkeit zu Programmierkonventionen besitzt, ist durch das Werkzeug Checkstyle nicht messbar, da weder die Anzahl der implementierten Fehlermeldungen, noch die Fehlermeldungen die auf Fehlerursachen hinweisen mit dem Werkzeug Checkstyle bestimmt werden konnten. Die Eignung der Messung des Software Qualitätsmaßes mit dem Werkzeug Checkstyle wird mit dem Wert 0 dokumentiert. Es konnte bei der Gegenüberstellung der Funktionen von Checkstyle und den Software Qualitätsmaßen des System- und Software Produkt Qualitätsmodells kein Software Qualitätsmaß identifiziert werden, welches mit dem Werkzeug direkt messbar ist.

#### 4.2.2.3 Slicing Werkzeuge

CodeSurfer ist ein proprietärer Quelltext Browser des Unternehmens GrammaTech zur statischen Analyse. Das Werkzeug ist für die Betriebssysteme Linux, Windows und Solaris geeignet. Die unterstützten Programmiersprachen sind in Tab. 16 abgebildet.

Programmierersprachen			
C	C++	Intel x86 machine code	

Tab. 16 – Unterstützte Programmiersprachen des Werkzeugs CodeSurfer<sup>53</sup>

Das Werkzeug CodeSurfer dient primär zur Steigerung der Verständlichkeit und der daraus resultierenden erhöhten Effizienz bei manuellen Analysen von Quellcode. Die im Rahmen der Untersuchung betrachtete Version des Werkzeugs entspricht 3.0. Die Funktionalitäten des Werkzeugs CodeSurfer sind in Tab. 17 aufgelistet.

Funktionalitäten			
Vollständige Programm Analyse	Kontrollfluss-abhängigkeitsanalyse	Erstellung von Aufrufgraphen	GMOD/GREF Analyse
Auswirkungsanalyse	Datenflussanalyse	Zeiger Analyse	Analyse von Präprozessor Effekten

Tab. 17 – Funktionalitäten des Werkzeugs CodeSurfer<sup>54</sup>

Durch die integrierten Funktionen der Erstellung von Aufrufgraphen, Kontrollflussabhängigkeits-, Auswirkungs- und Zeiger Analysen, deren Übereinstimmung mit den Definitionen Fehlverhalten von Software zu identifizieren (vgl [25], S. 398) oder Informationen über Wirkungszusammenhänge von Anweisungen und Variablen bereitzustellen (vgl [25], S. 285), ist der CodeSurfer dem Werkzeugtyp Slicing Werkzeuge zugeordnet. Zusätzlich stellt das Werkzeug CodeSurfer Funktionalitäten des Werkzeugtyps zur Erzeugung von Grafiken und Tabellen bereit.

<sup>53</sup> Eigene Darstellung nach ([50])

<sup>54</sup> Eigene Darstellung nach ([50])



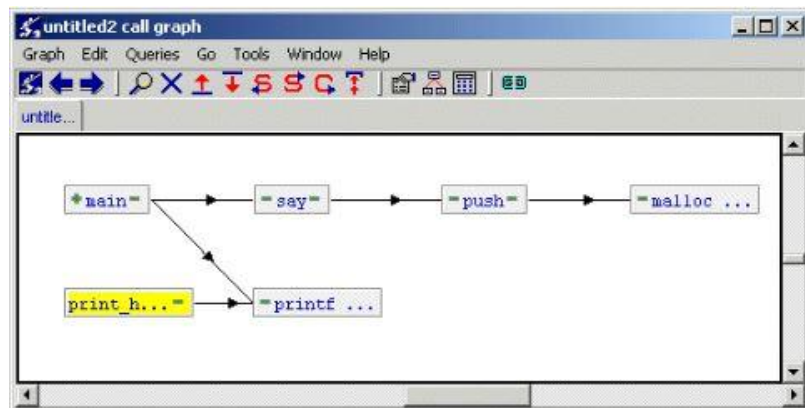


Abb. 37 – Call Graph des Werkzeugs CodeSurfer

In Abb. 37 ist die Funktionalität der Visualisierung eines Aufrufgraphen mit dem Werkzeug CodeSurfer dargestellt. Die Knoten beschreiben einzelne Funktionen. Die Abhängigkeiten zwischen zwei Funktionen werden durch Pfeile symbolisiert. Das in Kapitel 4.2.1.2 beschriebene Software Qualitätsmaß TEST RESTARTABILITY mit ID MTe-3-S berechnet sich aus der Anzahl der Tests die an einem definierten Breakpoint angehalten und neugestartet werden können im Verhältnis zur Anzahl der pausierbaren Tests. Die durchführbaren Tests mit dem Werkzeug Code Surfer können vollständig pausiert und neugestartet werden. Eine direkte Berechnung der Summe von Tests und des Quotienten der Berechnungsvorschrift ist mit dem Werkzeug nicht möglich. Die Bewertung zur Operationalisierung erfolgt mit 0,5. MODIFICATION CORRECTNESS ist ein Software Qualitätsmaß des Qualitätsmerkmals Wartbarkeit und besitzt die ID MMd-2-G. Die Bestimmung der Software Qualitätsmaßausprägung erfolgt durch die Bildung des Quotienten der Anzahl von Modifikationen die zu einem Incident geführt haben zur Summe aller implementierten Modifikationen, subtrahiert vom Wert 1. Beide Parameter der Metrik sind mit dem Werkzeug CodeSurfer nicht messbar. Für die Eignung zur Messung des Software Qualitätsmaßes MMd-2-G wird der Wert 0 vergeben. Während der Analyse konnte kein Software Qualitätsmaß des untersuchten Qualitätsmodells ermittelt werden, dessen Wert mit dem Werkzeug CodeSurfer direkt bestimmt werden kann.

### 4.2.3 Formale Verifikationswerkzeuge

Formale Verifikationswerkzeuge werden überwiegend im universitären Bereich zur Forschung eingesetzt (vgl. [25], S. 398 f.). Die Verifikation von Software untersucht, ob ein Produkt der zugehörigen Spezifikation entspricht und somit das Produkt als korrekt bezeichnet werden kann (vgl. [51], S. 1226 ff.). Das Jessie Plugin des Instituts Inria ist ein in das Framework Frama-C integriertes Werkzeug zur formalen Verifikation und statischen Code Analyse von C Programmen für die Betriebssysteme Windows, FreeBSD, Linux und MAC OS X. Die im Rahmen der Thesis analysierte Version des Jessie Plugins entspricht 2.35 (vgl. [52], S. 5).

#### Programmiersprachen

C
---

Tab. 18 – Unterstützte Programmiersprachen durch das Jessie Plugin<sup>55</sup>

Eine Zusammenfassung der vom Jessie Plugin bereitgestellten Funktionen ist in Tab. 19 aufgeführt.

<sup>55</sup> Eigene Darstellung nach ([52], S. 5)

Funktionalitäten	Beschreibung
Verifikation von Sicherheitskriterien	Verifikationsbedingungen für Speicher Sicherheit durch Identifikation der Dereferenzierung von Nullzeigern, arithmetischen Überläufen und Terminierung von Schleifen.
Funktionale Verifikation	Prüfung Default- und nutzerdefiniertes Verhalten anhand von Schleifeninvarianten, Vor-/ und Rahmenbedingungen.
Separierung von Speicherbereichen	Definition für welche Speicherbereiche spezifische Verifikationsbedingungen relevant sind und ausgeführt werden.

Tab. 19 – Funktionalitäten des Werkzeugs Jessie Plugin<sup>56</sup>

Die Verifikation von Software ermöglicht das Jessie Plugin durch Anwendung von Prüfanweisungen auf offenliegenden Quellcode. Die Semantik der von Jessie genutzten Syntax ist in Tab. 20 dargestellt.

Statement	Semantik
/*@ ... @*/	Spezifikation des Verhaltens einer Methode.
ensures	Verifikationsbedingung für Werte von Argumenten, die nach Ausführung von Methoden geprüft werden.
requires	Verifikationsbedingung die vor Ausführung von Methoden geprüft werden.
predicate	Konsolidierung doppelter oder ähnlicher Verifikationsbedingungen.
invariant	Verifikationsbedingung die über Ausführung von Programmbefehlen hinweg gilt.
assigns	Zuweisung von Variableninhalten zur Verifikationsbedingung.
behavior	Festlegung des Verhaltens von Verifikationsbedingungen.
assumes	Kontext für den Verifikationsbedingung gültig ist.
\result	Kennzeichnet Rückgabewert von Methoden.

Tab. 20 – Semantik der Anweisungen des Jessie Plugins<sup>57</sup>

Im Codeausschnitt 2 ist zum Verständnis exemplarisch der Code einer Prüfanweisung für eine Funktion zur Kalkulation von maximalen Werten von Übergabeparametern aufgeführt, die den Wert 0 bei erfüllter Verifikationsbedingung und den Wert -1 im Fehlerfall zurückliefert.

```

/*@ requires \valid(i) && \valid(j);
   @ requires r == \null || \valid(r);
   @ assigns *r;
   @ behavior zero:
   @   assumes r == \null;
   @   assigns \nothing;
   @   ensures \result == -1;
   @ behavior normal:
   @   assumes \valid(r);
   @   assigns *r;
   @   ensures *r == \max(*i,*j);
   @   ensures \result == 0;
   @*/
int max(int *r, int* i, int* j) {
    if (!r) return -1;
    *r = (*i < *j) ? *j : *i;
    return 0;
}

```

Codeausschnitt 2 – Prüfung von Source Code mit dem Werkzeug Jessie<sup>58</sup>

<sup>56</sup> Eigene Darstellung nach ([52], S. 9 ff.)

<sup>57</sup> Eigene Darstellung nach ([53], S. 7 ff.)

<sup>58</sup> Codeausschnitt aus ([52], S. 6 ff.)

Die Definition der Verifikationsbedingung erfolgt im Kommentarbereich. Die Parameter  $i$  und  $j$  werden mittels einer Vorbedingung auf valide Werte geprüft. Das Verhalten für den Übergabewert *null* ist so spezifiziert, dass der Wert -1 zurückgeliefert und ein Fehler angezeigt wird. Enthalten die Übergabewerte gültige Werte wird der Wert 0 zurückgegeben.

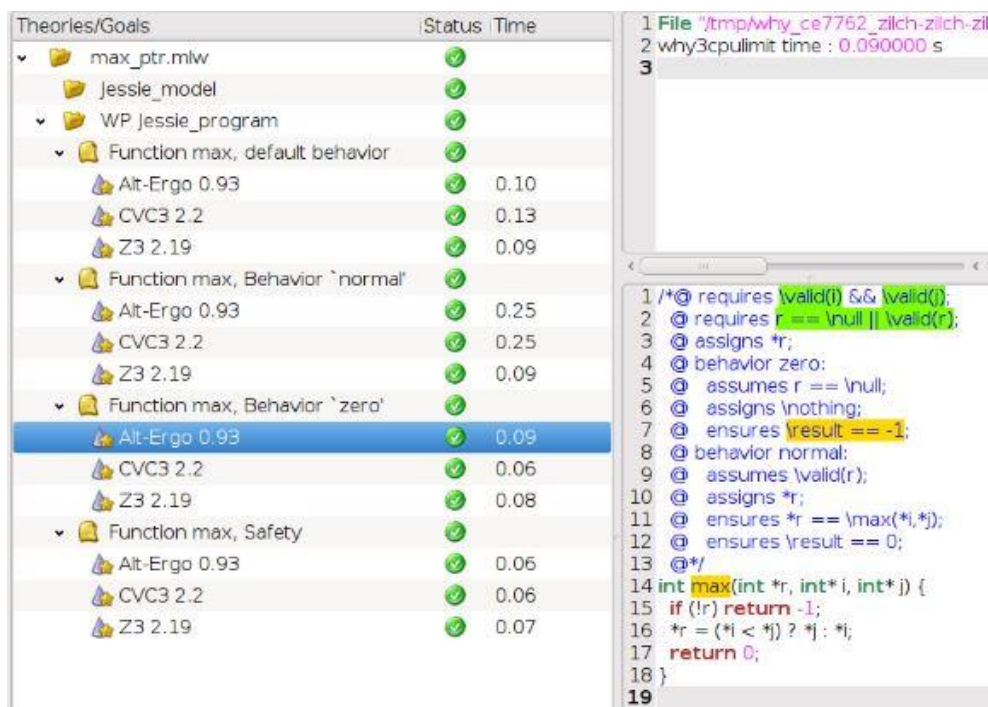


Abb. 38 – Verifikation von Nutzer definierten Verhalten mit dem Jessie Plugin<sup>59</sup>

Abb. 38 zeigt die funktionale Verifikation des beschriebenen Szenarios mit dem Jessie Plugin. Das Jessie Plugin eignet sich aufgrund der fehlenden Möglichkeit des Nachweises von Eigenschaften eines Software Produktes durch Anwendung algebraischer Verfahren nur bedingt als Repräsentant der Werkzeugklasse der formalen Verifikationswerkzeuge. Ein erforderlicher Abgleich von Spezifikationen mit Software Produkten und der daraus resultierenden Verifikation von Eigenschaften ist mit dem Jessie Plugin nur eingeschränkt möglich. Als Alternative konnte kein Werkzeug identifiziert werden, welches der Definition der formalen Verifikationswerkzeuge nach LIGGESMEYER entspricht. Aufgrund fehlender Alternativen erfolgt der Abgleich der Software Qualitätsmaße des System- und Software Produkt Qualitätsmodells mit dem Jessie Plugin. Die Software Metrik mit der ID SIn-3-S und der Bezeichnung BUFFER OVERFLOW PREVENTION besteht aus dem Verhältnis der Anzahl von Nutzereingaben für die eine Prüfung von Grenzwerten implementiert ist zur Summe der Speicherzugriffe durch Nutzereingaben von Software Modulen. Das Jessie Plugin bietet die Möglichkeit Speicherzugriffe und Schwellwerte zu prüfen, allerdings erfordert diese für jede Methode im Quellcode eine explizite Definition einer Prüfanweisung. Eine Aufbereitung der Information über die Anzahl von Prüfanweisungen wird durch das Werkzeug nicht vorgenommen und kann lediglich aus den Daten im Werkzeug abgeleitet werden. Die Bewertung der Eignung des Jessie Plugins zur Messung von SIn-3-S wird mit 0,5 dokumentiert. Das Software Qualitätsmaß mit der ID PCa-3-S und der Bezeichnung USER ACCESS INCREASE ADEQUACY wird aus dem Verhältnis von in einem Zeitraum erfolgreich hinzugefügten Nutzern zum Zeitraum der Observation gebildet. Beide Parameter können nicht mit dem Jessie Plugin bestimmt werden. Die Bewertung der Eignung zur Messung des Software Qualitätsmaßes PCa-3-S mit dem Jessie Plugin erfolgt mit 0. Es konnte kein Software Qualitätsmaß im System- und Software Produkt Qualitätsmodell identifiziert werden, welches direkt mit dem Jessie Plugin erhoben werden kann.

<sup>59</sup> Abb. aus ([52], S. 8)

#### 4.2.4 Modellierende und analysierende Werkzeuge

Im folgenden Abschnitt werden modellierende und analysierende Werkzeuge daraufhin geprüft, ob diese zur Bestimmung der Parameter der Berechnungsvorschriften im System- und Software Produkt Qualitätsmodells der ISO/IEC 25010 geeignet sind. Für die Analyse wird jeweils ein Werkzeug der Kategorie Markov Modellierungswerkzeuge, Fehlermöglichkeits-Einfluss- und Kritikalitätsanalyse (FMECA) Werkzeuge und Fehlerbaumanalyse Werkzeuge vorgestellt, anhand der zugehörigen Eigenschaften untersucht und bewertet.

##### 4.2.4.1 Markov Modellierungswerkzeuge

Open Markov ist ein Open Source Werkzeug des Research Center on Intelligent Decision Support Systems (CISIAD) zur Editierung und Bewertung von probabilistischen graphischen Modellen (PGMs). Zu den PGMs gehören Bayes'sche Netze, Einflussdiagramme, Markov Netze und versteckte Markov Modelle. Die integrierten Funktionen des Werkzeugs Open Markov der Version 0.2.0 sind in Tab. 21 aufgeführt.

Funktionalitäten			
Lernen und Editierung von Bayes'schen Netzen	Erstellung von Einflussdiagrammen	Aufstellung von Kosten-Effizienz Analysen	Erstellung von Markov Netzen

Tab. 21 – Funktionalitäten des Werkzeugs Open Markov<sup>60</sup>

Das Werkzeug Open Markov ist aufgrund der Modulierungsfunktionen von graphischen Modellen keiner spezifischen Programmiersprache zugeordnet. Es eignet sich durch die bereitgestellten Funktionen als Repräsentant des Werkzeugtyps Markov Modellierungswerkzeuge.

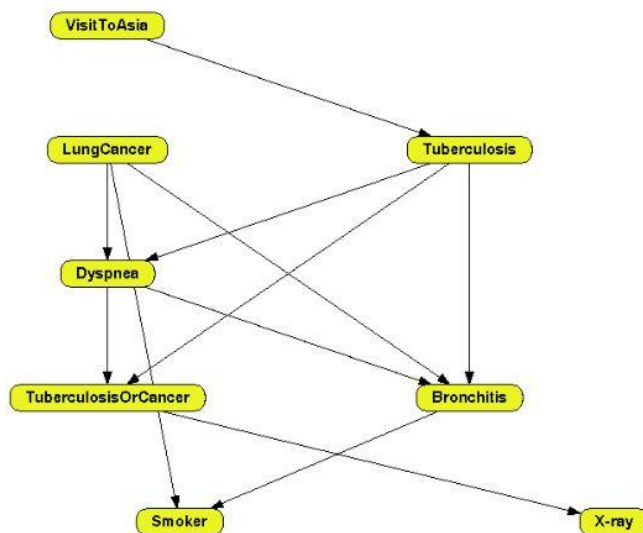
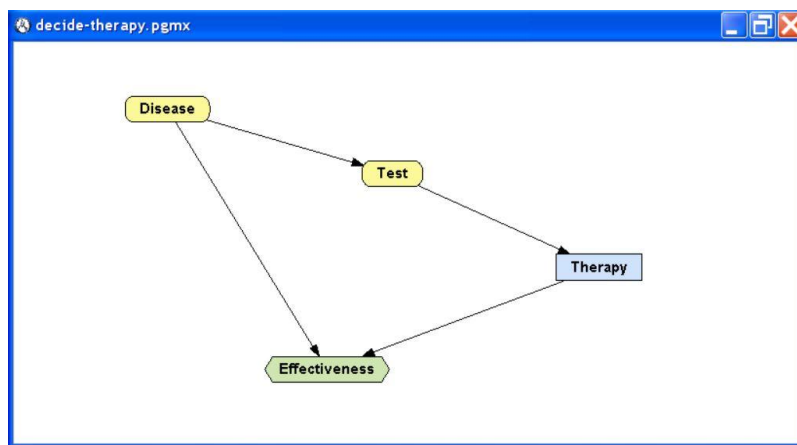


Abb. 39 – Bayes'sches Netz mit dem Werkzeug Open Markov<sup>61</sup>

In Abb. 39 ist ein von Open Markov erlerntes Bayes'sches Netz dargestellt. Die einzelnen Knoten symbolisieren eine Abhängigkeitsstruktur von Ereignissen für deren Auftreten Wahrscheinlichkeiten hinterlegt sind und die mit dem Werkzeug automatisiert ausgewertet werden können.

<sup>60</sup> Eigene Darstellung nach ([54], S. 3 ff.)

<sup>61</sup> Darstellung aus ([54], S. 11)

Abb. 40 – Einflussdiagramm mit dem Werkzeug Open Markov<sup>62</sup>

Ein Einflussdiagramm zur Darstellung einer Abhängigkeitsstruktur mit den Bestandteilen Entscheidungs-, Nutzwert- und Wahrscheinlichkeitsknoten ist in Abb. 40 abgebildet. Die zusätzlichen Knotentypen bei Einflussdiagrammen bilden den Unterschied zu Bayes'schen Netzen. Das Software Qualitätsmaß mit der ID MMo-1-G und der Bezeichnung COUPLING OF COMPONENTS wird aus dem Verhältnis von Komponenten die keinen Einfluss auf andere besitzen zu spezifizierten unabhängigen Komponenten berechnet. Durch die Modellierung von Abhängigkeiten mit dem Werkzeug Open Markov können beide Parameter der Berechnungsvorschrift aus dem Datenbestand abgeleitet werden. Die Bewertung der Eignung des Werkzeugs zur Messung von MMo-1-G wird mit 0,5 dokumentiert. Das Software Qualitätsmaß mit der ID SAu-2-S und dem Namen AUTHENTICATION RULES CONFORMITY wird aus dem Quotient der implementierten zu spezifizierten Authentifizierungsmechanismen gebildet. Beide Parameter der Berechnungsvorschrift können nicht mit dem Werkzeug Open Markov bestimmt werden. Der Eignung der Messbarkeit des Software Qualitätsmaßes mit dem Werkzeug Open Markov wird Wert 0 zugewiesen. Es konnte kein Software Qualitätsmaß im System- und Software Produkt Qualitätsmodells identifiziert werden, welches vollständig durch Open Markov erhoben werden kann.

#### 4.2.4.2 FMECA Werkzeuge

Ein FMECA Werkzeug ist die von ReliaSoft entwickelte Software Xfmea zur Erleichterung des Datenmanagements und Berichterstattung für Fehlermöglichkeits- und Einflussanalysen (FMEA). Xfmea der Version 11.1.2 unterstützt die Betriebssysteme Microsoft Windows 7, 8, 8.1, 10 und Microsoft Windows Server 2008 SP2 und R2 (vgl. [55]). Das Werkzeug bietet integrierte Industriestandards und Normen für FMEA Analysen (vgl. [56]). FMECA ist eine analytische Methode zur Steigerung der technischen Zuverlässigkeit von Produkten durch proaktive Vermeidung von Fehlern und Ausfällen. Potenzielle Fehler und Risiken werden anhand ihrer Auftrittswahrscheinlichkeit und ihrem Schweregrad durch Kennzahlen bewertet und in Berichten oder Grafiken aufbereitet. Auf dieser Grundlage können Maßnahmen eingeleitet werden, um Kosten zu vermeiden die durch das Auftreten der Fehler entstehen würden. Xfmea ist keiner spezifischen Programmiersprache zugeordnet. Die Funktionalitäten von Xfmea sind in Tab. 22 dargestellt.

Funktionalitäten	Beschreibung
FMEA	Aufzeichnung und Verwaltung für Fehlermöglichkeits- und Einflussanalysen.

<sup>62</sup> Darstellung aus ([54], S. 34)



Funktionalitäten	Beschreibung
FMEA Datenübertragungsfunktion	Übertragung von Daten zwischen Konstruktions-, Design- und Prozess-FMEAs.
Risikobewertungsfunktion	Bewertung von Fehlern und Kalkulation von Auswirkungen von Fehlern mittels Risiko-Prioritätszahlen (RPZ) oder Kritikalitätsanalysen.
Konsolidierte Informationsbasis	Unterstützungsfunktion zur Erstellung von Testplänen.
Verknüpfung sachbezogener Analysen	Integrierte Unterstützungsfunktion für Designverifikationspläne, Prozess-Flussdiagramme, Produktionslenkungspläne, P-Diagramme, Design Reviews Based on Failure Mode.

Tab. 22 – Funktionalitäten des Werkzeugs Xfmea<sup>63</sup>

Durch die Kombination der Funktionalitäten der Risikobewertung und FMEA ist Xfmea prädestiniert als Repräsentant des Werkzeugtyps der FMECA Werkzeuge.

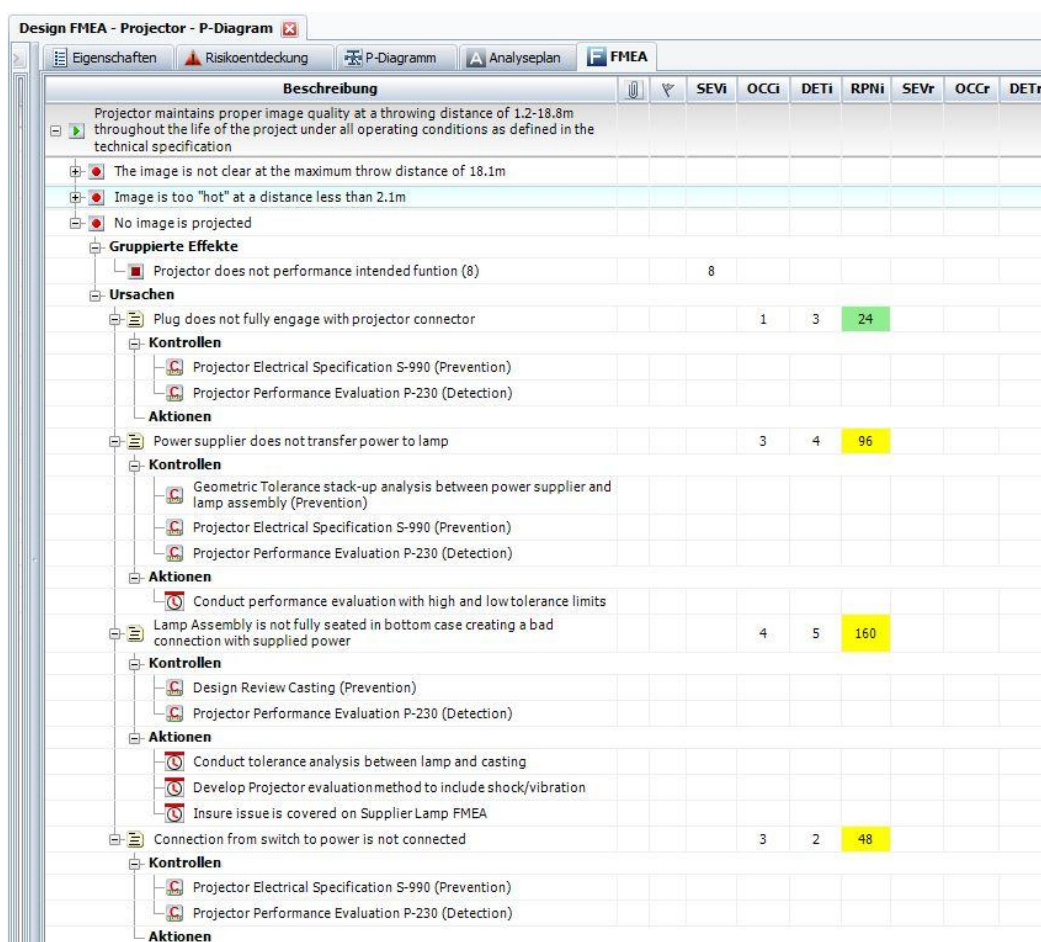
Abb. 41 – FMEA mit dem Werkzeug Xfmea<sup>64</sup>

Abb. 41 zeigt exemplarisch einen Design FMEA eines Default Projekts von Xfmea zur Risikobewertung von Projektoren. Diese umfasst die gruppierten Effekte, Ursachen, zugehörige Kontrollen und mögliche Aktionen zur Fehlerbehebung. Den gruppierten Effekten ist ein Schweregrad hinterlegt. Den Ursachen ist eine Bewertung der Auftritts- und Entdeckungswahrscheinlichkeit auf einer Skala von 1 bis 10 zugewiesen. Je höher der vergebene Wert,

<sup>63</sup> Eigene Darstellung nach ([57])<sup>64</sup> Eigene Darstellung

desto größer sind die Auswirkungen auf das zugrundeliegende System, der Aufwand der Fehlerkorrektur und die Schwierigkeit den Fehler zu identifizieren. Die RPZ berechnet sich wie in Formel (4.2) dargestellt.

$$RPZ = \text{Schweregrad} \cdot \text{Auftreten} \cdot \text{Entdeckung} \quad (4.2)$$

Das Software Qualitätsmaß des korrelierenden Qualitätsmerkmals Zuverlässigkeit mit der ID RFt-1-G und der Bezeichnung FAILURE AVOIDANCE berechnet sich aus dem Verhältnis der Anzahl vermiedener kritischer Fehler zur Anzahl von identifizierten Fehlerbildern durch bereits ausgeführte Testfälle. Die Anzahl der vermiedenen Fehler kann der Berichtsfunktion von Xfmea entnommen werden. Die Generierung von Testfällen wird durch Xfmea zwar unterstützt, allerdings werden die Ausführung und Ergebnisse von Tests nicht im Werkzeug dokumentiert. Die Bewertung von Xfmea zur Bestimmung der Parameter des Software Qualitätsmaßes RFt-1-G entspricht 0,5. Das Software Qualitätsmaß des Qualitätsmerkmals Benutzbarkeit mit der ID RRe-2-S und der Bezeichnung BACKUP DATA COMPLETENESS wird bestimmt durch die Anzahl der regelmäßig gesicherten Daten im Verhältnis zu den benötigten Daten zur Wiederherstellung eines Systems. Beide Parameter der Metrik sind mit Xfmea nicht bestimmbar. Der Eignung der Messung des Software Qualitätsmaßes wird der Wert 0 zugewiesen. Software Qualitätsmaße des System- und Software Produkt Qualitätsmodells, deren Ausprägung vollständig mit dem Werkzeug Xfmea bestimmt werden können, wurden im Rahmen der Analyse nicht identifiziert.

#### 4.2.4.3 Fehlerbaumanalyse Werkzeuge

Die Isograph Reliability Workbench ist ein von Isograph Inc. entwickeltes Werkzeug für das Betriebssystem Microsoft Windows (vgl. [58]), um Aussagen über die Zuverlässigkeit, Sicherheit und Wartbarkeit von Systemen zu ermöglichen (vgl. [59]). Das Werkzeug ist aufgrund seiner Funktionalitäten programmiersprachenunabhängig. Die betrachtete Version des Werkzeugs entspricht 13.0.0.0. Die Funktionalitäten der Isograph Reliability Workbench sind in Tab. 23 beschrieben.

Funktionalitäten			
FMECA	FMEA	Markov Analyse	Ereignisbaum Analyse
Zuverlässigkeits-Blockdiagramme	Weibull Analyse	Fehlerbaumanalyse	

Tab. 23 – Funktionalitäten des Werkzeugs Isograph<sup>65</sup>

Isograph eignet sich aufgrund der Unterstützung von Fehlerbaumanalysen als Repräsentant des Werkzeugtyps Fehlerbaumanalyse Werkzeuge. Während der Analyse der Funktionalitäten des Werkzeugs Isograph wurde festgestellt, dass das Werkzeug nicht nur dem Werkzeugtyp der Fehlerbaumanalyse Werkzeuge zugeordnet werden kann, sondern ebenfalls Merkmale anderer modellierender und analysierender Werkzeugtypen wie FMECA- und Markov Modellierungswerkzeuge aufweist.

<sup>65</sup> Eigene Darstellung nach ([58])

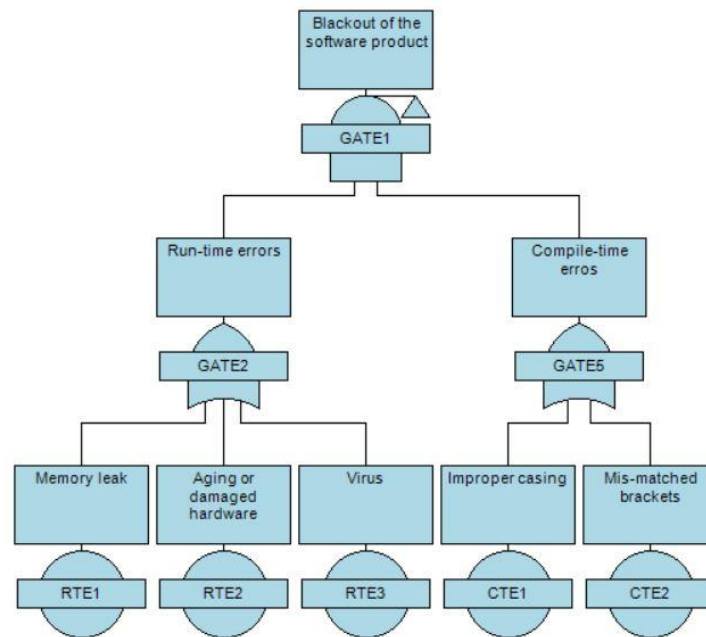


Abb. 42 – Fehlerbaumanalyse mit der Isograph Reliability Workbench<sup>66</sup>

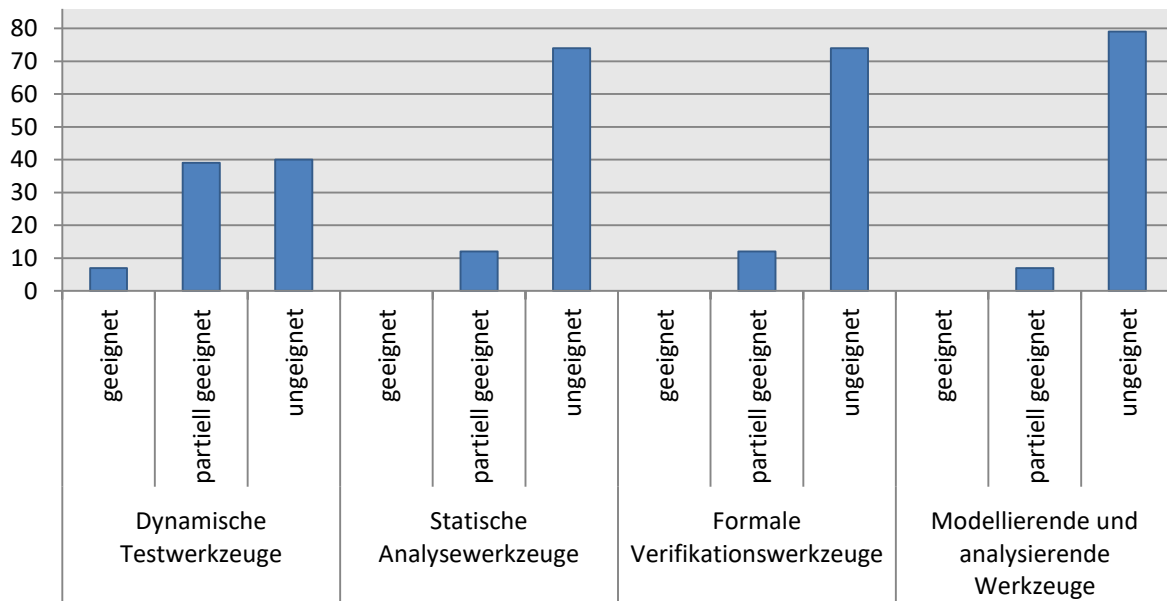
Abb. 42 visualisiert einen Fehlerbaum eines Software Produkts, welcher mit dem Werkzeug Isograph Reliability Workbench erstellt wurde. Der Fehlerbaum beschreibt die Wahrscheinlichkeiten aufeinanderfolgend eintretender Ereignisse und die damit verbundenen Risiken. Das Modell des Fehlerbaums kann im Werkzeug Isograph in einen Ereignisbaum konvertiert werden, um mögliche Fehlerkosten der definierten Szenarien zu bestimmen. Das Software Qualitätsmaß des korrelierenden Qualitätsmerkmals Zuverlässigkeit mit der ID RFt-2-S und der Bezeichnung REDUNDANCY OF COMPONENTS wird aus dem Verhältnis von redundanten Komponenten zur Anzahl der Systemkomponenten berechnet. Die Funktionen von Isograph erlauben die grafische Darstellung von Abhängigkeiten der Komponenten. Die Anzahl der redundanten und Systemkomponenten kann aus den Daten im Werkzeug abgeleitet werden. Eine direkte Messung des Software Qualitätsmaßes ist mit Isograph nicht möglich. Der Eignung von Isograph zur Bestimmung der Parameter der Metrik RFt-2-S wird der Wert 0,5 zugewiesen. Das Software Qualitätsmaß mit der ID UOp-2-G und dem Namen MESSAGE CLARITY wird aus dem Quotient der vom Anwender erwarteten Benachrichtigung der Software zur Anzahl implementierter Benachrichtigungen gebildet. Beide Parameter der Metrik sind mit dem Werkzeug Isograph nicht bestimmbar. Dem Grad der Eignung zur Messung von UOp-2-G wird der Wert 0 zugewiesen. Eine direkte Messung eines Software Qualitätsmaßes aus ISO/IEC 25023 ist mit Isograph nicht möglich.

#### 4.2.5 Auswertung Werkzeugklassen zur Operationalisierung von Merkmalen

Im Rahmen der Untersuchung der T<sub>2</sub> wurde festgestellt, dass die Funktionalitäten der betrachteten Werkzeuge mehreren Werkzeugtypen einer Werkzeugklasse zugeordnet werden können. Die Evaluation der Werkzeugklassen erfolgte auf Grundlage von jeweils einem analysierten Werkzeug jeden Werkzeugtyps.

<sup>66</sup> Eigene Darstellung



Abb. 43 – Evaluation von Werkzeugklassen zur Messung von Software Maßen<sup>67</sup>

Die aufbereiteten Ergebnisse der Analyse der Eignung von Werkzeugklassen zur Messung der System- und Software Produkt Qualitätsmaße sind in Abb. 43 enthalten. Die Visualisierungsform entspricht einem Säulendiagramm, welches für jede Werkzeugklasse die Kategorien geeignet, partiell geeignet und ungeeignet unterscheidet und auf Grundlage der Bewertungen in Tab. 33 im Anhang F die Anzahl der zugewiesenen 1, 0,5 und 0 Werte für jede Werkzeugklasse kumuliert. Die Skala der Grafik besitzt einen Minimalwert von 0 und eine maximale Ausprägung von 86, welche die Gesamtzahl der Software Qualitätsmaße im System- und Software Produkt Qualitätsmodell repräsentiert. Dem Diagramm kann entnommen werden, dass dynamische Testwerkzeuge am qualifiziertesten zur Operationalisierung der untersuchten Qualitätsmerkmale sind. Während dynamische Testwerkzeuge für wenige Parameter als geeignet bewertet wurden, können statische Analysewerkzeuge, formale Verifikationswerkzeuge und modellierende und analysierende Werkzeuge kein Software Qualitätsmaß aus der Norm ISO/IEC 25023 direkt messen.

Werkzeug	Bewertung		Summe	Prozentuales Potenzial der Erfassung von Metrik Parametern
	1	0,5		
Dynamische Testwerkzeuge	11	41	52	53,49%
Statische Analysewerkzeuge	2	10	12	13,95%
Formale Verifikationswerkzeuge	0	12	12	13,95%
Modellierende und analysierende Werkzeuge	0	6	6	6,98%

Tab. 24 – Ermittlung der qualifiziertesten Werkzeugklassen<sup>68</sup>

Eine detaillierte Übersicht der prozentualen Eignung der untersuchten Werkzeugklassen zur Operationalisierung der Qualitätsmerkmale ist in Tab. 24 aufgeführt. Die Werkzeugklasse der dynamischen Testwerkzeuge ist mit einer Abdeckung von 53,49 Prozent der Parameter der

<sup>67</sup> Eigene Darstellung nach Tab. 33<sup>68</sup> Eigene Darstellung nach Tab. 33

Metriken die geeignetste der betrachteten Kategorien. Statische Analysewerkzeuge und formale Verifikationswerkzeuge besitzen mit 13,95 Prozent eine identische Wertung. Modellierende und analysierende Werkzeuge sind mit 6,98 Prozent die Werkzeugklasse mit der geringsten Eignung zur Bestimmung der untersuchten Software Qualitätsmaße.

### 4.3 Auswertung der Teilhypothese 3

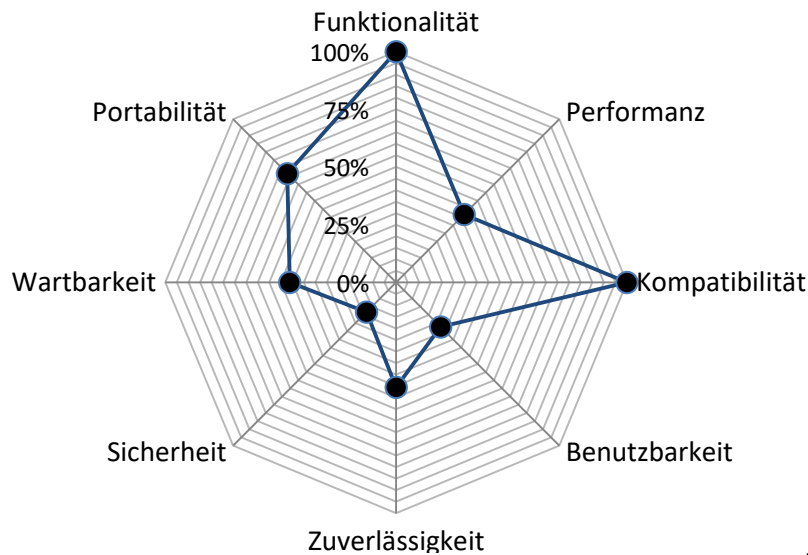
Die Untersuchung der Teilhypothese  $T_3$  basiert auf dem Fazit der Teilhypthesen  $T_2$  und  $T_1$ . Die Ergebnisse der  $T_3$  werden aus Tab. 32 abgeleitet und resultieren aus der Kombination der in  $T_1$  ermittelten Klassifizierungen von Software Qualitätsmaßen des System- und Software Qualitätsmodells mit der in  $T_2$  ermittelten Werkzeugklasse, die sich für die Operationalisierung von Qualitätsmerkmalen am qualifiziertesten erwiesen hat.

Werkzeug	Bewertung		Summe	Prozentuales Potenzial der Erfassung von Metrik Parametern
	1	0,5		
SonarQube	0	14	<b>14</b>	<b>16,28%</b>
ALM	7	31	<b>38</b>	<b>44,19%</b>
SilkTest	2	21	<b>23</b>	<b>26,74%</b>
JProfiler	0	5	<b>5</b>	<b>5,81%</b>

Tab. 25 – Abdeckung der Software Qualitätsmaße durch dynamische Testwerkzeuge<sup>69</sup>

Zur Ermittlung, welche autonomen Werkzeuge der Kategorie dynamische Testwerkzeuge die höchste Anzahl an Software Qualitätsmaußausprägungen bestimmen können, wurde in Tab. 25 die Anzahl der mit 1 und 0,5 bewerteten Software Qualitätsmaße summiert. Als Werkzeug mit der prozentual höchsten Abdeckung der Software Qualitätsmaße der Norm ISO/IEC 25023 wurde mit 44,19 Prozent das Werkzeug ALM identifiziert, gefolgt vom Werkzeug SilkTest mit 26,74 Prozent und SonarQube mit 16,28 Prozent. Das Werkzeug JProfiler mit der Abdeckung von 5,81 Prozent der Software Qualitätsmaße aus dem System und Software Produkt Qualitätsmodell wird, aufgrund des geringen Potenzials, in  $T_3$  nicht untersucht. Die Reihenfolge der Darstellung der Ergebnisse von  $T_3$  erfolgt absteigend zur evaluierten Abdeckung der Software Qualitätsmaße. Zur Visualisierung werden Kiviat-Diagramme genutzt, um Aussagen über die Art der messbaren Qualitätsmaßkategorien der Werkzeuge abzuleiten. Jede Achse der Kiviat-Diagramme ist unabhängig voneinander und resultiert aus der Anzahl der zugeordneten Software Qualitätsmaße der Achsenbezeichnungen. Exemplarisch liegen in Abb. 44 dem Qualitätsmerkmal der Funktionalität vier Software Qualitätsmaße zugrunde, die das Maximum der Achse repräsentieren. Alle vier Software Qualitätsmaße können mit dem Werkzeug ALM erhoben werden. Die Bewertung von ALM zur Operationalisierung des Qualitätsmerkmals Funktionalität entspricht 100 Prozent. Die Achse der Benutzbarkeit basiert auf den 22 zugeordneten Software Qualitätsmaßen und besitzt demzufolge eine Skala mit differenzierten Eigenschaften.

<sup>69</sup> Eigene Darstellung nach Tab. 32

Abb. 44 – Abdeckung der Qualitätsmerkmale durch das Werkzeug ALM<sup>70</sup>

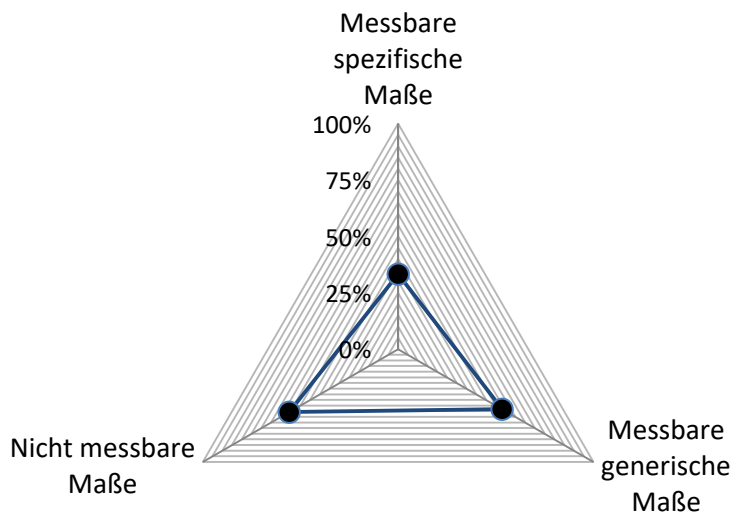
Die Messpunkte auf den Achsen der erstellten Kiviat-Diagramme resultieren aus der Anzahl der Software Qualitätsmaße der eruierten Kategorien (vgl. Tab. 26). Die Grundlage der Achse der nicht messbaren Software Qualitätsmaße bildet sich aus der Summe der definierten Software Qualitätsmaße im System- und Software Produkt Qualitätsmodell (vgl. Abb. 6).

Klassifikationskriterium	Anzahl der Software Qualitätsmaße
Funktionalität	4
Performanz	12
Kompatibilität	4
Benutzbarkeit	22
Zuverlässigkeit	11
Sicherheit	11
Wartbarkeit	13
Portabilität	9
Generisch	47
Spezifisch	39
HR	30
R	35
UD	21
Intern	3
Extern	21
Hybrid	62
Statisch	65
Dynamisch	21

Tab. 26 – Anzahl der Software Qualitätsmaß der T<sub>1</sub>-Kategorien<sup>71</sup>

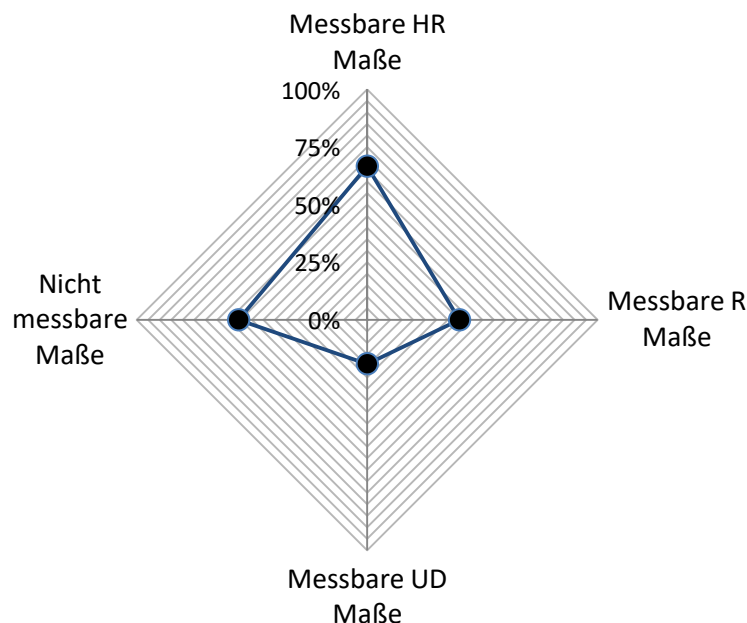
Aus Abb. 44 kann entnommen werden, dass das Werkzeug ALM sich vollständig zur Operationalisierung der Qualitätsmerkmale Funktionalität und Kompatibilität eignet. Maße zur Portabilität, Wartbarkeit, Effizienz und Zuverlässigkeit lassen sich im geringeren Umfang durch ALM bestimmen. Weniger geeignet ist ALM zur Messung der Qualitätsmerkmale der Sicherheit und Benutzbarkeit.

<sup>70</sup> Eigene Darstellung nach Tab. 32<sup>71</sup> Eigene Darstellung



**Abb. 45 – Abdeckung der generischen und spezifischen Maße durch ALM<sup>72</sup>**

Abb. 45 visualisiert die Tauglichkeit von ALM zur Messung der Software Qualitätsmaße mit generischen und spezifischen Eigenschaften. Aus der Darstellung geht hervor, dass der prozentual höhere Anteil der bestimmaren Software Qualitätsmaße einen generischen Charakter und eine höhere Nutzungsempfehlung als spezifische Maße durch das System- und Software Produkt Qualitätsmodell aufweist.

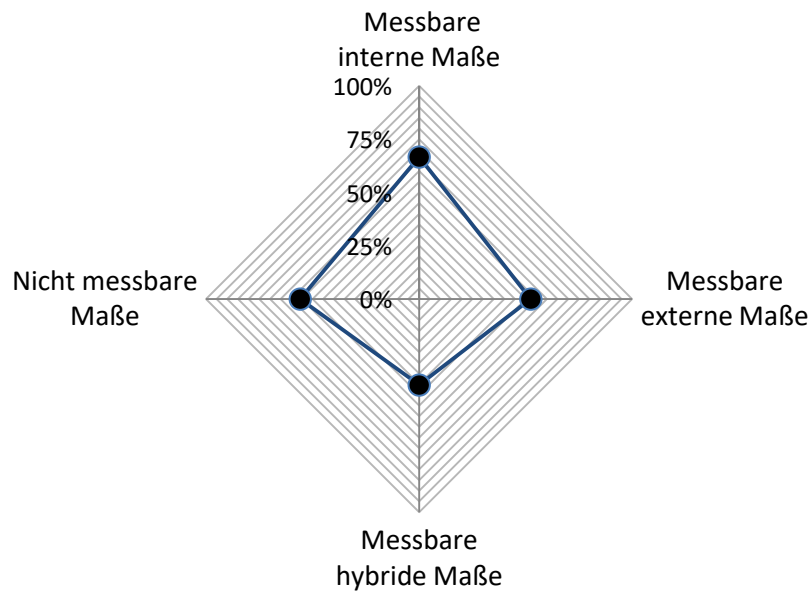


**Abb. 46 – Abdeckung der HR, R, UD Software Qualitätsmaße durch ALM<sup>73</sup>**

Die Abb. 46 beschreibt die prozentualen Anteile der vom Werkzeug ALM ermittelbaren Software Qualitätsmaße und deren Validität der Messresultate. Von den 44,19 Prozent der mit ALM auswertbaren Software Qualitätsmaße ist der prozentual größte Anteil der Kategorie HR angehörig, gefolgt von Maßen der Kategorie R und UD. Der Schwerpunkt der messbaren Maße besteht dementsprechend aus Metriken die zuverlässige Messresultate generieren.

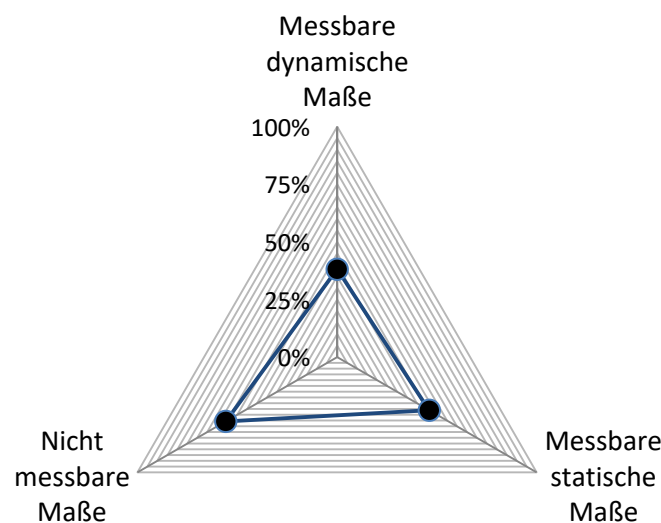
<sup>72</sup> Eigene Darstellung nach Tab. 32

<sup>73</sup> Eigene Darstellung nach Tab. 32



**Abb. 47 – Abdeckung der internen, externen, hybriden Maße durch ALM<sup>74</sup>**

Das Kiviat-Diagramm in Abb. 47 zeigt den prozentualen Anteil der mit dem Werkzeug ALM bestimmbaren internen, externen und hybriden Software Qualitätsmaßen des System- und Software Produkt Qualitätsmodells. Der Grafik ist zu entnehmen, dass die verhältnismäßig höchste Anzahl der Maße interne Eigenschaften besitzen, die aus der inneren Struktur von Software hervorgehen. Der prozentuale Anteil der äußerlich sichtbaren, externen Maße übersteigt den der hybriden Maße.



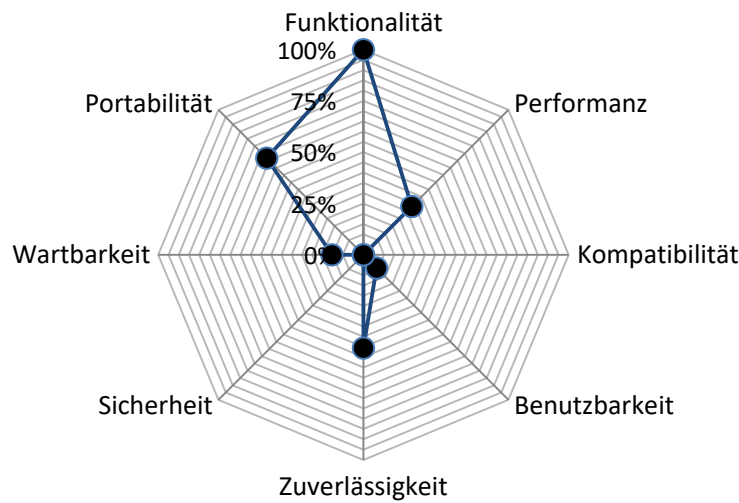
**Abb. 48 – Abdeckung der statischen und dynamischen Maße durch ALM<sup>75</sup>**

In Abb. 48 werden die prozentualen Anteile der durch ALM erfassbaren Software Qualitätsmaße mit statischen und dynamischen Charakteristika abgebildet. Der prozentuale Anteil der aus den Daten in ALM ableitbaren statischen Software Qualitätsmaße übersteigt die der zeitbezogenen dynamischen Maße. Das Werkzeug Silk Test kann zur Bestimmung der Ausprä-

<sup>74</sup> Eigene Darstellung nach Tab. 32

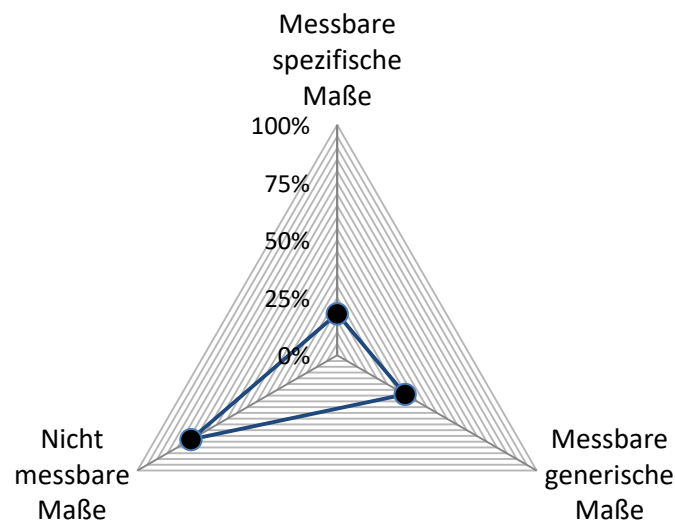
<sup>75</sup> Eigene Darstellung nach Tab. 32

gungen von 26,74 Prozent der im System- und Software Qualitätsmodell definierten Maße genutzt werden.



**Abb. 49 – Abdeckung der Qualitätsmerkmale durch das Werkzeug Silk Test<sup>76</sup>**

Abb. 49 visualisiert die Eignung des Werkzeugs Silk Test zur Ableitung der Parameterwerte der Metriken im System- und Software Produkt Qualitätsmodells für korrelierende Qualitätsmerkmale. Das Qualitätsmerkmal der Funktionalität ist mit dem Werkzeug Silk Test vollständig operationalisierbar. Die prozentualen Anteile der Qualitätsmerkmale Portabilität, Zuverlässigkeit und Performanz übersteigen signifikant die der Qualitätsmerkmale Wartbarkeit, Kompatibilität, Benutzbarkeit und Sicherheit.

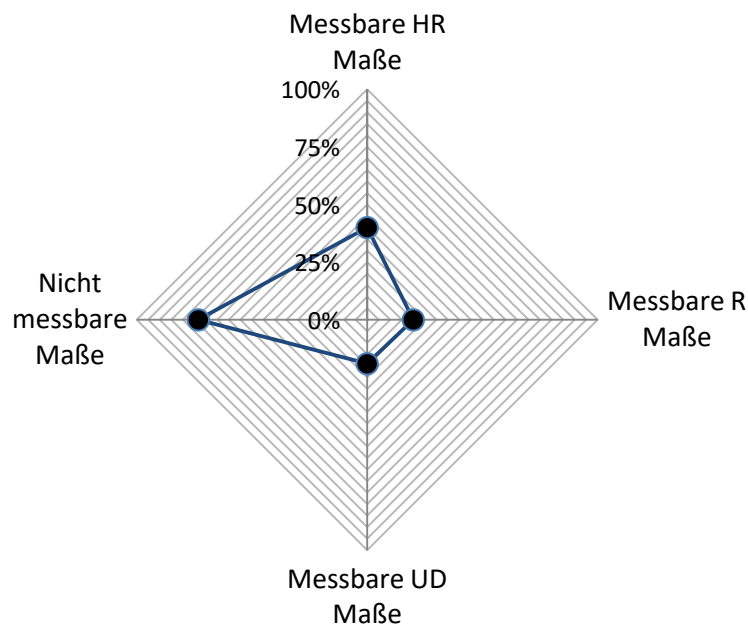


**Abb. 50 – Abdeckung der generischen und statischen Maße durch Silk Test<sup>77</sup>**

Aus Abb. 50 geht hervor, dass die mit dem Werkzeug Silk Test verhältnismäßig größere Anzahl der operationalisierbaren Software Qualitätsmaße generische Eigenschaften und eine dementsprechend höhere Nutzungsempfehlung durch die Norm ISO/IEC 25023 als spezifische Software Qualitätsmaße besitzen.

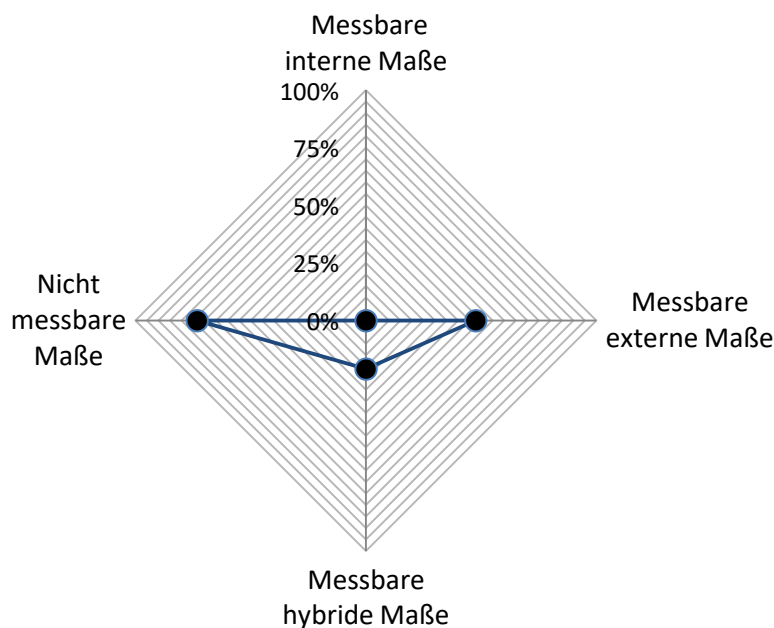
<sup>76</sup> Eigene Darstellung nach Tab. 32

<sup>77</sup> Eigene Darstellung nach Tab. 32



**Abb. 51 – Abdeckung der HR, R, UD Software Qualitätsmaße durch Silk Test<sup>78</sup>**

Die Abb. 51 beschreibt den prozentualen Anteil der von Silk Test messbaren Software Qualitätsmaße der Kategorien HR, R, UD und der damit verbundenen Validität der Messresultate. Von den 26,74 Prozent der mit Silk Test auswertbaren Software Qualitätsmaße besitzen Maße der Kategorie HR den größten prozentualen Anteil, gefolgt von Maßen der Kategorie UD und R. Der prozentual höchste Anteil der mit Silk Test nutzbaren Metriken generiert valide Messresultate.



**Abb. 52 – Abdeckung der internen, externen, hybriden Maße durch Silk Test<sup>79</sup>**

Das Kiviat-Diagramm in Abb. 52 zeigt die Eignung des Werkzeugs Silk Test zur Messung von internen, externen und hybriden Maßen des System- und Software Produkt Qualitätsmodells. Die verhältnismäßig höchste Abdeckung der durch Silk Test messbaren

<sup>78</sup> Eigene Darstellung nach Tab. 32

<sup>79</sup> Eigene Darstellung nach Tab. 32



Software Qualitätsmaße sind der Kategorie der externen Maße, die auswertbar sind ohne die innere Struktur einer Software zu kennen, zugeordnet. Der prozentuale Anteil der aus Silk Test ableitbaren hybriden Maße ist geringer als die der externen Software Qualitätsmaße. Interne Software Qualitätsmaße sind mit dem Werkzeug Silk Test nicht auswertbar.

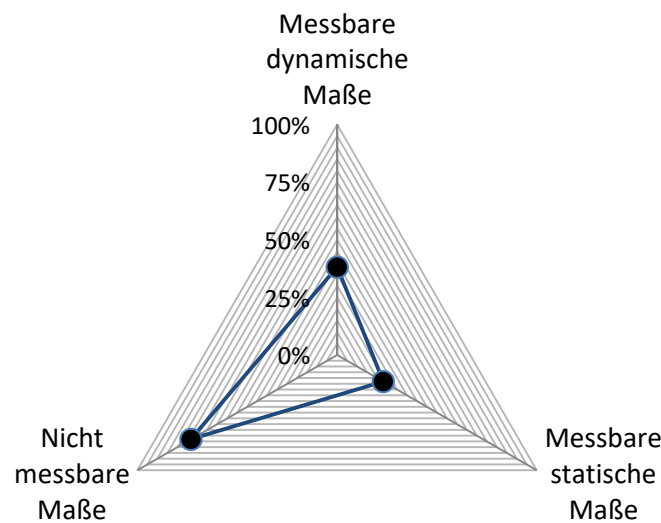


Abb. 53 – Abdeckung der statischen und dynamischen Maße durch Silk Test<sup>80</sup>

In Abb. 53 wird die prozentuale Aufteilung der von Silk Test bestimmbaren statischen und dynamischen Software Qualitätsmaße des System- und Software Produkt Qualitätsmodells abgebildet. Der prozentuale Anteil der messbaren Software Qualitätsmaße die dynamische Charakteristika und einen zeitlichen Bezug aufweisen übersteigt die der bestimmbaren statischen Software Qualitätsmaße. Das Werkzeug SonarQube kann zur Bestimmung von 16,28 Prozent der Parameterausprägungen der Metriken im System- und Software Qualitätsmodell genutzt werden.

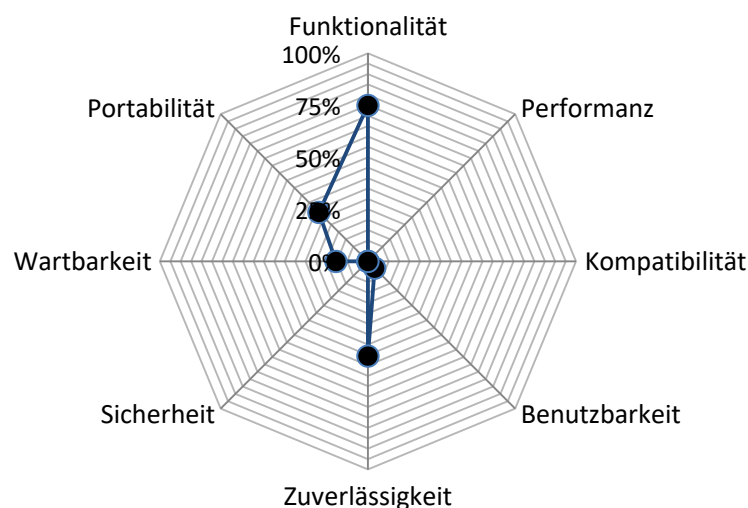
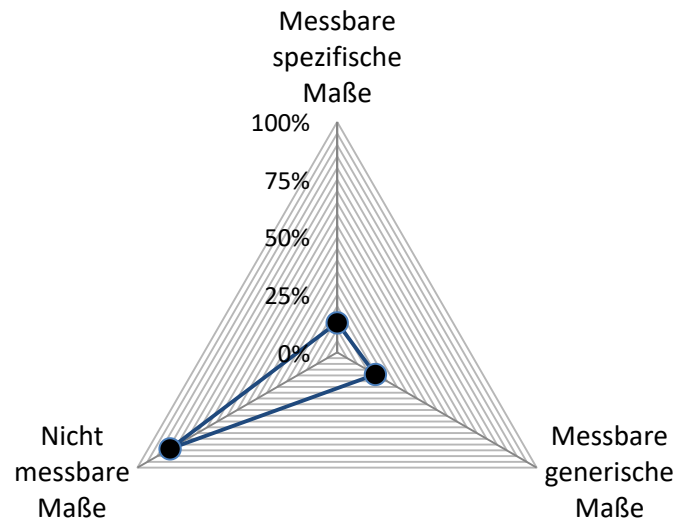


Abb. 54 – Abdeckung der Qualitätsmerkmale durch das Werkzeug SonarQube<sup>81</sup>

<sup>80</sup> Eigene Darstellung nach Tab. 32

<sup>81</sup> Eigene Darstellung nach Tab. 32

Abb. 54 visualisiert die Tauglichkeit des Werkzeugs SonarQube zur Operationalisierung von Qualitätsmerkmalen des System- und Software Produkt Qualitätsmodells. Das mit dem Werkzeug SonarQube am Besten bestimmbare Qualitätsmerkmal ist die Funktionalität. Die Ausprägung der Qualitätsmerkmale Zuverlässigkeit, Portabilität und Wartbarkeit können partiell aus den Daten in SonarQube abgeleitet werden. Die Qualitätsmerkmale Sicherheit, Performanz, Kompatibilität und Benutzbarkeit sind nicht oder nur sehr geringfügig mit dem Werkzeug SonarQube operationalisierbar.



**Abb. 55 – Abdeckung der generischen und statischen Maße durch SonarQube<sup>82</sup>**

Das Kiviat-Diagramm in Abb. 55 beschreibt die Tauglichkeit des Werkzeugs SonarQube zur Messung von Software Qualitätsmaßen mit generischen und spezifischen Eigenschaften. Aus dem Diagramm geht hervor, dass der prozentual höhere Anteil der mit Silk Test auswertbaren Software Qualitätsmaße aufgrund von generischen Charakteristika eine explizite Nutzungsempfehlung durch das System- und Software Produkt Qualitätsmodell aufweist. Der prozentual geringere Anteil der aus den Daten im Werkzeug SonarQube ableitbaren Software Qualitätsmaße weist spezifische Eigenschaften auf.

<sup>82</sup> Eigene Darstellung nach Tab. 32

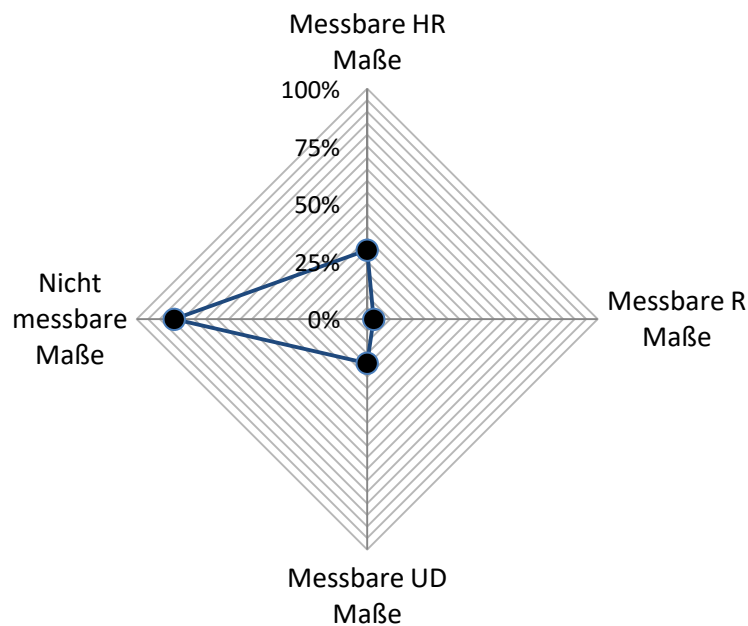


Abb. 56 – Abdeckung der HR, R, UD Software Qualitätsmaße durch SonarQube<sup>83</sup>

Aus Abb. 56 geht der prozentuale Anteil der mit dem Werkzeug SonarQube messbaren Software Qualitätsmaße der Kategorien HR, R, UD und die damit verbundene Validität der Messresultate hervor. Von den mit SonarQube auswertbaren Software Qualitätsmaßen ist der verhältnismäßig größte Anteil der Kategorie HR zugeordnet, gefolgt von der Kategorie UD. Die Software Qualitätsmaße mit der prozentual höchsten Abdeckung der untersuchten Metriken generiert zuverlässige Messresultate. Der Anteil der messbaren Software Qualitätsmaße der Kategorie R ist im Verhältnis zu den Kategorien HR und UD minimal.

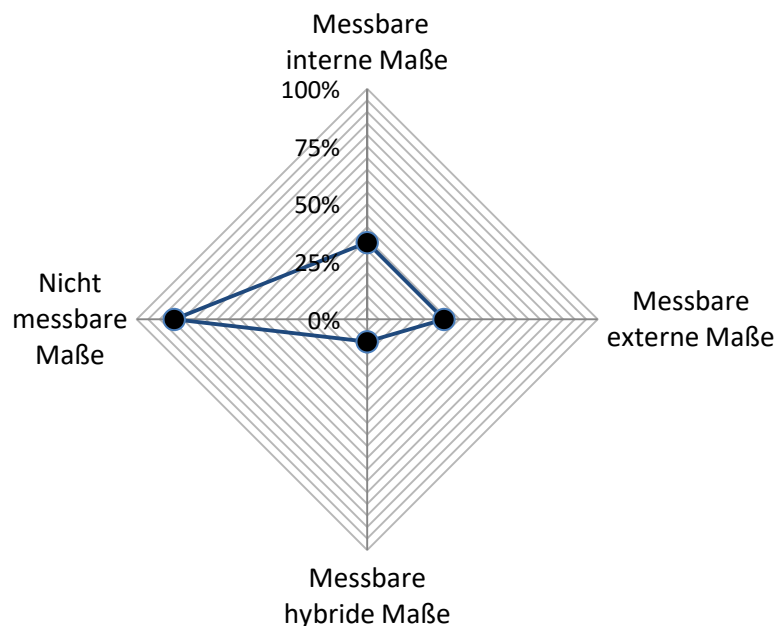


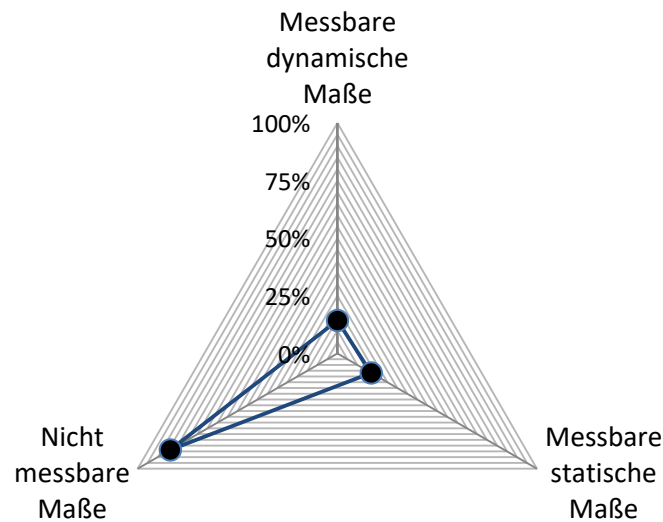
Abb. 57 – Abdeckung der internen, externen, hybriden Maße durch SonarQube<sup>84</sup>

Das Kiviat-Diagramm in Abb. 57 beschreibt den aus den Daten im Werkzeug SonarQube ableitbaren prozentualen Anteil von internen, externen und hybriden Maßen des System- und Software Produkt Qualitätsmodells. Die Grafik zeigt, dass der verhältnismäßig höchste Wert

<sup>83</sup> Eigene Darstellung nach Tab. 32

<sup>84</sup> Eigene Darstellung nach Tab. 32

Software Qualitätsmaße mit internen Eigenschaften repräsentiert, bei denen die innere Struktur von Software relevant ist. Der prozentuale Anteil der äußerlich sichtbaren externen Maße übersteigt den der hybriden Maße.



**Abb. 58 – Abdeckung der statischen und dynamischen Maße durch SonarQube<sup>85</sup>**

In Abb. 58 wird die prozentuale Aufteilung der mit dem Werkzeug SonarQube messbaren Software Qualitätsmaße mit statischen und dynamischen Eigenschaften abgebildet. Der prozentuale Anteil der messbaren statischen Software Qualitätsmaße ohne zeitlichen Bezug übersteigt die der durch SonarQube messbaren dynamischen Maße mit einer Abhängigkeit zum Faktor Zeit.

<sup>85</sup> Eigene Darstellung nach Tab. 32

## 4.4 Auswertung der Gesamthypothese

Die Beantwortung der Gesamthypothese  $H_0$  setzt die Möglichkeit der Bewertung und Klassifizierung von Software Qualitätsmaßen und Werkzeugen anhand der integrierten Funktionalitäten voraus. Exemplarisch wurden die Software Qualitätsmaße der Norm ISO/IEC 25023 des zugehörigen System- und Software Produkt Qualitätsmodells aus ISO/IEC 25010 in Teilhypothese  $T_1$  anhand einem Ansatz aus der Literatur und Ansätzen die aus dem SQuaRE Framework abgeleitet werden konnten voneinander abgegrenzt. Die zur Beantwortung von  $H_0$  notwendige Klassifizierung von Werkzeugen zur Operationalisierung von Qualitätsmerkmalen wurde auf Basis von Fachliteratur ermittelt und im Rahmen der Bachelorthesis beschrieben. Zur Evaluation der Tauglichkeit von einzelnen Werkzeugklassen zur Operationalisierung von Qualitätsmerkmalen wurde in  $T_2$  untersucht, ob durch die Funktionalitäten der analysierten Werkzeuge die Parameterausprägungen der Metriken des System- und Software Produkt Qualitätsmodells bestimmt oder aus den vorliegenden Daten im Werkzeug abgeleitet werden können. Auf Grundlage des Bewertungsmodells wurde aus den Kategorien der statischen Analysewerkzeuge, dynamischen Testwerkzeuge, formalen Verifikationswerkzeuge und modellierenden- und analysierenden Werkzeuge die Gruppe der dynamischen Testwerkzeuge als qualifizierteste eruiert. Hierbei ist aufgefallen, dass die Funktionalitäten einzelner untersuchter Werkzeuge Merkmale mehrerer Werkzeugtypen der Definition von LIGGESMEYER aufweisen. Das in  $T_3$  untersuchte Werkzeug mit der prozentual höchsten Abdeckung der Software Qualitätsmaße aus ISO/IEC 25023 ist das dynamische Testwerkzeug ALM des Unternehmens HP. Dennoch zeigt das Ergebnis, dass weniger als die Hälfte der im Qualitätsmodell enthaltenen Software Qualitätsmaße, mit einem einzelnen im Rahmen der Ausarbeitung betrachteten Werkzeug, messbar sind und eine hohe Abhängigkeit zu den Eingangsgrößen Anforderung, Betrieb, Umgebungsbedingungen, Kontext und Anwender aufweisen. Ebenfalls unterscheiden sich die Qualitätsmerkmale im System- und Software Qualitätsmodell durch die Komplexität ihrer Messbarkeit. Die Qualitätsmerkmale der Funktionalität, Kompatibilität, Zuverlässigkeit, Portabilität und Wartbarkeit sind besser mit den geprüften Werkzeugen auswertbar, als die Qualitätsmerkmale Performanz, Benutzbarkeit und Sicherheit. Den Schwerpunkt der mit den untersuchten Werkzeugen messbaren Software Qualitätsmaße bilden die Kategorien der generischen, internen und statischen Maße mit der Charakteristik HR. Software Qualitätsmaße die schlechter zur Messung geeignet sind weisen spezifische, UD, hybride und dynamische Eigenschaften auf. Die Hypothese  $H_0$  ist auf Grundlage der Ergebnisse aus  $T_1$ ,  $T_2$  und  $T_3$  beantwortet.

## 5 Zusammenfassung der Ergebnisse

Im Rahmen der Bachelorthesis wurde untersucht, in welchem Umfang der Begriff Software Qualität auf Grundlage der ISO/IEC 25000 Normenreihe durch Werkzeuge operationalisierbar ist und wie sich die Werkzeuge und Software Qualitätsmaße des SQuaRE Frameworks klassifizieren lassen. Als Leitfaden der Untersuchung wurde in Kapitel 3 eine Hypothese  $H_0$  aufgestellt, die in die drei Teilhypothesen  $T_1$ ,  $T_2$ ,  $T_3$  unterteilt wurde. Jede Teilhypothese thematisiert unterschiedliche Bereiche, die zur Beantwortung von  $H_0$  beitragen. Die Teilhypothesen bauen dabei partiell aufeinander auf. Die für das Verständnis der Bachelorthesis notwendigen Grundlagen über den allgemeinen Qualitätsbegriff, den Begriff Software Qualität, Möglichkeiten zur Visualisierung, relevante Normen, Qualitätsmodelle und Qualitätsmerkmale wurden in Kapitel 2 ausgeführt. Im Kapitel 2.2.3 wurden die in den Normen definierten Qualitätsmodelle voneinander abgegrenzt und die für die weitere Untersuchung relevanten identifiziert. Das zur Untersuchung eruierte Qualitätsmodell ist das System- und Software Produkt Qualitätsmodell der Norm ISO/IEC 25010 und die damit verbundenen 86 Software Qualitätsmaße der Norm ISO/IEC 25023. Das Qualitätsmodell der Datenqualität wurde von der Untersuchung ausgeschlossen, da dieses mit dem jeweiligen Kontext der Nutzung korreliert. Das Qualitätsmodell der Nutzungsqualität wurde von der Untersuchung ausgeschlossen, da dieses laut Definition der Norm nicht frei von subjektiven Faktoren und abhängig von der Anwendung des Qualitätsmodells der Datenqualität ist. Zur Klassifikation von Werkzeugen wurde in Kapitel 2.4 ein Ansatz aus der Literatur nach BALZERT verwendet. Die Werkzeugklassen kategorisieren sich in dynamische Testwerkzeuge, statische Analysewerkzeuge, formale Verifikationswerkzeuge und modellierende- und analysierende Werkzeuge (vgl. Tab. 3). Die 86 Software Qualitätsmaße des System- und Software Produkt Qualitätsmodells wurden durch sechs verschiedene Ansätze klassifiziert (vgl. Tab. 27).

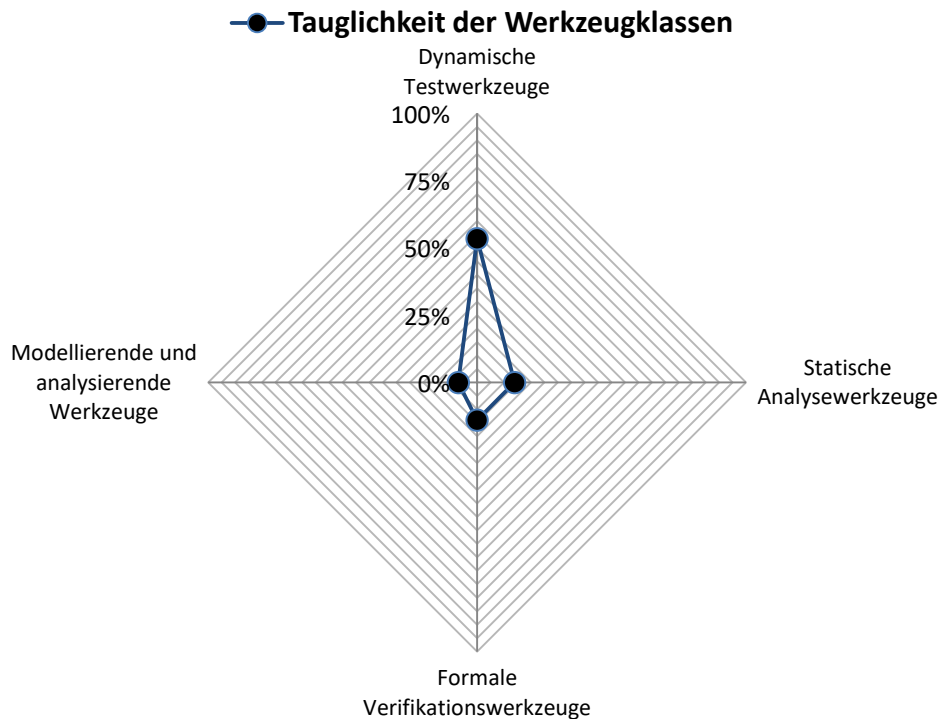
Ansatz der Klassifikation	Ausprägungen
Klassifikation nach BALZERT	Statisch, dynamisch
Klassifikation nach Perspektive des Maßes	Intern, extern, hybrid
Klassifikation nach Validität der Ergebnisse	HR, R, UD
Klassifikation nach zugehörigen Qualitätsmerkmalen	Funktionalität, Effizienz, Kompatibilität, Benutzbarkeit, Zuverlässigkeit, Sicherheit, Wartbarkeit, Portabilität
Klassifikation nach Nutzungsempfehlung	Generisch, spezifisch
Klassifikation nach Eingangsgrößen	Anforderung, Betrieb, Programmierrichtlinien, Kontext und Anwender, Umgebungsbedingungen

Tab. 27 – Zusammenfassung der Klassifizierungen von Software Qualitätsmaßen<sup>86</sup>

Die Teilhypothese  $T_1$  wurde durch die Zuordnung der Software Qualitätsmaße zu eruierten Kategorien beantwortet. In Kapitel 4.2 wurden exemplarisch Werkzeuge verschiedener Werkzeugtypen auf Grundlage des Ansatzes zur Klassifizierung aus der Literatur nach LIGGESMEYER vorgestellt und untersucht, ob die Funktionalitäten der Werkzeuge sich zur Bestimmung der Parameter der Metriken im System- und Software Produkt Qualitätsmodell eignen. Die bei der Analyse in  $T_2$  betrachteten Werkzeuge und deren unterstützte Programmiersprachen sind in Tab. 30 im Anhang C aufgeführt.

<sup>86</sup> Eigene Darstellung

Die Werkzeuge Security Fortify on Demand und SonarQube unterstützen die größte Anzahl von Programmiersprachen. Die Nutzung von Silk Test, Code Inspector, CodeSurfer, JProfiler, Checkstyle und dem Jessie Plugin ist auf wenige Programmiersprachen beschränkt. ALM, Open Markov, Xfmea und Isograph sind unabhängig von Programmiersprachen. Im Rahmen der Untersuchung der  $T_2$  wurde durch die Analyse der Funktionalitäten festgestellt, dass einige Werkzeuge mehreren Werkzeugtypen einer Werkzeugklasse zugeordnet werden können. Eine Zusammenfassung der Ergebnisse der Eignung von Werkzeugtypen nach LIGGESMEYER zur Operationalisierung von Qualitätsmerkmalen des System- und Software Produkt Qualitätsmodells ist in Abb. 59 dargestellt.



**Abb. 59 – Zusammenfassung der Tauglichkeit der Werkzeugklassen<sup>87</sup>**

Der Grafik ist zu entnehmen, dass von der Gruppe der dynamischen Testwerkzeuge die numerisch größte Anzahl an Parametern der Metriken des untersuchten Qualitätsmodells erhoben oder direkt bestimmt werden können. Aufgrund der geringen Anzahl von internen Software Qualitätsmaßen im Qualitätsmodell eignen sich statische Analysewerkzeuge nur geringfügig zur Operationalisierung der definierten Qualitätsmerkmale. Formale Verifikationswerkzeuge, modellierende- und analysierende Werkzeuge eignen sich ebenfalls nur eingeschränkt zur Messung der Software Qualitätsmaße des System- und Software Produkt Qualitätsmodells. Die Teilhypothese  $T_2$  wurde anhand der Bewertung der individuellen Werkzeugklassen zur Unterstützung der Messung von Software Qualitätsmaßen beantwortet. Herauszustellen ist, dass die Eignung der Messung von Software Qualitätsmaßen in  $T_2$  aus den kumulierten Bewertungen mehrerer Werkzeugtypen der untersuchten Werkzeugklassen generiert und dennoch eine hohe Anzahl der Software Qualitätsmaße als nicht als quantifizierbar eingestuft wurde. In Kapitel 4.3 wurden die drei Werkzeuge, die auf Grundlage der Ergebnisse von  $T_2$  als qualifizierteste evaluiert wurden, mit den Klassifizierungen aus  $T_1$  kombiniert. Das hiermit verfolgte Ziel bestand darin Aussagen über die Art der als messbar identifizierten Software Qualitätsmaße abzuleiten. Die Zusammenfassung der Ergebnisse der  $T_3$  sind in Abb. 60 für

<sup>87</sup> Eigene Darstellung nach Tab. 33

einzelne Qualitätsmerkmale des untersuchten Qualitätsmodells und Abb. 61 für die restlichen Klassifizierungen aus Tab. 27, mit Ausnahme die der externen Abhängigkeiten, dargestellt.

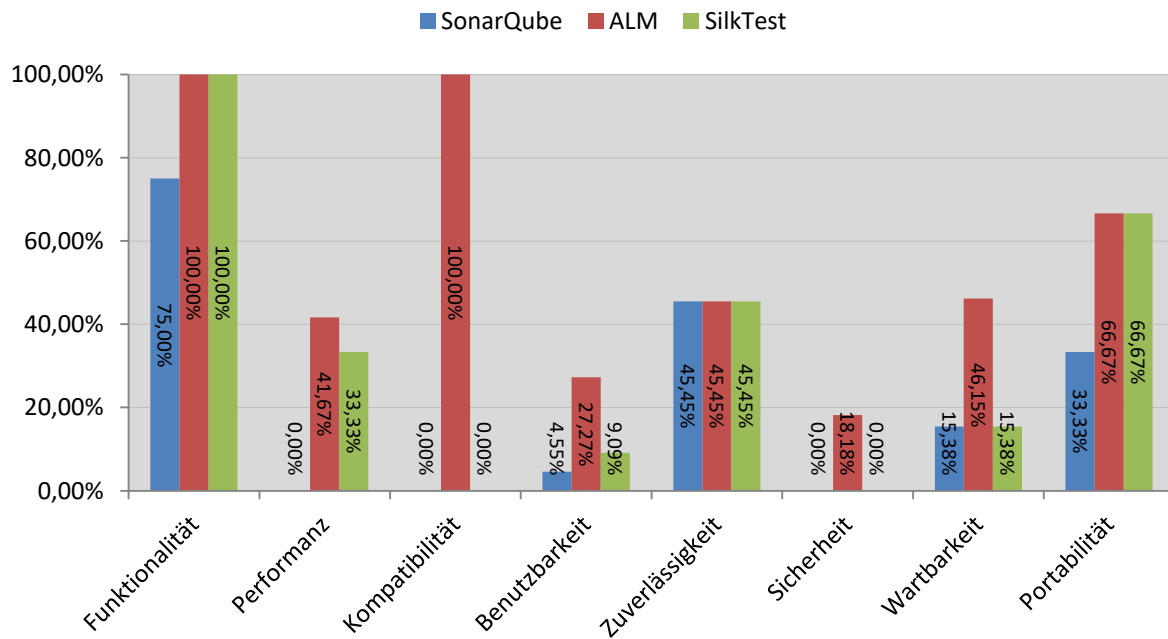


Abb. 60 – Übersicht qualifizierteste Werkzeuge zur Qualitätsmerkmal Operationalisierung<sup>88</sup>

Die Abb. 60 zeigt, dass die Qualitätsmerkmale der Funktionalität, Kompatibilität, Zuverlässigkeit, Wartbarkeit und Portabilität mit einer jeweiligen Abdeckung von über 45 Prozent der Software Qualitätsmaße sich besser zur Messung mit den Werkzeugen ALM, SonarQube und Silk Test eignen, als die Qualitätsmerkmale der Effizienz, Benutzbarkeit und Sicherheit, die einen Wert der Tauglichkeit von weniger als 45 Prozent aufweisen.

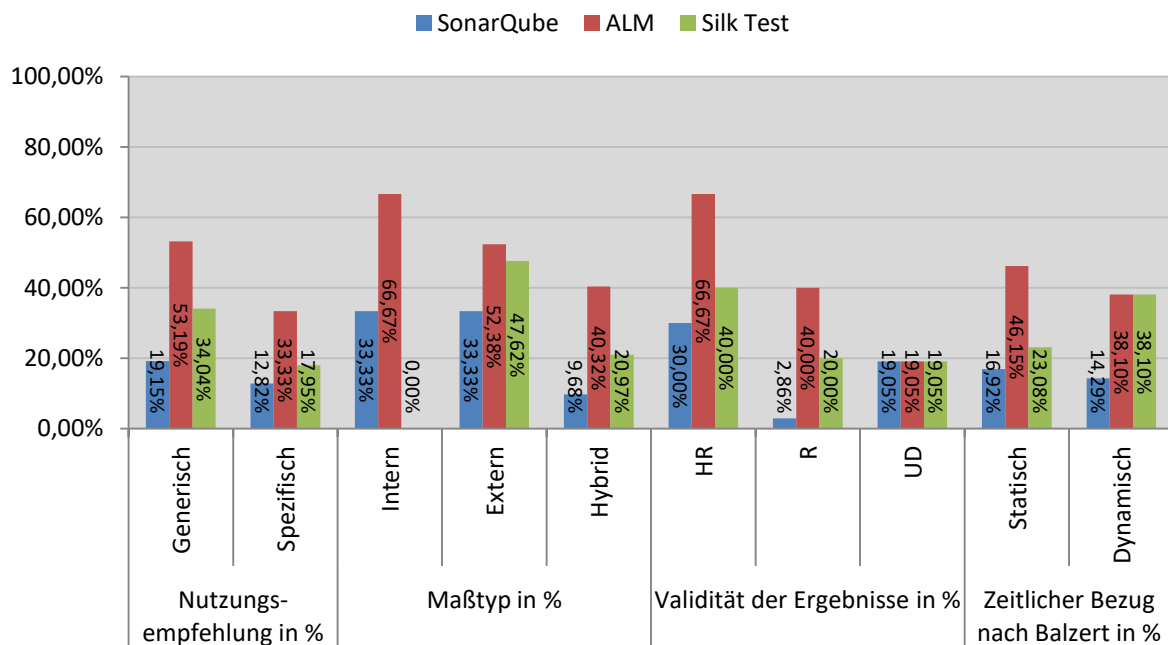


Abb. 61 – Übersicht qualifizierteste Werkzeuge zur Abdeckung der T<sub>1</sub>-Klassifizierungen<sup>89</sup>

<sup>88</sup> Eigene Darstellung nach Tab. 32

<sup>89</sup> Eigene Darstellung nach Tab. 32



Das Diagramm in Abb. 61 beschreibt die Art der mit den Werkzeugen ALM, SonarQube und Silk Test bestimmbaren Software Qualitätsmaße und in welchem Umfang diese erhoben werden können. Die Software Qualitätsmaße die am besten quantifizierbar sind und den maximalen Wert der jeweiligen Klassifikation aufweisen, entsprechen den Kategorien HR, interne, externe, statische, spezifische und generische Software Qualitätsmaße. Die Kategorien UD, dynamische, spezifische und hybride Software Qualitätsmaße beinhalten die meisten Software Qualitätsmaße, die nicht von den untersuchten Werkzeugen messbar sind. Die Teilhypothese  $T_3$  wurde anhand der Gegenüberstellung von Werkzeugen und Werkzeugklassen beantwortet. Die Auswertung der Gesamthypothese  $H_0$  basiert auf den Ergebnissen der  $T_1$ ,  $T_2$  und  $T_3$ . Es wurden Software Qualitätsmaße und Werkzeuge nach verschiedenen Ansätzen klassifiziert und Aussagen abgeleitet, in welchem Umfang die Werkzeuge sich zur Messung der einzelnen Software Qualitätsmaß Kategorien eignen. Das Werkzeug mit der höchsten Abdeckung von 44,19 Prozent der definierten Software Qualitätsmaße im System- und Software Produkt Qualitätsmodell ist das dynamische Testwerkzeug ALM des Unternehmens HP. Es wurde nachgewiesen, dass sich kein Werkzeug vollständig zur Bestimmung der Software Qualitätsmaße des untersuchten Qualitätsmodells eignet. Eine Zusammenfassung der Resultate aller untersuchten Werkzeuge ist in Abb. 62 aufgezeigt.

—●— Abdeckung Software Qualitätsmaße durch untersuchte Werkzeuge

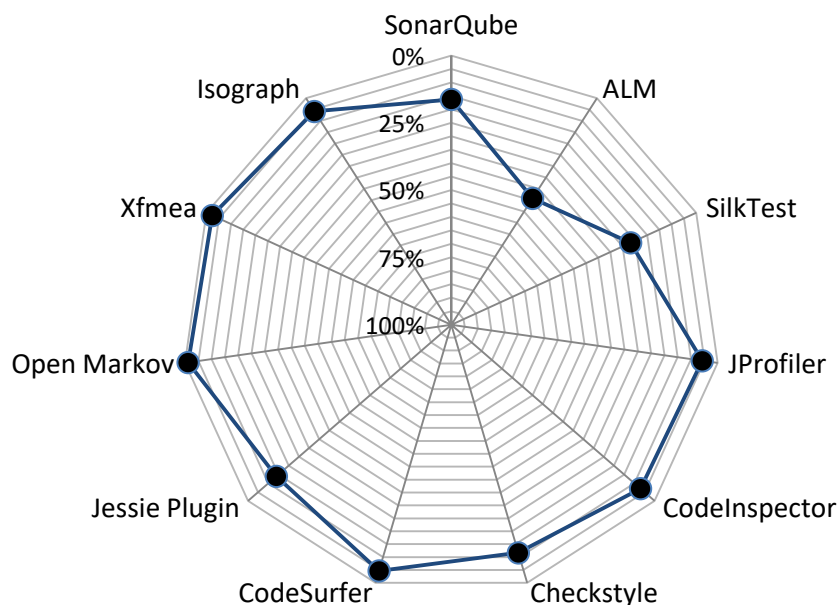


Abb. 62 – Abdeckung Software Qualitätsmaße durch untersuchte Werkzeuge<sup>90</sup>

Die Visualisierungsform entspricht einem Kiviat-Diagramm. Die Achsen des Diagramms wurden invertiert. Ein Werkzeug ist zur Messung qualitativ hochwertiger, je geringer der Abstand des Messpunktes auf der jeweiligen Achse zum Mittelpunkt des Diagramms ist. Die Grafik zeigt, dass die Werkzeuge Isograph, Xfmea, Open Markov, Jessie, CodeSurfer, Checkstyle, CodeInspector und JProfiler nur einen geringen Teil der Software Qualitätsmaße des System- und Software Produkt Qualitätsmodells abdecken. Die Werkzeuge SonarQube, Silk Test und ALM weisen einen höheren Grad der Tauglichkeit zur Bestimmung der Ausprägung von Qualitätsmerkmalen einer Software auf. Die Hypothese  $H_0$  wurde dadurch beantwortet, dass Werkzeuge Teilbereiche von Qualitätsmerkmalen operationalisieren, eine Bewertung anhand der Eignung der Werkzeuge vorgenommen und Klassifizierungen von Software Qualitätsmaßen zugeordnet wurden. Ein Schwellwert ab wann ein Werkzeug als nutzbar einge-

<sup>90</sup> Eigene Darstellung nach Tab. 32

stuft wird, wurde im Rahmen der Untersuchung nicht festgelegt, da es sich hierbei um keinen allgemeingültigen Grenzwert handelt, sondern dieser situationsabhängig ist und subjektivem Einfluss unterliegt.

		Werkzeugklassen				Nicht abgedeckte Software Qualitätsmaße
		Dynamische Testwerkzeuge	Statische Analysewerkzeuge	Formale Verifikationswerkzeuge	Modellierende und analysierende Werkzeuge	
Qualitätsmerkmale	Funktionalität	4	-	3	-	0
	Performanz	6	-	-	1	5
	Kompatibilität	4	-	-	-	0
	Benutzbarkeit	1 7	-	2	1	14
	Zuverlässigkeit	1 5	2	2	3	3
	Sicherheit	3	1	1	-	8
	Wartbarkeit	1 6	5	1	1	5
	Portabilität	8	4	3	-	1

Tab. 28 – Abdeckung der Qualitätsmerkmale durch Werkzeugklassen<sup>91</sup>

In Tab. 28 sind die identifizierten Forschungslücken in den Ergebnissen dieser Bachelorthesis dargestellt. Die aus den analysierten Werkzeugklassen resultierenden Forschungslücken aufgrund von nur partiell geeigneten oder fehlenden Werkzeugen für eine Gegenüberstellung von Funktionalitäten und Metriken, sind rot abgebildet. Dies trifft für die dynamischen Testwerkzeuge des Werkzeugtyps Stresstestwerkzeuge, für formale Verifikationswerkzeuge und statische Analysewerkzeuge des Typs Datenflussanomalieanalyse-Werkzeuge und Werkzeuge zur Erzeugung von Grafiken und Tabellen zu. Die Anzahl der direkt mit den Werkzeugklassen bestimmbaren Maße sind grün und die aus den Daten im Werkzeug ableitbaren Software Qualitätsmaße sind gelb hinterlegt. Zusammengefasst kann festgehalten werden, dass die Qualitätsmerkmale des System- und Software Produkt Qualitätsmodells nicht von einem einzelnen Werkzeug gemessen werden können. Die Werkzeugtypen weisen untereinander stark abweichende Eignungsgrade zur Operationalisierung des Software Qualitätsbegriffs der Norm ISO/IEC 25000 auf. Die Anzahl der mit den untersuchten Werkzeugen nicht messbaren Software Qualitätsmaße ist für die Qualitätsmerkmale Benutzbarkeit, Sicherheit, Wartbarkeit und Performanz signifikant höher als die der Qualitätsmerkmale Zuverlässigkeit, Portabilität, Funktionalität und Kompatibilität.

<sup>91</sup> Eigene Darstellung nach Tab. 32

## 6 Weiterführender Forschungs- und Gestaltungsbedarf

Die Ergebnisse dieser Bachelorthesis können als Basis für weitere empirische Arbeiten zur Evaluation von Qualitätsmodellen, Software Qualitätsmaßen und Werkzeugen genutzt werden. In dieser Arbeit wurde der Begriff Software Qualität auf Grundlage der Qualitätsmodelle der Normenreihe ISO/IEC 25000 untersucht. Von den drei definierten Qualitätsmodellen wurde ausschließlich das System- und Software Produkt Qualitätsmodell analysiert, woraus weiterer Forschungsbedarf aus der Betrachtung von Software Qualitätsmaßen der Qualitätsmodelle der Daten- und Nutzungsqualität resultiert. Ebenfalls können Qualitätsmodelle, die nicht dem SQuaRE Framework zugeordnet sind daraufhin untersucht werden, ob diese zur Operationalisierung von Qualitätsmerkmalen oder als Grundlage für ein Software Gütesiegel zur Zertifizierung und Sicherung eines Mindestmaßes an Qualitätseigenschaften von Software qualifizierter sind. Ist kein geeignetes Qualitätsmodell als Basis für ein Software Gütesiegel identifizierbar, kann ein neues Qualitätsmodell entwickelt werden, welches besser als Fundament nutzbar ist. Zusätzlicher Gestaltungsbedarf bezogen auf die Klassifikationsansätze in dieser Thesis besteht darin, die Klassifizierungen von Software Qualitätsmaßen aus Kapitel 4.1 und Werkzeugen aus Abschnitt 2.4 durch Ansätze der Literatur auszuweiten, zu evaluieren und zu verifizieren, ob diese Einfluss auf die im Rahmen der Untersuchung generierten Ergebnisse besitzen. Ebenfalls können die entstandenen Forschungslücken für Stresstestwerkzeuge, formale Verifikationswerkzeuge, Datenflussanomalien Werkzeuge und Werkzeuge zur Erzeugung von Grafiken und Tabellen, für die nur partiell qualifizierte oder keine Werkzeuge identifiziert wurden die der Beschreibung der Werkzeugtypen und deren Funktionalitäten von LIGGESMEYER entsprechen, weiter untersucht werden. Der Klassifizierungsansatz nach BALZERT wurde exemplarisch anhand statischer und dynamischer Eigenschaften der einzelnen Software Qualitätsmaße des System- und Software Produkt Qualitätsmodells vorgenommen. Die Gruppierung nach objektiven und subjektiven, absoluten und relativen, expliziten und abgeleiteten, vorhersagenden und erklärenden, prozess- und produktorientierten und globalen und speziellen Kriterien wurden in Kapitel 2.2.5 nur genannt und nicht praktisch durchgeführt. Durch die Aquirierung und Anwendung abweichender Methoden zur Untersuchung von Software Qualitätsmaßen, können die Resultate dieser Bachelorthesis validiert werden. Die Untersuchung von Abhängigkeiten der Software Qualitätsmaße des System- und Software Produkt Qualitätsmodells von spezifischen Eingangsgrößen wurde mit einer quantitativen Analyse durchgeführt, indem die Definitionen von Metriken auf die Existenz bestimmter Schlagworte geprüft wurden. Diese Analyse kann in weiteren Arbeiten qualitativ belegt werden. Die Stichproben der Werkzeuge aus Kapitel 4.2 können quantitativ erweitert werden, da kein Werkzeug identifiziert wurde, welches alle Software Qualitätsmaße des System- und Software Produkt Qualitätsmodells bestimmt oder eine Unterstützungsfunktion zur Ableitung der Qualitätsmaßausprägungen besitzt. Daraus resultiert weiterer Forschungsbedarf in der Ermittlung eines vollständig qualifizierten Werkzeugs. Ebenfalls kann ein Werkzeug als Eigenentwicklung erstellt werden, welches die Metriken des SQuaRE Frameworks abbildet und deren Auswertung automatisiert. Dieses kann sukzessive mit weiteren Funktionalitäten erweitert werden bis die Qualitätsmerkmale des zugrundeliegenden Qualitätsmodells vollständig operationalisiert sind.

# Literaturverzeichnis

- [1] KELVIN, William Thomson: *Popular lectures and addresses*. URL <https://archive.org/stream/popularlecturesa01kelvuoft#page/72/mode/2up> – Überprüfungsdatum 2017-04-26
- [2] LUDEWIG, Jochen ; LICHTER, Horst: *Software Engineering : Grundlagen, Menschen, Prozesse, Techniken*. 3., korrigierte Aufl. Heidelberg : dpunkt.verl., 2013
- [3] SNEED, Harry M. ; SEIDL, Richard ; BAUMGARTNER, Manfred: *Software in Zahlen : Die Vermessung von Applikationen*. München : Hanser, 2010
- [4] HOFFMANN, Dirk W.: *Software-Qualität*. 2., aktualisierte u. korr. Aufl. 2013. Berlin, Heidelberg : Springer, 2013 (eXamen.press)
- [5] SCHNEIDER, Kurt: *Abenteuer Softwarequalität: Grundlagen und Verfahren für Qualitätssicherung und Qualitätsmanagement*. 2. Aufl. s.l. : dpunkt.verlag, 2012
- [6] WINTER, Mario ; EKSSIR-MONFARED, Mohsen ; SNEED, Harry M. ; SEIDL, Richard ; BORNER, Lars: *Der Integrationstest : Von Entwurf und Architektur zur Komponenten- und Systemintegration*. München : Hanser, 2013
- [7] BALZERT, Helmut ; EBERT, Christof: *Softwaremanagement*. 2. Aufl. Heidelberg : Spektrum Akad. Verl., 2008 (Lehrbücher der Informatik / Helmut Balzert ; 3)
- [8] SOMMERVILLE, Ian: *Software engineering*. 9., aktualisierte Auflage. München, Harlow, Amsterdam : Pearson, 2012 (it - Informatik)
- [9] BALZERT, Helmut: *Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Heidelberg, Berlin, Oxford : Spektrum Akad. Verl., 1998 (Lehrbuch der Software-Technik ; 2)
- [10] EUL, Marcus: *Qualitätsmanagementsystem für Geschäftsfeldmodelle : Projekttechniken als Instrumente zur Qualitätsplanung und -lenkung*. Gabler Edition Wissenschaft. Wiesbaden, s.l. : Deutscher Universitätsverlag, 1996 (Gabler Edition Wissenschaft)
- [11] ISO/IEC 25000. 2014-03-15. *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE*
- [12] JAHNEL, Lars: *Evaluation und Anwendung von Qualitätsmaßen nach ISO 25000 Standard auf Schnittstellen in einer bestehenden IT-Landschaft am Beispiel der operational services GmbH & Co. KG*. Leipzig, Hochschule für Telekommunikation (FH), Fakultät Informations- und Kommunikationstechnik. Bachelorarbeit. 2016
- [13] SIMBÜRGER, Markus ; SCHÖN, Eckhardt: *Entwicklung eines Konzepts zur Bewertung der Softwarequalität bei agiler Softwareentwicklung*, 2016
- [14] ENRICO KRAUSE: *Untersuchung und Klassifikation von Software Qualitätsmaßen und entsprechenden Werkzeugen*. Leipzig, Hochschule für Telekommunikation (FH). Bachelorarbeit. 2017
- [15] GARVIN, David A.: *What does Product Quality Really Mean?* In: *Sloan Management Review* (1984), S. 25–43
- [16] DIN EN ISO 9000. 2015. *Qualitätsmanagementsysteme – Grundlagen und Begriffe (ISO 9000:2015); Deutsche und Englische Fassung EN ISO 9000:2015*
- [17] DIN 55350. 2005. *Begriffe zum Qualitätsmanagement – Teil 11: Ergänzung zu DIN EN ISO 9000:2005*

- [18] TIEMEYER, Ernst: *Handbuch IT-Projektmanagement : Vorgehensmodelle, Management-instrumente, Good Practices*. 2., überarb. und erw. Aufl. München : Hanser, 2014
- [19] DEMARCO, Tom: *Controlling software projects : Management, measurement & estimation*. Englewood Cliffs, N.J. : Yourdon, 1982 (Yourdon computing series)
- [20] BALZERT, Helmut ; LIGGESMEYER, Peter ; SCHWICHTENBERG, Holger: *Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb*. 3. Aufl. Heidelberg : Spektrum Akademischer Verlag, 2011 (Lehrbücher der Informatik)
- [21] ISO/IEC 25010. 2011-03-01. *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*
- [22] ISO/IEC 25012. 2008-12-15. *Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Data quality model*
- [23] ISO/IEC 25022. 2016-06-15. *Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Measurement of quality in use*
- [24] ISO/IEC 25023. 2016-06-15. *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality*
- [25] LIGGESMEYER, Peter: *Software-Qualität : Testen, Analysieren und Verifizieren von Software*. 2. Aufl. s.l. : Spektrum Akademischer Verlag, 2009
- [26] EBERT, Christof ; DUMKE, Reiner: *Software Measurement : Establish ; Extract ; Evaluate ; Execute*. 1. Aufl. s.l. : Springer-Verlag, 2007
- [27] ARTUR WAGNER: *Codeanalyse : Einleitung und Einführung in das Thema Codeanalyse mit dem Schwerpunkt statische Codeanalyse*. URL [https://www.researchgate.net/publication/268419707\\_Codeanalyse\\_Einleitung\\_und\\_Einfuehrung\\_in\\_das\\_Thema\\_Codeanalyse\\_mit\\_dem\\_Schwerpunkt\\_statische\\_Codeanalyse](https://www.researchgate.net/publication/268419707_Codeanalyse_Einleitung_und_Einfuehrung_in_das_Thema_Codeanalyse_mit_dem_Schwerpunkt_statische_Codeanalyse). – Aktualisierungsdatum: 2006-09-27 – Überprüfungsdatum 2017-05-22
- [28] ANON.: *Metric Definitions*. URL <https://docs.sonarqube.org/display/SONAR/Metric+Definitions> – Überprüfungsdatum 2017-05-19
- [29] ANON.: *Multi-Language*. URL <https://www.sonarqube.org/features/multi-languages/> – Überprüfungsdatum 2017-06-12
- [30] ANON.: *Documentation*. URL <https://docs.sonarqube.org/display/SONAR/Documentation/> – Überprüfungsdatum 2017-07-10
- [31] SIMON BRANDHOF: *Plugin Version Matrix*. URL <https://docs.sonarqube.org/display/PLUG/Plugin+Version+Matrix>. – Aktualisierungsdatum: 2017-01-16 – Überprüfungsdatum 2017-07-10
- [32] ANN CAMPBELL: *Code Viewer*. URL <https://docs.sonarqube.org/display/SONAR/Code+Viewer> – Überprüfungsdatum 2017-05-19
- [33] ANN CAMPBELL: *Seeing Coverage*. URL <https://docs.sonarqube.org/display/SONAR/Seeing+Coverage> – Überprüfungsdatum 2017-05-19

- [34] ANON.: *HP Application Lifecycle Management User Guide : Softwareversion: 12.50*. URL [http://alm-help.saas.hpe.com/de/12.53/online\\_help/Content/alm\\_ug.htm](http://alm-help.saas.hpe.com/de/12.53/online_help/Content/alm_ug.htm) – Überprüfungsdatum 2017-05-20
- [35] ANON.: *Feature comparison guide for HP ALM 12.00*. URL [https://softwaresupport.hpe.com/web/softwaresupport/document/-/facetsearch/attachment/KM01178723?fileName=ALM\\_12\\_COMPARISON\\_GUIDE.pdf](https://softwaresupport.hpe.com/web/softwaresupport/document/-/facetsearch/attachment/KM01178723?fileName=ALM_12_COMPARISON_GUIDE.pdf) – Aktualisierungsdatum: 2014-09-24 – Überprüfungsdatum 2017-05-31
- [36] ANON.: *Funktionen*. URL <https://www.microfocus.com/de-de/products/silk-portfolio/silk-test/features/> – Überprüfungsdatum 2017-05-20
- [37] ANON.: *Systemanforderungen*. URL <https://www.microfocus.com/de-de/products/silk-portfolio/silk-test/system-requirements/> – Überprüfungsdatum 2017-07-03
- [38] ANON.: *Micro Focus Online Documentation*. URL [http://documentation.microfocus.com/help/index.jsp?topic=%2Fcom.borland.silktest.wordkbench.doc%2FSILKTEST-4C899BC5-SILKTESTPRODUCTSUITE-CON.html&cp=8\\_3\\_1\\_2\\_1\\_2](http://documentation.microfocus.com/help/index.jsp?topic=%2Fcom.borland.silktest.wordkbench.doc%2FSILKTEST-4C899BC5-SILKTESTPRODUCTSUITE-CON.html&cp=8_3_1_2_1_2) – Aktualisierungsdatum: 2017-05-20
- [39] ANON.: *Silk Test 17.5*. Data Sheet. URL [https://www.microfocus.com/media/data-sheet/silk\\_test\\_ds.pdf](https://www.microfocus.com/media/data-sheet/silk_test_ds.pdf) – Überprüfungsdatum 2017-06-01
- [40] ANON.: *JProfiler Manual*. URL <https://resources.ej-technologies.com/jprofiler/help/doc/help.pdf> – Aktualisierungsdatum: 2017 – Überprüfungsdatum 2017-05-20
- [41] ANON.: *Features : IDE's*. URL <https://www.ej-technologies.com/products/jprofiler/java-profiler-IDE.html> – Überprüfungsdatum 2017-05-20
- [42] ANON.: *HPE Security Fortify on Demand : Application security as a service*. URL <https://www.hpe.com/h20195/V2/GetPDF.aspx/4AA4-0664ENW.pdf> – Aktualisierungsdatum: 11.2016 – Überprüfungsdatum 2017-06-03
- [43] ANON.: *Fortify Taxonomy: Software Security Errors*. URL <https://vulncat.hpefod.com/en> – Überprüfungsdatum 2017-06-03
- [44] ANON.: *Weaknesses*. URL <https://vulncat.hpefod.com/en/weakness> – Überprüfungsdatum 2017-07-03
- [45] ANON.: *Benutzerhandbuch : Codeinspector@HfTL*. URL <http://gitlab.infolabs.test/DAI15/SWECODEinspector> – Aktualisierungsdatum: 2016-08-14 – Überprüfungsdatum 2017-06-04
- [46] ANON.: *Checkstyle Overview*. URL <http://checkstyle.sourceforge.net/index.html> – Aktualisierungsdatum: 2017-04-30 – Überprüfungsdatum 2017-05-20
- [47] ANON.: *Checkstyle Checks*. URL <http://checkstyle.sourceforge.net/checks.html> – Aktualisierungsdatum: 2017-04-30 – Überprüfungsdatum 2017-05-20
- [48] ANON.: *Running*. URL <http://checkstyle.sourceforge.net/running.html> – Aktualisierungsdatum: 2017-07-01 – Überprüfungsdatum 2017-07-03
- [49] ANON.: *Checkstyle Results*. URL <http://checkstyle.sourceforge.net/reports/google-style/guava/> – Aktualisierungsdatum: 2017-05-28 – Überprüfungsdatum 2017-06-27
- [50] ANON.: *CodeSurfer*. URL <https://www.grammatech.com/products/codesurfer> – Überprüfungsdatum 2017-05-21
- [51] BOEHM: *Software Engineering*. In: *IEEE Transactions on Computers* C-25 (1976), Nr. 12, S. 1226–1241

- [52] CLAUDE MARCHÉ, Yannick Moy: *The Jessie plugin for Deductive Verification in Frama-C : Tutorial and Reference Manual*. Version 2.35. URL <http://krakatoa.lri.fr/jessie.pdf>. – Aktualisierungsdatum: 2015-03-25 – Überprüfungsdatum 2017-06-19
- [53] CLAUDE MARCHÉ: *The Krakatoa Verification Tool for JAVA programs : Tutorial and Reference Manual*. URL <http://krakatoa.lri.fr/krakatoa.pdf>. – Aktualisierungsdatum: 2016-07-22 – Überprüfungsdatum 2017-06-17
- [54] ANON.: *OPENMARKOV TUTORIAL*. URL <http://www.openmarkov.org/docs/tutorial/openmarkov-tutorial.pdf>. – Aktualisierungsdatum: 2016-01-14 – Überprüfungsdatum 2017-05-21
- [55] ANON.: *Frequently Asked Questions*. URL <http://www.reliasoft.com/xfmea/faq.htm> – Überprüfungsdatum 2017-07-10
- [56] ANON.: *Xfmea Software Capability Highlights*. URL <http://www.reliasoft.com/xfmea/capabilities.htm> – Überprüfungsdatum 2017-05-26
- [57] ANON.: *Xfmea Software Features*. URL <http://www.reliasoft.com/xfmea/features1.htm>. – Aktualisierungsdatum: 2017 – Überprüfungsdatum 2017-05-21
- [58] ANON.: *Reliability Workbench*. URL <https://www.isograph.com/software/reliability-workbench/> – Überprüfungsdatum 2017-07-10
- [59] ANON.: *FaultTree+ in Reliability Workbench : Quick, Accurate Fault Tree, Event Tree and Markov Analysis*. URL <https://www.isograph.com/brochures/faulttree-reliability-workbench/> – Überprüfungsdatum 2017-05-21
- [60] ANON.: *Content*. URL [http://checkstyle.sourceforge.net/config\\_annotation.html](http://checkstyle.sourceforge.net/config_annotation.html). – Aktualisierungsdatum: 2017-07-01 – Überprüfungsdatum 2017-07-03
- [61] ANON.: *Blocks*. URL [http://checkstyle.sourceforge.net/config\\_blocks.html](http://checkstyle.sourceforge.net/config_blocks.html). – Aktualisierungsdatum: 2017-07-01 – Überprüfungsdatum 2017-07-03
- [62] ANON.: *Design*. URL [http://checkstyle.sourceforge.net/config\\_design.html](http://checkstyle.sourceforge.net/config_design.html). – Aktualisierungsdatum: 2017-07-01 – Überprüfungsdatum 2017-07-03
- [63] ANON.: *Coding*. URL [http://checkstyle.sourceforge.net/config\\_coding.html](http://checkstyle.sourceforge.net/config_coding.html). – Aktualisierungsdatum: 2017-07-01 – Überprüfungsdatum 2017-07-03
- [64] ANON.: *Header*. URL [http://checkstyle.sourceforge.net/config\\_header.html](http://checkstyle.sourceforge.net/config_header.html). – Aktualisierungsdatum: 2017-07-01 – Überprüfungsdatum 2017-07-03
- [65] ANON.: *Imports*. URL [http://checkstyle.sourceforge.net/config\\_imports.html](http://checkstyle.sourceforge.net/config_imports.html). – Aktualisierungsdatum: 2017-07-01 – Überprüfungsdatum 2017-07-03
- [66] ANON.: *Comments*. URL [http://checkstyle.sourceforge.net/config\\_javadoc.html](http://checkstyle.sourceforge.net/config_javadoc.html). – Aktualisierungsdatum: 2017-07-01 – Überprüfungsdatum 2017-07-03
- [67] ANON.: *Metrics*. URL [http://checkstyle.sourceforge.net/config\\_metrics.html](http://checkstyle.sourceforge.net/config_metrics.html). – Aktualisierungsdatum: 2017-07-01 – Überprüfungsdatum 2017-07-03
- [68] ANON.: *Misc*. URL [http://checkstyle.sourceforge.net/config\\_misc.html](http://checkstyle.sourceforge.net/config_misc.html). – Aktualisierungsdatum: 2017-07-01 – Überprüfungsdatum 2017-07-03
- [69] ANON.: *Modifier*. URL [http://checkstyle.sourceforge.net/config\\_modifier.html](http://checkstyle.sourceforge.net/config_modifier.html). – Aktualisierungsdatum: 2017-07-01 – Überprüfungsdatum 2017-07-03
- [70] ANON.: *Conventions*. URL [http://checkstyle.sourceforge.net/config\\_naming.html](http://checkstyle.sourceforge.net/config_naming.html). – Aktualisierungsdatum: 2017-07-01 – Überprüfungsdatum 2017-07-03

- [71] ANON.: *Regexp*. URL [http://checkstyle.sourceforge.net/config\\_regexp.html](http://checkstyle.sourceforge.net/config_regexp.html). – Aktualisierungsdatum: 2017-07-01 – Überprüfungsdatum 2017-07-03
- [72] ANON.: *Size violation*. URL [http://checkstyle.sourceforge.net/config\\_sizes.html](http://checkstyle.sourceforge.net/config_sizes.html). – Aktualisierungsdatum: 2017-07-01 – Überprüfungsdatum 2017-07-03
- [73] ANON.: *Whitespace*. URL [http://checkstyle.sourceforge.net/config\\_whitespace.html](http://checkstyle.sourceforge.net/config_whitespace.html). – Aktualisierungsdatum: 2017-07-01 – Überprüfungsdatum 2017-07-03
- [74] ANON.: *Rules*. URL <https://sonarcloud.io/organizations/default/rules> – Überprüfungsdatum 2017-06-12



# Selbständigkeitserklärung

Hiermit erkläre ich, dass die von mir an der Hochschule für Telekommunikation Leipzig (FH) eingereichte Abschlussarbeit zum Thema

## **Evaluation der Zuordnung von Werkzeug- und Softwarequalitätsmaß Klassifizierungen zu Qualitätsmerkmalen der Norm ISO/IEC 25000**

vollkommen selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Abbildungen in dieser Arbeit sind von mir selbst erstellt oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Hochschule/Universität eingereicht worden.

Leipzig, den 04. August 2017

---

Vorname Nachname

# Anlagen

## A Glossar

### A

Apache Ant Task	Programm zur automatisierten Erstellung von installierbaren Software Paketen aus Quelltexten und Bibliotheken.
Application under Test Topology	Verwaltung von Umgebungsparametern für funktionelle und Leistungstests.
Aufrufgraphen	Visualisierung des Kontrollflussgraphen von Subroutinen eines Programms.
Auswirkungsanalyse	Abhängigkeitsanalyse von Statements und Instruktionen.

### B

Baselining	Markierung von Meilensteinen im Lebenszyklus einer Anwendung und den zugehörigen Anforderungen, Tests und Testressourcen.
Bayes'schen Netze	Stochastische Beschreibungsmöglichkeit von unbekannten und zufälligen Einflüssen auf modellierende Zusammenhänge, um wahrscheinlichkeitstheoretische Aussagen über eine Software zu generieren.
Black-Box Prinzip	Beobachtung des äußerlich sichtbaren Verhaltens eines Systems ohne Kenntnis des inneren Aufbaus.
Breakpoint	Markierung einer Stelle im Programmcode an der die Verarbeitung während der Fehlerbereinigung angehalten wird.

### C

Call Stack	Stapelspeicher der zur Laufzeit eines Programms Informationen über aufgerufene Unterprogramme enthält.
Compiler	Programm zur Transformation von Maschinencode in höhere Programmiersprachen und Identifikation von Fehlern.
Continuous Delivery Automation	Bereitstellung von Build-Paketen in dynamischen Testumgebungen.
CPU Profiling	Monitoring und kumulierte Übersichten der CPU Auslastung. Detailansicht für Pakete und Klassen. Sequenz Visualisierung durch integrierte Aufrufgraphen.
Cross-browser Tests	Test der prüft, ob eingebettete Inhalte, unabhängig vom Typ des Web-Browsers, gleiche Ausgabe erzeugen.

### D

Dashboard	Übersichtsfunktion zur Anzeige aufbereiteter Daten.
Datenflussanalyse	Analyse der Variablendeklaration und Initialisierung.
Datenübertragungsfunktion	Übertragung von Daten zwischen Konstruktions-, Design-, und Prozess-FMEAs.
Debugging Test	Schrittweise Ausführung visueller Test zur Identifizierung obsoleter Testfälle, Umgebungsänderungen oder geänderter Testapplikationen. Schrittweise Ausführung von Skripten zur Analyse von Laufzeit Fehlern durch Änderungen an der Testapplikation, Quelltextfehlern oder Umgebungsänderungen.
Default	Standardzustand.
Defect	Nichterfüllung einer Anforderung.
Designverifikationspläne	Entwurfbestätigungsplan.

Design Reviews Based on Failure Mode	Entwicklungsmethode im Erstellungsprozess von Produkten zur Fehlervermeidung.
Diagnostische Inferenz	Auf Grundlage eines Symptoms erfolgt Schlussfolgerung über Diagnose.
<b>E</b>	
Einflussdiagramme	Darstellung von Abhängigkeitsstrukturen mit Entscheidungsknoten bei der bestimmte Ursachen zu Ergebnissen führen und diese beeinflussen.
Ereignisbaum Analyse	Verfahren zur Bestimmung möglicher Auswirkungen von Fehlern einzelner Komponenten oder eines Gesamtsystems.
<b>F</b>	
Fehlerbaumanalyse	Top-down Methode zur Risikoanalyse, Bestimmung von Fehler Auftrittswahrscheinlichkeiten und Auffinden von Ereigniskombinationen, die zu Fehlerereignis im Rahmen der Planung von Systemen führen.
FMEA	Fehlermöglichkeits- und Einflussanalyse. Analytische Methode zur Bewertung von Produktfehlern anhand von Auftrittswahrscheinlichkeiten, Entdeckungswahrscheinlichkeiten und deren Bedeutung auf Basis von Kennzahlen.
FMECA	Fehlermöglichkeits-, Einfluss- und Risikoanalyse. Analytische Methode zur Bewertung von Produktfehlern anhand von Auftrittswahrscheinlichkeiten, Entdeckungswahrscheinlichkeiten, deren Bedeutung und den verbundenen Risiken auf Basis von Risikoprioritätskennzahlen.
Framework	Ordnungsrahmen zur Anordnung von Elementen und ihrer Beziehungen untereinander.
<b>G</b>	
Garbage Collector	Automatische Speicherbereinigung zur Minimierung des Speicherbedarfs zur Laufzeit.
Gemischte Inferenz	Kombination aus interkausaler-, kausaler- und diagnostischer Inferenz.
GMOD/GREF Analyse	Analyse zur Identifikation von nicht lokalen Variablen die von Funktionen genutzt oder von ihnen geändert werden.
<b>H</b>	
Heap Speicher	Dynamische Speicherallokation und Freigabe zusammenhängender Speicherabschnitte eines Programms.
Heap Walker	Erzeugung von Snapshots des Heap Speichers. Detailansicht von einzelnen Objekten, Klassen, Referenzen, Allokationen zur Identifikation von Memory Leaks.
Heuristische Regel	Regel zur Erstellung von wahrscheinlichen Aussagen auf Grundlage von begrenztem Wissen.
<b>I</b>	
Incident	Ungeplante Unterbrechung oder Qualitätsminderung.
Inferenz	Aus einem Regelsystem erzeugte Schlussfolgerung.
Integrated development	Programmierungsumgebung zur Unterstützung von Entwicklungs- und

environment	Routinearbeiten.
Interkausale Inferenz	Auf Grundlage von Ursachen erfolgt Schlussfolgerung eines resultierenden Effekts, der andere Effekte weniger wahrscheinlich macht.
<b>J</b>	
JVM	Teil der Java Laufzeitumgebung, der für die Ausführung des vom Java Compilers erzeugten Java Bytecodes verantwortlich ist.
<b>K</b>	
Kausale Inferenz	Auf Grundlage einer Ursache erfolgt Schlussfolgerung des resultierenden Effekts.
Keyword-driven Tests	Definierte Kombination von einer oder mehreren Aktionen eines Testobjekts im Testdesign. Automatisiertes Testen wiederkehrender Operationen nach einmaliger Test Entwicklung.
Konstruktion FMEAs	Betrachtung systematischer Fehler während Konstruktionsphase eines Produkts.
Kontrollfluss-abhängigkeitsanalyse	Identifikation von Quellcode der Ausführung von Statements beeinflusst.
Kosten-Effizienz Analysen	Modell als Grundlage zur multidimensionalen Entscheidungsfindung.
<b>L</b>	
Lexikalische Analyse	Gruppierung des Quelltextes in zusammengehörige Token verschiedener Kategorien.
<b>M</b>	
Markov Analyse	Modell für probabilistische Inferenzen eines Systemzustandes, wenn sichere Informationen vorliegen und Zusammenhänge garantiert eintreten.
Memory Leak	Speicherverwaltungsfehler durch fehlende Freigaben von Speicherbereichen.
Memory Profiling	Anzeige Speicher Last und zugehörige Detailinformationen wieviel Speicher von Methoden allokiert wird. Detailinformationen zu Paketen, Klassen und Sequenzvisualisierung von Aufrufgraphen.
Mobile testing	Automatisierte Tests für mobile Applikationen unterschiedlicher Betriebssysteme und Hardware.
Modultest	Prüfung funktionaler Einzelteile von Software auf korrekte Funktionalität.
Monitor Profiling	Monitoring, Historisierung und Statistiken von blockierten Elementen in der Java Virtual Machine (JVM), Threads und Klassen.
<b>P</b>	
P-Diagramme	Parameterdiagramm zur Beschreibung von physikalischen und funktionellen Zusammenhängen.
Präprozessor Effekte	Visualisierung welcher Code außerhalb des Build Prozesses kompiliert wurde.
Probabilistisch	Wahrscheinlich.
Probabilistische graphische Modelle	Graphische Modelle zur Erstellung von Schlussfolgerungen bei vorliegenden Unsicherheiten: Bayes'sche Netze, Hidden Markov Modelle, Markov Random Fields.
Probabilistische Inferenzen	Schlussfolgerung der Wahrscheinlichkeit des Eintretens bestimm-

Prozess FMEAs	<p>ter Ereignisse. Gliederung in diagnostische-, kausale-, interkausale- und gemischte Inferenz.</p> <p>Betrachtung von möglichen Schwachstellen im Produktions- oder Leistungsprozess auf Grundlage von Konstruktions-FMEAs.</p>
<b>R</b>	
Referenzmodell	Idealtypisches und abstraktes Modell zum Aufbau von Sachverhalten mit dem Ziel Wiederverwendbarkeit zu gewährleisten und einen Standard zur Beschreibung von Systemen zu schaffen.
Risikobewertungsfunktion	Bewertung von Fehlern und Kalkulation von deren Auswirkungen mittels Risiko Prioritätszahlen (RPZ) oder von Kritikalitätsanalysen.
Risikoprioritätszahl	Kennzahl zur Erstellung einer Rangfolge von Risiken.
Sampling	Methode zur periodischen Messung, die in einem eigenständigen Thread läuft und die JVM alle 5ms pausiert, um den Call Stack der laufenden Threads zu inspizieren. Dabei wird zusätzlicher Byte-code in die zu analysierenden Klassen eingefügt, der Aufrufe der Klasse und der einzelnen Methoden dokumentiert.
<b>S</b>	
Security as a Service	Geschäftsmodell basierend auf der Cloud Technologie, bei dem ein Service Provider sicherheitsbezogene Dienste bereitstellt.
Semantische Analyse	Prüfung der Bedingungen an ein Programm, die über syntaktische Analysen hinausgehen.
Subroutine	Unterprogramm das bestimmte Funktionalitäten bereitstellt und nach Abarbeitung an Stelle des Aufrufs zurückkehrt.
Supply Chain Management	Prozessorientierter Ansatz zur Steuerung und Kontrolle von Teilen der Wertschöpfungskette.
Syntaktische Analyse	Prüfung, ob Quelltext der Syntax (Grammatik) der Quellsprache entspricht.
<b>T</b>	
Taxonomie	Klassifikationsschema.
Testweiterentwicklung	Verbesserung von Skripten durch Wiederverwendung von Test Daten, Verifikations- und Referenzfunktionen. Verbesserung visueller Tests durch Implementierung von Testlogiken, Wiederverwendung von Test Daten und Kommentarfunktion.
Thread	Ausführungsreihenfolge bei Abarbeitung eines Programms in einem eigenständigen Prozess.
Thread Profiling	Monitoring und Analyse von Problemen bei Threadsynchronisation.
Tracking-Funktion	Abschlussverfolgung von Aktionen und Maßnahmen zur Fehlerprävention.
<b>U</b>	
URL	Zeiger zur Identifikation und Lokalisierung von Ressourcen.
<b>V</b>	
Virtual Machine Telemetry	Beobachtung und Messung des internen Status der JVM.
Visuelle Tests	Automatisierte Durchführung von sichtbaren Tests.

Vollständige Programm  
Analyse

Analyse der Interaktionen von Quelldateien oder innerhalb von Binärdateien.

## **W**

Weibull Analyse

Ableitung von mathematischen Modellen aus statistischen Ausfalldaten in Form von Ausfallzeiten und Ausfallarten eines Systems zur Bestimmung und Bewertung von Zuverlässigkeitskenngrößen.

White-Box Prinzip

Methode des Software Tests, der Kenntnis der inneren Struktur einer Software voraussetzt.

Zeiger Analyse

Analyse auf welche Variablen und Prozeduren Zeiger referenzieren.

## **Z**

Zuverlässigkeits-  
Blockdiagramme

Modellierung von Systemen und Prozessen. Durchführung von Analysen der Zuverlässigkeit mit Berechnung von Ausfall-, Stillstandszeiten und der Verfügbarkeit. Berücksichtigen Wiederherstellungsfaktoren, Instandhaltbarkeit und Wartbarkeit von Objekten. Identifikation von Komponenten, die hohe Auswirkungen auf das Gesamtsystem besitzen. Ermöglichen Zuverlässigkeitsoptimierung. Durchsatzberechnung zur Identifikation von Engpässen, Betriebsmittelzuweisung und Lebenszykluskosten.

## B Qualitätsmerkmale, Maße und Metriken der ISO/IEC 25023

Quality characteristic	Quality sub-characteristic	ID	Quality measure name	Type	Recommendation level	Metric	Values	Description
<b>Functional suitability</b>	Functional completeness	FCp-1-G	Functional coverage	Both	HR	$X = 1 - \frac{A}{B}$	A = Number of functions missing B = Number of functions specified	What proportion of the specified functions has been implemented?
	Functional correctness	FCr-1-G	Functional correctness	Both	HR	$X = 1 - \frac{A}{B}$	A = Number of functions that are incorrect B = Number of functions considered	What proportion of functions provides the correct results?
	Functional appropriateness	FAp-1-G	Functional appropriateness of usage objective	Both	HR	$X = 1 - \frac{A}{B}$	A = Number of functions missing or incorrect among those that are required for achieving a specific usage objective B = Number of functions required for achieving a specific usage objective	What proportion of the functions required by the user provides appropriate outcome to achieve a specific usage objective?
		FAp-2-G	Functional appropriateness of system	Both	HR	$X = \sum_{i=1 \text{ to } n} \frac{A_i}{n}$	A <sub>i</sub> = Appropriateness score for usage objective i, that is, the measured value of FAp-1-G for i-th specific usage objective n = Number of usage objectives	What proportion of the functions required by the users to achieve their objectives provides appropriate outcome?
<b>Performance efficiency</b>	Time behaviour	PTb-1-G	Mean response time	Both	HR	$X = \sum_{i=1 \text{ to } n} \frac{(A_i)}{n}$	A <sub>i</sub> = Time taken by the system to respond to a specific user task or system task at i-th measurement n = Number of responses measured	How long is the mean time taken by the system to respond to a user task or system task?
		PTb-2-G	Response time adequacy	Both	R	$X = \frac{A}{B}$	A = Mean response time measured by PTb-1-G B = Target response time specified	How well does the system response time meet the specified target?

Quality characteristic	Quality sub-characteristic	ID	Quality measure name	Type	Recommendation level	Metric	Values	Description
Performance efficiency	Time behaviour	PTb-3-G	Mean turnaround time	Both	R	$X = \sum_{i=1 \text{ to } n} \frac{(B_i - A_i)}{n}$	Ai = Time of starting a job i Bi = Time of completing the job i n = Number of measurements	What is the mean time taken for completion of a job or an asynchronous process?
		PTb-4-G	Turnaround time adequacy	Both	R	$X = \frac{A}{B}$	A = Mean turnaround time measured by PTb-3-G B = Target turnaround time specified	How well does the turnaround time meet the specified targets?
		PTb-5-G	Mean throughput	Both	R	$X = \frac{\left(\frac{A_i}{B_i}\right)}{n}$	Ai = Number of jobs completed during the i-th observation time Bi = i-th observation time period n = Number of observations	What is the mean number of jobs completed per unit time?
	Resource utilization	PRu-1-G	Mean processor utilization	External	HR	$X = \frac{\left(\frac{A_i}{B_i}\right)}{n}$	Ai = Processor time actually used to execute a given set of tasks in observation i Bi = Operation time to perform the tasks in observation i n = Number of observations	How much processor time is used to execute a given set of tasks compared to the operation time?
		PRu-2-G	Mean memory utilization	External	R	$X = \frac{\left(\frac{A_i}{B_i}\right)}{n}$	Ai = Size of memory actually used to perform a given set of tasks for i-th sample processing Bi = Size of memory available to perform the tasks during i-th sample processing n = Number of samples processed	How much of memory is used to execute a given set of tasks compared to the available memory?



Quality characteristic	Quality sub-characteristic	ID	Quality measure name	Type	Recommendation level	Metric	Values	Description
Performance efficiency	Resource utilization	PRu-3-G	Mean I/O devices utilization	External	R	$X = \frac{\sum_{i=1}^n \frac{A_i}{B_i}}{n}$	<p><math>A_i</math> = Duration of I/O device(s) busy time to perform a given set of tasks for i-th observation</p> <p><math>B_i</math> = Duration of I/O operations to perform the tasks for i-th observation</p> <p><math>n</math> = Number of observations</p>	How much of I/O device busy time is used to perform a given set of tasks compared to the I/O operation time?
		PRu-4-S	Bandwidth utilization	External	UD	$X = \frac{A}{B}$	<p><math>A</math> = Bandwidth of actual transmission measured over time to perform a given set of tasks</p> <p><math>B</math> = Bandwidth capacity available to perform a given set of tasks</p>	What proportion of the available bandwidth is utilized to perform a given set of tasks?
	Capacity	PCa-1-G	Transaction processing capacity	Both	R	$X = \frac{A}{B}$	<p><math>A</math> = Number of transactions completed during observation time</p> <p><math>B</math> = Duration of observation</p>	How many transactions can be processed per unit time?
		PCa-2-G	User access capacity	Both	R	$X = \sum_{i=1}^n \frac{A_i}{n}$	<p><math>A_i</math> = Maximum number of users who can simultaneously access the system at i-th observation</p> <p><math>n</math> = Number of observations</p>	How many users can access the system simultaneously at a certain time?
		PCa-3-S	User access increase adequacy	External	UD	$X = \frac{A}{B}$	<p><math>A</math> = Number of users successfully added during observation time</p> <p><math>B</math> = Duration of observation</p>	How many users can be added successfully per unit time?
Compatibility	Co-existence	CCo-1-G	Co-existence with other products	Both	HR	$X = \frac{A}{B}$	<p><math>A</math> = Number of other specified software products with which this product can co-exist</p> <p><math>B</math> = Number of other software products specified to co-exist with this product in the operation environment</p>	What proportion of specified software products can share the environment with this software product without adverse impact on their quality characteristics or functionality?

Quality characteristic	Quality sub-characteristic	ID	Quality measure name	Type	Recommendation level	Metric	Values	Description
Compatibility	Interoperability	CIn-1-G	Data formats exchangeability	Both	HR	$X = \frac{A}{B}$	A = Number of data formats exchangeable with other software or systems B = Number of data formats specified to be exchangeable	What proportion of the specified data formats is exchangeable with other software or systems?
		CIn-2-G	Data exchange protocol sufficiency	Both	R	$X = \frac{A}{B}$	A = Number of data exchange protocols supported B = Number of data exchange protocols specified to be supported	What proportion of the specified data exchange protocols is supported?
		CIn-3-S	External interface adequacy	Both	HR	$X = \frac{A}{B}$	A = Number of external interfaces that are functional B = Number of external interfaces specified	What proportion of the specified external interfaces (interfaces with other software and systems) is functional?
Usability	Appropriateness recognizability	UAp-1-G	Description completeness	Both	HR	$X = \frac{A}{B}$	A = Number of usage scenarios described in the product description or user documents B = Number of usage scenarios of the product	What proportion of usage scenarios is described in the product description or user documents?
		UAp-2-S	Demonstration coverage	Both	UD	$X = \frac{A}{B}$	A = Number of tasks with demonstration features B = Number of tasks that could benefit from demonstration features	What proportion of tasks has demonstration features for users to recognize the appropriateness?
		UAp-3-S	Entry point selfdescriptiveness	Both	UD	$X = \frac{A}{B}$	A = Number of landing pages that explain the purpose of website B = Number of landing pages in a website	What proportion of the commonly used landing pages on a website explains the purpose of the website?

Quality characteristic	Quality sub-characteristic	ID	Quality measure name	Type	Recommendation level	Metric	Values	Description
Usability	Learnability	ULe-1-G	User guidance completeness	Both	HR	$X = \frac{A}{B}$	A = Number of functions described in user documentation and/or help facility as required  B = Number of functions implemented that are required to be documented	What proportion of functions is explained in sufficient detail in user documentation and/or help facility to enable the user to apply the functions?
		ULe-2-S	Entry fields defaults	Both	R	$X = \frac{A}{B}$	A = Number of entry fields whose default values have been automatically filled in during operation  B = Number of entry fields that could have default values	What proportion of entry fields that could have default values are automatically filled with default values?
	Learnability	ULe-3-S	Error message understandability	Both	R	$X = \frac{A}{B}$	A = Number of error messages which state the reason of occurrence and suggest the ways of resolution where this is possible  B = Number of error messages implemented	What proportion of the error messages state the reason why the error occurred and how to resolve it?
		ULe-4-S	Self-explanatory user interface	Both	UD	$X = \frac{A}{B}$	A = Number of information elements and steps that are presented in a way that the user could understand  B = Number of information elements and steps needed to complete common tasks for a first time user	What proportion of information elements and steps presented to the user enable common tasks to be completed by a first-time user without prior study or training or seeking external assistance?
	Operability	UOp-1-G	Operational consistency	Both	HR	$X = 1 - \frac{A}{B}$	A = Number of specific interactive tasks that are performed inconsistently  B = Number of specific interactive tasks that need to be consistent	To what extent do interactive tasks have a behaviour and appearance that is consistent both within the task and across similar tasks?

Quality characteristic	Quality sub-characteristic	ID	Quality measure name	Type	Recommendation level	Metric	Values	Description
Usability	Operability	UOp-2-G	Message clarity	Both	R	$X = \frac{A}{B}$	A = Number of messages that convey the right outcome or instructions to the user B = Number of messages implemented	What proportion of messages from a system conveys the right outcome or instructions to the user?
		UOp-3-S	Functional customizability	Both	UD	$X = \frac{A}{B}$	A = Number of functions and operational procedures which can be customized for user's convenience B = Number of functions and operational procedures for which users could benefit from customization	What proportion of functions and operational procedures can a user customize for his convenience?
		UOp-4-S	User interface customizability	Both	UD	$X = \frac{A}{B}$	A = Number of user interface elements that can be customized B = Number of user interface elements that could benefit from customization	What proportion of user interface elements can be customized in appearance?
		UOp-5-S	Monitoring capability	Both	UD	$X = \frac{A}{B}$	A = Number of functions having state monitoring capability B = Number of functions that could benefit from monitoring capability	What proportion of function states can be monitored during operation?
		UOp-6-S	Undo capability	Both	R	$X = \frac{A}{B}$	A = Number of tasks that provide undo capability or prompt for re-confirmation B = Number of tasks for which users could benefit from having re-confirmation or undo capability	What proportion of tasks that has a significant consequence provides an option for re-confirmation or undo capability?
		UOp-7-S	Understandable categorization of information	Both	R	$X = \frac{A}{B}$	A = Number of information structures that are familiar and convenient for the intended users B = Number of information structures used	To what extent does the software organize information in categories that are familiar to the intended users and convenient for their tasks?

Quality characteristic	Quality sub-characteristic	ID	Quality measure name	Type	Recommendation level	Metric	Values	Description
Usability	Operability	UOp-8-S	Appearance consistency	Both	UD	$X = 1 - \frac{A}{B}$	A = Number of user interfaces with similar items but with different appearances B = Number of user interfaces with similar items	What proportion of user interfaces with similar items has a similar appearance?
		UOp-9-S	Input device support	Both	UD	$X = \frac{A}{B}$	A = Number of tasks that can be initiated by all appropriate input modalities B = Number of tasks supported by the system	To what extent can tasks be initiated by all appropriate input modalities (such as keyboard, mouse or voice)?
	User error protection	UEp-1-G	Avoidance of user operation error	Both	HR	$X = \frac{A}{B}$	A = Number of user actions and inputs that are protected from causing any system malfunction B = Number of user actions and inputs that could be protected from causing any system malfunction	What portion of user actions and inputs are protected against causing any system malfunction?
		UEp-2-S	User entry error correction	Both	HR	$X = \frac{A}{B}$	A = Number of entry errors for which the system provides a suggested correct value B = Number of entry errors detected	To what extent does the system provide a suggested correct value for detected user entry errors with an identifiable cause?
		UEp-3-S	User error recoverability	Both	R	$X = \frac{A}{B}$	A = Number of user errors that are designed and tested to be recovered by the system B = Number of user errors which can occur during operation	What proportion of user errors can be corrected or recovered by the system?

Quality characteristic	Quality sub-characteristic	ID	Quality measure name	Type	Recommendation level	Metric	Values	Description
Usability	User interface aesthetics	UIn-1-S	Appearance aesthetics of user interfaces	Both	UD	$X = \frac{A}{B}$	A = Number of display interfaces aesthetically pleasing to the users in appearance B = Number of display interfaces	To what extent are user interfaces and the overall design aesthetically pleasing in appearance?
	Accessibility	UAc-1-G	Accessibility for users with disabilities	Both	R	$X = \frac{A}{B}$	A = Number of functions successfully usable by the users with a specific disability B = Number of functions implemented	To what extent can potential users with specific disabilities successfully use the system (with assistive technology if appropriate)?
		UAc-2-S	Supported languages adequacy	Both	UD	$X = \frac{A}{B}$	A = Number of languages actually supported B = Number of languages needed to be supported	What proportion of needed languages is supported?
Reliability	Maturity	RMa-1-G	Fault correction	Both	HR	$X = \frac{A}{B}$	A = Number of reliability-related faults corrected in design/coding/testing phase B = Number of reliability-related faults detected in design/coding/testing phase	What proportion of detected reliability-related faults has been corrected?
		RMa-2-G	Mean time between failures (MTBF)	External	HR	$X = \frac{A}{B}$	A = Operation time B = Number of system/software failures actually occurred	What is the MTBF during the system/software operation?
		RMa-3-S	Failure rate	External	R	$X = \frac{A}{B}$	A = Number of failures detected during observation time B = Duration of observation	What is the average number of failures during a defined period?

Quality characteristic	Quality sub-characteristic	ID	Quality measure name	Type	Recommendation level	Metric	Values	Description
Reliability	Maturity	RMa-4-S	Test coverage	External	R	$X = \frac{A}{B}$	<p>A = Number of system or software capabilities, operational scenarios or functions that are actually performed</p> <p>B = Number of system or software capabilities, operational scenarios or functions which are included in their associated test suites</p>	What percentage of the system or software capabilities, operational scenarios or functions that are included in their associated test suites are actually performed?
	Availability	RAv-1-G	System availability	External	HR	$X = \frac{A}{B}$	<p>A = System operation time actually provided</p> <p>B = System operation time specified in the operation schedule</p>	For what proportion of the scheduled system operational time is the system actually available?
		RAv-2-G	Mean down time	External	R	$X = \frac{A}{B}$	<p>A = Total down time</p> <p>B = Number of breakdowns observed</p>	How long does the system stay unavailable when a failure occurs?
	Fault tolerance	RFt-1-G	Failure avoidance	External	HR	$X = \frac{A}{B}$	<p>A = Number of avoided critical and serious failure occurrences (based on test cases)</p> <p>B = Number of executed test cases of fault pattern (almost causing failure) during testing</p>	What proportion of fault patterns has been brought under control to avoid critical and serious failures?
		RFt-2-S	Redundancy of components	Both	R	$X = \frac{A}{B}$	<p>A = Number of system components redundantly installed</p> <p>B = Number of system components</p>	What proportion of system components is installed redundantly to avoid system failure?
		RFt-3-S	Mean fault notification time	External	UD	$X = \sum_{i=1 \text{ to } n} \frac{(A_i - B_i)}{n}$	<p>A<sub>i</sub> = Time at which the fault i is reported by the system</p> <p>B<sub>i</sub> = Time at which fault i is detected</p> <p>n = Number of faults detected</p>	How quickly does the system report the occurrence of faults?

Quality characteristic	Quality sub-characteristic	ID	Quality measure name	Type	Recommendation level	Metric	Values	Description
<b>Reliability</b>	Recoverability	RRe-1-G	Mean recovery time	External	HR	$X = \sum_{i=1 \text{ to } n} \frac{A_i}{n}$	<p><math>A_i</math> = Total time to recover the downed software /system and re-initiate operation for each failure <math>i</math></p> <p><math>n</math> = Number of failures</p>	How long does it take for the software/system to recover from failure?
		RRe-2-S	Backup data completeness	Both	R	$X = \frac{A}{B}$	<p><math>A</math> = Number of data items actually backed up regularly</p> <p><math>B</math> = Number of data items requiring backup for error recovery</p>	What proportion of data items is backed up regularly?
<b>Security</b>	Confidentiality	SCo-1-G	Access controllability	Both	HR	$X = 1 - \frac{A}{B}$	<p><math>A</math> = Number of confidential data items that can be accessed without authorization</p> <p><math>B</math> = Number of data items that require access control</p>	What proportion of confidential data items are protected from unauthorized accesses?
		SCo-2-G	Data encryption correctness	Both	R	$X = \frac{A}{B}$	<p><math>A</math> = Number of data items encrypted/decrypted correctly</p> <p><math>B</math> = Number of data items that require encryption/decryption</p>	How correctly is the encryption/decryption of data items implemented as stated in the requirement specification?
		SCo-3-S	Strength of cryptographic algorithms	Both	UD	$X = 1 - \frac{A}{B}$	<p><math>A</math> = Number of cryptographic algorithms broken or unacceptably risky in use</p> <p><math>B</math> = Number of cryptographic algorithms used</p>	What proportion of cryptographic algorithms has been well-vetted?
	Integrity	SIn-1-G	Data integrity	Both	HR	$X = 1 - \frac{A}{B}$	<p><math>A</math> = Number of data items which are actually corrupted by unauthorized access</p> <p><math>B</math> = Number of data items for which data corruption or modification have to be prevented</p>	To what extent is the data corruption or modification by unauthorized access prevented?



Quality characteristic	Quality sub-characteristic	ID	Quality measure name	Type	Recommendation level	Metric	Values	Description
Security	Integrity	SIn-2-G	Internal data corruption prevention	Both	R	$X = \frac{A}{B}$	A = Number of data corruption prevention methods actually implemented B = Number of data corruption prevention methods available and recommended	To what extent are the available prevention methods for data corruption implemented?
		SIn-3-S	Buffer overflow prevention	Internal	UD	$X = \frac{A}{B}$	A = Number of memory accesses with user input that are bounds checked B = Number of memory accesses with user input in software modules	What portion of memory accesses with user input in software modules has been done bounds checking for preventing buffer overflow?
	Non-repudiation	SNo-1-G	Digital signature usage	Both	R	$X = \frac{A}{B}$	A = Number of events that ensure non-repudiation using digital signature B = Number of events requiring non-repudiation using digital signature	What proportion of events requiring non-repudiation is processed using digital signature?
	Accountability	SAC-1-G	User audit trail completeness	Both	HR	$X = \frac{A}{B}$	A = Number of accesses recorded in all logs B = Number of accesses to system or data actually tested	How complete is the audit trail concerning the user access to the system or data?
		SAC-2-S	System log retention	Both	R	$X = \frac{A}{B}$	A = Duration for which the system log is actually retained in stable storage B = Retention period specified for keeping the system log in stable storage	For what percent of the required retention period is the system log retained in stable storage?
	Authenticity	SAu-1-G	Authentication mechanisms sufficiency	Both	HR	$X = \frac{A}{B}$	A = Number of authentication mechanisms provided (e.g., User ID/password or IC card) B = Number of authentication mechanisms specified	How well does the system authenticate the identity of a subject?

Quality characteristic	Quality sub-characteristic	ID	Quality measure name	Type	Recommendation level	Metric	Values	Description
Security	Authenticity	SAu-2-S	Authentication rules conformity	Both	R	$X = \frac{A}{B}$	A = Number of authentication rules implemented B = Number of authentication rules specified	What proportion of the required authentication rules is established?
Maintainability	Modularity	MMo-1-G	Coupling of components	Both	R	$X = \frac{A}{B}$	A = Number of components which are implemented with no impact on others B = Number of specified components which are required to be independent	How strongly are the components independent and how many components are free from impacts from changes to other components in a system or computer program?
		MMo-2-S	Cyclomatic complexity adequacy	Internal	UD	$X = 1 - \frac{A}{B}$	A = Number of software modules which have a cyclomatic complexity score that exceeds the specified threshold B = Number of software modules implemented	How many software modules have acceptable cyclomatic complexity?
	Reusability	MRe-1-G	Reusability of assets	Both	HR	$X = \frac{A}{B}$	A = Number of assets which are designed and implemented to be reusable B = Number of assets in a system	How many assets in a system can be reusable?
		MRe-2-S	Coding rules conformity	Internal	R	$X = \frac{A}{B}$	A = Number of software modules conforming to coding rules for a specific system B = Number of software modules implemented	How many modules conform to required coding rules?
	Analysability	MAAn-1-G	System log completeness	Both	HR	$X = \frac{A}{B}$	A = Number of logs that are actually recorded in the system B = Number of logs for which audit trails are required during operation	To what extent does the system record its operations in logs so that they are to be traceable?

Quality characteristic	Quality sub-characteristic	ID	Quality measure name	Type	Recommendation level	Metric	Values	Description
Maintainability	Analysability	MAn-2-S	Diagnosis function effectiveness	Both	R	$X = \frac{A}{B}$	A = Number of diagnostic functions useful for causal analysis B = Number of diagnostic functions implemented	What proportion of the diagnosis functions meets the requirements of causal analysis?
		MAn-3-S	Diagnosis function sufficiency	Both	R	$X = \frac{A}{B}$	A = Number of diagnostic functions implemented B = Number of diagnostic functions required	What proportion of the required diagnosis functions has been implemented?
	Modifiability	MMd-1-G	Modification efficiency	Both	HR	$X = \frac{(\frac{A_i}{B_i})}{n}$	A <sub>i</sub> = Total work time spent for making a specific type of modification i B <sub>i</sub> = Expected time for making the specific type of modification i n = Number of modifications measured	How efficiently are the modifications made compared to the expected time?
		MMd-2-G	Modification correctness	Both	HR	$X = 1 - (\frac{A}{B})$	A = Number of modifications that caused an incident or failure within a defined period after being implemented B = Number of modifications implemented	What proportion of modifications has been implemented correctly?
		MMd-3-S	Modification capability	Both	UD	$X = \frac{A}{B}$	A = Number of items actually modified within a specified duration B = Number of items required to be modified within a specified duration	To what extent are the required modifications made within a specified duration?
	Testability	MTe-1-G	Test function completeness	Both	R	$X = \frac{A}{B}$	A = Number of test functions implemented as specified B = Number of test functions required	How completely are test functions and facilities implemented?

Quality characteristic	Quality sub-characteristic	ID	Quality measure name	Type	Recommendation level	Metric	Values	Description
Maintainability	Testability	MTe-2-S	Autonomous testability	Both	UD	$X = \frac{A}{B}$	A = Number of tests that can be simulated by stub among the tests which depend on other systems  B = Number of tests which depend on other systems	How independently can the software be tested?
		MTe-3-S	Test restartability	Both	UD	$X = \frac{A}{B}$	A = Number of cases in which maintainer can pause and restart executing test run at desired points to check step by step  B = Number of cases in which executing test run can be paused	How easily can the operation test be carried out from the restart point after maintenance?
Portability	Adaptability	PAd-1-G	Hardware environmental adaptability	External	HR	$X = 1 - \frac{A}{B}$	A = Number of functions which were not completed or results which were insufficient to meet requirements during testing  B = Number of functions which were tested in different hardware environment	Is software or system capable enough to adapt itself to different hardware environment?
		PAd-2-G	System software environmental adaptability	External	HR	$X = 1 - \frac{A}{B}$	A = Number of functions which were not completed or results which were insufficient to meet requirements during testing  B = Number of functions which were tested in different system software environment	Is software or system capable enough to adapt itself to different system software environment?

Quality characteristic	Quality sub-characteristic	ID	Quality measure name	Type	Recommendation level	Metric	Values	Description
Portability	Adaptability	PAd-3-S	Operational environment adaptability	External	UD	$X = 1 - \frac{A}{B}$	<p>A = Number of functions which were not completed or results which were insufficient to meet requirements during operational testing with user's environment</p> <p>B = Number of functions which were tested in different operational environment</p>	Is software or system capable enough to adapt itself to different operational environment?
	Installability	PIn-1-G	Installation time efficiency	External	R	$X = \frac{(\frac{A_i}{B_i})}{n}$	<p>A<sub>i</sub> = Total work time spent for making an installation i</p> <p>B<sub>i</sub> = Expected time for making an installation i</p> <p>n = Number of installations measured</p>	How efficient is the actual installation time compared to expected time?
		PIn-2-G	Ease of installation	External	R	$X = \frac{A}{B}$	<p>A = Number of cases in which a user succeeds to customize the installation procedure</p> <p>B = Number of cases in which a user attempted to customize the installation procedure for user's convenience</p>	Can users or maintainers customize the installation procedure for their convenience?
	Replaceability	PRE-1-G	Usage similarity	Both	HR	$X = \frac{A}{B}$	<p>A = Number of user functions which can be performed without any additional learning or workaround</p> <p>B = Number of user functions in the replaced software product</p>	What proportion of user functions of the replaced product can be performed without any additional learning or workaround?
		PRE-2-S	Product quality equivalence	Both	R	$X = \frac{A}{B}$	<p>A = Number of quality measures of the new product which are better or equal to the replaced product</p> <p>B = Number of quality measures of the replaced software product that are relevant</p>	What proportion of the quality measures is satisfied after replacing previous software product by this one?

Quality characteristic	Quality sub-characteristic	ID	Quality measure name	Type	Recommendation level	Metric	Values	Description
Portability	Replaceability	Pre-3-S	Functional inclusiveness	External	R	$X = \frac{A}{B}$	A = Number of functions which produce similar results as before B = Number of functions which have to be used in the replaced software product	Can the similar functions easily be used after replacing previous software product by this one?
		Pre-4-S	Data reusability/import capability	External	UD	$X = \frac{A}{B}$	A = Number of data which can be used continuously as before B = Number of data which are to be used continuously in the replaced software product	Can the same data be used after replacing previous software product by this one?

**Tab. 29 – Qualitätsmerkmale, Maße und Metriken der ISO/IEC 25023<sup>92</sup>**

<sup>92</sup> Eigene Darstellung nach (vgl. [24], S. 8 ff.)

## C Übersicht der unterstützten Programmiersprachen

		Werkzeuge											
		SonarQube	ALM	Silk Test	JProfiler	Security Fortify on Demand	Code Inspector	Checkstyle	CodeSurfer	Jessie Plugin	Open Markov	Xfmea	Isograph
Programmiersprachen	N/A		x								x	x	x
	.NET			x									
	ABAP	x		x		x							
	ActionScript/MXML					x							
	AJAX					x							
	ASP.NET					x							
	Apex					x							
	C	x				x	x		x	x			
	C++	x				x			x				
	C#	x				x							
	ClassicASP												
	COBOL	x				x							
	ColdFusion CFML					x							
	DHTML			x									
	Flex	x		x									
	HTML					x							
	HTLM5			x									
	Intel x86 machine code								x				
	Java	x		x	x	x	x	x					
	JavaScript	x				x							
	JSP					x							
	Objective-C	x				x							
	PHP	x				x							
	PL/I	x											
	PL/SQL	x				x							
	Python	x				x							
	RPG	x											
	Ruby					x							
	Swift	x				x							
	T-SQL					x							
	VB6	x											
	VB.NET	x				x							
	Visual Basic					x							
	VBScript					x							
	Web	x											
	XML	x				x							
Σ Programmiersprachen		19	N/A	6	1	25	2	1	3	1	N/A	N/A	N/A

Tab. 30 – Übersicht der unterstützten Programmiersprachen der untersuchten Werkzeuge<sup>93</sup>

<sup>93</sup> Eigene Darstellung

## D Zuordnung dynamischer und statischer Eigenschaften nach Balzert

ID	Software Qualitätsmaß Bezeichnung	Statisch vs. dynamisch	Objektiv vs. subjektiv	Absolut vs. relativ	Explizit vs. abgeleitet	vorhersagend vs. erklärend	Prozess-orientiert vs. produkt-orientiert	Global vs. speziell
FCp-1-G	Functional coverage	s	Nicht relevant für Nachweis der Anwendbarkeit auf Qualitätsmodell der System-und Software Produktqualität					
FCr-1-G	Functional correctness	s						
FAp-1-G	Functional appropriateness of usage objective	s						
FAP-2-G	Functional appropriateness of system	s						
PTb-1-G	Mean response time	d						
PTb-2-G	Response time adequacy	d						
PTb-3-G	Mean turnaround time	d						
PTb-4-G	Turnaround time adequacy	d						
PTb-5-G	Mean throughput	d						
PRu-1-G	Mean processor utilization	d						
PRu-2-G	Mean memory utilization	s						
PRu-3-G	Mean I/O devices utilization	d						
PRu-4-S	Bandwidth utilization	d						
PCa-1-G	Transaction processing capacity	d						
PCa-2-G	User access capacity	s						
PCa-3-S	User access increase adequacy	d						
CCo-1-G	Co-existence with other products	s						
CIn-1-G	Data formats exchangeability	s						
CIn-2-G	Data exchange protocol sufficiency	s						
CIn-3-S	External interface adequacy	s						
UAp-1-G	Description completeness	s						
UAp-2-S	Demonstration coverage	s						
UAp-3-S	Entry point selfdescriptiveness	s						
ULe-1-G	User guidance completeness	s						
ULe-2-S	Entry fields defaults	s						
ULe-3-S	Error message understandability	s						
ULe-4-S	Self-explanatory user interface	s						
UOp-1-G	Operational consistency	s						
UOp-2-G	Message clarity	s						
UOp-3-S	Functional customizability	s						



ID	Software Qualitätsmaß Bezeichnung	Statisch vs. dynamisch	Objektiv vs. subjektiv	Absolut vs. relativ	Explizit vs. abgeleitet	vorhersagend vs. erklärend	Prozessorientiert vs. produktorientiert	Global vs. speziell
UOp-4-S	User interface customizability	s	Nicht relevant für Nachweis der Anwendbarkeit auf Qualitätsmodell der System-und Software Produktqualität					
UOp-5-S	Monitoring capability	s						
UOp-6-S	Undo capability	s						
UOp-7-S	Understandable categorization of information	s						
UOp-8-S	Appearance consistency	s						
UOp-9-S	Input device support	s						
UEp-1-G	Avoidance of user operation error	s						
UEp-2-S	User entry error correction	s						
UEp-3-S	User error recoverability	s						
UIn-1-S	Appearance aesthetics of user interfaces	s						
UAc-1-G	Accessibility for users with disabilities	s						
UAc-2-S	Supported languages adequacy	s						
RMa-1-G	Fault correction	s						
RMa-2-G	Mean time between failures (MTBF)	d						
RMa-3-S	Failure rate	d						
RMa-4-S	Test coverage	s						
RAv-1-G	System availability	d						
RAv-2-G	Mean down time	d						
RFt-1-G	Failure avoidance	s						
RFt-2-S	Redundancy of components	s						
RFt-3-S	Mean fault notification time	d						
RRe-1-G	Mean recovery time	d						
RRe-2-S	Backup data completeness	s						
SCo-1-G	Access controllability	s						
SCo-2-G	Data encryption correctness	s						
SCo-3-S	Strength of cryptographic algorithms	s						
SIn-1-G	Data integrity	s						
SIn-2-G	Internal data corruption prevention	s						
SIn-3-S	Buffer overflow prevention	s						
SNo-1-G	Digital signature usage	s						

ID	Software Qualitätsmaß Bezeichnung	Statisch vs. dynamisch	Objektiv vs. subjektiv	Absolut vs. relativ	Explizit vs. abgeleitet	vorhersagend vs. erklärend	Prozessorientiert vs. produktorientiert	Global vs. speziell
SAC-1-G	User audit trail completeness	s	Nicht relevant für Nachweis der Anwendbarkeit auf Qualitätsmodell der System-und Software Produktqualität					
SAC-2-S	System log retention	d						
SAu-1-G	Authentication mechanisms sufficiency	s						
SAu-2-S	Authentication rules conformity	s						
MMo-1-G	Coupling of components	s						
MMo-2-S	Cyclomatic complexity adequacy	s						
MRe-1-G	Reusability of assets	s						
MRe-2-S	Coding rules conformity	s						
MAn-1-G	System log completeness	s						
MAn-2-S	Diagnosis function effectiveness	s						
MAn-3-S	Diagnosis function sufficiency	s						
MMd-1-G	Modification efficiency	d						
MMd-2-G	Modification correctness	d						
MMd-3-S	Modification capability	d						
MTe-1-G	Test function completeness	s						
MTe-2-S	Autonomous testability	s						
MTe-3-S	Test restartability	s						
PAd-1-G	Hardware environmental adaptability	s						
PAd-2-G	System software environmental adaptability	s						
PAd-3-S	Operational environment adaptability	s						
PIn-1-G	Installation time efficiency	d						
PIn-2-G	Ease of installation	s						
PR-1-G	Usage similarity	s						
PR-2-S	Product quality equivalence	s						
PR-3-S	Functional inclusiveness	s						
PR-4-S	Data reusability/import capability	s						

**Tab. 31 – Zuordnung dynamischer und statischer Eigenschaften nach BALZERT<sup>94</sup>**

<sup>94</sup> Eigene Darstellung

## E Eignung Werkzeuge zur Bestimmung der Metrikparameter

ID	Quality measure name	Dynamische Testwerkzeuge				Statische Analysewerkzeuge			Formale Verifikationswerkzeuge	Modellierende und analysierende Werkzeuge		
		SonarQube	ALM	Silk Test	JProfiler	Code Inspector	Checkstyle	CodeSurfer	Jessie Plugin	Open Markov	Xfmea	Isograph
FCp-1-G	Functional coverage	0	1	0,5	0	0	0	0	0	0	0	0
FCr-1-G	Functional correctness	0,5	1	1	0	0	0	0	0,5	0	0	0
FAp-1-G	Functional appropriateness of usage objective	0,5	1	1	0	0	0	0	0,5	0	0	0
FAP-2-G	Functional appropriateness of system	0,5	1	0,5	0	0	0	0	0,5	0	0	0
PTb-1-G	Mean response time	0	0,5	0,5	0	0	0	0	0	0	0	0
PTb-2-G	Response time adequacy	0	0,5	0,5	0	0	0	0	0	0	0	0
PTb-3-G	Mean turnaround time	0	0	0,5	0	0	0	0	0	0	0	0
PTb-4-G	Turnaround time adequacy	0	0,5	0,5	0	0	0	0	0	0	0	0
PTb-5-G	Mean throughput	0	0	0	0	0	0	0	0	0	0	0
PRu-1-G	Mean processor utilization	0	0,5	0	0,5	0	0	0	0	0	0	0
PRu-2-G	Mean memory utilization	0	0,5	0	0,5	0	0	0	0	0	0	0
PRu-3-G	Mean I/O devices utilization	0	0	0	0	0	0	0	0	0	0	0
PRu-4-S	Bandwidth utilization	0	0	0	0	0	0	0	0	0	0	0,5

ID	Quality measure name	Dynamische Testwerkzeuge				Statische Analysewerkzeuge			Formale Verifikationswerkzeuge	Modellierende und analysierende Werkzeuge		
		SonarQube	ALM	Silk Test	JProfiler	Code Inspector	Checkstyle	CodeSurfer	Jessie Plugin	Open Markov	Xfmea	Isograph
PCa-1-G	Transaction processing capacity	0	0	0	0	0	0	0	0	0	0	0
PCa-2-G	User access capacity	0	0	0	0	0	0	0	0	0	0	0
PCa-3-S	User access increase adequacy	0	0	0	0	0	0	0	0	0	0	0
CCo-1-G	Co-existence with other products	0	0,5	0	0	0	0	0	0	0	0	0
CIn-1-G	Data formats exchangeability	0	0,5	0	0	0	0	0	0	0	0	0
CIn-2-G	Data exchange protocol sufficiency	0	0,5	0	0	0	0	0	0	0	0	0
CIn-3-S	External interface adequacy	0	0,5	0	0	0	0	0	0	0	0	0
UAp-1-G	Description completeness	0	0,5	0	0	0	0	0	0	0	0	0
UAp-2-S	Demonstration coverage	0	0	0	0	0	0	0	0	0	0	0
UAp-3-S	Entry point selfdescriptiveness	0	0	0	0	0	0	0	0	0	0	0
ULe-1-G	User guidance completeness	0	0,5	0	0	0	0	0	0	0	0	0
ULe-2-S	Entry fields defaults	0	0,5	0	0	0	0	0	0	0	0	0
ULe-3-S	Error message understandability	0	0	0	0	0	0	0	0	0	0	0

ID	Quality measure name	Dynamische Testwerkzeuge				Statische Analysewerkzeuge			Formale Verifikationswerkzeuge	Modellierende und analysierende Werkzeuge		
		SonarQube	ALM	Silk Test	JProfiler	Code Inspector	Checkstyle	CodeSurfer	Jessie Plugin	Open Markov	Xfmea	Isograph
ULe-4-S	Self-explanatory user interface	0	0	0	0	0	0	0	0	0	0	0
UOp-1-G	Operational consistency	0	0,5	0,5	0	0	0	0	0	0	0	0
UOp-2-G	Message clarity	0	0	0	0	0	0	0	0	0	0	0
UOp-3-S	Functional customizability	0	0	0	0	0	0	0	0	0	0	0
UOp-4-S	User interface customizability	0	0	0	0	0	0	0	0	0	0	0
UOp-5-S	Monitoring capability	0	0	0	0	0	0	0	0	0	0	0
UOp-6-S	Undo capability	0	0	0	0	0	0	0	0	0	0	0
UOp-7-S	Understandable categorization of information	0	0	0	0	0	0	0	0	0	0	0
UOp-8-S	Appearance consistency	0	0	0	0	0	0	0	0	0	0	0
UOp-9-S	Input device support	0	0	0,5	0	0	0	0	0	0	0	0
UEp-1-G	Avoidance of user operation error	0,5	0	0	0	0	0	0	0,5	0	0	0
UEp-2-S	User entry error correction	0	0	0	0	0	0	0	0,5	0	0	0
UEp-3-S	User error recoverability	0	0	0	0	0	0	0	0	0	0	0,5

ID	Quality measure name	Dynamische Testwerkzeuge				Statische Analysewerkzeuge			Formale Verifikationswerkzeuge	Modellierende und analysierende Werkzeuge		
		SonarQube	ALM	Silk Test	JProfiler	Code Inspector	Checkstyle	CodeSurfer	Jessie Plugin	Open Markov	Xfmea	Isograph
UIn-1-S	Appearance aesthetics of user interfaces	0	0	0	0	0	0	0	0	0	0	0
UAc-1-G	Accessibility for users with disabilities	0	0,5	0	0	0	0	0	0	0	0	0
UAc-2-S	Supported languages adequacy	0	1	0	0	0	0	0	0	0	0	0
RMa-1-G	Fault correction	0,5	0,5	0,5	0	0	0	0	0	0	0	0
RMa-2-G	Mean time between failures (MTBF)	0,5	0,5	0,5	0,5	0	0	0	0,5	0	0	0
RMa-3-S	Failure rate	0,5	0,5	0,5	0,5	0	0	0	0,5	0	0	0
RMa-4-S	Test coverage	0	1	0,5	0	0	0	0	0	0	0	0
RAv-1-G	System availability	0	0	0	0	0	0	0	0	0	0	0
RAv-2-G	Mean down time	0	0	0	0	0	0	0	0	0	0	0,5
RFt-1-G	Failure avoidance	0,5	0,5	0,5	0	0	0	0	0	0	0,5	0
RFt-2-S	Redundancy of components	0	0	0	0	0	0	0	0	0	0	0,5
RFt-3-S	Mean fault notification time	0,5	0	0	0	0	0,5	0	0	0	0	0
RRe-1-G	Mean recovery time	0	0	0	0	0	0,5	0	0	0	0	0

ID	Quality measure name	Dynamische Testwerkzeuge				Statische Analysewerkzeuge			Formale Verifikationswerkzeuge	Modellierende und analysierende Werkzeuge		
		SonarQube	ALM	Silk Test	JProfiler	Code Inspector	Checkstyle	CodeSurfer	Jessie Plugin	Open Markov	Xfmea	Isograph
RRe-2-S	Backup data completeness	0	0	0	0	0	0	0	0	0	0	0
SCo-1-G	Access controllability	0	0	0	0	0	0	0	0	0	0	0
SCo-2-G	Data encryption correctness	0	0	0	0	0	0	0	0	0	0	0
SCo-3-S	Strength of cryptographic algorithms	0	0	0	0	0	0	0	0	0	0	0
SIn-1-G	Data integrity	0	0	0	0	0	0	0	0	0	0	0
SIn-2-G	Internal data corruption prevention	0	0	0	0	0	0	0	0	0	0	0
SIn-3-S	Buffer overflow prevention	0	0	0	0,5	0	0,5	0	0,5	0	0	0
SNo-1-G	Digital signature usage	0	0	0	0	0	0	0	0	0	0	0
SAC-1-G	User audit trail completeness	0	0	0	0	0	0	0	0	0	0	0
SAC-2-S	System log retention	0	0	0	0	0	0	0	0	0	0	0
SAu-1-G	Authentication mechanisms sufficiency	0	0,5	0	0	0	0	0	0	0	0	0
SAu-2-S	Authentication rules conformity	0	0,5	0	0	0	0	0	0	0	0	0
MMo-1-G	Coupling of components	0	0	0	0	0,5	0	0,5	0	0,5	0,5	0,5

ID	Quality measure name	Dynamische Testwerkzeuge				Statische Analysewerkzeuge			Formale Verifikationswerkzeuge	Modellierende und analysierende Werkzeuge		
		SonarQube	ALM	Silk Test	JProfiler	Code Inspector	Checkstyle	CodeSurfer	Jessie Plugin	Open Markov	Xfmea	Isograph
MMo-2-S	Cyclomatic complexity adequacy	0,5	0,5	0	0	0,5	0,5	0,5	0	0	0	0
MRe-1-G	Reusability of assets	0	0	0	0	0	0	0	0	0	0	0
MRe-2-S	Coding rules conformity	0	0,5	0	0	0,5	0,5	0,5	0,5	0	0	0
MAAn-1-G	System log completeness	0	0	0	0	0	0	0	0	0	0	0
MAAn-2-S	Diagnosis function effectiveness	0	0	0	0	0	0	0	0	0	0	0
MAAn-3-S	Diagnosis function sufficiency	0	1	0	0	0	0	0	0	0	0	0
MMd-1-G	Modification efficiency	0	0	0	0	0	0	0	0	0	0	0
MMd-2-G	Modification correctness	0	0,5	0,5	0	0	0,5	0	0	0	0	0
MMd-3-S	Modification capability	0	0,5	0	0	0	0	0	0	0	0	0
MTe-1-G	Test function completeness	0	0,5	0	0	0	0	0	0	0	0	0
MTe-2-S	Autonomous testability	0	0	0	0	0	0	0	0	0,5	0	0
MTe-3-S	Test restartability	0,5	0	0,5	0	0,5	0	0,5	0	0	0	0
PAd-1-G	Hardware environmental adaptability	0,5	0,5	0,5	0	0	0,5	0	0,5	0	0	0



ID	Quality measure name	Dynamische Testwerkzeuge				Statische Analysewerkzeuge			Formale Verifikationswerkzeuge	Modellierende und analysierende Werkzeuge		
		SonarQube	ALM	Silk Test	JProfiler	Code Inspector	Checkstyle	CodeSurfer	Jessie Plugin	Open Markov	Xfmea	Isograph
PAd-2-G	System software environmental adaptability	0,5	0,5	0,5	0	0	0,5	0	0,5	0	0	0
PAd-3-S	Operational environment adaptability	0,5	0,5	0,5	0	0	0	0	0,5	0	0	0
PIIn-1-G	Installation time efficiency	0	0	0,5	0	0	0	0	0	0	0	0
PIIn-2-G	Ease of installation	0	0	0	0	0	0	0	0	0	0	0
PRe-1-G	Usage similarity	0	0,5	0	0	0	0	0	0	0	0	0
PRe-2-S	Product quality equivalence	0	0,5	0	0	0	0,5	0	0	0	0	0
PRe-3-S	Functional inclusiveness	0	0,5	0,5	0	0	0,5	0	0	0	0	0
PRe-4-S	Data reusability/import capability	0	0	0,5	0	0	0	0	0	0	0	0

**Tab. 32 – Eignung Werkzeuge zur Bestimmung der Metrikparameter**<sup>95</sup>

<sup>95</sup> Eigene Darstellung

## F Aggregation der Bewertungsergebnisse der Werkzeugklassen

ID	Software Qualitätsmaß Bezeichnung	Dynamische Testwerkzeuge	Statische Analysewerkzeuge	Formale Verifikationswerkzeuge	Modellierende und analysierende Werkzeuge
FCp-1-G	Functional coverage	1	0	0	0
FCr-1-G	Functional correctness	1	0	0,5	0
FAP-1-G	Functional appropriateness of usage objective	1	0	0,5	0
FAP-2-G	Functional appropriateness of system	1	0	0,5	0
PTb-1-G	Mean response time	0,5	0	0	0
PTb-2-G	Response time adequacy	0,5	0	0	0
PTb-3-G	Mean turnaround time	0,5	0	0	0
PTb-4-G	Turnaround time adequacy	0,5	0	0	0
PTb-5-G	Mean throughput	0	0	0	0
PRu-1-G	Mean processor utilization	0,5	0	0	0
PRu-2-G	Mean memory utilization	0,5	0	0	0
PRu-3-G	Mean I/O devices utilization	0	0	0	0
PRu-4-S	Bandwidth utilization	0	0	0	0,5

<b>ID</b>	<b>Software Qualitätsmaß Bezeichnung</b>	<b>Dynamische Test- werkzeuge</b>	<b>Statische Analyse- werkzeuge</b>	<b>Formale Verifikati- onswerkzeuge</b>	<b>Modellierende und analysierende Werkzeuge</b>
PCa-1-G	Transaction processing capacity	0	0	0	0
PCa-2-G	User access capacity	0	0	0	0
PCa-3-S	User access increase adequacy	0	0	0	0
CCo-1-G	Co-existence with other products	0,5	0	0	0
CIn-1-G	Data formats exchangeability	0,5	0	0	0
CIn-2-G	Data exchange protocol sufficiency	0,5	0	0	0
CIn-3-S	External interface adequacy	0,5	0	0	0
UAp-1-G	Description completeness	0,5	0	0	0
UAp-2-S	Demonstration coverage	0	0	0	0
UAp-3-S	Entry point selfdescriptiveness	0	0	0	0
ULe-1-G	User guidance completeness	0,5	0	0	0
ULe-2-S	Entry fields defaults	0,5	0	0	0
ULe-3-S	Error message understandability	0	0	0	0
ULe-4-S	Self-explanatory user interface	0	0	0	0

<b>ID</b>	<b>Software Qualitätsmaß Bezeichnung</b>	<b>Dynamische Test- werkzeuge</b>	<b>Statische Analyse- werkzeuge</b>	<b>Formale Verifikati- onswerkzeuge</b>	<b>Modellierende und analysierende Werkzeuge</b>
UOp-1-G	Operational consistency	0,5	0	0	0
UOp-2-G	Message clarity	0	0	0	0
UOp-3-S	Functional customizability	0	0	0	0
UOp-4-S	User interface customizability	0	0	0	0
UOp-5-S	Monitoring capability	0	0	0	0
UOp-6-S	Undo capability	0	0	0	0
UOp-7-S	Understandable categorization of in- formation	0	0	0	0
UOp-8-S	Appearance consistency	0	0	0	0
UOp-9-S	Input device support	0,5	0	0	0
UEp-1-G	Avoidance of user operation error	0,5	0	0,5	0
UEp-2-S	User entry error correction	0	0	0,5	0
UEp-3-S	User error recoverability	0	0	0	0,5
UIn-1-S	Appearance aesthetics of user inter- faces	0	0	0	0
UAc-1-G	Accessibility for users with disabilities	0,5	0	0	0

<b>ID</b>	<b>Software Qualitätsmaß Bezeichnung</b>	<b>Dynamische Test- werkzeuge</b>	<b>Statische Analyse- werkzeuge</b>	<b>Formale Verifikati- onswerkzeuge</b>	<b>Modellierende und analysierende Werkzeuge</b>
UAc-2-S	Supported languages adequacy	1	0	0	0
RMa-1-G	Fault correction	0,5	0	0	0
RMa-2-G	Mean time between failures (MTBF)	0,5	0	0,5	0
RMa-3-S	Failure rate	0,5	0	0,5	0
RMa-4-S	Test coverage	1	0	0	0
RAv-1-G	System availability	0	0	0	0
RAv-2-G	Mean down time	0	0	0	0,5
RFt-1-G	Failure avoidance	0,5	0	0	0,5
RFt-2-S	Redundancy of components	0	0	0	0,5
RFt-3-S	Mean fault notification time	0,5	0,5	0	0
RRe-1-G	Mean recovery time	0	0,5	0	0
RRe-2-S	Backup data completeness	0	0	0	0
SCo-1-G	Access controllability	0	0	0	0
SCo-2-G	Data encryption correctness	0	0	0	0

<b>ID</b>	<b>Software Qualitätsmaß Bezeichnung</b>	<b>Dynamische Test- werkzeuge</b>	<b>Statische Analyse- werkzeuge</b>	<b>Formale Verifikati- onswerkzeuge</b>	<b>Modellierende und analysierende Werkzeuge</b>
SCo-3-S	Strength of cryptographic algorithms	0	0	0	0
SIn-1-G	Data integrity	0	0	0	0
SIn-2-G	Internal data corruption prevention	0	0	0	0
SIn-3-S	Buffer overflow prevention	0,5	0,5	0,5	0
SNo-1-G	Digital signature usage	0	0	0	0
SAc-1-G	User audit trail completeness	0	0	0	0
SAc-2-S	System log retention	0	0	0	0
SAu-1-G	Authentication mechanisms sufficiency	0,5	0	0	0
SAu-2-S	Authentication rules conformity	0,5	0	0	0
MMo-1-G	Coupling of components	0	0,5	0	0,5
MMo-2-S	Cyclomatic complexity adequacy	0,5	0,5	0	0
MRe-1-G	Reusability of assets	0	0	0	0
MRe-2-S	Coding rules conformity	0,5	0,5	0,5	0
MAAn-1-G	System log completeness	0	0	0	0

<b>ID</b>	<b>Software Qualitätsmaß Bezeichnung</b>	<b>Dynamische Test- werkzeuge</b>	<b>Statische Analyse- werkzeuge</b>	<b>Formale Verifikati- onswerkzeuge</b>	<b>Modellierende und analysierende Werkzeuge</b>
MAAn-2-S	Diagnosis function effectiveness	0	0	0	0
MAAn-3-S	Diagnosis function sufficiency	1	0	0	0
MMd-1-G	Modification efficiency	0	0	0	0
MMd-2-G	Modification correctness	0,5	0,5	0	0
MMd-3-S	Modification capability	0,5	0	0	0
MTe-1-G	Test function completeness	0,5	0	0	0
MTe-2-S	Autonomous testability	0	0	0	0,5
MTe-3-S	Test restartability	0,5	0,5	0	0
PAd-1-G	Hardware environmental adaptability	0,5	0,5	0,5	0
PAd-2-G	System software environmental adaptability	0,5	0,5	0,5	0
PAd-3-S	Operational environment adaptability	0,5	0	0,5	0
PIn-1-G	Installation time efficiency	0,5	0	0	0
PIn-2-G	Ease of installation	0	0	0	0
PRE-1-G	Usage similarity	0,5	0	0	0

<b>ID</b>	<b>Software Qualitätsmaß Bezeichnung</b>	<b>Dynamische Test- werkzeuge</b>	<b>Statische Analyse- werkzeuge</b>	<b>Formale Verifikati- onswerkzeuge</b>	<b>Modellierende und analysierende Werkzeuge</b>
Pre-2-S	Product quality equivalence	0,5	0,5	0	0
Pre-3-S	Functional inclusiveness	0,5	0,5	0	0
Pre-4-S	Data reusability/import capability	0,5	0	0	0

**Tab. 33 – Aggregation der Bewertungsergebnisse der Werkzeugklassen<sup>96</sup>**

<sup>96</sup> Eigene Darstellung



## G Quantitative Analyse der System- und Software Produkt Maße

ID	Parameter Beschreibung	Schlagwörter	Eingangsgrößen
FCp-1-G	A = Number of functions missing B = Number of functions specified	specified	Anforderung
FCr-1-G	A = Number of functions that are incorrect B = Number of functions considered	-	-
FAP-1-G	A = Number of functions missing or incorrect among those that are required for achieving a specific usage objective B = Number of functions required for achieving a specific usage objective	specific, objective, require	Anforderung, Kontext und Anwender
FAP-2-G	A <sub>i</sub> = Appropriateness score for usage objective i, that is, the measured value of FAP-1-G for i-th specific usage objective n = Number of usage objectives	specific, objective, appropriate	Anforderung, Kontext und Anwender
PTb-1-G	A <sub>i</sub> = Time taken by the system to respond to a specific user task or system task at i-th measurement n = Number of responses measured	specific, user, system	Anforderung, Kontext und Anwender, Umgebungsbedingungen
PTb-2-G	A = Mean response time measured by PTb-1-G B = Target response time specified	specified	Anforderung
PTb-3-G	A <sub>i</sub> = Time of starting a job i B <sub>i</sub> = Time of completing the job i n = Number of measurements	job	Betrieb
PTb-4-G	A = Mean turnaround time measured by PTb-3-G B = Target turnaround time specified	specified	Anforderung
PTb-5-G	A <sub>i</sub> = Number of jobs completed during the i-th observation time B <sub>i</sub> = i-th observation time period n = Number of observations	job	Betrieb
PRu-1-G	A <sub>i</sub> = Processor time actually used to execute a given set of tasks in observation i B <sub>i</sub> = Operation time to perform the tasks in observation i n = Number of observations	processor, operation	Umgebungsbedingungen, Betrieb
PRu-2-G	A <sub>i</sub> = Size of memory actually used to perform a given set of tasks for i-th sample processing B <sub>i</sub> = Size of memory available to perform the tasks during i-th sample processing n = Number of samples processed	memory	Umgebungsbedingungen
PRu-3-G	A <sub>i</sub> = Duration of I/O device(s) busy time to perform a given set of tasks for i-th observation B <sub>i</sub> = Duration of I/O operations to perform the tasks for i-th observation n = Number of observations	device, operation	Umgebungsbedingungen, Betrieb
PRu-4-S	A = Bandwidth of actual transmission measured over time to perform a given set of tasks B = Bandwidth capacity available to perform a given set of tasks	capacity	Umgebungsbedingungen

ID	Parameter Beschreibung	Schlagwörter	Eingangsgrößen
PCa-1-G	A = Number of transactions completed during observation time B = Duration of observation	-	-
PCa-2-G	A <sub>i</sub> = Maximum number of users who can simultaneously access the system at i-th observation n = Number of observations	user, system	Kontext und Anwender, Umgebungsbedingungen
PCa-3-S	A = Number of users successfully added during observation time B = Duration of observation	user	Kontext und Anwender
CCo-1-G	A = Number of other specified software products with which this product can co-exist B = Number of other software products specified to co-exist with this product in the operation environment	specified, software, environment, operation, other	Anforderung, Umgebungsbedingungen, Betrieb
CIn-1-G	A = Number of data formats exchangeable with other software or systems B = Number of data formats specified to be exchangeable	software, system, specified, other	Anforderung, Umgebungsbedingungen
CIn-2-G	A = Number of data exchange protocols supported B = Number of data exchange protocols specified to be supported	specified	Anforderung
CIn-3-S	A = Number of external interfaces that are functional B = Number of external interfaces specified	specified	Anforderung
UAp-1-G	A = Number of usage scenarios described in the product description or user documents B = Number of usage scenarios of the product	scenario, user document, user	Betrieb, Kontext und Anwender
UAp-2-S	A = Number of tasks with demonstration features B = Number of tasks that could benefit from demonstration features	benefit	Kontext und Anwender
UAp-3-S	A = Number of landing pages that explain the purpose of website B = Number of landing pages in a website	explain	Kontext und Anwender
ULe-1-G	A = Number of functions described in user documentation and/or help facility as required B = Number of functions implemented that are required to be documented	user document, require, user	Betrieb, Anforderung
ULe-2-S	A = Number of entry fields whose default values have been automatically filled in during operation B = Number of entry fields that could have default values	default value, operation	Programmier-richtlinien, Betrieb
ULe-3-S	A = Number of error messages which state the reason of occurrence and suggest the ways of resolution where this is possible B = Number of error messages implemented	error message	Programmier-richtlinien

ID	Parameter Beschreibung	Schlagwörter	Eingangsgrößen
ULe-4-S	A = Number of information elements and steps that are presented in a way that the user could understand B = Number of information elements and steps needed to complete common tasks for a first time user	user, understand	Kontext und Anwender
UOp-1-G	A = Number of specific interactive tasks that are performed inconsistently B = Number of specific interactive tasks that need to be consistent	specific	Anforderung
UOp-2-G	A = Number of messages that convey the right outcome or instructions to the user B = Number of messages implemented	user	Kontext und Anwender
UOp-3-S	A = Number of functions and operational procedures which can be customized for user's convenience B = Number of functions and operational procedures for which users could benefit from customization	user, benefit, operation	Kontext und Anwender, Betrieb
UOp-4-S	A = Number of user interface elements that can be customized B = Number of user interface elements that could benefit from customization	benefit, user	Kontext und Anwender
UOp-5-S	A = Number of functions having state monitoring capability B = Number of functions that could benefit from monitoring capability	monitoring, benefit	Programmier-richtlinien, Kontext und Anwender
UOp-6-S	A = Number of tasks that provide undo capability or prompt for re-confirmation B = Number of tasks for which users could benefit from having re-confirmation or undo capability	user, benefit	Kontext und Anwender
UOp-7-S	A = Number of information structures that are familiar and convenient for the intended users B = Number of information structures used	user	Kontext und Anwender
UOp-8-S	A = Number of user interfaces with similar items but with different appearances B = Number of user interfaces with similar items	user	Kontext und Anwender
UOp-9-S	A = Number of tasks that can be initiated by all appropriate input modalities B = Number of tasks supported by the system	system, appropriate	Umgebungsbedingungen, Anforderung
UEp-1-G	A = Number of user actions and inputs that are protected from causing any system malfunction B = Number of user actions and inputs that could be protected from causing any system malfunction	user, system	Umgebungsbedingungen, Kontext und Anwender
UEp-2-S	A = Number of entry errors for which the system provides a suggested correct value B = Number of entry errors detected	system	Umgebungsbedingungen

ID	Parameter Beschreibung	Schlagwörter	Eingangsgrößen
UEp-3-S	A = Number of user errors that are designed and tested to be recovered by the system B = Number of user errors which can occur during operation	user, system, operation	Kontext und Anwender, Umgebungsbedingungen, Betrieb
UIn-1-S	A = Number of display interfaces aesthetically pleasing to the users in appearance B = Number of display interfaces	user	Kontext und Anwender
UAc-1-G	A = Number of functions successfully usable by the users with a specific disability B = Number of functions implemented	specific	user
UAc-2-S	A = Number of languages actually supported B = Number of languages needed to be supported	-	-
RMa-1-G	A = Number of reliability-related faults corrected in design /coding/testing phase B = Number of reliability-related faults detected in design/coding/testing phase	-	-
RMa-2-G	A = Operation time B = Number of system/software failures actually occurred	operation, system, software	Betrieb, Umgebungsbedingungen
RMa-3-S	A = Number of failures detected during observation time B = Duration of observation	-	-
RMa-4-S	A = Number of system or software capabilities, operational scenarios or functions that are actually performed B = Number of system or software capabilities, operational scenarios or functions which are included in their associated test suites	system, software, scenario, operation	Umgebungsbedingungen, Kontext und Anwender, Betrieb
RAv-1-G	A = System operation time actually provided B = System operation time specified in the operation schedule	system, operation, specified	Umgebungsbedingungen, Anforderung, Betrieb
RAv-2-G	A = Total down time B = Number of breakdowns observed	-	-
RFt-1-G	A = Number of avoided critical and serious failure occurrences (based on test cases) B = Number of executed test cases of fault pattern (almost causing failure) during testing	-	-
RFt-2-S	A = Number of system components redundantly installed B = Number of system components	system, component	Umgebungsbedingungen
RFt-3-S	A <sub>i</sub> = Time at which the fault i is reported by the system B <sub>i</sub> = Time at which fault i is detected n = Number of faults detected	system	Umgebungsbedingungen
RRe-1-G	A <sub>i</sub> = Total time to recover the downed software /system and re-initiate operation for each failure i n = Number of failures	software, system, operation	Umgebungsbedingungen, Betrieb

ID	Parameter Beschreibung	Schlagwörter	Eingangsgrößen
RRe-2-S	A = Number of data items actually backed up regularly B = Number of data items requiring backup for error recovery	requiring	Anforderung
SCo-1-G	A = Number of confidential data items that can be accessed without authorization B = Number of data items that require access control	require	Anforderung
SCo-2-G	A = Number of data items encrypted/decrypted correctly B = Number of data items that require encryption/decryption	require	Anforderung
SCo-3-S	A = Number of cryptographic algorithms broken or unacceptably risky in use B = Number of cryptographic algorithms used	-	-
SIn-1-G	A = Number of data items which are actually corrupted by unauthorized access B = Number of data items for which data corruption or modification have to be prevented	modification	Anforderung
SIn-2-G	A = Number of data corruption prevention methods actually implemented B = Number of data corruption prevention methods available and recommended	-	-
SIn-3-S	A = Number of memory accesses with user input that are bounds checked B = Number of memory accesses with user input in software modules	user, software, module, memory	Kontext und Anwender, Umgebungsbedingungen
SNo-1-G	A = Number of events that ensure non-repudiation using digital signature B = Number of events requiring non-repudiation using digital signature	require, requiring	Anforderung
SAc-1-G	A = Number of accesses recorded in all logs B = Number of accesses to system or data actually tested	system	Umgebungsbedingungen
SAc-2-S	A = Duration for which the system log is actually retained in stable storage B = Retention period specified for keeping the system log in stable storage	system, specified	Umgebungsbedingungen, Anforderung
SAu-1-G	A = Number of authentication mechanisms provided (e.g., User ID/password or IC card) B = Number of authentication mechanisms specified	specified	Anforderung
SAu-2-S	A = Number of authentication rules implemented B = Number of authentication rules specified	specified	Anforderung
MMo-1-G	A = Number of components which are implemented with no impact on others B = Number of specified components which are required to be independent	other, specified, require, component	Umgebungsbedingungen, Anforderung

ID	Parameter Beschreibung	Schlagwörter	Eingangsgrößen
MMo-2-S	A = Number of software modules which have a cyclomatic complexity score that exceeds the specified threshold B = Number of software modules implemented	software, module, threshold, specified	Umgebungsbedingungen, Programmierrichtlinien, Anforderung
MRe-1-G	A = Number of assets which are designed and implemented to be reusable B = Number of assets in a system	system	Umgebungsbedingungen
MRe-2-S	A = Number of software modules conforming to coding rules for a specific system B = Number of software modules implemented	software, module, coding rules, system, specific	Umgebungsbedingungen, Programmierrichtlinien, Anforderung
MAAn-1-G	A = Number of logs that are actually recorded in the system B = Number of logs for which audit trails are required during operation	system, require, operation	Umgebungsbedingungen, Anforderung, Betrieb
MAAn-2-S	A = Number of diagnostic functions useful for causal analysis B = Number of diagnostic functions implemented	diagnostic	Programmierrichtlinien
MAAn-3-S	A = Number of diagnostic functions implemented B = Number of diagnostic functions required	diagnostic, require	Programmierrichtlinien, Anforderung
MMd-1-G	A <sub>i</sub> = Total work time spent for making a specific type of modification i B <sub>i</sub> = Expected time for making the specific type of modification i n = Number of modifications measured	specific, modification, expect	Anforderung, Kontext und Anwender
MMd-2-G	A = Number of modifications that caused an incident or failure within a defined period after being implemented B = Number of modifications implemented	modification	Anforderung
MMd-3-S	A = Number of items actually modified within a specified duration B = Number of items required to be modified within a specified duration	modified, specified, require	Anforderung
MTe-1-G	A = Number of test functions implemented as specified B = Number of test functions required	specified, require	Anforderung
MTe-2-S	A = Number of tests that can be simulated by stub among the tests which depend on other systems B = Number of tests which depend on other systems	other, system	Umgebungsbedingungen
MTe-3-S	A = Number of cases in which maintainer can pause and restart executing test run at desired points to check step by step B = Number of cases in which executing test run can be paused	desire	Kontext und Anwender

ID	Parameter Beschreibung	Schlagwörter	Eingangsgrößen
PAd-1-G	A = Number of functions which were not completed or results which were insufficient to meet requirements during testing B = Number of functions which were tested in different hardware environment	require, hardware, environment	Anforderung, Umgebungsbedingungen
PAd-2-G	A = Number of functions which were not completed or results which were insufficient to meet requirements during testing B = Number of functions which were tested in different system software environment	require, system, software, environment	Anforderung, Umgebungsbedingungen
PAd-3-S	A = Number of functions which were not completed or results which were insufficient to meet requirements during operational testing with user's environment B = Number of functions which were tested in different operational environment	require, user, environment, operation	Anforderung, Kontext und Anwender, Umgebungsbedingungen, Betrieb
PIn-1-G	A <sub>i</sub> = Total work time spent for making an installation i B <sub>i</sub> = Expected time for making an installation i n = Number of installations measured	installation, expect	Betrieb, Kontext und Anwender
PIn-2-G	A = Number of cases in which a user succeeds to customize the installation procedure B = Number of cases in which a user attempted to customize the installation procedure for user's convenience	user, installation	Kontext und Anwender, Betrieb
PRe-1-G	A = Number of user functions which can be performed without any additional learning or work-around B = Number of user functions in the replaced software product	user, software	Kontext und Anwender, Umgebungsbedingungen
PRe-2-S	A = Number of quality measures of the new product which are better or equal to the replaced product B = Number of quality measures of the replaced software product that are relevant	software	Umgebungsbedingungen
PRe-3-S	A = Number of functions which produce similar results as before B = Number of functions which have to be used in the replaced software product	software	Umgebungsbedingungen
PRe-4-S	A = Number of data which can be used continuously as before B = Number of data which are to be used continuously in the replaced software product	software	Umgebungsbedingungen

Tab. 34 – Ergebnisse der quantitativen Analyse der System- und Software Produkt Maße<sup>97</sup>

<sup>97</sup> Eigene Darstellung nach (Tab. 3; Tab. 29)

## H Kategorien messbarer Code Konventionen des Werkzeugs Checkstyle

Kategorie	Konventionen
Annotations	AnnotationLocation, AnnotationUseStyle, MissingDepecated, Missing Override, PackageAnnotation, SuppressWarnings, SuppressWarningsHolder
BlockChecks	AvoidNestedBlocks, EmptyBlock, EmptyCatchBlock, LeftCurly, NeedBraces, RightCurly
Class Design	DesignForExtension, FinalClass, HideUtilityClassConstructor, InnerTypeLast, InterfaceIsType, MutableException, OneTopLevelClass, ThrowsCount, VisibilityModifier
Coding	ArrayTrailingComma, AvoidInlineConditionals, CovariantEquals, DeclarationOrder, DefaultComesLast, EmptyStatement, EqualsAvoidNull, EqualsHashCode, ExplicitInitialization, FallThrough, FinalLocalVariable, HiddenField, IllegalCatch, IllegalInstantiation, IllegalThrows, IllegalToken, IllegalTokenText, IllegalType, InnerAssignment, MagicNumber, MissingCtor, MissingSwitchDefault, ModifiedControlVariable, MultipleStringLiterals, MultipleVariableDeclarations, NestedForDepth, NestedIfDepth, NestedTryDepth, NoClone, NoFinalizer, OneStatementPerLine, OverloadMethodsDeclarationOrder, PackageDeclaration, ParameterAssignment, RequireThis, ReturnCount, SimplifyBooleanExpression, SimplifyBooleanReturn, StringLiteralEquality, SuperClone, SuperFinalize, UnnecessaryParentheses, VariableDeclarationUsageDistance
Headers	Header, RegexpHeader
Imports	AvoidStarImport, AvoidStaticImport, CustomImportOder, IllegalImport, ImportControl, ImportOrder, RedundantImport, UnusedImports
Javadoc Comments	AtclauseOrder, JavadocMethod, JavadocPackage, JavadocParagraph, JavadocStyle, JavadocTagContinuationIntendation, JavadocType, JavadocVariable, NonEmptyAtclauseDescription, SingleLineJavadoc, SummaryJavadoc, WriteTag
Metrics	BooleanExpressionComplexity, ClassDataAbstractionCoupling, ClassFanOutComplexity, CyclomaticComplexity, JavaNCSS, NPathComplexity
Miscellaneous	ArrayTypeStyle, AvoidEscapedUnicodeCharacters, CommentsIndentation, DescendantToken, FileContentHolder, FinalParameters, Intendation, NewlineAtEndOfFile, OuterTypeFilename, TodoComment, TrailingComment, Translation, UncommentedMain, UniqueProperties, UpperEll
Modifiers	ModifierOrder, RedundantModifier
Naming Conventions	AbbreviationAsWordInName, AbstractClassName, CatchParameterName, ClassTypeParameterName, ConstantName, InterfaceTypeParameterName, LocalFinalVariableName, LocalVariableName, MemberName, MethodName, MethodTypeParameterName, PackageName, ParameterName, StaticVariableName, TypeName
Regexp	Regexp, RegexpMultiline, RegexpOnFilename, RegexpSingleline, RegexpSinglelineJava (vgl.)
Size Violations	AnonInnerLength, ExecuteableStatementCount, FileLength, LineLength, MethodCount, MethodLength, OuterTypeNumber, ParameterNumber
Whitespace	EmptyForInitializerPad, EmptyForIteratorPad, EmptyLineSeperator, FileTabCharacter, GenericWhitespaces, MethodParamPad, NoLineWrap, NoWhitespaceAfter, NoWhitespaceBefore, OperatorWrap, ParenPad, SeperatorWrap, SingleSpaceSeperator, TypecastParenPad, WhitespaceAfter, WhitespaceAround

Tab. 35 – Kategorien messbarer Code Konventionen des Werkzeugs Checkstyle <sup>98</sup>

<sup>98</sup> Eigene Darstellung nach ([60]; [61]; [62]; [63]; [64]; [65]; [66]; [67]; [68]; [69]; [70]; [71]; [72]; [73])



## I Kategorien von Metriken des Werkzeugs SonarQube

Kategorie	Metrik Bezeichnung	Beschreibung
Komplexität	Komplexität	Anzahl der Verzweigungen abhängig von der Programmiersprache.
	Kognitive Komplexität	Mathematisches Verfahren zur Bestimmung der Schwierigkeit des Quellcode Verständnisses.
	Komplexität /Klasse	Durchschnittliche Komplexität pro Klasse.
	Komplexität /Datei	Durchschnittliche Komplexität pro Datei.
	Komplexität /Methode	Durchschnittliche Komplexität pro Methode.
Dokumentation	Anzahl Kommentare	Anzahl auskommentierter Zeilen
	Kommentare in %	Kommentardichte im Quelltext berechnet durch die Anzahl der Kommentare im Verhältnis zur Summe von Kommentaren und Code Zeilen
	API Dokumentation	Dichte der API Dokumentation.
	Fehlende API Dokumentation	Öffentliche API ohne Kommentar Header.
	Auskommentierte LOC	Anzahl kommentierter Quellcode Zeilen.
Duplikate	Duplizierte Blöcke	Anzahl duplizierter Code Zeilen Blöcke.
	Duplizierte Dateien	Anzahl involvierter Dateien mit Duplikaten.
	Duplizierte Zeilen	Anzahl involvierter Zeilen mit Duplikaten.
	Duplizierte Zeilen in %	Dichte von Duplikaten berechnet aus der Anzahl duplizierter Zeilen im Verhältnis zu Gesamtzeilen.
Probleme	Neue Probleme	Anzahl neu identifizierter Probleme.
	Neue Probleme nach Schweregrad	Anzahl neu identifizierter Probleme mit vorgegebenen Schweregrad blockierend, kritisch, wichtig, unbedeutend oder informativ.
	Probleme	Anzahl offener Probleme.
	Probleme nach Schweregrad	Anzahl identifizierter Probleme mit vorgegebenen Schweregrad blockierend, kritisch, wichtig, unbedeutend oder informativ.
	Falsch positive Probleme	Anzahl falsch positiver Probleme.
	Offene Probleme	Probleme mit dem Status Open.
	Bestätigte Probleme	Probleme mit dem Status Confirmed.
	Wiedereröffnete Probleme	Probleme mit dem Status Reopened.
Schweregrad	Blockierend	Betriebs- oder Sicherheitsrisiko: Problem führt möglicherweise zur Instabilität des Produkts.
	Kritisch	Betriebs- oder Sicherheitsrisiko: Problem führt zu unerwartetem Verhalten des Produkts ohne die Integrität der Applikation zu beeinflussen.
	Wichtig	Problem hat erheblichen Einfluss auf die Produktivität.
	Unbedeutend	Problem hat potenziellen oder wenigen Einfluss auf die Produktivität.
	Information	Unbekannter/s oder noch nicht definiertes Sicherheitsrisiko / Einfluss auf Produktivität.
Wartbarkeit	Schlecht strukturierter Code	Anzahl schlecht strukturierten Quellcode.
	Neuer schlecht strukturierter Code	Anzahl von neuem schlecht strukturiertem Quellcode.
	Bewertung Wartbarkeit	Bewertung des technischen Schulden Verhältnisses.

Kategorie	Metrik Bezeichnung	Beschreibung
Wartbarkeit	Technische Schuld	Aufwand um alle Probleme mit Bezug zur Wartbarkeit zu beheben.
	Technische Schuld neuen Codes	Aufwand um alle Probleme mit Bezug zur Wartbarkeit von neuem Quellcode zu beheben.
	Technische Schuld Verhältnis	Verhältnis zwischen Entwicklungskosten der Software und Kosten um die technischen Schulden zu eliminieren.
	Technische Schuld Verhältnis neuen Codes	Verhältnis zwischen den Entwicklungskosten und den Kosten durch identifizierte Probleme.
Quality Gates	Quality Gate Status	Status von Quality Gates mit Bezug zur Software.
	Quality Gate Details	Fehlerhafte und korrekte Bedingungen des Quality Gates.
Zuverlässigkeit	Programmfehler	Anzahl von identifizierten Programmfehlern.
	Neue Programmfehler	Anzahl von neuen identifizierten Programmfehlern.
	Zuverlässigkeitsbewertung	Gruppierung von Programmfehlern nach Schweregrad.
	Zuverlässigkeitsspezifische Sanierungskosten	Aufwand alle Programmfehler zu beheben.
	Zuverlässigkeitsspezifische Sanierungskosten neuen Codes	Aufwand alle zuverlässigkeitsspezifischen Programmfehler zu beheben im Verhältnis zur potenziellen Ausfallzeit.
Sicherheit	Sicherheitsverletzungen	Anzahl identifizierter Sicherheitsverletzungen im Quellcode.
	Neue Sicherheitsverletzungen	Anzahl neu identifizierter Sicherheitsverletzungen im Quellcode.
	Sicherheitsbewertung	Gruppierung Sicherheitsverletzungen nach Schweregrad.
	Sicherheitsspezifische Sanierungskosten	Aufwand alle sicherheitsspezifischen Programmfehler zu beheben.
	Sicherheitsspezifische Sanierungskosten neuen Codes	Aufwand alle sicherheitsspezifischen Programmfehler zu beheben im Verhältnis zur potenziellen Ausfallzeit.
Ohne	Klassen	Anzahl der Klassen
	Verzeichnisse	Anzahl der Verzeichnisse
	Dateien	Anzahl der Dateien
	Zeilen	Anzahl der technischer Zeilen
	Codezeilen	Anzahl technischer Codezeilen, die mindestens ein Zeichen ungleich einem Leerzeichen oder Tabstop entsprechen.
	Codezeilen per Programmiersprache	Anzahl Codezeilen per verwendeter Programmiersprache
	Anzahl Methoden	Anzahl der programmiersprachenabhängigen Funktionen, Methoden oder Paragraphen.
	Anzahl Projekte	Anzahl der Projekte.
	Öffentliche API	Anzahl der öffentlichen Klassen, Funktionen und Eigenschaften.
	Statements	Anzahl der Statements.
Tests	Bedingungsüberdeckung	Anzahl der boolschen Ausdrücke, deren True und False Zustände evaluiert wurden.

Kategorie	Metrik Bezeichnung	Beschreibung
Tests	Bedingungsüberdeckung neuen Codes	Anzahl der boolschen Ausdrücke in neuem Code, deren True und False Zustände evaluiert wurden.
	Bedingungsüberdeckung Treffer	Anzahl abgedeckter Bedingungen im Quellcode.
	Bedingungen per Zeile	Anzahl Bedingungen pro Codezeile.
	Abgedeckte Bedingungen per Zeile	Anzahl abgedeckter Bedingungen pro Codezeile.
	Code Überdeckung	Anzahl überdeckten Quellcodes durch Modultests.
	Überdeckung neuen Codes	Anzahl überdeckten neuen und modifizierten Quellcodes durch Modultests.
	Zeilenüberdeckung	Anzahl der während eines Modultests ausgeführten Zeilen eines vordefinierten Quellcodeabschnitts.
	Zeilenüberdeckung neuen Codes	Anzahl der während eines Modultests ausgeführten Zeilen eines neuen oder modifizierten Quellcodeabschnitts.
	Zeilenüberdeckungstreffer	Anzahl überdeckter Code Zeilen.
	Anzahl Zeilen für Überdeckung	Anzahl an Code Zeilen die durch Modultests abgedeckt sind.
	Anzahl Zeilen für Überdeckung neuen Codes	Anzahl abgedeckter neuer oder modifizierter Code Zeilen durch Modultests.
	Übersprungende Modultests	Anzahl Modultests die nicht ausgeführt wurden.
	Nicht abgedeckte Bedingungen	Anzahl nicht abgedeckter Bedingungen durch Modultests.
	Nicht abgedeckte Bedingungen neuen Codes	Anzahl nicht abgedeckter Bedingungen neuen oder modifizierten Quellcodes durch Modultests.
	Nicht abgedeckte Zeilen	Anzahl von Quellcode Zeilen, die nicht durch Modultests abgedeckt sind.
	Nicht abgedeckte Zeilen neuen Codes	Anzahl von neuen oder modifizierten Quellcode Zeilen, die nicht durch Modultests abgedeckt sind.
	Modultest	Anzahl von Modultests.
	Modultest Dauer	Benötigte Zeit, um alle Modultests auszuführen.
	Modultest Error	Anzahl fehlgeschlagender Modultests.
	Modultest Fehler	Anzahl der mit einer unerwarteten Exception fehlgeschlagenen Modultests.
	Erfolgreiche Modultests %	Dichte der erfolgreichen Modultests.

Tab. 36 – Kategorien von Metriken des Werkzeugs SonarQube<sup>99</sup>

<sup>99</sup> Eigene Darstellung nach ([74]; [28])