Stuttgarter Schriften zur Unternehmenssoftware
Nummer 6

SOFTWARE-QUALITÄTSMANAGEMENT 4.0

Qualitätsmanagement in Zeiten des digitalen Wandels

Carina Klumpp
Georg Herzwurm

Stuttgart, Januar 2020

Lehrstuhl für ABWL und Wirtschaftsinformatik II

Universität Stuttgart

AUTOREN
Carina Klumpp
Prof. Dr. Georg Herzwurm

TITEL

SOFTWARE-QUALITÄTSMANAGEMENT 4.0

Qualitätsmanagement in Zeiten des digitalen Wandels

Stuttgarter Schriften zur Unternehmenssoftware
Nummer 6
Universität Stuttgart, 2020

ISSN: 1865-388X

IMPRESSUM

Universität Stuttgart
Betriebswirtschaftliches Institut
Abteilung für ABWL und Wirtschaftsinformatik II
(Software-intensive Business)
Keplerstraße 17
70174 Stuttgart
Tel.: +49 (0)711/685-82385

http://www.wius.bwi.uni-stuttgart.de

GESTALTUNG
Carina Klumpp

BILDNACHWEIS
everythingispossible – fotalia.com (Titelblatt)

SOFTWARE-QUALITÄTSMANAGEMENT 4.0

Qualitätsmanagement in Zeiten des digitalen Wandels

Abstract

Durch die fortschreitende Digitalisierung und wachsende Bedeutung der IT in allen Branchen werden in Zukunft erhöhte Anforderungen an das Software-Qualitätsmanagement gestellt werden. Qualitätsmängel in softwareintensiven Konzepten wie Industrie 4.0 und autonomem Fahren bergen nicht nur hohe Risiken für den geschäftlichen Erfolg, sondern auch für Gesundheit und Sicherheit. Diese Studie untersucht den Stand von Forschung und Praxis im Bereich Software-Qualitätsmanagement und gibt eine Einschätzung, inwieweit vorliegende Ansätze den Anforderungen neuer software-intensiver Systeme genügen.

Dazu werden 53 ausgewählte wissenschaftliche Journals und Konferenzen mittels strukturierter Literaturanalyse untersucht. Für den praktischen Schwerpunkt werden Beiträge in 36 IT-Zeitschriften, die Schwerpunkte von 11 Praktiker-Konferenzen und 35 Stellenanzeigen untersucht.

Die Ergebnisse zeigen eine Diskrepanz zwischen Wissenschaft und Praxis. Schien SQM zunächst ein stark praxisgetriebenes Thema, hat die Forschung inzwischen aufgeholt und stellt Ansätze zur Verfügung, die in der Praxis nicht vollständig nachgefragt werden. Das Forschungsinteresse konzentriert sich auf Anforderungserhebung und Anwendung von künstlicher Intelligenz für Software-Qualitätsmanagement. Modellbasierte Entwicklung und Testen spielen in Wissenschaft und Praxis gegenwärtig eine zentrale Rolle. Praktische Schwerpunkte sind die Anwendung agiler Modelle und die Testautomatisierung. Wertschöpfungskettenübergreifendes vernetztes „Software-Qualitätsmanagement 4.0“ wird sich zukünftig mit der Komplexität in und zwischen Systemen, Sicherheitsaspekten, künstlicher Intelligenz und neuen Technologien auseinandersetzen müssen. Vorhandene Maßnahmen, Werkzeuge und Ansätze sowie deren Automatisierung und Integration werden Forschung und Praxis vor neue Herausforderungen stellen.

Inhalt

Abkürzungsverzeichnis

CMM Capability Maturity Model

QM Qualitätsmanagement

SQ Softwarequalität

SQM Software-Qualitätsmanagement

TQM Total Quality Management

SIS software-intensives System

SPI Software Process Improvement

SW Software

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

1.1 Problemstellung und Motivation

Seit inzwischen 30 Jahren belegen empirische Studien aus unterschiedlichen Branchen die hohe Relevanz von Qualität für den wirtschaftlichen Erfolg von Unternehmen.[1] Dies gilt auch für Software-intensive Produkte. Softwarespezifische Untersuchungen über die Einführung des Qualitätsmanagements (QM) belegen, dass mit Software-Qualitätsmanagement (SQM) nicht nur der Unternehmenserfolg steigt, sondern auch Produktivität in der Softwareentwicklung und Kundenzufriedenheit verbessert werden.[2]

Software wird heute nicht mehr nur zur Automatisierung und Rationalisierung von Prozessen eingesetzt, sondern findet sich als essentieller Bestandteil vieler Produkte wieder. Sogenannte Smart Products und Services bilden die Grundlage für neue Software-intensive Geschäftsmodelle.[3] Diese Entwicklung wird durch das Internet der Dinge (Internet of Things) weiter gefördert.[4] Beratungsunternehmen gehen davon aus, dass in den nächsten Jahren mehr als 20 Milliarden Endgeräte weltweit über das Internet verbunden werden.[5] Um ein reibungsloses Zusammenspiel der Endgeräte zu gewährleisten, werden Software-Qualitätssicherungsmaßnahmen benötigt.[6] Es kann daher davon ausgegangen werden, dass Software-Qualitätsmanagement auch in Zukunft relevant sein wird.

1.2 Stand der Forschung

Zu Beginn des zwanzigsten Jahrhunderts wuchs die Bedeutung der wissenschaftlich fundierten Qualitätskontrolle. In der zweiten Hälfte des Jahrhunderts fand ein Umschwung zur strukturierten „Quality Assurance“ bzw. Qualitätssicherung statt, bis sich in den 1990er Jahren der ganzheitliche Managementansatz des Total Quality

[1] Vgl. Buzzell u.a. (1989);Easton, Jarrell (1998);Giebel (2011);Große Böckmann (2011);Kaynak (2003);Rommel (1995).
[2] Vgl. Herbsleb u.a. (1994), S.15; Gibbs (1994), S.90; Jones, Bonsignour (2012), S.18; Nikolik (2012), S.1237.
[3] Vgl. Porter, Heppelmann (2014).
[4] Vgl. Geisberger, Broy (2012), S.21f.
[5] Vgl. IOT Analytics (2018), URL siehe Literaturverzeichnis;Gartner Inc. (2017), URL siehe Literaturverzeichnis.
[6] Vgl. Udoh, Kotonya (2018), S.71f.

Managements (TQM) durchgesetzt hat.[7] Dieser gilt als letzte große Entwicklung im Qualitätsmanagement.[8]

Im Bereich der Softwareentwicklung standen nach der „Qualitätswelle“ in den 90er Jahren durch TQM, ISO 9000, Capability Maturity Model Integration (CMMI)[9], in den späteren Jahren eher innovative und agile Ansätze wie Scrum[10], Design Thinking[11] und Extreme Programming[12] im Vordergrund.

Das allgemeine Interesse an Software-Qualitätsmanagement hat in den letzten 20 Jahren abgenommen.[13] Erfahrungen aus der Vergangenheit zeigen jedoch, dass die Aufmerksamkeit bezüglich der Qualität eng mit der Produktreife zusammenhängt. Zu Einführungsbeginn neuer Technologien steht die Neuartigkeit im Vordergrund. Produktfehler und Mängel werden vom Kunden akzeptiert, weil der neue Zusatznutzen die Qualitätsdefizite überwiegt. Mit der im Zeitverlauf des Lebenszyklus wachsenden Abhängigkeit vom Produkt steigt die Bedeutung der Qualität für den Kunden.[14] Die fortschreitende Digitalisierung und zunehmende Bedeutung der IT für alle Branchen werden somit in Zukunft ein noch stärkeres Umdenken erfordern: Bei softwareintensiven Konzepten wie Industrie 4.0[15] und autonomem Fahren[16] bergen Mängel in der Software-Qualität nicht nur hohe Risiken für den geschäftlichen Erfolg, sondern auch für Gesundheit und Sicherheit.

1.3 Zielsetzung und Forschungsfragen

Ziel dieser Arbeit ist es, den Stand von Forschung und Praxis im Bereich Software-Qualitätsmanagement zu analysieren. Anschließend soll eine Einschätzung getroffen

---

[7] Vgl. Sommerlatte (2007).

[8] Vgl. American Society for Quality (2019), URL siehe Literaturverzeichnis.

[9] Vgl. Software Engineering Institute (2002).

[10] Vgl. Cockburn (2001);Schwaber, Beedle (2002).

[11] Vgl. Brown (2008).

[12] Vgl. Beck, Andres (2005).

[13] Diese Entwicklung zeigt sich beispielsweise an der Zahl der veröffentlichten Bücher zum Stichwort „software quality management“[Google LLC (2019a), URL siehe Literaturverzeichnis]. Die meisten Bücher zum Thema wurden 1995 veröffentlicht. Seitdem ist ein nahezu linearer Rückgang der Veröffentlichungszahlen bis zum Jahr 2008 zu erkennen. Im Jahr 2008 enden die verfügbaren Daten von Google Books. Jedoch sind ab 2004 Daten von Google Trends verfügbar, die den weiteren Rückgang der Suchanfragen zum Thema bis 2019 bestätigen. [Google LLC (2019b), URL siehe Literaturverzeichnis].

[14] Vgl. Mellis u.a. (1996), S.7.

[15] Vgl. Fortmann, H. R. und Kolocek, B. (2019), S.305.

[16] Vgl. Ritz (2018), S.201.

werden, inwieweit die vorliegenden Ansätze den Anforderungen an Softwarequalität und deren Management von agilen und digitalen Geschäftsfeldern genügen.

Hierzu sind im einzelnen folgende Forschungsfragen zu beantworten

FF1: Wie ist der Stand der Forschung im Bereich Software-Qualitätsmanagement?

FF2: Wie ist der Stand der Praxis im Bereich Software-Qualitätsmanagement?

Sowohl bei FF1 als auch bei FF2 sind zeitliche sowie thematische Tendenzen bzw. Schwerpunkte von Interesse.

FF3: Welche Defizite ergeben sich aus dem Stand der Forschung und Praxis für die Herausforderungen des Software-Qualitätsmanagement in Zeiten des Software-intensive Business?

1.4 Wissenschaftliche Vorgehensweise

Die Forschungsfragen FF1 und FF 2 werden anhand strukturierter Literaturanalysen und Internet-Recherchen beantwortet.

Der Stand der Forschung ergibt sich aus dem Stand der wissenschaftlichen Beiträge in renommierten Zeitschriften und Konferenzproceedings.

Der Stand der Praxis wird anhand populärwissenschaftlicher Veröffentlichungen und Praktikerkonferenzen untersucht. Ein weiterer Indikator für die Praxisrelevanz sind veröffentlichte Stellenausschreibungen im Bereich SQM.

Die Untersuchung der FF3 erfolgt durch den Abgleich der Herausforderungen des software-intensive Business bei der Entwicklung software-intensiver Systeme (SIS) mit dem Stand im SQM auf der Basis von Wissenschaft und Praxis. Hierbei werden ausgewählte Themen gegebenenfalls näher untersucht mit dem Ziel, Handlungsempfehlungen für Wissenschaftler und Praktiker zum Abbau der Defizite zu liefern.

2 Wissenschaftliche Grundlagen

2.1 Begriff der Qualität

Die Formalisierung der Begrifflichkeit ist auf Grund der fehlenden unmittelbaren Mess- und Beobachtbarkeit der Qualität von großer Bedeutung. Erst die Strukturierung des Qualitätsbegriffs erlaubt wissenschaftliche Untersuchungen.[17]

Das Wort Qualität kommt vom Lateinischen Qualis und bedeutet „wie beschaffen", Qualitas wird mit Beschaffenheit übersetzt. Der Begriff wird in Abhängigkeit vom Anwendungszusammenhang unterschiedlich definiert.[18]

Die nachfolgende Tabelle beinhaltet in Anlehnung an Wallmüller (2011) eine Auflistung von Definitionen des Qualitätsbegriffs bekannter Qualitätsexperten. Die Tabelle ist ohne Anspruch auf Vollständigkeit und stellt eine Basis für aufbauende Definitionen dar.

Autor	**Qualitätsdefinition**
Philip B. Crosby	Qualität wird nach Crosby als **Erfüllung von Anforderungen** definiert und nicht als Güte. Qualitätsmaßstab sind dabei die Kosten der Nichterfüllung von Anforderungen. Der Grenzwert von Leistung ist „Null-Fehler". Das QM-System beschäftigt sich mit Problemvermeidung und Sicherstellung der sofort fehlerfreien Produktion.
Edwards W. Deming	Schwerpunkt in Demings Definition sind statistische Methoden zur Abweichungserfassung mit dem **Ziel der Ursachenanalyse und Prozessverbesserung**. Seine 14 Managementprinzipien fokussieren unternehmenskulturelle Faktoren beim QM. Problemlösung und Verbesserung werden mit dem PDCA-Zyklus realisiert.
Armand V. Feigenbaum	Qualität bedeutet nach Feigenbaum **Nutzerzufriedenheit**. Produkte und Dienstleistungen erfüllen die Verbrauchererwartungen. Die Qualität wird vom Kunden bestimmt. Im Rahmen der Total Quality Control ist jeder Mitarbeiter für Qualität verantwortlich.
Kaoru Ishikawa	Das japanische Qualitätsverständnis stützt sich auf folgende Grundsätze: Der Verbraucher definiert die Qualität, Qualitätsziele haben Vorrang für die Unternehmensleitung unter Einbeziehung der Mitarbeiter in die **kontinuierliche Verbesserung**. Dafür stehen sieben Qualitätswerkzeuge (wie sog. Ishikawa-Diagramm) zur Verfügung.
Joseph M. Juran	Nach Juran hat Qualität zwei Bedeutungen: Qualität bezeichnet die Produktbesonderheiten zur Befriedigung der Kundenbedürfnisse sowie Erhöhung der **Kundenzufriedenheit**. und die **Fehlerfreiheit**, die keine Nacharbeit benötigt, Ausfälle bewirkt oder zu unzufriedenen Kunden führt. Bei beiden Bedeutungen steht die Gebrauchstauglichkeit, die sog. „fitness for use" im Mittelpunkt.
W.A. Shewhart	Nach Shewhart ist Qualität gegeben, wenn **Zufriedenheit** erreicht wird. Verbraucherbedürfnisse werden anhand quantifizierbarer

[17] Vgl. Bächle (1996), S.135f; Lehner (1999), S.19.
[18] Vgl. Zollondz (2006), S.9.

	Steuerungsgrößen für die Produktentwicklung definiert. Prozesse werden mit statistischen Methoden und der Prozessregelkarte optimiert.
Genichi Taguchi	Taguchi definiert Qualität als den **Verlust**, den ein Produkt für die Gemeinschaft nach seiner Bereitstellung verursacht, im Gegensatz zu jenen Verlusten, die durch seine eigentlichen Funktionen hervorgerufen werden.

Tabelle 1: Verschiedene Definitionen des Qualitätsbegriffs (in Anlehnung an Wallmüller (2011), S.6f)

Das Verständnis des Qualitätsbegriffs wurde durch die Arbeiten von Deming, Crosby, Garvin und Juran geprägt. Standards und Normen sollen die Verbindlichkeit und ein einheitliches Verständnis des Begriffs fördern. Die ISO 9000 definiert Qualität als Grad, in dem ein Satz inhärenter Merkmale bestimmte Anforderungen erfüllt. Diese Anforderungen sind als ständiges Merkmal festgelegt und objektiv messbar.[19]

Eine Möglichkeit zur Strukturierung der vielfältigen Interpretationen bieten die von Garvin auf Basis einer Partialanalyse entwickelten fünf Ansätze der Qualitätsvorstellung:

„transcendent“ Transzendenter Ansatz	Die Qualität ist universell erkennbar, absolut und vollkommen. Es herrschen kompromisslos hohe Standards und Ansprüche an die Funktionsweise des Produktes. Qualität kann nur durch Erfahrungen und nicht durch Messungen bewertet werden.
„product-based“ Produktbezogener Ansatz	Qualität ist die präzise messbare und inhärente Eigenschaft eines Produktes. Qualitätsdifferenzen spiegeln Unterschiede in der beobachtbaren Quantität bestimmter Eigenschaftsausprägungen eines Produktes wieder. Subjektive Wahrnehmungen werden vernachlässigt und Rangordnungen können erstellt werden.
„user-based“ Anwenderbezogener Ansatz	Hierbei entschiedet der Nutzer eines Produktes nach seinen persönlichen Präferenzen über die Qualität. Die Qualität beruht auf einem rein kundensubjektiven Bewertungszusammenhang.
„manufacturing-based“ Prozessbezogener Ansatz	Qualität ist gleichzusetzten mit der Einhaltung exakter Spezifikationen und Kontrolle des Herstellungsprozesses, um Nacharbeiten zu minimieren und Kundenwünsche schnell umzusetzen. Der Automatisierungsgrad spielt eine große Rolle.
„value-based“ Kosten/Nutzen-bezogener Ansatz	Ein Qualitätsprodukt ist ein demnach ein Erzeugnis, das einen bestimmten Nutzen oder die Erfüllung vorgegebener Spezifikationen zu akzeptablen Kosten erbringt. Qualität ist eine relative Größe in Abhängigkeit von wahrgenommener Qualität und Kosten.

[19] Vgl. Norm DIN EN ISO 9000:2015-11, S.39.

Tabelle 2: Ansätze der Qualitätsvorstellungen nach Garvin[20]

Die unterschiedlichen Ansätze des Qualitätsbegriffs zeigen, dass kein einheitliches Begriffsverständnis existiert. Um resultierenden Problemen zu begegnen hat Garvin folgende acht abgegrenzte Kategorien der Qualität identifiziert. Sie kann über diese Dimensionen vollständig strukturiert werden:[21]

- Leistung (Performance): Die Leistungen beinhalten den Leistungsumfang einer Einheit[22] wie primäre Merkmale, die sich auf Betrieb oder Nutzung beziehen.
- Features (Features): Unter Features werden ergänzende oder besondere Leistungen subsummiert. Sie ergänzen die primären Leistungen der Einheit.
- Zuverlässigkeit (Reliability): Die Zuverlässigkeit einer Einheit stellt die Wahrscheinlichkeit von Fehlern, Defekten oder Mängeln innerhalb eines definierten Zeitraums dar. Diese Dimension ist bei langlebigen Investitionsgütern relevant.
- Konformität (Conformance): Der Übereinstimmungsgrad zwischen den Merkmalsausprägungen einer Einheit und a priori festgelegten Anforderungen wird als Konformität bezeichnet.
- Beständigkeit (Durability): Die Dimension der Beständigkeit kann mit dem Nutzen einer Einheit bis zu deren definitivem Verschleiß ohne Reparaturmöglichkeit beschrieben werden.
- Instandhaltbarkeit (Serviceability): Diese Qualitätsdimension bewertet die Wartung und Reparatur einer Einheit. Dabei sind bspw. Geschwindigkeit und Kompetenz bei der Reparatur bedeutsam.
- Ästhetik (Aesthetics): Diese Dimension beinhaltet die Bewertung der äußeren Erscheinung einer Einheit wie Aussehen, Form, Beschaffenheit. Ästhetik ist eng mit der wahrgenommenen Qualität verbunden.
- Wahrgenommene Qualität (Perceived quality): Die wahrgenommene Qualität umfasst Auffassung oder Empfindung der Qualität. Bei dieser Einschätzung sind Marken, Werbung und Reputation von Bedeutung.

[20] Vgl. Garvin (1988), S.40f.
[21] Vgl. Garvin (1988), S49f.
[22] Die Einheit wird benötigt, um den Bezugspunkt der Qualität darzustellen. Eine Einheit kann einzeln beschrieben und betrachtet werden und ist materieller oder immaterieller Gegenstand der Qualitätsbetrachtung. (Vgl. Zollondz (2006), S.170f.)

Um den Qualitätsbegriff anwenden zu können, sind weitere Definitionen erforderlich, im Rahmen dieser Arbeit wird keine eigene Definition erarbeitet. Innerhalb eines Softwareentwicklungsprozesses müssen Begriffe wie Qualitätseigenschaften, Qualitätsmerkmale und Qualitätsanforderungen einheitlich definiert sein. Diese Bezeichnungen sind nachstehend erläutert:[23]

- Die Qualitätseigenschaft dient der subjektiven oder objektiven Unterscheidung von Elementen (z. B. Produkten oder Prozessen).
- Qualitätsmerkmale sind Qualitätseigenschaften, die im konkreten Fall relevant sind und als Entscheidungsgrundlage dienen. Eine Elementeigenschaft wird also bei Nutzung zur Qualitätsbeurteilung zum Merkmal des Elements. Ein Merkmal wird in der ISO 9000:2005 daher als kennzeichnende Eigenschaft bezeichnet.
- Qualitätsanforderungen beschreiben die Anforderungen an die Ausprägungen von Qualitätsmerkmalen. Sie können festgelegt, üblicherweise vorausgesetzt oder verpflichtend sein. Das Kano-Modell unterteilt Kundenanforderungen in drei Ebenen: Die Basisanforderungen müssen zwingend erfüllt werden und führen nur zu „Nichtunzufriedenheit“. Leistungsanforderungen hingegen sind Soll- Anforderungen. Sie führen bei Erfüllung zu Zufriedenheit und bei Nichterfüllung zu Unzufriedenheit. Begeisterungsanforderungen werden vom Kunden nicht explizit erwartet, haben aber einen positiven Einfluss auf seine Qualitätswahrnehmung.[24]

2.2 Das Qualitätsmanagement

„Qualitätsmanagement ist Management bezüglich Qualität“ definiert die ISO 9000 prägnant. Zollondz definiert Qualitätsmanagement als Lehre der Führung von Organisationen bezüglich Qualität, in die auch Mitarbeiter unterer Hierarchieebenen eingebunden sind. Schwerpunktaufgaben sind Gestaltung, Lenkung und Entwicklung von Systemen durch qualitätsbezogene Tätigkeiten der Prozessplanung, -organisation, -durchsetzung und -kontrolle.[25]

[23] Vgl. Liggesmeyer (2009), S.5f.
[24] Vgl. Hinterhuber, Matzler (2009), 19f, S.15.
[25] Vgl. Zollondz u.a. (2016), S.931.

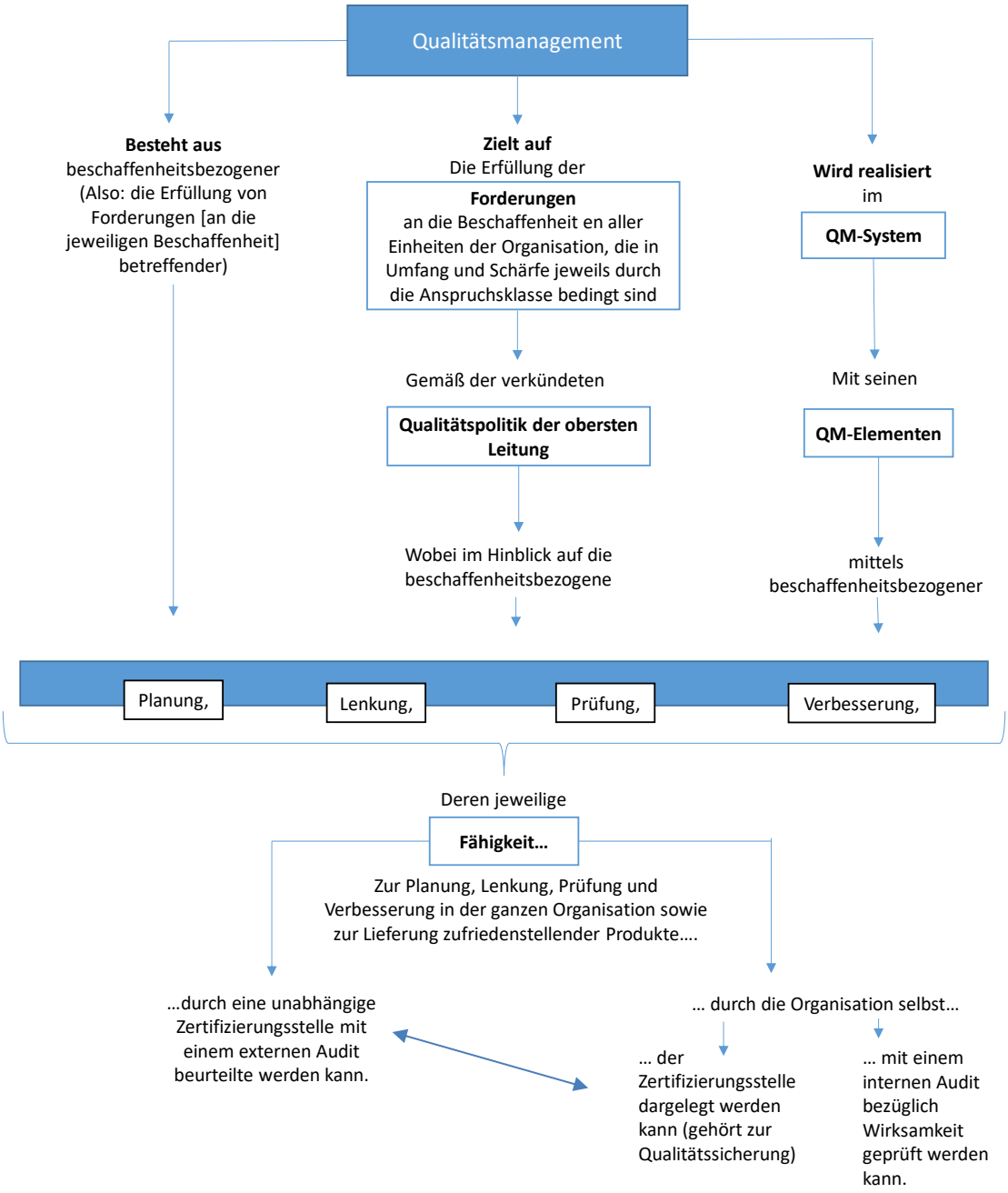

Abbildung 1: Übersicht über das Qualitätsmanagement als Managementaufgabe (Vgl. Geiger, Kotte (2008))

Die Managementaufgaben werden in einem QM-System mit seinen QM-Elementen realisiert. Wie in Abbildung 1 zu sehen sind die wichtigsten Elemente qualitätsbezogene Planung, Lenkung, Prüfung und Verbesserung, die auf der Qualitätspolitik des Topmanagements beruhen. Die qualitätsbezogene Planung dient der Festlegung der Qualitätsziele und der dafür notwendigen Ausführungsprozesse und Ressourcen. Die qualitätsbezogene Lenkung ist direkt auf die Erfüllung der beschaffenheitsbezogenen Forderungen gerichtet und umfasst vorbeugende, überwachende und korrigierende Tätigkeiten. Die qualitätsbezogene Prüfung ermittelt,

inwieweit Qualitätsanforderungen in allen Bereichen des Qualitätsmanagements und im gesamten Produktlebenszyklus erfüllt sind. Die qualitätsbezogene Verbesserung steigert die Effizienz des Qualitätsmanagements. Während die Qualitätssteigerung mit einer Ausweitung und Verschärfung der Anforderungen zusammenhängt, werden in der Qualitätserhöhung durch umfassendes Qualitätsmanagement bewirkte Verbesserungen einbezogen[26]. Elemente des heutigen Qualitätsmanagements haben sich sukzessive in den einzelnen Epochen der neueren Wirtschaftsgeschichte entwickelt.[27]

In der Praxis werden bei kontinuierlichen Techniken Entwicklungsprozessprobleme einer Organisation in wiederholenden Verbesserungszyklen betrachtet.[28] Diese werden an die konkreten Organisationsbedürfnisse angepasst und sind selbstgesteuert. Beispiele sind Plan-Do-Check-Act (PDCA), Quality Improvement Paradigm (QIP), Goal Question Metric (GQM), Total Quality Management (TQM) und Six Sigma.

Über einen Vergleich eines Referenzmodells auf Basis erfahrungsbasierter Anforderungen mit dem aktuellen Zustand einer Organisation wird mit modellbasierten Techniken die Produkt- oder Prozessqualität bewertet. Beispiele sind hier ISO 9126, ISO 12207, Capability Maturity Model Integration (CMMI) und IT Infrastructure Library (ITIL).[29] Modellbasierte Techniken identifizieren Verbesserungsbereiche, kontinuierliche Techniken ergänzen diese und implementieren Verbesserungsmöglichkeiten.[30]

2.3 Software-Qualität

Nach allgemeingültigen Qualitätsdefinitionen setzt sich dieses Kapitel mit der Softwarequalität als konkretem Anwendungsfall auseinander. In einem ersten Schritt wird auf die Besonderheiten von Software als Qualitätseinheit eingegangen.

---

[26] Vgl. Geiger, Kotte (2008), 4f, S.4f.

[27] Vgl. Zollondz u.a. (2016);Geiger, Kotte (2008), 4f; Geiger, Kotte (2008), S.4f.

[28] Verbesserungszyklen bestehen meist aus Problem- und Potenzialanalyse, Definition quantifizierbarer Qualitätsziele, Bestandsaufnahme der derzeitigen Situation, Auswahl, Anpassung und Anwendung von Verbesserungsmaßnahmen und Messung des Grades der resultierenden Verbesserung.(Vgl. Liggesmeyer (2009), S.11ff.)

[29] Vgl. Liggesmeyer (2009), S.11ff.

[30] Vgl. Liggesmeyer (2009), S.13.

Eigenschaft	Erläuterung
Immaterialität	Software ist immateriell und bedarf graphischer Hilfsmittel, um ihre Eigenschaften darzustellen.[31]
Eingeschränkte Wahrnehmbarkeit	Digitale Güter können nur über Sehen und Hören wahrgenommen werden. Die fehlende physische Wahrnehmbarkeit führt zu einer erschwerten Vergleichbarkeit.
Nicht-Abnutzbarkeit	Digitale Güter unterliegen keinerlei Abnutzung; die Unterscheidung zwischen neuem und altem Gut entfällt.
Einfache Reproduzierbarkeit	Digitale Güter werden bei Weitergabe vermehrt, nicht aufgeteilt. (Hohe Fixkosten für Entwicklung, geringe variable Kosten)[32]
Einfache Veränderbarkeit	Digitale Güter können ohne großen Aufwand in Produktvarianten überführt und angeboten werden. Folgen von Änderungen sind schwerer nachvollziehbar[33]
Eingeschränkte Fehleridentifikation und -bearbeitbarkeit	Softwarefehler sind schwerer identifizierbar. Software hat eine sehr hohe Produktkomplexität. Softwareprobleme und resultierende Entwicklungsaufgaben lassen sich nicht beliebig zwischen Entwicklern aufteilen. [34]
Diffizile Bewertung Entwicklungsfortschritt	Der Entwicklungsfortschritt eines Softwareprojektes lässt sich objektiv schwer ermitteln.
Softwareentwicklung nicht deterministisch	Softwareentwicklung verläuft nicht deterministisch: Erkenntnisse während der Entwicklung haben Auswirkungen auf bisherige Ergebnisse. [35]

Die im Kapitel 2.1 aufgeführte Qualitätsdefinitionen sind auch auf Software als Element des QM anwendbar. Ähnlich wie beim Qualitätsbegriff sind unter dem Begriff der Softwarequalität teilweise widersprüchliche Definitionen subsummiert.[36] Aus den allgemeinen Qualitätsdefinitionen nach Deming, Juran und Crosby wurde die in der ISO 9126[37] standardisierte Begriffsbestimmung abgeleitet.[38] Demnach kann unter dem

[31] Vgl. Thaller (2000), 13ff
[32] Vgl. Urbach (2017), S.39f.
[33] Vgl. Thaller (2000), S.13ff.
[34] Vgl. Galin (2004), 4f
[35] Vgl. Balzert, Ebert (2008), S.151f.
[36] Vgl. Balzert (1998), S.257; Ferstl, Sinz (2006), S.473; Heinrich, Lehner (2005), S.140; Mertens, P. und Back, A. (2001), S.422; Stahlknecht, Hasenkamp (2005), S.313.
[37] Weiterentwickelter aktueller Stand 2019: ISO 25000:2014
[38] Vgl. Balzert, Ebert (2008), 151f

Begriff Softwarequalität die Gesamtheit der Funktionen und Merkmale eines Softwareproduktes, die festgelegte oder implizit vorausgesetzte Bedürfnisse befriedigen, verstanden werden.[39] Dabei wird zwischen der Sicht des Anwenders (externe Aspekte), des Entwicklers (interne Qualität und Zwischenprodukte) und des Managers (Gesamteindruck) unterschieden.[40]

Die IEEE-Norm 727-1983 charakterisiert Software-Qualität durch:

- Die Gesamtheit der Funktionalitäten und Merkmale eines Softwareprodukts, die gegebene Bedürfnisse befriedigen, wenn sie Spezifikationen entsprechen.
- Das Ausmaß der gewünschten Eigenschaftskombinationen der SW.
- Das Ausmaß, in dem Kunde und Anwender erkennen, dass die Software den Erwartungen entspricht.
- Die zusammengesetzten Softwaremerkmale, die bestimmen, bis zu welchem Grad die Software den Kundenerwartungen gerecht wird.[41]

Diese Definition deckt sich mit dem produktbezogenen und dem anwenderbezogenen Ansatz von Garvin (siehe Kapitel 2.1). Wallmüller (2011) erweitert die Definitionen der beiden Normen um den prozessorientierten Ansatz und definiert Software-Qualität als *Summe aller relevanten Eigenschaften eines Software-Produkts zur Zufriedenstellung des Kunden sowie die Summe aller dafür notwendigen Eigenschaften von Software-Prozessen*.[42] Diese Definition soll auch im weiteren Verlauf der Arbeit relevant sein. Vorteil dieser Definition ist, dass neben der Zufriedenstellung des Kunden durch Produktqualität auch der Gewinn des Unternehmens durch Prozessqualität gesichert wird.[43] Diese Definitionserweiterung erfordert einen Bewertungsansatz für Entwicklungs- und Pflegeprozesse. Dafür müssen Bewertungsmaßstäbe für Zwischen- und Endprodukte bzw. deren Entwicklungsaktivitäten in allen Prozess-phasen entwickelt werden.[44]

Software-Qualitätsdefinitionen innerhalb der Fachdiskussionen reichen nicht für praktische Anwendungen. Merkmale der Software-Qualität werden deshalb in sog.

[39] Vgl. Wallmüller (2011), S.9.
[40] Vgl. Liggesmeyer (2009), S.16.
[41] Vgl. IEEE Standards Board (1983), S.38.
[42] Vgl. Wallmüller (2011), S.10.
[43] Vgl. Pfeifer (2002), S.75.
[44] Vgl. Wallmüller (2011), S.11.

Qualitätsmodellen konkretisiert. Dabei wird der allgemeine Qualitätsbegriff durch Ableiten von (Sub-)Merkmalen operationalisiert.[45] Ein Qualitätsmodell[46] beschreibt kausale Zusammenhänge zwischen konkreten Sichten[47] auf Qualität und Softwaremerkmalen. Es setzt sich aus verbundenen und hierarchisch geordneten Qualitätsaspekten zusammen, die schlussendliche Softwaremerkmale definieren.[48]

Modelle, die gemäß der Unterscheidung in Prozess- und Produktqualität auf den Entwicklungsprozess einer Software bzw. auf die Software selbst ausgerichtet sind, heißen Prozess- bzw. Produktmodelle. Produktmodelle ermöglichen die Darstellung differenzierter Sichten auf das Produkt und die Durchführung von Produkt-Assessments. Dabei werden die Produktmerkmale über Qualitätsfaktoren, auch Attribute oder Charakteristiken genannt, modelliert. Bekannte Modelle sind jene von McCall[49], Boehm[50], Willmer[51], NEC[52] oder das FURPS-Modell[53]. Prozessorientierte Qualitätsmodelle beschreiben Prozesseigenschaften, die für Software-Entwicklung und -Management relevant sind. Dafür werden Best-Practice Ansätze mit Bewertungsansätzen und Methoden für die Prozessgestaltung verwendet. Klassische Beispiele sind SW-CMM, CMMI-DEV[54] usw. sowie die ISO 15504 und Automotive bzw. Enterprise SPICE.[55]

Die weitere Unterscheidung in zwei Arten von Softwarequalitätsmodellen ist in der Literatur nicht einheitlich. Balzert, Ebert (2008) unterscheiden zwischen sog. FCM- und GQM-Modellen. In einem FCM-Modell (factor-criteria-metrics-models) wird die Softwarequalität durch Qualitätsmerkmale, sog. „factors“, beschrieben, die durch Teilmerkmale, sog. „criterions“, verfeinert werden. Sie werden durch

---

[45] Vgl. Wallmüller (2011), S.11.

[46] In der Wirtschaftsinformatik dienen Modelle der Beschreibung, Abstraktion und Verdeutlichung von Informationskonzepten. Sie werden in Beschreibungs- und Erfassungsmodelle, Erklärungsmodelle, Gestaltungsmodelle, Meta- und generischen Modelle und sonstige Modelle unterschieden. Sie sind meist symbolisch, immateriell und oft eng mit Fachkonzepten aus Mathematik und BWL verbunden. (Lehner (1999), S.73.)

[47] In der ISO 9000 werden Sichten mit denen von Organisationen, ihren Kunden und anderen interessierten Parteien, die ein erhebliches Risiko für die Nachhaltigkeit der Organisation darstellen, beschrieben (Norm DIN EN ISO 9000:2015-11, S.11ff.)

[48] Vgl. Liggesmeyer (2009), S.16.

[49] Vgl. McCall u.a. (1977).

[50] Vgl. Boehm u.a. (1976).

[51] Vgl. Willmer (1985).

[52] Vgl. Sunazuka u.a. (1985).

[53] Vgl. Grady, Caswell (1987).

[54] Vgl. Wallmüller (2007), S.11f.

[55] Vgl. Höhn (2009); Müller (2007).

Qualitätsindikatoren mess- und bewertbar gemacht, indem Indikatoren als Softwareeigenschaften in Beziehung zu den Qualitätsmerkmalen gesetzt werden. Der Messwert als Ergebnis wird auf einer Skala abgebildet. Bei mehreren Teilmerkmalen bilden FCM-Modelle (Abbildung 2) oft einen Baum oder ein Netz. FCM-Modelle können sowohl für Produktqualität als auch für Prozessqualität aufgestellt werden.[56]

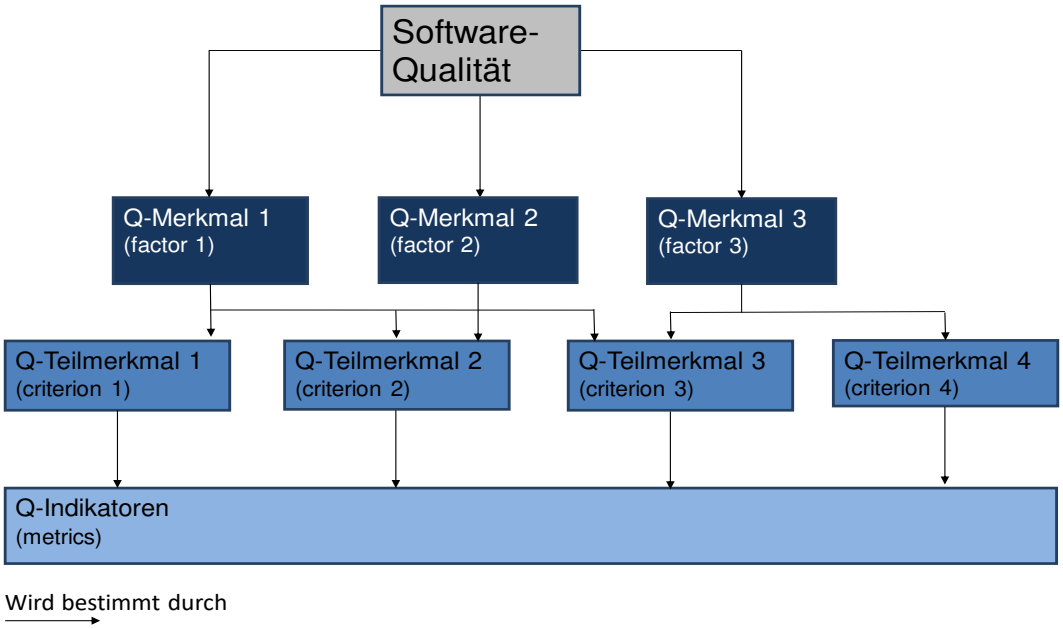

Abbildung 2: FCM-Modell (Eigene Darstellung in Anlehnung an Balzert, Ebert (2008), S.462)

Modelle, die zu einem unternehmens- bzw. entwicklungsspezifischen Qualitätsmodell führen, werden als GQM-Modelle (goal-question-metric-model) bezeichnet. Es beschreibt eine systematische Vorgehensweise zur Erstellung eines entwicklungsspezifischen Qualitätsmodells. Dabei werden zunächst relevante Eigenschaften der Organisation bzw. des zu vermessenden Produkts identifiziert. Anschließend werden Messziele identifiziert und ein Messplan erarbeitet sowie das Daten-Sammelverfahren festgelegt, die Daten gesammelt, analysiert und interpretiert. Über mehrere Projekte werden Post-Mortem-Analysen durchgeführt und die Erfahrungen anschließend zusammengefasst.[57] Ergebnis der sechs Schritte ist ein in

[56] Vgl. Balzert, Ebert (2008), 461ff.
[57] Vgl. Balzert, Ebert (2008), S.466.

Abbildung 3 dargestelltes Bewertungsmodell. Ein entsprechendes Beziehungsmuster ergibt sich zwischen den Fragen und Maßen.[58]

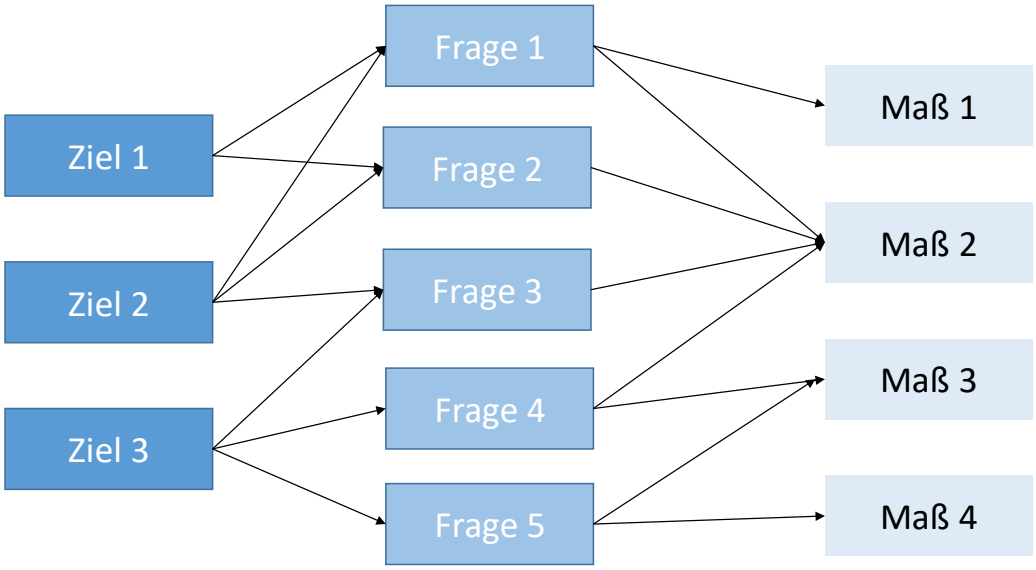

Abbildung 3: GQM-Modell (eigene Darstellung in Anlehung an Rombach, Basili (1987), S.150.)

Bächle (1996) hingegen unterschiedet normative und generische (kontextspezifische) Qualitätsmodelle. Normative Modelle umfassen Gesamtmodelle und Qualitätssichten, die Qualität ist allgemeingültig für alle Anspruchsgruppen. Die Gesamtheit von normativen Qualitätssichten bildet das allgemeingültige Qualitätsmodell. Generische Qualitätsmodelle berücksichtigen das Umfeld und die Interessen der Anspruchsgruppen für konkrete Softwareentwicklungsprojekte.[59] Dabei sind FCM-Modelle mit den normativen und Vorgehensmodelle wie das GQM-Modell mit den generischen Qualitätsmodellen von Bächle vergleichbar (siehe Tabelle 3).

	Produktmodell	Prozessmodell
FCM-Modell bzw. normatives Modell	x	x
GQM bzw. generisches Modell	x	x

Tabelle 3: Arten von Qualitätsmodellen

[58] Vgl. Rombach, Basili (1987).
[59] Vgl. Bächle (1996), S.132.

Um unterschiedlichen Terminologien und Strukturierungstiefen entgegen zu wirken, definiert die ISO 9126 drei Sichten auf Software-Qualität: interne, externe und Gebrauchsqualität.[60] Die Dekomposition ist dennoch oft zu grob-granular, um direkt einsetzbar zu sein. Darüber hinaus fehlt oft der Bezug zu Entwicklung, Betrieb oder Nutzung des Systems und somit zu Kosten und Nutzen, Qualitätskriterien sind teilweise nicht disjunkt und auf sehr unterschiedlichen Abstraktionsebenen. Qualitätsmodelle existieren oft nur in Form von Text und Graphiken und sind kein lebendiger Teil des Entwicklungsprozesses.[61]

Die Qualität einer Softwarekomponente als inhärente Eigenschaft hängt vom Kontext ab. Es ist eine grundlegende Herausforderung und gilt für jedes Qualitätsmodell.[62] Qualitätsbewertungen beinhalten demnach einen Vergleich zwischen Qualitätsvorgaben (Soll-Werten) und den tatsächlich erreichten Merkmalsausprägungen (Ist-Werte).[63]

2.4 Software-Qualitätsmanagement

Der in Kapitel 2.3 definierte Software-Qualitätsbegriff spiegelt den produkt- und prozessbezogenen Qualitätsansatz wider. Demnach bezieht sich Qualität auf die Summe aller Produkteigenschaften sowie die Prozesseigenschaften zur Software-Erstellung. Entsprechend ist zwischen einem produkt- und prozessorientierten SQM zu unterscheiden. Maßnahmen des produktorientierten Qualitätsmanagements setzen unmittelbar an der Software und deren Zwischenprodukten an. Maßnahmen des prozessorientierten Qualitätsmanagements gehen darüber hinaus und betrachten den Entwicklungsprozess[64] sowie die Überprüfung der Einhaltung a priori festgelegter

[60] Die interne Qualität betrachtet Merkmale des isolierten unausgeführten Produkts. Die externe Qualität betrachtet Merkmale des Produkts bei der Ausführung in einer definierten Umgebung, häufig basierend auf Tests. Die Gebrauchsqualität betrachtet Produktmerkmale beim Einsatz in Endbenutzerumgebung. Es wird davon ausgegangen, dass die interne Qualität die externen Eigenschaften des Softwareproduktes beeinflusst, die wiederum die Nutzungsqualität beeinflussen (Vgl. Balzert, Ebert (2008), S.462ff).

[61] Vgl. Wagner u.a. (2008), S.2.

[62] Vgl. Jones, Bonsignour (2012), S.9.

[63] Vgl. Wallmüller (2011), S.11.

[64] Im Sinne des prozessorientierten Qualitätsmanagements ist ein qualitativ hochwertiger Prozess stabil und damit wiederholbar sowie hinsichtlich Kosten, Zeit und Qualität vorhersehbarer, planbar und steuerbar. Nur ein solcher Prozess kann zu der Erstellung qualitativ hochwertiger Software führen. (Vgl. Mellis (2004), S.327ff).

Prozess-Qualitätsmerkmale.[65] Da die Beziehung zwischen Prozess- und Produktqualität in der Softwareentwicklung komplexer ist als in der produzierenden Industrie[66] sind Annahmen und Wirksamkeit des prozessorientierten Qualitätsmanagements empirisch nicht endgültig nachgewiesen. [67]

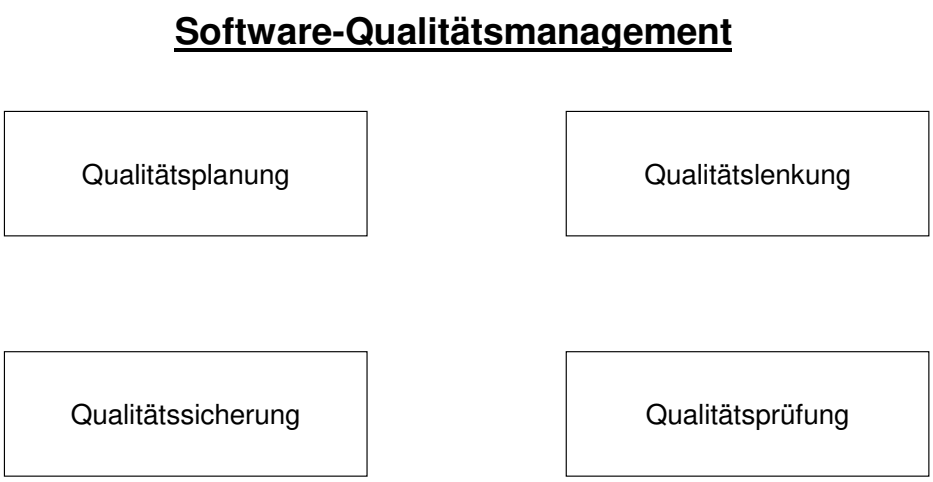

Abbildung 4: SQM-Prozess (eigene Darstellung in Anlehnung an Mistrík (2016)

Die **Qualitätspolitik** bildet die Grundlage für die Ableitung der Qualitätsziele im Rahmen der Qualitätsplanung.[68] Strategische Qualitätsziele sind oft Teil des strategischen Geschäftsplans. Taktische Geschäftsziele werden als Teilziele von den strategischen abgeleitet.[69] Die **Qualitätsplanung** legt überprüfbare Qualitätsanforderungen an Prozess und Software fest.[70] Qualitätsanforderungen beziehen sich dabei auf eine Software produzierende Unternehmenseinheit, ein Softwareprodukt oder dessen Teile, sowie auch auf den gesamten bzw. auf Teilprojekte des Softwareentwicklungsprozesses.[71] Software-Anforderungen müssen insbesondere in Bezug auf nicht-funktionale Anforderungen sauber spezifiziert sein.

[65] Vg. Herzwurm, Mikusz (2016), URL siehe Literaturverzeichnis.
[66] Vgl. Sommerville (2012), S.713.
[67] Vgl. Mellis (2004), S.353ff.
[68] Mit Qualitätsmanagement werden prinzipiell wirtschaftliche Ziele verfolgt. Großes Potential des Qualitätsmanagements im Bereich der Softwareentwicklung liegt in der Bestimmung der optimalen Kombination aus Fehlerkosten (Fehlerfolgekosten und Fehlerbehebungskosten) und den gegenläufigen Fehlerverhütungskosten (Prüfkosten und Kosten von präventiven Maßnahmen). (Vgl. Frühauf u.a. (2002), S.127.).
[69] Vgl. Wallmüller (2011), S.41.
[70] Vgl. Balzert, Ebert (2008), S.479.
[71] Vgl. Frühauf u.a. (2002), S.128.

Software-Qualitätsmodelle sind operationelle Hilfsmittel im Bereich der Qualitätsplanung. [72]

Die Qualitätsplanung des Softwareproduktes ist ein wesentlicher Bestandteil des Requirements Engineering-Prozesses, bei dem die Anforderungen des Auftraggebers ermittelt und bewertet werden. Funktionale Anforderungen beschreiben dabei die konkreten Softwarefunktionen. Hinzu kommen nicht-funktionale sog. Qualitätsanforderungen, die gewünschte Qualitätsmerkmale wie Performanz oder Zuverlässigkeit definieren.[73] Das Ergebnis der Qualitätsplanung sind Qualitätsanforderungen im Lastenheft als das fachliche Ergebnisdokument der Softwareplanung. Die funktionalen sowie die Qualitätsanforderungen werden im Pflichtenheft weiter verfeinert, das wiederum Grundlage für die spätere Ableitung der Softwarearchitektur und der Implementierung ist.[74]

Um den wirtschaftlichen Zielen gerecht zu werden, sind neben den Produktanforderungen weitere Anforderungen an den Prozess zu berücksichtigen. Dies kann die Einhaltung zeitlicher bzw. finanzieller Restriktionen oder Prozessrichtlinien im militärischen oder medizinischen Bereich sein. Demnach müssen auch bei den Anforderungen an den Softwareentwicklungsprozess (Qualitäts-) Zielbestimmung und Priorisierung erfolgen.[75] Nach dem Verständnis des prozessorientierten Qualitätsmanagements verläuft die Prozess-Qualitätsplanung primär über Auswahl und Anpassung von Prozessstandards, die Prozesse definieren und Beschreibungen von zu erstellenden Dokumenten beinhalten. Insbesondere bei kleineren Entwicklungsvorhaben kann der weniger formale Ansatz der agilen Methoden zielführend sein. [76] Das Endergebnis der Prozess-Qualitätsplanung wird in einem Projektauftrag festgehalten.

Die **Qualitätslenkung** steuert, überwacht und korrigiert den Entwicklungsprozess, um die Anforderungen zu erfüllen oder bei Bedarf korrektive Maßnahmen zu ergreifen.[77] Heutzutage geschieht Qualitätslenkung primär über Prozessmanagement mit Hilfe von präventiv und konstruktiv vorrausschauenden Methoden und Hilfsmitteln, um die

[72] Vgl. Wallmüller (2011), S.43.
[73] Vgl. Pohl (2008), S.15.
[74] Vgl. Balzert, Ebert (2008), S. 443ff.
[75] Vgl. Mellis (2004), , S,39ff
[76] Vgl. Sommerville (2012), S.710f.
[77] Vgl. Balzert, Ebert (2008), S.480.

Prozessausführung zu optimieren. Zusätzliche analytische Maßnahmen stellen das entsprechende Qualitätsniveau der Zwischen- und Endprodukte sicher.[78]

Neben der Anpassung der konstruktiven und analytischen Maßnahmen geht es um die zeitliche Gestaltung und die Intensität der **Qualitätsprüfung**.[79] Um zu gewährleistet, dass die festgelegten Verfahren und Standards des Qualitätsmanagements eingehalten wurden, werden Ist-Werte erfasst und überwacht, ob die konstruktiven Maßnahmen durchgeführt wurden.[80] Maßnahmen im Rahmen der **Qualitätssicherung**[81] können in konstruktive und analytische Maßnahmen unterteilt werden. Konstruktive Qualitätsmanagementmaßnahmen sind technische Maßnahmen wie Methoden, Sprachen, Werkzeuge, und organisatorische Maßnahmen wie Checklisten, Richtlinien und Standards. Diese sorgen dafür, dass der Erstellungsprozess bzw. das Produkt à priori definierte Eigenschaften besitzt. Bei analytischen Qualitätsmanagementmaßnahmen handelt es sich um diagnostische Maßnahmen, die in das Produkt oder den Prozess keine Qualität per se bringen sondern das Qualitätsniveau messen. Dabei sammeln analysierende Verfahren mittels analytischer Mittel gezielt Informationen über die Software ohne ihre dynamische Ausführung. Testende Verfahren, häufig „Testing“[82] genannt, testen die Software mit konkreten Eingaben. Generelles Ziel ist es, durch konstruktive Maßnahmen den Analyseaufwand zu minimieren.[83] Entscheidend für den Erfolg ist ein entwicklerunabhängiges Qualitätsteam mit ausreichend Personal, Qualifikation und Erfahrung.[84]

Mit zunehmender Verweildauer eines Fehlers im Produkt steigen die Behebungskosten aufgrund der Fehlerfortpflanzung überproportional an. Um Fehler frühzeitig entdecken und beheben zu können sowie jederzeit das Qualitätsniveau von

---

[78] Vgl. Wallmüller (2011), S.44.

[79] Vgl. Mellis (2004), S181ff.

[80] Vgl. Herzwurm, Mikusz (2016), URL siehe Literaturverzeichnis

[81] In früheren Normen wurde nicht der Oberbegriff Qualitätsmanagement verwendet, sondern der Begriff Qualitätssicherung. Heutzutage werden unter dem Begriff Qualitätsmanagement managementbezogene Aktivitäten und unter dem Begriff Qualitätssicherung technikorientierte Aktivitäten subsummiert. (Vgl. Balzert, Ebert (2008), S.475)

[82] Testen von Software ist eine wichtige und umfangreiche Teildisziplin des SQM. Umfangreiche Literatur beschäftigt sich ausschließlich mit dem Testen von Software. Da diese Arbeit sich mit SQM auf einer Metaebene beschäftigt, wird das „testing“ an dieser Stelle nicht vertieft behandelt. Für weitere Ausführungen wird auf Liggesmeyer (2009) verwiesen.

[83] Vgl. Balzert, Ebert (2008), S.478f.

[84] Vgl. Wallmüller (2011), S.45.

Produkt und Prozess zu überblicken ist eine stringente entwicklungsbegleitende und in den Softwareentwicklungsprozess integrierte Qualitätslenkung notwendig.[85]

2.5 Software-intensive Business bzw. software-intensive Systeme

Nahezu branchenübergreifend bedeutet „software-intensive Business“ eine Verlagerung der Wertschöpfung von der Entwicklung, Produktion und Vermarktung von monolithischen Produkten zu branchenübergreifenden digitalen Unternehmensnetzwerken.[86] Sogenannte Cyber-Physical-Systems ermöglichen Industrieunternehmen den Zugang zur digitalen Welt der Softwareunternehmen. Digitale und analoge Märkte wachsen zusammen. Die Strategie, nur einzelne Produkte für nahezu homogene Kundengruppen anzubieten, ist veraltet und wird zunehmend ersetzt durch hybride Produktpakete, die auf vielseitigen Märkten angeboten werden. Plattformen ermöglichen die Zusammenarbeit von Wertschöpfungspartnern.[87] Dabei werden im Rahmen des software-intensive Business zwei Aspekte betrachtet: Maßnahmen und Methoden sowie Reaktionen auf Umweltveränderungen. Bei der Untersuchung der Maßnahmen und Methoden wird unterschieden zwischen Phänomenen innerhalb und zwischen Organisationen wie beispielsweise Softwareunternehmen, Start-ups, Datenunternehmen. Von Interesse sind innerhalb solcher Organisationen Methoden wie Produktmanagement, Geschäftsmodelle, Agilität und DevOps. Untersuchungsgegenstände zwischen Organisationen sind Ökosysteme, Plattformen und Open Source Software-Communities.[88]

Da Untersuchungsgegenstand dieser Arbeit die Softwarequalität ist, wird der Schwerpunkt im weiteren Verlauf nicht auf den software-intensiven Business als Ganzes gelegt, sondern software-intensive Systeme betrachtet. In software-intensiven Systemen hat Software wesentliche Einflüsse auf Design, Konstruktion, Bereitstellung und Entwicklung des gesamten Systems. Beispiele für softwareintensive Systeme sind eingebettete Systeme[89] für Luftfahrt- und Automobilanwendungen, heterogene

[85] Vgl. Balzert, Ebert (2008), S.488.
[86] Vgl. acatech (2011), URL siehe Literaturverzeichnis, S.5.
[87] Vgl. Schönhofen u.a. (2018).
[88] Vgl. Werder u.a. (2018).
[89] Häufig werden software-intensive Systeme mit eingebetteten Systemen gleichgesetzt. Ein „embedded system“ ist ein binärwertiges Computersystem, das in ein umgebendes technisches System eingebettet ist und mit diesem wechselwirkt. „Embedded Systems“ sind also eine Ausprägung von software-intensiven Systemen. (Vgl. Siemers (2011), S.5.).

Großsysteme oder Geschäftsanwendungen mit besonderem Schwerpunkt auf Webdiensten.[90]

Die zentralen Eigenschaften und Aspekte softwareintensiver Systeme, die aus den Besonderheiten von Software resultieren (Kapitel 2.3), sind:[91]

- Die **Geräte** als einzelnen Knoten eines SIS: Die zunehmende Verbreitung SIS liegt primär an der Verfügbarkeit kostengünstiger, sehr kleiner und energiesparender Computer bzw. Prozessoren. Dies ermöglicht es, Hard-und Software in Geräten zu installieren, in denen es bisher unwirtschaftlich war. Je mehr und unterschiedlichere Geräte in einem SIS sind, desto komplexer ist die Orchestrierung des Systems.
- **Inhalt**: Daten, Informationen und Wissen, die vom System erfasst, gespeichert und verarbeitet werden. Der Trend geht dabei von der reinen Datenverarbeitung zur Verarbeitung von Informationen zu Wissen. Dabei werden Sensordaten und bereits im System vorhandene digitale Daten integriert. SIS müssen in der Lage sein, die Korrektheit der übermittelten Daten zu überprüfen und mögliche Betriebssituationen zu antizipieren, um die Datenverarbeitung und –speicherung steuern zu können. Die Konsistenz und Integrität der Daten ist über alle verteilten Systeme hinweg gesichert. Sicherheitsrelevant ist dabei besonders die Zugangsverwaltung.
- Grundlage für ein funktionierendes SIS ist die **Interaktion** zwischen Komponenten und Nutzern über ein Netzwerk. Aktuelle Netzwerke haben eine dynamische Konfiguration für multifunktionale Netzwerkgeräte. Endgeräteabhängige Kommunikation und Datenübertragung werden häufig über plattformübergreifende Integrationstechniken wie Webservices ermöglicht. Die Mensch-System-Interaktion geschieht häufig über einfache und selbsterklärende Benutzerschnittstellen bedienbar per Tastatur und Maus oder Touchscreen. Die Bedienung mittels natürlicher Sprache verbreitet sich.
- Die **Anpassbarkeit** eines Systems an neue Anforderungen, Richtlinien und Systemumgebungen wird zukünftig ein wichtiger Erfolgsfaktor von SIS sein. Hier herrscht Forschungsbedarf.

[90] Vgl. Mendes, Winkler (2018), S.657.
[91] Vgl. Hölzl u.a. (2008), S.3ff.

- **Sicherheit und Zuverlässigkeit**: Servicequalität umfasst viele Elemente wie Verfügbarkeit oder Antwortzeiten und wird in Service-Level-Agreements festgehalten. Die vom Nutzer wahrgenommene Qualität hängt von den technischen Fähigkeiten des Systems und von der Mensch-System-Schnittstelle ab. Die zunehmende Netzwerkverknüpfung erhöht das Risiko für das System. Schlüssel- und Identitätsmanagement, Zugriffskontrollen für verteilte Systeme spielen bei der Risikoreduktion eine zentrale Rolle. Hardwarebasierte Sicherheitsmechanismen können die Sicherheit zusätzlich erhöhen. Die Fähigkeit von SIS, Sicherheitslücken innerhalb des Systems zu ermitteln, ist wichtig um dem Nutzer ein Gefühl von Sicherheit zu vermitteln.
- Aufgrund der Komplexität SIS ergeben sich Besonderheiten in der **Systementwicklung**. Es muss ein Gleichgewicht zwischen Software- und Systementwicklung herrschen. Die Entwicklung von heterogenen Komponenten findet häufig verteilt statt. Aus wirtschaftlichen Gründen sollten Systemteile wiederverwendet werden. Die Strukturierung multidimensionaler Anforderungen an das System beeinflusst den Entwicklungserfolg nachhaltig. Definierte Sprachen für Anforderungen und Werkzeuge unterstützen bei der Entwicklung komplexer SIS.

3 Strukturierte Literaturanalyse zum Stand der Forschung im SQM

Zur Erhebung des Stands der Forschung im SQM wird eine strukturierte Literaturanalyse, genauer gesagt eine Mapping-Studie, durchgeführt. Das allgemeine Ziel einer Literaturanalyse ist es, existierendes Wissen in der Literatur zu aggregieren und Gemeinsamkeiten bzw. Widersprüche zu identifizieren. Sie gibt nicht nur eine Übersicht über bestehende Literatur sondern konsolidiert diese hinsichtlich eines bestimmten Forschungsschwerpunkts. Vergangene Forschung wird kritisch beleuchtet und Forschungsergebnisse erklärt.[92] Forscher werden befähigt, bestehendes Wissen zu verstehen und Forschungslücken zu identifizieren, um weitere Forschungsarbeit darauf aufzubauen.[93] Besonders in der Wirtschaftsinformatik wird die Literatur immer unübersichtlicher, die Zahl der veröffentlichten Bücher, Zeitschriften sowie veranstalteten Konferenzen steigt. Pro Zeitschrift werden mehr Beiträge publiziert, diese Beträge werden immer länger. Der im Zeitverlauf stattfindende Ausdifferenzierungsprozess der Theorien und Methoden einer Wissenschaftsdisziplin macht die Forschung immer komplexer. Um das Risiko von Doppelarbeiten oder das Vernachlässigen von relevanten Erkenntnissen zu vermeiden, bedarf die Recherche nach relevanter Literatur immer mehr Zeit.[94]

Systematische Literaturanalysen (Reviews) verringern die impliziten Fehler entstehend durch Ausrichtung der Forscher. Durch die Einführung umfassender Suchstrategien, vordefinierter Suchzeichenfolgen und einheitlicher Einschluss- und Ausschlusskriterien zwingen systematische Reviews Forscher, effektiv nach Studien jenseits ihrer Fachgebiete und Netzwerke zu suchen, um eine objektivere Antwort auf die Forschungsfrage zu erhalten. Es reduziert jedoch nicht die Publikationsverzerrung durch die fehlende Veröffentlichung von negativen Ergebnissen.[95] Durch systematische Überprüfungen lässt sich die Robustheit der Beweise effektiv einschätzen. Klassifizierung der Qualität und Eigenschaften von Wirkungsanalysen anhand standardisierter Kriterien werden ebenfalls ermöglicht. Die Verwendung eines systematischen Überprüfungsprotokolls ermöglicht die methodische Transparenz der Überprüfung und zukünftige Replikationen des Forschungsansatzes. Peer Reviews des Protokolls und Prozesses sorgen für eine weitere Fehlerreduktion. Darüber hinaus

[92] Vgl. Rowe (2014), S.242.
[93] Vgl. Paré u.a. (2015), S.183.
[94] Vgl. Fettke (2006), S.257f, Cooper (1988), S.114.
[95] Vgl. Kitchenham (2007), S.15.

ermöglichen systematische Reviews eine objektive Ausgangsbasis für zukünftige Forschung und haben Vorteile gegenüber herkömmlichen Literaturrecherchen. Es ist jedoch schwieriger, systematische Überprüfungen durchzuführen.[96]

Die Literaturanalyse dieser Arbeit lässt sich als sog. systematische Mapping Studie einordnen. Sie bietet einen umfassenden Überblick über ein Forschungsgebiet, kann Bereiche identifizieren, die für die Durchführung systematischer Literaturrecherchen geeignet sind oder eine Primärstudie erfordern. Die Hauptunterschiede zwischen einer Mapping-Studie und einer klassischen systematischen Literaturanalyse sind: [97]

- Mapping-Studien haben im Allgemeinen umfassendere Forschungsfragen und stellen häufig mehrere Forschungsfragen.
- Die Suchbegriffe für Mapping-Studien sind weniger fokussiert und geben oft eine sehr große Anzahl von Ergebnissen zurück. Dies ist jedoch weniger problematisch, weil die systematische Überprüfung hier eher auf eine breite Abdeckung zielt als auf einen engen Fokus.
- Der Datenextraktionsprozess ist eher ein Klassifizierungs- oder Kategorisierungsprozess. Dabei werden Beiträge so detailliert klassifiziert, dass die allgemeinen Forschungsfragen beantwortet und Beiträge für spätere Überprüfungen identifiziert werden können. Techniken wie Metaanalysen werden selten einbezogen, dafür aber Summen und Zusammenfassungen. Eine wirksame grafische Aufbereitung ist die Darstellung anhand des Klassifizierungstyps.
- Die Ergebnisse umfassen eine Anzahl an Veröffentlichungen, die einzelnen Subkategorien zugeordnet werden. Sie eignen sich in erster Linie für die Forschungsentwicklung und weniger für praktische Implikationen.

3.1 Vorgehensweise innerhalb der strukturierten Mapping-Studie

Strukturierte Literaturanalysen wie auch Mapping-Studien lassen sich, wie in Abbildung 5 zu erkennen, in die drei Hauptphasen Review-Planung, -Durchführung und Ergebnispräsentation einteilen.[98] Unabhängig von den Unterschieden in der

[96] Vgl. Mallett u.a. (2012), S.447f; Kitchenham (2007), S.4f; Arksey, O'Malley (2005); Vom Brocke u.a. (2009), S.2.
[97] Vgl. Kitchenham (2007), S.44; Kitchenham u.a. (2010), S.26.
[98] Vgl. Brereton u.a. (2007), S.572; Kitchenham (2007), S.6.

Durchführung der Review-Typen können alle Reviews nach folgenden allgemeinen Schritten durchgeführt werden: (1) Definition des Forschungsbereichs, (2) Aufstellung und Bewertung eines Review-Protokolls, (3) Literatursuche, (4) Ein- und Ausschluss-kriterien nutzen, (5) Qualität überprüfen, (6) Datenextraktion, (7) Analyse und Synthese der Publikationen, (8) Dokumentation der Ergebnisse. Dabei handelt es sich nicht um einen zwingend sequenziellen Ablauf. Einzelne Schritte können in einem iterativen Prozess wiederholt werden. [99]

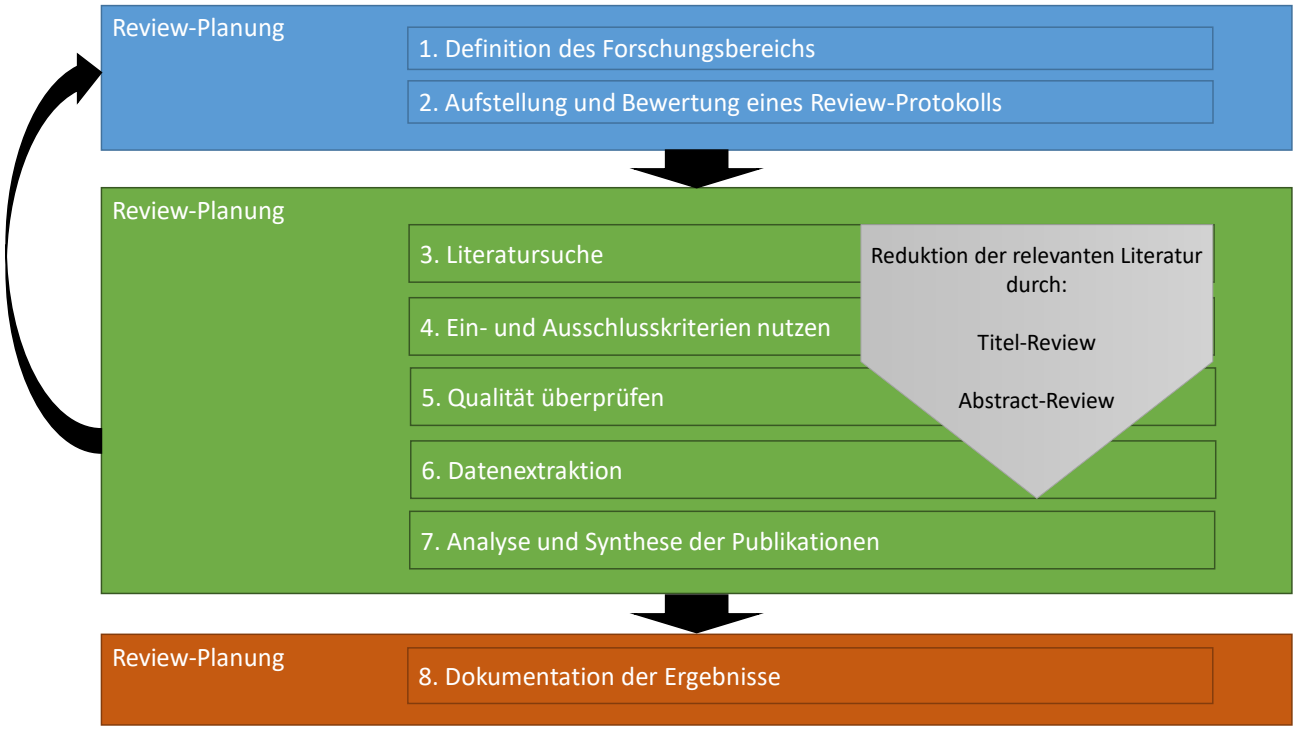

Abbildung 5: Prozess der systematischen Mapping-Studie

Der erste Schritt liegt in der **Definition des Forschungsbereichs** aus der Forschungsfrage der Arbeit. Die Identifikation kann ein iterativer Prozess sein. Brereton u.a. (2007) schlagen vor, eine Voruntersuchung der Literatur zu machen, um bisherige Forschungsaktivitäten zu identifizieren.[100] Im Rahmen dieser Arbeit entspricht die Voruntersuchung der in Kapitel 2 dargestellten wissenschaftlichen Grundlagen. Der Analysefokus liegt auf dem Qualitätsmanagement von Software (siehe Kapitel 1.3). Um den Gestaltungsbereich umfassend zu erforschen erfolgt eine strukturierte Literaturanalyse unter Berücksichtigung der in Kapitel 2 thematisierten Aspekte.

[99] Vgl. Kitchenham (2007), S.6; Xiao, Watson (2019), S.102f; Brereton u.a. (2007), S.572.
[100] Vgl. Brereton u.a. (2007), S.578.

Die **Aufstellung und Bewertung des Review-Protokolls** spezifiziert die im durchgeführten Review verwendeten Methoden und ist deshalb absolut notwendig für rigide strukturierte Reviews. Ein Review-Protokoll steigert die Qualität der Analyse, da es die Forscher-Verzerrung in Datenauswahl und –analyse reduziert. Außerdem erhöht es die Reliabilität, wenn andere Forscher das Protokoll nutzen, um das Review zu wiederholen und zu überprüfen. Das Protokoll beschreibt alle Elemente der Literaturanalyse inklusive Ziel, Forschungsfragen, Einschlusskriterien, Suchstrategien, Qualitätssicherungskriterien und Strategien für Datenextraktion und –analyse. Bevor das Protokoll angewendet wird, muss es überprüft werden. Das Review-Protokoll dieser Arbeit befindet sich im Anhang. Wie von Xiao, Watson (2019) vorgeschlagen, wurde das Protokoll vom Betreuer dieser Arbeit, Herrn Professor Dr. Herzwurm und Mitarbeitern seines Lehrstuhls in einem iterativen Prozess bewertet.[101]

Im Rahmen der **Literatursuche** wird die Literatur für die Mapping-Studie identifiziert, um einen Überblick über die relevanten Publikationen des Forschungsgebietes zu geben. Es gibt drei Hauptquellen: Elektronische Datenbanken, Rückwärtssuche und Vorwärtssuche. Im Rahmen dieser Arbeit werden auf Grund der weit gefassten Forschungsfrage nur ausgewählte Journal- und Konferenzbeiträge mit hoher Bewertung manuell untersucht. Auf Grund der Interdisziplinarität des Themas werden Quellen aus dem Gebiet der Wirtschaftsinformatik und des Software Engineering untersucht. Für Wirtschaftsinformatik (Information Systems) werden Journals nach dem Senior Scholars' Basket of Journals, A+ und A gerankte Journals aus dem VHB Jourqual 3 des Verbands der Hochschullehrer für Betriebswirtschaft und VHB Journals mit Software Schwerpunkt (B und C gerankt) untersucht. Zusätzlich wird ein Journal im Bereich SQM untersucht, um als Vergleichsgrundlage zu nicht-SQM-spezifischen Quellen zu dienen. Für Software Engineering werden Journals nach CAPES (2017)[102] A1 und A2 und nach dem Google Scholar Ranking auf Basis des h5-Index berücksichtigt. Die ausgewählten wissenschaftlichen Zeitschriften werden in Tabelle 4 dargestellt.

[101] Vgl. Xiao, Watson (2019), S.103.

[102] Die Förderagentur für Hochschulbildung (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, CAPES) vom brasilianischen Bildungsministerium veröffentlicht Konferenz- und Journal-Rankings, die den H-Index (Kennzahl für das weltweite Ansehen eines Wissenschaftlers in Fachkreisen) als Leistungsmaß für Konferenzen/Journals verwenden. Basierend auf den H-Index-

Name des Journals	Bereich
(Mathematical Programming)	IS (Information Systems)
ACM SIGSOFT International Symposium on Foundations of Software Engineering	SE (Software Engineering)
ACM Transactions on Software Engineering and Methodology	SE
Communications of the ACM	SE
Empirical Software Engineering	SE
European Journal of Information Systems	IS
IEEE Software	SE
IEEE Transactions on Software Engineering	SE
INFORM Journal on Computing	IS
Information & Software Technology	SE
Information Systems Journal	IS
Information Systems Research	IS
Journal of AIS	IS
Journal of Information Technology	IS
Journal of MIS	IS
Journal of Strategic Information Systems	IS
Journal of Systems and Software	SE
MIS Quarterly	IS
Science of Computer Programming	SE
SIAM Journal on Computing	IS
Software Quality Journal	Software Quality
Software: Practice & Experience	SE

Tabelle 4: Relevante Journals für die Literatursuche im Bereich SQM

Konferenzen werden einerseits betrachtet, um einen systematischen Fehler durch den sog. Publication Bias zu vermeiden.[103] Andererseits wird ein breites Feld für eine Trendanalyse im Bereich der Forschungsthemen untersucht. Die aktuellsten Forschungsergebnisse finden sich häufig in Konferenzbänden, da hier die Begutachtungsverfahren schneller als bei Zeitschriften sind. Die Auswahl der Konferenzen im Bereich Wirtschaftsinformatik nach wird nach Walstrom u.a. (1995) durchgeführt. Im Bereich Software Engineering werden Konferenzen nach CORE (2018)[104] und CAPES (2012) A bzw. A1A/A2 gerankte Konferenzen sowie vorgeschlagene Konferenzen zum Thema der Gesellschaft für Informatik untersucht.

Name der Konferenz	Abkürzung	Bereich
Academy of Management Conference	AOM	IS
ACM SIGSOFT Conference on the Foundations of Software Engineering	FSE_A	SE

[103] Vgl. Kitchenham (2007), S.22.

[104] Dieses Konferenz-Ranking wurde im Rahmen der Excellence in Research in Australia (ERA) Initiative erstellt. Das Ranking wurde von australischen Dekanen und der Australian Computing Research and Education Association of Australasia (CORE eine Vereinigung von Universitätsinstituten für Informatik in Australien und Neuseeland) erstellt. Die Ranglisten reichen von A (=beste) bis C (=schlechteste).

ACM SIGSOFT International Symposium on the Foundations of Software Engineering	FSE	SE
Americas Conference on Information Systems	AMCIS	IS
Automated Software Engineering Conference	ASE	SE
Decision Sciences Institute - National and Regional Conferences	DSI	IS
European Conference on Information Systems	ECIS	IS
European Conference on Software Maintenance and Reengineering	CSMR	SE
European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering	ESEC/FSE	SE
Fundamental Approaches to Software Engineering	FASE	SE
Hawaii International Conference on System Sciences	HICSS	IS
Information Resources Management Association Conference	IRMA	IS
Institute for Operations Research and the Management Sciences Conference	INFORMS	IS
International Association for Computer Information Systems	IACIS	IS
International Conference on Decision Support Systems	DSS	IS
International Conference on Evaluation and Assessment in Software Engineering	EASE	SE
International Conference on Information Systems	ICIS	IS
International Conference on Software Engineering	ICSE	SE
International Conference on Software Testing, Verification and Validation	ICST	SE
International Federation for Information Processing	IFIP	IS
International Symposium Component-Based Software Engineering	CBSE	SE
International Symposium on Empirical Software Engineering and Measurement	ESEM	SE
International Symposium on Software Reliability Engineering	ISSRE	SE
Internationale Tagung Wirtschaftsinformatik	WI	IS
Multikonferenz Wirtschaftsinformatik	MKWI	IS
Society of Information Management Conference	SIM	IS
Software Engineering	SE	SE
Software Management	SWM	SE
IEEE International Conference on Software Maintenance and Evolution	ICSME/ICSM	SE
European Conference on Software Architecture	ECSA	SE
IEEE International Working Conference on Mining Software Repositories	SMS	SE
ACM Symposium on User Interface Software and Technology	UIST	SE

Tabelle 5: Relevante Konferenzen für die Literatursuche im Bereich SQM

Als Bestandteil der Literatursuche werden zunächst Suchbegriffe definiert. Die Suchterme sollten von der Forschungsfrage abgeleitet werden.[105] Da der Großteil der Quellen auf Englisch ist, wird auch in englischer Sprache gesucht. In deutschsprachigen Journals und Konferenzen wird auch mit deutschen Suchbegriffen gesucht. Die Suchbegriffe sind sehr weit gefasst, da das Ziel dieser Literaturanalyse die Darstellung des Forschungstands im SQM auf einer hohen Ebene ist.

[105] Vgl. Kitchenham (2007), S.9.

Englische Suchbegriffe	Deutsche Suchbegriffe
„ software quality“	„Softwarequalitäts?“ OR „Software-Qualtäts?“[106]
	„Software-Qualität“ OR „Software-qualität“

Tabelle 6: Suchbegriffe Literaturanalyse

Eine Pilotstudie in zehn Journals im Bereich Wirtschaftsinformatik ergibt, dass die Suchbegriffe teilweise relevante Ergebnisse liefern. Die Zahl der gefundenen Artikel mit dem weiten Suchbegriff „software quality“ ist jedoch viel zu groß. Deshalb werden in einem weiteren Schritt **Ein- und Ausschlusskriterien** genutzt.

Einschlusskriterien dabei sind:

Einschlusskriterium	Begründung
„Software Quality“ in Titel und Abstract	alle Ausprägungen und Aspekte von Software Quality Management können abgedeckt werden
„Software-Qualitäts?“, „Softwarequalität?“ in Titel und Abstract	alle Ausprägungen und Aspekte von Software Quality Management können abgedeckt werden
Keine Einschränkung des Veröffentlichungsdatums	Abdeckung aller zeitlichen Entwicklungen
Beiträge auf Englisch und Deutsch	Es werden deutsch- und englischsprachige Quellen durchsucht

Tabelle 7: Einschlusskriterien Literaturanalyse SQM

Ausschlusskriterien werden gebildet von:

Ausschlusskriterium	Begründung
Artikel bei denen „überall“ gesucht wird	Die Ergebnisse wurden in einer ersten Suche als zu umfangreich und irrelevant identifiziert, da häufig „Software quality“ nur im Text genannt

[106] Die Wildcard „*?*“ dient zudem der Berücksichtigung der Plurale bzw. weiterer zusammengesetzter Nomen.

	wird, das Thema Software Qualität aber eine zu geringe Wichtigkeit im Artikel hat.
Artikel die nicht peer-reviewed wurden	Die Qualität des Artikels kann nicht gewährleistet werden
Artikel die Software für allgemeines Qualitätsmanagement thematisieren	Treffen nicht den Schwerpunkt der Arbeit
Editorials etc.	Sind keine vertiefenden Artikel zum Thema
Kein Zugriff auf Volltext vorhanden	Detailliertere Untersuchung des Textes nicht möglich

Tabelle 8: Ausschlusskriterien Literaturanalyse

Dabei wird in zwei Schritten vorgegangen. In einem ersten Schritt werden die Artikel auf Grund des Titels und des Abstracts überflogen und sortiert.

Im zweiten Schritt wird die Auswahl der Artikel basierend auf einer **Qualitätsüberprüfung** mittels einer Volltextanalyse verfeinert. Dabei wird der Volltext erneut auf Übereinstimmung mit Einschlusskriterien bzw. Freiheit von Ausschlusskriterien überprüft. Mapping-Studien, wie in dieser Arbeit, sollten Literatur aller Qualitätsstufen beinhalten um die Breite eines Forschungsgebietes und das Gesamtbild zu präsentieren.[107] Deshalb wird über die Volltextprüfung auf Ein- und Ausschlusskriterien hinaus keine weitere Qualitätsuntersuchung durchgeführt. Die hohe Qualität die Beiträge wird über die Auswahl von hochqualitativen Journals und Konferenzen sichergestellt.

Ziel der **Datenextraktion** ist es, Datenextraktionsformulare zu entwerfen, um Informationen aus den identifizierten Quellen genau zu erfassen. Um die Möglichkeit einer Verzerrung zu verringern sollten Datenextraktionsformulare definiert und bei der Definition des Studienprotokolls pilotiert werden.[108] Das Datenformular dieses Reviews umfasst wie in Tabelle 9 dargestellt folgende Informationen.

[107] Vgl. Whittemore, Knafl (2005), S.459.
[108] Vgl. Kitchenham (2007), S.29.

Quelle (Journal oder Konferenz)
Veröffentlichungsdatum (Jahr)
Klassifikation (Studie, SLR,…, Forschungstrend)
Thema
Autor und Zugehörigkeit zu…
Erarbeitete Praxisempfehlungen ja/nein
Forschungsfrage
Zusammenfassung

Tabelle 9: Datenextraktionsformular

In der **Analyse und Synthese der Publikationen** werden die Inhalte für den Leser strukturiert zusammengefasst. Webster, Watson (2002) unterscheiden dabei zwischen einem autorenzentrierten und einem konzeptzentrierten Ansatz. In dieser Arbeit wird in erster Linie der konzeptzentrierte Ansatz verfolgt, da der autorenzentrierte Ansatz das Risiko birgt, Konzepte zu vermischen. [109] Lediglich um UFF4 zu beantworten wird eine Auswertung nach Autoren durchgeführt.

Unterforschungsfrage	**Auswertung**
UFF1: Wie groß ist und war die Forschungsaktivität im Bereich SQM?	Anzahl der Beiträge nach Jahr
UFF2: Welche Forschungsthemen werden adressiert?	Anzahl der Beiträge nach Thema
UFF3: Wie haben sich diese Schwerpunkte über die Zeit verändert?	Themen nach Jahr aufgegliedert
UFF4: Wer sind die wichtigsten Forscher im Bereich SQM?	Anzahl der Veröffentlichungen pro Forscher
UFF5: In welchen Foren wird häufig zum Thema SQM veröffentlicht?	Anzahl der Beiträge pro Journal/Konferenz

Tabelle 10: Unterforschungsfragen mit entsprechendem Auswertungsmaß

Um die Veröffentlichungen anhand der behandelten Themen zu bewerten wurden den Beiträgen thematische Schlagworte zugeordnet (siehe Abbildung 6). Dafür wurden die Abstracts nach Stichworten und Konzepten durchsucht, um für die Grundgesamtheit repräsentative Dimensionen zu identifizieren. Wenn Abstracts keine Auswahl sinnvoller Keywords ermöglichen, wurden die Einführungs- oder

[109] Vgl. Webster, Watson (2002), S.xvi f.

Schlussfolgerungsabschnitte untersucht. Weitere Artikel wurden dann den Dimensionen zugeordnet oder falls nötig um neue Dimensionen erweitert. Anschließend wurden die identifizierten Dimensionen zu Oberthemen zusammengefasst um die Übersichtlichkeit für weiterführende Analysen zu erhöhen.

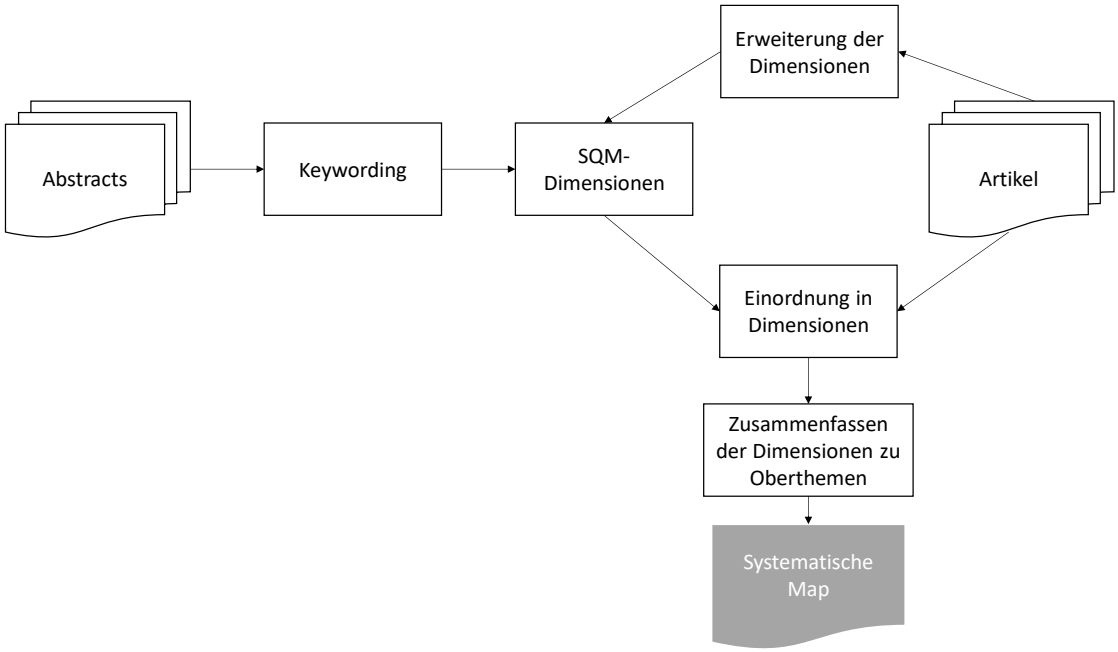

Abbildung 6: Klassifikationsprozess der Artikel

Der Forschungsansatz wurde analog der Einteilung nach Wieringa u.a. (2005) bewertet (siehe Tabelle 11). Die Kategorien sind einfach zu interpretieren und für die Klassifizierung zu verwenden, ohne jeden Beitrag im Detail zu bewerten zu müssen. Darüber hinaus ermöglicht das System die Klassifizierung der nicht-empirischen Forschung in die Kategorien Lösungsvorschlag, philosophische Beiträge, Meinungsbeiträge und Erfahrungsberichte.[110]

Kategorie	**Beschreibung**
Validierungsforschung	Die untersuchten Techniken sind neuartig und wurden noch nicht in die Praxis umgesetzt. Als Techniken werden z.B. Experimente verwendet.
Evaluationsforschung	Es werden Techniken in der Praxis umgesetzt und eine Bewertung der Technik durchgeführt und Folgen der Umsetzung in Bezug auf Vor- und Nachteile bewertet.
Lösungsvorschlag	Eine Lösung für ein Problem wird vorgeschlagen, die Lösung kann entweder neuartig oder eine signifikante Erweiterung einer bestehenden Technik sein.
Philosophisches Paper	Diese Beiträge skizzieren eine neue Sichtweise auf bestehende Dinge, indem sie das Feld in Form einer Taxonomie oder eines konzeptionellen Rahmens strukturieren.

[110] Vgl. Wieringa u.a. (2005), S.102ff.

Meinungspaper	Diese Beiträge drücken die persönliche Meinung aus. Sie sind nicht auf entsprechende Arbeiten und Forschungsmethoden angewiesen.
Erfahrungsbericht	Erfahrungsberichte erläutern, was und wie etwas in der Praxis getan wurde. Es muss die persönliche Erfahrung des Autors sein.

Tabelle 11: Überblick über Forschungsansätze nach Wieringa u.a. (2005), S.105f.

Ziel der **Dokumentation der Ergebnisse** ist es, die Diskrepanz zwischen bereits Erforschtem und dem, was bekannt sein sollte, aufzuzeigen. Bestehendes Wissen soll erweitert oder eine neue Theorie entwickelt werden. Detaillierte Ergebnisse dieser Mapping-Studie werden im nächsten Kapitel dargestellt.

3.2 Ergebnisse der strukturierten Mapping-Studie zum Stand der Forschung im SQM

Mittels definierter Suchbegriffe und einer Überprüfung der Titel wurden in den ausgewählten Journalen und Konferenzen 1156 potentiell relevante Veröffentlichungen identifiziert. Nach der Überprüfung der Abstracts anhand der Ein- und Ausschlusskriterien (siehe 3.1) verblieben für die weitere Auswertung 265 relevante Beiträge.

3.2.1 Zeitliche Verteilung der Beiträge

Dieser Abschnitt dient der Beantwortung der Unterforschungsfrage *UFF1: Wie groß ist und war die Forschungsaktivität im Bereich SQM?* Eine Untersuchung der Beiträge nach Jahren (Abbildung 7) zeigt demnach, dass das Thema Software-Qualitätsmanagement durchaus an Relevanz gewonnen hat. Seit 1980 wurde in den betrachteten Journals und Konferenzen zum Thema SQM veröffentlicht. Schwerpunkt waren dabei Qualitätsmerkmale für die Softwarewartung. Erfahrungen aus der Vergangenheit (vgl. 1.2) zeigen, dass die Aufmerksamkeit bezüglich der Qualität eng mit der Produktreife zusammenhängt. So kann die zeitliche Entwicklung der Veröffentlichungen mit der Verbreitung der Software in Unternehmen und Gesellschaft erklärt werden. Mit der aufkommenden Softwarekrise seit 1960[111] und der wachsenden Etablierung komplexer IT-Systemlandschaften in Unternehmen in den 1990er Jahren gewann die SW-Qualität deutlich an Relevanz. Die Etablierung des Internets und des Mobiltelefons sowie Handheld-Computer könnten dieses Relevanzwachstum

[111] Vgl. Naur, Randell (1969)

unterstützt haben. Ein Grund für den Einbruch der Veröffentlichungszahlen nach 2000 könnte die Dotcom-Blase[112] sein. Bis auf das Jahr 2003 sind die Veröffentlichungszahlen bis 2005 relativ gering. In diesen fünf Jahren erholte sich die Internet-Branche von dem Crash. Neue Pionier-Unternehmen wie Google und Facebook mit klaren überschaubaren Ideen und minimaler Ausstattung nutzten Standard-Software-Lösungen[113], um neue Produkte im sog. Web 2.0[114] zu etablieren. Die Vernetzung über das Internet im betrieblichen wie auch im privaten Umfeld hatte sich etabliert. Aus dem Web 2.0 entwickelte sich das Web 3.0. Diese Entwicklungen erklären die Zahl der SQM-Veröffentlichungen der folgenden 10 Jahre. Seit 2016 geht die Zahl der Veröffentlichungen zurück. Dies könnte am Aufkommen des Internet 4.0, dem Internet der Dinge und verwandten Konzepten wie Industrie 4.0 liegen. Der Innovationscharakter einer Technologie steht im Fokus. Qualitätsaspekte treten in den Hintergrund.

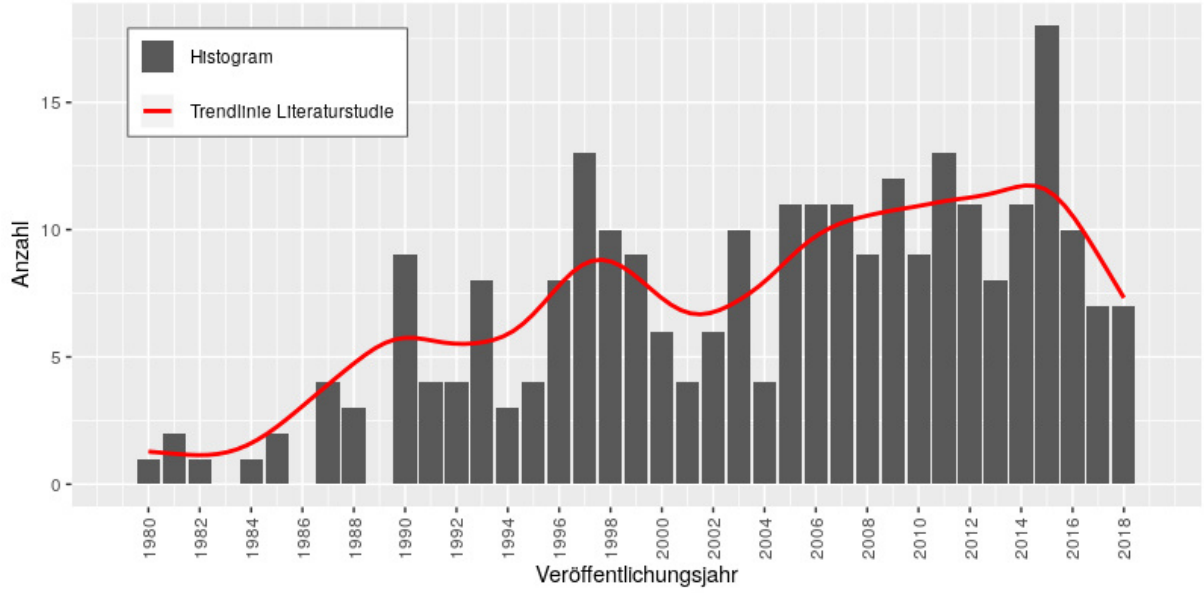

Abbildung 7: Veröffentlichungen im Bereich SQM aufgeteilt nach Jahren

[112] Der Begriff Dotcom-Blase ist ein Kunstbegriff der die im März 2000 geplatzte Spekulationsblase beschreibt, die insbesondere die sogenannten Dotcom-Unternehmen (Internetunternehmen) betraf und zu Vermögensverlusten für Kleinanleger führte.

[113] Vgl. Fischermann (2006)

[114] Web 1.0 ist ein Informationsnetz. Der Benutzer kann nur Informationen über Webseiten lesen und teilen. Web 2.0 ist die Read-Write-Netzwerkplattform, auf der Benutzer untereinander kommunizieren können. Web3.0 wird definiert als „Semantic Web“. Das Internet wird zu einer Sprache, die vom System gelesen und kategorisiert werden kann. Das Web 4.0 wird "Symbiotic Web“ genannt. Die Kommunikation mit dem Internet wird der menschlichen Kommunikation immer ähnlicher werden. (Vgl. Nath (2015))

3.2.2 Verteilung der Beiträge in Konferenzen oder Journals

Um die *UFF5 „In welchen Foren wird häufig zum Thema SQM veröffentlicht?“* zu beantworten, wurden die Beiträge anhand ihrer Veröffentlichung in Journals oder Konferenzen analysiert. Die 265 identifizierten Veröffentlichungen wurden in insgesamt 31 der 53 ausgewählten Journals und Konferenzen publiziert. Die Übereinstimmung von rund 60 Prozent zeigt, dass die Vorauswahl der Zeitschriften und Konferenzen den Untersuchungsgegenstand getroffen hat und dass das Thema Software-Qualitätsmanagement als relevant in der hochqualitativen Forschung angesehen wird. Auf 82 Prozent der in der Vorauswahl selektierten Konferenzen wurde zum Thema SQM veröffentlicht. Allerdings wurden nur in 57 Prozent der vorselektierten Journals relevante Veröffentlichungen identifiziert. Es scheint, dass auf wichtigen wissenschaftlichen Konferenzen das Thema SQM präsenter ist. Wie in Abbildung 8 zu sehen finden sich umfangreichere Beiträge in Journals. „Sonstige“ fassen sämtliche Journals und Konferenzen zusammen, in denen weniger als 10 relevante Veröffentlichungen identifiziert wurden.[115] Die meisten Veröffentlichungen sind im Journal IEEE Software, gefolgt vom Information and Software Technology Journal (IST). Die große Zahl der Veröffentlichungen in beiden Zeitschriften lässt sich durch deren Scherpunktsetzung erklären: Beide Journals dienen der Veröffentlichung von Forschungsbeiträgen zur Verbesserung der Softwareentwicklung. Dabei liegt ein ausdrücklicher Schwerpunkt von IEEE Software auf dem Testen von Software, wohingegen das IST-Journal auch andere SQM Aspekte berücksichtigt. Auch die große Zahl der Veröffentlichungen auf der International Conference on Software Engineering (ICSE) und dem International Symposium on Software Reliability Engineering (ISSRE) lassen sich mit deren Schwerpunkt auf Softwareentwicklung erklären. Americas Conference on Information Systems (AMCIS) und International Conference on Information Systems (ICIS) sind wichtige und große Konferenzen im Bereich Informationssysteme, wodurch sich die Zahl der Veröffentlichungen erklären lässt.

[115] Eine vollständige Liste der Quellen (Journals und Konferenzen) kann dem Protokoll der Literaturanalyse im Anhang entnommen werden.

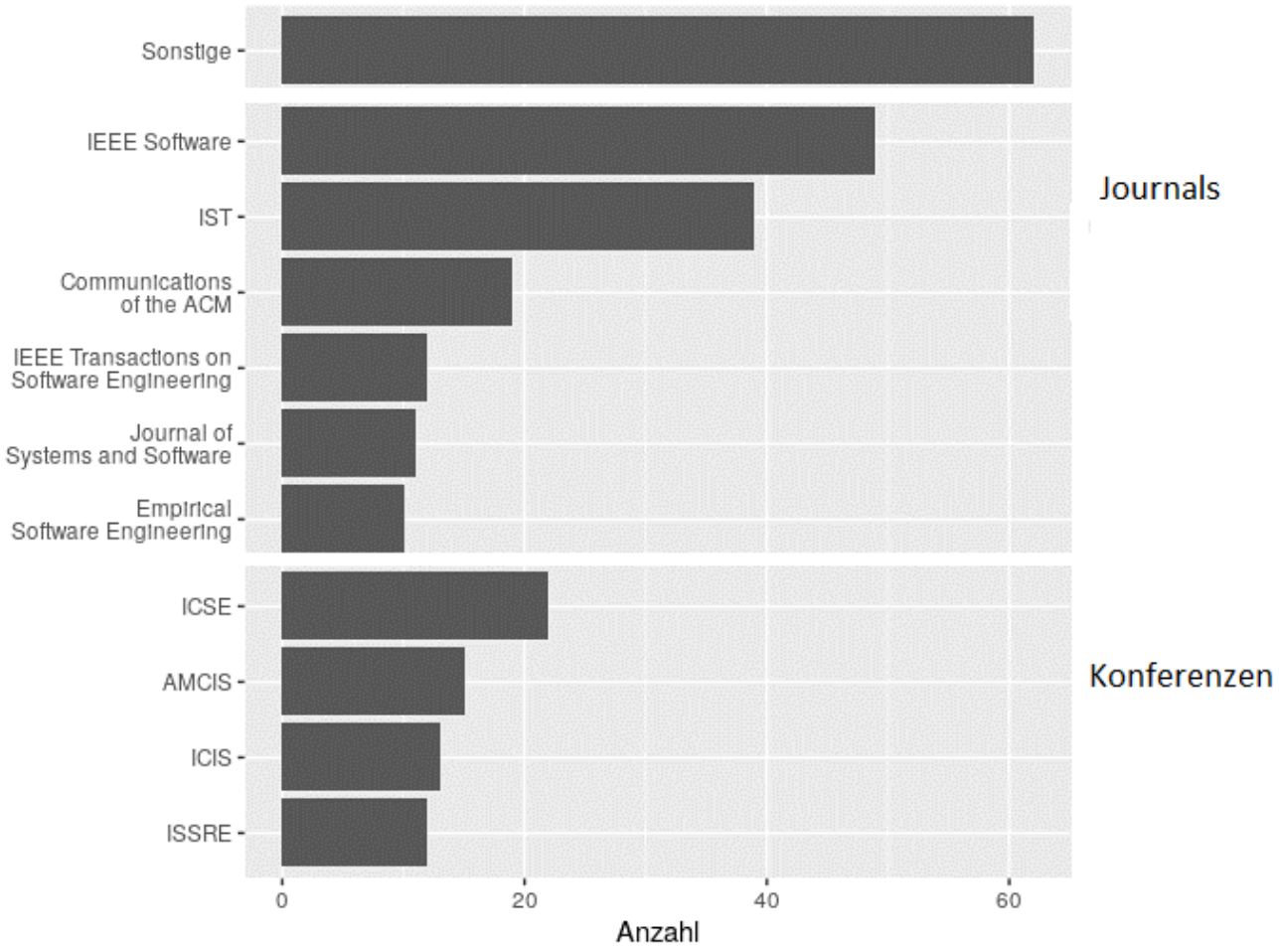

Abbildung 8: Anzahl der Veröffentlichungen im SQM aufgeteilt nach Quellen (Journals oder Konferenzen)

3.2.3 Ergebnisse der Studie nach Autoren

Die 265 Veröffentlichungen der Mapping-Studie wurden von 619 Autoren verfasst. Der Großteil der Autoren war nur an einer Veröffentlichung beteilig. Damit ist die Antwort auf *UFF4 „Wer sind die wichtigsten Forscher im Bereich SQM?“*: Mit Abstand am meisten Veröffentlichungen hat Taghi Khoshgoftaar, gefolgt von seinem Co-Autor Edward B. Allen. (Siehe Tabelle 12) Veröffentlichungen der beiden Autoren finden sich primär im Bereich von Modelling, Qualitätsvorhersage und Qualitätsmetriken. Weitere Zusammenarbeiten im Bereich der Forschung finden sich bei Khoshgoftaar, Allen und Naeem Seliya sowie bei Sandra Slaughter, Mayuram Krishnan und Donald Harter im Bereich Software Process Improvement. Barbara Kitchenham und Jørgen Bøegh forschten im Bereich SQ-Evaluation und Werkzeuge.

Name des Autors	Zugehörigkeit zu Forschungsorganisation	Anzahl der Veröffentlichungen
Taghi M. Khoshgoftaar	Empirical Software Engineering Laboratory, Department of Computer Science and Engineering, Florida Atlantic University	14
Edward B. Allen	Department of Computer Science and Engineering, Mississippi State University	6
Mayuram Krishnan	University of Michigan, Ann Arbor	4
Naeem Seliya	Empirical Software Engineering Laboratory, Department of Computer Science and Engineering, Florida Atlantic University	4
Sandra Slaughter	Carnegie Mellon University	4
Barbara Kitchenham	Keele University	3
Chris Kemerer	University of Pittsburgh	3
Donald. Harter	University of Michigan	3
E. Shihab	Queen's University	3
Jørgen Bøegh	Danish Electronics, Light, and Acoustics	3

Tabelle 12: wichtigsten Autoren der Mapping-Studie

3.2.4 Studienergebnisse in Abhängigkeit vom Thema

Zur Beantwortung von *UFF2 „Welche Forschungsthemen werden adressiert?"* wurden die Veröffentlichungen der Mapping-Studie den Oberthemen mit dem in Abbildung 6 definierten Klassifikationsprozess zugeordnet.

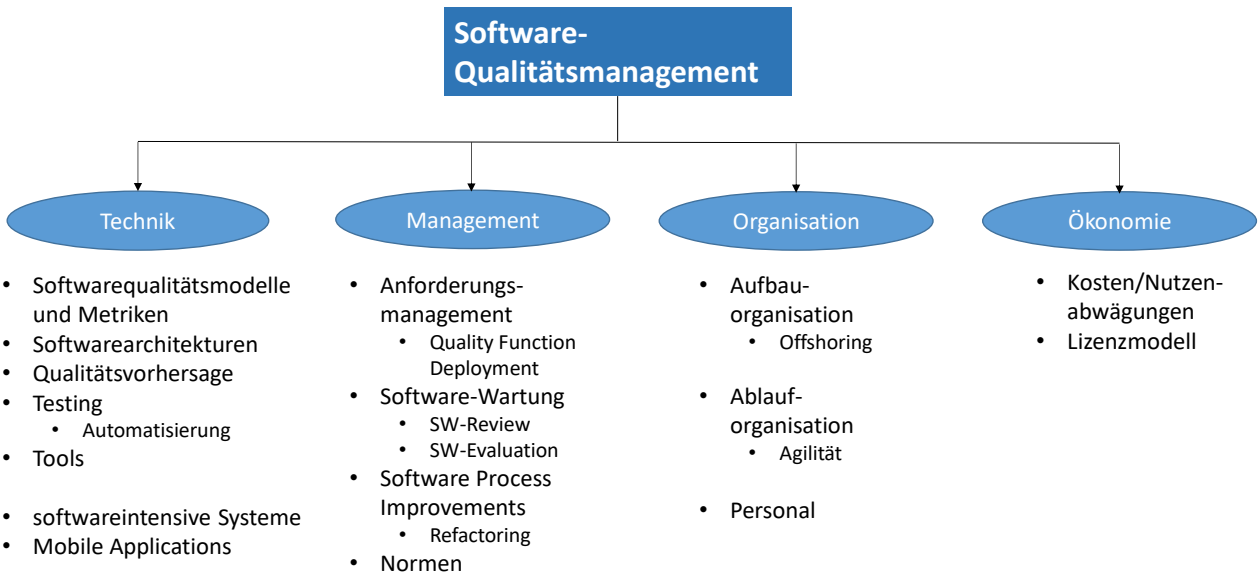

Abbildung 9: Oberthemen und Dimensionen im Softwarequalitätsmanagement

Das Oberthema **Software-Qualitätsmanagement** im Allgemeinen umfasst Artikel, die SQM auf einer generellen Ebene thematisieren. Dabei werden häufig allgemeine Modelle oder Fallstudien in einer bestimmten Branche, Region oder Firma behandelt. Auch die Untersuchung von Einflüssen und Methoden im SQM aus anderen Bereichen werden berücksichtigt. Unter dem Oberthema **Technik** werden Aspekte des SQM beleuchtet, die aus technische Gegebenheiten resultieren oder dadurch realisiert werden. Modelle und abgeleitete Metriken zur Messung und Vorhersage der Qualität werden berücksichtigt. Durchführung, Automatisierung und Einsatz von Tools dienen der Messung von SW-Qualität. Hierzu gehören auch neue technische Entwicklungen wie software-intensive Systeme und Apps. Das Oberthema **Management** befasst sich mit Regelungsstrukturen, die die Verbesserung der SW-Qualität unterstützen. Dabei werden sowohl Anforderungsmanagement vor der Softwareentwicklung als auch Software-Wartungsprozesse und Verbesserungsmanagement nach der Entwicklung berücksichtigt. Normen unterstützen die Einführung entsprechender Managementprozesse. Das Oberthema **Organisation** berücksichtigt Ablauf- wie auch Aufbauorganisation des SQM im Unternehmen. Damit im Zusammenhang steht der Personaleinsatz. Das fünfte Oberthema befasst sich mit der **Ökonomie** von SQM. Dabei werden vor allem Kosten- und Nutzenabwägungen berücksichtigt. Eine Besonderheit bei der Bereitstellung von Software ist das Lizenzmodell.

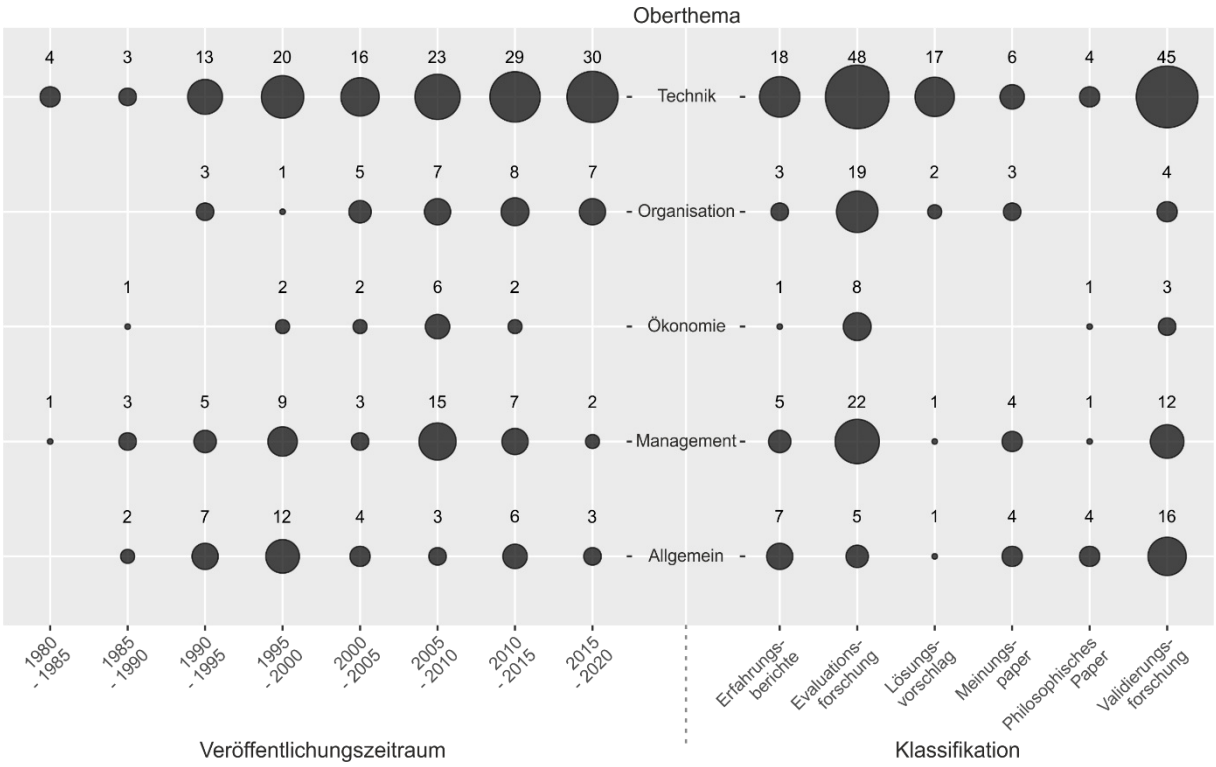

Abbildung 10: SQM-Map der Wissenschaft

Um die Unterforschungsfrage 3 (*UFF3 „Wie haben sich diese Schwerpunkte über die Zeit verändert?“*) zu beantworten, wurde ein Blasendiagramm über die Zeit in Abhängigkeit von den eben erläuterten Oberthemen erstellt (siehe Abbildung 10). Die Größe einer Blase ist proportional zur Anzahl der Artikel pro Oberthema bzw. pro fünf Jahre. Die Ordinate stellt die Klassifikation nach Themen dar. Auf der Abszisse werden zeitliche Verteilung und Forschungsansatz dargestellt. Vorteil eines so gestalteten Blasendiagramms gegenüber einem Histogramm ist, dass drei Dimensionen gleichzeitig dargestellt werden können, die einen schnellen Überblick über den untersuchten Forschungsbereich bieten. Im weiteren Verlauf dieses Kapitels wird die **zeitliche Entwicklung der Oberthemen** anhand der identifizierten Dimensionen im Detail aufgezeigt.

3.2.4.1 SQM im Allgemeinen

Es wurden 34 Beiträge identifiziert, die SQM im Allgemeinen thematisieren. Die erste Veröffentlichung aus 1987 beschäftigt sich bereits mit der Zukunft des Software-Qualitätsmanagements.[116] Dennoch behandeln weitere Beiträge aus den frühen 1990ern Grundlagen wie messorientierte[117], objektorientierte[118] oder integrierte Ansätze[119]. Es werden weitere Qualitätsmetriken und statistische Modelle zur Qualitätsbewertung und der Verbesserung des Software Engineering Prozesses erarbeitet.[120] Diese Beiträge sind primär theoretischer Natur (Validierungsforschung). 1996 wurde erstmals das SQM für Unterhaltungselektronik thematisiert.[121] Veröffentlichungen aus den späten 90ern sind stärker praktisch orientiert. Die zunehmende Durchdringung der Geschäftsprozesse mit Informationssystemen führt dazu, dass SQM von einem passiven zu einem aktiven Prozess, von der Fehlererkennung zu Fehlervermeidung übergeht. Themen sind dabei Faktoren, die das Management von Softwarequalität verbessern und es wird untersucht, inwieweit sich Total Quality Management im Bereich der Softwareentwicklung anwenden lässt.[122] 1998 führten Rai u.a. (1998) eine erste Literaturstudie zum Bereich SQM durch.[123] Ergebnisse dieser Studie werden im weiteren Verlauf mit Ergebnissen der

[116] Vgl. Cavano, LaMonica (1987).
[117] Vgl. Card (1988).
[118] Vgl. Henderson-Sellers (1991).
[119] Vgl. Card (1990).
[120] Vgl. Dalal u.a. (1993); Rooijmans u.a. (1996).
[121] Vgl. Rooijmans u.a. (1996).
[122] Vgl. Alexander, Ambrose (1999).
[123] Vgl. Rai u.a. (1998).

Mapping-Studie dieser Arbeit verglichen. Ende der 1990er gewinnt die Prozessoptimierung mittels CMM an Bedeutung.[124] Neue flexible Entwicklungsmodelle für den gesamten Softwarelebenszyklus werden wichtiger. Entwickler lernen agile Methoden zu schätzen. 2008 kommt der Ansatz, Software als Vermögenswert zu betrachten und zu managen erstmals auf.[125] Die Qualität von mobilen Applikationen wird relevant.[126] Zentrum der Qualitätsbewertung sind nicht mehr nur quantitative Qualitätsmetriken, sondern Kunden- und Benutzeranforderungen.[127] Eine Fallstudie aus dem Jahr 1990 beschreibt die Bellcore Quality Assurance Engineering Software, die Qualitätssicherungsprogramme für Endverbrauchersoftware entwickelt und implementiert.[128] Eine Studie aus 2013 untersuchte den Stand der Softwarequalität in belgischen Unternehmen und die gängigsten Branchenpraktiken (Prozesse, Techniken und Tools) bezüglich der Softwarequalität.[129]

3.2.4.2 Technik

Quantitative Softwarequalitätsabschätzung ist in den 90er Jahren sehr verbreitet, es gibt 30 Beiträge. Typische Schätzungsmodelle wie von McCabe und Halstead beschäftigen sich mit der quantitativen Abschätzung des Quellcodes.[130] Grundlegende Fragen im Zusammenhang mit dem Modellierungsprozess einschließlich der Probleme der gemeinsamen Varianz zwischen Metriken werden untersucht.[131] Häufigste Metriken sind Metriken zur Messung von Qualitätsattributen, Reduktion des Wartungsaufwands, Erhöhung der Zuverlässigkeit und Vorhersage-Metriken.[132] Ende der 90er Jahre rückt die Kundenzufriedenheit in den Fokus der Untersuchungen.[133] In den frühen 2000ern liegt das Augenmerk auf der Kombination vorhandener Modelle, um die Vorhersagegenauigkeit für den jeweiligen Anwendungskontext zu optimieren.[134] Um die Qualität komplexer softwareintensiver Systeme zu verbessern werden Modelle aus der Biologie übernommen und angepasst.[135] Mit Hilfe von

[124] Vgl. Rooijmans u.a. (1996); Baker (2001).
[125] Vgl. Ben-Menachem (2008).
[126] Vgl. Denning (2016).
[127] Vgl. Steidl u.a. (2014).
[128] Vgl. Hon (1990).
[129] Vgl. Perez u.a. (2013).
[130] Vgl. Hirayama u.a. (1990).
[131] Vgl. Munson, Khoshgoftaar (1990).
[132] Vgl. Yau, Collofello (1980); Kafura, Henry (1981); Poore (1988).
[133] Vgl. Ambrose, Eynon (1998).
[134] Vgl. Bouktif u.a. (2002).
[135] Vgl. Azar, Vybihal (2011).

prädiktiven Modellen können einzelne Komponenten von SIS identifiziert werden, die für eine zukünftige Wartung potenziell problematisch sind. Webanwendungen im B2B-Bereich werden untersucht.[136] Nach 2010 gewinnen agile Methoden zur Anwendung von Modellen und deren Kombination weiter an Relevanz. Unterschiedliche Qualitäts-wahrnehmungen führender operativer Qualitätsmodelle bei der Bewertung von Softwaresystemen werden aus Branchensicht untersucht.[137] Der Bereich der Modellierung ist eher theoretischer Natur: es gibt viel Validierungsforschung.

Die Untersuchung von Softwarearchitekturen wurde als relativ junges Feld identifiziert. Die erste Veröffentlichung stammt aus dem Jahr 2001. Das Meinungspaper präsentiert zwei Steuerungsmöglichkeiten der Qualitätskosten: Einerseits kann die Softwarearchitekturauswertung verwendet werden, um Eigenschaften wie Sicherheit und Vollständigkeit zu überprüfen, andererseits können Qualitäten wie Wartbarkeit, Leistung und Zuverlässigkeit vorhergesagt werden.[138] Es werden Möglichkeiten zur Erhöhung der Akzeptanz automatisierter Werkzeuge zur Architekturentwicklung in der Praxis diskutiert. [139]Softwaresysteme leiden bei der Entwicklung tendenziell unter architektonischen Problemen. Während moderne Softwareentwicklungsmethoden wie Agile und Dev-Ops verschiedene Möglichkeiten zur Sicherung der Codequalität vorschlagen, wird weniger Wert auf die Qualitätssicherung der Architektur der sich entwickelnden Systeme gelegt. Ein Beitrag aus 2015 stellt eine Lösung zur automatischen Erkennung von Architekturverletzungen in Software-Artefakten vor und implementiert diese.[140] Um vorhandenes Wissen aus Architekturanalysen wieder verwenden zu können, wurden in einem Beitrag aus 2018 Architektur-Templates entwickelt.[141]

Vorhersage der SW-Qualität zielt darauf ab, das Qualitätsniveau regelmäßig zu bewerten und Probleme frühzeitig zu erkennen, es wurden 21 Veröffentlichungen identifiziert. Im Idealfall helfen Software-Fehlerprognosemodelle bei der Ressourcenoptimierung und senken Kosten durch Identifikation der fehleranfälligsten Softwaremodule. Ein früher Ansatz misst die Qualität an der Ausfallintensität.[142]

[136] Vgl. Behkamal u.a. (2009).
[137] Vgl. Izurieta u.a. (2017).
[138] Vgl. Barber, Holt (2001)
[139] Vgl. Avgeriou u.a. (2014).
[140] Vgl. Goldstein, Segall (2015).
[141] Vgl. Wu u.a. (2018).
[142] Vgl. Okumoto (1985).

Anwendung einer Softwarezuverlässigkeitsmessung in Echtzeit soll die vom Benutzer erhaltene Softwarequalität bei kontrollierten Lasttests überwachen.[143] Mittels Regressionsanalysen und Schätzverfahren wird eine Vorhersage über die Softwarequalität getroffen.[144] Ab 2000 nutzen Ansätze der Qualitätsbewertung Techniken der künstlichen Intelligenz wie beispielsweise neuronalen Netzwerken und Support Vector Machines.[145] Ein neuartiger Beitrag verfolgt den Ansatz, mittels Anforderungs-Rückverfolgbarkeitsmatrix prädiktive Informationen bereitzustellen, bevor ein Code geschrieben wird.[146] Ein Software-Qualitätsvorhersagemodell nutzt einen genetischen Algorithmus, bei dem Ausreißererkennung und Merkmalsauswahl gleichzeitig ausgeführt werden, um die Prognosequalität zu steigern.[147] 2013 wird in einem Beitrag die Kosteneffizienz der Vorhersagemodelle untersucht.[148]

Erstaunlicherweise wurde nur ein Beitrag der Konferenz zum Thema Testing, Überprüfung und Validierung dem Schwerpunkt Testing zugeordnet, wo doch mit 26 Beiträgen das Testen zu den wichtigsten Dimensionen gehört.[149] Untersuchungen der 90er zu Code-Inspektion und Testen bestätigen deren positive Auswirkungen auf die Softwarequalität.[150] Neuere Beiträge fokussieren test-basierte Softwareentwicklung. Test Driven Development ist eine agile Praxis, die an Popularität gewonnen hat.[151] Die Zahl der Veröffentlichungen im Bereich Testing hat in den letzten 10 Jahren zugenommen. Neuere Testauswahl- und Bewertungsstrategien senken die Produktionskosten und verbessern die Agilität der Entwicklung.[152] Neben modellbasiertem Entwickeln werden bei SIS die modellbasierte Qualitätssicherung und die engere Integration von statischen und dynamischen Qualitätssicherungsaktivitäten immer wichtiger.[153] Die Wichtigkeit des Wissens und der Fähigkeiten der Entwickler im Test-Bereich werden empirisch überprüft und belegt.[154] Ein zukünftiges

[143] Vgl. Ehrlich u.a. (1990).
[144] Vgl. Khoshgoftaar u.a. (1992).
[145] Vgl. Reformat u.a. (2002).
[146] Vgl. Hayes u.a. (2005).
[147] Vgl. Wang u.a. (2007).
[148] Vgl. Zhang, Cheung (2013).
[149] Vgl. Masuda u.a. (2018).
[150] Vgl. Laitenberger (1998); Cobb, Mills (1990).
[151] Vgl. Crispin (2006).
[152] Vgl. Herzig u.a. (2015).
[153] Vgl. Kläs u.a. (2015).
[154] Vgl. Ayaburi u.a. (2016).

Forschungsfeld im Bereich Testing ist das Testen mit Machine Learning Algorithmen.[155]

Nach 2013 wurden Beiträge zum jungen Bereich Test-Automatisierung veröffentlicht. Grund dafür könnte der erst dann ausreichende Reifegrad der Testing-Ansätze sein. Es werden verschiedene Strategien zur automatisierten Verbesserung des Vorhersagemodells durch eine adaptive Auswahl neuer Messpunkte zur Senkung der Testkosten vorgestellt.[156] Qualitätsmodelle gelten als anerkannter Ansatz zur Beurteilung hochrangiger abstrakter Qualitätsmerkmale durch Aggregation aus Kennzahlen. Ein neuer Ansatz stellt ein probabilistisches Gewicht anstelle des subjektiven Gewichts früherer Aggregationsmethoden der Kennzahlen dar.[157] Der aktuellste Beitrag (2018) untersucht die Anwendung von drei komplementären Techniken zur automatisierten Analyse der Softwarearchitektur, die die Lokalisierung von fehleranfälligen und wartungsintensiven Architekturelementen ermöglicht. [158]

Die Dimension praxisorientierte Tools umfasst einzelne Werkzeuge, die die Entwicklung von Standards, Konventionen und Methoden für alle Phasen des Entwicklungsprozesses unterstützen.[159] OOMeter ist ein Werkzeug, das Qualitätsmerkmale quantitativ messbar macht.[160] Die Telekommunikations-Software & Systems Group arbeitet mit Open-Source-Tools zum Aufbau von Qualitätsebenen für qualitative Produkte.[161] Lediglich die neuste Studie (2017) trägt auf einer Metaebene zur systematischen Qualitätsbewertung einzelner QM-Tools und möglicher Verbesserungsvorschläge bei.[162]

Aus der Integration heterogener unabhängiger Systeme sind heute komplexe softwareintensive Systeme entstanden, der erste Beitrag von vieren stammt aus 2009. Der Stand der Technik offenbart jedoch qualitätsbezogene Mängel, da ihre inhärenten Eigenschaften oft nicht ausreichend berücksichtigt werden. Das SERIOUS-Projekt schlägt einen Ansatz für die Anwendung evolutionärer Softwareentwicklung vor. [163] Ein

[155] Vgl. Masuda u.a. (2018)
[156] Vgl. Westermann u.a. (2012).
[157] Vgl. Yan u.a. (2017).
[158] Vgl. Mo u.a. (2018).
[159] Vgl. Gustafson, Kerr (1982).
[160] Vgl. Alghamdi u.a. (2005).
[161] Vgl. Dowling, McGrath (2015).
[162] Vgl. Guzman u.a. (2017).
[163] Vgl. van Rompaey u.a. (2009).

weiterer architekturzentrierter Ansatz für die Entwicklung softwareintensiver Systeme mit Fokus auf die formale Spezifikation und dynamische Rekonfiguration ihrer Architekturen wird vorgeschlagen.[164]

Mobile Applikationen werden zu komplexen Softwaresystemen, die schnell entwickelt werden müssen und sich ständig weiterentwickeln, um neuen Benutzeranforderungen und Ausführungskontexten gerecht zu werden. Da kaum Ansätze vorhanden sind, werden Qualitätsmanagement-Ansätze benötigt.[165]

3.2.4.3 Management

In den 90er Jahren wurde die Bedeutung der wahrgenommenen Produktqualität zwar weltweit anerkannt, es gab jedoch keine strengen Messmethoden für die Kundenwahrnehmung.[166] Anforderungen, sog. „Requirements“, sind ein kleiner Bereich des SQM mit fünf Veröffentlichungen. Herausforderungen sind sich widersprechende ungenaue Softwareanforderungen und unklare Prioritäten der Qualitätsanforderungen. Ansätze behandeln die möglichst objektive Priorisierung von Anforderungen[167], Ermittlung von Architekturanforderungen[168], Bewertung von erwarteter Qualität und die Kategorisierung von Anforderungsfehlerquellen in eine formale Taxonomie.[169] Eng verbunden damit ist Quality Function Deployment (QFD). Zwei Veröffentlichungen aus dem Jahr 1993 untersuchen die Nutzung von QFD im Softwarebereich. Unrichtigkeit, Unvollständigkeit oder Inkonsistenz der Benutzeranforderungen führen zu erheblichen Budgetüberschreitungen. SQFD hilft bei der korrekten Spezifikation der Nutzeranforderungen erheblich.[170]

Zum Thema der praxisorientierten Softwarewartung wurden vor allem in den 80er Jahren sechs Problemfaktoren identifiziert: Benutzerwissen, Programmierer-Effektivität, Produktqualität, Verfügbarkeit der Programmierer, Maschinenanforderungen und Systemzuverlässigkeit.[171] Ein weiterer Beitrag untersucht die Einflussfaktoren auf die Wartungsproduktivität. Bedarf der

[164] Vgl. Cavalcante (2015).
[165] Vgl. Goldbach u.a. (2014).
[166] Vgl. Xenos, Christodoulakis (1997).
[167] Vgl. Liu (1998).
[168] Vgl. Niemelä, Immonen (2007).
[169] Vgl. Walia, Carver (2009).
[170] Vgl. Erikkson, McFadden (1993); Haag u.a. (1996).
[171] Vgl. Lientz, Swanson (1981).

Automatisierung im Bereich der Wartung wird ermittelt.[172] Neuere Beiträge befassen stärker mit Metriken für die Wartung. Überprüfungen und Inspektionen von Software-Artefakten sind effektive Techniken zur Fehlererkennung und Verbesserung der Softwarequalität. Baker stellte 1997 fest, dass bei Kosten von 1-2% des Projekts ein Rückgang der Problemzahl um 40% erreicht wurde.[173] Während Überprüfungsmethoden für textbasierte Artefakte (z.B. Code) gut verstanden werden, gibt es nur sehr wenige Anleitungen für die Überprüfung von Softwarediagrammen.[174] Auch im Bereich Review wurden zuerst theoretische Überlegungen identifiziert. Mit der ersten Veröffentlichung aus 2010 ist der Bereich Software-Evaluation ein junges, praxisorientiertes Themenfeld. Die Veröffentlichungen bieten Maßnahmen zur Bewertung der Softwarequalität von Java oder C++ Anwendungen.[175] Ein weiterer Ansatz ist die Messung der Softwarequalität und Zuverlässigkeit, die auf der komplexen Netzwerktheorie basiert.[176]

Die Dimension des Software Process Improvements (SPI) ist stark praktisch orientiert. Alle Beiträge bis auf den neusten sind Erfahrungsberichte oder dienen der Evaluationsforschung. In den acht Jahren zwischen den ersten beiden Beiträgen (1992 und 1996) hat Raytheon nachhaltige, signifikante und messbare Verbesserungen seines Software-Engineering-Prozesses erzielt.[177] Auch PRC Inc. und weitere Unternehmen konnten mit Hilfe des CMM Modells des Software Engineering Institutes Software-Prozessverbesserungen durchführen.[178] Nach 2000 verschiebt sich der Fokus der Beiträge von der reinen Anwendung und Untersuchung der Effizienz von Software Process Improvement auf die Untersuchung der Einflussfaktoren auf erfolgreiches SPI.[179] Spätere Veröffentlichungen fokussieren kosten- und zeiteffizientere Ansätze für beispielsweise die Prozessbewertung oder Nutzung von Werkzeugen. Refactoring als eine Methode zur Verbesserung der Softwarequalität, beschreibt die Designverbesserung des bestehenden Codes durch Änderung seiner internen Struktur, ohne sein externes Verhalten zu beeinflussen. Mit nur zwei Veröffentlichungen zum Thema hat es nur eine untergeordnete Rolle, sie

[172] Vgl. Grady (1993).
[173] Vgl. Baker (1997).
[174] Vgl. Zhang, Kim (2010).
[175] Vgl. Darmawan u.a. (2010); Nakamura, Hamagami (2012).
[176] Vgl. Ai u.a. (2018).
[177] Vgl. Dion (1992);Haley (1996).
[178] Vgl. Hollenbach u.a. (1997).
[179] Vgl. Gree u.a. (2005); Mathiassen u.a. (2005).

widersprechen sich bezüglich des Einflusses von Refactoring auf die Softwarequalität.[180]

Drei Beiträge befassen sich mit der Anwendung von ISO-Normen. Es wurde eine Bewertung der Softwareumgebung entwickelt, die auf einem in ISOAEC 9126 definierten Prozessmodell basiert. Die Ergebnisse einer Studie aus 2004 zeigen Unklarheiten in der Art und Weise, wie die ISO/IEC 9126 in Bezug auf Merkmale und Submerkmale strukturiert ist.[181] Ein Meinungspaper von 2008 beschreibt die neue Norm ISO/IEC 25030 über Software-Qualitätsanforderungen.[182] Neben den Normen ist vor allem das Capability Maturity Model (CMM) Version 1.1 von 1993 relevant. Es stellt einen Common-Sense-Engineering-Ansatz zur Verbesserung von Software-Prozessen dar. Die Reifegrade, die wichtigsten Prozessbereiche und die wichtigsten Praktiken wurden in der Software-Community überprüft. Das CMM stellt es einen breiten Konsens der Software-Community dar. In den darauffolgenden Jahren haben weitere empirische Studien den Erfolgsbeitrag von CMM bestätigt.[183]

3.2.4.4 Organisation

Der Themenbereich Organisation und Personal ist ein relativ großer und junger Bereich mit 24 Beiträgen. Nur drei sind aus den 90er Jahren. Sie beschreiben, wie SW-Teams idealerweise gebildet werden, wie die Qualitätsmessung an lokale Entwicklergruppen angepasst werden können und untersuchen die Standards von Entwicklern abhängig von deren Geschlecht.[184] Alle anderen Beiträge wurden nach 2000 verfasst. Beiträge untersuchen die Rolle der Kommunikation innerhalb eines Entwicklungsteams um Wissen weiter zu geben und zwischen Entwicklungsteams und Management.[185] Der Einfluss der Unternehmenspolitik auf SW-Qualität wird beleuchtet. Die Zentralisierung ist ein Treiber des Projekterfolgs, um Kosten zu senken. Verteiltes Arbeiten unterscheidet sich u.a. in Prozessvarianz, Ausführung und Expertise, unterschiedliche Arbeitsumgebungen, Werkzeuge und Entwicklungspraxis.[186] Neuere Studien zeigen, dass fehlende Verbindungen innerhalb des Teams häufige Fehlerquelle sind. Auch persönliche Merkmale und

[180] Vgl. Alshayeb (2009); Liu u.a. (2007).
[181] Vgl. Miyoshi, Azuma (1993).
[182] Vgl. Bøegh (2008).
[183] Vgl. Paulk u.a. (1993).
[184] Vgl. Rettig (1990); Trammell, Poore (1992); Hall (1995).
[185] Vgl. Huffman Hayes (2003).
[186] Vgl. Nguyen-Duc u.a. (2015).

Teamzufriedenheit beeinflussen Softwarequalität. Eine stärkere Identifikation mit dem gemeinsamen Ziel verringert die Fehlerzahl. Bekannte Qualitätskennzahlen für agile Softwareentwicklung werden kombiniert mit sozialen Kenngrößen, um die Beteiligten frühzeitig einzubinden.[187]

Die Dimension Offshoring ist sehr klein (drei Veröffentlichungen), jung und praxisorientiert. Die Beiträge untersuchen Qualitätsveränderung, wenn die Entwicklung in einer verteilten Umgebung stattfindet, den Einfluss von Prozessreife und Personal und wie sich verschiedene Anreizstrukturen in Verträgen auswirken.[188] Eine Studie von 2009 untersucht empirisch die Gesamtentwicklung von Windows Vista und vergleicht die Ausfälle von Komponenten, die nach dem Release dezentral entwickelt wurden, mit denen, die von regionalen Teams entwickelt wurden.[189]

Die erste Veröffentlichung der Dimension Agilität war 2002. Seit der Softwarekrise wurden Anstrengungen unternommen um Zeit-, Kosten- und Qualitätsprobleme der Softwareentwicklung anzugehen. Erste Bemühungen führten zu präskriptiven strukturierten Methoden. Auf Grund steigender Dynamik der Anforderungen haben sich agile Ansätze entwickelt.[190]

3.2.4.5 Ökonomie

Die Ökonomie des SQM wird in acht Beiträgen thematisiert. In Veröffentlichungen der 80er und 90er Jahre wurde Qualitätssicherung als kritischer Faktor für die erfolgreiche Entwicklung von Software-Systemen erkannt. Die Bedeutung der Kosteneffizienz von Qualitätssicherung wurde kaum betrachtet. Zielkonflikte wurden identifiziert, ein integratives dynamisches Modell zur Kontrolle SQM-Strategien erarbeitet.[191] Die Fehlerdichte reduzierte sich mit sinkenden Grenznutzen durch SW-Qualitätsinitiativen. Um Synergien zu realisieren müssen Hauptprobleme in den Bereichen Entwicklung, Management, Qualitätssicherung und Anforderungskonformität frühzeitig eliminiert werden.[192] Nach 2006 wurden auf Basis von COCOMO, ein algorithmisches Kostenmodell, das zur Kosten- bzw. Aufwandsschätzung genutzt wird, COSEEKMO und COQUALMO entwickelt, um beispielsweise ein Kosten/Nutzen-Optimum

[187] Vgl Datta (2018); Çaglayan, Bener (2016); Gómez, Acuña (2014).
[188] Vgl. Kannabiran, Sankaran (2011);
[189] Vgl. Bird u.a. (2009). Gopal, Koka (2010).
[190] Vgl. Baskerville u.a. (2002)
[191] Vgl. Abdel-Hamid (1988).
[192] Vgl. Slaughter u.a. (1998).

zwischen schnellem Release und Umsatzverlust zu finden.[193] Bei der SW-Qualitätssicherung spielt Testen eine große Rolle. Eine Fallstudie zeigt den großen ROI des Testens bei deutlichem Wertverfall eines Tests durch Produktänderungen und im Zeitablauf.[194]

Das Thema Open Source ist ein sehr junger und kleiner Bereich (vier Artikel). Diese Dimension wird dem Oberthema Ökonomie zugeordnet, weil es sich dabei um eine spezielle Form der Lizensierung handelt. Die erste Veröffentlichung dieser Literaturstudie ist aus 2002 und beschreibt die Ergebnisse einer Pilotstudie, die die Auswirkungen der strukturellen Code-Qualität in Open Source Projekten untersucht.[195] In einer weiteren Studie wird die Codequalität von open-source Projekten untersucht, deren Qualität nicht unbedingt niedriger ist als die von Closed-Source-Software.[196] Der Erfolg von Open-Source-Software wird auf ihre Praktiken und Organisationsstruktur zurückgeführt. Folglich gibt es einen Trend im Unternehmensumfeld, einige dieser Praktiken zu kopieren und anzupassen. Zwei Beiträge untersuchen deshalb die Erfolgsfaktoren und erarbeiten ein Modell bzw. Framework.[197]

3.3 Vergleich der Ergebnisse der Mapping-Studie mit den Veröffentlichungen im Software Quality Journal

Der Vergleich der Ergebnisse der Mapping-Studie dieser Arbeit mit den Veröffentlichungen im Software Quality Journals erfolgt auf einer hohen Abstraktionsebene. Ziel des Software Quality Journals ist die Förderung des Bewusstseins für die entscheidende Rolle des Qualitätsmanagements bei der effektiven Konstruktion von Unternehmenssoftwaresystemen. Es bietet ein Forum für den Erfahrungs- und Informationsaustausch über SQM, seine Methoden, Werkzeuge und Produkte und veröffentlicht wissenschaftliche Arbeiten, die sich auf alle Aspekte der Softwarequalität beziehen. Beiträge von Praktikern und Wissenschaftlern sowie

[193] Vgl. Mens, T. (2012).
[194] Vgl. Nikolik (2012).
[195] Vgl. Stamelos u.a. (2002).
[196] Vgl. Mishra u.a. (2002).
[197] Vgl. Nelson, Santos (2007), Kozlov u.a. (2007).

von nationalen und internationalen Politik- und Normungsgremien werden veröffentlicht.[198]

Name des Autors	Anzahl der Veröffentlichungen, die in der Mapping Studie identifiziert wurden	Anzahl der Veröffentlichungen im Software Quality Journal (Rang)
Taghi M. Khoshgoftaar	14	16 (1)
Edward B. Allen	6	6 (11)
Mayuram Krishnan	4	-
Naeem Seliya	4	5 (22)
Sandra Slaughter	4	-

Tabelle 13: Autorenvergleich der Mapping-Studie mit den Autoren des Software Quality Journals

Die 724 untersuchten Artikel des Software Quality Journals wurden von 1493 Autoren verfasst. 100 Autoren, die in der Mapping-Studie identifiziert wurde, publizierten auch im Software Quality Journal. Da es sich um Autoren mit besonders vielen Veröffentlichungen handelt, kann angenommen werden, dass in der Literaturstudie relevante Beiträge im Bereich SQM identifiziert wurden. Wie in der Mapping-Studie identifiziert, ist auch im Software Quality Journal Taghi M. Khoshgoftaar der Autor mit den meisten Veröffentlichungen, gefolgt von seinem Coautor Edward B. Allen. Mayuram Krishnan, Sandra Slaughter haben im Software Quality Journal nicht veröffentlicht. Der Großteil ihrer Veröffentlichungen im Bereich SQM erfolgte auf Konferenzen und nicht in Journals.

[198] Springer Nature Switzerland AG (2018), URL siehe Literaturverzeichnis

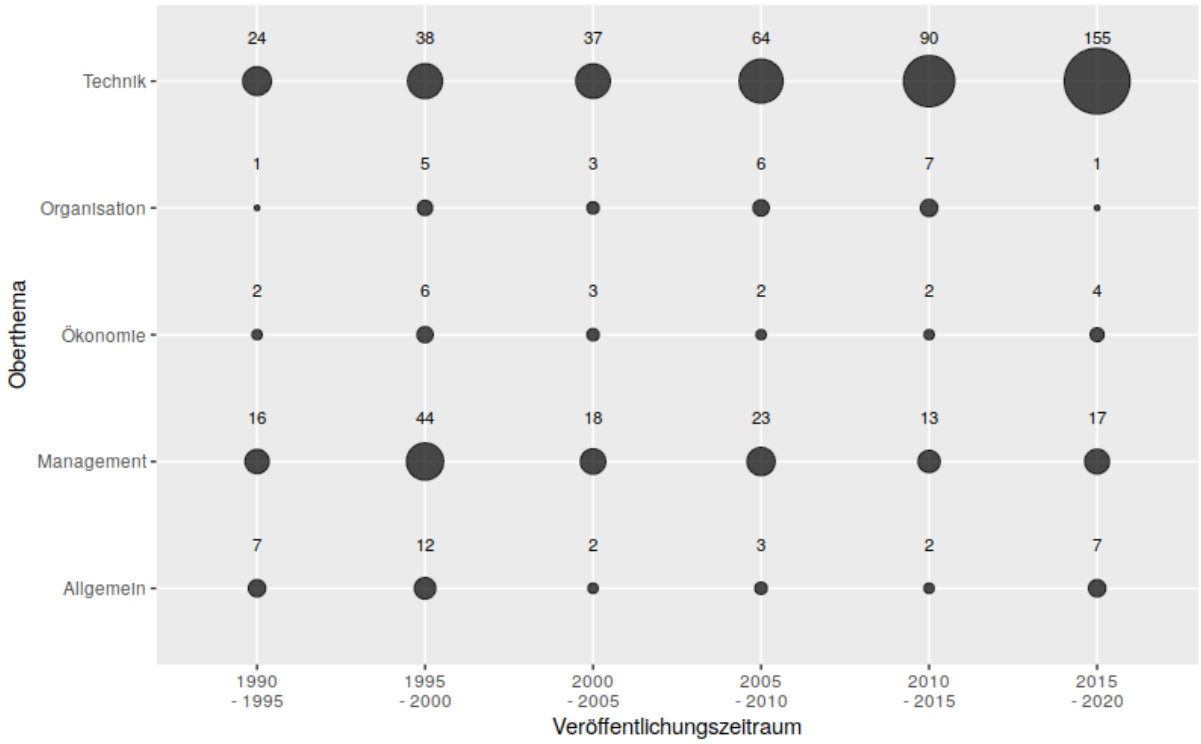

Abbildung 11: Software Quality Journal Map

Die Artikel des Journals wurden anhand der Titel manuell den Dimensionen der Mapping-Studie zugeordnet und zeigen ein ähnliches Bild. Technische Themen sind auch im Software Quality Journal dominierend und werden im Zeitverlauf signifikant mehr. Vor allem im Bereich Testing nehmen die Veröffentlichungszahlen überproportional zu. Auf Grund der zunehmenden Komplexität von Software und der gestiegenen Hardwareleistung wird das Testen von Software immer herausfordernder und performanter. Lediglich in den Jahren 2000 bis 2005 ist ein kleiner Rückgang der Veröffentlichungszahl im Bereich Technik zu vermerken. Grund dafür kann auch hier die Dotcom-Blase 2000 gewesen sein. Die unerwartet geringe Behandlung der organisationalen Verankerung des SQM kann in Zukunft zu Problemen führen, da komplexer werdende Lieferketten Organisation und Management vor neue Herausforderungen stellen. Bezüglich der Veröffentlichungszahlen zum Thema Management zeigt sich, dass sowohl in der Mapping-Studie als auch im Journal in den Jahren zwischen 1995 und 2000 sowie zwischen 2005 und 2010 mehr zum Thema veröffentlicht wurde. Im Jahr 2000 gab es keine Ausgabe des Journals. Die Veröffentlichungszahlen nach 2000 waren entgegen der Erwartungen trotz Internetblase konstant. Allgemeine Beiträge zum Thema SQM wie Fallstudien spielen entgegen der Erwartungen im Journal eine untergeordnete Rolle, obwohl das Journal explizit Praxisbeiträge veröffentlicht. Wie in Abbildung 11 und Abbildung 12 zu

erkennen, spiegeln die Ergebnisse der Mapping-Studie die zeitliche Schwerpunktsetzung der Forschung im Software Quality Journal gut wieder. Lediglich in den letzten 5 Jahren (seit 2014) ist eine Diskrepanz zwischen den Verläufen der roten und blauen Kurve zu erkennen. In den softwarespezifischen Journals und Konferenzen nahmen die Veröffentlichungszahlen ab. Anscheinend ist das allgemeine Interesse an SQM rückläufig. Im Software Quality Journal werden es signifikant mehr Beiträge. Um den Grund für diese Diskrepanz genauer zu untersuchen, kann ein weiterer Forschungsschritt die Analyse der Journal-Beiträge auf Dimensionsebene sein, um wie bei der Mapping-Studie den Zeitverlauf der Forschungsschwerpunkte vertieft zu analysieren. Dies ist auf Grund der hohen Zahl und teilweise mangelnder Verfügbarkeit der Artikel im Rahmen dieser Arbeit nicht möglich.

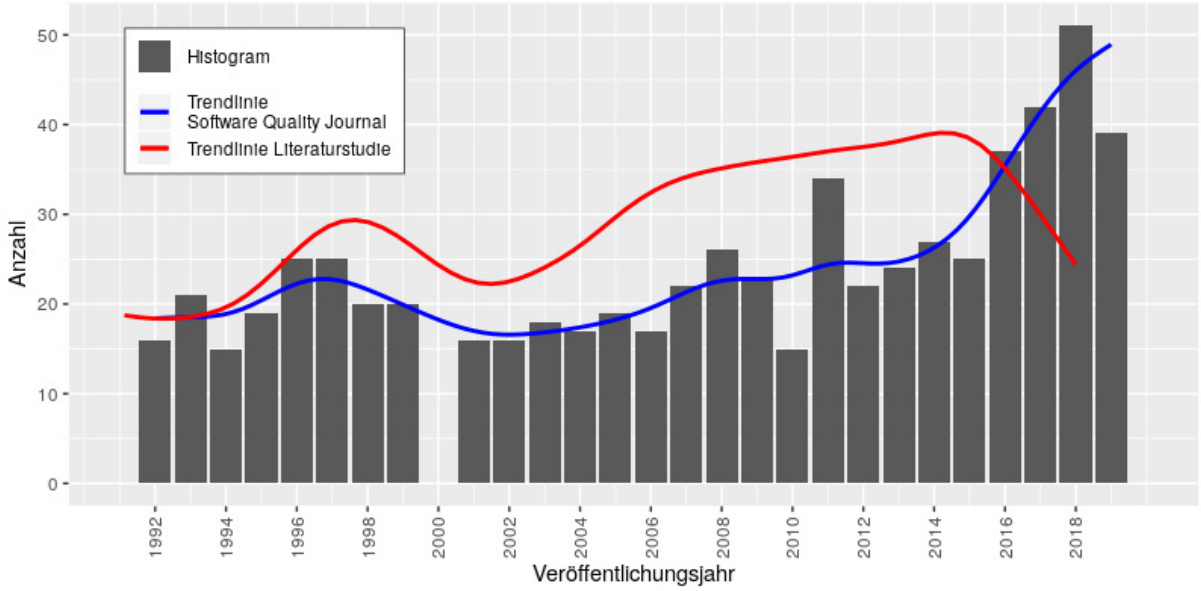

Abbildung 12: zeitlicher Verlauf der Veröffentlichungszahlen im Software Quality Journal

3.4 Validität der Mapping-Studie

Die größte Validitätseinschränkung dieser Mapping-Studie ist die unvollständige Auswahl von Publikationen durch Einschränkung der untersuchten Quellen. Obwohl ein systematischer Ansatz verfolgt wurde, ist es immer noch möglich, dass einige relevante Studien nicht identifiziert wurden, wenn sie nicht in den berücksichtigten Quellen veröffentlicht wurden. Da verwandte Artikel andere Begriffe verwenden könnten, wurde der gewählte Suchbegriff möglichst weit gefasst und ein Pretest durchgeführt, um möglichst viele relevante Beiträge zu identifizieren und einzubeziehen. Die ausgewählten Veröffentlichungen wurden geprüft und

anschließend als genauer zu untersuchenden Papiere aufgenommen. In der weiterführenden Forschung sollte die Datengrundlage erweitert werden, indem weitere Quellen im Bereich Software Engineering und Information Systems und weitere Fachbereiche wie beispielsweise Maschinenbau und Automobilindustrie berücksichtigt werden. Um den Stand des Software-Qualitätsmanagements noch breiter zu erheben wird deshalb in einem weiteren Schritt die Relevanz des Themas in der Praxis untersucht.

3.5 Vergleich Ergebnisse der strukturierten Mapping-Studie mit Rai u.a. (1998)

Rai u.a. (1998) gaben im Jahr 1998 einen Überblick über den Forschungsbereich Software Quality Assurance. Eine umfassende Überprüfung der Literatur wurde durchgeführt, um damalige Schwerpunkte zu identifizieren. Das Kategorisierungsschema umfasst vier Kerndimensionen: technisch, managementbezogen, organisatorisch und ökonomisch. Diese primären Dimensionen wurden aus der Literatur abgeleitet und Subdimensionen wurden induziert, um zu einem feineren Kategorisierungsschema zu führen. In ihrem damaligen Zustand hat sich die SQM-Domäne weitgehend auf Prinzipien und Theorien aus anderen Referenzgebieten gestützt. Die Integration in den eigentlichen Aufgaben- und Technologiekontext war oberflächlich. Dies gilt insbesondere für die Forschung, die sich mit Organisations- und Managementfragen beschäftigt. Die Kerndimensionen verbindende Forschung war wenig verbreitet.[199]

Die Einteilung nach den Kerndimensionen Technik, Management und Organisation wurde in dieser Arbeit auch identifiziert, jedoch unterschieden sich die Subdimensionen. Software-Qualitätscharakteristiken haben sich weiter entwickelt zu vollwertigen vielseitigen Qualitätsmodellen. Softwarearchitekturen und deren Auswertung zur Leitungs- und Sicherheitssteigerung hat sich erst nach 2000 entwickelt. Die von Rai u.a. (1998) identifizierte Relevanz der Metriken hält bis heute an. Es wurden auch in dieser Arbeit viele Metriken für die Softwareentwicklung identifiziert. Aktuelle Metriken unterstützen darüber hinaus die Vorhersage von Software-Qualität. Es zeigt sich, dass das SQM sich von einer reaktiven Disziplin zu proaktivem Management gewandelt hat. Das Testen von Software hat sich zu einer großen Subdimension entwickelt. Automatisierte Tests sind immer weiter verbreitet.

[199] Vgl. Rai u.a. (1998), S.67f.

Da Qualitätssichermanagementansätze aus der Fertigung anscheinend nicht uneingeschränkt für Software anwendbar sind, wurden neue Werkzeuge und Normen für das SQM entwickelt. Die Wirksamkeit und Anwendbarkeit von Tools und Normen muss weiter überprüft werden. Softwareintensive Systeme und mobile Apps wurden von Rai u.a. (1998) gar nicht thematisiert, da sie damals nicht bzw. kaum vorhanden waren. SIS waren damals vor allem im militärischen Bereich zu finden. Die Dimension Management befasst sich mit dem Management der Software entlang des gesamten Softwarelebenszyklus. Anders als bei Rai u.a. (1998) wurden in dieser Dimension keine Personalthemen berücksichtigt, diese wurden aufgrund ihres Schwerpunkts im Bereich verteilte Softwareentwicklung und deren Einfluss auf Mitarbeiterkommunikation und Qualität der Dimension Organisation zugeordnet. Die Relevanz des Mitarbeitereinflusses auf die SW-Qualität hat nach 2000 zugenommen. Verhalten und Motivation der Mitarbeiter und der Einfluss der Entwicklungsumgebung sind auch heute noch von Bedeutung. Agile und verteilte Entwicklung und Organisationsstruktur waren damals noch kein Thema. Rai u.a. (1998) haben ökonomische Modelle zur Kosten-Nutzen-Betrachtung von SQM identifiziert. Schwerpunkte heutiger Veröffentlichungen thematisieren den Trade-off zwischen Qualitätskosten und Release-Zyklen um Synergiepotentiale zu ermitteln.

Da in dieser Studie auf Grund der begrenzten untersuchten Quellen kaum Beiträge aus dem Zeitraum vor 1990 und damit den Anfängen von SQM identifiziert wurden, kann sie als in erster Linie Weiterführung der Studie von Rai u.a. (1998) angesehen werden. Viele Dimensionen des SQM sind heute noch ähnlich relevant wie beispielsweise Metriken und Modelle. Technologiegetrieben sind neue Dimensionen wie mobile oder cloud-basierte Applikationen hinzugekommen.

4 Stand der Praxis im SQM

Um den Stand der Praxis zu untersuchen werden zum einen ausgewählte IT-Zeitschriften für Praktiker ausgewertet. Darüber hinaus werden Konferenzen im Bereich SQM auf deren Schwerpunkte im Zeitverlauf untersucht, um jeweilige Interessen in der Praxis zu identifizieren. Um den aktuellen Bedarf in der Praxis vertieft zu erforschen, werden darüber hinaus noch Stellenanzeigen zu diesem Bereich betrachtet.

4.1 Untersuchung von IT-Zeitschriften zum Stand der Praxis im SQM

Um ein umfassendes Bild über den Stand im SQM zu geben und möglichen Bedarf des SQM in der Zukunft zu erheben, wird die Praxis untersucht. Dazu werden 36 Zeitschriften mit praktischem Schwerpunkt auf Veröffentlichungen zum Thema SQM untersucht und der zeitliche Verlauf relevanter Beiträge untersucht. Die untersuchten Zeitschriften wurden über die Datenbank WISO im Bereich Fachzeitschriften zu IT-Informationstechnologie ausgewählt. Es wurde auch hier mit den Suchbegriffen „Software-Qualität*“ und „software quality“ recherchiert. Die Ergebnisse sind in Tabelle 14 dargestellt. Es wurden mittels Stichwortsuche 378 relevante Artikel identifiziert. Nach einer Überprüfung von Titel und Abstract wurden insgesamt 101 relevante Artikel ermittelt, wovon die meisten Artikel in der Computerwoche veröffentlicht wurden. [200] Dies kann darauf zurückgeführt werden, dass diese Zeitschrift laut Wikipedia schon seit 1974 wöchentlich erscheint. Andere Zeitschriften sind jünger und werden seltener publiziert. Die c’t beispielsweise erscheint alle zwei Wochen seit 1983.

Name der Zeitschrift	Relevante Artikel in der Stichwortsuche	Relevante Artikel nach Überprüfung des Textes
Computerwoche	220	53
c't - magazin für computertechnik	18	2
Markt & Technik	34	12
Computer @ Produktion	4	0
CYbiz	1	0
IT-BUSINESS	5	3
it&t business	58	13
Der Betriebsleiter	0	0
ERP Management	1	0
Database Marketing	0	0

[200] Eine Liste der relevanten Artikel findet sich im Anhang.

Direkt Marketing	0	0
HMD - Praxis der Wirtschaftsinformatik	10	5
IM+io Fachmagazin	9	4
Password	1	0
Fabriksoftware	0	0
ProFirma	0	0
retail technology journal	0	0
wissensmanagement	1	0
Wittener Jahrbuch für ökonomische Literatur	0	0
a3 - Fachbeiträge	8	0
business impact	0	0
IT-Governance	0	0
Internet World Business	0	0
CIO.de	3	2
Dataquest	0	0
The DQWeek	0	0
DQ Channels	0	0
Japan Economic Foundation	0	0
LEAD digital	0	0
Zeitschrift für wirtschaftlichen Fabrikbetrieb	1	1
Qualität und Zuverlässigkeit	4	6

Tabelle 14: IT-Zeitschriften für die Untersuchung von SQM in der Praxis

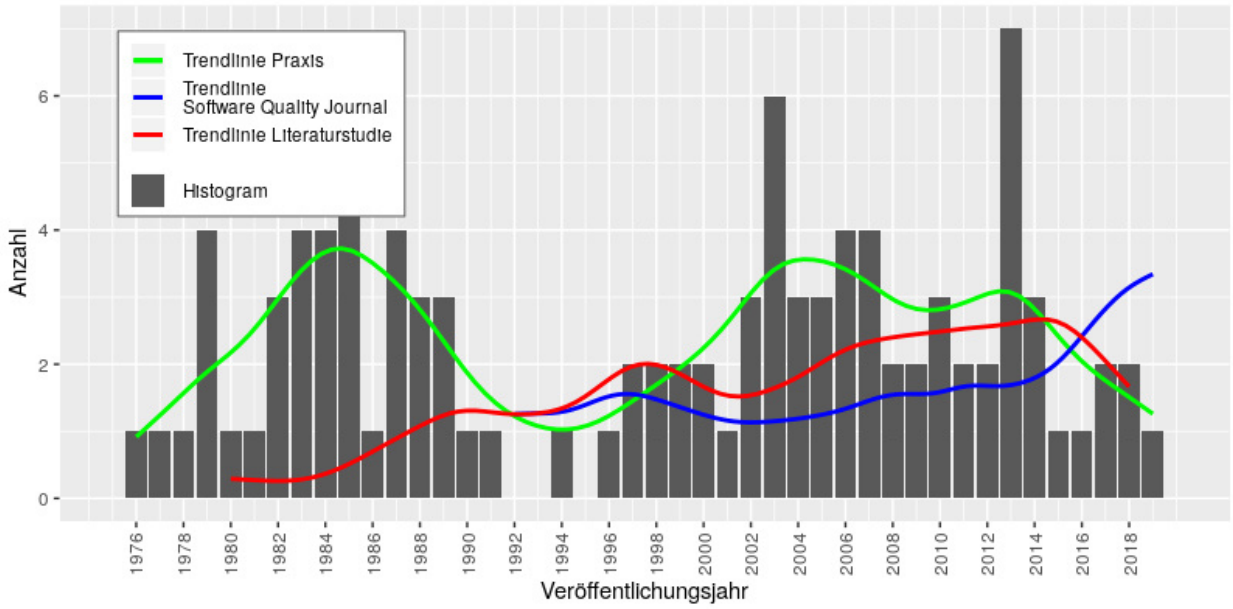

Abbildung 13: zeitlicher Verlauf der Veröffentlichungszahlen in SQM in der Praxis

Abbildung 13 und Abbildung 14 zeigen, dass die Relevanz von SQM, gemessen an der Zahl der Veröffentlichungen, in der Praxis zyklisch zu und wieder abnimmt.

Zwischen 1980 und 1990 spielten Technik und Management in der Praxis eine ähnlich wichtige Rolle. Die große Zahl managementbezogener Beiträge aus der Praxis zeigt die Betonung des Managements. Bis 2000 gab es wesentlich weniger Veröffentlichungen, nach 2000 dann wieder unerwartet viele verglichen mit den 10 Jahren davor. Die Dotcom-Blase führte offensichtlich anders als in der Forschung dazu, dass die Veröffentlichungszahlen wuchsen. Die große Zahl an managementbezogenen Beiträgen könnte eine direkte Reaktion auf die Krise gewesen sein. Seitdem ist die Zahl der Veröffentlichungen wieder rückläufig. Managementthemen und technische Beiträge bilden den Schwerpunkt der praktischen Veröffentlichungen. Beiträge zu organisatorischen Themen sind und waren konstant niedrig. Betrachtungen von SQM-Kosten und Nutzen spielten entgegen der Erwartungen kaum eine Rolle in der Praxis. Lediglich in den Jahren zwischen 2005 und 2010 gab es eine kleine Häufung bezüglich Beiträgen zum Thema Kosten und Nutzen. Dies könnte eine Reaktion auf den wirtschaftlichen Aufschwung nach Internetblase sein. Nachdem das Geschäft wieder stabil lief wurden Ansätze auf ihre Wirtschaftlichkeit überprüft. Die geringe Zahl der Veröffentlichungen der letzten Jahre kann auf die Fokussierung der Praxis auf Digitalisierung und Innnovation zurückgeführt werden. Bei der Einführung neuer innovativer Technologien liegt der Fokus auf dem innovativen Charakter der Produkte und weniger auf der Qualität. Es bleibt abzuwarten, ob es in den nächsten Jahren erneut zu einem Anstieg der Relevanz von SQM in der Praxis kommt. Das allgemeine Interesse an SQM scheint in den letzten Jahren zurück gegangen zu sein, da offensichtlich auch das Interesse in softwarespezifischen aber nicht SQM-spezifischen Journals und Konferenz und die damit verbundenen Veröffentlichungszahlen zurückgegangen sind.

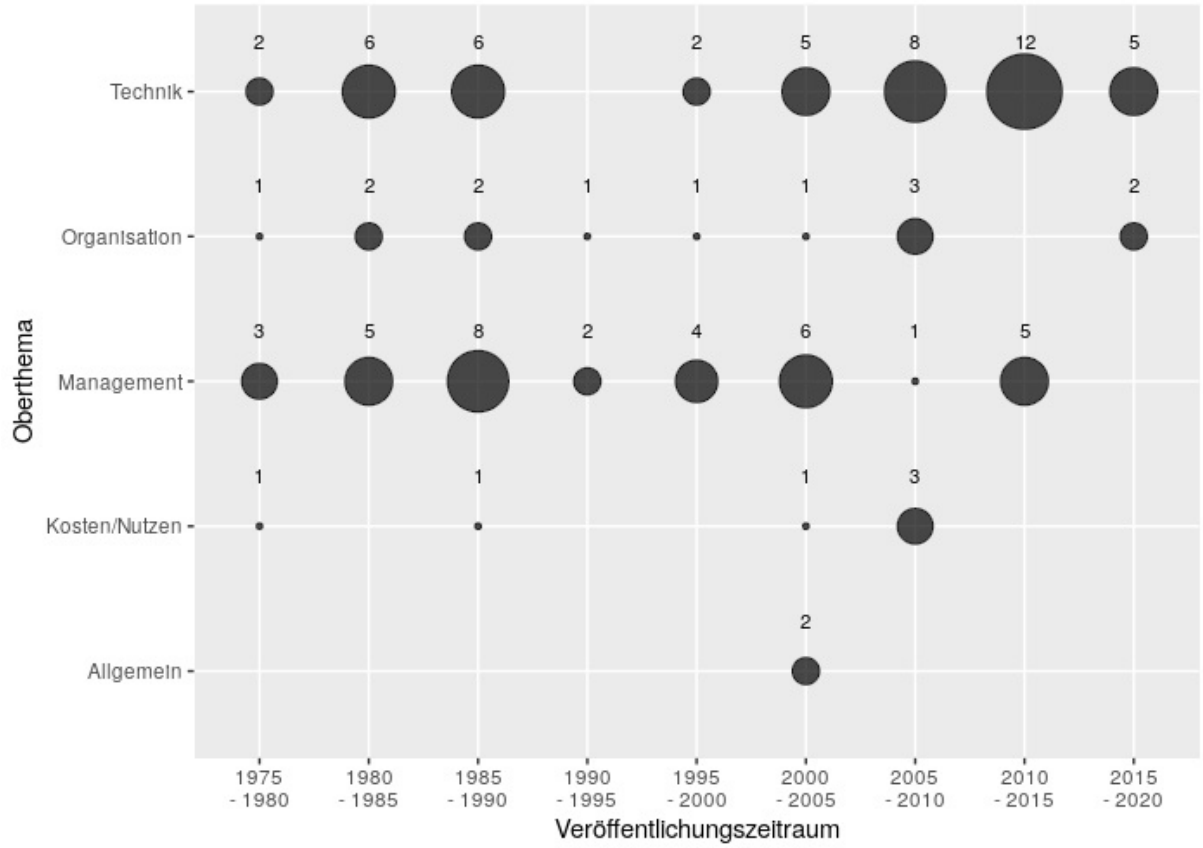

Abbildung 14: Ergebnisse der Praxis-Literaturstudie

Im weiteren Verlauf dieses Kapitels wird der zeitliche Verlauf der SQM-Dimensionen, der in Abbildung 14 dargestellt ist, detaillierter beschrieben. Der erste Beitrag zum Thema SQM findet sich 1976 in der „Computerwoche“. Er thematisiert das Fehlen von allgemein anerkannten Standards für die Beurteilung der Qualität von Software, Prüfzeichen für Software werden hinterfragt, Forschungsarbeit wird gefordert.[201] Ähnliche Beiträge finden sich vor 1980. Es wird für eine Software-Qualitätskontrolle zu verschiedenen Zeitpunkten des Software-Entwicklungsprozesses plädiert. Software-Design wird immer wichtiger.[202] Hochqualitative wirtschaftliche Programmierung wird gegenwärtig in der Praxis mit den damals vorherrschenden Vorgehensweisen als kaum erreichbar angesehen.[203] Der Aufwand für Software-Qualitätskontrolle ist groß, wird aber gerechtfertigt durch die gravierenden Folgen mangelnder Qualitätssicherung. Seit den **1970er Jahren** sind zwar vielversprechende Ansätze in der Praxis vorhanden, jedoch mangelt es an der Umsetzung. Das Thema gewinnt an Bedeutung und das Interesse an Konferenzen und Tagungen zum Thema steigt.

[201] Vgl. Jekat (1976).
[202] Vgl. Sneed (1977).
[203] Vgl. Grochla (1978).

Unternehmen gehen untereinander und mit der Forschung Kooperationen ein, um die SW-Qualität zu erhöhen.[204]

War im Jahre **1980** noch kaum die Rede von SW-Qualitätssicherung als eigenes Gebiet, erscheint dieser Begriff in den darauffolgenden Jahren immer häufiger. Software-Produktion sollte methodisch erfolgen, umstritten ist die Eignung und Wirkung der Methoden. Sie sind bisher nur für einzelne Aspekte und Phasen, nicht aber aufeinander abgestimmt verfügbar.[205] Auf Grund von fehlendem ingenieurmäßigem Vorgehen bei der Systementwicklung, mangelnder Tradition und Akzeptanz in der noch jungen Disziplin der Software-Technologie, sowie fehlender geeigneter Methoden kommt es in den 80er Jahren zu einer sog. Software-Lücke.[206] Anwender vermissen eine unabhängige Einrichtung, die mit einem "TÜV-Gütesiegel" Software-Qualität auszeichnet.[207] Testwerkzeuge verbreiten sich. Relativ schnell hat sich die Erkenntnis durchgesetzt, dass SW-Qualität nicht nur eine Sache der Codierung ist. Großen Einfluss auf die SW-Qualität haben bereits die ersten Phasen des Entwicklungsprozesses wie Problembeschreibung, Definition der Benutzeranforderungen und Grobentwurf. Um den gesamten SW-Entwicklungsprozess zu unterstützen sind mehrere Werkzeuge erforderlich. Dabei ist die Kompatibilität der einzelnen Tools miteinander von großer Bedeutung.[208] Auch die organisatorische Verankerung der Softwarequalitätssicherung im Unternehmen gewinnt an Relevanz. Die Praxis berücksichtigt neben dem modernen Software-Engineering den Faktor Mensch. Die Frage der Software-Qualitätssicherung ist längst nicht mehr nur Angelegenheit der Softwareentwickler sondern auch des Managements. Die Einsicht in die Notwendigkeit der QS-Maßnahmen ist die Basis für den SQM-Erfolg.[209] Der Umbruch innerhalb des SQM in den 1980er Jahren zeigt sich vor allem in den widersprüchlichen Beiträgen, die zum einen mehr Struktur und Werkzeuge für das SQM fordern und zum anderen betonen, dass preiswertes und

[204] Vgl. Computerwoche (1979a), Computerwoche (1979b), Klaus (1979).
[205] Vgl. Hausen u.a. (1981).
[206] Vgl. Schneiter (1982)
[207] Vgl. Computerwoche (1983)
[208] Vgl. Ahlers, Emmerich (1984).
[209] Vgl. Wintersteiger (1984).

qualitatives SQM durchaus mit schon vorhanden Werkzeugen möglich ist.[210] Zweifel an der Durchführbarkeit einer einheitlichen Prüfung für ein Prüfsiegel bestehen.[211]

Die Wirkung der DIN-(Vor-)Norm 66285 war bereits an vielen Stellen spürbar, auch wenn das Gütezeichen für Software damals noch umstritten war. Auch die Veröffentlichungen von IEEE und anderen Institutionen wurden als wertvolle Beiträge angesehen, die Arbeit der Forschung wirkt in den späten 80er Jahren.[212] Zwar waren Werkzeuge und Methoden inzwischen vorhanden, dennoch scheiterten Unternehmen häufig an der ganzheitlichen Umsetzung von SQM. Die Mehrzahl der deutschen Unternehmen hatte keine für die Software-Qualitätssicherung zuständige Stelle, setzte keine QS-Werkzeuge, rudimentären Standards ein.[213] 1989 wird ein Gütesiegel für Qualitätsgeprüfte Software eingeführt: die DIN-Norm V.66285, deren Einhaltung der TÜV im Auftrag der Gütegemeinschaft Software e.V. überwacht. [214]

Qualität ist eine wachsende Forderung des Softwaremarktes in den **1990er Jahren**. Grundlage des Qualitätsmanagements sind dabei die DIN-ISO-9000-Normen bzw. deren Vorgänger. Die Schwierigkeit der Qualitätssicherung war damals das Umsetzen im Unternehmen. Weitergehende Paradigmen wie Kundenorientierung und Innovation mussten im Management verwurzelt und an Mitarbeiter weitergegeben werden.[215] Mitte der 90er Jahre wuchs der Markt der Testwerkzeuge. Testen wurde immer mehr als strategische Aufgabe der Software-Entwicklung wahrgenommen. Je mehr sich Testprozesse etablierten, desto stärker unterstützten Test- und "Cast"-Tools (Cast = Computer Aided Software Testing) die einzelnen Abläufe.[216] SQM wurde meistens mit einer Zertifizierung nach ISO 9001 verbunden. Für die Gestaltung der Prozesse etablierte sich in der Softwarebranche "Bootstrap", gefördert durch das Esprit-Programm der EU. Dieses Assessment-Verfahren integriert wichtige ISO-Standards für die spezifischen Abläufe der Software-Entwicklung. Es bewertet die Prozesse, die ISO 9001 nur beschreibt und schafft damit die Grundlage für gezielte Verbesserungen.[217] Vorreiter im SQM waren die großen Dienstleister aus dem Finanz-

[210] Vgl. Haase (1985); Merbeth, Abbenhardt (1984); Wintersteiger (1984);Computerwoche (1985b).
[211] Vgl. Computerwoche (1985a).
[212] Vgl. Wintersteiger (1987).
[213] Vgl. Müller-Ettrich (1988).
[214] Vgl. Computerwoche (1989)
[215] Vgl. Burghartz (1994).
[216] Vgl. Schmitz (1997).
[217] Vgl. Seidel (1997).

und Telekommunikationsbereich. Ihr Geschäft wurde zu 80 Prozent von IT-Systemen gesteuert. Sowohl mit CMM als auch mit Bootstrap wurde über Assessments eine Bestandsaufnahme der Softwareprozesse oder der Organisation im Unternehmen erstellt. CMM operiert dabei mit einer Kennzahl für das gesamte Unternehmen, während Bootstrap die Reife von insgesamt 36 einzelnen Softwareprozessen bewertet.[218] Mit wachsenden Anforderungen an die SW-Qualität wandelt sich auch das Berufsbild des SW-Qualitätssicherers vom Tester zum Vermittler zwischen IT-Abteilung und Fachbereichen, wobei es kaum feste Ausbildungsgänge oder Lehrveranstaltungen an der Universität gab.[219]

In den **2000ern** wurden trotz umfangreicher Literatur stereotype Lösungsansätze wie V-Modell und Capability Maturity Model (CMM) als zu theoretisch angesehen, was Praktiker abschreckte. Viele Standardvorgehensweisen waren zu komplex und trotz aller Variabilität zu unflexibel. Pauschalisierte oder starre Umsetzung mindern Flexibilität und Innovationskraft. [220] In der Praxis hat sich deshalb eine Kombination verschiedener Modelle und Konzepte bewährt, die in ein Rahmenwerk eingebunden je nach Art und Umfang der Aufgaben und Projekte zum Einsatz kommen. Es wurde aufgezeigt, wie derartige Gesamtmodelle im prozessorientierten Qualitätsmanagement als Instrument für eine effektive Kooperation zwischen Mitarbeitern unterschiedlicher Standorte und Zusammenarbeit mit Partnern genutzt werden können.[221] Automatisierungs-Tools wie automatisierte Tests zur IT-Optimierung waren auf Grund häufiger drastischer Budgetkürzungen sehr gefragt. Vor allem die Marktsegmente Monitoring und Testing legten kräftig zu.[222] Sukzessive vollzog sich ein Paradigmenwechsel in der IT: Software entwickelte sich zu einem autonomen Produkt, das eigenständig verkauft werden konnte. Auf die Frage, wie sich Qualität in IT-Projekten etablieren und nachhaltig verankern lässt, haben sich in den Jahren zwei unterschiedliche Antworten durchgesetzt: einerseits die Einführung eines ingenieurmäßigen Qualitäts-Managements (QM) und anderseits die Umstellung auf agile Prozesse.[223] Das Voranschreiten der Software- Entwicklung hin zur industriellen Produktionsweise macht IT-Outsourcing immer attraktiver. Unternehmen müssen in

[218] Vgl. Quack (1999).
[219] Vgl. Stürze (1998).
[220] Vgl. Mücke (2002).
[221] Vgl. Müller (2003).
[222] Vgl. it&t-business (2005).
[223] Vgl.Computerwoche (2005a);Computerwoche (2005b).

arbeitsteiligen Netzwerken agieren können.[224] **Nach 2005** setzt sich embedded Software, als Teil von SIS, beispielsweise in der Automobilindustrie immer weiter durch. „Application Intelligence“ als neue Disziplin erhöht die Produktzuverlässigkeit. Gleichzeitig verschlankt sie die Produktion der IT-Komponenten. Application Intelligence sammelt, speichert und analysiert fortlaufend Informationen zum Software-Code von Anwendungen. Dieser Ansatz der begleitenden technischen Qualitätssicherung erhielt einen zusätzlichen Schub, da entsprechende Werkzeuge verfügbar waren, die diese Aufgaben größtenteils automatisiert erledigen und auch mit der maschinennahen Programmierung der Embedded-Welt zurechtkommen. Die Komplexität und Vernetzung der Systeme erfordert einen professionellen, operativen Testprozess, der alle Phasen der Software-Entwicklung begleitet und so jederzeit die Validierung des Gesamtprojektes sicherstellt.[225] Das SQM wurde bislang in der Praxis als notwendiges Übel eingestuft. Unternehmen erkennen **2008** allmählich die mit durchgängigen Testprozessen mögliche Wertschöpfung. SQM entwickelt sich zum industriellen Standard, seine zunehmende Bedeutung und Systematisierung wird durch die fortschreitende Standardisierung und Industrialisierung der Softwareproduktion getrieben. Zum anderen schaffen Investitionen in professionelles Softwaretesten einen gut messbaren Mehrwert.[226] Softwaretesten hat sich als eigenständiger Karriereweg etabliert. Anbieter von Software-Qualitätsmanagement und -Testen bündeln Expertise und Tools zu Problemlösungspaketen.[227] Unternehmen mit professionellen Software-Testern klagen seltener über mangelnde Software-Qualität und resultierenden finanziellen Aufwand.[228]

Eine Studie von **2011** bestätigt, dass Softwaretests professioneller werden: 40 Prozent der Unternehmen lassen ihren Testprozess auditieren: Das Ausbildungsniveau der Tester ist durch weltweit etablierte Zertifizierung sehr hoch.[229] Die Chance der wirtschaftlichen Turbulenzen der frühen 2000er nutzten einige Unternehmen, um die Organisation in den Bereichen Service und Qualität neu auszurichten und zu optimieren. Dies ermöglichte Kostenoptimierung und Wachstumschancen durch IT-induzierte Wettbewerbsvorteile, was bislang wenig beachtet und diskutiert wurde.

[224] Vgl. Bartram (2006).
[225] Vgl.Markt & Technik (2007).
[226] Vgl. Kaufmann (2009).
[227] Vgl.it&t-business (2009).
[228] Vgl. Kaufmann (2009).
[229] Vgl. Anecon (2011).

Beispiel ist die wachsende Verbreitung von Infotainmentsystemen im Fahrzeug, bei denen eine große Menge an Codezeilen anfallen. Hier sind herkömmliche Software-Qualitätssicherungsmethoden nur schwer anzuwenden. Neue Technologien und Systeme kommen in immer kürzeren Abständen auf den Markt. Oft ist es eine Gratwanderung zwischen Zeit, Ressourcenplanung, Kostenfaktoren und Qualität.[230] Software-Tests und Qualitätssicherung erfolgen zunehmend über Dienstleistungen. Viele Unternehmen setzen auf mobile Applikationen, die nur dann erfolgreich sind, wenn sie die vorgesehenen Funktionalitäten bieten, stabil laufen und kein Sicherheitsrisiko darstellen.[231] Der aktuelle Hype um mobile Apps führt dazu, dass sich vermehrt auch Studenten oder Berufseinsteiger privat mit der Entwicklung von mobilen Apps beschäftigen und mit diesem "Wissen“ auf Jobsuche gehen.[232] Als Weiterentwicklung werden Testservices 2013 auch als Cloud-Service angeboten.[233]

Das Internet der Dinge und Industrie 4.0 intensivieren die Kommunikation zwischen Geräten. Große Datenmengen werden erzeugt und verarbeitet. Diese neue Form der Komplexität gepaart mit hoher Agilität der Systeme erfordert neue Ansätze in der Qualitätssicherung. Modellbasiertes Testen hat nach Jahren der Vorarbeit nun einen relevanten Stellenwert in der Praxis. Der modellzentrierte Test ist eine Best-Practice Methode des modellbasierten Tests für die Industrie und bietet eine umfassende methodische Basis. Darauf aufbauend lassen sich die Anforderungen und Lösungsansätze aus der individuellen Projektsituation ableiten, der Werkzeugeinsatz kann geplant und die konkrete Umsetzung gesteuert werden. Er bringt viele Vorteile mit sich, die sich in hohen Synergien begründen.[234]

Trotz immer kürzerer Entwicklungszyklen **in den letzten Jahren** steigen die Anforderungen an SW-Qualität und funktionale Sicherheit eingebetteter Systeme. Bei embedded-Geräten muss die Software hinreichend getestet werden, bevor das Produkt marktreif ist. Die die von vielen Sicherheitsnormen geforderte Überwachung des Testfortschritts ist hier jedoch nicht ganz trivial.[235] Dies kann mit Hilfe koppelbarer Test-Tools gemeistert werden. Tool-Hersteller, die auf offene Schnittstellen zur

[230] Vgl. Veselko (2012).
[231] Vgl. Heinen (2013).
[232] Vgl. Feßl (2013).
[233] Vgl.it&t-business (2012); it&t-business (2013).
[234] Vgl. Geck (2015b)
[235] Vgl. Markt & Technik (2018).

effizienten Tool-Kopplung setzen, sind erfolgreicher, da das Zusammenspiel verschiedener Test-Werkzeuge einen durchgängigen und lückenlosen Workflow für Software- und Systemtests ermöglicht. Eine sinnvolle Kombination aus den Fähigkeiten eines menschlichen Entwicklers sowie der automatisierten Analyse-Tools stellt Sicherheit und Qualität für Embedded-Software sicher.[236] Klassische Verfahren zur Qualitätssicherung von Softwaresystemen wurden in einer Zeit entwickelt, in der phasengetriebene Vorgehensmodelle Standard waren. Diese werden jedoch zunehmend von agilen Prozessmodellen wie Scrum abgelöst. Wesentliche Gründe hierfür sind unter anderem eine flexiblere Reaktionsmöglichkeit auf sich ändernde Anforderungen oder Rahmenbedingungen, schnelleres Kundenfeedback und ein damit verbundenes geringeres Risiko des Scheiterns im Projekt.[237]

4.2 Untersuchung zum Stand der Praxis anhand von SQM-Konferenzen

Da auf Konferenzen aktuelle Themen diskutiert werden, eigenen sie sich gut um den Stand im SQM zu zeigen. Die untersuchten elf Konferenzen sind sowohl für Praktiker als auch für Wissenschaftler. Manche der Konferenzen wie beispielsweise die Software Quality Days und die REFSQ (Int. Working conference on Requirements Engineering: Foundation for Software Quality) haben einen separaten wissenschaftlichen Konferenzteil. Viele Konferenzen behandeln nur das Testen von Software. Es kann deshalb angenommen werden, dass Testing eine sehr wichtige SQM-Dimension ist. Es werden nur Konferenzen, die SQM als Ganzes behandeln, betrachtet, da dies der Schwerpunkt dieser Arbeit ist. Sie berücksichtigen gleichzeitig auch Software-Tests, womit diese Dimension nicht vernachlässigt wird. Wie Tabelle 15 zu entnehmen werden Konferenzen aus der ganzen Welt untersucht. Alle Konferenzen finden grundsätzlich in einem jährlichen Turnus statt. Die älteste Konferenz Pacific NW Software Quality Conference existiert seit 1983. Gefolgt vom Software-QS-Tag, der seit 1992 stattfindet.

Name der Konferenz	**gegründet**	**Land/Region**
expo:QA	2005	Spanien
Pacific NW Software Quality Conference	1983	Nordamerika

[236] Vgl. Markt & Technik (2016).
[237] Vgl. Söhnlein (2018).

QRS (IEEE International Conference in Software Quality, Reliabilty and Security)	2015 davor seit 2006 die SERE/SSIRI seit 2000 die QSIC/APAQS	Weltweit
QUEST (North America Quality Engineered Software and Testing Conferences)	2008	Nordamerika
Quest for Quality	2016	Europa
REFSQ (Int. Working conference on Requirements Engineering: Foundation for Software Quality)	1994	Europa
SOFTECAsia	2008	Asien (Malaysia)
Software Quality Assurance Days	1993	Russland/Osteuropa
Software Quality Days	2009	Österreich
Software-QS-Tag	1992	Deutschland
SQA Days EU	2019	Europa

Tabelle 15: untersuchte SQM-Konferenzen

Im Folgenden werden soweit möglich die Themenschwerpunkte der Konferenzen im Zeitverlauf untersucht. Konferenzen, die keine zeitliche Untersuchung auf einer Metaebene zulassen, können dennoch in der Betrachtung des aktuellen Stands berücksichtigt werden, da häufig die Themen der letzten Jahre ausführlich dargestellt werden.

Nur die Pacific NW Software Quality Conference fand schon in den **80er Jahren** statt. Themen betrafen eher Grundlagen wie das Design und die Messung von Softwarequalität, die Umsetzung und Verankerung in Organisation und Management von Unternehmen. In den **90er Jahren** gab es mehrere Konferenzen mit dem Schwerpunkt eher auf wirtschaftlichen Betrachtungen des SQM und den Architekturen von qualitativer Software. Die Wirtschaftlichkeit und Toolunterstützung für das Testen sowie Metriken und Design der Softwaretests war ein großes Thema. **Zwischen 2000 und 2010** wurden neben dem Testen vermehrt organisatorische Aspekte des SQM betrachtet. Der Einfluss der Mitarbeiter auf die Software Qualität wurde thematisiert.

Das Requirements Engineering und verbesserte Softwarequalität durch die systematische Anforderungserhebung waren im Zentrum des Interesses. Prozessverbesserung durch Software Process Improvement wurden thematisiert. Weitere Qualitätsverbesserungen durch die Zusammenarbeit mit anderen Unternehmen und die Gestaltung dieser Kooperationen waren von Bedeutung. Das modellbasierte Testen bzw. die modellbasierte Softwareentwicklung wurden erstmals auf dem Software-QS-Tag und der REFSQ diskutiert.

Nach 2010 spielten vor allem agile und kontinuierliche Ansätze eine große Rolle. Testen und Softwarequalität in verteilten und mobilen Umgebungen gerieten erstmals in den Fokus. Softwarequalität in der Cloud spielt aufgrund der zunehmenden Verbreitung der Technologie eine wachsende Rolle. Bestehende Testansätze, -methoden und –automatisierung wurden vertieft diskutiert und verbessert. Die Wirtschaftlichkeit von Tests wurde zunehmend wichtig. In diesem Zug wurden Tests als Service oder im Rahmen einer Plattformökonomie betrachtet. Als Weiterentwicklung der vergangenen Jahre wurde die Prozessautomatisierung in der Softwareentwicklung vertiefend diskutiert. Die digitale Transformation und ihre Auswirkungen auf das SQM wurden erstmals thematisiert. Unter dem Begriff „Quality of Things“ werden Komplexität und Herausforderungen durch die Verbreitung von neuartigen Technologien wie Internet der Dinge und Industrie 4.0 in der Softwareentwicklung diskutiert. DevOps und resultierende Veränderungen im SQM spielen aktuell eine große Rolle. Passend zum Titel dieser Arbeit war das Thema der Software Quality Days im Jahr 2018 „Software Quality 4.0: Methods and Tools for better Software and Systems“. Der Ansatz, Softwarequalität von Beginn an für das Produkt zu planen und zu implementieren, erhält neue Aufmerksamkeit. Schwerpunkt im **Jahr 2019** und der Zukunft ist deshalb einerseits die Anwendung von Ansätzen der Künstlichen Intelligenz im Bereich Testen und Softwarequalität sowie andererseits Softwarequalität des Bereichs Künstliche Intelligenz. Testing im Bereich neuer Technologien und Ansätze im Bereich Neuronale Netze, Machine Learning, Deep Learning bzw. Künstliche Intelligenz, Blockchains und Smart Contracts, Edge Computing, autonome Fahrzeuge, virtuelle Assitenten, Augmented und Virtual Reality sowie Conversational User Interfaces wie Sprachassistenten werden in Zukunft noch eine große Rolle spielen.

Die Erkenntnis aktueller Konferenzen ist, dass Softwarequalität von SIS ein entscheidender Erfolgsfaktor der Zukunft und in der Praxis angekommen ist (vgl. 4.1). Im Vergleich mit den Themenschwerpunkten in Praxiszeitschriften fällt auf, dass Konferenzen bis 2010 eher auf Entwicklungen in der Praxis reagiert und diese aufgegriffen haben. Dies zeigt sich daran, dass Themen wie Zertifizierung, Tools, Mitarbeiterfokus oder Testen von Software häufig zuerst in den Zeitschriftbeiträgen zu finden waren und erst einige Jahre später auf den Konferenzen und Tagungen vertieft wurden. Inzwischen sind die Konferenzen stärker auf neue und zukünftige Technologien fokussiert. Es scheint, dass die zunehmende Digitalisierung in Unternehmen dazu geführt hat, dass das Interesse an SQM für neue Technologien wächst, jedoch die Umsetzung eines „Software-Qualitätsmanagement 4.0“ im Unternehmen nicht stattfindet, was sich in den fehlenden Praxisberichten in den Zeitschriften zeigt.

4.3 Analyse des aktuellen Praxisbedarfs im SQM anhand von Stellenanzeigen

Um den aktuellen Bedarf der Unternehmen im Bereich SQM vertieft zu erforschen werden Stellenanzeigen zum SQM untersucht. Dazu wurden 7. bis 29. Oktober 2019 sämtliche Stellen, die mit dem Suchbegriff „Softwarequalität“ bzw. „software quality“ auf der Jobbörse Stepstone gefunden worden, ausgewertet. Manuelle Auswertungen weiterer Jobbörsen zeigten, dass es eine sehr große Überschneidung der Stellenanzeigen mit Stepstone gab, weshalb nur Stepstone-Stellen berücksichtigt wurden. Um wie auch bei den Konferenzen alle Aspekte des SQM einzubeziehen, wurden nur Stellenanzeigen in die Auswertung miteinbezogen, die als Softwarequalitätsmanager (software quality/assurance manager/engineer) oder als Mitarbeiter Softwarequalitätssicherung ausgeschrieben waren. Stellenausschreibungen, die ausschließlich Testmanager suchten, wurden nicht berücksichtigt. Auch hier zeigt sich wie bei den Konferenzen die besondere Wichtigkeit des Testens. Es gilt zu beachten, dass Stellen, die nicht explizit für SQM ausgeschrieben waren wie beispielsweise Softwareentwickler, auch Anforderungen bezüglich Softwarequalitätssicherung haben. Diese wurden aufgrund des Umfangs nicht in dieser Arbeit berücksichtigt. Es wurden 35 relevante Stellen deutschlandweit

identifiziert.[238] Die Stellen wurden auf die jeweiligen Aufgaben, die zu erfüllenden Anforderungen, die Branche und den geforderten Abschluss hin untersucht. Als weiterer Schritt könnten Stellenanzeigen über mehrere Jahre untersucht werden, um das Bild zu verfeinern. Dabei können auch saisonale Schwankungen auf dem Stellenmarkt berücksichtigt werden. Darüber hinaus würde es sich anbieten, SQM-Anforderungen im Vergleich mit anderen Disziplinen und damit verbundenen Anforderungen zu vergleichen und das Gewicht von SQM in der Praxis vertieft zu beleuchten.

Im Bereich der **Aufgaben** liegt der Schwerpunkt stark auf dem Testen von Software. Dies umfasst neben der Testkonzeption, -spezifikation und Testplänen vor allem die Automatisierung von Testfällen mittels Tools und Frameworks. Softwarequalitätsmanager müssen im Anschluss Testergebnisse bewerten und dokumentieren und diesen Qualitätsstatus kontinuierlich im Team und an Vorgesetzte kommunizieren. Für Führungspositionen wie Leiter Softwarequalität besteht auch Personalverantwortung. Für Qualitätsmanager ist die Zusammenarbeit innerhalb der Abteilung wichtig. Schulungen im Bereich SQM müssen selten durchgeführt werden. Die Einführung von Qualitätsstandards und Normen wie DIN EN ISO 9001, Automotive SPICE, ISO 26262 spielt eine wichtige Rolle. Eine weitere Aufgabe ist die Anwendung von QM-Methoden wie TQM, kontinuierliche Verbesserung. Dabei spielt entgegen der Erwartung in wenigen Stellenanzeigen (22% der Stellen) die agile Softwareentwicklung eine Rolle.

Bezüglich der **Anforderungen** an Bewerber fällt auf, dass sehr gute Deutsch- und Englischkenntnisse am häufigsten erwartet werden. Grund dafür könnten internationale Kunden- und Lieferantenkontakte im Bereich der Softwareentwicklung sein. Häufig wird Berufserfahrung im Bereich Softwaretests bzw. SQM gefordert (33%). Kenntnisse über Tools wie Bugtrackingsysteme (Jira, Trac, svn, git, Bugzilla, Mantis Bug Track), Testmanagement- und -automatisierungssysteme (Jenkins, Squish, TestRail, TOSCA Testsuite, HP Unified Functional Testing, Ranorex Studio, Selenium, Katalon, Silk Test, QFTest) Programmierkenntnisse (Python, C++, Java, C#, Delphi) werden vorausgesetzt. Besonders häufig werden Kenntnisse im Umgang mit dem Framework Selenium gefordert (18% der Anzeigen). Darüber hinaus werden Wissen und Fertigkeiten in Methoden und Prozessen der Softwarequalitätssicherung erwartet, ohne näher spezifiziert zu werden. Erfahrungen in der Umsetzung von

[238] Eine ausführliche Liste der Stellenanzeigen findet sich im Anhang.

Normen und Standards wie Automotive SPICE werden gern gesehen. Eine weitere häufige Anforderung ist ausgeprägtes Qualitätsbewusstsein der Bewerber. Von Vorteil ist eine ISTQB-Zertifizierung der Bewerber im Bereich Testing. Auf der persönlichen Ebene werden Bewerbern strukturierte und qualitätsorientierte Arbeitsweise, Kommunikationskompetenz und Teamfähigkeit abverlangt.

Die ausgeschriebenen Stellen sind vor allem in der Automobil- und Maschinenbau**branche**, aber auch in der Softwareentwicklung. Wenige Unternehmen haben dabei eigenständige SW-Qualitätsabteilungen, häufig wird der Qualitätsingenieur als Teil des Softwareentwicklungsteams eingesetzt. 63% der Unternehmen suchen Bewerber mit einem abgeschlossenen Informatik**studium**. Für 41% der Unternehmen reicht eine **Ausbildung** in der Softwareentwicklung mit entsprechender Berufserfahrung im Bereich SQM. Auch Absolventen verwandter Fachrichtungen wie Elektrotechnik, Wirtschaftsinformatik oder Mathematik können sich stellenweise auf die ausgeschriebenen Stellen bewerben.

Die Stellenanzeigen verdeutlichen den steigenden Bedarf an Qualitätssicherung, da Software immer komplexer wird und mehr Fremdbibliotheken integriert werden. Trotz immer kürzerer Entwicklungszyklen steigen die Anforderungen an SW-Qualität und funktionale Sicherheit der Systeme Diese Entwicklung zeigt sich in allen untersuchten Praxisaspekten. Als Resultat werden häufiger agile und kontinuierliche Prozessmodelle wie Scrum eingeführt. Agilität und Testing werden sowohl in den Praxiszeitschriften wie auch auf den Konferenzen und in den Stellenanzeigen thematisiert. Die Anwendung neuer Technologien in Unternehmen scheint langsam voranzuschreiten wie in der Analyse von Zeitungsbeiträgen und Konferenzen identifiziert und in den Anforderungen der untersuchten Stellenausschreibungen gefunden. Ansätze wie Cloudcomputing, Internet der Dinge oder Künstliche Intelligenz werden in den Ausschreibungen kaum thematisiert. Wo genau sich Diskrepanzen zwischen vorhandenen Ansätzen, Wissen und dem Bedarf neuer Technologien, insbesondere software-intensiver Systeme ergibt, wird im nächsten Kapitel abschließend diskutiert.

5 Vergleich der Herausforderungen von software-intensiven Systemen mit dem Stand im SQM

Der Aufbau komplexer softwareintensiver Systeme ist immer noch eine Herausforderung. In vielen Anwendungsbereichen wird eine typische Lernkurve beobachtet: Zu Beginn des Lernzyklus wird Software nur als Nebensache betrachtet. Mit fortschreitendem Lernen wird Software zunehmen zu einem zentralen technischen Thema. Softwareingenieure beginnen individuelle Prozesse und Modelle zu entwickeln, um domänenspezifische Softwareprobleme zu lösen. Die Bewältigung der Entwicklungsherausforderungen komplexer Softwaresysteme erfordert neue Kompetenzen wie ein gutes Verständnis der Systemmodellierung und der effektiven Nutzung von Modellen. Strukturiertes Systems-Engineering benötigt technische Notationen, Systemarchitekturen, die sich sowohl mit Software als auch mit Hardware-Architekturen befassen, geeignete Implementierungssprachen und Werkzeugunterstützung.[239] Besondere Herausforderungen für die Softwarequalität und –funktionalität und damit für das SQM ergeben sich aus den Eigenschaften software-intensiver Systeme, die in Abschnitt 2.5 erläutert wurden. Sie werden im weiteren Verlauf des Kapitels vertieft und vorhandene Ansätze aus Kapitel 3 und 4 aufgezeigt, um diesen Herausforderungen zu begegnen.

5.1 Multifunktionalität

Die Multifunktionalität der Systeme begleitet von einer zunehmenden Heterogenität der Systemkomponenten ist in den letzten Jahren gewachsen. Multifunktionale Geräte haben oft viele und unabhängige Funktionen. Die geforderten Funktionalitäten werden während der Requirements Engineering-Phase erhoben. Unterschiedliche konzeptionell unabhängige Funktionalitäten können voneinander abhängig sein, wenn sie gleiche Ressourcen nutzen, wie z.B. Zugriff über die gleiche Tastatur. Daher müssen verschiedene Funktionalitäten und deren Interaktionen detailliert beschrieben werden.[240]

Aufgrund der Multifunktionalität hat jedes SIS einen spezifischen Kontext, der durch Benutzungsumgebung und Maschinenkontext und resultierender Benutzerinteraktion

[239] Vgl. Broy (2006b), S.72.
[240] Vgl. Broy (2006b), S.71.

charakterisiert wird. Die Benutzerumgebung beschreibt die spezifische Situation des Systembenutzers. Der Maschinenkontext beschreibt die Eigenschaften des eingebetteten Geräts und resultiert in diversen maßgeschneiderten Lösungen für die Mensch-Maschine-Interaktion.[241]

Herausfordernd ist dabei die Anpassung jedes Systems an seinen spezifischen Nutzungskontext mit teilweise widersprüchlichen Belangen der Zuverlässigkeit und Selbst-Adaptivität. Da SIS oft sicherheitskritische Anwendungen hosten müssen sie auch bei hoher Dynamik zuverlässig sein. Um in verschiedenen Nutzungskontexten arbeiten zu können, müssen SIS sich selbst anpassen können. Die Erzielung von Synergien aus Zuverlässigkeit und Selbst-Adaptivität bei Vorliegen von Betriebsunsicherheiten ist schwierig. Bestehende Ansätze schaffen es in der Regel, nur einen Aspekt des Problems zu lösen.[242]

Die Multifunktionalität der Systeme und die daraus resultierende Komplexität der Software betrifft alle Bereiche des SQM wie in Kapitel 2.4 aufgeführt. Schon in der Qualitätsplanung müssen die Funktionen des zukünftigen Systems antizipiert werden, um die richtigen Verfahren und Werkzeuge bereitzustellen. Der Aufwand der Qualitätssicherung nimmt mit jeder Funktion zu. Die Sicherstellung der Qualität eines SIS ist extrem aufwendig.

Um Systeme systematisch an den spezifischen Nutzungskontext anzupassen, können Maßnahmen des Software Process Improvements, wie sie in Kapitel 3.2.4.3 erläutert werden, durchgeführt werden. In dem dynamischen Umfeld der SIS können agile Methoden einen Vorteil gegenüber klassisch sequenziellen Ansätzen haben.[243]

Um die Selbst-Adaptivität der Systeme zu unterstützen können Ansätze der künstlichen Intelligenz in die Systeme und die Qualitätssicherung implementiert werden. In der Forschung wird die Nutzung der Künstlichen Intelligenz zur Softwareverbesserung schon lange untersucht.[244] Ansätze wie Application Intelligence werden in der Praxis zunehmend genutzt.[245]

---

[241] Vgl. Broy (2006b), S.72.
[242] Vgl. Gerostathopoulos u.a. (2016), S.378.
[243] Vgl. Baskerville u.a. (2002).
[244] Vgl. Khoshgoftaar u.a. (1992); Bouktif u.a. (2002).
[245] Vgl. Markt & Technik (2007)

5.2 Zeitanforderungen an die Software

Sensoren und Aktoren verbinden SIS mit der Umgebung und senden Signale zurück. Da ein software-intensives System häufig physische oder technische Prozesse steuert, muss es in Echtzeit arbeiten und bestimmte Zeitmerkmale einhalten, um auf Ereignisse zu reagieren. Die Ausführungszeit der Algorithmen muss mit den Zeitanforderungen für die Ausgabe übereinstimmen. Traditionell wird Echtzeitprogrammierung mit Low-Level-Maschinencode unter Verwendung von Maschinenzykluszeiten durchgeführt. Die rechtzeitige Reaktion ist daher abhängig von den Maschinenzyklen, was die Portierbarkeit zeitabhängiger Softwaresysteme erschwert. Infolgedessen muss die gleiche Funktionalität bei neuen Produkten immer wieder neu implementiert werden.[246]

Das softwareintensive Systemdesign in Echtzeit muss aufgrund dieser zeitlichen Einschränkungen neue Arten von Komplexität berücksichtigen. Systemspezifikationen müssen die strukturelle und verhaltensmäßige Modellierung von Komponenten und deren Wechselwirkungen unter strikter Berücksichtigung der Zeit unterstützen. Es genügt nicht, die Zeit informell oder abstrakt zu modellieren. Die softwareintensive Systementwicklung in Echtzeit erfordert die Berücksichtigung von Soft- und Hardwareteilen. Die erfolgreiche Entwicklung solch komplexer Systeme basiert auf Modellen für die hochrangige Systemsimulation, die eine Grundlage für das Design und die Entwicklung von Low-Level-Software bilden können.[247]

Das Systemdesign für Echtzeit-Systeme stellt vor allem die Entwicklung und die Qualitätsprüfung vor Herausforderungen. In der Qualitätsprüfung müssen real-time Tests für die Systeme durchgeführt werden. Auch hier hat die Forschung vielversprechende Ansätze, die für SIS erweitert wurden.[248]

5.3 Komplexität

Insbesondere in den letzten 10 Jahren ist eine deutliche Zunahme der Größe und Komplexität softwareintensiver Systeme zu beobachten. Die Hardware-Strukturen in SIS wurden leistungsfähiger. Ein System besteht nicht mehr nur aus einer einzigen Steuerung, sondern aus mehreren Controllern. Beispielsweise verfügen Fahrzeuge über bis zu 80 Steuerungen, die durch bis zu fünf verschiedene untereinander

[246] Vgl. Broy (2006b), S.73.
[247] Vgl. Huang, Sarjoughian (2004).
[248] Vgl. Hessel u.a. (2008), Wegener u.a. (1997).

verbundene Bussysteme verbunden sind. Auf diese Weise können Entwickler sehr komplexe Architekturen erstellen.[249]

Auch die Anzahl der softwarebasierten Funktionen in Fahrzeugen hat permanent zugenommen. 2007 enthielt ein Oberklassefahrzeug etwa 270 implementierte Funktionen. Inzwischen sind es mehr als 500 solcher Funktionen. Nicht nur die Anzahl der softwarebasierten Funktionen hat sich erhöht. Auch die Softwaremenge hat zugenommen. Die Menge des Binärcodes in einem Oberklassefahrzeug betrug 2007 rund 65 Megabyte, inzwischen ist es mehr als ein Gigabyte. Die wahrnehmbaren Funktionen eines Fahrzeugs werden zunehmend durch die Integration feinkörniger eingebetteter SIS realisiert. Daher nimmt die Komplexität in Fahrzeugen auf zwei Arten zu:[250]

1. Komplexität innerhalb des Systems: Da die Kundenanforderungen an bekannte Funktionalitäten zunehmend individuellere softwareintensive Systeme wie z.B. Fahrassistenzsystem erfordern, sind vielfältige und anspruchsvollere Funktionen erforderlich. Die einzelnen Systeme werden komplexer.
2. Systemübergreifende Komplexität: Immer mehr Fahrzeugfunktionen werden durch die Integration von Systemen in höhere Verbundstrukturen realisiert, die gemeinsam umfangreiche Dienstleistungen erbringen ("System of Systems"). Daher nimmt die Anzahl und Vielfalt der systemübergreifenden Beziehungen deutlich zu.

Die Zunahme der Hardwarekomplexität findet keine Berücksichtigung im SQM. Bisher werden Hardwarefragen kaum diskutiert. Ein gemeinsamer Ansatz für Hard- und Software findet sich in der FMECA (Failure Mode, Effects and Criticality Analysis).[251]Um der Zunahme der Softwarekomplexität innerhalb des Systems und systemübergreifend zu begegnen, wurden beispielsweise ein architekturzentrierter Ansatz entwickelt, der die dynamische Rekonfiguration der Architektur unterstützt.[252]

[249] Vgl. Broy (2006b), S.72.
[250] Vgl. Braun u.a. (2014), S.23.
[251] Vgl. Liggesmeyer (2009), S.75.
[252] Vgl. Cavalcante (2015)

5.4 Sicherheit

Sicherheit als zentrales Qualitätsmerkmal von Software ist eine wesentliche Herausforderung für die SIS von morgen. Laut einer Umfrage von Gartner im Frühjahr 2018 wurden fast 20 Prozent der Unternehmen in den vergangenen drei Jahren bereits Opfer eines IoT-basierten Angriffs. Einer der Gründe dafür ist laut Gartner, dass sich die Security Standards langsam entwickeln. Es mangele noch an Security by Design.[253] Systeme werden komplexer und bestehen aus zunehmend heterogenen Komponenten, Systemumgebungen werden unvorhersehbarer. Ein Ausfall oder eine Fehlfunktion kann Menschenleben gefährden oder zu erheblichen Sachschäden führen. Sicherheitsprobleme sind häufig in der Anforderungs- und Systemtechnik und nicht in der Detailkonstruktion der Software begründet. In der Folge muss die Sicherheit designorientiert in das System integriert werden.

Die Art der Sicherheit bzw. relevante Sicherheitslücken innerhalb eines komplexen SIS werden häufig missverstanden, bekannte Systemsicherheitstechniken nicht angewendet. Dieses Problem erfordert Ausbildung und Schulung der Entwickler. Methoden der Qualitätsplanung, --prüfung und –lenkung , wie sie in Kapitel 2.4 dargestellt wurden, sollten von Beginn an verankert werden, um Sicherheit in das System zu bringen. Forschungsbedarf herrscht in der Bewältigung des verteilten Charakters von SIS. Sowohl die Softwareentwicklung als auch die Systemkomponenten selbst sind verteilt. Es resultieren auch verteilte Systemrisiken.[254] Neue Modelle zur Ermittlung dieser verteilten Systemrisiken werden benötigt.

Eine weitere Herausforderung ist die Zertifizierung zur Erfüllung der Sicherheitsanforderungen. Die aktuellen Normen unterscheiden typischerweise zwischen verschiedenen "Sicherheitsniveaus“ mit strengen Entwicklungspraktiken, die in heterogenen SIS unter Umständen nicht umgesetzt werden können. Ein evidenzbasierter Ansatz könnte die Herausforderung meistern [255] und zur Qualitätszertifizierung beitragen. Spezielle weltweit anerkannte Zertifizierungen für die

[253] Vgl. Kreuzer (2018), S.18f.

[254] Vgl. Heimdahl (2007), S.150.

[255] Die vorgelegten Nachweise würden in Form eines Sicherheitsfalls vorliegen. Dabei handelt es sich um dokumentierte Beweisstücke, die ein überzeugendes und valides Argument liefern, dass ein System für eine bestimmte Anwendung in einer bestimmten Umgebung ausreichend sicher ist. (Vgl. Heimdahl (2007), S.142).

Sicherheit von SIS gibt es bisher noch nicht. Vorhandene Normen und Zertifizierungen sind nur begrenzt für SIS nutzbar.

Um die Produktivität zu steigern werden modellbasierte Entwicklung und automatisierte Tools, wie z.B. Codegenerierung und automatisiertes Testen, erfolgreich eingeführt.[256] Diese Abhängigkeit von Tools und nicht von Menschen führt jedoch zu neuen und wenig verstandenen Problemen. Es bedarf neuer Qualitätssicherungsansätze für Modelle, die die Grundlage für eine werkzeugintensive modellbasierte Entwicklung bilden. Die Korrektheit der automatisierten Testwerkzeuge muss sichergestellt sein. Automatisierte und modellbasierte Tests haben sich inzwischen in Forschung und Praxis etabliert.[257] Schwerpunkt der aktuellen Forschung in diesem Bereich liegt auf der Adaption vorhandener Modelle für SIS.[258]Ein weiterer Forschungsbereich ist die Untersuchung des Substitutionseffekts von menschlichen Aktivitäten durch automatisierte Werkzeuge.[259]

5.5 Entwicklungsprozess

Nicht nur die inhärenten Merkmale von software-intensiven Systemen stellen Herausforderungen für deren Softwarequalität dar. Auch der Entwicklungsprozess der Systeme beeinflusst wesentlich die Qualität der Systeme.

Richtiges Requirements Engineering stellt eine Herausforderung für das Design von SIS dar, da sie für spezifische Anforderungen, die sich direkt an die Nutzer und Märkte richten, entwickelt werden müssen. Es bedarf einer angemessenen Anforderungserhebung, um die Teilprozesse der Lieferanten an den Kundenbedürfnissen auszurichten.[260]

Wie in Kapitel 2.3 ausgeführt, beschreibt SW-Qualität die *Summe aller relevanten Eigenschaften eines Software-Produkts zur Zufriedenstellung des Kunden.* Dies ist die Definition der Anforderungen und Grundlage für SQM. Herausforderungen bezüglich der Anforderungserhebung in SIS können sein:[261]

[256] Vgl. Kläs u.a. (2015).
[257] Vgl. Drave u.a. (2018), Kläs u.a. (2015), Geck (2015a).
[258] Vgl. Briand u.a. (2016).
[259] Vgl. Wirsing (2004), S.13.
[260] Vgl. Broy (2006b), S.75.
[261] Vgl. Konrad, Gall (2008), S.218.

- Eine große Anzahl von Kundenanforderungen stellt Herausforderungen an die Analyse, Spezifikation und Verwaltung von Anforderungen. Ein Ansatz der Forschung ist die Anwendung der ISO 16355 für QFD im Bereich von SIS.[262] Anforderungen aus verschiedenen Domänen können mit einem SysML-Anforderungsdiagramm organisiert werden.[263]
- Management der Kundenerwartungen: Bei der Entwicklung von SIS können dem Kunden erste Prototypen des Systems gezeigt werden. Die Forschung hilft bei der Identifikation möglicher Fehlerquellen der Anforderungserhebung, um Problemen vorzubeugen. [264]
- geografische Verteilung der Projektteams: Frühere Studien haben gezeigt, dass diese Verteilung oft zu Problemen wie Koordinations- und Kommunikationsherausforderungen führt und in minderer SW-Qualität resultiert.[265] Hier besteht Forschungsbedarf. Modellierungssprachen für verteilte Qualitätskontrollen werden benötigt.
- Rückverfolgbarkeit: Eine vollständige bidirektionale Rückverfolgbarkeit der Anforderungen ist erforderlich. Dies hat einen hohen zeitlichen und manuellen Aufwand. Die Anforderungsrückverfolgbarkeitsmatrix von Hayes u.a. (2005) bietet hier eine Hilfestellung.
- Anforderungsänderungen: In großen Projekten ist es unvermeidlich, dass der Kunde bspw. aufgrund von neuer Technologien und Prozesse Änderungen an einigen seiner Anforderungen fordert. Die Anpassung komplexer Systeme wie SIS an die veränderten Rahmenbedingungen bei gleichzeitiger Einhaltung der Projektmeilensteine kann eine Herausforderung sein. Agile oder Evolutionäre Qualitätssicherung und Softwareentwicklung wie in dem europäischen Forschungsprojekt SERIOUS kann Änderungen der Anforderungen besser berücksichtigen.[266]

Die Kombination mechanischer Geräte, Hard- und Software in SIS erfordert die Beteiligung verschiedener Anbieter. Da unterschiedliche Teilnehmer der Lieferkette an der Produktion eines Systems beteiligt sind, bedarf es anspruchsvoller

[262] Vgl. Schönhofen u.a. (2018)
[263] Vgl. Soares u.a. (2011).
[264] Vgl. Walia, Carver (2009).
[265] Vgl. Nguyen-Duc u.a. (2015).
[266] Vgl. van Rompaey u.a. (2009)

systemtechnischer Fähigkeiten, um diese Interaktionen zu orchestrieren und die erfolgreiche Integration der Systemteile zu gewährleisten. Concurrent Engineering führt die Entwicklung durch mehrere gleichzeitige Aktivitäten, synchronisiert sie gelegentlich zu Meilensteinen, damit das Endprodukt gut ineinandergreift. Experten unterschiedlicher Disziplinen müssen zusammenarbeiten und dabei Modelle und Theorien verwenden, die die Grundlage für die Entwicklung eines gemeinsamen Verständnisses und der gemeinsamen Kommunikation bilden können. [267] Theoretische Frameworks zum Concurrent Engineering sind vorhanden. Die Implementierung in der Praxis geschieht erst sukzessive.[268]

Ein großer Trend und Perspektive in der Entwicklung SIS ist die modellbasierte Softwareentwicklung. Dabei werden für die klassischen Entwicklungsaktivitäten Modelle entwickelt (Anforderungs-, Design-, Implementierungs- und Modellierung von Testfällen). Die Vision ist ein integrierter Modellierungsansatz, bei dem die Beziehung zwischen allen Modellen erfasst und Teile der nächsten Modelle aus den vorhergehenden Modellen generiert werden. Hier herrscht ein großer Forschungsbedarf.[269] Derzeit wenden Entwickler und Betreiber für jedes Probleme unterschiedliche Modellierungs- und Analysetools an. Jedes Werkzeug benötigt spezifische Eingabemodelle in verschiedenen Sprachen. Qualität wird meist nicht in einer domänenspezifischen Modellierungssprache dargestellt.[270] Es gibt zahlreiche vielversprechende Ansätze vor allem im Bereich Testing und vielfältige Forschungsanstrengungen.[271] Auch in der Praxis werden modellbasierte Entwicklung und Testen bereits eingesetzt.

5.6 Architektur

Die Komplexität der Soft- und Hardwarestrukturen, die Multifunktionalität der Systeme und die Existenz verteilter und konkurrierender Engineering-Prozesse haben die dominierende Rolle der Hard- und Softwarearchitekturen verstärkt. Die Architektur hilft, die Fähigkeiten des Systems sicherzustellen, indem sie den Entwurf für den Entwicklungsprozess und die Integration liefert. Nur durch die richtige Architekturbeschreibung können Entwickler sicherstellen, dass Subsysteme integriert

[267] Vgl. Broy (2006b), S.75.
[268] Vgl. Daun u.a. (2016)
[269] Vgl. Broy (2006a).
[270] Vgl. Heinrich (2018), S.336.
[271] Vgl. Drave u.a. (2018); Kläs u.a. (2015).

werden können. Derzeit fehlt die Fähigkeit, kombinierte Systeme zu beschreiben. Viele der verwendeten architektonischen Ideen sind zu spezifisch.[272]

Softwareintensive Systeme werden oft unabhängig voneinander entwickelt, betrieben, verwaltet und weiterentwickelt. Kommunikationsnetze ermöglichen die sukzessive Interaktion dieser unabhängigen Systeme und ergeben ein neuartiges komplexes Gesamtsystem. Bisher hat sich die Softwarearchitekturforschung hauptsächlich auf einzelne große Systeme konzentriert, deren Softwarearchitektur als Design-Zeit-Konfigurationen von Komponenten beschrieben werden. Die Eigenschaften einzelner großer Systeme führen zu architektonischen Lösungen (in Form von Theorien, Sprachen, Werkzeugen und Methoden), die sich nicht zu Gesamtsystemen skalieren lassen. Herausforderung sind neuartige Architekturlösungen, die sich aus der evolutionären verteilten Entwicklung ergeben. Die separate Betrachtung von Hard- und Softwarearchitekturen ist nicht ausreichend, architektonische Ideen sind zu spezifisch, beheben entweder Hardware- oder Softwareprobleme. Es braucht eine mehrstufige Architektursicht, die verschiedenen Aspekte des Systems in einem integrierten architektonischen Rahmen betrachtet.[273]

Die Forschung im Bereich Architekturen für SIS ist sehr aktiv. Es werden automatisierte Werkzeuge für die Architekturentwicklung designed.[274] Ein weiterer Ansatz dient der Erkennung von Architekturverletzungen. [275] Für Probleme anfällige Architekturelemente werden identifiziert. Dies unterstützt die Entwicklungsabteilung bei der optimalen Ressourcenallokation.[276]

5.7 Wartung

Die Wartung von SIS kann extrem teuer sein und die Kosten der ursprünglichen Entwicklung übertreffen. Empirische Daten aus Langzeitstudien haben gezeigt, dass in großen, langlebigen, softwareintensiven Systemen bis zu 80% der gesamten Lebenszykluskosten nach der ersten Implementierung anfallen können. Instandhaltungs-Metriken konzentrieren sich typischerweise auf die Produktkomplexität mit der Annahme, dass Fehler in komplexen Systemen oder Teilen

[272] Vgl. Broy (2006b).
[273] Vgl. Oquendo (2016), Cavalcante (2015).
[274] Vgl. Avgeriou u.a. (2014).
[275] Vgl. Goldstein, Segall (2015).
[276] Vgl. Mo u.a. (2018).

eines Systems wahrscheinlicher und diese schwerer zu verstehen und zu ändern sind. Eine große Herausforderung ist, dass der Instandhaltungsaufwand eines SIS nicht durch die Analyse des isolierten Systems unabhängig von den Domänen-, Entwicklungs- und Einsatzumgebung vorhergesagt werden kann.[277] Dabei besteht die Herausforderung, die Softwareentwicklung und -wartung über Ländergrenzen hinweg zu steuern. Um Anwendungen mit einem verteilten Team zu entwickeln und zu warten, versteht ein kleines kundenorientiertes Team die Anwendungsdetails und kommuniziert die Informationen an eine größere Gruppe, die in der Regel nicht mit dem kundenorientierten Team zusammenarbeitet. In vielen Fällen gibt es mehrere Ad-hoc-Dokumente und Tabellenkalkulationen, um Anwendungswissen zu erfassen. Das Wartungsteam pflegt und verbessert die Anwendung im Laufe der Jahre weiter, aktualisiert aber selten die zugehörigen Dokumente.[278] Die Wartung von SIS wurde bisher in der Forschung wenig behandelt. Hier herrscht Bedarf, um die Wartung verteilter komplexer SIS zukünftig zu vereinfachen. Vorhandene Ansätze wurden in Kapitel 3.2.4.3 aufgezeigt.

Die folgende Tabelle stellt zusammenfassend die in diesem Kapitel identifizierten Herausforderungen, Lösungsansätze und den resultierenden Forschungsbedarf dar. Grün hinterlegt sind dabei vorhandene Forschungsansätze. Orange sind nötige Erweiterungen vorhandener Ansätze und rot markiert sind wichtige Forschungslücken im Bereich Software-Qualitätsmanagement SIS.

Herausforderungen software-intensiver Systeme	**SQM-Methoden und Ansätze die den Herausforderungen begegnen**	**Zukünftiger Forschungsbedarf**
Multifunktionalität der Systeme	Betrifft alle Bereiche des SQM, da die Komplexität zunimmt	
Systemanpassung an spezifischen Nutzungskontext	Durchführung von Software Process Improvement Maßnahmen (Kapitel 3.2.4.3) Nutzung von Agilen Entwicklungsmethoden	
Selbst-Adaptivität der Systeme	Implementierung von AI in Systemen und Qualitätssicherungssystemen Bouktif u.a. (2002); Khoshgoftaar u.a. (1992)	Anpassung der Modelle für große komplexe Softwaresysteme

[277] Vgl. Brown u.a. (1995), S.243.
[278] Vgl. Sarkar, Panayappan (2008), S.2.

	Application Intelligence in Praxis genutzt	
Systemdesign für Echtzeit-Systeme mit Berücksichtigung von Soft- und Hardwareteilen	Neue Herausforderungen für die Qualitätsprüfung z.B. Testing für realtime Systeme	Entwicklung von realtime Werkzeugen für SIS
Zunahme der Hardwarekomplexität	Keine Berücksichtigung im SQM	SQM-Modelle, die auch Hardwareaspekte berücksichtigen
Zunahme der Softwarekomplexität innerhalb des Systems und systemübergreifend	Architekturzentrierter Ansatz mit Rekonfiguration Cavalcante (2015)	
Identifikation von relevanten Sicherheitsrisiken und Ableitung adäquater Präventionsmaßnahmen	Entsprechende Schulungen die im SQM verankert sind	Erarbeitung neuer Modelle zur Ermittlung (verteilter) Systemrisiken, die auch aus verteilter Systementwicklung, Systemkomponenten und resultieren
Zertifizierung zur Erfüllung der Sicherheitsanforderungen	Bisherige Normen begrenzt für SiS nutzbar	evidenzbasierter Ansatz zur Qualitätszertifizierung
Einführung modellbasierter Entwicklung und Testing	Modellbasierte und automatisierte Testverfahren sowohl in Forschung und Praxis untersucht und implementiert	Nötige Anpassungen und Erweiterungen für SIS ermitteln
Automatisierte Werkzeuge	OOMeter, Open Source Tools, Metamodell zur Toolbewertung vorhanden Guzman u.a. (2017)	Verbindung der Werkzeuge bisher noch schwierig
Große Anzahl von Kundenanforderungen	Anwendung der ISO 16355 im SIB Schönhofen u.a. (2018)	neue Verfahren zur Erfassung, Formalisierung und Validierung von Anforderungen
Management der Kundenerwartungen	Identifikation möglicher Fehlerquellen bei der Anforderungserhebung um Problemen vorzubeugen Walia, Carver (2009)	
geografische Verteilung der Projektteams	Auswirkung verteilter Entwicklung auf die SW-Qualität wurde untersucht Nguyen-Duc u.a. (2015)	Modellierungssprachen für verteilte Qualitätskontrollen

Anforderungsveränderung z.B. durch Berücksichtigung neuer Technologien bei der Anforderungsdefinition	Agile oder Evolutionäre Softwareentwicklung (SERIOUS-Projekt) und Qualitätssicherung kann Änderungen der Anforderungen besser berücksichtigen van Rompaey u.a. (2009)	
Rückverfolgbarkeit der Anforderungen	Anforderungs-rückverfolgbarkeitsmatrix von Hayes u.a. (2005)	
Concurrent Engineering meistern	Theoretische Frameworks vorhanden, Implementierung in der Praxis erst sukzessive Daun u.a. (2016)	Anpassung das SQM an concurrent Engineering
modellbasierte Softwareentwicklung	Viele neue Ansätze auch im Bereich Testing vorhanden und teilweise schon implementiert	
neuartige Architekturlösungen, um die Komplexität zu bewältigen	Automatisierte Werkzeuge zur Architekturentwicklung Avgeriou u.a. (2014) Ansatz zur Erkennung von Architekturverletzungen Goldstein, Segall (2015) Identifizierung anfälliger Architekturelemente Mo u.a. (2018)	Vereinheitlichung der Architekturbeschreibung soweit möglich
Wartung über Ländergrenzen hinweg zu steuern	Keine Ansätze zu verteilter Wartung	Erarbeitung von Ansätzen, die die Wartung verteilter Systeme vereinfachen

Tabelle 16: Vergleich Stand SQM mit Herausforderungen von SIS

6 Fazit und kritische Würdigung

In dieser Arbeit wurde der Stand des Software-Qualitätsmanagements untersucht. Erste Praxisveröffentlichungen wurden in den 70er Jahren gefunden. Die wissenschaftlichen Grundlagen und Anfänge des SQM konnten in der untersuchten Wissenschaftsliteratur nicht identifiziert werden, da keine Literatur dazu in den untersuchten Quellen gefunden wurde. In den 80er Jahren verbreitete sich die Erkenntnis, dass SQM ein Erfolgsfaktor für Unternehmen ist. Problemfaktoren der Softwarewartung wurden durch die Forschung herausgearbeitet. Die Praxis forderte ein TÜV Gütesiegel für Softwarequalität. In den 90er Jahren wandelte sich SQM von

der passiven Fehlererkennung zur aktiven Fehlervermeidung. Zertifizierungen nach ISO 9001 und Prozessverbesserungen mittels CMM und Bootstrap standen im Fokus. Das Testen von Software wurde relevant. Es kann ein Auseinanderdriften der Ergebnisse zwischen Wissenschaft und Praxis beobachtet werden. Während in der Praxis Testwerkzeuge und Wirtschaftlichkeit des SQM im Vordergrund stehen, stellte die Wissenschaft eher den Kunden und seine Anforderungen in den Fokus. Nach 2000 wurde in der Forschung ein Schwerpunkt auf die Optimierung von Modellen und Architekturen gelegt. Als Reaktion auf die Praxisnachfrage wurden Modelle zur Kostenoptimierung erarbeitet. Erfolgsfaktoren von Software Process Improvement wurden untersucht. In der Praxis hingegen wurden die vorhanden Modelle schon kombiniert und es wurde versucht, ein Rahmenwerk dafür zu implementieren. Aufgrund von Budgetkürzungen nach der Internetblase wurden vermehrt Automatisierungswerkzeuge wie automatisierte Tests zur IT-Optimierung eingesetzt. Auch Praxiskonferenzen legen den Schwerpunkt auf Testen als Erfolgsfaktor. Nach 2010 wurde Testen in der Wissenschaft zum zentralen Thema. Testbasierte Entwicklung unterstützt Agilität und Kostensenkung. Testautomatisierung und Integration von Ansätzen der Künstlichen Intelligenz in Software-Tests und zur Vorhersage der SW-Qualität werden aktuell untersucht. Neue Kennzahlen und Werkzeuge werden evaluiert. SIS und daraus resultierende verteilte Entwicklung spielen eine wichtige Rolle. In der Praxis hat sich Softwaretesten als eigenständiger Karriereweg etabliert. Trotz immer kürzerer Entwicklungszyklen steigen die Anforderungen an SW-Qualität aktueller Systeme. Agilität als Vorgehensmodell im SQM und Softwaretests sind die wichtigsten Themen der Praxisveröffentlichungen. Auch auf Konferenzen spielen agile und kontinuierliche Ansätze eine große Rolle. Testen und Softwarequalität in verteilten und mobilen Umgebungen gerät in den Fokus. Ansätze der Künstlichen Intelligenz im SQM sowie Testing im Bereich neuer Technologien werden zunehmend auf Konferenzen diskutiert.

Die Analyse der Stellenanzeigen komplettiert das Bild des aktuellen Standes im Software-Qualitätsmanagement. Um den Anforderungen der Dynamik und den komplexen software-intensiven Systemen heutzutage gerecht zu werden, bedarf es eines stringenten SQM mit automatisiertem Testen und agilem Vorgehen. Die Stellenanzeigen verdeutlichen, dass das Thema Software-Qualitätsmanagement in der Praxis aktuell einen geringen Stellenwert hat. Nur wenige Firmen bauen aktuell eine eigenständige SW-Qualitätsabteilung auf. Obwohl sich neue Technologien nach

und nach in der Praxis etablieren, spielt SW-Qualitätssicherung neuer Techniken bisher eine untergeordnete Rolle.

Heutzutage sind Systeme softwareintensiv, heterogen und dynamisch geworden. Dies betrifft Komponenten, Benutzer, Anforderungen und Architekturen. Zahlreiche Systeme erfordern viele Schnittstellen und nutzen Komponenten von Drittanbietern. Bei der Entwicklung von SIS müssen viele verschiedene Qualitätsherausforderungen beachtet werden. In den Entwicklungsprozessen müssen geeignete Qualitätssicherungstechniken und -werkzeuge integriert werden. Dazu gehören die Qualitätsbewertung von Anforderungen, Architektur und Zieltechnologien. Das Testen von Software ist traditionell eine wichtige Basis der Qualitätssicherung. Aufgrund komplexer Entwicklungsprozesse wird Testen zunehmend schwierig. Verteilte Entwicklungsteams können jeweils unterschiedliche Qualitätssicherungspraktiken haben. Ein sehr anspruchsvolles SQM-Gebiet ist die Sicherheit und der Datenschutz. Zukünftige SQM-Ansätze und unterstützende Techniken, Werkzeuge und Prozesse müssen diese Herausforderungen angehen. Qualitätsprozesse, -messungen und --management müssen auf verschiedene Nicht-Software-Komponenten angewendet werden. Verteilte, agile Teams und die Einbindung einer großen Anzahl von Drittanbieter-Diensten erfordern eine genauere Definition von Qualitätsmerkmalen. Zu den Forschungsthemen gehören dabei neue Verfahren zur Erfassung, Formalisierung und Validierung von Anforderungen, vor allem fehlen Modellierungssprachen und Werkzeuge für verteilte Qualitätskontrollen, die die Komplexität der Systeme bewältigen können.

Schien SQM zunächst ein stark praxisgetriebenes Thema, hat die Forschung inzwischen aufgeholt und stellt Ansätze zur Verfügung, die in der Praxis nicht vollständig nachgefragt werden. Bei der Analyse der Herausforderungen von SIS fällt auf, dass das Thema schon seit über 10 Jahren in der Wissenschaft diskutiert wird. Inzwischen wurden für viele der identifizierten Herausforderungen Forschungsvorhaben durchgeführt und Lösungen bzw. Lösungsansätze in der Wissenschaft erarbeitet. Die Forschung im Bereich von Software-Ökosystemen wie SIS ist aktuell sehr aktiv.[279] Software-intensive Systeme gibt es schon seit vielen Jahren. Die grundlegende Veränderung der letzten Jahre ist die Durchdringung aller Wirtschaftszweige mit SIS und die Verbreitung des sog. Software-intensive Business.

[279] Vgl. Jansen u.a. (2019).

Diese Etablierung zeigt sich auch in der Gründung der neuen Wissenschaftsdisziplin des „Software-Intensive Business (SIB)“ im Mai 2018 im Rahmen eines Dagstuhl-Seminars. Wichtigste Herausforderung der Software intensiven Systeme und damit für das software-intensive Business nach dem Dagstuhl-Report ist die Erhebung der Anforderungen an ein SIS.[280]

Der Rückgang des allgemeinen Interesses zu SQM, das über die Zahl der veröffentlichten Bücher und Trends bei Google in Kapitel 1.2 identifiziert wurde, konnte durch die Literaturstudie und die Untersuchung der Beiträge in Praxiszeitschriften nur teilweise bestätigt werden. Die absoluten Veröffentlichungszahlen im Bereich SQM nehmen zu, wie das Software Quality Journal gezeigt hat. Das allgemeine Interesse, das sich in den Veröffentlichungen der Praxis und der Literaturstudie zeigt, verzeichnet jedoch einen Rückgang der Veröffentlichungszahlen. Grund dafür könnte die Auseinandersetzung der Praxis mit der fortschreitenden Digitalisierung und sukzessiven Einführung neuer Technologien sein. Diese Arbeit konnte den Stand im SQM sowohl in der Praxis wie auch in der Forschung zeigen. Um zukünftig zuverlässige softwarebasierte Geschäftsmodelle zu haben, muss es in der Praxis zu einem Umdenken kommen. Neben der Optimierung von Softwaretests durch KI sollten die organisationalen Herausforderungen durch verteilte Systeme und deren Entwicklung stärker in den Fokus gerückt werden. Wertschöpfungskettenübergreifendes vernetztes „Software-Qualitätsmanagement 4.0“ wird auf lange Sicht der Erfolgsfaktor von Unternehmen werden.

Der wichtigste Kritikpunkt ist die Einschränkung der untersuchten Quellen auf Grund des begrenzten Rahmens dieser Arbeit. Die Anfänge des SQM konnten nicht identifiziert werden. Es kann nicht ausgeschlossen werden, dass relevante Beiträge nicht identifiziert wurden. Für den Stand der Wissenschaft sollte eine umfangreiche Untersuchung weiterer Journals und Konferenzen erfolgen.

Um die Diskrepanz bezüglich der zu- bzw. abnehmenden Veröffentlichungszahlen der letzten fünf Jahre (siehe Abbildung 13) erklären zu können, sollten die Veröffentlichungen im Software Quality Journal genauer untersucht und den Dimensionen des SQM zugeordnet werden. Eine Kritik ist die mögliche Verzerrung der Ergebnisse zwischen Wissenschaft und Praxis, da größtenteils nur deutsche

[280] Vgl. Abrahamsson u.a. (2018).

Praxiszeitschriften, aber internationale Konferenzen und Wissenschaftsjournals untersucht wurden. Diese Ungleichheit kann weiter verkleinert werden, indem zusätzlich internationale Praxiszeitschriften untersucht werden. Möglicherweise werden Praxisansätze nicht immer in den untersuchten Formaten diskutiert und werden ohne weitere empirische Arbeit nicht identifiziert.

Die Untersuchung der Stellenanzeigen kann durch saisonale oder regionale Schwankungen auf dem Stellenmarkt verzerrt sein. Als weiterer Schritt können internationale Stellenanzeigen über mehrere Jahre untersucht werden und SQM-Anforderungen mit anderen Disziplinen sowie damit verbundene Anforderungen verglichen werden, um das reale Gewicht von SQM in Unternehmen vertieft zu beleuchten.

Ein weiterer Kritikpunkt bezüglich der Arbeit ist die starke Fokussierung auf absolute Zahlen. Um eine detailliertere Aussage über die Relevanz des SQM treffen zu können, sollte in einem weiteren Schritt ein Vergleich zu Veröffentlichungszahlen im Bereich Software Engineering und Information Systems erfolgen. Zusätzlich könnten Konferenzen mit dem Schwerpunkt SIS und embedded Software auf Veröffentlichungen mit dem Thema Software-Qualitätsmanagement untersucht werden. Dadurch kann der aktuelle Stand der Forschung und der Bedarf, der durch SIS entsteht, noch detaillierter erhoben und bewertet werden.

Die zunehmende Durchdringung der Arbeitswelt und des gesellschaftlichen Lebens mit software-intensiven Systemen wird das software-intensive Business zwingen, qualitative hochwertige Software zu liefern. Unternehmen werden es sich nicht leisten können, minderwertige oder halbfertige Software zu verkaufen und damit im schlimmsten Fall Menschenleben zu gefährden. Es bedarf eines Umdenkens von klassischem SQM zu vernetztem und intelligentem „Software-Qualitätsmanagement 4.0“. Dafür müssen vorhandene Modelle und Werkzeuge übernommen und in einen holistischen Ansatz integriert werden.

Anhang

Protokoll für die Literaturanalyse zum Stand der Forschung im SQM

Hintergrund

Untersuchungen zum Software-Qualitätsmanagement (SQM) belegen, dass der Unternehmenserfolg mit der Einführung von SQM steigt und auch Produktivität in der Softwareentwicklung und Kundenzufriedenheit verbessert werden. [Vgl. Herbsleb u.a. (1994), S.15; Gibbs (1994), S.90; Jones, Bonsignour (2012), S.18; Nikolik (2012), S.1237]

Software wird heute nicht mehr nur zur Automatisierung und Rationalisierung von Prozessen eingesetzt, sondern findet sich als essentieller Bestandteil in vielen Produkten. Sogenannte Smart Products und Services bilden die Grundlage für neue Software-intensive Geschäftsmodelle. Auf Grund der zunehmenden Verbreitung von software-intensiven Systemen kann davon ausgegangen werden, dass Software-Qualitätsmanagement auch in Zukunft relevant sein wird.[Porter, Heppelmann (2014)]

Ziel dieser Arbeit ist es, den Stand der Forschung im Bereich Software-Qualitätsmanagement zu analysieren. Anschließend soll eine Einschätzung getroffen werden, inwieweit die vorliegenden Ansätze den Anforderungen an Softwarequalität und deren Management von agilen und digitalen Geschäftsfeldern genügen. Die Methode ist eine strukturierte Literaturanalyse.

Forschungsfragen

Die Hauptforschungsfrage dabei ist:

Wie ist der Stand der Forschung im Bereich Software-Qualitätsmanagement?

Hierzu sind im einzelnen folgende Unterforschungsfragen zu beantworten:

Unterforschungsfrage	Motivation
UFF1: Wie groß ist und war die Forschungsaktivität im Bereich SQM?	Zeitlichen Verlauf der Relevanz von SQM ermitteln
UFF2: Welche Forschungsthemen werden adressiert?	Identifikation von Trends und Forschungslücken
UFF3: Wie haben sich diese Schwerpunkte über die Zeit verändert?	Identifikation von Trends und zusammenhängenden möglicherweise kausalen Entwicklungen

Suchprozess

Um möglichst hochwertige Literatur zu untersuchen werden nur ausgewählte Journal- und Konferenzbeiträge mit hoher Bewertung manuell untersucht.

Betrachtete Journals:

Auswahl im Bereich Wirtschaftsinformatik (Information Systems) nach Senior Scholars' Basket of Journals, A+ und A gerankte Journals aus dem VHB Jourqual 3 und VHB Journals mit Software Schwerpunkt (B und C gerankt)

Zusätzliches Journal im Bereich SQM: Kann dafür als Grundlage im Vergleich zu nicht SQM spezifischen Quellen genommen werden

Software Engineering Journals nach Qualis (2017) A1 und A2 und Google Scholar Ranking (auf h5-Index)

Name des Journals	Bereich
(Mathematical Programming)	IS
ACM SIGSOFT International Symposium on Foundations of Software Engineering	SE
ACM Transactions on Software Engineering and Methodology	SE
Communications of the ACM	(SE)
Empirical Software Engineering	SE
European Journal of Information Systems	Information Systems (IS)
IEEE Software	SE
IEEE Transactions on Software Engineering	Software Engineering (SE)
INFORM Journal on Computing	IS
Information & Software Technology	SE
Information Systems Journal	IS
Information Systems Research	IS
Journal of AIS	IS
Journal of Information Technology	IS
Journal of MIS	IS
Journal of Strategic Information Systems	IS
Journal of Systems and Software	SE
MIS Quarterly	IS
Science of Computer Programming	SE
SIAM Journal on Computing	IS
Software Quality Journal	Software Quality
Software: Practice & Experience	SE

Betrachtete Konferenzen:

Konferenzen werden betrachtet um einen systematischen Fehler durch den sog. Publication Bias zu vermeiden (Kitchenham 2007)

Auswahl der Konferenzen im Bereich Wirtschaftsinformatik nach Hardgrave and Walstorm (1997)

Im Bereich Software Engineering nach ERA (2010) und QUALIS (2012) A bzw. A1A/A2 gerankte Konferenzen, sowie vorgeschlagene Konferenzen zum Thema der Gesellschaft für Informatik

Name der Konferenz	Abkürzung	Bereich
Academy of Management Conference	AOM	IS
ACM SIGSOFT Conference on the Foundations of Software Engineering	FSE_A	SE
ACM SIGSOFT International Symposium on the Foundations of Software Engineering	FSE	SE
Americas Conference on Information Systems	AMCIS	IS
Automated Software Engineering Conference	ASE	SE
Decision Sciences Institute - National and Regional Conferences	DSI	IS
European Conference on Information Systems	ECIS	IS
European Conference on Software Maintenance and Reengineering	ECSMR	SE
European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering	ESEC/FSE	SE
Fundamental Approaches to Software Engineering	FASE	SE
Hawaii International Conference on System Sciences	HICSS	IS
Information Resources Management Association Conference	IRMA	IS
Institute for Operations Research and the Management Sciences Conference	INFORMS	IS
International Association for Computer Information Systems	IACIS	IS
International Conference on Decision Support Systems	DSS	IS
International Conference on Evaluation and Assessment in Software Engineering	EASE	SE
International Conference on Information Systems	ICIS	IS
International Conference on Software Engineering	ICSE	SE
International Conference on Software Testing, Verification and Validation	ICST	SE
International Federation for Information Processing	IFIP	IS
International Symposium Component-Based Software Engineering	CBSE	SE
International Symposium on Empirical Software Engineering and Measurement	ESEM	SE
International Symposium on Software Reliability Engineering	ISSRE	SE
Internationale Tagung Wirtschaftsinformatik	WI	IS
Multikonferenz Wirtschaftsinformatik	MKWI	IS
Society of Information Management Conference	SIM	IS
Software Engineering	SE	SE
Software Management	SWM	SE
IEEE International Conference on Software Maintenance and Evolution	ICSME/ICSM	SE
European Conference on Software Architecture	ECSA	SE
IEEE International Working Conference on Mining Software Repositories	SMS	SE
ACM Symposium on User Interface Software and Technology	UIST	SE

Suchbegriffe

Einschlusskriterien

- „Software Quality“ in Titel und Abstract: um alle Ausprägungen und Aspekte von Software Quality Management abzudecken
- „Software-Qualitäts?“, „Softwarequalität?“ in Titel und Abstract: um alle Ausprägungen und Aspekte von Software Quality Management abzudecken
- Keine Einschränkung des Veröffentlichungsdatums
- Beiträge auf Englisch und Deutsch

Ausschlusskriterien

- Artikel bei denen „überall“ gesucht wird, da die Ergebnisse in einer ersten Suche als zu umfangreich und irrelevant identifiziert wurden, da häufig wenn „Software quality“ nur im Text genannt wird, das Thema Software Qualität eine zu geringe Wichtigkeit im Artikel hat
- Artikel die nicht peer-reviewed wurden
- Artikel die Software für allgemeines Qualitätsmanagement thematisieren
- Editorials/Book reviews etc.
- Kein Zugriff auf Volltext vorhanden
- Artikel die nur „Softwarequalität“ beschreiben

Erster Auswahlprozess

Ergebnisse werden in einer Tabelle dargestellt nach:

- Anzahl der identifizierten Beiträge pro Jahr pro Quelle
- Anzahl der Kandidaten pro Jahr pro Quelle
- Anzahl der ausgewählten Beiträge pro Jahr pro Quelle

Qualitätssicherung der Beiträge

Scoping Reviews, wie in dieser Arbeit, befassen sich mit der Entdeckung der Breite eines Forschungsgebietes, nicht ausschließlich der Qualität, und sollten Literatur aller Qualitätsstufen beinhalten, um das Gesamtbild abzubilden.[281] Deshalb wird über die Volltextprüfung auf Ein- und Ausschlusskriterien hinaus keine weitere Qualitätsuntersuchung durchgeführt. Die hohe Qualität die Beiträge wird darüber hinaus über die Auswahl von hochqualitativen Journals und Konferenzen sichergestellt.

Datensammlung

Gesammelte Informationen aus Papern und Konferenzproceedings:

Quelle (Journal oder Konferenz)
Veröffentlichungsdatum (Jahr)
Klassifikation
Thema
Autor und Zugehörigkeit zu…

[281] Vgl. Whittemore, Knafl (2005), S.459.

Erarbeitete Praxisempfehlungen ja/nein
Forschungsfrage
Zusammenfassung

Datenanalyse

Daten werden in einer Tabelle dargestellt und die Anzahl jeder Hauptkategorie, die sich aus den Forschungsfragen ableiten lässt wird gezählt.

Unterforschungsfrage	Auswertung
UFF1: Wie groß ist und war die Forschungsaktivität im Bereich SQM?	Anzahl der Beiträge nach Jahr
UFF2: Welche Forschungsthemen werden adressiert?	Anzahl der Beiträge nach Thema
UFF3: Wie haben sich diese Schwerpunkte über die Zeit verändert?	Themen nach Jahr aufgegliedert
UFF4: Wer sind die wichtigsten Forscher im Bereich SQM?	Anzahl der Veröffentlichungen pro Forscher
UFF5: In welchen Foren wird häufig zum Thema SQM veröffentlicht?	Anzahl der Beiträge pro Journal/Konferenz

Ergebnisdokumentation

Flow-Diagramm des Analyseprozess

Darstellung der Häufigkeiten in Histogrammen und Zeitstrahl-Diagrammen und Bubble Plots

Veröffentlichungen der Literaturstudie

(1990), Software Maintenance, '90 Conference On, Los Alamitos Oct. 1990
(1996), Software Engineering, 1996., Proceedings of the 18th International Conference on, Place of publication not identified 1996
(Hrsg., 1997), ICSE 97 - 19th Annual Conference on Software Engineering, New York Oct. 1997
(1999), International Conference on Software Maintenance, Los Alamitos Jan. 1999
(2002), Proceedings of the ICIS 2002 2002
(2004), The software evaluation framework 'SEF' extended, in: Information and Software Technology, 46, 2004, 15, S. 1037-1047
(2010), Journal of Computer Information Systems 2010
(2012), ISSRE 2012 2012
(2014 - 2014), 2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops
(2015 - 2015), 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering

Abdel-Hamid, T.K. (1988), The Economics of Software Quality Assurance: A Simulation-Based Case Study, in: MIS Quarterly, 12, 1988, 3, S. 395
Aberdour, M. (2007), Achieving Quality in Open-Source Software, in: IEEE Software, 24, 2007, 1, S. 58-64
(2017), 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM 2017 : 9-10 November 2017, Toronto, Ontario, Canada : proceedings, Piscataway, NJ 2017
Acuna, S.T., Gomez, M.N., Hannay, J.E., Juristo, N. und Pfahl, D. (2015), Are team personality and climate related to satisfaction and software quality? Aggregating results from a twice replicated experiment, in: Information and Software Technology, 57, 2015, S. 141-156
Acuña, S.T., Gómez, M. und Juristo, N. (2008), Towards understanding the relationship between team climate and software quality—a quasi-experimental study, in: Empirical Software Engineering, 13, 2008, 4, S. 401
Adrion, W.R. (1997), Proceedings of the 19th international conference on Software engineering, New York, NY 1997
Adrion, W. R. (1997), Proceedings of the 19th international conference on Software engineering - ICSE '97, New York, New York, USA 1997
Ai, J., Su, W. und Wang, F. (2018), Software Reliability Evaluation Method Based on a Software Network, in: Ghosh (Hrsg., 2018), S. 136-137
Alghamdi, J.S., Rufai, R.A. und Khan, S.M. (2005), OOMeter: A Software Quality Assurance Tool, in: (Hrsg., 2005), S. 190-191
Alshayeb, M. (2009), Empirical investigation of refactoring effect on software quality, in: Information and Software Technology, 51, 2009, 9, S. 1319-1326
Ambrose, P. und Eynon, J. (1998), A Theoretical Model for Software Quality Management, in: AMCIS 1998 Proceedings, 1998
Ampatzoglou, A., Frantzeskou, G. und Stamelos, I. (2012), A methodology to assess the impact of design patterns on software quality, in: Information and Software Technology, 54, 2012, 4, S. 331-346
Andersson, T. und Hellens, L.A. von (1997), Information systems work quality, in: Information and Software Technology, 39, 1997, 12, S. 837-844
Aranha, E. und Borba, P. (2007), An Estimation Model for Test Execution Effort, in: (Hrsg., 2007), S. 107-116
Arthur, L.J. (1997), Quantum improvements in software system quality, in: Communications of the ACM, 40, 1997, 6, S. 46-52
Austin, R.D. (2001), The Effects of Time Pressure on Quality in Software Development: An Agency Model, in: Information Systems Research, 12, 2001, 2, S. 195-207
Avgeriou, P., Zdun, U., Caracciolo, A., Lungu, M. F. und Nierstrasz, O. (2014), How Do Software Architects Specify and Validate Quality Requirements? - Software Architecture 2014
Ayaburi, E., Ko, M. und Walz, D. (2016), Understanding Antecedents and Consequence of Knowledge Sharing in Software Testing Teams: A Conceptual Model, in: AMCIS 2016 Proceedings, 2016
Azar, D., Harmanani, H. und Korkmaz, R. (2009), A hybrid heuristic approach to optimize rule-based software quality estimation models, in: Information and Software Technology, 51, 2009, 9, S. 1365-1376
Azar, D. und Vybihal, J. (2011), An ant colony optimization algorithm to improve software quality prediction models: Case of class stability, in: Information and Software Technology, 53, 2011, 4, S. 388-393
Babar, M. A., Gorton, I., Bode, S. und Riebisch, M. (2010), Impact Evaluation for Quality-Oriented Architectural Decisions regarding Evolvability - Software Architecture 2010
Babar, M. A., Gorton, I., Christensen, H. B., Hansen, K. M. und Lindstrøm, B. (2010), Lightweight and Continuous Architectural Software Quality Assurance Using the aSQA Technique - Software Architecture 2010
Baddoo, N. und Hall, T. (2003), De-motivators for software process improvement: an analysis of practitioners' views, in: Journal of Systems and Software, 66, 2003, 1, S. 23-33
Baker, E.R. (2001), Which Way, SQA?, in: IEEE Software, 18, 2001, 1, S. 16-18
Baker, R.A. (1997), Code reviews enhance software quality, in: Adrion (Hrsg., 1997), S. 570-571
Bakota, T., Hegedus, P., Kortvelyesi, P., Ferenc, R. und Gyimothy, T. (2011), A probabilistic software quality model, in: (Hrsg., 2011), S. 243-252
Banker, R.D. und Kemerer, C.F. (1993), Understanding the factors influencing the performance of software development groups: An exploratory group-level analysis, in: Information and Software Technology, 35, 1993, 8, S. 468-473
Barata, J. und Coyle, S. , Developing Socially-Constructed Quality Metrics in Agile: A Multi-Faceted Perspective, in: (Hrsg.,),
Barbagallo, D., Francalenei, C. und Merlo, F. (2008), The Impact of Social Netowrking on Software Design Quality and Development Effort in Open Source Projects, in: Proceedings of the International Conference on Information Systems, ICIS 2008, 2008
Barber, K.S. und Holt, J. (2001), Software architecture correctness, in: IEEE Software, 18, 2001, 6, S. 64-65
Baresi, L. und Heckel, R. (2006), ETAPS 2006 - Joint European Conferences on Theory and Practice of Software, Vienne, Austria, March 25 - April 2, 2006, Berlin 2006
Barney, S., Mohankumar, V., Chatzipetrou, P., Aurum, A. und Wohlin, C. (2014), Software quality across borders: Three case studies on company internal alignment, in: Information and Software Technology, 56, 2014, 1, S. 20-38
Barney, S., Petersen, K., Svahnberg, M., Aurum, A. und Barney, H. (2012), Software quality trade-offs: A systematic map, in: Information and Software Technology, 54, 2012, 7, S. 651-662
Baskerville, R., Levine, L., Ramesh, B. und Slaughter, S. (2002), Balancing quality and agility in Internet speed software development, in: (Hrsg., 2002),
Bauer, V., Heinemann, L., Hummel, B., Juergens, E. und Conradt, M. (2012), A framework for incremental quality analysis of large software systems, in: (Hrsg., 2012), S. 537-546

Behkamal, B., Kahani, M. und Akbari, M.K. (2009), Customizing ISO 9126 quality model for evaluation of B2B applications, in: Information and Software Technology, 51, 2009, 3, S. 599-609
Ben-Menachem, M. (2008), Towards management of software as assets: A literature review with additional sources, in: Information and Software Technology, 50, 2008, 4, S. 241-258
Bettenburg, N. und Hassan, A.E. (2013), Studying the impact of social interactions on software quality, in: Empirical Software Engineering, 18, 2013, 2, S. 375-431
Binder, L.H. und Poore, J.H. (1990), Field experiments with local software quality metrics, in: Software: Practice and Experience, 20, 1990, 7, S. 631-647
Binder, R.V. (1997), Can a manufacturing quality model work for software?, in: IEEE Software, 14, 1997, 5, 101-102, 105
Bird, C., Nagappan, N., Devanbu, P., Gall, H. und Murphy, B. (2009), Does distributed development affect software quality? An empirical case study of Windows Vista, in: (Hrsg., 2009), S. 518-528
Bissi, W., Serra Seca Neto, A.G. und Emer, M.C.F.P. (2016), The effects of test driven development on internal quality, external quality and productivity: A systematic review, in: Information and Software Technology, 74, 2016, S. 45-54
Boegh, J., Depanfilis, S., Kitchenham, B. und Pasquini, A. (1999), A method for software quality planning, control, and evaluation, in: IEEE Software, 16, 1999, 2, S. 69-77
Bøegh, J. (2008), A New Standard for Quality Requirements, in: IEEE Software, 25, 2008, 2, S. 57-63
Borodaev, L., Telea, A., Groenboom, R. und Smedinga, R. (2018), Software Metrics for Policy-Driven Software Development Life Cycle Automation, in: (Hrsg., 2018), S. 169-174
Bouktif, S., Kegl, B. und Sahraoui, H. (Oct. 2002), Combining software quality predictive models: an evolutionary approach, in: (Hrsg., Oct. 2002), S. 385-392
Çaglayan, B. und Bener, A.B. (2016), Effect of developer collaboration activity on software quality in two large scale projects, in: Journal of Systems and Software, 118, 2016, S. 288-296
Çalikli, G., Bener, A.B., Caglayan, B. und Misirli, A.T. (2012), Modeling Human Aspects to Enhance Software Quality Management, in: ICIS 2012 Proceedings, 2012
Card, D.N. (1988), Software product assurance: measurement and control, in: Information and Software Technology, 30, 1988, 6, S. 322-330
Cavalcante, E. (2015), On the Architecture-Driven Development of Software-Intensive Systems-of-Systems, in: (Hrsg., 2015), S. 899-902
Cavano, J.P. und LaMonica, F.S. (1987), Quality Assurance in Future Development Environments, in: IEEE Software, 4, 1987, 5, S. 26-34
Cesare, S. de, Lycett, M., Macredie, R.D., Patel, C. und Paul, R. (2010), Examining perceptions of agility in software development practice, in: Communications of the ACM, 53, 2010, 6, S. 126
Chen, C.-B., Lin, C.-T., Wang, C.-H. und Chang, C.-W. (2006), Model for measuring quality of software in DVRS using the gap concept and fuzzy schemes with GA, in: Information and Software Technology, 48, 2006, 3, S. 187-203
Chun, Y. (2006), Analysis of Software Quality via a Goal Programming Approach, in: undefined, 2006
Cleve, A. (2013), 2013 17th European Conference on Software Maintenance and Reengineering (CSMR) - 5 - 8 March 2013, Genova, Italy, Piscataway, NJ 2013
Cobb, R.H. und Mills, H.D. (1990), Engineering software under statistical quality control, in: IEEE Software, 7, 1990, 6, S. 45-54
Collofello, J.S. und Buck, J.J. (1987), Software Quality Assurance for Maintenance, in: IEEE Software, 4, 1987, 5, S. 46-51
Coskun, E. und Baydar, M. (2005), Selections of Software Quality Attributes: Joint Optimization Model for Customer Needs and Producers' Software Quality Expenditures, in: AMCIS 2005 Proceedings, 2005
Councill, W.T. (1999), Third-party testing and the quality of software components, in: IEEE Software, 16, 1999, 4, S. 55-57
Coupe, R.T. und Onodu, N.M. (1996), An Empirical Evaluation of the Impact of Case on Developer Productivity and Software Quality, in: Journal of Information Technology, 11, 1996, 2, S. 173-181
Crispin, L. (2006), Driving Software Quality: How Test-Driven Development Impacts Software Quality, in: IEEE Software, 23, 2006, 6, S. 70-71
Crnkovic, I., Chechik, M. und Grünbacher, P. (2014), ASE'14 - Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering, September 15-19, 2014, Västerås, Sweden, New York, NY 2014
Crnkovic, I., Gruhn, V., Book, M., Oliveira, L. B. R. und Nakagawa, E. Y. (2011), A Service-Oriented Reference Architecture for Software Testing Tools - Software Architecture 2011
Cuesta, C. E., Garlan, D. und Pérez, J. (2018), Software Architecture - 12th European Conference on Software Architecture, ECSA 2018, Madrid, Spain, September 24-28, 2018, Proceedings, Cham 2018
Cuesta, C. E., Garlan, D., Pérez, J., Schneider, Y., Busch, A. und Koziolek, A. (2018), Using Informal Knowledge for Improving Software Quality Trade-Off Decisions - Software Architecture 2018
Cukic, B. (2005), The Virtues of Assessing Software Reliability Early, in: IEEE Software, 22, 2005, 3, S. 50-53
Dalal, S.R., Horgan, J.R. und Kettenring, J.R. (Hrsg., 1993, Reliable software and communication: software quality, reliability, and safety: IEEE Computer Society Press, http://dl.acm.org/ft_gateway.cfm?id=257668&type=pdf
Darmawan, N., Chong, A., Ooi, K.-B. und Boon, I.T. (2010), Factor strategy model: proofs of prototype concept for software quality evaluation, in: (Hrsg., 2010),
Datta, S. (2018), How does developer interaction relate to software quality? an examination of product development data, in: Empirical Software Engineering, 23, 2018, 3, S. 1153-1187

Defmaie, M., Jacobs, P. und Jaques, C. (1991), Quality Assurance Experience Database, in: ISSRE, 1991
Denning, P.J. (2016), Software quality, in: Communications of the ACM, 59, 2016, 9, S. 23-25
Dion, R. (1992), Elements of a process-improvement program (software quality), in: IEEE Software, 9, 1992, 4, S. 83-85
DomíNguez-Mayo, F.J., Escalona, M.J., MejíAs, M., Ross, M. und Staples, G. (2012), Quality evaluation for Model-Driven Web Engineering methodologies, in: Information and Software Technology, 54, 2012, 11, S. 1265-1282
Dowling, P. und McGrath, K. (2015), Using Free and Open Source Tools to Manage Software Quality, in: Queue, 13, 2015, 4, S. 20
Drehmer, D. und Dekleva, S. (1993), MEASURING SOFTWARE ENGINEERING MATURITY: A RASCH CALIBRATION, in: ICIS 1993 Proceedings, 1993
Dromey, R.G. (1996), Cornering the Chimera [software quality], in: IEEE Software, 13, 1996, 1, S. 33-43
Ehrlich, W.K., Stampfel, J.P. und Wu, J.R. (1990), Application of software reliability modeling to product quality and test process, in: (Hrsg., 1990), S. 108-116
Eicker, S., Hegmanns, C. und Malich, S. (2008), Projektbezogene Auswahl von Bewertungsmethoden für Softwarearchitekturen, in: Multikonferenz Wirtschaftsinformatik, 2008
Entin, V., Winder, M., Zhang, B. und Claus, A. (2015), A process to increase the model quality in the context of model-based testing, in: (Hrsg., 2015), S. 1-7
Erikkson, I. und McFadden, F. (1993), Quality function deployment: a tool to improve software quality, in: Information and Software Technology, 35, 1993, 9, S. 491-498
(2005), Ninth European Conference on Software Maintenance and Reengineering - Proceedings : 21-23 March, 2005, Manchester, UK, Los Alamitos, Calif 2005
Febrero, F., Calero, C. und Ángeles Moraga, M. (2016), Software reliability modeling based on ISO/IEC SQuaRE, in: Information and Software Technology, 70, 2016, S. 18-29
Ferenc, R. (2009), 13th European Conference on Software Maintenance and Reengineering, 2009 - CSMR '09 ; 24 - 27 March 2009, Kaiserslautern, Germany, Piscataway, NJ 2009
Fiadeiro, J. L., Inverardi, P., Kopetz, R. und Moreau, P.-E. (2008), Software Quality Improvement Via Pattern Matching - Fundamental Approaches to Software Engineering 2008
Foucault, M., Teyton, C., Lo, D., Blanc, X. und Falleri, J.-R. (2015), On the usefulness of ownership metrics in open-source software projects, in: Information and Software Technology, 64, 2015, S. 102-112
Franch, X. und Carvallo, J.P. (2003), Using quality models in software package selection, in: IEEE Software, 20, 2003, 1, S. 34-41
Fucci, D., Turhan, B. und Oivo, M. (2015), On the effects of programming and testing skills on external quality and productivity in a test-driven development context, in: Lv, Zhang, Babar, Ali Babar (Hrsg., 2015), S. 1-6
Ghosh, S. (2018), 29th IEEE International Symposium on Software Reliability Engineering workshops - ISSREW 2018 : 15-18 October 2018, Memphis, Tennessee, USA : proceedings, Piscataway, NJ 2018
Ghotra, B., McIntosh, S. und Hassan, A.E. (2015), Revisiting the Impact of Classification Techniques on the Performance of Defect Prediction Models, in: (Hrsg., 2015), S. 789-800
Glick, B. (Oct. 1990), An SQA quality tracking methodology, in: (Hrsg., Oct. 1990), S. 178-188
Gligoric, M., Negara, S., Legunsen, O. und Marinov, D. (2014), An empirical evaluation and comparison of manual and automated test selection, in: Crnkovic, Chechik, Grünbacher (Hrsg., 2014), S. 361-372
Goedicke, M., Menzies, T. und Saeki, M. (2012), 2012 proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE) - 3 - 7 Sept. 2012, Essen, Piscataway, NJ 2012
Goldbach, T., Kemper, V. und Benlian, A. (2014), Mobile Application Quality and Platform Stickiness under Formal vs. Self-Control — Evidence from an Experimental Study, in: ICIS 2014 Proceedings, 2014
Goldstein, M. und Segall, I. (2015), Automatic and Continuous Software Architecture Validation, in: (Hrsg., 2015), S. 59-68
Gómez, M.N. und Acuña, S.T. (2014), A replicated quasi-experimental study on the influence of personality and team climate in software development, in: Empirical Software Engineering, 19, 2014, 2, S. 343-377
Gonzalez-Sanchez, A., Piel, É., Abreu, R., Gross, H.-G. und van Gemund, A.J.C. (2011), Prioritizing tests for software fault diagnosis, in: Software: Practice and Experience, 12, 2011, 3, n/a-n/a
Gopal, A. und Koka, B.R. (2010), The Role of Contracts on Quality and Returns to Quality in Offshore Software Development Outsourcing, in: Decision Sciences, 41, 2010, 3, S. 491-516
Gorla, N. und Lin, S.-C. (2010), Determinants of software quality: A survey of information systems project managers, in: Information and Software Technology, 52, 2010, 6, S. 602-610
Gorton, I., Heineman, G. T., Crnkovic, I., Schmidt, H. W., Stafford, J. A., Szyperski, C. A. und Wallnau, K. (2006), Component-based software engineering - 9th international symposium, CBSE 2006, Västerås, Sweden, June 29 - July 1, 2006 ; proceedings, Berlin 2006
Grady, R.B. (1987), Measuring and Managing Software Maintenance, in: IEEE Software, 4, 1987, 5, S. 35-45
Grady, R.B. (1993), Practical results from measuring software quality, in: Communications of the ACM, 36, 1993, 11, S. 62-68
Gree, G.C., Hevner, A.R. und Collins, R. (2005), The impacts of quality and productivity perceptions on the use of software process improvement innovations, in: Information and Software Technology, 47, 2005, 8, S. 543-553
Guinan, P., Hopkins, G.N. und Cooprider, J. (1992), AUTOMATION OF THE APPLICATIONS DEVELOPMENT LIFE CYCLE: MEASURING THE VALUE ADDED, in: ICIS 1992 Proceedings, 1992
Güldali, B., Funke, H., Jahnich, M., Sauer, S. und Engels, G. (2009), Semi-automated Test Planning for e-ID Systems by Using Requirements Clustering, in: (Hrsg., 2009), S. 29-39

Gustafson, G.G. und Kerr, R.J. (1982), Some practical experience with a software quality assurance program, in: Communications of the ACM, 25, 1982, 1, S. 4-12
Guzman, L., Vollmer, A.M., Ciolkowski, M. und Gillmann, M. (2017), Formative Evaluation of a Tool for Managing Software Quality, in: (Hrsg., 2017), S. 297-306
Haag, S., Raja, M.K. und Schkade, L.L. (1996), Quality function deployment usage in software development, in: Communications of the ACM, 39, 1996, 1, S. 41-49
Hackbarth, R., Mockus, A., Palframan, J. und Sethi, R. (2016), Improving Software Quality as Customers Perceive It, in: IEEE Software, 33, 2016, 4, S. 40-45
Haley, T.J. (1996), Software process improvement at Raytheon, in: IEEE Software, 13, 1996, 6, S. 33-41
Hall, T. (1995), No Quality Without Equality, in: IEEE Software, 1995
Harter, D.E., Krishnan, M.S. und Slaughter, S.A. (Hrsg., 1998, The life cycle effects of software process improvement: a longitudinal analysis: Association for Information Systems, http://dl.acm.org/ft_gateway.cfm?id=353093&type=pdf
Harter, D.E., Krishnan, M.S. und Slaughter, S.A. (2000), Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development, in: Management Science, 46, 2000, 4, S. 451-466
Harter, D.E. und Slaughter, S.A. (Hrsg., 2000, Process maturity and software quality: a field study: Association for Information Systems, http://dl.acm.org/ft_gateway.cfm?id=359769&type=pdf
Harter, D.E. und Slaughter, S.A. (2003), Quality Improvement and Infrastructure Activity Costs in Software Development: A Longitudinal Analysis, in: Management Science, 49, 2003, 6, S. 784-800
Hayes, J.H., Dekhtyar, A. und Sundaram, S.K. (2005), Improving After-the-Fact Tracing and Mapping: Supporting Software Quality Predictions, in: IEEE Software, 22, 2005, 6, S. 30-37
Hecht, G., Benomar, O., Rouvoy, R., Moha, N. und Duchien, L. (2015), Tracking the Software Quality of Android Applications Along Their Evolution (T), in: (Hrsg., 2015), S. 236-247
Henderson-Sellers, B. (1991), Parallels between Object-Oriented Software Development and Total Quality Management, in: Journal of Information Technology, 6, 1991, 2, S. 63-67
Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W. und Paulk, M. (1997), Software quality and the Capability Maturity Model, in: Communications of the ACM, 40, 1997, 6, S. 30-40
Herrera, E.M. und Ramírez, R.A.T. (2003), A Methodology for Self-Diagnosis for Software Quality Assurance in Small and Medium-Sized Industries in Latin America, in: The Electronic Journal of Information Systems in Developing Countries, 15, 2003, 1, S. 1-13
Herzig, K., Greiler, M., Czerwonka, J. und Murphy, B. (2015), The art of testing less without sacrificing quality, 2015
Hinley, D. (1996), Software evolution management: a process-oriented perspective, in: Information and Software Technology, 38, 1996, 11, S. 723-730
Hirayama, M., Sato, H., Yamada, A. und Tsuda, J. (1990), Practice of quality modeling and measurement on software life-cycle, in: (Hrsg., 1990), S. 98-107
Hneif, M. und Lee, S.P. (2011), Using Guidelines to Improve Quality in Software Nonfunctional Attributes, in: IEEE Software, 28, 2011, 6, S. 72-77
Hofman, R. (2011), Behavioral economics in software quality engineering, in: Empirical Software Engineering, 16, 2011, 2, S. 278-293
Hollenbach, C., Young, R., Pflugrad, A. und Smith, D. (1997), Combining quality and software improvement, in: Communications of the ACM, 40, 1997, 6, S. 41-45
Hon, S.E. (1990), Assuring software quality through measurements: A buyer's perspective, in: Journal of Systems and Software, 13, 1990, 2, S. 117-130
Huang, J., Keung, J.W., Sarro, F., Li, Y.-F., Yu, Y.T., Chan, W.K. und Sun, H. (2017), Cross-validation based K nearest neighbor imputation for software quality datasets: An empirical study, in: Journal of Systems and Software, 132, 2017, S. 226-252
Huang, L. und Boehm, B. (2006), How Much Software Quality Investment Is Enough: A Value-Based Approach, in: IEEE Software, 23, 2006, 5, S. 88-95
Huchard, M., Käster, C. und Fraser, G. (2018), ASE '18 - Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering : September 3-7, 2018, Montpellier, France, New York, NY 2018
Hudepohl, J.P., Aud, S.J., Khoshgoftaar, T.M., Allen, E.B. und Mayrand, J. (1996), Integrating metrics and models for software risk assessment, in: (Hrsg., 1996), S. 93-98
Huffman Hayes, J. (2003), Do you like Pina Coladas? How improved communication can improve software quality, in: IEEE Software, 20, 2003, 1, S. 90-92
Hungerford, B.C., Hevner, A.R. und Collins, R.W. (2004), Reviewing software diagrams: a cognitive study, in: IEEE Transactions on Software Engineering, 30, 2004, 2, S. 82-96
(2007), First International Symposium on Empirical Software Engineering and Measurement, 2007 - ESEM 2007 ; 20 - 21 Sept. 2007, Madrid, Spain ; proceedings, Los Alamitos, Calif. 2007
(1999), Software Reliability Engineering - Proceedings of the 10th International Symposium, Boca Raton, Florida, 1999, Los Alamitos Nov. 1999
(2002), 18th International Conference on Software Maintenance (ICSM 2002) - Maintaining Distributed Heterogeneous Systems, Los Alamitos Oct. 2002
(2002), 13th International Symposium on Software Reliability Engineering - Proceedings, Annapolis, Maryland, 12-15 November 2002, Los Alamitos Nov. 2002
(2003), 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2003), Los Alamitos, Piscataway Jan. 2003

(2002), Proceedings of the 2002 IEEE International Conference on Fuzzy Systems - FUZZ-IEEE'02 : May 12-17, 2002, Hilton Hawaiian Village Hotel, Honolulu, Hawaii, Piscataway, NJ 2002
(2015), 2015 30th IEEE/ACM International Conference on Automated Software Engineering - 9-13 November 2015, Lincoln, Nebraska : proceedings, Piscataway, NJ 2015
(2017), 2017 IEEE/ACM 39th International Conference on Software Engineering companion - ICSE-C 2017 : 20-28 May 2017, Buenos Aires, Argentina : proceedings, Piscataway, NJ 2017
In, H.P., Baik, J., Kim, S., Yang, Y. und Boehm, B. (2006), A quality-based cost estimation model for the product line life cycle, in: Communications of the ACM, 49, 2006, 12, S. 85
(2009), IEEE 31st International Conference on Software Engineering, 2009 - ICSE 2009 ; 16 - 24 May 2009, Vancouver, Canada ; proceedings, Piscataway, NJ 2009
(2009), 24th IEEE/ACM International Conference on Automated Software Engineering, 2009 - ASE '09 ; date: 16 - 20 November 2009, Auckland, Piscataway, NJ 2009
(2014), IEEE International Conference on Software Maintenance and Evolution (ICSME), 2014 - Sept. 29 [i.e. 28], 2014 - Oct. 3, 2014, Victoria, British Columbia, Canada, Piscataway, NJ 2014
(2007), IEEE International Conference on Software Maintenance, 2007 - ICSM 2007 ; 2 - 5 Oct. 2007, Maison Internationale, Paris, France, Piscataway, NJ 2007
(2011), 27th IEEE International Conference on Software Maintenance (ICSM), 2011 - 25 - 30 Sept. 2011, Williamsburg, VA, USA, Piscataway, NJ 2011
(2012), 28th IEEE International Conference on Software Maintenance (ICSM), 2012 - 23 - 28 Sept. 2012, Riva del Garda, Trento, Italy, Piscataway, NJ 2012
(2015), IEEE Eighth International Conference on Software Testing, Verification and Validation workshops (ICSTW), 2015 - 13 - 17 April 2015, Graz, Austria ; proceedings, Piscataway, NJ 2015
(2013), 2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE) - 4 - 7 Nov. 2013, Pasadena, CA, USA, Piscataway, NJ 2013
(1999), Software Maintenance and Reengineering - Proceedings. European Conference on Software Maintenance and Reengineering (3rd: 1999: University of Amsterdam, the Netherlands), Los Alamitos March 1999
(1990), 12th International Conference on Software Engineering, March 26-30, 1990-Nice, France - Proceedings, Washington, D.C 1990
(1994), Proceedings, 16th International Conference on Software Engineering - May 16-21, 1994, Sorrento, Italy, Los Alamitos, CA 1994
International Conference on Software Engineering, IEEE Computer Society, Association for Computing Machinery und International Conference on Software Engineering (ICSE) (Hrsg., 1997), Proceedings of the 1997 International Conference on Software Engineering - May 17 - 23, 1997, Boston, Massachusetts, USA, Los Alamitos, Calif. 1997
(2003), International Conference on Software Maintenance - ICSM 2003 : proceedings : 22-26 September 2003, Amsterdam, the Netherlands, Los Alamitos, Calif 2003
(1994), Proceedings - 5th International Symposium on Software Reliability Engineering : November 6-9, 1994, Monterey, California, Los Alamitos, Calif 1994
(1997), Proceedings - The Eighth International Symposium on Software Reliability Engineering : November 2-5, 1997, Albuquerque, New Mexico, Los Alamitos, Calif 1997
(2005), 16th IEEE International Symposium on Software Reliability Engineering - ISSRE 2005 : proceedings : 8-11 November, 2005, Chicago, Illinois, Los Alamitos, Calif 2005
(1995), Proceedings - The Sixth International Symposium on Software Reliability Engineering : October 24-27, 1995, Toulouse, France, Los Alamitos, Calif 1995
(1998), The Ninth International Symposium on Software Reliability Engineering - Proceedings : Paderborn, Germany, November 4-7, 1998, Los Alamitos, Calif 1998
(1996), Proceedings - The Seventh International Symposium on Software Reliability Engineering : October 30 - November 2, 1996, White Plains, New York, Los Alamitos, Calif 1996
(2000), Proceedings, 11th International Symposium on Software Reliability Engineering - ISSRE 2000 : October 8-11,2000, San Jose, California, USA, Los Alamitos, Calif 2000
(2018), 2018 IEEE 11th International Conference on Software Testing, Verification and Validation workshops - ICSTW 2018 : 9-13 April 2018, Västerås, Sweden : proceedings, Piscataway, NJ 2018
(2018), 2018 IEEE 11th International Conference on Software Testing, Verification and Validation workshops - ICSTW 2018 : 9-13 April 2018, Västerås, Sweden : proceedings, Piscataway, NJ 2018
Izurieta, C., Griffith, I. und Huvaere, C. (2017), An Industry Perspective to Comparing the SQALE and Quamoco Software Quality Models, in: (Hrsg., 2017), S. 287-296
Jabangwe, R., Šmite, D. und Hessbo, E. (2016), Distributed software development in an offshore outsourcing project: A case study of source code evolution and quality, in: Information and Software Technology, 72, 2016, S. 125-136
Janzen, D. und Saiedian, H. (2008), Does Test-Driven Development Really Improve Software Design Quality?, in: IEEE Software, 25, 2008, 2, S. 77-84
Jiang, B., Tse, T.H., Grieskamp, W., Kicillof, N., Cao, Y., Li, X. und Chan, W.K. (2011), Assuring the model evolution of protocol software specifications by regression testing process improvement, in: Software: Practice and Experience, 16, 2011, 5, n/a-n/a
Jinzenji, K., Hoshino, T., Williams, L. und Takahashi, K. (2013), An experience report for software quality evaluation in highly iterative development methodology using traditional metrics, in: (Hrsg., 2013), S. 310-319

Joblin, M., Mauerer, W., Apel, S., Siegmund, J. und Riehle, D. (2015), From Developer Networks to Verified Communities: A Fine-Grained Approach, in: (Hrsg., 2015), S. 563-573
Johnson, P.M. (1994), An instrumented approach to improving software quality through formal technical review, in: (Hrsg., 1994), S. 113-122
Jones, W.D., Hudepohl, J.P., Khoshgoftaar, T.M. und Allen, E.B. (March 1999), Application of a usage profile in software quality models, in: (Hrsg., March 1999), S. 148-157
Jung, H.-W., Kim, S.-G. und Chung, C.-S. (2004), Measuring Software Product Quality: A Survey of ISO/IEC 9126, in: IEEE Software, 21, 2004, 05, S. 88-92
Kafura, D. und Henry, S. (1981), Software quality metrics based on interconnectivity, in: Journal of Systems and Software, 2, 1981, 2, S. 121-131
Kajihara, J., Amamiya, G. und Saya, T. (1993), Learning from bugs (software quality control), in: IEEE Software, 10, 1993, 5, S. 46-54
Kajko-Mattsson, M. (1998), A conceptual model of software maintenance, in: Torii (Hrsg., 1998), S. 422-425
Kamp, P.-H. (2014), Quality Software Costs Money - Heartbleed Was Free, in: Queue, 12, 2014, 6, S. 10-14
Kannabiran, G. und Sankaran, K. (2011), Determinants of software quality in offshore development – An empirical study of an Indian vendor, in: Information and Software Technology, 53, 2011, 11, S. 1199-1208
Kelly, D.P. und Oshana, R.S. (2000), Improving software quality using statistical testing techniques, in: Information and Software Technology, 42, 2000, 12, S. 801-807
Kemerer, C.F. und Paulk, M.C. (2009), The Impact of Design and Code Reviews on Software Quality: An Empirical Study Based on PSP Data, in: IEEE Transactions on Software Engineering, 35, 2009, 4, S. 534-550
Khoshgoftaar, T.M. und Allen, E.B. (Jan. 1999), Can a software quality model hit a moving target?, in: (Hrsg., Jan. 1999), S. 68-70
Khoshgoftaar, T.M., Allen, E.B., Jones, W.D. und Hudepohl, J.P. (2000), Classification-tree models of software-quality over multiple releases, in: IEEE Transactions on Reliability, 49, 2000, 1, S. 4-11
Khoshgoftaar, T.M. und Lanning, D.L. (1994), On the impact of software product dissimilarity on software quality models, in: (Hrsg., 1994), S. 104-114
Khoshgoftaar, T.M., Munson, J.C., Bhattacharya, B.B. und Richardson, G.D. (1992), Predictive modeling techniques of software quality from software measures, in: IEEE Transactions on Software Engineering, 18, 1992, 11, S. 979-987
Khoshgoftaar, T.M. und Seliya, N. (Nov. 2002), Improving usefulness of software quality classification models based on Boolean discriminant functions, in: (Hrsg., Nov. 2002), S. 221-230
Khoshgoftaar, T.M., Shan, R. und Allen, E.B. (2000), Improving tree-based models of software quality with principal components analysis, in: (Hrsg., 2000), S. 198-209
Khoshgoftaar, T.M. und Seliya, N. (2003), Analogy-Based Practical Classification Rules for Software Quality Estimation, in: Empirical Software Engineering, 8, 2003, 4, S. 325-350
Khoshgoftaar, T.M. und Seliya, N. (2004), Comparative Assessment of Software Quality Classification Techniques: An Empirical Case Study, in: Empirical Software Engineering, 9, 2004, 3, S. 229-257
Khoshgoftaar, T.M., Seliya, N. und Herzberg, A. (2005), Resource-oriented software quality classification models, in: Journal of Systems and Software, 76, 2005, 2, S. 111-126
Kitchenham, B. und Pfleeger, S.L. (1996), Software quality: the elusive target [special issues section], in: IEEE Software, 13, 1996, 1, S. 12-21
Klas, M., Bauer, T., Dereani, A., Soderqvist, T. und Helle, P. (5/16/2015 - 5/24/2015), A Large-Scale Technology Evaluation Study: Effects of Model-based Analysis and Testing, in: (Hrsg., 5/16/2015 - 5/24/2015), S. 119-128
Koschke, R., Krinke, J. und Robillard, M. (2015), 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME 2015) - Bremen, Germany, 29 September - 1 October 2015, Piscataway, NJ 2015
Kozlov, D., Koskinen, J., Markkula, J. und Sakkinen, M. (2007), Evaluating the Impact of Adaptive Maintenance Process on Open Source Software Quality, in: (Hrsg., 2007), S. 186-195
Krishnan, M.S., Kriebel, C.H., Kekre, S. und Mukhopadhyay, T. (2000), An Empirical Analysis of Productivity and Quality in Software Products, in: Management Science, 46, 2000, 6, S. 745-759
Lago, P., Koçak, S.A., Crnkovic, I. und Penzenstadler, B. (2015), Framing sustainability as a property of software quality, in: Communications of the ACM, 58, 2015, 10, S. 70-78
Laitenberger, O. (1998), Studying the effects of code inspection and structural testing on software quality, in: (Hrsg., 1998), S. 237-246
Lehrig, S., Hilbrich, M. und Becker, S. (2018), The architectural template method: templating architectural knowledge to efficiently conduct quality-of-service analyses, in: Software: Practice and Experience, 48, 2018, 2, S. 268-299
Li, J., Stalhane, T., Conradi, R. und Kristiansen, J.M.W. (2012), Enhancing Defect Tracking Systems to Facilitate Software Quality Improvement, in: IEEE Software, 29, 2012, 2, S. 59-66
Li, X., Mutha, C. und Smidts, C.S. (2016), An automated software reliability prediction system for safety critical software, in: Empirical Software Engineering, 21, 2016, 6, S. 2413-2455
Lientz, B.P. und Swanson, E.B. (1981), Problems in application software maintenance, in: Communications of the ACM, 24, 1981, 11, S. 763-769
Liu, H., Xie, X., Yang, J., Lu, Y. und Chen, T.Y. (2011), Adaptive random testing through test profiles, in: Software: Practice and Experience, 17, 2011, 5, n/a-n/a
Liu, H., Li, G., Ma, Z. und Shao, W. (2007), Scheduling of conflicting refactorings to promote quality improvement, in: Stirewalt, Egyed, Fischer (Hrsg., 2007), S. 489

Liu, X., Kane, G. und Bambroo, M. (Jan. 2003), An intelligent early warning system for software quality improvement and project management, in: (Hrsg., Jan. 2003), S. 32-38
Liu, X.F. (1998), A quantitative approach for assessing the priorities of software quality requirements, in: Journal of Systems and Software, 42, 1998, 2, S. 105-113
Liu, Y., Khoshgoftaar, T.M. und Seliya, N. (2010), Evolutionary Optimization of Software Quality Modeling with Multiple Repositories, in: IEEE Transactions on Software Engineering, 36, 2010, 6, S. 852-864
Lu, Y. und Käkölä, T. (2014), A dynamic life-cycle model for the provisioning of software testing services, in: Systems Science & Control Engineering, 2, 2014, 1, S. 549-561
Lutellier, T., Chollak, D., Garcia, J., Tan, L., Rayside, D., Medvidović, N. und Kroeger, R. , Comparing software architecture recovery techniques using accurate dependencies, in: (Hrsg.,), S. 69-78
Lv, J., Zhang, H., Babar, M. A. und Ali Babar, M. (2015), Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering, New York, New York 2015
MacCormack, A., Kemerer, C.F., Cusumano, M. und Crandall, B. (2003), Trade-offs between productivity and quality in selecting software development practices, in: IEEE Software, 20, 2003, 5, S. 78-85
Maletic, J., Soliman, K. und Moreno, M. (1999), Identification of Test Cases from Business Requirements of Software Systems,, in: AMCIS 1999 Proceedings, 1999
Mangalaraj, G. und Bhadauria, V.S. (2007), Pair Programming: Effects of Trust on Software Quality, in: AMCIS 2007 Prroceedings, 2007
Marín, B., Giachetti, G., Pastor, O., Vos, T.E.J. und Abran, A. (2013), Using a functional size measurement procedure to evaluate the quality of models in MDD environments, in: ACM Transactions on Software Engineering and Methodology, 22, 2013, 3, S. 1
Masuda, S., Ono, K., Yasue, T. und Hosokawa, N. (2018), A Survey of Software Quality for Machine Learning Applications, in: (Hrsg., 2018), S. 279-284
Mathiassen, L., Ngwenyama, O.K. und Aaen, I. (2005), Managing Change in Software Process Improvement, in: IEEE Software, 22, 2005, 6, S. 84-91
McCaffery, F., Pikkarainen, M. und Richardson, I. (2008), Ahaa --agile, hybrid assessment method for automotive, safety critical smes, in: Schäfer, Dwyer, Gruhn (Hrsg., 2008), S. 551
Mendes, E., Counsell, S. und Petersen, K. (2017), Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering - EASE'17 : 15-16 June 2017, Karlskrona, Sweden, New York, New York 2017
Mens, T. (2012), 16th European Conference on Software Maintenance and Reengineering (CSMR), 2012 - 27 - 30 March 2012, Szeged, Hungary ; proceedings, Piscataway, NJ 2012
Menzies, M. und Hihn, J. (2006), Evidence-based cost estimation better-quality for software, in: IEEE Software, 23, 2006, 4, S. 64-66
Meyer, B., Baresi, L. und Mezini, M. (2013), 2013 9th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE) - Proceedings : August 18-26, 2013, Saint Petersburg, Russia, New York, NY 2013
Meyer, B., Baresi, L. und Mezini, M. (2013), 2013 9th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE) - Proceedings : August 18-26, 2013, Saint Petersburg, Russia, New York, NY 2013
Mishra, B.K., Prasad, A. und Raghunathan, S. (2002), Quality and Profits Under Open Source Versus Closed Source, in: undefined, 2002
Miyoshi, T. und Azuma, M. (1993), An empirical study of evaluating software development environment quality, in: IEEE Transactions on Software Engineering, 19, 1993, 5, S. 425-435
Mo, R., Snipes, W., Cai, Y., Ramaswamy, S., Kazman, R. und Naedele, M. (2018), Experiences applying automated architecture analysis tool suites, in: Huchard, Käster, Fraser (Hrsg., 2018), S. 779-789
(2015), Second ACM International Conference on Mobile Software Engineering and Systems - MOBILESoft 2015 - May 16-17, 2015, Florence, Italy, Piscataway, NJ 2015
(2015), Second ACM International Conference on Mobile Software Engineering and Systems - MOBILESoft 2015 - May 16-17, 2015, Florence, Italy, Piscataway, NJ 2015
Motoei, A. (1996), Software products evaluation system: quality models, metrics and processes—International Standards and Japanese practice, in: Information and Software Technology, 38, 1996, 3, S. 145-154
Munson, J.C. und Khoshgoftaar, Y.M. (1990), Regression modelling of software quality: empirical investigation, 1990
Nagappan, N., Murphy, B. und Basili, V. (2008), The influence of organizational structure on software quality, in: Schäfer, Dwyer, Gruhn (Hrsg., 2008), S. 521
Nakamura, M. und Hamagami, T. (2012), A Software quality evaluation method using the change of source code metrics, in: (Hrsg., 2012),
Nelson, K. und Santos, C. (2007), Attractiveness of Open Source Projects: A Path to Software Quality, in: AMCIS 2007 Proceedings, 2007
Nguyen, C.D., Marchetto, A. und Tonella, P. (2013), Automated oracles: an empirical study on cost and effectiveness, in: Meyer, Baresi, Mezini (Hrsg., 2013), S. 136
Nguyen-Duc, A., Cruzes, D.S. und Conradi, R. (2015), The impact of global dispersion on coordination, team performance and software quality – A systematic literature review, in: Information and Software Technology, 57, 2015, S. 277-294
Niemelä, E. und Immonen, A. (2007), Capturing quality requirements of product family architecture, in: Information and Software Technology, 49, 2007, 11-12, S. 1107-1120
Nikolik, B. (2012), Software quality assurance economics, in: Information and Software Technology, 54, 2012, 11, S. 1229-1238

Nikora, A.P. und Munson, J.C. (2006), Building high-quality software fault predictors, in: Software: Practice and Experience, 36, 2006, 9, S. 949-969
Notkin, D. (2013), 35th International Conference on Software Engineering (ICSE), 2013 - 18 - 26 May 2013, San Francisco, California, USA ; proceedings, Piscataway, NJ 2013
O. Melo, C. de, Santos, V., Katayama, E., Corbucci, H., Prikladnicki, R., Goldman, A. und Kon, F. (2013), The evolution of agile software development in Brazil, in: Journal of the Brazilian Computer Society, 19, 2013, 4, S. 523-552
Obele, B.O. und Kim, D. (3/31/2014 - 4/4/2014), On an Embedded Software Design Architecture for Improving the Testability of In-vehicle Multimedia Software, in: (Hrsg., 3/31/2014 - 4/4/2014), S. 349-352
Ogasawara, H., Yamada, A. und Kojo, M. (1996), Experiences of software quality management using metrics through the life-cycle, in: (Hrsg., 1996), S. 179-188
Okumoto, K. (1985), A Statistical Method for Software Quality Control, in: IEEE Transactions on Software Engineering, SE-11, 1985, 12, S. 1424-1430
Oliveira Neto, F.G. de, Torkar, R. und Machado, P.D.L. (5/16/2015 - 5/24/2015), An Initiative to Improve Reproducibility and Empirical Evaluation of Software Testing Techniques, in: (Hrsg., 5/16/2015 - 5/24/2015), S. 575-578
Parnas, D.L. und Lawford, M. (2003), The role of inspection in software quality assurance, in: IEEE Transactions on Software Engineering, 29, 2003, 8, S. 674-676
Paulk, M.C., Curtis, B., Chrissis, M.B. und Weber, C.V. (1993), Capability maturity model, version 1.1, in: IEEE Software, 10, 1993, 4, S. 18-27
Peischl, B. (2015), Software quality research: From processes to model-based techniques, in: (Hrsg., 2015), S. 1-6
Perez, J., Mens, T. und Kamseu, F. (2013), A Pilot Study on Software Quality Practices in Belgian Industry, in: Cleve (Hrsg., 2013), S. 395-398
Plant, R. (1991), Factors in software quality for knowledge-based systems, in: Information and Software Technology, 33, 1991, 7, S. 527-536
Poore, J.H. (1988), Derivation of local software quality metrics (software quality circles), in: Software: Practice and Experience, 18, 1988, 11, S. 1017-1027
Porter, A.A. und Johnson, P.M. (1997), Assessing software review meetings: results of a comparative analysis of two experimental studies, in: IEEE Transactions on Software Engineering, 23, 1997, 3, S. 129-145
Poth, A. und Sunyaev, A. (2014), Effective Quality Management: Value- and Risk-Based Software Quality Management, in: IEEE Software, 31, 2014, 6, S. 79-85
Prell, E.M. und Sheng, A.P. (1984), Building Quality and Productivity into a Large Software System, in: IEEE Software, 1, 1984, 3, S. 47-54
Procaccino, J.D., Verner, J.M., Darter, M.E. und Amadio, W.J. (2005), Toward Predicting Software Development Success from the Perspective of Practitioners: An Exploratory Bayesian Model, in: Journal of Information Technology, 20, 2005, 3, S. 187-200
Quah, T.-S. und Thwin, M.M.T. (2003), Application of neural networks for software quality prediction using object-oriented metrics, in: (Hrsg., 2003), S. 116-125
Radjenović, D., Heričko, M., Torkar, R. und Živkovič, A. (2013), Software fault prediction metrics: A systematic literature review, in: Information and Software Technology, 55, 2013, 8, S. 1397-1418
Raffo, D. und Harrison, W. (2000), Moving Toward CMM Levels 4 and 5: Combining Models and Metrics to Achieve Quantitative Process Management, in: AMCIS 2000 Proceedings, 2000
Rai, A., Song, H. und Troutt, M. (1998), Software quality assurance: An analytical survey and research prioritization, in: Journal of Systems and Software, 40, 1998, 1, S. 67-83
Ravichandran, T. und Rai, A. (2000), Quality Management in Systems Development: An Organizational System Perspective, in: MIS Quarterly, 24, 2000, 3, S. 381
Reformat, M., Pedrycz, W. und Pizzi, N.J. (2002), Software quality analysis with the use of computational intelligence, in: (Hrsg., 2002), S. 1156-1161
Rettig, M. (1990), Software Teams, in: Communications of the ACM, 33, 1990, S. 23-28
Reynoso, J.M.G. und Sandoval, Monica del Refugio Brizuela (2008), Improving Software Quality Through the Use of Statistics: An Initial Approach, 2008
Rojas, T., Perez, M., Grimán, A. und Mendoza, L.E. (2001), Decision Support System To Support Software Quality Through The Selection of Case Tools, in: AMCIS 2001 Proceedings, 2001
Rooijmans, J., Aerts, H. und van Genuchten, M. (1996), Software quality in consumer electronics products, in: IEEE Software, 13, 1996, 1, S. 55-64
Ruiz-Rube, I., Dodero, J.M. und Colomo-Palacios, R. (2015), A framework for software process deployment and evaluation, in: Information and Software Technology, 59, 2015, S. 205-221
Russell, B. und Chatterjee, S. (2003), Relationship quality, in: Communications of the ACM, 46, 2003, 8, S. 85-89
Russo, D., Ciancarini, P., Falasconi, T. und Tomasi, M. (2018), A Meta-Model for Information Systems Quality: A Mixed Study of the Financial Sector, in: ACM Transactions on Management Information Systems (TMIS), 9, 2018, 3, S. 11
Sacha, K. (2006), Evaluation of Expected Software Quality: A Customer's Viewpoint, in: Baresi, Heckel (Hrsg., 2006), S. 170-183
Sahraoui, H., Briand, L.C., Guéhéneuc, Y.-G. und Beaurepaire, O. (2010), Investigating the impact of a measurement program on software quality, in: Information and Software Technology, 52, 2010, 9, S. 923-933
Saraiva, J. (2013), A roadmap for software maintainability measurement, in: Notkin (Hrsg., 2013), S. 1453-1455

Savolain, J. und Mannisto, T. (2010), Conflict-Centric Software Architectural Views: Exposing Trade-Offs in Quality Requirements, in: IEEE Software, 27, 2010, 6, S. 33-37
Schäfer, W., Dwyer, M. B. und Gruhn, V. (2008), ACM/IEEE 30th International Conference on Software Engineering, 2008 - ICSE '08 ; 10 - 18 May 2008, Leipzig, Germany, Piscataway, NJ 2008
Schäfer, W., Dwyer, M. B. und Gruhn, V. (2008), ACM/IEEE 30th International Conference on Software Engineering, 2008 - ICSE '08 ; 10 - 18 May 2008, Leipzig, Germany, Piscataway, NJ 2008
Schamp, A. und Owens, H. (1997), Successfully implementing configuration management, in: IEEE Software, 14, 1997, 1, S. 98-101
Schmidt, C., Spohrer, K., Kude, T. und Heinzl, A. (2012), The Impact of Peer-Based Software Reviews on Team Performance: The Role of Feedback and Transactive Memory Systems, in: ICIS 2012 Proceedings, 2012
Schneidewind, N.F. (1997), Software metrics model for integrating quality control and prediction, in: (Hrsg., 1997), S. 402-415
Schneidewind, N.F. und Nikora, A.P. (Nov. 1999), Predicting deviations in software quality by using relative critical value deviation metrics, in: (Hrsg., Nov. 1999), S. 136-146
Schneidewind, N.F. (2000), in: Annals of Software Engineering, 9, 2000, 1/4, S. 79-101
Schrettner, L., Fülöp, L.J., Beszédes, Á., Kiss, Á. und Gyimóthy, T. (2012), Software Quality Model and Framework with Applications in Industrial Context, in: Mens (Hrsg., 2012), S. 453-456
Shi, Y., Li, M., Arndt, S. und Smidts, C. (2017), Metric-based software reliability prediction approach and its application, in: Empirical Software Engineering, 22, 2017, 4, S. 1579-1633
Shihab, E. (Hrsg., 2011, Pragmatic prioritization of software quality assurance efforts: ACM, http://dl.acm.org/ft_gateway.cfm?id=1986007&type=pdf
Shihab, E. (2014), Practical Software Quality Prediction, in: (Hrsg., 2014), S. 639-644
Shihab, E., Jiang, Z.M., Adams, B., Hassan, A.E. und Bowerman, R. (2011), Prioritizing the creation of unit tests in legacy software systems, in: Software: Practice and Experience, 41, 2011, 10, S. 1027-1048
Slaughter, S.A., Harter, D.E. und Krishnan, M.S. (1998), Evaluating the cost of software quality, in: Communications of the ACM, 41, 1998, 8, S. 67-73
Sneed, H.M. und Merey, A. (1985), Automated Software Quality Assurance, in: IEEE Transactions on Software Engineering, SE-11, 1985, 9, S. 909-916
Stamelos, I., Angelis, L., Oikonomou, A. und Bleris, G.L. (2002), Code quality analysis in open source software development, in: undefined, 2002
Steen, O. (2007), Practical knowledge and its importance for software product quality, in: Information and Software Technology, 49, 2007, 6, S. 625-636
Steidl, D., Deissenboeck, F., Poehlmann, M., Heinke, R. und Uhink-Mergenthaler, B. (2014), Continuous Software Quality Control in Practice, in: (Hrsg., 2014), S. 561-564
Stirewalt, K., Egyed, A. und Fischer, B. (2007), Proceedings of the twenty-second IEEEACM international conference on Automated software engineering, New York, NY 2007
Streit, J. und Pizka, M. (2011), Why software quality improvement fails, in: Taylor, Gall, Medvidović (Hrsg., 2011), S. 726
Stribrny, S. und Mackin, F.B. (2006), When Politics Overshadow Software Quality, in: IEEE Software, 23, 2006, 5, S. 72-73
Subramanian, G.H., Jiang, J.J. und Klein, G. (2007), Software quality and IS project performance improvements from software development process maturity and IS implementation strategies, in: Journal of Systems and Software, 80, 2007, 4, S. 616-627
Subramanyam, R., Weisstein, F.L. und Krishnan, M.S. (2010), User participation in software development projects, in: Communications of the ACM, 53, 2010, 3, S. 137
Szabo, R.M. und Khoshgoftaar, T.M. (1995), An assessment of software quality in a C++ environment, in: (Hrsg., 1995), S. 240-249
Takagi, Y., Tanaka, T., Niihara, N., Sakamoto, K., Kusumoto, S. und Kikuno, T. (1995), Analysis of review's effectiveness based on software metrics, in: (Hrsg., 1995), S. 34-39
Takahashi, R., Muraoka, Y. und Nakamura, Y. (1997), Building software quality classification trees: approach, experimentation, evaluation, in: (Hrsg., 1997), S. 222-233
Takahashi, R. (1997), Software quality classification model based on McCabe's complexity measure, in: Journal of Systems and Software, 38, 1997, 1, S. 61-69
Tanaka, T., Aizawa, M., Ogasawara, H. und Yamada, A. (1998), Software quality analysis and measurement service activity in the company, in: Torii (Hrsg., 1998), S. 426-429
Tarhan, A. und Giray, G. (2017), On the Use of Ontologies in Software Process Assessment, in: Mendes, Counsell, Petersen (Hrsg., 2017), S. 2-11
Tarvo, A. (2009), Mining Software History to Improve Software Maintenance Quality: A Case Study, in: IEEE Software, 26, 2009, 1, S. 34-40
Taylor, R. N., Gall, H. und Medvidović, N. (2011), 33rd International Conference on Software Engineering (ICSE), 2011 - 21 - 28 May 2011, Waikiki, Honolulu, HI, USA, Piscataway, NJ 2011
Torii, K. (1998), Proceedings of the 20th international conference on Software engineering, Washington, DC 1998
Torii, K. (1998), Proceedings of the 20th international conference on Software engineering, Washington, DC 1998
Trammell, C.J. und Poore, J.H. (1992), A group process for defining local software quality: Field applications and validation experiments, in: Software: Practice and Experience, 22, 1992, 8, S. 603-636
Tymchuk, Y. (2015), Treating software quality as a first-class entity, in: Koschke, Krinke, Robillard (Hrsg., 2015), S. 594-597

van Rompaey, B., Du Bois, B., Demeyer, S., Pleunis, J., Putman, R., Meijfroidt, K., Duenas, J.C. und Garcia, B. (2009), SERIOUS: Software Evolution, Refactoring, Improvement of Operational and Usable Systems, in: Ferenc (Hrsg., 2009), S. 277-280
Vitharana, P. und Mone, M.A. (2008), Measuring Critical Factors of Software Quality Management, in: Information Resources Management Journal, 21, 2008, 2, S. 18-37
Vivanco, R. (2007), Improving Predictive Models of Software Quality Using an Evolutionary Computational Approach, in: (Hrsg., 2007), S. 503-504
Voas, J. (1999), Software quality's eight greatest myths, in: IEEE Software, 16, 1999, 5, S. 118-120
Voas, J. (2000), Can Chaotic Methods Improve Software Quality Predictions?, in: IEEE Software, 17, 2000, 5, S. 20-22
Walia, G.S. und Carver, J.C. (2009), A systematic literature review to identify and classify software requirement errors, in: Information and Software Technology, 51, 2009, 7, S. 1087-1109
Wallmüller, E. (2011), Software quality engineering - Ein Leitfaden für bessere Software-Qualität ; [basiert auf den standardisierten Wissenssammlungen für Software-Qualität von ASQ und JSQC ; ideal für die Vorbereitung auf die CSQ- und QAMP-Zertifizierung, 3. Auflage, München 2011
Wallmüller, E. (2011), Software Quality Engineering - Ein Leitfaden für bessere Software-Qualität ; [basiert auf den standardisierten Wissenssammlungen für Software-Qualität von ASQ und JSQC ; ideal für die Vorbereitung auf die CSQ- und QAMP-Zertifizierung, 3. Aufl., München 2011
Wang, H. und Wang, C. (2003), Taxonomy of security considerations and software quality, in: Communications of the ACM, 46, 2003, 6, S. 75-78
Wang, Q., Zhu, J. und Yu, B. (2007), Feature Selection and Clustering in Software Quality Prediction, in: EASE, 2007
Weerahandi, S. und Hausman, R.E. (1994), Software quality measurement based on fault-detection data, in: IEEE Transactions on Software Engineering, 20, 1994, 9, S. 665-676
Weiss, D.M., Bennett, D., Payseur, J.Y., Tendick, P. und Zhang, P. (Hrsg., 2002, Goal-oriented software assessment: ACM, http://dl.acm.org/ft_gateway.cfm?id=581369&type=pdf
Westermann, D., Happe, J., Krebs, R. und Farahbod, R. (2012), Automated inference of goal-oriented performance prediction functions, in: Goedicke, Menzies, Saeki (Hrsg., 2012), S. 190
Wu, W., Cai, Y., Kazman, R., Mo, R., Liu, Z., Chen, R., Ge, Y., Liu, W. und Zhang, J. (2018), Software Architecture Measurement—Experiences from a Multinational Company, in: Cuesta, Garlan, Pérez (Hrsg., 2018), S. 303-319
Xenos, M. und Christodoulakis, D. (1997), Measuring perceived software quality, in: Information and Software Technology, 39, 1997, 6, S. 417-424
Xing, F., Guo, P. und Lyu, M.R. (2005), A Novel Method for Early Software Quality Prediction Based on Support Vector Machine, in: (Hrsg., 2005), S. 213-222
Yan, M., Zhang, X., Liu, C., Zou, J., Xu, L. und Xia, X. (2017), Learning to Aggregate: An Automated Aggregation Method for Software Quality Model, in: (Hrsg., 2017), S. 268-270
Yang, Onita, Zhang und Dhaliwal (2011), TESTQUAL: Conceptualizing Software Testing as a Service, in: e-Service Journal, 7, 2011, 2, S. 46
Yau, S.S. und Collofello, J.S. (1980), Some Stability Measures for Software Maintenance, in: IEEE Transactions on Software Engineering, SE-6, 1980, 6, S. 545-552
Zhang, H. und Cheung, S.C. (2013), A cost-effectiveness criterion for applying software defect prediction models, in: Meyer, Baresi, Mezini (Hrsg., 2013), S. 643
Zhang, H. und Kim, S. (2010), Monitoring Software Quality Evolution for Defects, in: IEEE Software, 27, 2010, 4, S. 58-64
Zhou, Z.Q., Xiang, S. und Chen, T.Y. (2016), Metamorphic Testing for Software Quality Assessment: A Study of Search Engines, in: IEEE Transactions on Software Engineering, 42, 2016, 3, S. 264-284

Artikel der Praxiszeitschriften

Prüfzeichen für Software? (1976). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__D4C837749864A44D836AB76E59EB2AE4.
Sneed, Harry (1977): Methodik allein genügt nicht. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__7759ADB7A6A1895CD841CCB512CDEA21.
Grochla, Erwin (1978): Brisante Mängelliste. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__0B9F0F148C30E103C9A457E9A43BE164.
Software auf dem Prüfstand. Ausgetauscht werden (1979). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__B42465EFB02C2DA204AE9C3ECD2BE77F.

Structo 79 (1979). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__A2622BCED59B6F8BC41995EE1925193B.
Zur Förderung der Softwarequalität möchte die Software Research Associated. Normen und Konventionen beschäftigt sind. Ziel der Gemeinschaft soll sein (1979). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__FA799D11DAD7BAF08BECAEFB504A89BF.
Klaus, Heinzgünther (1979): Software-Qualität durch Kooperation. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__4DAD70721D177C019C9330022F2698E5.
Konferenz über Sofware-Qualitätskontrolle (1980). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__C913DF53070AB095671E84E96379EA6D.
Software-Qualitätskontrolle (1980). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__A69FFA9A23737891FECCCE7AEE46F126.
Hausen; Müllerburg; Züllighoven (1981): Software-Produktion. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__8F346FE6A6DD961BC800BF3D758101C4.
German Chapter of the ACM (1982). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__FD58C071F782996997C25EEBE0426826.
Schneiter, Werner (1982): Gegen die Software-Lücke in den 80er Jahren. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__01EAB0C2F2BDAFF4C303591FB9EC3B3C.
Bei SW-Metrik kommen auch Fachleute ins Schleudern (1983). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__5688FA8B5EB4A1B4AAFE648F78D9FA86.
Konventionen können die Erfahrung des Systementwicklers nicht ersetzen (1983). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__6EDF1766743F5171337A032A125F9D42.
Standard-Programme: "TÜV-Plakette" für Software erwünscht (1983). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__5A0545C204C00726A4ACFE614FA4D107.
Zu oft bleibt die Qualitätssicherung auf der Strecke (1983). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__71EF69209EB81CC208D122CC2870E82E.
DV-Service: Mehr als nur Verfügbarkeit (1984). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__21BB3F29A31CF63B107FE85EC12751DC.
Ahlers, Jörg; Emmerich, Wilfried (1984): QS erfordert konstruktive und analytische Maßnahmen. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__F962F8DF4B2A1D6661F013F69D6FDCBE.
Andreas Haase (1984): Software-Qualitätssicherung. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__A7D401904E40865535584EE26F8D0DDB.
Merbeth, Günther; Abbenhardt, Helmut (1984): Qualitätssicherung - der Glaube allein genügt nicht. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__C4741B802F1CDBE2E8DA0F75AF911296.
Walter Wintersteiger (1984): QS-Maßnahmen nur zäh zu realisieren. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__147BC73ED477CE87D322DECCC0F42769.
Anwender müssen Verantwortung für SW-Qualität mittragen. Datenbankbegriffe sowie aller Testtransaktionen kan (1985). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__FB3A6D667D068996AD5A04BC63E18A73.
Gütezeichen für Software gerät ins Kreuzfeuer der Kritik. Fachausstellung FE-Lösungen verschiedener Anbieter kennenzulernen (1985). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__A7EB8994C8506C17AC4BFF7EFB5CDF00.
Qualitätsaspekte in der Diskussion (1985). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__2CE2320FF104B3450590B74B544E4D52.
Qualitätssicherung in der Praxis (1985). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__668DD4169A2EFF319BEA8E5006C67C67.
Qualitätssicherung in der Praxis noch mit Organisations-Problemen. Konsequenter Ergebnis-Check spart bare MünzeDie absolute Voraussetzung (1985). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__D98A969A5F0BE4F6A29D7BE68E78D9F4.
Unter dem Thema "Software-Qualitätssicherung - Vorstellung und. Universität zu Köln (Bifoa) am 21. und 22. November 1985 in Köln ein (1985). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__26D0D715D2ACEA33773DEB838DBA26CB.
Andreas Haase (1985): Schlechtes Image erschwert die Arbeit der QS-Abteilung. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__7F1C96325241C93E4C17F7787D16BE54.
Rudolf van Megen, Heiner Diesteldorf (1985): Der Testbestand als wesentlicher Teil der ingenieurmäßigen SW-Produktion. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__6F5AE7B5ED40054E23E4D6C0DD9EAD9B.
Gute Software sollte vor allem auch rechenzentrumsfreundlich sein (1986). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__12FD9842A8228426ABF65745C650AE2A.
Schlamperei bei der Entwicklung kann sich böse rächen (1987). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__AB810FF70D75062E5DEFAB6724E967F4.
Heinz Bons, Rudolf van Megen (1987): Kooperation zwischen Entwicklern und Usern läßt in der Praxis zu wunschen. SW-Eingangskontrolle muß viele Hürden nehmen. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__60883884B87296A040693264B0673C5A.
Rolf Breitsprecher, Hans Meintzschel, Reinhard Pöhlmann (1987): "Wie handhaben Sie in Ihrem Unternehmen die Software-Qualitäts-Sicherung?". In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__96B525EE5FCE07C7125375F18E862D60.

Wintersteiger (1987): Halbherziger Ansatz bringt oft mehr Schaden als Nutzen. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__05ACD8DA45A457E057BF0E07B225EB47.
Hohe Verluste durch mangelhafte Software (1988). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__A20879AF40ACA04647DDD15FA4C842E8.
Dieter Eckbauer (1988): Software-Qualität. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__33D6DECE0C88B32D0AFB374A1EC46ADA.
Müller-Ettrich, Gunter (1988): Die Manager der Unternehmen müssen noch viel dazulernen, denn. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__8975C38C822244FE5738A2079E734821.
Wolfgang Grandel (1988): Stichwort "Integrierte Projektunterstützungs-Umgebung" (Teil 4). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__2E5376709AA82750528404E74FA3CE4E.
Die alten Mythen des Software-Engineering leben noch (Teil 1) (1989). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__0C8F4EF4DF277B0612ADB382A5DD95FA.
Die alten Mythen des Software-Engineering leben noch (Teil 2) (1989). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__33D8A9FEC327C33C77722FB82E6BBDB7.
Die alten Mythen des Software-Engineering leben noch (Teil 3 und Schluß) (1989). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__1A4F74C11732233C5894FF17C8BE2AF0.
Qualitätsgeprüfte Software erhält Gütesiegel (1989). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__E10BACDE734B2B39108832E31CD17E0E.
Softwarequalität im europäischen Rahmen (1989). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__7F7363D49CC1CE7E3D48E7BEAC22E8C7.
Sneed, Harry (1989): Mangelhaftes Konzept der Qualitätssicherung erzeugt unnötigen Aufwand bei. Redundanter Code bläst Wartungskosten auf. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__44BDBD0F65945BA711C2A29881B4C6B1.
Sneed, Harry (1989): Mangelhaftes Konzept der Qualitätssicherung erzeugt unnötigen Aufwand bei. Redundanter Code bläst Wartungskosten auf. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__BC64EF3282359A836323F807BF0F9091.
Konferenz über die Qualität von Software (1990). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__861C5865E4BE5075B5C043E6308203CA.
Knöll; Schäfer (1990): Phasenmodelle des Software-Entwicklungsprozesses. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__5B082CB65E943FC3722AFCF15818E48C.
Sicherung der Software-Qualität (1991). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__50B7A7D2A950356821B6A3669D5D4A7A.
Sicherung der Software-Qualität (1991). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__05EF003F8AB6AD363FEFAC27AA376539.
Abel, Michael (1991): Unverzichtbares Instrument für Systementwickler. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__3B2FBF1BF311113EBFA9B33153890A86.
Auto-Test (1992). In: *c't - Magazin für Computertechnik* (7), S. 62. Online verfügbar unter https://www.wiso-net.de/document/CT__1992070623.
Die Kosten der Technik sind schneller gestiegen als die Produktivität (1992). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__723514AC0437576D649531FB41A7E614.
Burghartz, Dieter (1994): Qualitaetsmanagement in der Softwareentwicklung. In: *Markt & Technik* (46), S. 47. Online verfügbar unter https://www.wiso-net.de/document/MT__MT199400002630102118291014292822102310161422142329.
Stuerze, Marion (1994): Die Sicherung der SW-Qualitaet nach ISO 9000. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__7FD6AB1B141814F8CC0A249139BE437C.
Stuerze, Marion (1994): Internationale Tagung ueber Softwaretests. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__2E26344854A406154F0F4562C5917158.
Stuerze, Marion (1994): SW-Qualitaet und Management von Dokumenten. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__4D7B82A28C8B5C28CB36AA58D47CADA0.
Stuerze, Marion (1995): Anwender berichten ueber ihre Erfahrungen. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__918F854ABD9F3F969CDBFB813A1AC107.
Stuerze, Marion (1995): Schulungen ueber SW-Qualitaet von der EU gefoerdert. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__C8DD6BF442C21C88D4985465751F95B3.
Stuerze, Marion (1995): Systeme zur Sicherung der SW-Qualitaet. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__9157E317A8035086053600D3C570C134.
Litzba, Ulrike (1996): Thema der Woche. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__E62247178E9B23D3B1F3F39849D7C653.
Stuerze, Marion (1996): Die SW-Qualitaet verbessern durch Inspektionen. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__210D598802179C66B11ED49177D0710B.
Stuerze, Marion (1996): Experten geben Tips zur Qualitaetssicherung. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__9627D7E1AD9296FD29E1A8D4010F1E42.
Assessment fuer den Mittelstand (1997). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__AF0C7A89D15F63701336AE60174E8F15.
Schmitz, Ludger (1997): Software-Optimierung/"A fool with a tool is still a fool". In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__0E85ACB3BCE6F185921BAA54783EB559.
Seidel, Bernd (1997): Bootstrap und Spice ergaenzen ISO-Norm. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__7C5EC96DAC4D9D58A305BF3D0F54CB32.

Stuerze, Marion (1997): Kurse zur Sicherung der SW-Qualitaet. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__E13E507CF76A805BC52EFBEEBF48A8E0.
Software Engineering (1998). In: *c't - Magazin für Computertechnik* (19), S. 166. Online verfügbar unter https://www.wiso-net.de/document/CT__1998192650.
Stürze, Marion (1998): Ausbildung zum Spezialisten für SW-Qualität. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__5E3DFA751BF5C41F989A1AF6F2E7710D.
Stürze, Marion (1998): Kommunikative Fähigkeiten sind gefragt. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__CAAD4603451AF44B319D18D02E7D23D5.
Stürze, Marion (1998): Schulung zur Sicherung der SW-Qualität. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__9ABA48CB202E67697B96BA49E54DDD1A.
Quack, Karin (1999): Projektplanung und Softwarequalität/Unternehmen sehen Softwarequalität als strategische Aufgabe. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__2B774AFE8AA97EF2DB7F0B1AD456AF73.
SQS (2000). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__2477754E87DB6BE4BD4AAFF24CAFB773.
Tests für den Softwaretest (2000). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__B584D81EE3BD211E4C50D053FBD2132B.
Scholz, Schmietendorf (2000): Performance Engineering - Aufgaben und Inhalte. In: *HMD Praxis der Wirtschaftsinformatik* (213), S. 112–119. Online verfügbar unter https://www.wiso-net.de/document/HMD__200005035.
Methodische Vorgehensweisen in der Embedded-Softwareentwicklung (2002). In: *Markt & Technik* (43), S. 30. Online verfügbar unter https://www.wiso-net.de/document/MT__MT20021018221429172413182812171431242716141714232 8.
Mücke, Volker (2002): Die sieben wichtigsten Erfolgsfaktoren. In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__4E9328A47B150ED6827001C83743B812.
Qualitätssicherung in der Entwicklung (2003). In: *it&t-business* (6/03), S. 28. Online verfügbar unter https://www.wiso-net.de/document/ITT__ITT_200306130238460052.
Qualitätsstandards für mehr Sicherheit (2003). In: *c't - Magazin für Computertechnik* (24), S. 56. Online verfügbar unter https://www.wiso-net.de/document/CT__2003243656.
Sieben Tools für das Qualitäts-Management (2003). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__94618812A11B027F67E6C33BA4CC7E27.
Müller, Volker (2003): Prozessorientiertes IT-Qualitätsmanagement. In: *HMD Praxis der Wirtschaftsinformatik* (232), S. 66–78. Online verfügbar unter https://www.wiso-net.de/document/HMD__200307048.
6. Automation Day des ASQF (2004). In: *Markt & Technik* (26), S. 10. Online verfügbar unter https://www.wiso-net.de/document/MT__MT20040625103029242210291824231310341314281028 2615.
Herausforderung 4 (2004). In: *Markt & Technik* (36), S. 20. Online verfügbar unter https://www.wiso-net.de/document/MT__MT20040903171427103028152427131427302316282415 2932.
UNTERSUCHUNG ZUR SOFTWAREQUALITÄT (2004). In: *IT Business* (13), S. 13. Online verfügbar unter https://www.wiso-net.de/document/ITB__2004130574.
Thiel, Christoph (2004): Ein Reifegradmodell für das IT-Sicherheitsmanagement. In: *HMD Praxis der Wirtschaftsinformatik* (236), S. 52–58. Online verfügbar unter https://www.wiso-net.de/document/HMD__200403018.
Berater müssen organisieren können (2005). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__E7AD389E83EE43EABD8850BE37F49DEA.
Effizientes Testmanagement (2005). In: *it&t-business* (3/05), S. 28. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520050304010728005 3.
Mehr Erfolg in IT-Projekten (2005). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__891F1ECE1389179B1DAC2D6B7E16D588.
Software - Qualität ist gefragt (2005). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__54EDE0BE1C70D62537DF81E40B5584F2.
Dietzsch, Andreas (2005): Prozessmodellkennzahlen zur Unterstützung des Prozessmanagements. In: *HMD Praxis der Wirtschaftsinformatik* (241), S. 55–66. Online verfügbar unter https://www.wiso-net.de/document/HMD__200501008.
Die Vision von Software ohne Bugs (2006). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__C2465D82FE150184BC9B61F2208FE01E.
SOA erschwert Qualitätssicherung (2006). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__0665C29F2F1B83358B6501407C6FFBE7.
Bartram, Axel (2006): Outsourcing der Software-Entwicklung: Risiken erkennen und minimieren. In: *IM Information Management & Consulting* (3), S. 61–66. Online verfügbar unter https://www.wiso-net.de/document/IMC__IMC20060828612430292824302712182.
Bienstock, Serafima (2006): Software auf Qualität abklopfen. In: *IM Information Management & Consulting* (1), S. 57–64. Online verfügbar unter https://www.wiso-net.de/document/IMC__IMC20060306572824152932102714103.
Schlatter, Anton (2006): Methodisches Testen als Grundlage für Qualitätssicherung. In: *IM Information Management & Consulting* (1), S. 53–56. Online verfügbar unter https://www.wiso-net.de/document/IMC__IMC20060306532214291724131828121.
Code Quality (2007). In: *c't - Magazin für Computertechnik* (8), S. 204. Online verfügbar unter https://www.wiso-net.de/document/CT__2007081133.

Neue Herausforderungen für Embedded-Entwickler (2007). In: *Markt & Technik* (24), S. 28. Online verfügbar unter https://www.wiso-net.de/document/MT__MT060715035.
Qualitätssicherung entlastet Projektbudgets (2007). In: *Markt & Technik* (24), S. 32. Online verfügbar unter https://www.wiso-net.de/document/MT__MT060715038.
Software mit Usern testen (2007). In: *it&t-business* (10/07), S. 20. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520071001195003040.
Software-Qualität (2007). In: *Markt & Technik* (40), S. 90. Online verfügbar unter https://www.wiso-net.de/document/MT__MT100705015.
Unternehmens- und IT-Prozesse im Fokus (2007). In: *COMPUTERWOCHE*. Online verfügbar unter https://www.wiso-net.de/document/CW__2007032938420438213811434084.
Pizka, Markus (2007): Web-basierte und andere junge Legacy-Systeme. In: *IM Information Management & Consulting* (2), S. 60–68. Online verfügbar unter https://www.wiso-net.de/document/IMC__IMC2007082960321411111010281814272.
Kaufmann, Ronald (2008): Vom Kostenfaktor zum Erfolgsgaranten. In: *COMPUTERWOCHE* (20), S. 16–17. Online verfügbar unter https://www.wiso-net.de/document/CW__2008051682361035601481203277.
Komplettlösungen für mehr Softwarequalität (2009). In: *it&t-business* (06), S. 22. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520090601123800042.
Software Quality Days 2010 expandieren (2009). In: *it&t-business* (12), S. 13. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520091211203027002 2.
Kaufmann, Ronald (2009): Vom Kostenfaktor zum Erfolgsgaranten. In: *it&t-business* (11), S. 24. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520091101141441004 9.
Basis für Zuverlässigkeit und Sicherheit (2010). In: *Markt & Technik* (47), S. 31. Online verfügbar unter https://www.wiso-net.de/document/MT__MT111019027.
Dritte Auflage des Qualitätskongresses (2010). In: *it&t-business* (12), S. 12. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520101201192131006 7.
Erfolgreich durch Testen (2010). In: *it&t-business* (11), S. 16. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520101101155205003 9.
Software Quality Days 2011 (2010). In: *it&t-business* (12), S. 4. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520101201192131000 4.
Treffpunkt der Software-Entwicklungs-Community (2010). In: *it&t-business* (02), S. 11. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520100201193410002 9.
Wieland, Sabine (2010): Dynamische SOA-Orchestrierung. In: *IM Information Management & Consulting* (04), S. 38–41. Online verfügbar unter https://www.wiso-net.de/document/IMC__551A3F93828A5293BDAB8FADD3F41880.
Die Codeanalyse allein reicht nicht (2011). In: *COMPUTERWOCHE* (33). Online verfügbar unter https://www.wiso-net.de/document/CW__2011081654254821600240138402.
Qualität beginnt im Kopf (2011). In: *it&t-business* (11), S. 21. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520111111114223700 51.
Software Quality Days 2012 (2011). In: *it&t-business* (12), S. 5. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520111212121464400 13.
Softwaretests werden professioneller (2011). In: *it&t-business* (10), S. 34. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520111100712425700 36.
Anecon (2011): Ergebnisse der Umfrage 2011: "Softwaretest in der Praxis“. In: *it&t-business* (10), S. 11. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520111100712425700 16.
Das unterschätzte Sicherheitsrisiko (2012). In: *COMPUTERWOCHE* (09). Online verfügbar unter https://www.wiso-net.de/document/CW__2012022701181274131841182364.
Gestiegene Komplexität beherrschen (2012). In: *Markt & Technik* (12), S. 22. Online verfügbar unter https://www.wiso-net.de/document/MT__MT031223027.
IT-Markt (2012). In: *COMPUTERWOCHE* (06). Online verfügbar unter https://www.wiso-net.de/document/CW__2012020634588824148340152500.
Optimal getestet an den Start (2012). In: *it&t-business* (12), S. 24. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520121218170142003 7.
Qualität - Investition in die Zukunft (2012). In: *it&t-business* (09), S. 31. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520120903161858004 8.
Software Quality Days 2013 (2012). In: *it&t-business* (11), S. 6. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520121109180801001 1.
Trend zu Managed Services (2012). In: *it&t-business* (07-08), S. 31. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520120706125543005 7.
Veselko, Klaus (2012): Nachhaltig erfolgreich. In: *it&t-business* (12), S. 25. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520121218170142003 9.
Den Wert von Qualität erleben (2013). In: *it&t-business* (09), S. 29. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520130903170248004 9.
Neue Modelle für Softwarequalität (2013). In: *it&t-business* (10), S. 9. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520131004694936066 0680666.
Software-Quality-Days 2013 (2013). In: *it&t-business* (02), S. 5. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520130215183523000 9.
Software-Tests in der Wolke (2013). In: *it&t-business* (06), S. 36. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520130607110825004 8.

Feßl, Stefan (2013): Die mobile-App-Falle. In: *it&t-business* (07-08), S. 31. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520130712153355005 1.
Heinen, Thomas (2013): App-sichern: eine effektive Mobility- und Security-Strategie. In: *IT Business* (003). Online verfügbar unter https://www.wiso-net.de/document/ITB__370993800.
Nicolas Zeitler (2013): Kosten und Support. In: *CIO - IT-Strategie für Manager*. Online verfügbar unter https://www.wiso-net.de/document/CIOD__2905034.
ÖBV sichert Software-Qualität (2014). In: *it&t-business* (10), S. 32. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520141003806 80 6539406963067.
Softwarequalität im Mittelpunkt (2014). In: *it&t-business* (11), S. 5. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520141107506 88 4606690671068.
Jürgen Egeling (2014): Scrum: Weniger Komplexität, mehr partnerschaftliche Zusammenarbeit. In: *wissensmanagement* (3), S. 38–39. Online verfügbar unter https://www.wiso-net.de/document/WIM__F8CA500158CC98457C99450ACD6D57EA.
Sneed, Harry (2014): Werterhaltung konsolidierter Informationssysteme. In: *HMD Praxis der Wirtschaftsinformatik* 51 (2), S. 175–187. Online verfügbar unter https://www.wiso-net.de/document/HMDS__101365s40702-014-0019-y.
Embedded World: Im Zeichen des Internet der Dinge (2015). In: *c't - Magazin für Computertechnik* (5), S. 50. Online verfügbar unter https://www.wiso-net.de/document/CT__1424381762240886.
Internationaler Treffpunkt der Software-Entwicklung (2015). In: *it&t-business* (02-03), S. 5. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520150224430846706973.
Geck, Bertram (2015): Übersicht zu den aktuellen Testing-Trends. In: *CIO - IT-Strategie für Manager*.
Architektur- und Codeprüfung von Software (2016). In: *Markt & Technik* (49), S. 64. Online verfügbar unter https://www.wiso-net.de/document/MT__MT121602055.
Smarte Zusammenkunft heimischer IT-Führungskräfte (2016). In: *it&t-business* (04), S. 10. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520160401206 60 67706506906806513.
Tools für mehr funktionale Sicherheit (2016). In: *Markt & Technik* (26), S. 52. Online verfügbar unter https://www.wiso-net.de/document/MT__MT061624061.
Schrauzer, Simon (2016): Computerisierung in der globalen Softwareentwicklung - Eine arbeitsmethodische Betrachtung. In: *HMD Praxis der Wirtschaftsinformatik* 53 (1), S. 42–54. Online verfügbar unter https://www.wiso-net.de/document/HMDS__101365s40702-015-0194-5.
SQS: Pothen in die Geschäftsführung (2017). In: *it&t-business* (04), S. 10. Online verfügbar unter https://www.wiso-net.de/document/ITT__07308408409520170331260 65 65069400683.
Statische Code-Analyse (2017). In: *Markt & Technik* (44), S. 46. Online verfügbar unter https://www.wiso-net.de/document/MT__MT101727024.
Söllner, Dierk (2017): DevOps by ScrumbanDevOps by Scrumban. In: *HMD Praxis der Wirtschaftsinformatik* 54 (2), S. 251–260. Online verfügbar unter https://www.wiso-net.de/document/HMDS__101365s40702-017-0301-x.
Herausforderung Embedded Devices (2018). In: *Markt & Technik* (32), S. 18. Online verfügbar unter https://www.wiso-net.de/document/MT__MT081803019.
Aus Assystem Technologies wird Expleo: Die Software Quality Systems AG (SQS) gehört zum Verbund (2019). In: *COMPUTERWOCHE* (08). Online verfügbar unter https://www.wiso-net.de/document/CW__20190218803146844086308221 38.
Sichere Software für Embedded-Systeme (2019). In: *Markt & Technik* (08), S. 42. Online verfügbar unter https://www.wiso-net.de/document/MT__MT021922022.

Stellenanzeigen

Stellenanzeige Name	Aufgaben	Anforderungen	Branche	Abschluss
(Senior/Junior) Qualitätsmanager (m/w/d) Softwareentwicklung	Konzeption sowie Durchführung von Testfällen in Servity, Regressionstests und explorative Tests, Test-Methoden, bewertet und weiterentwickelt, agilen Entwicklung	JUnit und Continuous Integration, Jenkins, Perfektionismus, Test-Kenntnisse und praktischen Erfahrungen	IT	abgeschlossenes Studium im Bereich der Informatik oder Wirtschaftsinformatik
Software Quality Engineer (m/w/d)	Spezifikation und Umsetzung von Softwaretests, Etablierung und Durchführung von Test-Strategien, Entwicklungsbegleitende und abschließende Überprüfung von Softwareprodukten, Kontinuierliche Verbesserung unserer existierenden Prozesse, Tools und Produkte	sehr gute Kenntnisse über Qualitätssicherung bzw. Software Testing, PowerShell und/oder Python, Umgang mit Werkzeugen zur Testautomatisierung (z.B. TestComplete, Selenium), Umgang mit API-Testwerkzeugen (z.B. Postman, Swagger UI), Kenntnisse in SW-Entwicklung C#, Programmierung mit Web-Technologien (ASP.NET MVC, HTML, JavaScript, CSS), CI/CD-Werkzeugen (z.B. Jenkins), Erfahrung im Testen von Microservices	IT	Ausbildung (beruflich oder akademisch)
Quality Assurance Manager (m/w/d) Mobilfunk	Etablierung QA-Managements für Testautomation, Teststrategie und Testlösung, Webanwendungen (Desktop, Tablet, Mobile), nativer Android- und iOS-App und weiterer Software mit manuellen und automatisierten Testmethoden, Testpläne, Automatisierung von Testfällen, Testergebnisse, Kontinuierliche Kommunikation, Problemlösung im Live-Betrieb	Sehr hohes Qualitätsbewusstsein, Interesse an der Automatisierung von Testabläufen und Lösung von Problemstellungen sowie Spaß an der akribischen Fehlersuche, Gutes Verständnis von und Interesse an Web-Technologien, Erfahrung im strukturierten Testaufbau und Testdurchführung, bei der Anwendung gängiger Testmethoden sowie automatisierter Tests (z.B. node.js oder Selenium), Eigeninitiative, Durchsetzungskraft und eine strukturierte sowie gewissenhafte Arbeitsweise, Schnelle Auffassungsgabe, Hands-on Mentalität, Teamfähigkeit, Sicherer Umgang mit MS Office	Mobilfunk	Abgeschlossenes analytisches / technisches Studium, Ausbildung mit mehrjähriger Berufserfahrung

Software Quality Assurance Manager (m/w/d)	Auditierung der hauseigenen Entwicklungen als auch für die Qualitätssicherung bei externen Dienstleistern verantwortlich, Konzeption und Umsetzung einer Lieferantenstrategie, Bewertung der Lieferantenfähigkeit, FMEA- bzw. FTA-Moderation, Schulungen zur Anwendung von QM-Methoden	langjährige, fundierte Berufspraxis, solides QM-Wissen und Erfahrung in der Weiterentwicklung von QM-Systemen, ISO 26262, Automotive SPICE, IATF 16949, kommunizieren souverän in englischer Sprache, analytisch-konzeptionellen Stärken, Planungs-, Organisations- und Koordinationstalent	Personalberatung	Hochschulabschluss in Informatik
Quality Assurance Engineer Online Marketing (w/m/divers) | Java	agile Softwareentwicklung und aktuelle Java- und Web-Technologien im Cloud Umfeld,Testautomatisierung, Performance-, Integrations- und Blackbox Tests, IMpulsgeber	Java, fundierte Erfahrung im Testen, hohes Qualitätsbewusstsein, ISTQB Zertifizierung, AWS Knowledge, MongoDB Erfahrung	Versandhandel	k.A.
Qualitätsmanager Software (w/m/x)	360-Grad Qualitätsstrategie für Software, ZUsammenarbeit Qualitätsmanagern externe Lieferanten, Bewertung Softwareprozesskompetenz, Einleitung Korrekturmaßnahmen, Risikoabschätzung, hohen Spielraum	Fundierte Erfahrung in Projektarbeit (agil & V-Modell). Hohes Qualitätsverständnis, ISO 9001 und 26262, Praktische Erfahrung, Verhandlungssichere Deutsch- und Englischkenntnisse	Automobil	Studienabschluss in (Wirtschafts-) Informatik, Softwareengineering, Elektrotechnik, Computer Science, Ingenieursstudiengang mit zusätzlichem IT-Background
Qualitätsmanager (m/w/div.) Digital Solutions	Einführung Qualitätsstandards Schwerpunkt Software-Produkte, TQM, KVP, Qualitätsbefragungen, Überwachung von Review-Meetings, Dokumentationsstandards, QM-Auditierung, Fehlerverfolgung	5 Jahre Berufserfahrung, Kenntnisse Software-Entwicklungs- und Projektmanagementprozesse (Scrum, SaFe, Requirements-Engineering) Nachweisbarer QM-Erfolg, DIN EN ISO 9001, Ausbildung zum internen Auditor, Flexibilität, hohe Einsatzbereitschaft, Fließende Deutsch- sowie Englischkenntnisse	IT	Abgeschlossenes Studium oder eine entsprechende Ausbildung im Bereich Informatik, Betriebswirtschaftslehre oder im technischen Bereich
(Senior) Quality Amplifier (m/w/d)	Teststrategie, Agile Softwareentwicklung, (Micro-)Services-Umgebung, Open Source Software wie Java, Spring, Apache Kafka, Hibernate, React, Puppet	Hands-on-Mentalität, automatisierte Tests mit Cucumber, Java und Selenium, Testfällen, Erfahrung in der Softwareentwicklung, analytische Fähigkeiten, Überblick, Teamplayer mit agilem Mindset	IT	k.A.

Senior Software Quality Engineer (m/w/d) Softwaretests / Systemtests	ZUsammenarbeit Fachabteilungen & IT, Testdurchführung, Testautomatisierung, Betreuung Automatisierungs- und Testtools, Automatisierungs-Frameworks	Mehrjährige Berufspraxis, Testautomatisierung, Testspezifikation, Testkonzepten, Testdurchführung, Softwareentwicklung und Programmierung, Analytischer Weitblick, Kommunikation, Teamgeist und Eigeninitiative	Energie	Abgeschlossenes Studium der Wirtschaftswissenschaften oder Informatik bzw. eine adäquate Ausbildung in einer vergleichbaren Fachrichtung
Qualitäts- und Testmanager (m/w/d) Softwareprojekte	Durchführung des SW-QS-Prozesses, automatisierte Softwaretests, Modul- und Integrationstests, Testspezifikationen, Testdurchführung, klassische und agile Vorgehensmodelle	Praktische Erfahrung, ISTQB-Zertifizierung, Strukturierte und präzise Arbeitsweise, Professionalität und Selbstständigkeit, Kommunikationsfähigkeiten, Kreativität und Teamgeist	IT	Hochschulausbildung informationswissenschaftlichen / betriebswirtschaftlichen Studiengang, vergleichbare Ausbildung
Software Entwickler im Bereich Qualitätssicherung (w/m/d)	Scrum-Teams,C#, Test-Framework, automatisierte Tests, Beratung während Softwareentwicklungsprozesses	.NET-Framework, Methoden der Software-Qualitätssicherung, analytischen Fähigkeiten, selbständig, Gewissenhaftigkeit, Teamplayer, hohe Lernbereitschaft, gute Kommunikationsfähigkeit englisch und deutsch	Energie	Du hast deine Ausbildung im Bereich Informatik (HTL oder Studium) abgeschlossen oder verfügst über einschlägige Berufserfahrung
Software Quality Engineer / Test Engineer (m/w/d)	Testen anhand von Tickets, Testlisten und freiem Testen, Testfälle, Testpläne, automatisierte Tests, Zusammenarbeit mit der Softwareentwicklung, Reporting, Verbesserung der Qualitätsprozesse, Projektmanagement und Service	Berufserfahrung im Bereich Softwaretests, Bugtrackingsysteme (Jira, Trac, svn, git, etc.), Testautomatisierung, Continuous integration (Jenkins, Squish), Testmanagementsystemen (TestRail), Programmierkenntnisse (Python, C++, Java, C#, Delphi), Durchsetzungsvermögen, Qualitäts- und Verantwortungsbewusstsein, Kommunikationsfähigkeiten in Deutsch und Englisch	Maschinenbau	Erfolgreich abgeschlossenes Studium mit Schwerpunkt Informatik / Software-Qualität oder einschlägige Berufserfahrung in der Software-Qualität oder Softwareentwicklung
Quality Assurance Manager IT (m/w/d)	Aufbau QA-Prozesse, Zusammenarbeit mit den IT- und Fachbereichen, Sicherstellung der Qualität, Testing in agilen Projektteams, Testfalldesign, Testdurchführung und Dokumentation, Fehleranalyse und Fehlerreporting, Bug-Management, Test Data Managements, Testautomatisierung, Evaluierung und Einführung Testmanagement- Tools	Expertise in Methoden und Prozessen der Softwarequalitätssicherung, Analytisches Denken, systematisches Vorgehen, Erfahrung mit Testmanagement-Tools, Testautomatisierung, Durchführung von Last- und Performance Tests, Verständnis für Datenbanken, Software-Architekturen und Schnittstellentechnologien, Skript- /Programmiersprache, hohes Qualitätsverständnis, Sehr gute Deutsch- und Englischkenntnisse, Teamfähigkeit, Eigeninitiative	Konsumgüter/ Handel	abgeschlossenes Studium oder gleichwertige Ausbildung, 2 Jahre praktische Erfahrung

(Associate) Software-Tester Qualitätssicherung (m/w/d)	Testplänen, automatisierte Tests, Testfälle, kontinuierlichen Weiterentwicklung der Testprozesse und -werkzeuge, Bewertung der Testergebnisse, Kundensupport	Erfahrungen mit Softwaretests, ISTQB-Zertifizierung, Teamorientiertes Arbeiten, analytisches Denkvermögen, kommunikative Fähigkeiten, hohe Lernbereitschaft	IT Consulting	Abgeschlossenes Studium der Wirtschaftsinformatik oder einer vergleichbaren Fachrichtung bzw. eine Ausbildung mit Bezug zur Informatik
Software Tester (m/w/d) Software Test Engineer - Quality Assurance Engineer	Testfällen, automatisierten Tests, Testing Tools (Selenium, JUnit, Mockito, FitNesse), Testergebnisse analysieren, Testautomatisierungsstrategien, Integrations- und Regressionstests, beratenden Funktion, KVP der Softwareprozesse	Berufserfahrung als Software Tester, Teststrategien sowie im Umgang mit Testingtools (Selenium, JUnit, Mockito, FitNesse, Java Testframeworks), Java Softwareentwicklung, Ticketsysteme JIRA oder Trac	Maschinenbau	Studium im Bereich (Wirtschafts-) Informatik, eine Ausbildung mit IT, Berufserfahrung im IT-Umfeld mit
Quality Assurance Manager/ Testmanager (m/w/d)	Testkonzeption, Testplanung, Testdurchführung, Reviews für funktionale und nichtfunktionale Anforderungen, Testautomatisierung, Koordination, on- und offshore Tests	Mehrjährige Berufserfahrung, ISTQB-Zertifizierung, Erfahrung in der Testautomatisierung, Qualitätsliebe, Englischkenntnisse in Wort und Schrift	Vertreib	Abgeschlossenes Studium im IT-Umfeld oder vergleichbare Qualifikation
Mitarbeiter/in Testmanagement und Qualitätssicherung (m/w/d)	Testkonzeption, Testplanung, Testdurchführung, Aufbau von Testumgebungen, Test-Tools, Last-/ Performance- /Oberflächentests, Fehleranalyse, Qualitätsstandards umsetzen, Verbesserung des QM-Systems	Erfahrung im Bereich, überzeugendes Auftreten, Durchsetzungsvermögen, Kommunikationsfähigkeit, Erfahrung im Release-Management, Skriptsprachen (Bash oder Python), Unix / Linux- Entwicklung, analytische, qualitätsorientierte Arbeitsweise, Teamfähigkeit, Deutsch- und Englischkenntnisse	IT	k.A.
Ingenieur (m/w/d) Qualitätsmanagement nach Automotive SPICE	Automotive SPICE Assessments, Software-Entwicklungsprozesse im Qualitätsmanagement, Schulungen, Verbesserungspotentiale identifizieren, Verbesserungsmaßnahmen durchführen	Automotive SPICE Provisional Assessor, Erfahrung Prozessverbesserungsprojekten, kommunikative und koordinative Fähigkeiten, Deutsch- und Englischkenntnisse in Wort und Schrift	Maschinenbau	Technisches Studium (Informatik, Mathematik, Elektrotechnik oder einem vergleichbaren Studiengang Berufserfahrung

Qualitätsmanager (m/w/d) im Bereich Software-Qualitätssicherung, Prozess- und Qualitätsmanagement	Beratung Qualitätssicherung nach Normen, Coaching für Quality Audits, Entwicklung Softwaresicherheitsmethoden, Einführung agiler Methoden ,Qualitätsvorausplanungen, Qualitätsüberwachung, Fehlerursachenanalyse, Maßnahmenfestlegung, Prozessverbesserung	MS-Office, Erfahrung im Bereich, Normen und Vorschriften (Automotive SPICE/ ISO/IEC 1550/ Funktionale Sicherheit ISO 26262), ergebnisorientiert, organisiert, Kommunikationsstärke, Teamorientierung, Deutsch- und Englischkenntnisse	Automobil	Sie haben ein abgeschlossenes Studium in den Bereichen Elektrotechnik, Informatik, Mathematik oder in einer vergleichbaren Richtung
Junior Entwicklungsingenieur für Prozesse und Qualität (m/w/d)	Einhaltung der Prozessqualität in der Steuergeräteentwicklung, Maßnahmen der Qualitätssicherung, Normen (Automotive SPICE / ISO 26262), Review-Methoden, Änderungs-/Fehler-/ Konfigurationsmanagement, Workshopdurchführung	Erste Erfahrungen im Automotive Umfeld, Affinität für den Bereich Fahrzeugentwicklung/Qualität, Kommunikationsstärke, akkurate Arbeitsweise, Deutsch- und Englischkenntnisse	Maschinenbau	Erfolgreich abgeschlossenes Studium im Bereich der Informatik, Wirtschaftsinformatik, der Mathematik, dem technischen Projektmanagement
Software QA Engineer/Quality Specialist (m/w/d)	Testspezifikationen, Testdaten, Testdurchführung, Auswahl Testmethoden, Testautomatisierungswerkzeugen, Testautomatisierung (Funktions-/ Performance-/Regressions-Tests), Feheleranalyse, Testergebnisse, Ansprechpartner	Qualitätsliebe, Teamfähigkeit, Kommunikationsstärke, Eigenverantwortliches Handeln, Berufserfahrung im Bereich Software-Testing Kenntnisse über Testmethoden/-werkzeuge, Teamplayer, präzise Arbeitsweise, Zuverlässigkeit und Eigenverantwortung, ISTQB-Zertifizierung, Deutsch- und Englischkenntnisse	IT	Abgeschlossenes technisches Studium oder eine vergleichbare Qualifikation
Software Qualitätssicherer (m/w/divers)	Testplanung, Testdurchführung, Projektmangement, Testkonzepte, Testautomatisierung, fachliche Führung von Mitarbeitern, Bewertung von Anforderungen/ Architektur/ Design /Kodierung, Testspezifikationen, Testprozeduren, Testreports, Unterstützung der Vorintegration	Erfahrungen im Umgang mit toolgestützter Softwareentwicklung, Programmierkentnisse (C++), Team- und Kommunikationsfähigkeit, Deutsch- und Englischkenntnisse in Wort und Schrift, ISTQB-Zertifikat	Elektronik	Studium vorzugsweise der Informatik, Elektrotechnik/Nachrichtentechnik , Physik oder Mathematik oder Ausbildung mit Weiterbildung

Solution Consultant Softwarequalitätssicherung oder Software Test Engineer (m/w/d)	Testerstellung, Testdurchführung, Testauswertung, Testautomatisierung, Unterstützung bei der Qualitätssicherung in Softwareentwicklungsprojekten, Testdesign	Testdurcshührung nach TQM, Anforderungserhebung, Testdokumente nach ISO/IEC/IEEE 29119, Erfahrung Softwarequalitätssicherung, Testziele konkretisieren, Testmethoden (Code Review, Debugging, Unit Testing, Stress /Regression Testing), Testwerkzeuge (TOSCA Testsuite/ HP Unified Functional Testing/ Ranorex Studio/ Selenium/ Katalon, Silk Test, QFTest , Redmine, Bugzilla, Mantis Bug Tracker, Atlassian JIRA),	Beratung	Hochschulstudium (Informatik, Wirtschaftsinformatik, Naturwissenschaften) oder vergleichbarer Ausbildung
Software Qualitätsmanager (m/w/d)	Steuerung Qualitäts- und Testmanagement, Unterstützung Programm-Management, KVP, Steuerung Softwarelieferanten, Testkonzepten, Testspezifikationen, Testplänen	Berufserfahrung im SQM, Software-Testautomatisierung, Multi-Projektmanagement-Skills, Kommunikations- und Präsentationsfähigkeiten, Deutsch- und Englischkenntnisse, Französisch, überzeugendes Auftreten, Motivation, Flexibilität, strukturierte Arbeitsweise	Finanz	Erfolgreich abgeschlossenes Studium der Betriebswirtschaftslehre, Wirtschaftsinformatik oder eines vergleichbaren Studiengangs
Verantwortliche/r für Softwarequalität (w/m/d) als Hauptsachbearbeiter/in-IT	Akzeptanz- und Abnahmetest für IT-Verfahren, Qualitätssicherungsmaßnahmen, Projektmanagementunterstützung, Ergebnispräsentation, Weiterentwicklung des Fachbereichs, Mitarbeiterführung, Ansprechperson	Kenntnisse statistischer Methoden, Kenntnisse SQM, Testdurchführung, Testbewetung, Testautomatisierung, Test-Werkzeuge, struktueirtes Arbeiten, Teamfähigkeit, Fähigkeit zur Führung, Genderkompetenz, Fähigkeit zur Wissensvermittlung, Kundenorientierung, Kommunikationskompetenz	Öffentlicher Dienst	Hochschule erworbenen Bachelor oder einen gleichwertigen Abschluss, vorzugsweise der Fachrichtung Informatik, Berufserfahrung
Prozessmanager ADAS-Software-Entwicklungsprozesse/Software-Qualitätsmanager (m/w/d)	Prozessdefinition, Tool-Entwicklung, Tool-Umsetzung, Prozessüberwachung, Schulung, Qualitätsziele, Software-Prozessreviews, Gegenmaßnahmen	Erfahrung mit Software-Entwicklungsmodellen (z. B. V-Modell, agile Entwicklung wie Scrum und Kanban etc.), Erfahrungen mit Bewertungsmodelle (z. B. Automotive SPICE®, CMMI etc), analytische Denkweise, Teamplaye, Englischkenntnisse in Wort und Schrift, Moderationstechniken, Prozessdesign, Projektmanagements, Software-Entwicklung, Software-Qualitätssicherung, Testmanagements, Tools PTC Integrity, Git, Jira, Confluence, Stages, Signavio	Automobil	Abgeschlossenes Studium der Fachrichtung Informatik, Wirtschaftsinformatik, Wirtschaftsingenieurwesen, Maschinenbau, Elektrotechnik oder vergleichbarer Studiengang

Group System & Software Quality Director (m/f/d)	Strategie für System- und Software-qualität, Ergebnisse messen und die Methode anpassen, Schulungen, KPIs, Fortschrittsberichte, Warn- und Eskalationsprozess	15 Jahren Berufserfahrung im Bereich Automotive, Kenntnisse im Bereich System und Software, Prozess und Management-Experte, Deutsch- und Englischkenntnise, Kommunikationsfähigkeiten, Führungsqualitäten	Automobil	Ingenieurstudium der Elektrotechnik, Informatik, Mechatronik etc.
Senior Software Supplier Quality Engineer	strategische Qualitätsmanagement für Software-, Daten- und IT-Services-Lieferanten,Management von Software-Supplier-Quality-Projekten, bewerten Software-Entwicklungsprozesse bei Lieferanten	Software-Projektmanagement, Software-Engineering und/oder Software-Qualitäts-Management. (z. B. Anforderungsmanagement, Software-Architektur, Software-Design), Automotive SPICE (z. B. ASPICE Provisional Assessor) oder vergleichbar (z. B. CMMI Assessor), Knowhow in Bosch-Einkaufsprozessen und Praxiserfahrung in Bosch Mobility Solutions wünschenswert	Maschinenbau	abgeschlossenes technisches Studium im Bereich Informatik, Wirtschaftsingenieurwesen, Maschinenbau oder Luftfahrtechnik
Qualitätsorientierte Softwareanalyse /-entwicklung (C, C++)	Konzeption, Entwicklung und Test von Embedded Systemen bis zur Serienreife, Optimierung der Software Design Rules, Analyse und Bewertung von Software Fehlern, Code Reviews, Kommunikation der Ergebnisse	Kenntnisse in der Codeanalyse und Softwarestrukturanalyse, Anwendung der Programmiersprachen C++ und C, präventiven Qualitätsmethoden und Software-Sicherheitsklassen, Teamfähigkeit, eine zielorientierte Arbeitsweise , Englischkenntnissen	Maschinenbau	Studium mit der mit der Fachrichtung Informatik, Informationstechnik oder vergleichbare Qualifikation erfolgreich absolviert
Qualitätsmanager (m/w/d) Software für den Bereich HAFAS Forschung und Entwicklung	Einarbeitung in komplexe bestehende Systeme Entwicklung und kontinuierliche Weiterentwicklung von Testmethoden Erstellen von Testspezifikationen Berücksichtigung der Testbarkeit beim Systementwurf Durchführung und Dokumentation von Tests Präsentation von Testergebnissen	Erfahrung mit Softwareentwicklung von Vorteil z.B. Java / C++ / XML, Client-/Server-Systeme, Mobile Applications Android oder iOS systemisches Denken Projekterfahrung im Umfeld von Softwareentwicklung Eigeninitiative und strukturierte sichere Deutsch- und Englischkenntnisse in Wort und Schrift	IT	abgeschlossenes Studium der Mathematik, Informatik (Bachelor, Master, Diplom) oder ähnlicher technischer Qualifikation

Qualitätsmanager (m/w/d) Softwareentwicklung	Qualitätssicherstellung der internen Softwareentwicklung, analysieren Anforderungen unserer Kunden an den Entwicklungsprozess, Qualitätsreviews der Entwicklungs-, Management-, Support-Prozesse, berichten den Qualitätsstatus, Weiterentwicklung des Softwareentwicklungsprozesses	Mehrjährige Berufserfahrung im Bereich Embedded-Softwareentwicklung/Software-Projektmanagement oder Software-Qualitätssicherung, idealerweise im internationalen Automotive-Umfeld, Ausbildungen oder Erfahrungen in den Bereichen Qualitätsmanagementstandards und –methoden,Reifegradmodellen (ASPICE, CMMI)	Automobil	Abgeschlossenes Studium der Fachrichtung Elektrotechnik, Informatik, Informationstechnik oder ein vergleichbarer Abschluss
Qualitätsingenieur (m/w/d) Software	Qualitätssicherung der Produkte des Softwareentwicklungsprozesses Qualitätssicherungsstrategien Software-Tests Prüfplänen und Prüfspezifikationen, Testumgebungen, Testprogrammen Definition der einzusetzenden Testwerkzeuge und Methoden	Erfahrung in der Anwendung der IEC 61508, Erfahrungen im Testmanagement,Normen und Verbesserungsmodelle wie CMMI, SPICE, V-Modell Kenntnisse in der Durchführung von Testaktivitäten und der Anwendung von Gute Kenntnisse in C, C++ und OO-Design, Deutsch- und Englischkenntnisse	Maschinenbau	Abgeschlossenes Studium der Informatik, Elektrotechnik oder Vergleichbares
Software Qualitätssicherungsingenieur*in / SPICE	Betreuung und Unterstützung der Softwareentwicklungsteams, Analyse, Definition und Bewertung von Softwareentwicklungsprozessen, Sicherstellung hoher Software-Qualität, Intensiver Austausch mit Kollegen	Mindestens 2 Jahre Berufserfahrung in der Entwicklung oder Unterstützung von Prozessanalysen und -bewertungen, IEC 15504 (SPICE), CMMI, gute Programmierkenntnisse in einer Hochsprache, Projektmanagement, Test und Absicherung von Embedded	Maschinenbau	Erfolgreich absolviertes Studium der Informatik, , Elektrotechnik oder vergleichbarer Abschluss
System & Software Germany Quality Manager (m/w/d)	Teams von System & Software Quality Engineers für den deutschen Standort (Erlangen) verantwortlich Sie ordnen den Projekten entsprechend die Mitarbeiter/innen der System- und Softwarequalität zu, basierend auf der geschätzten Arbeitsbelastung	Erfahrung in der Entwicklung oder im Support von Embedded Software Automotive SPICE,CMMI Erfahrungen im System- und Softwareentwicklungsprozess für Embedded-Systeme gutes Auftreten und besitzen die Fähigkeit zur Kommunikation mit Engineering-Teams	Maschinenbau	abgeschlossenes Studium im Ingenieurwesen, in Elektrotechnik, Elektronik oder Informatik

Software Quality Assurance Engineer	Acceptance, exploratory, and regression testing of our application across multiple OSs Logging defect and improvement tickets and following up on their resolution Creating and maintaining test cases in TestRail MongoDB	Experience in testing software, working in testing software having English Experience testing desktop applications Familiarity with MongoDB or other NoSQL databases Experience working with SQL databases Experience working with TestRail	IT	Computer Science, Software Engineering or a related discipline

7 Literaturverzeichnis

(2017), 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM 2017 : 9-10 November 2017, Toronto, Ontario, Canada : proceedings, Piscataway, NJ 2017

(2017), 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM 2017 : 9-10 November 2017, Toronto, Ontario, Canada : proceedings, Piscataway, NJ 2017

(1990), 12th International Conference on Software Engineering, March 26-30, 1990- Nice, France - Proceedings, Washington, D.C 1990

(1990), 12th International Conference on Software Engineering, March 26-30, 1990- Nice, France - Proceedings, Washington, D.C 1990

(2008), 16th IEEE International Requirements Engineering, 2008 - RE '08 ; September 8 - 12, 2008, Barcelona, Catalunya, Spain ; proceedings, Los Alamitos, Calif. 2008

(2002), 18th International Conference on Software Maintenance (ICSM 2002) - Maintaining Distributed Heterogeneous Systems, Los Alamitos Oct. 2002

(2016), 2016 3rd International Workshop on Emerging Ideas and Trends in Engineering of Cyber-Physical Systems (EITEC) - April 11, 2016, Vienna, Austria, Piscataway, NJ 2016

(2017), 2017 IEEE/ACM 39th International Conference on Software Engineering companion - ICSE-C 2017 : 20-28 May 2017, Buenos Aires, Argentina : proceedings, Piscataway, NJ 2017

(2018), 2018 IEEE 11th International Conference on Software Testing, Verification and Validation workshops - ICSTW 2018 : 9-13 April 2018, Västerås, Sweden : proceedings, Piscataway, NJ 2018

Abdel-Hamid, T.K. (1988), The Economics of Software Quality Assurance: A Simulation-Based Case Study, in: MIS Quarterly, 12, 1988, 3, S. 395

Abrahamsson, P., Bosch, J., Brinkkemper, S. und Mädche, A. (2018), Software Business, Platforms, and Ecosystems: Fundamentals of Software Production Research (Dagstuhl Seminar 18182), 35 pages, 2018

acatech (2011), Cyber-Physical Systems. Innovationsmotor für Mobilität, Gesundheit, Energie und Produktion, Herausgegeben von acatech, Auf den Seiten von: ,

https://www.acatech.de/publikation/cyber-physical-systems/, Zugriff am 26.06.2019

Adrion, W. R. (1997), Proceedings of the 19th international conference on Software engineering - ICSE '97, New York, New York, USA 1997

Ahlers, J. und Emmerich, W. (1984), QS erfordert konstruktive und analytische Maßnahmen, in: Computerwoche, 1984

Ai, J., Su, W. und Wang, F. (2018), Software Reliability Evaluation Method Based on a Software Network, in: Ghosh (Hrsg., 2018), S. 136-137

Alexander, C. und Ambrose, P. (1999), The Impact of Software Development Process on Information Systems Success: A Theoretical Perspective Integrating Customer Satisfaction, Technical and Organizational Issues, in: AMCIS 1999 Proceedings, 1999

Alghamdi, J.S., Rufai, R.A. und Khan, S.M. (2005), OOMeter: A Software Quality Assurance Tool, in: (Hrsg., 2005), S. 190-191

Alshayeb, M. (2009), Empirical investigation of refactoring effect on software quality, in: Information and Software Technology, 51, 2009, 9, S. 1319-1326

Ambrose, P. und Eynon, J. (1998), A Theoretical Model for Software Quality Management, in: AMCIS 1998 Proceedings, 1998

American Society for Quality (2019), The History of Quality, Auf den Seiten von: , https://asq.org/quality-resources/history-of-quality, Zugriff am 16.04.2019

Anecon (2011), Ergebnisse der Umfrage 2011: "Softwaretest in der Praxis“, in: it&t-business, 2011, 10, S. 11

Arksey, H. und O'Malley, L. (2005), Scoping studies: towards a methodological framework, in: International Journal of Social Research Methodology, 8, 2005, 1, S. 19-32

Avgeriou, P., Zdun, U., Caracciolo, A., Lungu, M.F. und Nierstrasz, O. (2014), How Do Software Architects Specify and Validate Quality Requirements? - Software Architecture, 2014

Ayaburi, E., Ko, M. und Walz, D. (2016), Understanding Antecedents and Consequence of Knowledge Sharing in Software Testing Teams: A Conceptual Model, in: AMCIS 2016 Proceedings, 2016

Azar, D. und Vybihal, J. (2011), An ant colony optimization algorithm to improve software quality prediction models: Case of class stability, in: Information and Software Technology, 53, 2011, 4, S. 388-393

Bächle, M. (1996), Qualitätsmanagement der Softwareentwicklung, Wiesbaden 1996

Baker, E.B. (2001), Which way, SQA? [software quality assurance], in: IEEE Software, 18, 2001, 1, S. 16-18

Baker, R.A. (1997), Code reviews enhance software quality, in: Adrion (Hrsg., 1997), S. 570-571

Balzert, H. (1998), Lehrbuch der Software-Technik - Software- Management, Software-Qualitätssicherung, Unternehmensmodellierung, Heidelberg, Berlin 1998

Balzert, H. und Ebert, C. (2008), Softwaremanagement, 2. Aufl., Heidelberg 2008

Barber, K.S. und Holt, J. (2001), Software architecture correctness, in: IEEE Software, 18, 2001, 6, S. 64-65

Bartram, A. (2006), Outsourcing der Software-Entwicklung: Risiken erkennen und minimieren, in: IM Information Management & Consulting, 2006, 3, S. 61-66

Basili, V., DeMillo, R. und Katayama, T. (1993), Software Engineering, 15th International Conference (ICSE-15), Los Alamitos Oct. 1993

Baskerville, R., Levine, L., Ramesh, B. und Slaughter, S. (2002), Balancing quality and agility in Internet speed software development, in: (Hrsg., 2002),

Beck, K. und Andres, C. (2005), Extreme programming explained - Embrace change, 2nd ed., Boston, MA 2005

Becker, J., König, W., Schütte, R., Wendt, O. und Zelewski, S. (1999), Wirtschaftsinformatik und Wissenschaftstheorie, Wiesbaden 1999

Behkamal, B., Kahani, M. und Akbari, M.K. (2009), Customizing ISO 9126 quality model for evaluation of B2B applications, in: Information and Software Technology, 51, 2009, 3, S. 599-609

Ben-Menachem, M. (2008), Towards management of software as assets: A literature review with additional sources, in: Information and Software Technology, 50, 2008, 4, S. 241-258

Bird, C., Nagappan, N., Devanbu, P., Gall, H. und Murphy, B. (2009), Does distributed development affect software quality? An empirical case study of Windows Vista, in: (Hrsg., 2009), S. 518-528

Bøegh, J. (2008), A New Standard for Quality Requirements, in: IEEE Software, 25, 2008, 2, S. 57-63

Boehm, Brown und Lipow (1976), Quantitative Evaluation of Software Quality, in: Yeh, Ramamoorthy (Hrsg., 1976), S. 592-605

Bouktif, S., Kegl, B. und Sahraoui, H. (Oct. 2002), Combining software quality predictive models: an evolutionary approach, in: (Hrsg., Oct. 2002), S. 385-392

Braun, P., Broy, M., Houdek, F., Kirchmayr, M., Müller, M., Penzenstadler, B., Pohl, K. und Weyer, T. (2014), Guiding requirements engineering for software-intensive embedded systems in the automotive industry, in: Computer Science - Research and Development, 29, 2014, 1, S. 21-43

Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M. und Khalil, M. (2007), Lessons from applying the systematic literature review process within the software engineering domain, in: Journal of Systems and Software, 80, 2007, 4, S. 571-583

Briand, L., Nejati, S., Sabetzadeh, M. und Bianculli, D. (2016), Testing the untestable, in: Dillon, Visser, Williams (Hrsg., 2016), S. 789-792

Briand, L. C. und Wolf, A. L. (2007), Future of software engineering, 2007 - FOSE '07 ; 23 - 25 May 2007, Minneapolis, Minnesota ; [at] ICSE 2007, [29th International Conference on Software Engineering, Los Alamitos, Calif. 2007

Brown, A.W., Carney, D.J. und Clements, P.C. (1995), A case study in assessing the maintainability of large, software-intensive systems, in: Melhart, Rozenblit (Hrsg., 1995), S. 240-247

Brown, T. (2008), Design thinking, in: Harvard Business Review, 86, 2008, 6, S. 84-93

Broy, M. (2006a), Challenges in automotive software engineering, in: Osterweil (Hrsg., 2006), S. 33-42

Broy, M. (2006b), The 'Grand Challenge' in Informatics: Engineering Software-Intensive Systems, in: Computer, 39, 2006, 10, S. 72-80

Bures, T. und Angelis, L. (2018), SEAA 2018 - 44th Euromicro Conference on Software Engineering and Advanced Applications : 29-31 August 2018, Prague, Czech Republic : proceedings, Piscataway, NJ 2018

Burghartz, D. (1994), Qualitaetsmanagement in der Softwareentwicklung, in: Markt & Technik, 1994, 46, S. 47

Buzzell, R.D., Gale, B.T. und Greif, H.-H. (1989), Das PIMS-Programm - Strategien und Unternehmenserfolg, Wiesbaden 1989

Çaglayan, B. und Bener, A.B. (2016), Effect of developer collaboration activity on software quality in two large scale projects, in: Journal of Systems and Software, 118, 2016, S. 288-296

CAPES (2012), COMUNICADO no004/2012–ÁREA DE CIÊNCIA DA COMPUTAÇÃO ATUALIZAÇÃO DA CLASSIFICAÇÃO DE CONFERÊNCIAS DA ÁREA, Herausgegeben von Ministério da Educação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, Auf den Seiten von: , http://www.capes.gov.br/images/stories/download/avaliacao/Comunicado_004_2012_Ciencia_da_Computacao.pdf, Zugriff am 23.07.2019

CAPES (2017), Qualis Ranking - Software engineering, Auf den Seiten von: Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, https://medium.com/@aserg.ufmg/qualis-capes-ranking-of-software-engineering-journals-2013-2016-a76fbb5205a4

Card, D. (1988), Software product assurance: measurement and control, in: Information and Software Technology, 30, 1988, 6, S. 322-330

Card, D. (1990), Software quality engineering, in: Information and Software Technology, 32, 1990, 1, S. 3-10

Cavalcante, E. (2015), On the Architecture-Driven Development of Software-Intensive Systems-of-Systems, in: (Hrsg., 2015), S. 899-902

Cavano, J.P. und LaMonica, F.S. (1987), Quality Assurance in Future Development Environments, in: IEEE Software, 4, 1987, 5, S. 26-34

Cleve, A. (2013), 2013 17th European Conference on Software Maintenance and Reengineering (CSMR) - 5 - 8 March 2013, Genova, Italy, Piscataway, NJ 2013

Cobb, R.H. und Mills, H.D. (1990), Engineering software under statistical quality control, in: IEEE Software, 7, 1990, 6, S. 45-54

Cockburn, A. (2001), Agile Software Development: The Business of Innovation, in: IEEE Computer, 34, 2001, 9, S. 120-122

Computerwoche (1979a), Structo 79, in: Computerwoche, 1979

Computerwoche (1979b), Zur Förderung der Softwarequalität möchte die Software Research Associated - Normen und Konventionen beschäftigt sind. Ziel der Gemeinschaft soll sein, in: Computerwoche, 1979

Computerwoche (1983), Standard-Programme: "TÜV-Plakette" für Software erwünscht, in: Computerwoche, 1983

Computerwoche (1985a), Gütezeichen für Software gerät ins Kreuzfeuer der Kritik, Fachausstellung FE-Lösungen verschiedener Anbieter kennenzulernen, in: Computerwoche, 1985

Computerwoche (1985b), Qualitätssicherung in der Praxis noch mit Organisations-Problemen - Konsequenter Ergebnis-Check spart bare MünzeDie absolute Voraussetzung, in: Computerwoche, 1985

Computerwoche (1989), Qualitätsgeprüfte Software erhält Gütesiegel, in: Computerwoche, 1989

Computerwoche (2005a), Mehr Erfolg in IT-Projekten, in: Computerwoche, 2005

Computerwoche (2005b), Software - Qualität ist gefragt, in: Computerwoche, 2005

Cooper, H.M. (1988), Organizing knowledge syntheses: A taxonomy of literature reviews, in: Knowledge in Society, 1, 1988, 1, S. 104-126

CORE (2018), CORE Conference Ranking, Herausgegeben von Computing Research and Education Association of Australasia, Auf den Seiten von: , http://portal.core.edu.au/conf-ranks/?search=software&by=all&source=CORE2018&sort=arank&page=1, Zugriff am 24.07.2019

Crispin, L. (2006), Driving Software Quality: How Test-Driven Development Impacts Software Quality, in: IEEE Software, 23, 2006, 6, S. 70-71

Cuesta, C. E., Garlan, D. und Pérez, J. (2018), Software Architecture - 12th European Conference on Software Architecture, ECSA 2018, Madrid, Spain, September 24-28, 2018, Proceedings, Cham 2018

Dalal, S.R., Horgan, J.R. und Kettenring, J.R. (Oct. 1993), Reliable software and communication: software quality, reliability, and safety, in: Basili, DeMillo, Katayama (Hrsg., Oct. 1993), S. 425-435

Darmawan, N., Chong, A., Ooi, K.-B. und Boon, I.T. (2010), Factor strategy model: proofs of prototype concept for software quality evaluation, in: (Hrsg., 2010),

Datta, S. (2018), How does developer interaction relate to software quality? an examination of product development data, in: Empirical Software Engineering, 23, 2018, 3, S. 1153-1187

Daun, M., Brings, J., Weyer, T. und Tenbergen, B. (2016), Fostering concurrent engineering of cyber-physical systems a proposal for an ontological context framework, in: (Hrsg., 2016), S. 5-10

Denning, P.J. (2016), Software quality, in: Communications of the ACM, 59, 2016, 9, S. 23-25

Dillon, L., Visser, W. und Williams, L. (2016), 2016 IEEE/ACM 38th IEEE International Conference on Software Engineering companion - ICSE 2016 - 14-22 May 2016, Austin, Texas, USA : proceedings, Piscataway, NJ 2016

Norm, DIN EN ISO 9000:2015-11, DIN EN ISO 9000:2015-11, Qualitätsmanagementsysteme- Grundlagen und Begriffe (ISO_9000:2015); Deutsche und Englische Fassung EN_ISO_9000:201530.11.2015

Dion, R. (1992), Elements of a process-improvement program (software quality), in: IEEE Software, 9, 1992, 4, S. 83-85

Dowling, P. und McGrath, K. (2015), Using Free and Open Source Tools to Manage Software Quality, in: Queue, 13, 2015, 4, S. 20

Drave, I., Hillemacher, S., Greifenberg, T., Rumpe, B., Wortmann, A., Markthaler, M. und Kriebel, S. (2018), Model-Based Testing of Software-Based System Functions, in: Bures, Angelis (Hrsg., 2018), S. 146-153

Easton, G.S. und Jarrell, S.L. (1998), The Effects of Total Quality Management on Corporate Performance: An Empirical Investigation, in: The Journal of Business, 71, 1998, 2, S. 253-307

Ehrlich, W.K., Stampfel, J.P. und Wu, J.R. (1990), Application of software reliability modeling to product quality and test process, in: (Hrsg., 1990), S. 108-116

Erikkson, I. und McFadden, F. (1993), Quality function deployment: a tool to improve software quality, in: Information and Software Technology, 35, 1993, 9, S. 491-498

Ferenc, R. (2009), 13th European Conference on Software Maintenance and Reengineering, 2009 - CSMR '09 ; 24 - 27 March 2009, Kaiserslautern, Germany, Piscataway, NJ 2009

Ferstl, O.K. und Sinz, E.J. (2006), Grundlagen der Wirtschaftsinformatik, 5., überarb. und erw. Aufl., München 2006

Feßl, S. (2013), Die mobile-App-Falle, in: it&t-business, 2013, 07-08, S. 31

Fettke, P. (2006), State-of-the-Art des State-of-the-Art, in: WIRTSCHAFTSINFORMATIK, 48, 2006, 4, S. 257

(2007), First International Symposium on Empirical Software Engineering and Measurement, 2007 - ESEM 2007 ; 20 - 21 Sept. 2007, Madrid, Spain ; proceedings, Los Alamitos, Calif. 2007

Fischermann, T. (2006), Boom hinter verschlossenen Türen, in: DIE ZEIT, 2006, Ausgabe 2011.05.2006, https://www.zeit.de/2006/20/Yahoo_komma_Google_xml, Zugriff am 22.10.2019

Fortmann, H. R. und Kolocek, B. (2019), Arbeitswelt der Zukunft - Trends - Arbeitsraum - Menschen - Kompetenzen, 1. Aufl., Wiesbaden 2019

Frühauf, K., Ludewig, J. und Sandmayr, H. (2002), Software-Projektmanagement und -Qualitätssicherung 2002

Galin, D. (2004), Software quality assurance - From theory to implementation, Harlow 2004

Gartner Inc. (2017), Leading the IoT - Gartner Insights on How to Lead in a Connected World, Auf den Seiten von: , https://www.gartner.com/imagesrv/books/iot/iotEbook_digital.pdf, Zugriff am 11.04.2019

Garvin, D.A. (1988), Managing quality - The strategic and competitive edge, New York, NY 1988

Geck, B. (2015a), Qualitätskontrolle in der Programmierung ein Muss, in: CIO, 2015

Geck, B. (2015b), Übersicht zu den aktuellen Testing-Trends, in: CIO - IT-Strategie für Manager, 2015

Geiger, W. und Kotte, W. (2008), Handbuch Qualität - Grundlagen und Elemente des Qualitätsmanagements: Systeme, Perspektiven, 5., vollständig überarbeitete und erweiterte Auflage, Wiesbaden 2008

Gerostathopoulos, I., Bures, T., Hnetynka, P., Keznikl, J., Kit, M., Plasil, F. und Plouzeau, N. (2016), Self-adaptation in software-intensive cyber–physical systems: From system goals to architecture configurations, in: Journal of Systems and Software, 122, 2016, S. 378-397

Ghosh, S. (2018), 29th IEEE International Symposium on Software Reliability Engineering workshops - ISSREW 2018 : 15-18 October 2018, Memphis, Tennessee, USA : proceedings, Piscataway, NJ 2018

Giebel, M. (2011), Wertsteigerung durch Qualitätsmanagement - Entwicklung eines Modells zur Beschreibung der Wirkmechanismen und eines Vorgehenskonzepts zu dessen Einführung, Kassel 2011

Goedicke, M., Menzies, T. und Saeki, M. (2012), 2012 proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE) - 3 - 7 Sept. 2012, Essen, Piscataway, NJ 2012

Goldbach, T., Kemper, V. und Benlian, A. (2014), Mobile Application Quality and Platform Stickiness under Formal vs. Self-Control — Evidence from an Experimental Study, in: ICIS 2014 Proceedings, 2014

Goldstein, M. und Segall, I. (2015), Automatic and Continuous Software Architecture Validation, in: (Hrsg., 2015), S. 59-68

Gómez, M.N. und Acuña, S.T. (2014), A replicated quasi-experimental study on the influence of personality and team climate in software development, in: Empirical Software Engineering, 19, 2014, 2, S. 343-377

Google LLC (2019a), Google Books: Ngram Viewer - Software Quality Management, Auf den Seiten von: , https://books.google.com/ngrams/graph?content=Software+Quality+Management&year_start=1980&year_end=2008&corpus=15&smoothing=2&share=&direct_url=t1%3B%2CSoftware%20Quality%20Management%3B%2Cc0#t1%3B%2CSoftware%20Quality%20Management%3B%2Cc0, Zugriff am 16.04.2019

Google LLC (2019b), Google Trends - Software Quality Management, Auf den Seiten von: , https://trends.google.de/trends/explore?date=all&q=Software%20quality%20management, Zugriff am 16.04.2019

Gopal, A. und Koka, B.R. (2010), The Role of Contracts on Quality and Returns to Quality in Offshore Software Development Outsourcing, in: Decision Sciences, 41, 2010, 3, S. 491-516

Grady, R.B. (1993), Practical results from measuring software quality, in: Communications of the ACM, 36, 1993, 11, S. 62-68

Grady, R.B. und Caswell, D.L. (1987), Software metrics - Establishing a company-wide program, Englewood Cliffs, NJ 1987

Gree, G.C., Hevner, A.R. und Collins, R. (2005), The impacts of quality and productivity perceptions on the use of software process improvement innovations, in: Information and Software Technology, 47, 2005, 8, S. 543-553

Grochla, E. (1978), Brisante Mängelliste: - Wirtschaftlichkeit durch Programmqualität erreichen, in: Computerwoche, 1978, 43

Große Böckmann, M. (Hrsg., 2011, Qualität in produzierenden Unternehmen Deutschlands 2010, Aachen: Fraunhofer IPT

Gustafson, G.G. und Kerr, R.J. (1982), Some practical experience with a software quality assurance program, in: Communications of the ACM, 25, 1982, 1, S. 4-12

Guzman, L., Vollmer, A.M., Ciolkowski, M. und Gillmann, M. (2017), Formative Evaluation of a Tool for Managing Software Quality, in: (Hrsg., 2017), S. 297-306

Haag, S., Raja, M.K. und Schkade, L.L. (1996), Quality function deployment usage in software development, in: Communications of the ACM, 39, 1996, 1, S. 41-49

Haase, A. (1985), Schlechtes Image erschwert die Arbeit der QS-Abteilung, in: Computerwoche, 1985

Haley, T.J. (1996), Software process improvement at Raytheon, in: IEEE Software, 13, 1996, 6, S. 33-41

Hall, T. (1995), No Quality Without Equality, in: IEEE Software, 1995

Hausen, Müllerburg und Züllighoven (1981), Software-Produktion, in: Computerwoche, 1981

Hayes, J.H., Dekhtyar, A. und Sundaram, S.K. (2005), Improving After-the-Fact Tracing and Mapping: Supporting Software Quality Predictions, in: IEEE Software, 22, 2005, 6, S. 30-37

Heimdahl, M.P.E. (2007), Safety and Software Intensive Systems: Challenges Old and New, in: Briand, Wolf (Hrsg., 2007), S. 137-152

Heinen, T. (2013), App-sichern: eine effektive Mobility- und Security-Strategie, in: IT Business, 2013, 003

Heinrich, L.J. und Lehner, F. (2005), Informationsmanagement - Planung, Überwachung und Steuerung der Informationsinfrastruktur, 8., vollst. überarb. und erg. Aufl., München 2005

Heinrich, R. (2018), Tailored quality modeling and analysis of software-intensive systems, in: (Hrsg., 2018), S. 336-341

Henderson-Sellers, B. (1991), Parallels between object-oriented software development and total quality management, in: Journal of Information Technology, 6, 1991, 2, S. 63-67

Herbsleb, J., Carleton, A., Rozum, J., Siegel, J. und Zubrow, D. (1994), Benefits of CMM-Based Software Process Improvement - Initial Results, Technical Report, Carnegie Mellon University, Pittsburgh 1994

Herzig, K., Greiler, M., Czerwonka, J. und Murphy, B. (2015), The art of testing less without sacrificing quality, 2015

Herzwurm, G. und Mikusz, M. (2016), Software-Qualitätsmanagement, Auf den Seiten von: , http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Systementwicklung/Management-der-

Systementwicklung/Software-Qualitatsmanagement/index.html, Stand: 07.12.2016, Zugriff am 02.07.2019

Hessel, A., Larsen, K.G., Mikucionis, M., Nielsen, B., Pettersson, P. und Skou, A. (2008), Testing real-time systems using UPPAAL, in: Hierson, Bowen, Harman (Hrsg., 2008), S. 77-117

Hierson, Bowen und Harman (2008), Formal methods and testing 2008

Hinterhuber, H.H. und Matzler, K. (2009), Kundenorientierte Unternehmensführung, Wiesbaden 2009

Hirayama, M., Sato, H., Yamada, A. und Tsuda, J. (1990), Practice of quality modeling and measurement on software life-cycle, in: (Hrsg., 1990), S. 98-107

Höhn, H. (2009), Software Engineering nach Automotive SPICE - Entwicklungsprozesse in der Praxis - ein Continental-Projekt auf dem Weg zu Level 3, 1. Aufl., Heidelberg 2009

Hollenbach, C., Young, R., Pflugrad, A. und Smith, D. (1997), Combining quality and software improvement, in: Communications of the ACM, 40, 1997, 6, S. 41-45

Hölzl, M., Rauschmayer, A. und Wirsing, M. (2008), Engineering of Software-Intensive Systems: State of the Art and Research Challenges, in: Wirsing (Hrsg., 2008), S. 1-44

Hon, S.E. (1990), Assuring software quality through measurements: A buyer's perspective, in: Journal of Systems and Software, 13, 1990, 2, S. 117-130

Huang, D. und Sarjoughian, H. (2004), Software and Simulation Modeling for Real-Time Software-Intensive Systems, in: Roberts, Wilson, Turner (Hrsg., 2004), S. 196-203

Huchard, M., Käster, C. und Fraser, G. (2018), ASE '18 - Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering : September 3-7, 2018, Montpellier, France, New York, NY 2018

Huffman Hayes, J. (2003), Do you like Pina Coladas? How improved communication can improve software quality, in: IEEE Software, 20, 2003, 1, S. 90-92

Hyrynsalmi, Suominen, Jud, Wang, Bosch und Münch (2018), Proceedings of the International Workshop on Software-intensive Business - Start-ups, Ecosystems and Platforms 2018

(2009), IEEE 31st International Conference on Software Engineering, 2009 - ICSE 2009 ; 16 - 24 May 2009, Vancouver, Canada ; proceedings, Piscataway, NJ 2009

(2014), IEEE International Conference on Software Maintenance and Evolution (ICSME), 2014 - Sept. 29 [i.e. 28], 2014 - Oct. 3, 2014, Victoria, British Columbia, Canada, Piscataway, NJ 2014

IEEE Standards Board (1983), ANSI/ IEEE Std 729-1983 - IEEE Standard Glossary of Software Engineering Terminology, [S.l.] 1983

IOT Analytics (2018), State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating, Herausgegeben von IOT Analytics, Auf den Seiten von: , https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/, Zugriff am 11.04.2019

it&t-business (2005), Effizientes Testmanagement, in: it&t-business, 2005, 3/05, S. 28

it&t-business (2009), Komplettlösungen für mehr Softwarequalität, in: it&t-business, 2009, 06, S. 22

it&t-business (2012), Trend zu Managed Services, in: it&t-business, 2012, 07-08, S. 31

it&t-business (2013), Software-Tests in der Wolke, in: it&t-business, 2013, 06, S. 36

Izurieta, C., Griffith, I. und Huvaere, C. (2017), An Industry Perspective to Comparing the SQALE and Quamoco Software Quality Models, in: (Hrsg., 2017), S. 287-296

Jekat, D. (1976), Prüfzeichen für Software?, in: Computerwoche, 1976, 26

Jones, C. und Bonsignour, O. (2012), The economics of software quality, Upper Saddle River, NJ 2012

(2010), Journal of Computer Information Systems 2010

Kafura, D. und Henry, S. (1981), Software quality metrics based on interconnectivity, in: Journal of Systems and Software, 2, 1981, 2, S. 121-131

Kannabiran, G. und Sankaran, K. (2011), Determinants of software quality in offshore development – An empirical study of an Indian vendor, in: Information and Software Technology, 53, 2011, 11, S. 1199-1208

Kaufmann, R. (2009), Vom Kostenfaktor zum Erfolgsgaranten, in: it&t-business, 2009, 11, S. 24

Kaynak, H. (2003), The relationship between total quality management practices and their effects on firm performance, in: Journal of Operations Management, 21, 2003, 4, S. 405-435

Khoshgoftaar, T.M., Munson, J.C., Bhattacharya, B.B. und Richardson, G.D. (1992), Predictive modeling techniques of software quality from software measures, in: IEEE Transactions on Software Engineering, 18, 1992, 11, S. 979-987

Kitchenham, B.A., Budgen, D. und Brereton, O.P. (Hrsg., 2010, The value of mapping studies: a participantobserver case study: BCS Learning & Development Ltd, in: EASE'10 Proceedings of the 14th international conference on Evaluation and Assessment in Software Engineering, S. 25-33

Kitchenham, C. (2007), Guidelines for performing Systematic Literature Reviews in Software EngineeringSoftware Engineering Group School of Computer Science and Mathematics, 2007

Kläs, M., Bauer, T., Dereani, A., Soderqvist, T. und Helle, P. (2015), A Large-Scale Technology Evaluation Study: Effects of Model-based Analysis and Testing, in: (Hrsg., 2015), S. 119-128

Klaus, H. (1979), Software-Qualität durch Kooperation, in: Computerwoche, 1979

Konrad, S. und Gall, M. (2008), Requirements Engineering in the Development of Large-Scale Systems, in: (Hrsg., 2008), S. 217-222

Kozlov, D., Koskinen, J., Markkula, J. und Sakkinen, M. (2007), Evaluating the Impact of Adaptive Maintenance Process on Open Source Software Quality, in: (Hrsg., 2007), S. 186-195

Kreuzer, M. (2018), Herausforderung Embedded Devices - Testabdeckung bei kleinen Targets, in: Markt & Technik, 2018, 32, S. 18-22

Laitenberger, O. (1998), Studying the effects of code inspection and structural testing on software quality, in: (Hrsg., 1998), S. 237-246

Lehner, F. (1999), Theoriebildung in der Wirtschaftsinformatik, in: Becker, König, Schütte, Wendt, Zelewski (Hrsg., 1999), S. 5-25

Lientz, B.P. und Swanson, E.B. (1981), Problems in application software maintenance, in: Communications of the ACM, 24, 1981, 11, S. 763-769

Liggesmeyer, P. (2009), Software-Qualität - Testen, Analysieren und Verifizieren von Software, 2. Aufl., s.l. 2009

Liu, H., Li, G., Ma, Z. und Shao, W. (2007), Scheduling of conflicting refactorings to promote quality improvement, in: Stirewalt, Egyed, Fischer (Hrsg., 2007), S. 489

Liu, X.F. (1998), A quantitative approach for assessing the priorities of software quality requirements, in: Journal of Systems and Software, 42, 1998, 2, S. 105-113

Mallett, R., Hagen-Zanker, J., Slater, R. und Duvendack, M. (2012), The benefits and challenges of using systematic reviews in international development research, in: Journal of Development Effectiveness, 4, 2012, 3, S. 445-455

Markt & Technik (2007), Neue Herausforderungen für Embedded-Entwickler, in: Markt & Technik, 2007, 24, S. 28

Markt & Technik (2016), Tools für mehr funktionale Sicherheit, in: Markt & Technik, 2016, 26, S. 52

Markt & Technik (2018), Herausforderung Embedded Devices, in: Markt & Technik, 2018, 32, S. 18

Masuda, S., Ono, K., Yasue, T. und Hosokawa, N. (2018), A Survey of Software Quality for Machine Learning Applications, in: (Hrsg., 2018), S. 279-284

Mathiassen, L., Ngwenyama, O.K. und Aaen, I. (2005), Managing Change in Software Process Improvement, in: IEEE Software, 22, 2005, 6, S. 84-91

McCall, Richards und Walter (1977), Factors in Software QualityRome Air Developement Center, 1977

Melhart, B. und Rozenblit, J. (1995), Proceedings of the 1995 International Symposium and Workshop on Systems Engineering of Computer Based Systems, March 6-9,1995, Tucson, Arizona, New York 1995

Mellis, W. (2004), Projektmanagement der SW-Entwicklung - Eine umfassende und fundierte Einführung, Wiesbaden 2004

Mellis, W., Herzwurm, G. und Stelzer, D. (1996), TQM der Softwareentwicklung - Mit Prozeßverbesserung, Kundenorientierung und Change Management zu erfolgreicher Software, Wiesbaden, s.l. 1996

Mendes, E. und Winkler, D. (2018), Special issue on “software quality in software-intensive systems”, in: Software Quality Journal, 26, 2018, 2, S. 657-660

Mens, T. (2012), 16th European Conference on Software Maintenance and Reengineering (CSMR), 2012 - 27 - 30 March 2012, Szeged, Hungary ; proceedings, Piscataway, NJ 2012

Merbeth, G. und Abbenhardt, H. (1984), Qualitätssicherung - der Glaube allein genügt nicht, in: Computerwoche, 1984

Mertens, P. und Back, A. (2001), Lexikon der Wirtschaftsinformatik, 4., vollst. neu bearb. und erw. Aufl., Berlin 2001

Meyer, B., Baresi, L. und Mezini, M. (2013), 2013 9th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the

Foundations of Software Engineering (ESEC/FSE) - Proceedings : August 18-26, 2013, Saint Petersburg, Russia, New York, NY 2013

Milberg, J. und Schuh, G. (2002), Erfolg in Netzwerken, Berlin, Heidelberg, s.l. 2002

Mishra, B.K., Prasad, A. und Raghunathan, S. (2002), Quality and Profits Under Open Source Versus Closed Source, in: undefined, 2002

Mistrík, I. (2016), Software quality assurance - In large scale and complex software-intensive systems, Waltham, MA 2016

Miyoshi, T. und Azuma, M. (1993), An empirical study of evaluating software development environment quality, in: IEEE Transactions on Software Engineering, 19, 1993, 5, S. 425-435

Mo, R., Snipes, W., Cai, Y., Ramaswamy, S., Kazman, R. und Naedele, M. (2018), Experiences applying automated architecture analysis tool suites, in: Huchard, Käster, Fraser (Hrsg., 2018), S. 779-789

Mücke, V. (2002), Die sieben wichtigsten Erfolgsfaktoren, in: Computerwoche, 2002

Müller, M. (2007), Automotive SPICE in der Praxis - Interpretationshilfe für Anwender und Assessoren, 1. Aufl., Heidelberg 2007

Müller, V. (2003), Prozessorientiertes IT-Qualitätsmanagement, in: HMD Praxis der Wirtschaftsinformatik, 2003, 232, S. 66-78

Müller-Ettrich, G. (1988), Die Manager der Unternehmen müssen noch viel dazulernen, denn, in: Computerwoche, 1988

Munson, J.C. und Khoshgoftaar, Y.M. (1990), Regression modelling of software quality: empirical investigation, 1990

Nakamura, M. und Hamagami, T. (2012), A Software quality evaluation method using the change of source code metrics, in: (Hrsg., 2012),

Nath, K. (2015), What Comes after Web 3.0? Web 4.0 and the Future, in: Proceedings of the International Conference on Computing and Communication Systems, 2015

Naur, P. und Randell, B. (Hrsg., 1969, Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO

Nelson, K. und Santos, C. (2007), Attractiveness of Open Source Projects: A Path to Software Quality, in: AMCIS 2007 Proceedings, 2007

Nguyen-Duc, A., Cruzes, D.S. und Conradi, R. (2015), The impact of global dispersion on coordination, team performance and software quality – A

systematic literature review, in: Information and Software Technology, 57, 2015, S. 277-294

Niemelä, E. und Immonen, A. (2007), Capturing quality requirements of product family architecture, in: Information and Software Technology, 49, 2007, 11-12, S. 1107-1120

Nikolik, B. (2012), Software quality assurance economics, in: Information and Software Technology, 54, 2012, 11, S. 1229-1238

(2005), Ninth European Conference on Software Maintenance and Reengineering - Proceedings : 21-23 March, 2005, Manchester, UK, Los Alamitos, Calif 2005

Okumoto, K. (1985), A Statistical Method for Software Quality Control, in: IEEE Transactions on Software Engineering, SE-11, 1985, 12, S. 1424-1430

Oquendo, F. (2016), Software Architecture Challenges and Emerging Research in Software-Intensive Systems-of-Systems, in: Tekinerdogan, Zdun, Babar (Hrsg., 2016), S. 3-21

Osterweil, L. J. (2006), Proceedings of the 28th international conference on Software engineering, New York, NY 2006

Paré, G., Trudel, M.-C., Jaana, M. und Kitsiou, S. (2015), Synthesizing information systems knowledge: A typology of literature reviews, in: Information & Management, 52, 2015, 2, S. 183-199

Paulk, M.C., Curtis, B., Chrissis, M.B. und Weber, C.V. (1993), Capability maturity model, version 1.1, in: IEEE Software, 10, 1993, 4, S. 18-27

Perez, J., Mens, T. und Kamseu, F. (2013), A Pilot Study on Software Quality Practices in Belgian Industry, in: Cleve (Hrsg., 2013), S. 395-398

Pfeifer, T. (2002), Produkt- und Prozessqualität in Netzwerken, in: Milberg, Schuh (Hrsg., 2002), S. 73-83

Pohl, K. (2008), Requirements engineering - Grundlagen, Prinzipien, Techniken, 2., korrigierte Aufl., Heidelberg 2008

Poore, J.H. (1988), Derivation of local software quality metrics (software quality circles), in: Software: Practice and Experience, 18, 1988, 11, S. 1017-1027

Porter, M.E. und Heppelmann, J.E. (2014), How Smart,Connected Products Are Transforming Competition, in: Harvard Business Review, 2014, 11, S. 1-23

(1985), Proceedings / 8th International Conference on Software Engineering, August 28 - 30, 1985, London, UK, Washington, DC 1985

(2002), Proceedings of the 2002 IEEE International Conference on Fuzzy Systems - FUZZ-IEEE'02 : May 12-17, 2002, Hilton Hawaiian Village Hotel, Honolulu, Hawaii, Piscataway, NJ 2002

(2002), Proceedings of the ICIS 2002 2002

(2012), Proceedings of the ISSRE 2012 2012

Quack, K. (1999), Projektplanung und Softwarequalität/Unternehmen sehen Softwarequalität als strategische Aufgabe, in: Computerwoche, 1999

Rai, A., Song, H. und Troutt, M. (1998), Software quality assurance: An analytical survey and research prioritization, in: Journal of Systems and Software, 40, 1998, 1, S. 67-83

Reformat, M., Pedrycz, W. und Pizzi, N.J. (2002), Software quality analysis with the use of computational intelligence, in: (Hrsg., 2002), S. 1156-1161

Rettig, M. (1990), Software Teams, in: Communications of the ACM, 33, 1990, S. 23-28

Ritz, J. (2018), Mobilitätswende - autonome Autos erobern unsere Straßen - Ressourcenverbrauch, Ökonomie und Sicherheit, Wiesbaden 2018

Roberts, D. J., Wilson, L. F. und Turner, S. J. (2004), Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications - Proceedings : Budapest, Hungary, October 21-23, 2004, Los Alamitos, Calif 2004

Rombach und Basili (1987), Quantitative Software-Qualitätssicherung: Eine Methode zur Definition und Nutzung geeigneter Maße, in: Informatik Spektrum, 1987, 10, S. 145-158

Rommel, G. (1995), Qualität gewinnt - Mit Hochleistungskultur und Kundennutzen an die Weltspitze, Stuttgart 1995

Rooijmans, J., Aerts, H. und van Genuchten, M. (1996), Software quality in consumer electronics products, in: IEEE Software, 13, 1996, 1, S. 55-64

Rowe, F. (2014), What literature review is not: diversity, boundaries and recommendations, in: European Journal of Information Systems, 23, 2014, 3, S. 241-255

Sarkar, S. und Panayappan, A. (2008), Formal architecture modeling of business application- software maintenance case study, in: (Hrsg., 2008), S. 1-6

Schmidt-Kessel, M., Kramme, M. F., Hauck, R., Urbach, N., van Erp, S., Loof, W., Röglinger, M., Maissen, E., Franceschi, A. de, Baier, D., Sänn, A., Pampel, J.,

Monego, D. D., Riehm, T., Bach, I. und Albrecht, C. (2017), Geschäftsmodelle in der digitalen Welt, Jena 2017

Schmitz, L. (1997), Software-Optimierung/"A fool with a tool is still a fool", in: Computerwoche, 1997

Schneiter, W. (1982), Gegen die Software-Lücke in den 80er Jahren, in: Computerwoche, 1982

Schönhofen, F., Schockert, S. und Herzwurm, G. (2018), ISO 16355 in software-intensive business, in: Hyrynsalmi, Suominen, Jud, Wang, Bosch, Münch (Hrsg., 2018), S. 158-163

Schwaber, K. und Beedle, M. (2002), Agile software development with Scrum, Pearson Education internal, Upper Saddle River, NJ 2002

(2015), Second ACM International Conference on Mobile Software Engineering and Systems - MOBILESoft 2015 - May 16-17, 2015, Florence, Italy, Piscataway, NJ 2015

(2015), Second ACM International Conference on Mobile Software Engineering and Systems - MOBILESoft 2015 - May 16-17, 2015, Florence, Italy, Piscataway, NJ 2015

Seidel, B. (1997), Bootstrap und Spice ergaenzen ISO-Norm, in: Computerwoche, 1997

(2018), SEKE 2018 - The 30th International Conference on Software Engineering & Knowledge Engineering : technical program, July 1-3, 2018, Hotel Pullman, Redwood City, San Francisco Bay, USA : proceedings, Pittsburgh, PA 2018

Siemers, C. (2011), Handbuch Embedded Systems Engineering 2011

Slaughter, S.A., Harter, D.E. und Krishnan, M.S. (1998), Evaluating the cost of software quality, in: Communications of the ACM, 41, 1998, 8, S. 67-73

Sneed, H. (1977), Methodik allein genügt nicht: - Plädoyer für einen Software-Prüfer, in: Computerwoche, 1977, 43

Soares, M.d.S., Vrancken, J. und Verbraeck, A. (2011), User requirements modeling and analysis of software-intensive systems, in: Journal of Systems and Software, 84, 2011, 2, S. 328-339

Software Engineering Institute (2002), Capability Maturity Model®Integration (CMMISM), Version 1.1 - CMMISM for Software EngineeringCarnegie Mellon University, Pittsburgh 2002

Söhnlein, S. (2018), Qualität und Zuverlässigkeit, in: Qualität und Zuverlässigkeit, 2018, 5, S. 48-51

Sommerlatte, T. (2007), Management von Spitzenqualität, 2., überarb. Aufl., Düsseldorf 2007

Sommerville, I. (2012), Software engineering, 9., aktualisierte Auflage, München, Harlow, Amsterdam 2012

Springer Nature Switzerland AG (2018), Software Quality Journal, Auf den Seiten von: , https://www.springer.com/journal/11219, Zugriff am 23.10.2019

Stahlknecht, P. und Hasenkamp, U. (2005), Einführung in die Wirtschaftsinformatik, 11., vollst. überarb. Aufl., 185. - 200. Tsd, Berlin 2005

Stamelos, I., Angelis, L., Oikonomou, A. und Bleris, G.L. (2002), Code quality analysis in open source software development, in: undefined, 2002

Steidl, D., Deissenboeck, F., Poehlmann, M., Heinke, R. und Uhink-Mergenthaler, B. (2014), Continuous Software Quality Control in Practice, in: (Hrsg., 2014), S. 561-564

Stirewalt, K., Egyed, A. und Fischer, B. (2007), Proceedings of the twenty-second IEEEACM international conference on Automated software engineering, New York, NY 2007

Stürze, M. (1998), Kommunikative Fähigkeiten sind gefragt, in: Computerwoche, 1998

Sunazuka, T., Azuma, M. und Yamagishi, N. (1985), Software quality assessment technology, in: (Hrsg., 1985),

Tekinerdogan, B., Zdun, U. und Babar, A. (2016), Software Architecture, Cham 2016

(2008), TENCON 2008 - IEEE Region 10 conference ; 19 - 21 Nov. 2008, Hyderabad, India, Piscataway, NJ 2008

Thaller, G.E. (2000), Software-Qualität - Der Weg zu Spitzenleistungen in der Software-Entwicklung, Berlin 2000

(1998), The Ninth International Symposium on Software Reliability Engineering - Proceedings : Paderborn, Germany, November 4-7, 1998, Los Alamitos, Calif 1998

Trammell, C.J. und Poore, J.H. (1992), A group process for defining local software quality: Field applications and validation experiments, in: Software: Practice and Experience, 22, 1992, 8, S. 603-636

Udoh, I.S. und Kotonya, G. (2018), Developing IoT applications: challenges and frameworks, in: IET Cyber-Physical Systems: Theory & Applications, 3, 2018, 2, S. 65-72

Urbach, N. (2017), Betriebswirtschaftliche Besonderheiten digitaler Güter, in: Schmidt-Kessel, Kramme, Hauck, Urbach, van Erp, Loof, Röglinger, Maissen, Franceschi, Baier, Sänn, Pampel, Monego, Riehm, Bach, Albrecht (Hrsg., 2017), Betriebswirtschaftliche Besonderheiten digitaler Güter

van Rompaey, B., Du Bois, B., Demeyer, S., Pleunis, J., Putman, R., Meijfroidt, K., Duenas, J.C. und Garcia, B. (2009), SERIOUS: Software Evolution, Refactoring, Improvement of Operational and Usable Systems, in: Ferenc (Hrsg., 2009), S. 277-280

Veselko, K. (2012), Nachhaltig erfolgreich, in: it&t-business, 2012, 12, S. 25

Vom Brocke, J., Simons, A., Niehaves, B., Riemer, K., Plattfaut, R. und Cleven, A. (2009), Reconstructing the giant: On the importance of rigour in documenting the literature search process, in: undefined, 2009

Wagner, S., Deißenböck, F., Feilkas, M. und Jürgens, E. (2008), Software-Qualit• atsmodelle in der Praxis - Erfahrungen mit aktivit• atenbasierten ModellenTUM, München 2008

Walia, G.S. und Carver, J.C. (2009), A systematic literature review to identify and classify software requirement errors, in: Information and Software Technology, 51, 2009, 7, S. 1087-1109

Wallmüller, E. (2007), SPI - Software Process Improvement mit CMMI, PSP/TSP und ISO 15504, 1. Aufl., s.l. 2007

Wallmüller, E. (2011), Software Quality Engineering - Ein Leitfaden für bessere Software-Qualität ; [basiert auf den standardisierten Wissenssammlungen für Software-Qualität von ASQ und JSQC ; ideal für die Vorbereitung auf die CSQ- und QAMP-Zertifizierung, 3. Aufl., München 2011

Walstrom, K.A., Hardgrave, B.C. und Wilson, R.L. (1995), Forums for management information systems scholars, in: Communications of the ACM, 38, 1995, 3, S. 93-107

Wang, Q., Zhu, J. und Yu, B. (2007), Feature Selection and Clustering in Software Quality Prediction, in: EASE, 2007

Webster, J. und Watson, R. (2002), ANALYZING THE PAST TO PREPARE FOR THE FUTURE: WRITING A LITERATURE REVIEW, in: Management Information Systems Quarterly, 26, 2002, 2

Wegener, J., Sthamer, H., Jones, B.F. und Eyres, D.E. (1997), Testing real-time systems using genetic algorithms, in: Software Quality Journal, 6, 1997, 2, S. 127-135

Werder, K., Brinkkemper, S., Bosch, J., Cusumano, M.A., Herzwurm, G. und Holmström (Hrsg., 2018, Working Group on Software-Intesive Business Research - Definition and Roadmap Abrahamsson, Bosch, Brinkkemper, Mädche: Software Business, Platforms, and Ecosystems: Fundamentals of Software Production Research (Dagstuhl Seminar 18182). 35 pages (8), S. 193-196

Westermann, D., Happe, J., Krebs, R. und Farahbod, R. (2012), Automated inference of goal-oriented performance prediction functions, in: Goedicke, Menzies, Saeki (Hrsg., 2012), S. 190

Whittemore, R. und Knafl, K. (2005), The integrative review: updated methodology, in: Journal of advanced nursing, 52, 2005, 5, S. 546-553

Wieringa, R., Maiden, N., Mead, N. und Rolland, C. (2005), Requirements engineering paper classification and evaluation criteria: a proposal and a discussion, in: Requirements Engineering, 11, 2005, 1, S. 102-107

Willmer, H. (1985), Systematische Software-Qualitätssicherung anhand von Qualitäts- und Produktmodellen 1985

Wintersteiger, W. (1984), QS-Maßnahmen nur zäh zu realisieren, in: Computerwoche, 1984

Wintersteiger, W. (1987), Halbherziger Ansatz bringt oft mehr Schaden als Nutzen, in: Computerwoche, 1987

Wirsing, M. (2004), Engineering Software-Intensive Systems - Report on the EU/NSF Strategic Workshop on, 2004

Wirsing, M. (2008), Software-intensive systems and new computing paradigms - Challenges and visions, Berlin 2008

Wu, W., Cai, Y., Kazman, R., Mo, R., Liu, Z., Chen, R., Ge, Y., Liu, W. und Zhang, J. (2018), Software Architecture Measurement—Experiences from a Multinational Company, in: Cuesta, Garlan, Pérez (Hrsg., 2018), S. 303-319

Xenos, M. und Christodoulakis, D. (1997), Measuring perceived software quality, in: Information and Software Technology, 39, 1997, 6, S. 417-424

Xiao, Y. und Watson, M. (2019), Guidance on Conducting a Systematic Literature Review, in: Journal of Planning Education and Research, 39, 2019, 1, S. 93-112

Yan, M., Zhang, X., Liu, C., Zou, J., Xu, L. und Xia, X. (2017), Learning to Aggregate: An Automated Aggregation Method for Software Quality Model, in: (Hrsg., 2017), S. 268-270

Yau, S.S. und Collofello, J.S. (1980), Some Stability Measures for Software Maintenance, in: IEEE Transactions on Software Engineering, SE-6, 1980, 6, S. 545-552

Yeh, R. und Ramamoorthy, C.V. (1976), ICSE '76 Proceedings of the 2nd international conference on Software engineering 1976

Zhang, H. und Cheung, S.C. (2013), A cost-effectiveness criterion for applying software defect prediction models, in: Meyer, Baresi, Mezini (Hrsg., 2013), S. 643

Zhang, H. und Kim, S. (2010), Monitoring Software Quality Evolution for Defects, in: IEEE Software, 27, 2010, 4, S. 58-64

Zollondz, H.-D. (2006), Grundlagen Qualitätsmanagement - Einführung in Geschichte, Begriffe, Systeme und Konzepte, 2., vollst. überarb. und erw. Aufl., München 2006

Zollondz, H.-D., Ketting, M. und Pfundtner, R. (2016), Lexikon Qualitätsmanagement - Handbuch des Modernen Managements auf Basis des Qualitätsmangements, 2., komplett überarbeitete und erweiterte Auflage 2016