MASTERARBEIT

UNTERSUCHUNG DER WECHSELSEITIGEN WIRKUNGEN VON UNTERNEHMENSKULTUR UND SOFTWAREARCHITEKTUR

ausgeführt an der

am Studiengang
Software Engineering Leadership

Von: André Hester
Personenkennzeichen: 1440030004

Münster, am 28.7.2017 .....
Unterschrift

EHRENWÖRTLICHE ERKLÄRUNG

Ich erkläre ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die benutzten Quellen wörtlich zitiert sowie inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

.....

Unterschrift

DANKSAGUNG

Zunächst möchte ich mich bei allen Personen bedanken, die mich bei der Anfertigung der Masterarbeit unterstützt und diese möglich gemacht haben. Dazu zähle ich insbesondere meine Kommilitonen und Dozenten des Studiengangs Software Engineering Leadership, die durch ihre zahlreichen Fachgespräche und Beiträge während und außerhalb der Studienzeit mein Interesse an Agilität, Kultur in Unternehmen und Softwarearchitektur verstärkt haben.

Ein besonderer Dank gilt meinem Betreuer Dr. Michael Hofmann von oose für die Unterstützung bei der Themenfindung und Anfertigung der vorliegenden Arbeit.

Ebenso möchte ich mich bei meinen Freunden Jan und Fabian bedanken. Durch unsere Diskussionen, eure Meinungen sowie eure Unterstützung während der Anfertigung meiner Masterarbeit habt ihr mich bestärkt und motiviert. Vielen Dank!

Meinem Geschäftspartner Peter danke ich für die eingeräumten Freiräume und den Rückhalt während meines Studiums als auch während der Anfertigung meiner Masterarbeit.

Ein weiterer besonderer Dank gilt meiner Freundin Anika: Vielen Dank für deine Motivation, deine Unterstützung, dein Zuhören, deinen Rückhalt und die Freiräume, die du mir während des Studiums und Anfertigung der Masterarbeit gegeben hast. Danke für alles!

KURZFASSUNG

Unternehmen müssen heute schnell am Markt agieren und auf die Bedürfnisse und Anforderungen der Kunden eingehen, um wettbewerbsfähig zu bleiben. Um diesen Anforderungen als Unternehmen gerecht werden zu können, bedarf es innovativen Informationssystemen mit flexiblen Softwarearchitekturen. Die Unternehmenskultur kann in Unternehmen eine starke Position einnehmen, wenn Veränderungen in der Unternehmensstrategie vorgenommen werden. Die Veränderung oder Einführung einer neuen Softwarearchitektur eines Informationssystems stellt hierbei eine solche Veränderung in der Strategie dar. Diese Arbeit beantwortet die Frage, wie sich die Kultur in einem Unternehmen und Softwarearchitektur gegenseitig beeinflussen.

Hierzu werden zunächst die theoretischen Grundlagen zu Unternehmenskultur und Softwarearchitektur dargelegt. Anschließend wird der mögliche Einfluss der Unternehmenskultur auf die Softwarearchitektur mittels fünf Dimensionen der Unternehmenskultur beschrieben und dadurch deduktiv 59 Hypothesen zur Wirkungsbeziehung generiert.

Das Ergebnis der Arbeit ist, dass Unternehmenskultur durch die gemeinsam geteilten Werte das Verhalten des Softwarearchitekten im Architekturzyklus und somit schließlich auch die entworfene Softwarearchitektur beeinflusst. Ebenso stellt die Unternehmenskultur ausgehend von verschiedenen Dimensionen Qualitätsanforderungen an die Softwarearchitektur. Der mögliche Einfluss der Softwarearchitektur auf die Unternehmenskultur wird in der Arbeit am Beispiel des Microservice-Architekturstils beschrieben. Die Einführung oder Veränderung einer Softwarearchitektur führt zu einer Veränderung der Unternehmenskultur, in dem sie gegenwärtige grundlegende Annahmen und Werte in Frage stellt. Ebenso kann eine neue oder veränderte Softwarearchitektur sich auf die Rahmenbedingungen in der Ablauf- und Aufbauorganisation des Unternehmens auswirken. So wird gezeigt, dass das Modularisierungskonzept des Microservice-Architekturstils verschiedene Dimensionen der Unternehmenskultur positiv begünstigen kann.

ABSTRACT

Todays companies need to operate fast within markets and cater for the customers needs and requirements to continue beeing competitive to other companies on the market. To meet the customers requirements the companies need innovative information systems with an flexible software architecture. The company culture can take a strong position within companies if the company changes it´s corporate strategy. The change or introduction of an information systems software architecture acts as a strategy change. This master thesis answers the question how company culture and software culture influence each other.

First the theory of company culture and software architecture are introduced. After that the possible influences of corporate culture and software architecture are described through five dimensions of corporate structure and 59 hypothesis for the influence are generated.

The thesis result ist that corporate culture influences the software architects behaviour during the architecture cycle and following the software architecture, too. Additionally the corporate culture adds some quality requirements proceeding from different dimensions of the corporate culture to the software architecture. The influence of the software architecture on the corporate culture is described in this thesis by the microservice architecture style. The change or introduction of a software architecture influences the corporate culture by challeging the staffs basic beliefs and values. Further the software architecture influences the basic conditions of company structure and processes. This thesis also shows that the microservice style can benefit to some corporate culture dimensions.

GLEICHHEITSGRUNDSATZ

Aus Gründen der Lesbarkeit wurde in dieser Arbeit darauf verzichtet, geschlechtsspezifische Formulierungen zu verwenden. Jedoch möchte ich ausdrücklich festhalten, dass die bei Personen verwendeten maskulinen Formen für beide Geschlechter zu verstehen sind.

INHALTSVERZEICHNIS

1 EINLEITUNG 1

1.1 Zielsetzung 1
1.2 Aufbau der Arbeit 2

2 UNTERNEHMENSKULTUR 3

2.1 Definition von Kultur 3
2.2 Definition und Inhalt von Unternehmenskultur 3
2.3 Forschungsperspektiven zur Betrachtung von Unternehmenskultur 6
2.3.1 Objektivistischen Forschungsperspektive 6
2.3.2 Subjektivistische Forschungsperspektive 6
2.3.3 Integrative Forschungsperspektive 6

2.4 Elemente von Unternehmenskultur 7
2.4.1 Drei-Ebenen-Modell der Unternehmenskultur nach Schein 8

2.5 Funktionen der Unternehmenskultur 9
2.5.1 Sensibilisierungsfunktion 10
2.5.2 Abgrenzungsfunktion 10
2.5.3 Identifikationsfunktion 11
2.5.4 Orientierungsfunktion 11
2.5.5 Koordinations- und Steuerungsfunktion 11
2.5.6 Stabilisierungsfunktion 12
2.5.7 Erfolgssteigerungsfunktion 12
2.5.8 Innovationsfunktion 12

2.6 Dimensionen und Werte von Unternehmenskulturen 12
2.6.1 Zielorientierung 13
2.6.1.1. Vision 14
2.6.1.2. Mission 16
2.6.1.3. Leitbild 16

2.6.1.4. Strategie 16
2.6.1.5. Zusammenfassung 19
2.6.2 Kommunikation und Information 20
2.6.2.1. Probleme in der Kommunikation 21
2.6.2.2. Werte der Kommunikation 22
2.6.3 Führung und Mitarbeiterorientierung 22
2.6.4 Kundenorientierung 24
2.6.5 Innovationsorientierung 25

3 SOFTWAREARCHITEKTUR 28

3.1 Definition und Inhalt von Softwarearchitektur 28
3.1.1 Abgrenzung von Softwarearchitektur zu anderen Architekturen 30

3.2 Ziele und Aufgaben von Softwarearchitektur 30
3.3 Einflüsse auf Softwarearchitektur 31
3.3.1 Organisatorische Einflussfaktoren 32
3.3.2 Technische Einflussfaktoren 34
3.3.3 System- und Produktfaktoren 35
3.3.4 Wissen des Softwarearchitekten 36

3.4 Entstehung von Softwarearchitekturen 37
3.5 Rolle des Softwarearchitekten 38
3.5.1 Besetzung der Rolle 39
3.5.2 Aufgaben des Softwarearchitekten 41
3.5.2.1. Aufgaben in der Analysephase 42
3.5.2.2. Aufgaben in der Architekturphase 44
3.5.2.3. Aufgaben in der Reflexionsphase 45
3.5.2.4. Aufgaben in der Umsetzungsphase 47
3.5.2.5. Aufgaben in der Review-Phase 47

3.6 Microservice-Architekturen 47
3.6.1 Monolithische Systeme vs. Microservices 49
3.6.2 Microservices und Conways´s Law 50

3.6.3 Unterstützende Techniken für Microservice-Architekturen 52

4 UNTERNEHMENSKULTUR UND SOFTWAREARCHITEKTUR 54

4.1 Mögliche Auswirkungen der Unternehmenskultur auf die Softwarearchitektur 56
4.1.1 Zielorientierung 58
4.1.2 Kommunikation und Information 61
4.1.3 Führung und Mitarbeiterorientierung 64
4.1.4 Kundenorientierung 65
4.1.5 Innovationsorientierung 69

4.2 Mögliche Wirkungen der Softwarearchitektur auf die Unternehmenskultur 70
4.2.1 Zielorientierung 72
4.2.2 Kommunikation und Information 72
4.2.3 Führung und Mitarbeiterorientierung 73
4.2.4 Kundenorientierung 74
4.2.5 Innovationsorientierung 75

5 FAZIT 77

5.1 Zusammenfassung 77
5.2 Diskussion der Ergebnisse 79
5.3 Ausblick 80

ABBILDUNGSVERZEICHNIS 82

TABELLENVERZEICHNIS 83

LITERATURVERZEICHNIS 84

1 EINLEITUNG

„Culture eats strategy for breakfast“

(Peter Drucker)

Unternehmen müssen heute schnell am Markt agieren und auf die Bedürfnisse und Anforderungen der Kunden eingehen, um wettbewerbsfähig zu bleiben. Was mit einem Unternehmen geschehen kann, wenn man dem nicht nachkommt, kann man am ehemaligen Kamera- und Foto-Spezialisten Kodak beobachten, der einst 140.000 Mitarbeiter beschäftigte und einen jährlichen Umsatz von rund 28 Milliarden Dollar erwirtschaftete: Kodak verpasste den digitalen Wandel und musste schließlich sogar Insolvenz anmelden (Abolhassan, 2016).

Der Kunde möchte heute individuelle Produkte. Ob analoges oder digitales Geschäftsmodell - die Digitalisierung ermöglicht es bei gleichzeitig schnelleren Produktionszyklen (Von Heynitz, 2017). Um den genannten Anforderungen als Unternehmen gerecht werden zu können, bedarf es innovativen Informationssystemen mit flexiblen Softwarearchitekturen, mit denen Veränderungen sowohl schnell als auch verlässlich vorangetrieben und umgesetzt werden können (Urbach & Ahlemann, 2017). Ein Architekturstil, der seit 2014 an Betrachtung gewonnen hat, sind Microservices (Google Trends, 2017). Diejenigen Unternehmen, die diesen Architekturstil einsetzen, berichten von einer fünffach schnelleren Auslieferungszeit als mit herkömmlichen Architekturen. Diesen Vorteil wollen sich weitere Unternehmen zu Nutze machen und so planen laut einer Studie von LeanIX in diesem Jahr 70% der befragten Teilnehmer ihre IT-Systeme mit einer Microservice-Architektur umsetzen (LeanIX, 2016).

Eine starke Unternehmenskultur führt zu leistungsfähigen Unternehmen (Deal & Kennedy, 1982), insbesondere wenn die Überzeugungen der Mitarbeiter mit denen der Kultur übereinstimmen (Cameron & Freeman, 1985). Das einführende Zitat des renommierten Betriebswirtschaftlers Peter Drucker verdeutlicht, welch starke Position Kultur in Unternehmen einnehmen kann, wenn Veränderungen in der Unternehmensstrategie vorgenommen werden. Die Veränderung hin zum Einsatz des genannten Microservices-Architekturstils zur Umsetzung von Informationssystemen stellt hierbei eine solche Veränderung in der Strategie dar. Es stellt sich die Frage, wie sich die Kultur in einem Unternehmen und Softwarearchitektur gegenseitig beeinflussen. Die vorliegende Arbeit möchte Antwort liefern und setzt sich mit der genannten Fragestellung auseinander.

1.1 Zielsetzung

Diese Arbeit soll dazu beitragen mehr Verständnis über das Zusammenwirken der Themengebiete Unternehmenskultur und Softwarearchitektur zu erlangen, indem diese theoretisch aufge-

arbeitet und zusammengeführt werden. Im Rahmen dieser Untersuchung sollen die nachfolgenden Forschungsfragen beantwortet werden. Ziel ist es deduktiv Hypothesen zu wechselseitigen Wirkungsbeziehungen zwischen der Kultur eines Unternehmens und des Softwarearchitekten, insbesondere auch der von ihm entworfenen Softwarearchitektur, zu generieren (vgl. Abbildung 1-1).

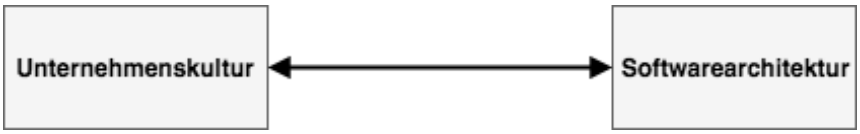

Abbildung 1-1: Untersuchungsgegenstand der vorliegenden Arbeit

Die zentrale Forschungsfrage lautet:

- Wie beeinflussen sich Unternehmenskultur und Softwarearchitektur gegenseitig?

Daraus abgeleitet ergeben sich die folgenden Forschungsfragen:

- Wie wirkt die Unternehmenskultur auf die Softwarearchitektur?
- Wie wirkt die Softwarearchitektur auf die Unternehmenskultur?

1.2 Aufbau der Arbeit

Die vorliegende Arbeit untergliedert sich in fünf Kapitel. Um das in Kapitel 1.1 genannte Ziel der Arbeit erreichen zu können werden zu Beginn die Themen Unternehmenskultur und Softwarearchitektur definiert und ihre Bestandteile erläutert. Nach der Einleitung wird in Kapitel 2 der Themenkomplex Unternehmenskultur eingeführt, seine Elemente und Funktionen dargestellt sowie einige ausgewählte Dimensionen von Unternehmenskultur beschrieben. Das darauf folgende Kapitel 3 beschreibt das Themengebiet Softwarearchitektur mit der zentralen Rolle des Softwarearchitekten und dessen Aufgaben im Lebenszyklus einer Softwarearchitektur. Das Kapitel 4 führt die Untersuchungsgegenstände Unternehmenskultur und Softwarearchitektur zusammen. Dort werden zur Beantwortung der Forschungsfragen zuerst mögliche Auswirkungen von Unternehmenskultur auf die Softwarearchitektur beschrieben. Anschließend werden die möglichen Auswirkungen von Softwarearchitektur auf die Unternehmenskultur am Beispiel des Microservice-Architekturstils beschrieben. In Kapitel 5 werden die Ergebnisse der Arbeit zusammengefasst und ein Ausblick auf weitere Forschungsbereiche gegeben.

2 UNTERNEHMENSKULTUR

Um den weiträumigen Begriff der Unternehmenskultur greifbar machen zu können, ist es zunächst lohnenswert sich näher mit dem abstrakten Begriff der Kultur auseinanderzusetzen. Anschließend wird auf die Kultur von Unternehmen, sowie deren Elemente und Funktionen eingegangen. Zum Ende des Kapitels werden Dimensionen von Unternehmenskultur vorgestellt, die relevant für den weiteren Verlauf der Arbeit sind.

2.1 Definition von Kultur

Kultur ist ein komplexes Phänomen, das ein weites Feld umfasst. Bereits 1990 existierten über 250 Begriffsbestimmungen des Wortes Kultur (Krulis-Randa, 1990). „Der Begriff der Kultur zeichnet sich im heutigen Wortverständnis durch eine Fülle von Bedeutungsinhalten aus. So ist bereits für den deutschen Sprachgebrauch eine beinahe verwirrend große Zahl von Verwendungen festzustellen. Dies zeigt sich insbesondere in den vielfältigen Möglichkeiten seiner Verknüpfung mit anderen Begriffen.“ (Dormayer & Kettnern, 1987, S. 50) So findet man den Begriff in den unterschiedlichsten Zusammenhängen wie beispielsweise Landeskultur, Esskultur, Feedback-Kultur, wo er abhängig vom Kontext unterschiedliche Bedeutungsinhalte hat.

Nach der Analyse von über 150 Kulturdefinitionen extrahierten Kroeber und Kluckhohn (1952) die essentiellsten Bestandteile der Definition. Kultur ist demnach ein dynamischer und variabler Prozess. Sie wird erlernt, leitet sich aus der menschlichen Vergangenheit ab und dient dem Individuum zur Anpassung an die Umwelt. Kultur lässt sich in einzelne Bestandteile zerlegen und zeigt wissenschaftlich erforschbare Regelmäßigkeiten. Im Laufe der Zeit entsteht Kultur als spezifisches Denkmuster innerhalb einer Gruppe von Menschen. Sie gibt den Menschen Halt und Richtungsinformationen und trägt zur innerlichen Stabilisierung sowie Charakterisierung bei (Scholz & Hofbauer, 1990).

2.2 Definition und Inhalt von Unternehmenskultur

Das Kulturkonzept wurde zu Beginn der 80er Jahre auf den organisationalen Kontext übertragen. Unternehmenskultur ist nach Deal und Kennedy (1982) die Mischung aus Werten, Mythen, Helden und Symbolen, die den Leuten, die im Unternehmen arbeiten, viel bedeuten,

nach Hofstede (1980) „die kollektive Programmierung des menschlichen Denkens, erworben im Laufe des Lebens, die die Mitglieder einer Gruppe von Menschen von denjenigen einer anderen Gruppe unterscheiden“ (S. 1168),

nach Schein (2006) die „Summe aller gemeinsamen, selbstverständlichen Annahmen, die eine Gruppe in ihrer Geschichte erlernt hat“ (S. 44),

sowie nach Sackmann (2004) die „grundlegenden, kollektiven Überzeugungen, die das Denken, Handel und Empfinden der Führungskräfte und Mitarbeiter im Unternehmen maßgeblich beeinflussen und die insgesamt sehr typisch für das Unternehmen bzw. eine Gruppe im Unternehmen sind“ (S. 24).

Zusammengefasst können diese Definitionen als die gemeinsam akzeptierten und gelebten Werte und Normen einer Unternehmung verstanden werden. Diese sind spezifisch für das Unternehmen und auch unabhängig von Modetrends, Präferenzen, Erfolg oder Misserfolg bei jedem Unternehmen vorhanden. Sie entstehen mit Gründung des Unternehmens und entwickeln sich mehr oder weniger stark in der Geschichte des Unternehmens (Sackmann, 2004).

Bei der Unternehmenskultur geht es nicht um „gut“ oder „schlecht“. Es zählt nach Homma, Bauschke und Hofmann (2014) „einzig und allein, ob die Kultur im Wesentlichen die Zielsetzungen des Unternehmens unterstützt und dabei genügend Flexibilität sowohl nach außen als auch nach innen besitzt, um auf relevante Veränderungen reagieren zu können“ (S. 11).

Die genannten abstrakten Definitionen tragen nicht dazu bei, Kulturen inhaltlich verstehen zu können. Die Abbildung 2-1 zeigt deshalb, welche Bereiche Unternehmenskultur abdecken kann. Nachfolgend werden nach Schein (2006) alle Bereiche skizziert, in denen kulturelle Annahmen wichtig sind.

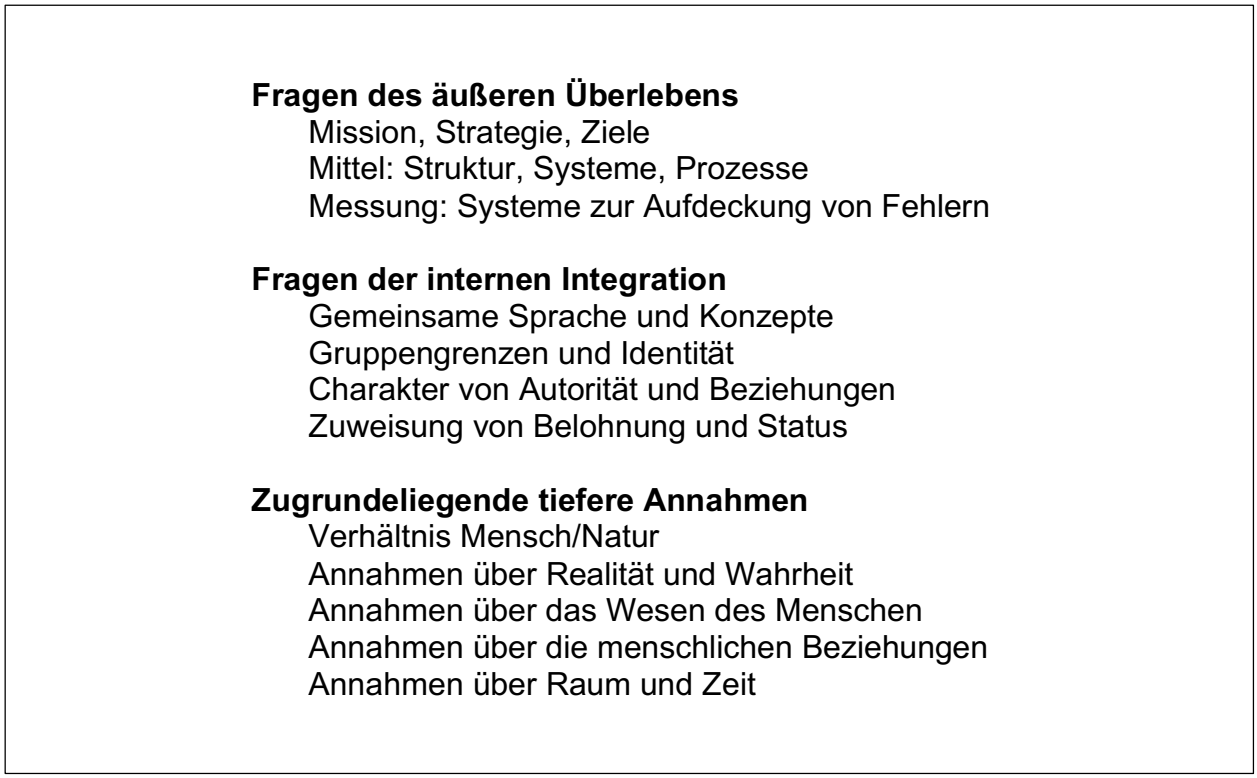

Abbildung 2-1: Inhalte von Unternehmenskultur [Eine Darstellung in Anlehnung an (Schein, 2006, S. 45)]

Damit ein Unternehmen seine Aufgabe erfüllen, überleben, wachsen und dadurch auch Erfolg haben kann, muss es den Anforderungen und Möglichkeiten seines Umfelds gerecht werden. Dazu muss es Annahmen über seine grundlegenden Aufgaben, Strukturen und Abläufe entwickeln. Abweichungen von den Annahmen müssen systematisch erfasst und korrigiert werden können. Die ersten Annahmen über den Kern des Unternehmens, über die **Ziele** und die daraus abgeleitete **Mission** und **Strategie** werden durch die Gründer oder ersten Leiter eines Unternehmens getroffen. Ist das Unternehmen erfolgreich, so entwickeln sich auf der Basis der anfänglichen Überzeugungen gemeinsame Annahmen bei Gründern und Mitarbeitern. Die gemeinsamen Annahmen werden mit der Zeit selbstverständlich, so dass sie nur bewusst werden, wenn jemand gegen sie verstößt (Schein 2006).

Mittel wie **Strukturen, Systeme und Prozesse** bilden die inhaltliche Ebene, auf der das Unternehmen seine Strategie und Ziele implementiert. Wenn das Unternehmen erfolgreich ist und die Struktur zu den Aufgaben und dem Umfeld des Unternehmens passt, dann bilden sich unausgesprochene Annahmen über die Organisationsform. In einem wachsenden Unternehmen können in einzelnen Teilbereichen Subkulturen entstehen, die eine andere Organisationsform wählen, um ihre festgelegten Ziele zu erreichen. Zur **Selbstevaluation** werden Systeme zum **Entdecken und Korrigieren von Fehlern** innerhalb der Strukturen, Systeme und Prozesse geschaffen. Als Indikatoren für die Fehler und Abweichungen können beispielsweise das finanzielle Ergebnis, der Aktienpreis oder das Verhältnis von Schulden zum Kapitel dienen. Die Art der gesammelten Informationen zur Evaluation und deren Interpretation als auch die Definition von Varianzen und Fehlern variieren und sind geprägt von kulturellen Annahmen (Schein 2006).

Neben den vorangegangenen Annahmen darf der menschliche Faktor nicht vernachlässigt werden. Kultur äußert sich am sichtbarsten über eine gemeinsame **Sprache, Konzepte, Denk- und Verhaltensweisen**, die in jedem Unternehmen einzigartig sind. Das macht es neuen Mitarbeitern schwer die Abkürzungen, Jargons, Normen, Arbeits- und Denkweisen im Unternehmen auf einfachem Wege zu erlernen, was dann schließlich nur durch Versuch und Irrtum funktioniert (Schein 2006).

Status und Zugehörigkeit lassen sich in Unternehmen durch unterschiedliche Formen wie beispielsweise Arbeitsausstattung, Parkplatzzuordnung, Aktienanteile und andere Vergünstigungen identifizieren. Neue Mitarbeiter müssen die Sprache und Denkweisen des Unternehmens lernen, damit sie mit einbezogen und zugehörig werden. Nach erworbenem Vertrauen werden ihnen dann auch Geheimnisse anvertraut, die Verpflichtungen zu Loyalität, die Geheimnisse zu wahren und höherem Engagement mit sich bringen. Ebenso sind die Annahmen über Autoritätsbeziehungen und das richtige Maß an Offenheit und Intimität zwischen den Mitarbeitern und Vorgesetzten in Unternehmen sehr unterschiedlich. Dies betrifft beispielsweise Offenbarungen zum Privat- und Familienleben oder die Anrede von Kollegen mit dem Vornamen. Auch die Bedeutung von **Belohnungs- und Statussystemen** ist in jedem Unternehmen unterschiedlich ausgeprägt. Die Belohnungen sind sehr variabel und spiegeln sich beispielsweise monetär über Gehalt, Gehaltserhöhungen, Aktienoptionen, Gewinnbeteiligungen, Gratifikationen oder Beförderungen wider. Als Statussymbole können Titel, Projektbudget, -umfang und -verantwortung oder die Zahl der untergebenen Mitarbeiter gelten. Neue Mitarbeiter müssen

neben der Sprache und der Denkmuster auch diese Systeme entschlüsseln, um ihre Leistung reflektieren und einschätzen zu können (Schein 2006).

2.3 Forschungsperspektiven zur Betrachtung von Unternehmenskultur

In der Unternehmenskulturforschung existieren nebeneinander zwei konkurrierende Perspektiven: Die objektivistische und subjektivistische Perspektive. Aus der Synthese dieser zwei Perspektiven wurde eine dritte Perspektive, die integrative Perspektive geschaffen. Im Folgenden werden die grundlegenden Sichten der drei Perspektiven betrachtet und in Anlehnung an beschrieben. Abschließend wird die für diese Arbeit relevante Perspektive bestimmt.

2.3.1 Objektivistischen Forschungsperspektive

*„ein Unternehmen **hat** eine Kultur“*

Die Anhänger der objektivistischen Forschungsperspektive nehmen an, dass die Unternehmenskultur eine von mehreren beeinflussbaren Variablen ist, die einen Beitrag zur Erklärung des Unternehmenserfolgs liefern. Unternehmenskultur als Variable existiert demnach als ein normatives Subsystem neben anderen Führungsteilsystemen wie beispielsweise der Strategie oder der Struktur des Unternehmens im sozialen System Unternehmen (Lippold, 2007; Fischl (2008).

2.3.2 Subjektivistische Forschungsperspektive

*„ein Unternehmen **ist** eine Kultur“*

Die Unternehmenskultur ist in der subjektivistischen Forschungsperspektive als eine von den Unternehmensmitgliedern gemeinsam konstruierte Wirklichkeit und als nicht real existent zu sehen. Die Unternehmenskultur ist demnach der Ausdruck des menschlichen Bewusstseins, die interpretiert, aber nicht beeinflusst werden kann (Lippold, 2007; Fischl (2008).

2.3.3 Integrative Forschungsperspektive

*„Unternehmen **sind** Kultur und **haben** auch kulturelle Aspekte“*

Die integrative Forschungsperspektive als Synthese der objektivistischen und subjektivistischen Ansätze ist die weitverbreitetste Auffassung von Unternehmenskultur. Demnach ist ein Unternehmen nach Auffassung der subjektivistischen Perspektive eine Kultur, die aber auch beeinflusst und gestaltet werden kann im Sinne der objektivistischen Perspektive. Unternehmenskultur besteht aus materiellen und ideellen Ebenen, die in komplexer und multikausaler Weise interagieren. Zu den bekanntesten Vertretern der integrativen Forschungsperspektive zählen Schein und Sackmann (Lippold, 2007; Fischl (2008).

In dieser Arbeit soll die Unternehmenskultur nicht operationalisiert und empirisch ausgewertet werden, wie es in vielen anderen Arbeiten zum Thema Unternehmenskultur durchgeführt wurde, sondern es wird der wechselseitige Einfluss von Unternehmenskultur und Softwarearchitektur untersucht und daraus resultierend Hypothesen abzuleiten. Für das Verständnis von Unternehmenskultur wird daher auf die Modelle mit integrativer Forschungsperspektive von Schein und Sackmann aufgebaut.

2.4 Elemente von Unternehmenskultur

Werte, Grundannahmen, Einstellungen und Normen sind die unsichtbaren Elemente von Unternehmenskultur und zeichnen diese aus:

Werte sind ein essentieller Bestandteil von Unternehmenskulturen und beschreiben die als moralisch gut empfundenen Eigenschaften. Diese Eigenschaften sind erstrebenswert und werden von anderen erwünscht als auch erwartet. Sie symbolisieren oder beschreiben die Qualität von Charaktereigenschaften bzw. Sittlichkeit (Subjekte) sowie die Nutzenmerkmale von Dingen bzw. Produkten (Objekte). Werte beeinflussen als implizite Regeln die Entscheidungen in einem Unternehmen (Scholz & Hofbauer, 1990).

Verhalten ist definiert als „jenes Geschehen, das, an einem Organismus oder von einem Organismus ausgehend, außenseitig wahrnehmbar ist“ (Wenninger, 2000). Verhalten ist durch Werte geprägt. Wenn diese Werte zum Erfolg führen und sich bewähren, dann werden sie in tieferliegende Gedankenschichten geführt, wo sie zu **Grundannahmen** werden. Grundannahmen beeinflussen das Verhalten als selbstverständliche und akzeptierte Antworten auf bestimmte Reize noch stärker als Werte (Scholz & Hofbauer, 1990).

Einstellungen beziehen sich auf bestimmte Objekte, Personen oder Situationen. Diese Einstellungen können sich kurzfristig auch sehr leicht ändern. Werte und Grundannahmen hingegen sind situationsübergreifend und zentraler als Einstellungen. Im Gegensatz zu den Einstellungen gibt es in einem Unternehmen relativ wenig Werte, die aber von zentraler Bedeutung sind und durch ihre langfristige Konstanz für die innere Stabilität sorgen. (Scholz & Hofbauer, 1990)

Erwartungen innerhalb eines Unternehmens gegenüber den Handlungen und dem Verhalten von Mitarbeitern werden auch als **Normen** bezeichnet. Das zur Wertung verwendete Schema differiert in jeder Unternehmenskultur. Was als angemessen, ideal oder richtig bewertet wird, ist nicht bei jedem Unternehmen identisch (Scholz & Hofbauer, 1990).

Die Beziehungen der Elemente werden im nächsten Abschnitt anhand des Drei-Ebenen-Modells von Schein und dem Eisbergmodell von Sackmann genauer erläutert.

2.4.1 Drei-Ebenen-Modell der Unternehmenskultur nach Schein

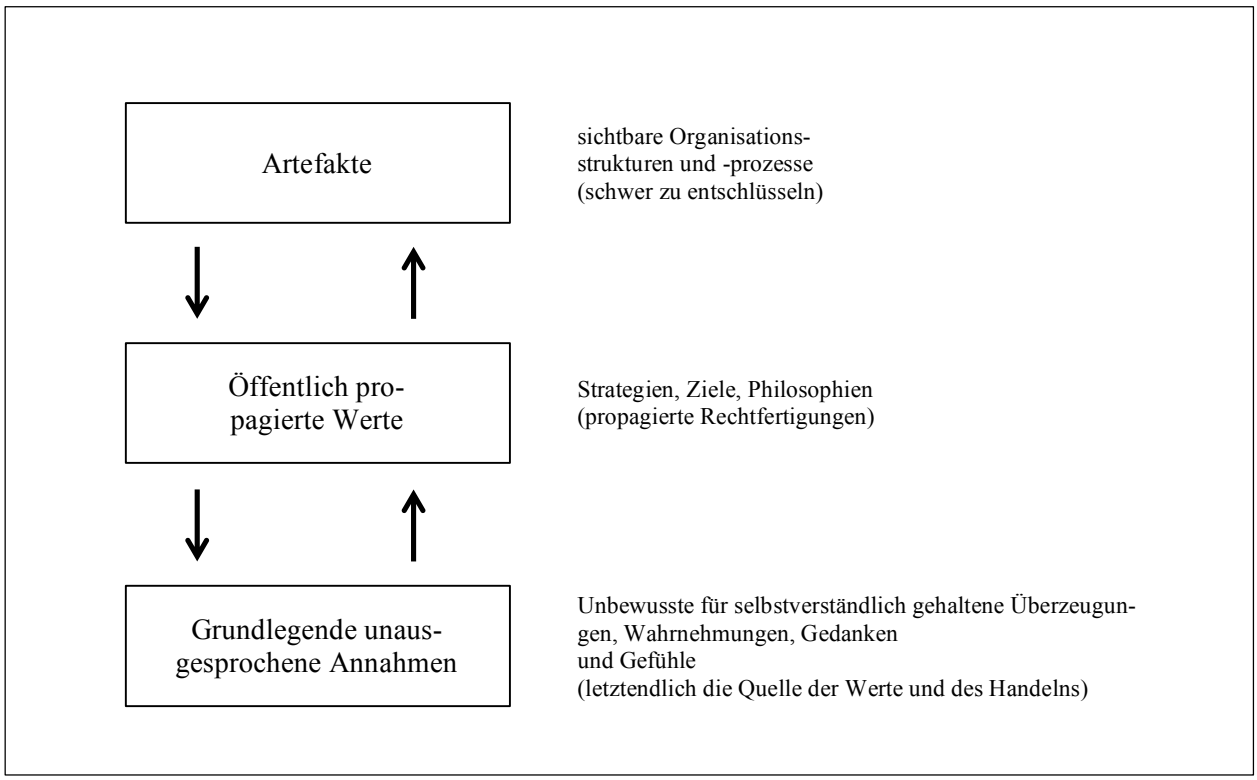

Abbildung 2-2: Die drei Ebenen der Unternehmenskultur nach Schein (2006)

Nach Schein (2006) existieren, wie in Abbildung 2-2 dargestellt, drei Ebenen der Unternehmenskultur. Die oberste Ebene der **Artefakte** enthält alle Elemente die nach außen sichtbar sind. Sichtbare Elemente sind Riten, Rituale, Mythen, Statussymbole als auch die Corporate Identity des Unternehmens, weshalb man sie auch als Symbolebene bezeichnen kann. Diese Ebene der Unternehmenskultur ist sehr klar und hat unmittelbare emotionale Auswirkungen.

Die mittlere Ebene enthält **Werte, Moral, Normen und Visionen**. Diese können beispielweise Teamarbeit, Konsens bei Entscheidungen, Kundenordientierung und Produktqualität sein und sind auf Flyern, Webseite veröffentlicht oder hängen als Leitbild verfasst und gedruckt im Foyer des Unternehmens. Mehrere Unternehmen können identische Werte über unterschiedliche Artefakte öffentlich propagieren. Jedoch können diese Werte widersprüchlich zum tatsächlichen Verhalten des Unternehmens sein, da das Verhalten von einer tieferen Denk- und Wahrnehmungsebene gesteuert wird Schein (2006).

Die **grundlegenden unausgesprochenen Annahmen** sind in der dritten Ebene enthalten. Diese für das Unternehmen allgemeinen und selbstverständlichen Überzeugungen und Werte sind die Essenz der Kultur des Unternehmens. Sie sind das Ergebnis eines gemeinsamen Lernprozesses. Bei Gründung des Unternehmens haben die Gründer und Inhaber diese Annahmen, Werte und Überzeugungen geprägt. Bei Erfolg des Unternehmens werden diese Werte und Überzeugungen dann von allen Mitarbeitern übernommen. Für das Verständnis einer Unternehmenskultur sind diese Annahmen sehr wichtig und müssen aufgespürt werden. Sie

wirken allumfänglich aber sind den Mitarbeitern nicht mehr bewusst, da sie bereits als selbstverständlich angesehen werden Schein (2006).

Das kulturelle Eisbergmodell von Sackmann (2004) in Abbildung 2-3 verdeutlicht den Zusammenhang zwischen den sichtbaren und leicht zugänglichen Manifestationen der Unternehmenskultur und den nicht sichtbaren Manifestationen wie den grundlegenden Überzeugungen, die sich auf Prioritäten, Prozesse, dem Zuschreiben von Ursachen und Verbesserungen beziehen.

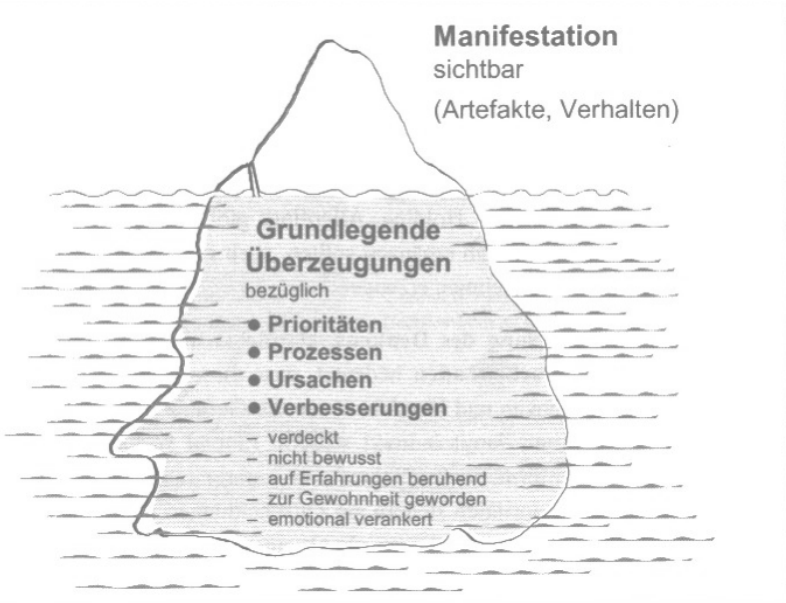

Abbildung 2-3: Kulturelles Eisbergmodell nach Sackmann (2004)

2.5 Funktionen der Unternehmenskultur

Unternehmenskultur stellt eine unsichtbare Einflussgröße auf die Mitarbeiter des Unternehmens dar. Die im Unternehmen vorhandenen grundlegenden Überzeugungen, das kollektive Denken, das Handeln und das Empfinden werden von ihr gelenkt, geordnet, organisiert und beeinflusst (Sackmann, 2004). Nach Nicolai (2012) werden dadurch bestimmte Handlungsweisen zugelassen und andere von vorne herein ausgeschlossen und es entsteht ein Zusammengehörigkeitsgefühl unter den Mitarbeitern. Nicolai (2012) spricht in diesem Zusammenhang von der „organischen Solidarität“, die auch als „Clan-Mechanismus“ bezeichnet wird. Als unsichtbare Einflussgröße erfüllt die Unternehmenskultur nach Sackmann (2004) vier zentrale und für das Bestehen sowie das Funktionieren des Unternehmens notwendige Funktionen:

- Reduktion der Komplexität
- Koordination des Handelns
- Identifikation mit dem Unternehmen
- Kontinuität

Unternehmenskultur unterstützt ein Unternehmen dabei, sich an seine Umwelt anzupassen und die Mitarbeiter zu integrieren. Homma et al. (2014) nennen sechs Funktionen der Unternehmenskultur und grenzen diese in „externe Anpassung“ und „interne Integration“ ab, wie in Abbildung Abbildung 2-4 dargestellt wird.

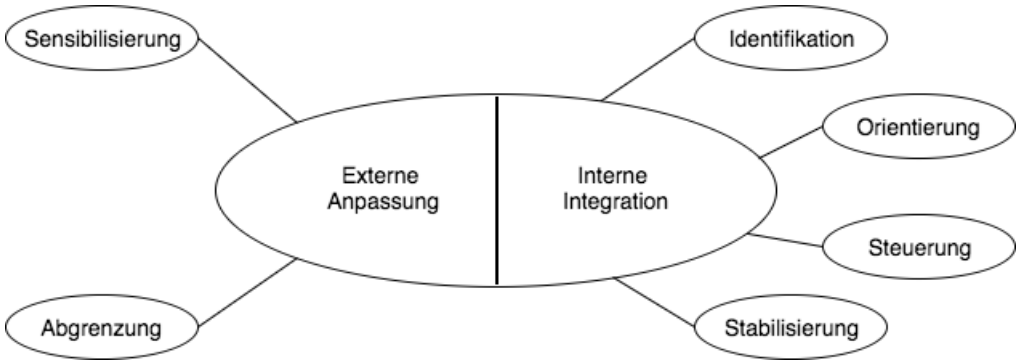

Abbildung 2-4: Abgrenzung der Funktionen der Unternehmenskultur [Eigene Darstellung in Anlehnung an Homma, Bauschke, & Hofmann (2014)]

Der Unternehmenskultur werden diverse Funktionen zugeordnet[1]:

2.5.1 Sensibilisierungsfunktion

Auf Unternehmen und deren Mitarbeiter haben viele Faktoren Einfluss. Die Unternehmenskultur dient als Sensibilisierung und Reduktion von Komplexität. Nach Homma et al. (2014) gewährleistet die Sensibilisierungsfunktion, dass das Unternehmen „wichtige Entwicklungen, Trends und Veränderungen im Umfeld wahrnimmt und in interne Prozesse integriert“ (S. 10). Nach Sackmann (2004) ermöglicht die Unternehmenskultur über diese Komplexitätsreduktionsfunktion ein schnelles und sicheres Handeln durch das Kategorisieren und Filtern der Wahrnehmung anhand der grundlegenden Überzeugungen in „relevant“ und „nicht-relevant“. Die gemeinsamen Denkmuster im Unternehmen bestimmen somit, wie mit den Informationen in Situationen umgegangen und dann gehandelt werden soll.

2.5.2 Abgrenzungsfunktion

Nach Homma et al. (2014) wirken die gemeinsamen Werte nach innen einheitsstiftend und ermöglichen gleichzeitig eine Abgrenzung gegenüber anderen Unternehmen und Organisationen.

[1] Es handelt sich hierbei um keine abschließende Aufzählung.

2.5.3 Identifikationsfunktion

Mitarbeiter und Führungskräfte teilen gemeinsame Annahmen und Werte, was nach Homma et al. (2014) und Sackmann (2000) zu höherer Motivation und Verbundenheit mit der Arbeitsgruppe, Abteilung und dem gesamten Unternehmen führt. Weil Unternehmenskultur den Sinn und Zweck sowie grundlegende Annahmen zur Existenzberechtigung eines Unternehmens beinhaltet kann der Mitarbeiter den Sinn seiner eigenen Aufgabe und damit verbundenen Arbeit begreifen und in den Kontext des gesamten Unternehmens einordnen. Das „Wir-Gefühl“ kann sich nach Baetge, Schrewe, Schulz und Solmecke (2007) nur dann entwickeln, wenn die Mitarbeiter „die Vision und Ziele des Managements kennen und sich zu eigen machen“ (S. 188). Die Stärke der Identifikation mit dem Unternehmen hängt nach Sackmann (2014) von der Ausgestaltung des Sinnsystems ab.

2.5.4 Orientierungsfunktion

Neben der Sinnstiftung durch die Identifikationsfunktion dient die Unternehmenskultur nach Homma et al. (2014) auch zur Orientierung für die Mitarbeiter bei Unklarheiten bezüglich Verhalten oder Entscheidungsfindung. Die Orientierungsfunktion wird nach Sackmann (2004) auch dadurch ermöglicht, dass die grundlegenden Überzeugungen gemeinsam, d.h. von allen Mitarbeitern und Führungskräften, getragen werden.

2.5.5 Koordinations- und Steuerungsfunktion

Die Unternehmenskultur unterstützt nach Homma et al. (2014) „aus Sicht des Managements [...] die Erfüllung von Aufgaben und reduziert das Potenzial für abweichendes Verhalten“ (S. 10). Diese Funktion kann auch als Koordinationsfunktion bezeichnet werden, wie man sie bei Sackmann (2004) und Nicolai (2012) wiederfindet. Nach Nicolai (2012) sind weniger Koordinationsmaßnahmen notwendig, je besser die gemeinsamen Denk- und Verhaltensmuster von allen Mitarbeitern und Führungskräften übereinstimmen. Davon hängt es auch ab, welche weiteren Instrumente zur Koordination überhaupt eingesetzt werden müssen. Die eingesetzten Koordinationsinstrumente wie beispielsweise Fremdkoordination durch persönliche Weisung oder Koordination durch Selbstbestimmung prägen ihrerseits die Kultur in einem Unternehmen.

Die grundlegenden Überzeugungen stellen laut Sackmann (2004) den Mitarbeitern und Führungskräften ein gemeinsames Sinnsystem bereit, was ein aufeinander abgestimmtes Verhalten ermöglicht. Das gemeinsame Sinnsystem ist besonders bei der Zusammenarbeit von Menschen aus verschiedenen Abteilungen, als auch verschiedenen Unternehmen, wichtig, da es sonst zu Verständnisproblemen zwischen den Parteien, durch beispielsweise die unterschiedliche Interpretation gemeinsam genutzter Begriffe, kommen kann.

Die Koordination durch die Unternehmenskultur bringt Vor- und Nachteile mit sich, die in Tabelle 1 dargestellt werden.

Koordination durch die Unternehmenskultur	
Vorteile	**Nachteile**
• Schnelle Entscheidungsfindung • Reibungslose Kommunikation • Einheitliche, voraussehbare Vorgehensweise • Motivationswirkung • Zügige Umsetzung • Hohe Loyalität • Leichtere Führung • Größere Sicherheit bei der Aufgabenerfüllung	• Mangelnde Anpassungsfähigkeit des Unternehmens • Mangelnde Anpassungsfähigkeit der Mitarbeiter • Blockierung neuer Vorgehensweisen • Geringere Beeinflussungsmöglichkeiten • Hohe zeitliche und finanzielle Belastung

Tabelle 1: Vor- und Nachteile der Koordination durch die Unternehmenskultur nach Nicolai (2012)

2.5.6 Stabilisierungsfunktion

Durch die Erfolgs- und Misserfolgsrezepte in der Lerngeschichte des Unternehmens erhält das Unternehmen nach Sackmann (2004) eine „gewisse Verhaltenssicherheit und Kontinuität im positiven Sinn, da nicht jeder Arbeitsvorgang neu überdacht oder erst entwickelt werden muss“ (S. 30). Das Unternehmen damit nach Homma et al. (2014) um ein gewisses Maß berechenbarer.

2.5.7 Erfolgssteigerungsfunktion

Die Unternehmenskultur kann nach Sackmann (2004) „die Produktivität und Wirtschaftlichkeit eines Unternehmens und damit seinen Erfolg zentral beeinflussen“ (S. 28).

2.5.8 Innovationsfunktion

Die Unternehmenskultur hat eine Innovationsfunktion, wenn eine gemeinsame Haltung gegenüber Kreativität vorliegt und die Unternehmensprozesse Innovation unterstützen. Personalentwicklung bestehend aus Aus- und Weiterbildung der Mitarbeiter und Führungskräfte ist dabei ein weiteres notwendiges Kriterium für die Innovationskraft in einem Unternehmen (Pittrof, 2011; Baetge et al., 2007).

2.6 Dimensionen und Werte von Unternehmenskulturen

Unternehmenskultur ist ein Wertgerüst, welches das Handeln und Denken der Mitarbeiter im Unternehmen beeinflusst. Kulturen können in einzelnen Dimensionen und Werten des Wertgerüsts unterschiedlich stark ausgeprägt sein.

Werte die von Unternehmen verfolgt werden[2]:

> Leistung, Innovation, Qualität, Schnelligkeit, Effektivität, Effizienz, Professionalität, Unternehmertum, Integrität, Kundenzufriedenheit, Kundenorientierung, Service, Nähe, Verlässlichkeit, Vertrauen, Ehrlichkeit, Vielfalt, Proaktivität, Zusammenarbeit, Offenheit, Kompetenz, Mut, Respekt, Einsatz, Zugehörigkeit, kontinuierliche Verbesserung, Loyalität, Integrität, Risikofreude, persönliche Verantwortung, Soziales Engagement, Einsatz, Nachhaltigkeit, Energieeffizienz, Partnerschaftlichkeit, Pragmatismus, Fairness, Anerkennung, Wirtschaftlichkeit, Feedback, Lernbereitschaft

Auf Basis der von Unterreitmeier (2004) und Sackmann (2006) zusammengetragenen Dimensionen von Unternehmenskulturen sollen im weiteren Verlauf die für diese Arbeit als relevant zur Beantwortung der Forschungsfragen identifizierten Dimensionen und deren Ausprägungen näher beschrieben werden:

- Zielorientierung
- Kommunikation und Information
- Führung und Mitarbeiterorientierung
- Kundenorientierung
- Innovationsfähigkeit

Die Dimensionen und Werte werden ausschließlich auf Unternehmensebene betrachtet und die aus nationalen Kulturen stammenden Werte, die ebenfalls wirken können, ausgeschlossen.

2.6.1 Zielorientierung

Ziele sind „angestrebte zukünftige Zustände“ (Watzka, 2017, S. 9). Sie bilden die zentralen Größen für Planung und Steuerung in Unternehmen. Ohne Ziele gäbe es keine Vergleichsmöglichkeiten. Lange hatten privatwirtschaftliche Unternehmen nur ein oberstes Ziel: Gewinnmaximierung. Heute kann man Unternehmen eher als Koalition mehrerer Stakeholder sehen, deren aller Interessen am Unternehmen in einem ausgewogenen Verhältnis berücksichtigt werden müssen, damit das Unternehmen langfristigen Erfolg erzielt (Watzka, 2017).

Nach Watzka (2017) lassen sich diese Ziele in drei Kategorien einteilen:

- **Leistungswirtschaftliche Ziele:** z. B. herzustellende Produkt-/Dienstleistungsarten, Mengen, Qualitätsniveaus, Prozessgeschwindigkeiten, Zielmärkte, Marktanteile
- **Finanzwirtschaftliche Ziele:** z. B. Gewinn, Umsatz, Kosten, Liquidität, Cashflow, Rentabilität, Sicherheit von Kapitalanlagen, Eigenkapitalausstattung, Unabhängigkeit

[2] Aufgelistet sind Werte, die sich in Visionen und Leitbildern von Unternehmen wiederfinden und Werte nach Wieland (1999)

- **Soziale Ziele:** z. B. Arbeitszufriedenheit, Arbeitsplatzsicherheit, Qualifizierung von Mitarbeitern, Umweltschutz

Als Zielorientierung wird im Rahmen dieser Arbeit die ganzheitliche Ausrichtung eines Unternehmens zur Erreichung von Zielen verstanden. Dies wird durch Management erreicht. In der Betriebswirtschaft bezeichnet Management die Koordination des Leistungsprozesses und der damit verbundenen Ressourcen, so dass die Ziele des Unternehmens erreicht werden (Hungenberg, 2014).

Management lässt sich nach Hungenberg (2014) in drei Aufgabenfelder unterscheiden:

- **Normatives Management**
    - Definition des Selbstverständnisses des Unternehmens
    - Festlegen der Vision, Mission und den grundlegenden Zielen des Unternehmens
    - Festlegen der Soll-Unternehmenskultur (Werte und Normen, Artefakte)
    - Formulierung des Leitbildes des Unternehmens
- **Strategisches Management**
    - Formulierung der Strategien
    - Implementierung und Kontrolle der Strategien
- **Operatives Management**
    - Festlegen der Ziele und Maßnahmen von Funktionsbereichen des Unternehmens (z.B. Beschaffung, Produktion, Absatz, Forschung und Entwicklung)

Das normative Management ist der Ausgangspunkt aller Managementaktivitäten im Unternehmen: Es gibt den Handlungsrahmen für das strategische Management vor. Dem strategischen Management ist das operative Management untergeordnet. Die Ziele der drei Management-Ebenen kaskadieren: Das normative Management legt die Oberziele des Unternehmens fest, welche durch die Strategie in Form von langfristigen Zielen weiter konkretisiert werden. Auf operativer Ebene werden die kurz- und mittelfristigen Ziele und Maßnahmen festgelegt, die die Strategie umsetzen (Hungenberg, 2014).

Nach Einführung in die Management-Ebenen und die damit verbundene Zielkaskadierung sollen nachfolgend Vision, Mission, Leitbild und Strategie genauer erläutert werden. Anschließend werden die Begriffe noch einmal zusammengeführt und in Beziehung gesetzt.

2.6.1.1. Vision

Die Vision eines Unternehmens ist „ein konkretes Zukunftsbild [eines Unternehmens], nahe genug, dass wir die Realisierbarkeit noch sehen können, aber fern genug, um die Begeisterung der Organisation für neue Wirklichkeit zu erwekken“ (Boston Consulting Group, 1988, zitiert nach Bleicher, 1994, S. 103). Sie ist ein geistiges Bild einer möglichen und gewollten Zukunft des Unternehmens, die sich als gemeinsam geschaffenes positives Vorstellungsbild über einen

zukünftigen Zustand selbst erfüllt (Scholz, 1992). Das Unternehmen beantwortet damit seine Grundsatzfrage, wo es in Zukunft stehen will (Bachert, 2007).

„Wenn du ein Schiff bauen willst,
dann trommele nicht Männer zusammen, um Holz zu beschaffen,
Werkzeuge vorzubereiten und Aufgaben zu vergeben,
sondern lehre sie die Sehnsucht nach dem endlosen Meer“
Antoine de Saint-Exupéry

Unternehmensvisionen werden in der Literatur einige Funktionen zugesprochen:

- **Legitimationsfunktion**: Die Vision soll dem Umfeld, wie beispielsweise Kunden und Lieferanten sowie den Mitarbeitern die Daseinsberechtigung und den Nutzen des Unternehmens vermitteln (Bleicher, 2004).
- **Abgrenzungsfunktion**: Durch die Vision grenzt sich das Unternehmen klar und eindeutig von anderen Wettbewerben ab und schärft damit das eigene Profil (Venzin, Rasner, & Mahnke, 2010). Dadurch gibt sie dem Unternehmen eine Identität (Sztuka, 2017). Bei richtiger Anwendung schafft sie damit dem Unternehmen Wettbewerbsvorteile als Faktor der Leistungs- und Wettbewerbsfähigkeit (Müller, o.J.).
- **Orientierungs- und Führungsfunktion**: Durch den beschriebenen Soll- und Ideal-Zustand des Unternehmens einer Vision dient sie als langfristiges Ziel und gibt den Mitarbeitern Orientierung für ihre Handlungen und erklärt den Sinn- und Nutzen der Handlungen (Sztuka, 2017). Damit kann die Vision als Führungs- und Steuerungsinstrument für Mitarbeiter von den Führungskräften eingesetzt werden (Menzenbach, 2012). Zusätzlich kann die Vision in Krisen Halt bieten und eine einheitliche Richtung für die Mitarbeiter vorgeben, an der sie sich orientieren können (Pohl, 2012).
- **Identifikations– und Motivationsfunktion**: Mitarbeiter können durch die Vision ihren Arbeit im Gesamtkontext des Unternehmens sehen und einordnen. Durch die Identifikation der Mitarbeiter mit der Vision führt dies zu höherer Motivation und bewirkt die Freisetzung von Energie, wovon das Unternehmen schließlich profitiert (Menzenbach, 2012; Venzin, Rasner, & Mahnke, 2010). Der Zielzustand der Vision ist die Analogie zum endlosen Meer im Zitat von Antoine de Saint-Exupéry. Eine positiv formulierte Vision steckt die Mitarbeiter an, begeistert und motiviert sie und sorgt für das „Wir-Gefühl“ und der Identifikation mit dem Unternehmen (Hans, 2003).

Die Vision eines Unternehmens wird durch die ersten Gründer des Unternehmens festgelegt und steht am Anfang aller Überlegungen des Managements (Bleicher, 1994; Schein, 2006). Im Laufe der Entwicklung des Unternehmens sollen aber auch die Mitarbeiter in die Visionsentwicklung mit einbezogen werden (Hans, 2003), da Menschen dazu neigen, die Entscheidungen anzunehmen und zu akzeptieren, die sie mit gefällt oder beeinflusst haben (Wheatley & Frieze, 2006).

2.6.1.2. Mission

Der Begriff „Mission“ kommt aus dem Lateinischen und bedeutet so viel wie Sendung und wird bildsprachlich für Aufgabe und Auftrag verwendet. Die Mission eines Unternehmens beschreibt den wesentlichen Zweck oder Auftrag des Unternehmens und damit dessen Existenzberechtigung zum gegenwärtigen Zeitpunkt (Müller-Stewens & Lechner, 2016, S. 224). Durch die Mission erfahren die Stakeholder, Kunden und Eigentümer des Unternehmens wofür das Unternehmen steht und was es für diese sein will (Fleig, 2016). Über die Definition der Kernkompetenzen des Unternehmens beschreibt die Mission, mit welchen Produkten oder Dienstleistungen das Unternehmen die Bedürfnisse der Kunden(-gruppen) befriedigt (Welge, 2008, S. 195).

Die Mission bildet die Grundlage zur Formulierung der Unternehmensstrategie, da aus ihr die strategischen Ziele abgeleitet werden können (Sztuka, 2017).

2.6.1.3. Leitbild

Die Vision und Mission finden sich zusammen mit den Werten des Unternehmens als verschriftlichte Form als Leitbild wieder. In vielen Unternehmen wird das Leitbild auf Webseiten, Broschüren und als Aushang in den Firmengebäuden veröffentlicht. Die Werte eines Unternehmens werden auch als Unternehmensphilosophie bezeichnet (Wilson, 1992; Matje, 1996). Neben dem allgemeinen Leitbild des Unternehmens finden sich auch Führungsleitbilder wieder, die das erwartete Verhalten der Führungskräfte definieren (Koch, 2016).

Im Drei-Ebenen-Modell nach Schein (vgl. Kapitel 2.4.1) sind diese „bekundeten Rechtfertigungen“ der mittleren Ebene, den bekundeten Werten“ zuzuordnen. Wenn die bekundeten Werte mit den Grundprämissen/-werten der Mitarbeiter im Unternehmen übereinstimmen, dann können diese Werte nach Schein (1995) den Mitarbeitern „ein Gefühl ihrer Identifikation und zentralen Botschaft vermitteln“ (S. 32). Nach Sackmann (2006) ist das beschriebene Ausmaß an Übereinstimmung des normativen Anspruchs und dem gelebten Verhalten ein zentrales Gütemaß und Indikator für eine spezifische Unternehmenskultur.

2.6.1.4. Strategie

Nachdem der Begriff „Strategie“ ursprünglich aus dem militärischen Bereich stammt, fand er in den fünfziger Jahren den Einzug als Langfristplanung in der Betriebswirtschaft (Schrader, 1995). Chandler definiert Strategie als „the determination of the basic long-term goals and objectives of an enterprise, and the adoption of courses of action and the allocation of resources necessary for carrying out these goals“ (1962, S. 13). Somit lässt sich die Strategie als die Gesamtheit aller Maßnahmen zur Erreichung der langfristigen Ziele des Unternehmens, wie der Vision, beschreiben.

Strategien beziehen sich immer auf ein Gestaltungsobjekt, auf das sie Bezug nehmen. Je nach Komplexität lassen sich diese nach Müller-Stewens und Lechner (2016) unterschiedlichen Gestaltungsebenen zuordnen. Bei der Geschäftsstrategie (Business Strategy) wird Bezug auf

eine einzelne unternehmerische Einheit genommen. Die Unternehmensstrategie (Corporate Strategy) hingegen bezieht sich auf das gesamte Unternehmen. Wenn sich mehrere Unternehmen kooperativ zusammenschließen verfolgen diese Unternehmen eine Netzwerkstrategie. In einzelnen Unternehmensbereichen findet man Funktionsstrategien, die sich auf direkte leistungswirtschaftliche Aktivitäten unternehmerischer Einheiten beziehen wie beispielsweise die Produktions- oder Marketingstrategie. Funktionsstrategien können sich daneben auch auf unterstützende Aktivitäten wie Personal, Finanzierung oder IT beziehen. Zwischen diesen Gestaltungsebenen bestehen Interdependenzen, so dass Änderungen auf einer Ebene Rückkopplungen auf andere Ebenen haben (Müller-Stewens & Lechner, 2016, S. 33ff.).

Nach Porter (1999) wirken in Abhängigkeit der Branche unterschiedliche Kräfte (vgl. Abbildung 2-5) auf Unternehmen, die sich letztendlich im Gewinnpotential der Branche widerspiegeln und in jeder Branche anders wirken. Unternehmen müssen eine Wettbewerbsstrategie innerhalb der Branche finden, in der sie tätig sind, „die es ihnen ermöglicht eine Position zu finden, in der es sich [das Unternehmen] am besten gegen diese Wettbewerbskräfte schützen oder sie zu seinen Gunsten beeinflussen kann.“ (S. 34).

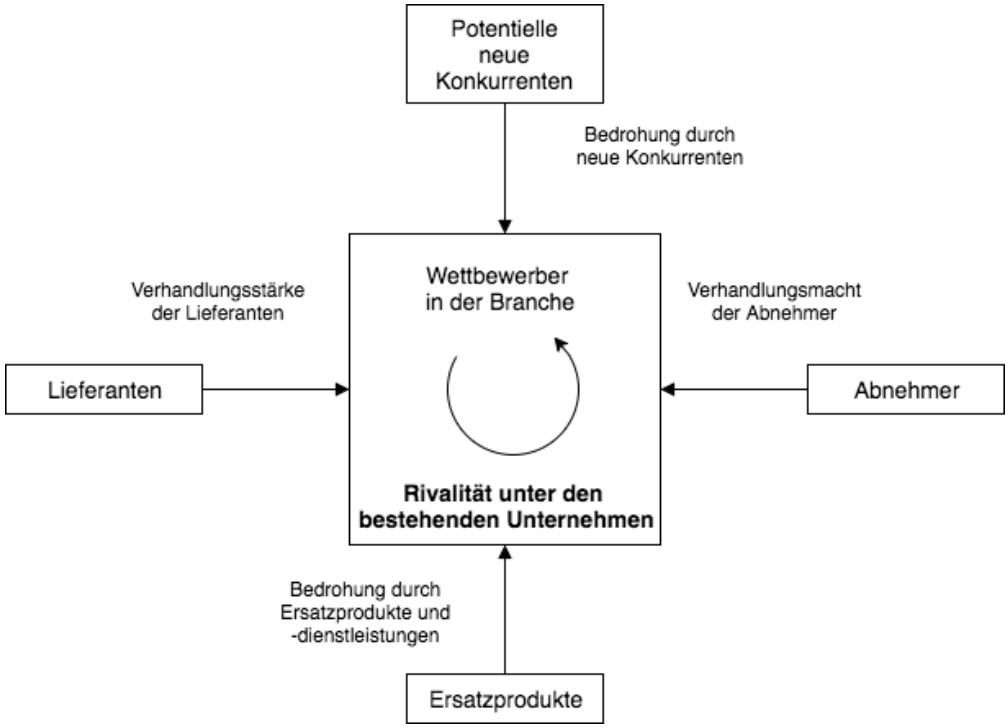

Abbildung 2-5: Wettbewerbskräfte innerhalb einer Branche [Eigene Darstellung in Anlehnung an (Porter, 1999, S. 34)]

Porter (1999) nennt drei wesentliche Gruppen von Wettbewerbsstrategien, die Unternehmen verfolgen können, um eine gefestigte Position in der Branche zu schaffen:

- **Umfassende Kostenführerschaft:** Durch eine ganzheitliche Fokussierung auf die Kosten innerhalb der Wertschöpfungskette des Unternehmens und der gesamten Strategie soll durch niedrige Kosten ein Vorteil gegenüber den Wettbewerbern erreicht werden. Zur Umsetzung der Kostenführerschaft sind ein hoher Marktanteil und Absatz sowie damit verbundene Einkaufsvorteile oder ein effizienter Herstellungsprozess notwendig. Trotz der Einsparungen sollen aber Qualität und Service weiterhin Berücksichtigung fin-

den. Mit Hilfe dieser Strategie lassen sich laut Porter alle fünf Wettbewerbskräfte bezwingen.

- **Differenzierung:** Bei der Differenzierungsstrategie setzt sich das Unternehmen durch ein einzigartiges Produkt oder eine Dienstleistung von den anderen Mitbewerbern der Branche ab. Als Differenzierungsmerkmal nennt Porter unter anderen Merkmalen Qualität, Technologie, Kundendienst/Service und Vertriebswege. Die Kosten dürfen bei dieser Strategie trotzdem nicht vernachlässigt werden, stehen aber nicht im Mittelpunkt.
- **Konzentration auf Schwerpunkte:** Im Gegensatz zur Kostenvorsprungs- und Differenzierungsstrategie geht es bei der Konzentrationsstrategie um die Bevorzugung einer Nische anstatt eine branchenweite Umsetzung. Durch die klare Konzentration auf eine Nische können nach Porter die Unternehmen, die diese Strategie einsetzen, wirkungsvoller und effizienter agieren als deren Mitbewerber, die einem breiteren Wettbewerb ausgesetzt sind.

Unternehmen denen es nicht gelingt, ihre Strategie in eine der genannten Richtungen zu entwickelt wird nach Porter (1999) sinnbildlich als *zwischen den Stühlen* bezeichnet. Diese Unternehmen müssen demnach eine fundamentale Entscheidung treffen und „entweder die notwendigen Schritte einleiten, um Kostenführerschaft [...] zu erreichen, [...]; oder es muß sich auf ein bestimmtes Zielobjekt richten (Konzentration) oder irgendeine Einmaligkeit schaffen (Differenzierung).“ (S. 80).

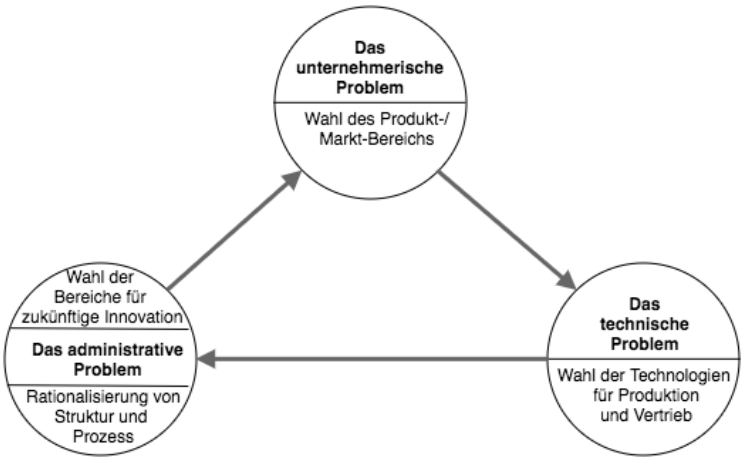

Abbildung 2-6: Anpassungskreislauf nach Miles und Snow [Eigene Darstellung in Anlehnung an (Schiffelholz, 2014)]

Neben Porter liefern Miles, Snow, Meyer und Coleman (1978) ein Modell, das strategische Grundausrichtungen beschreibt, die Unternehmen annehmen können. Unternehmen unterliegen nach Miles und Snow (2003) einem Anpassungskreislauf (vgl. Abbildung 2-6). In diesem adaptiven Prozess beschreiben Sie drei Probleme, die gelöst werden müssen: das unternehmerische Problem, das technologische Problem und das administrative Problem.

Das Modell von Miles et al. definiert drei stabile Strategietypen, welche die Probleme im Anpassugskreislauf unterschiedlich lösen (Miles R. E., Snow, Meyer, & Coleman, 1978; Miles & Snow, 2003; Schiffelholz, 2014):

- **Defender:** Der erste Strategietyp Defender agiert in stabilen Produkt- und Marktbereichen und schottet einen Teil des Marktes ab. Seinen Marktanteil verteidigt er durch eine agressive Preispolitik und hochqualitative Produkte. Defender wachsen durch Steigerung des Marktanteils. Sie streben technologische Effizienz an und nutzen wenige Kerntechnologien auf die sie ihren Fokus legen. Die Stabilität ermöglicht gleichbleibenden Output und somit bleibt auch die Auslastung entsprechend hoch. Zusätzlich wird durch vertikale Integration versucht, die Effizienz und Sicherheit der Wertschöpfungskette des Unternehms zu erhöhen. Typisch für Defender sind eine intensive Kostenplanung, zentrale Steuerung und Kommunikation über formale Hierarchieebenen. Die Führungsebene besteht aus Spezialisten für Produktion und Controlling. Der Defender beachtet die Zukunft kaum und kann deshalb nur unzureichend auf größere Veränderungen frühzeitig und ausreichend reagieren.
- **Prospector:** Im Gegensatz zu den Defendern agieren Prospectors in einem dynamischeren Umfeld, aber oft auch in der gleichen Industrie wie diese. Sie wachsen durch Hinzunahme weiterer Produkte und Märkte, nach denen sie ständig auf der Suche sind. Durch die ständige Veränderung erhalten sie Vorteile gegenüber ihren Wettbewerbern. Anstatt hoher Profitablität wollen sie als anerkannter Innovationsführer auf den Markt beschreiten. Prospectors besitzen die Fähigkeit, neue Markt- und Produktchancen zu identifizieren und für sich zu nutzen. Dafür ist Flexibilität in der Technologie die wichtigste Anforderung wohingehen langfristige Festlegungen auf eine bestimmte Art von Technologie werden vermieden. Neben den flexibeleren Technologien ist auch die Organisation ebenso flexibel, d.h. weniger formell und zentralistisch als auch weniger hierarchisch aufgebaut. Kommunikation geschieht im Gegensatz zum Defender auch über Hierarchieebenen hinweg statt. Geführt werden Prospectors von Experten aus dem Marketing, aus der Forschung und der Entwicklung. Für Prospectors hat die Zukunft einen hohen Stellenwert, was sich auch in der Entwicklung ihrer Systeme widerspiegelt.
- **Analyzer:** Zwischen den anderen beiden extremen Strategietypen Prospector und Defender liegt der Analyzer, der sich deren Vorteile zu Eigen macht. Analyzer sind keine Innovatoren: Sie kopieren bei Mitbewerben bereits funktionierde bekannte Ansätze, um für sich neue Produkte und Märkte zu erschließen.

Neben den drei genannten stabilen Strategietypen existiert der *Reactor*, der sich durch Instabilität auszeichnet. Die Reactor-Strategie ergibt sich, wenn die anderen Strategien nur unzureichend verfolgt werden und gleicht Porters Strategieausprägung „Zwischen den Stühlen“ (Schiffelholz, 2014).

2.6.1.5. Zusammenfassung

Zielorientierte Unternehmen haben eine Vision und Mission, aus denen die Strategie und die Ziele des Unternehmens abgeleitet sind (Sackmann, 2004). Die Abbildung 2-7 stellt den Zusammenhang zwischen den einzelnen Begriffen noch einmal grafisch dar: Die Vision liefert das

langfristige Ziel, auf das das Unternehmen hinarbeitet. Die im Leitbild eingebettete Mission und propagierten Werte des Unternehmens begrenzen den Handlungsraum der Strategie.

Wie zielorientiert ein Unternehmen ist hängt davon ab, ob Vision, Mission und Strategie existieren. Neben der Existenz von Vision, Mission und Strategie hängt es weiterhin davon ab, in wie weit die Kenntnis über diese Elemente bei den Führungskräften als auch den Mitarbeitern in allen organisationalen Einheiten besteht und diese Elemente verankert sind (Sackmann, 2004).

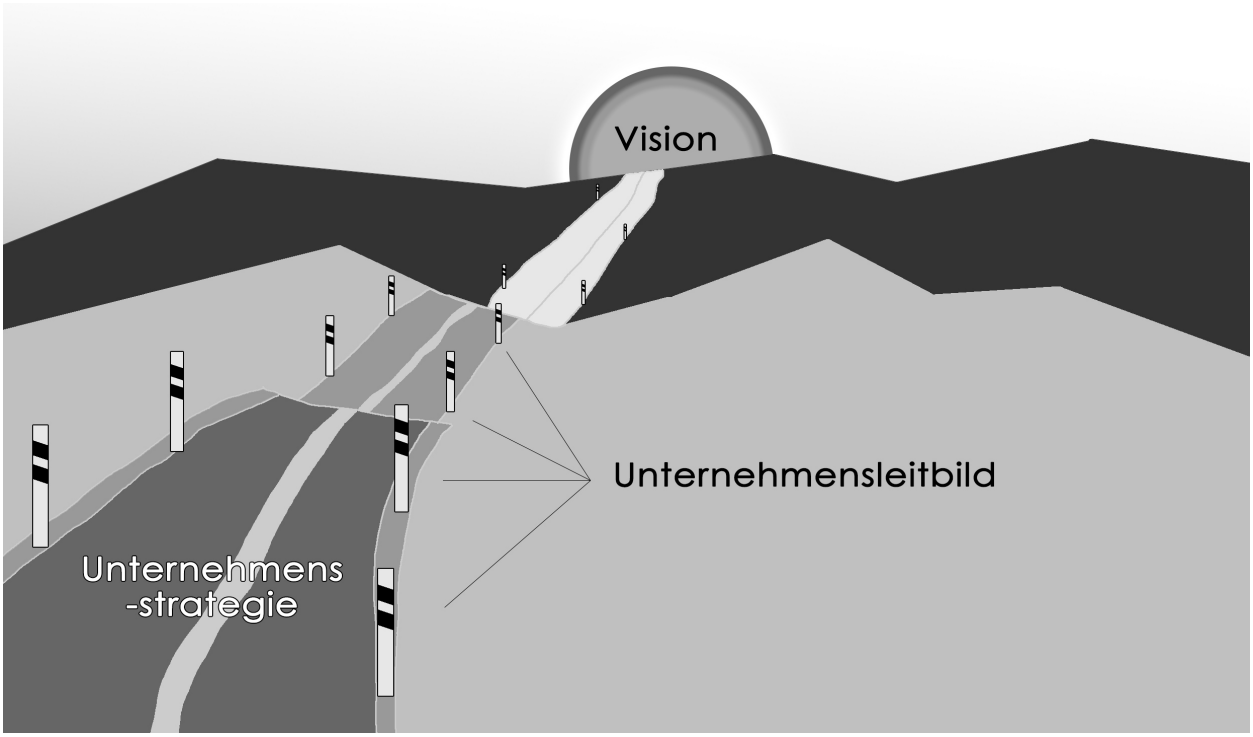

Abbildung 2-7: Zusammenhang von Strategie, Leitbild, Mission und Vision [Eigene Darstellung in Anlehnung an (Vahs & Schäfer-Kunz, 2005, S. 192)]

2.6.2 Kommunikation und Information

Unternehmenskultur zeigt sich unter anderem im Kommunikationsverhalten in Unternehmen (Mast, 2016, S. 186f.). Bevor aber die Kultur eines Unternehmens und Kommunikation in Zusammenhang gesetzt werden, soll nachfolgend in das Thema Kommunikation eingeführt werden.

Für den Begriff Kommunikation existieren, wie auch für die Unternehmenskultur eine Vielzahl von Definitionen, die sich darin unterscheiden, wie weit der Begriff gefasst wird (Mast, 2004). Das Wort „Kommunikation“ leitet sich vom lateinischen Begriff „communicatio“ ab und kann mit „Mitteilung“, „Gemeinschaft“ und „Teilnahme“ übersetzt werden. Kommunikation ist neben der Mitteilung auch ein „[...] interaktiver Prozess [...] mit mindestens zwei beteiligten Personen“ (Haug, 2012, S. 96). Dieser wechselseitige Prozess kann verbal oder nonverbal ablaufen und ist Teil des sozialen Handels, wie auch das Axiom von Watzlawick (2007) ausdrückt: „man kann nicht nicht kommunizieren“ (S. 53). Kommunikation ist nach dieser These also immer präsent.

Kommuniziert wird jedoch meist bewusst: Nach Mast (2004) wird bei Kommunikation immer eine Intention verfolgt, da der Kommunikator als auch der Rezipient einer Botschaft eine bestimmte Absicht und einen Zweck durch ihre Kommunikation verfolgen. Information wird im Gegensatz zur Kommunikation als einseitiger Prozess verstanden (Mast, 2004).

Abbildung 2-8 zeigt ein Kommunikationsmodell, angelehnt an das viel zitierte Sender-Empfänger-Kommunikationsmodell von Shannon und Weaver (1949), welches eine sehr technische Sicht auf Kommunikation darstellt.

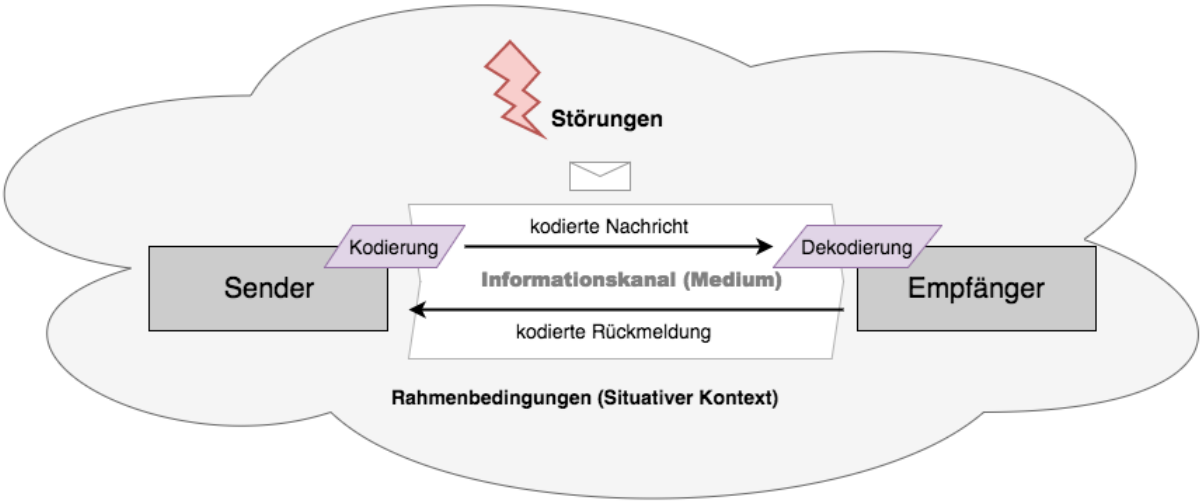

Abbildung 2-8: Allgemeines Sender-Empfänger-Modell [Eigene Darstellung]

Der Kommunikationsprozess startet beim Sender als Quelle der Information. Der Empfänger wählt eine Nachricht aus und kodiert sie. Der Code für die Kodierung der Nachricht kann beispielsweise eine bestimmte Sprache sein. Über den Informationskanal, auch Medium genannt, erfolgt die Übermittlung der Nachricht an den Empfänger. Wenn die Nachricht ohne Störungen beim Empfänger eingegangen ist, kann sie von diesem dekodiert werden. Voraussetzung für die Dekodierung ist zumindest teilweise ein identisches Zeichen- und Bedeutungswissen (Sprache) (Röhner & Schütz, 2012). Störungen können bei verbaler Kommunikation beispielsweise Umgebungsgeräusche, das Rauschen eines Telefons oder in Zeiten digitaler Telefonie nur die teilweise Übermittlung der Nachricht sein. Der Empfänger kann über den gleichen Kanal eine Rückmeldung an den Empfänger geben. Der geschilderte Prozess der Kommunikation findet unter Rahmenbedingungen in einem situativen Kontext statt. Der Kontext umfasst vorherrschende Kommunikationsregeln, die den Prozess beeinflussen als auch die sichtbaren (z.B. Gesten) und nicht sichtbaren Aktivitäten (Eindrucksbildung vom Gegenüber) der Kommunikationsteilnehmer.

2.6.2.1. Probleme in der Kommunikation

Neben dem Dekodieren der Nachricht durch den Empfänger ist auch das Verständnis ein wichtiger Faktor bei der Kommunikation. Nach Schulz von Thun (2006) hat eine Nachricht vier simultane Botschaften, die der Empfänger einer Nachricht interpretieren kann:

- **Sachinhalt:** Informationen, die die sendende Person der empfangenden Person mitteilen möchte,
- **Selbstoffenbarung:** Informationen über die sendende Person und dessen Aussage zu ihrer eigenen Kompetenz oder Selbstenthüllung von Gefühlen.
- **Beziehungsaussagen:** Informationen über die Beziehung zwischen Sender und Empfänger.
- **Apell:** Aufforderung etwas zutun, zu unterlassen, zu denken oder zu fühlen.

Auf allen der vier genannten Ebenen kann es beim Interpretieren einer Nachricht durch den Empfänger zu Problemen kommen, so dass es zu störenden Missverständnissen kommt. Die Fähigkeit einer Person diese vier Botschaften der Nachricht vor dem Senden an die jeweilige Situation und den Gesprächspartner anzupassen wird als *Kommunikationskompetenz* bezeichnet (Röhner & Schütz, 2012).

2.6.2.2. Werte der Kommunikation

Das Verhalten der Kommunizierenden wird durch deren Werte beeinflusst. Die nachfolgende Abbildung 2-9 zeigt Werte, die bei der Kommunikation und Kooperation berücksichtigt werden können:

Abbildung 2-9: Allgemeines Sender-Empfänger-Modell [Eigene Darstellung in Anlehnung an (Schwegler, 2009)]

Je nachdem wie transparent und zugänglich Informationen in einem Unternehmen sind kann die Kommunikation als offen bzw. geschlossen bezeichnet werden.

2.6.3 Führung und Mitarbeiterorientierung

Führung ist die „Ausrichtung des Handelns von Individuen und Gruppen auf die Verwirklichung vorgegebener Ziele“ und stellt damit einen „komplexen sozialen Prozess“ dar (Alisch, Arentzen, & Winter, 2004, S. 1121). Wie bereits in Kapitel 2.4 geschildert ist das Handeln von Individuen von deren Werten abhängig. Auf Basis der genannten Definition von Führung lässt sich damit

feststellen, dass Führung das Festlegen von Werten ist, die dann zu Grundannahmen werden können und dadurch das Verhalten von Mitarbeitern beeinflussen können.

Damit die Werte bei den Mitarbeitern auch zu Grundannahmen werden, müssen die Werte mit den Werten übergeordneter Kulturen kompatibel sein, in die der Mitarbeiter eingebettet ist (z.B. die Branchenkultur oder Landeskultur) (Schein, 2017, S. 181ff.).

Nach Schein ist Charisma für Führungskräfte die einfachste Möglichkeit ihre Werte preis zu geben. Jedoch ist diese Möglichkeit aus Sicht des Unternehmens nicht sehr verlässlich, da nicht jede Führungskraft diese überzeugende persönliche Ausstrahlung hat und die Wirkung vorher nicht absehbar ist (Schein, 2017, S. 182).

Neben der Wertevermittlung über die Persönlichkeit und Charisma zeigt Abbildung 2-10 sechs primäre sowie sechs sekundäre verstärkende und stabilisierende Werkzeuge, mit denen Führungskräfte ihre Überzeugungen, Werte und Annahmen an Mitarbeiter weitergeben können, so dass diese verankert werden. Insgesamt werden diese Mechanismen auch als „Betriebsklima“ bezeichnet (Schein, 2017, S. 183).

Wie Führungskräfte ihre Überzeugungen, Werte und Annahmen verankern

Primäre verankernde Mechanismen

- Was Führungskräfte regelmäßig beachten, beurteilen und kontrollieren
- Wie Führungskräfte auf problematische Ereignisse und Krisen im Unternehmen reagieren
- Wie Führungskräfte Mittel und knappe Ressourcen zuteilen
- Bewusste Vorbildfunktion, Training und Coaching
- Wie Führungskräfte Belohnungen und Status zuteilen
- Wie Führungskräfte einstellen, befördern und entlassen

Sekundäre verstärkende und stabilisierende Mechanismen

- Gestaltung und Struktur des Unternehmens
- Systeme und Prozesse
- Bräuche und Rituale des Unternehmens
- Raumgestaltung
- Geschichten über wichtige Ereignisse und Personen
- Offizielle Aussagen zur Unternehmensphilosophie und Glaubenssätzen

Abbildung 2-10: Verankerungswerkzeuge für Kultur [Eigene Darstellung in Anlehnung an (Schein, 2017, S. 183)]

Die zu vermittelnden Überzeugungen, Werte und Annahmen der Führungskraft werden von der vorherrschenden Unternehmenskultur beeinflusst, welche einst durch den Gründer des Unternehmens etabliert wurde und sich dann im Laufe der Unternehmensentwicklung gefestigt hat.

Führungsverhalten lässt sich durch mehrere Dimensionen unterscheiden[3]: Mitbareiter-, Aufgaben- und Ergebnisorientierung. Bei einem mitarbeiterorientierten Führungsstil steht der Mitarbeiter im Vordergrund. Dieser Stil zeichnet sich durch die folgenden Merkmale aus:

> Fairness, Zusammenarbeit, Kreativität, Feedback, Unterstützung, Hilfsbereitschaft, Vertrauen, Freundlichkeit, Bedürfnisse und Probleme der Mitarbeiter werden berücksichtigt, Mitarbeiter werden bei der Entscheidungsfindung mit eingebunden, Mitarbeiter können sich mit einbringen, Mitarbeiter erhalten Freiräume für selbständiges Arbeiten, freie Arbeitsgestaltung, hohe Anzahl an Weiterbildungsmaßnahmen, offene Kommunikation, Informationsaustausch unter den Mitarbeitern (Sackmann, 2006; Unterreitmeier, 2004; Herberhold, 2016).

Dadurch das der Mitarbeiter bei diesem Stil im Vordergrund steht kann die Ziel- oder Ergebnisorientierung unter Umständen leiden.

Bei der Aufgabenorientierung wird das Erledigen der Aufgabe in den Mittelpunkt gerückt. Die Führungskraft legt die Ziele fest und kontrolliert diese auch. Durch die Priorisierung der Ziele ist dieser Führungsstil weniger empathisch als der mitarbeiterorientierte Stil der Führung.

Das Gegenteil zur Aufgabenorientierung stellt der ergebnisorientierte Führungsstil dar. Die Führungskraft konzentriert sich nur auf das erreichte Ergebnis.

2.6.4 Kundenorientierung

Als Bestandteil der Unternehmenskultur bezeichnet Kundenorientierung die grundlegende Einstellung der Mitarbeiter in einem Unternehmen zu ihren Kunden und deren Bedürfnissen, welche variabel und situativ beurteilt werden müssen (Kühn, 1992). Ein Unternehmen mit einer kundenorientierten Unternehmenskultur zeigt sich durch regen Informationsaustausch zwischen den Mitarbeitern und den Kunden des Unternehmens (Zollner, 1995).

Kundenorientierung kann sich aber auch in Form einer Differenzierungsstrategie des Unternehmens durch die Ausrichtung auf die Kundenbedürfnisse wiederfinden (Herzwurm, 1998). Dies äußert sich nach Herzwurm (1998) in einer „Priorisierung von Qualitäts- gegenüber Kosten- und Produktivitätszielen oder dem Streben, stets nach dem neusten Stand der Technik zu entwickeln“ (S. 283). Peters und Waterman identifizierten Kundenorientierung in Form von Kundennähe als einen von acht Erfolgsfaktoren von Unternehmen:

„The excellent companies really are close to their customers. That’s it. Other companies talk about it, the excellent companies do it.“ (Peters & Waterman, 1982)

Die Arbeit von Peters und Waterman wurde in wissenschaftlichen Kreisen z.T. heftig aufgrund fehlender exakter Definitionen und Operationalisierungen kritisiert. Zusätzlich geriet ein großer

[3] Es existieren noch weitere Dimensionen, nach denen sich Führungsverhalten beschreiben lässt. Diese sollen jedoch im Rahmen dieser Arbeit nicht weiter betrachtet werden.

Teil der befragten Unternehmen ein paar Jahre später in Schwierigkeiten. Trotz der Kritik wird Peters und Waterman der Verdienst zugesprochen, die Kundenorientierung in das Bewusstsein gerückt zu haben, auch wenn der Betriebswirtschaftler Drucker bereits 1954 ähnliche Aussage formulierte:

„There is only one valid definition of business purpose: to create a satisfied customer. It is the customer who determines what the business is.“ (Drucker, 1954)

In der Aussage von Drucker lässt sich Kundenzufriedenheit als ein Ziel von Kundenorientierung erkennen. Die Zufriedenheit der Kunden soll durch eine hohe Servicequalität erreicht werden. Produkte haben heute oft einen sehr hohen Reifegrad und sind aus Sicht des Kunden austauschbar, weshalb Unternehmen überdurchschnittlichen Service und auch zusätzliche Services anbieten, mit dem Ziel die Kunden an sich zu binden und langfristig zu halten (Zollner, 1995).

Homburg (1998) untersuchte Kundennähe in einer umfangreichen Untersuchung, bei der als Ergebnis die in Abbildung 2-11 dargestellte Konzeptualisierung entstand. Kundennähe entsteht demnach durch das Leistungsangebot des Unternehmens als auch durch das Interaktionsverhalten mit dem Kunden.

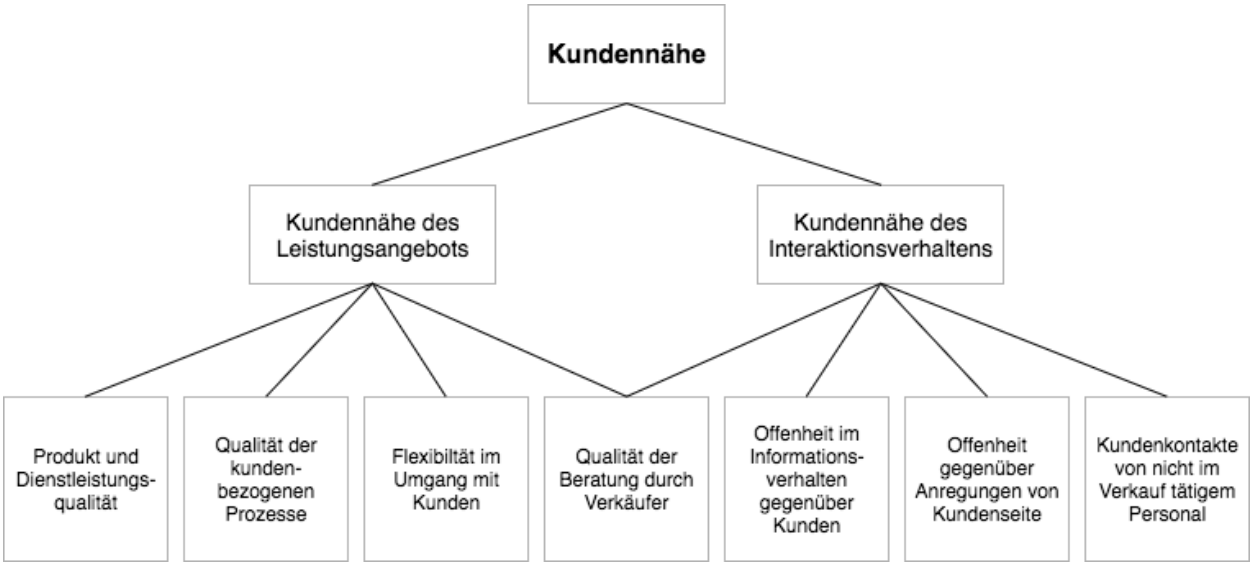

Abbildung 2-11: Konzeptualisierung von Kundennähe [Eigene Darstellung in Anlehnung an (Homburg, 1998, S. 120)]

Im Bereich der Innovationsforschung wird Kundenorientierung mit dem Erfolg von Innovationen in Verbindung gebracht. Dies soll im nachfolgenden Kapitel näher beschrieben werden.

2.6.5 Innovationsorientierung

Aufgrund steigender Rate von von sich ändernden Bedürfnissen, sinkender Loyalitätsquote und der Tendenz von Kunden, immer mehr Alternativen zu vergleichen, sind Innovationen für Un-

ternehmen notwendig, um deren lang anhaltendes Wachstum und auch deren Fortbestand zu sichern (Derenthal, 2009).

Der Begriff Innovation kommt aus dem lateinischen „innovatio“ und bedeutet so viel wie Neuerung und Neuheit. Schumpeter (1939) führte den Begriff erstmalig im Bereich der Wirtschaft ein. Er bezeichnet Innovation als „schöpferische Zerstörung“, welche demnach eine wesentliche Voraussetzung für die Entwicklung einer Volkswirtschaft sei. Innovation ist schöpferisch, da kontinuierlich neue Kombinationen von Produktfaktoren geschaffen werden und sogleich zerstörerisch, weil dafür bestehende Kombinationen von Produktfaktoren verändert oder aufgelöst werden.

Innovationen sind nach Hauschildt und Salomo (2007) „qualitativ neuartige Produkte oder Verfahren, die sich gegenüber dem Vergleichszustand ‚merklich’ – wie auch immer das zu bestimmen ist – unterscheiden“ (S. 4).

Zur weiteren Erläuterung des Begriffs Innovation soll nachfolgend der multidimensionale Ansatz von Hauschildt und Salomo (2007) verwendet werden. Abbildung 2-12 zeigt die Dimensionen von Innovation dieses Ansatzes. Innovationen lassen sich demnach über deren Gegenstand beschreiben und nach Produkt- und Prozessinnovation gruppieren.

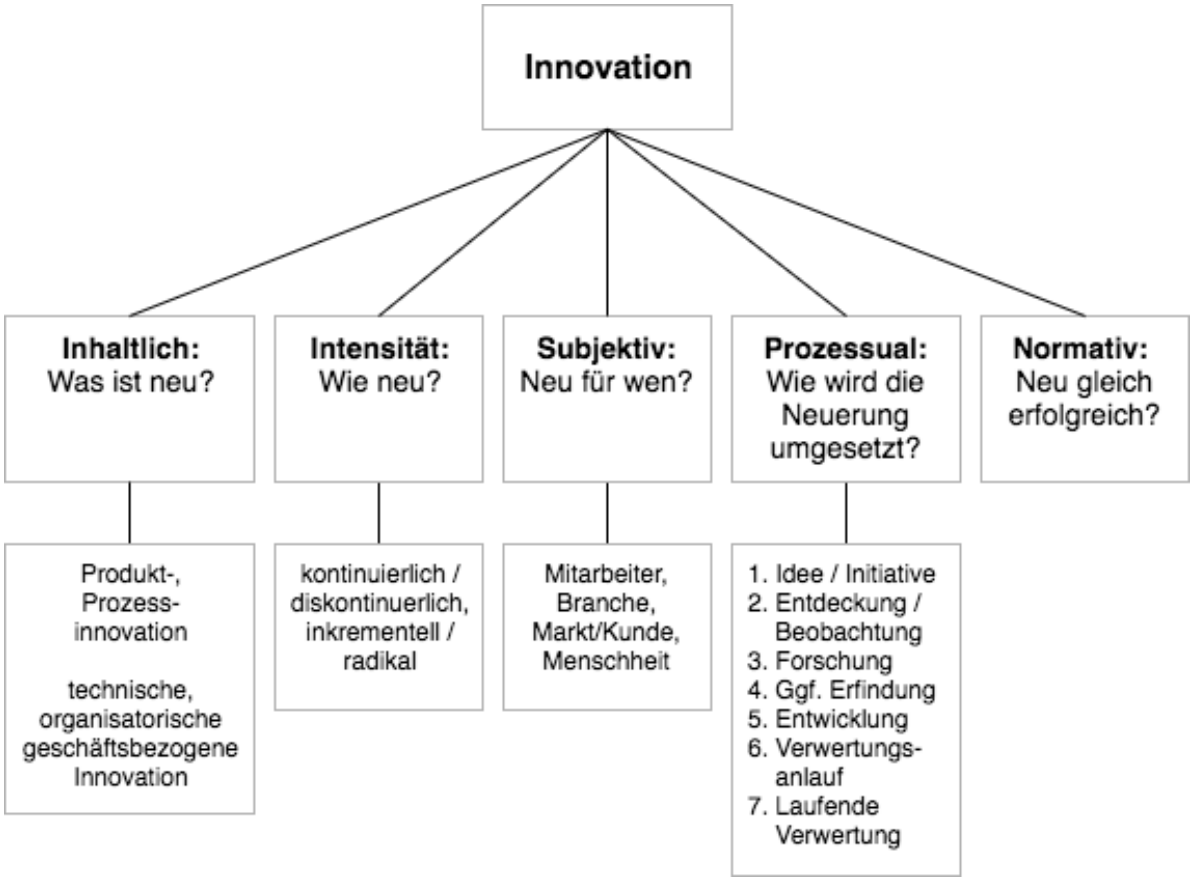

Abbildung 2-12: Dimensionen von Innovation [Eigene Darstellung]

Nachdem ein Verständnis von Innovation aufgebaut wurde, soll nachfolgend auf innovationsorientierte Unternehmenskulturen eingegangen werden. Nach Derenthal (2009) und der dort ge-

nannten Untersuchungen zeichnen sich innovationsorientierte Unternehmen durch die folgenden Merkmale aus:

- Neigung sich mit neuen Ideen und kreativen Prozessen zu beschäftigen
- Technologische Überlegenheit: ganzheitliche Konzentration auf bessere Produkte als Wettbewerber zu entwickeln
- Offenheit und positive Einstellung gegenüber Innovationen
- Entwicklung von neuen Ideen wird unterstützt, belohnt und aktiv verfolgt
- Unternehmerisches Denken
- Risikofreude
- Lern- und Anpassungsfähigkeit, Personalentwicklung
- Stolz auf die Mitarbeiter und Enthusiasmus über ihre Fähigkeiten
- Vertrauen, Unterstützung, Zusammenarbeit und Teamwork
- Partizipative Entscheidungsfindung
- Gemeinsam gelebte Vision
- Offensive strategische Ausrichtung

Derenthal (2009) und einige der dort betrachteten Untersuchungen konnten eine positive Wirkung von Innovationsorientierung auf den Innovationserfolg von Unternehmen feststellen. Der Innovationserfolg führt dann wiederum zu Unternehmenserfolg. Aber auch die Zielorientierung bzw. die strategische Grundausrichtung eines Unternehmens wirkt sich auf die Innovationsorientierung und den Erfolg des Unternehmens aus: In der Studie von Slater, Olson und Hult (2006), die den Fit von Strategie, Unternehmenskultur und Struktur untersucht, konnte die positive Wirkung von Innovationsorientierung beim Strategietyp Prospector auf den Unternehmenserfolg und die negative Wirkung bei den Strategietypen Analyzer und Defender nachgewiesen werden.

Die Innovationsorientierung wird positiv durch Kunden- und Lern- und Unternehmerorientierung beeinflusst. Gute Beziehungen der Mitarbeiter zu ihren Vorgesetzten sowie eine positive Einstellung der Führungskräfte zu Veränderungen wirken ebenfalls positiv auf die Innovationsorientierung eines Unternehmens (Derenthal, 2009).

Für das Verständnis der vorliegenden Arbeit wird, wie auch von Derenthal (2009), eine „kulturelle Sichtweise von Innovationsorientierung gewählt, die sich durch eine positive Einstellung gegenüber Innovationen als auch innovationsfreudige Handlungen bzw. ein innovationsorientiertes Verhalten der Organisationsmitglieder, also sowohl des Managements als auch der Mitarbeiter, beinhaltet, auszeichnet“ (S. 22).

3 SOFTWAREARCHITEKTUR

Softwaresysteme[4] werden entwickelt, um Geschäftsziele eines Unternehmens zu erreichen. Softwarearchitektur ist nach Bass, Clements und Kazmann (2012) die Brücke zwischen den unkonkreten Geschäftszielen und dem konkreten final entwickelten Softwaresystem. Diese abstrakte Definition soll im nachfolgenden Kapitel konkretisiert werden. Ähnlich wie bei der Unternehmenskultur existieren für den Begriff Softwarearchitektur viele Definitionen. So hat das Software Engineering Institute der Carnegie-Mellon-Universität knapp 200 Definitionen von Softwarearchitektur gesammelt (SEI, o.J.). Nachfolgend soll zuerst die Definition und Abgrenzung von Softwarearchitektur für diese Arbeit erfolgen. Anschließend werden die Elemente von Softwarearchitektur genauer erläutert. Danach wird die Rolle des Softwarearchitekten, die unterschiedlichen Möglichkeiten der Besetzung der Rolle und die mit der Rolle verbundenen Aufgaben beschrieben. Das Kapitel schließt mit der Vorstellung des Microservice-Architekturstils und unterstützenden Techniken für Microservice-Architekturen ab.

3.1 Definition und Inhalt von Softwarearchitektur

Softwarearchitektur ist nach Bass et al. (2012) „the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both“,

nach Fowler (2004) „[...] a notion of the core elements of the system, the pieces that are difficult to change. A foundation on which the rest must be built.“

und nach Kruchten (1999) die Summe verschiedener wichtiger Entscheidungen über die Organisation eines Softwaresystems, die Auswahl von Strukturelementen und deren Schnittstellen aus denen das System zusammengesetzt ist, das Verhalten und Zusammenspiel dieser Elemente, den hierarchischen Aufbau von Subsystemen und den zugrundeliegenden Architekturstilen.

Nach Beck ist Softwarearchitektur „what software architects do" (Beck, zitiert nach Toth (2014)).

Auf Basis der vorgestellten Definitionen beinhaltet die Softwarearchitektur im Rahmen dieser Arbeit die Menge an wichtigen Entscheidungen über die Organisation und Struktur eines Softwaresystems, die nur schwer zu ändern sind. Schwere Änderbarkeit definiert sich nach Toth (2014) darüber, „dass Änderungen teuer, aufwendig oder qualitätsgefährdend sind“ und den Projekterfolg beeinflussen in Bezug auf dessen Budget, Zeit oder Qualität. Eoin Woods betont die Wichtigkeit dieser Entscheidungen: „[...] if made incorrectly, may cause your project to be canceled“ (Toth, 2014, S. 3).

[4] Nachfolgend werden die Begriffe Software, Softwaresystem und IT-System synonym genutzt.

Softwarearchitektur besteht nach Bass et al. (2012) aus seinen Strukturen, der *Zerlegung in Komponenten*, deren Schnittstellen und Beziehungen untereinander. Sie definiert die Komponenten, die auch als Bausteine bezeichnet werden, und deren von außen beobachtbaren Merkmale, Beziehung und Interaktionen zwischen den Komponenten. Damit beschreibt die Architektur nach Starke (2011) mit den Architekturbausteinen statische Aspekte und erfüllt die Aufgabe als Bauplan als auch dynamische Aspekte und kann darüber als Ablaufplan dienen.

Durch die Beschreibung der Software auf obersten Ebenen dient die Softwarearchitektur der *Abstraktion von Komplexität*. Die Details wie das Innenleben der Komponenten können nach Hofmeister, Nord und Soni (2000) ausgeblendet werden, wenn sie sich nicht an den Grenzen von Architekturelementen befinden und nicht beschreiben, wie die Komponenten mit der Außenwelt oder der Ausführungsplattform interagieren. Der Detaillierungsgrad hängt nach Bosch (2000) von der Größe des Systems, den zu erfüllenden Qualitätsmerkmalen und dem Maß an Gewissheit der Umsetzung der Anforderungen ab, der erreicht werden soll. Er kann aber variieren und je nach Risiko für einzelne Bereiche höher sein, um sicher zu stellen, dass die Architektur alle Anforderungen erfüllt.

Die Dokumentation von Softwarearchitekturen besteht aus *verschiedenen Sichten* auf die *statischen und dynamischen Strukturen* der Architektur. Die Architektur von komplexe Systemen weist nach Posch, Birken und Gerdom (2011) vielfältige Aspekte auf, die in ihrer Gesamtheit nicht mehr dargestellt werden können und den Betrachter von der Menge an Informationen überfordert würden, weshalb sich die dargestellten Informationen auf bestimmte Aspekte beschränken müssen. Posch et al. (2011) schlagen folgende Sichten vor:

- **Kontextsicht:** Betrachtung des Systems von außen
- **Struktursicht:** Betrachtung der statischen Struktur des Innenlebens
- **Verhaltenssicht:** Betrachtung des dynamischen Verhaltes des Innenlebens
- **Verteilungssicht:** Betrachtung der Abbildung auf Artefakte, Prozessoren oder Teams

Kruchten (1995) fügt mit den *Szenarien* eine weitere Sicht ein, die wichtige Anwendungsfälle und -szenarien zeigt.

Softwarearchitektur basiert auf Entwurfsentscheidungen zu den Strukturen und Elementen der Architektur. Sie beschreibt eine Lösung und soll Anforderungen und Qualitätsziele verfolgen. Die Entscheidungen unterliegen mehreren Einflussfaktoren, die nachfolgend in Kapitel 3.3 genauer beschrieben werden.

Einen hohen Stellenwert für diese Arbeit stellt die Rolle des Softwarearchitekten dar, wie sie bereits von Beck im Rahmen der Definitionen von Softwarearchitektur im Kapitel 3.1 genannt wurde. Die Rolle des Softwarearchitekten kann unterschiedlich besetzt und ausgeführt werden. Der Softwarearchitekt übernimmt Aufgaben rund um den Lebenszyklus einer Softwarearchitektur. Deshalb soll in Kapitel 3.4 auf den Lebenszyklus von Softwarearchitekturen und in Kapitel 3.5 auf die Aufgaben des Softwarearchitekten genauer eingegangen werden.

3.1.1 Abgrenzung von Softwarearchitektur zu anderen Architekturen

Unternehmensarchitekturen (Enterprise Architectures) beschreiben Prinzipen und Methoden für den Entwurf und der Umsetzung einer Organisationsstruktur eines Unternehmens und dessen Geschäftsprozesse, IT-Systeme und Infrastruktur. Wenn nur der IT-Teil betrachtet wird, dann spricht man von der IT-Unternehmensarchitektur, welche mit der Softwarearchitektur eine gemeinsame Basis hat und sich ebenfalls mit Komponenten und deren Beziehungen beschäftigt. Jedoch ist nach Keller (2007) „[...] die Granularität [der IT-Unternehmensarchitektur] deutlich gröber, der Technikbezug geringer, dafür aber der Bezug zu betriebswirtschaftlichen Themen stärker“ (S. 15). Softwarearchitektur befasst sich mit der Architektur eines Systems, IT-Unternehmensarchitektur im Gegensatz dazu mit einer Menge an Systemen in einem Unternehmen (Keller, 2007).

Nach Dern (2009) ist Softwarearchitektur neben der „Fachlichen Architektur“ und „System- und Sicherheitsarchitektur“ Teil der Anwendungsarchitektur. Diese können in Form von Referenzarchitekturen mehrere IT-Systeme prägen und Teil einer IT-Architektur sein. Eine IT-Architektur ist nach Dern (2009) die „strukturierte Abstraktion existierender oder geplanter Informationssysteme“ (S. 18).

3.2 Ziele und Aufgaben von Softwarearchitektur

Bass et al. (2012) beschreiben folgende Ziele und Aufgaben von Softwarearchitektur:

- **Komplexität abstrahieren:** Komplexe Zusammenhänge werden durch Modelle auf unterschiedlichen Ebenen beschrieben. Dabei werden nicht relevante Informationen verdeckt und wichtige Details hervorgehoben.
- **Qualität schaffen:** Durch die Architektur werden Qualitätsattribute des Systems festgelegt.
- **Veränderung ermöglichen:** Die Entscheidungen über eine Architektur ermöglichen über das System zu urteilen und Veränderung zu ermöglichen, wenn das System weiterentwickelt wird.
- **Frühe Entscheidungen fördern:** Die Architektur legt die grundlegenden und schwer zu änderbaren Entwurfsentscheidungen fest, wenn ein System entwickelt wird.
- **Implementierung einschränken:** Die Architektur legt Strukturen und Muster für die Implementierung fest.
- **Evolutionäre Prototypen ermöglichen:** Nachdem eine Architektur erstellt wurde kann diese als Grundgerüst für Prototypen dienen und darüber analysiert als auch bewertet werden. Darüber kann das potentielle Risiko des Projekts vermindert werden.
- **Kosten und Zeitpläne verbessern:** Anhand der Architektur können der Softwarearchitekt und das Projektmanagement die Kosten und Zeitpläne zur Umsetzung des Projekts abschätzen.

- **Übertragbares und wiederverwendbares Modell erstellen:** Strukturen, Muster und Prinzipien einer Architektur können in Produktlinien und weiteren Projekten wiederverwendet werden.
- **Entwurfsalternativen einschränken:** Wenn sich Entwurfsentscheidungen bewährt haben können diese immer wiederverwendet werden und es muss nicht für jeden Fall eine neue Lösung gefunden werden. Dadurch wird die Komplexität in der Architektur niedrig gehalten.
- **Kommunikation fördern:** Dokumentation von Architekturen in Textform oder in Form von Modellen unterstützen den Softwarearchitekten in der Kommunikation und Abstimmung mit den Stakeholdern[5] des Systems.
- **Basis für Schulungen bilden:** Auf Basis der Dokumentation der Architektur können (neue) Stakeholder wie Entwickler zur Architektur geschult werden.

3.3 Einflüsse auf Softwarearchitektur

Die Architektur einer Software wird von verschiedenen Faktoren beeinflusst (vgl. Abbildung 3-1). Nach Hofmeister et al. (2000) lassen sich diese Einflussfaktoren wie folgt kategorisieren:

- organisatorische Faktoren
- technische Faktoren
- System- und Produktfaktoren

Die Faktoren können flexibel sein: Starke (2011) beschreibt Flexibilität eines Faktors als die „Aspekte eines Faktors, die zur Disposition stehen können“. Die Priorität und die konkrete Ausgestaltung von flexiblen Faktoren hängen vom jeweiligen Projekt ab und müssen unter den Projektbeteiligten abgestimmt werden.

Nicht alle Faktoren die ein Projekt beeinflussen sind architekturrelevant: Nach Starke (2011) sind aber Faktoren oft architekturrelevant, wenn sie neuartige Verfahren oder Techniken betreffen, wie beispielsweise Programmiersprachen und Entwicklungswerkzeuge.

Auch die Stabilität der Einflussfaktoren ist unterschiedlich: So ändern sich in den meisten Projekten pro Jahr 10-20% der Anforderungen an Software (Rupp & Sophist Group, 2001). Etablierte Werkzeuge und Technologien sowie auch bereits bestehende Alt- und Vorgängersysteme beeinflussen das System ebenso (Posch et al., 2011).

[5] Ein Stakeholder ist eine Person oder Gruppe, die durch ein Thema von Interesse beeinflusst wird und die Fähigkeit hat, es signifikant zu beeinflussen (positiv oder negativ) (Glicken, 2000).

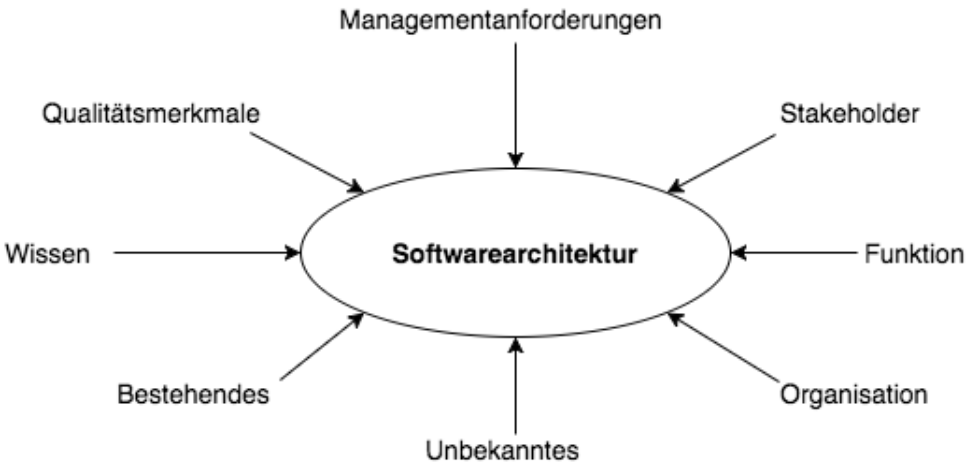

Abbildung 3-1: Einflüsse auf Softwarearchitektur [Eigene Darstellung in Anlehnung an Posch et al. (2011)]

Die Unternehmenskultur ist eine der organisatorischen Einflussfaktoren von Softwarearchitekturen. Wie die Kultur eines Unternehmens auf die Architektur einer Software wirkt, wird in dieser Arbeit in Kapitel 4 genauer beschrieben.

Nachfolgend sollen die einzelnen Faktorkategorien genauer erläutert und beschrieben werden.

3.3.1 Organisatorische Einflussfaktoren

Die organisatorischen Einflussfaktoren sind nach Hofmeister et al. (2000) diejenigen, die von der Organisation ausgehen und den Entwurf beeinflussen. Die Faktoren beziehen sich auf die Zeitplanung, das Budget, Fähigkeiten und Interessen der beteiligten Menschen. Sie beschreiben nicht direkt das Produkt, sondern stellen Aspekte der Organisation dar, die sich auf die Architektur auswirken können. Nach Posch et al. (2011) hat „auch [neben den anderen Einflussfaktoren] das organisatorische Umfeld, die Natur und Struktur der Organisation eine entscheidende Bedeutung.“ (S. 20)

Die nachfolgende Tabelle stellt typische organisatorische Einflussfaktoren in Anlehnung an Starke (2011) dar:

Faktor	**Aspekte**
Organisation und Struktur	
Organisationsstruktur beim Auftraggeber	Verteilung von Verantwortlichkeiten und Ansprechpartnern
Organisationsstruktur des Projektteams	Einsatz von Subunternehmern, Entscheidungsbefugnisse
Entscheidungsträger	Erfahrung, Risiko-/Innovationsbereitschaft
Bestehende Partnerschaften oder Kooperationen	Kooperationen mit anderen Softwareher-

	stellern und Dienstanbietern,
Eigenentwicklung oder externe Vergabe	Know-How des Lieferanten
Entwicklung als Produkt oder zur eigenen Nutzung	Wachstums- und Änderungsrate des Marktes
Ressourcen (Budget, Zeit, Personal)	
Festpreis oder Zeit/Aufwand?	
Zeitplan	Flexibilität und Festlegung der Zeitpunkte
Zeitplan und Funktionsumfang	Priorisierung zwischen Termineinhaltung und Funktionsumfang
Release-Plan	Zuordnung von Funktionsumfang und Terminen
Projektbudget	Budgetart (fest/variabel)
Budget für technische Ressourcen	Budgethöhe, Kauf oder Miete von Entwicklungswerkzeugen (Hard- und Software)
Team	Anzahl der Mitarbeiter und deren Qualifikation, Motivation, Verfügbarkeit
Organisatorische Standards	
Vorgehensmodell	Art des Vorgehensmodells (klassisch, agil)
Qualitätsstandards	Eingesetzte Qualitätsnormen (z.B. ISO 9000)
Entwicklungswerkzeuge	Vorgaben bezüglich der Entwicklungswerkzeuge (z.B. Integrierte Entwicklungsumgebung, Ticketsystem, Kommunikationssoftware, Middleware-Systeme, Modellierungswerkzeuge)
Konfigurations- und Versionsverwaltung	Vorgaben bezüglich Prozessen und Werkzeugen
Testwerkzeuge und -prozesse	
Abnahme- und Freigabeprozesse	Datenmodellierung und Datenbankdesign, Benutzeroberflächen, Geschäftsprozesse, Nutzung externer Systeme
Service Level Agreements	Einzuhaltende Verfügbarkeiten, Geplante Nichtverfügbarkeitszeiten
Juristische Faktoren	

Haftungsfragen	Rechtliche Konsequenzen von Nutzung oder Betrieb des Systems
Datenschutz	Anforderungen an „schutzwürdige“ Daten
Nachweispflichten	Juristische Nachweispflichten, Einhaltung von Revisionssicherheit
Internationale Rechtsfragen	Anforderungen von internationale Gesetzten bei internationalem Einsatz

Tabelle 3-1: Typische organisatorische Einflussfaktoren von Softwarearchitekturen in Anlehnung an Starke (2011)

3.3.2 Technische Einflussfaktoren

Neben den organisatorischen Faktoren beeinflussen auch technische Faktoren die Softwarearchitektur.

Die nachfolgende Tabelle stellt typische organisatorische Einflussfaktoren nach Starke (2011) dar:

Faktor	**Aspekte**
Hardware-Infrastruktur	Prozessoren, Speicher, Netzwerke, Firewalls und andere Hardware
Software-Infrastruktur	Betriebssystem, Datenbanksysteme, Middleware-Systeme, Kommunikationssysteme, Transaktionsmonitor, Webserver, Verzeichnisdienste
Systembetrieb	Betriebsart (z.B. Batch- oder Onlinebetrieb)
Verfügbarkeit der Laufzeitumgebung	24x7 Betrieb
Grafische Oberfläche	Vorgaben zum Design (Style Guides)
Bibliotheken, Frameworks und Komponenten	
Programmiersprachen	Art der Sprache (z.B. objektorientiert, funktional, strukturiert, deklarativ, Regelsprachen, kompilierte Sprachen)
Referenzarchitekturen	Vorgaben aus vorherigen oder übergeordneten Projekten/Produkten, Produktlinien
Analyse- und Entwurfsmethoden	Art der Methode (z.B. objektorientiert oder strukturiert)

Datenstrukturen	Bestehende Datenbanken und deren Strukturen
Programmierschnittstellen	Schnittstellen zu bestehenden Programmen
Programmiervorgaben	Vorgaben zur Programmierung (Coding Guidlines)
Technische Kommunikation	Art der Kommunikation (z.B. synchron, asynchron oder über bestimmte Protokolle)

Tabelle 3-2: Typische technische Einflussfaktoren von Softwarearchitekturen in Anlehnung an Starke (2011)

3.3.3 System- und Produktfaktoren

Die funktionalen (Required Features) und nicht-funktionalen (Required Constraints) Anforderungen an das System oder Produkt sind nach Starke (2011) die System- und Produktfaktoren, die eine Architektur beeinflussen. Sie werden innerhalb der Anforderungsanalyse (Requirements Engineering) erarbeitet und dokumentiert. Die funktionalen Anforderungen beschreiben die Funktionalität des Systems und können zur Laufzeit festgestellt werden. Die nicht-funktionalen Anforderungen, auch Qualitätsanforderungen genannt, schränken die Möglichkeiten des Entwurfs ein, um Qualitätsmerkmale der Architektur zu erreichen (Bass et al., 2012; Starke, 2011; Zörner, 2015). Ein Qualitätsmerkmal, auch Qualitätsattribut genannt, ist nach Zörner (2015) ein messbares oder testbares Merkmal eines Systems, das angibt, „wie eine Funktion bereitgestellt werden soll." (S.42)

Die ISO/IEC 9126[6] (ISO, 2001) definiert Kategorien für Qualitätsmerkmale (vgl. Abbildung 3-2).

Nach Starke (2011) fehlen der ISO/IEC 9126 jedoch zwei wichtige Qualitätsmerkmale von Architekturen:

- Verständlichkeit der Architektur
- Testbarkeit der Architektur

Systeme werden entworfen, um Geschäftsziele von einem oder mehrere Unternehmen zu erreichen und Erfolg zu haben. So wollen Unternehmen Gewinne machen, Marktanteile verbessern, Aktienkurse steigern. Viele dieser Ziele haben in Form von Qualitätsanforderungen Einfluss auf die Architektur. Ein System soll beispielsweise eine bestimmte Reaktionszeit aufzeigen, um das Produkt von denen der Mitbewerber abzuheben (Bass et al., 2012).

Hofmeister et al. (2000) sind der Meinung, dass die Produktfaktoren einen höheren Einfluss auf die Architektur haben, als die organisatorischen oder technischen Faktoren. So können sich die

[6] Es existiert ein Nachfolgestandard ISO 25000 - 25030. Nach Starke (2011) und Zörner (2015) findet dieser aber noch wenig Anwendung in der Praxis, weshalb auch in dieser Arbeit auf die ISO 9126 verwiesen wird.

Produktfaktoren drastisch ändern, wenn sich der Markt ändert, für den das Produkt bestimmt ist oder auch während der Entwicklung des Systems.

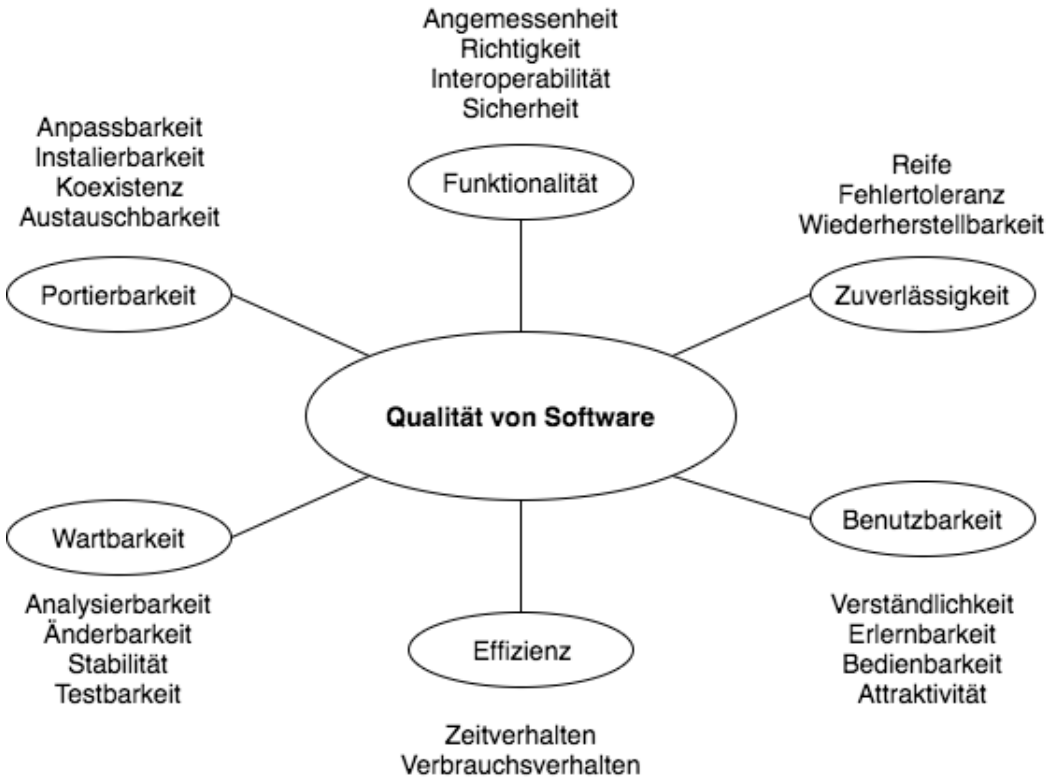

Abbildung 3-2: Kategorien für Qualitätsmerkmale nach ISO/IEC 9126 [Eigene Darstellung]

3.3.4 Wissen des Softwarearchitekten

„We are all products of our experiences, architects included."
(Bass, Clements, & Kazmann, 2012, S. 51)

Der Entwurf einer Softwarearchitektur wird vom Softwarearchitekten durchgeführt. Dabei beeinflussen ihn Erfahrungen, die er mit seiner Arbeit mit bestimmten Architekturen und Architekturmustern gemacht hat. Hat er gute Erfahrungen mit einem Architekturmuster gemacht, dann wird das Architekturmuster mit hoher Wahrscheinlichkeit wiederverwendet, bei schlechten Erfahrungen hingegen wird des er Architekt meiden das Muster erneut einzusetzen. Neben den Erfahrungen ist auch das Wissen, das der Architekt im Rahmen seiner Ausbildung und in Schulungen erworben hat wichtig beim Entwurf der Architektur. (Bass et al., 2012)

Auf die Rolle des Softwarearchitekten und seinen Aufgaben soll in Kapitel 3.5 wird genauer eingegangen werden. Zuvor soll beschrieben werden, wie Softwarearchitekturen entstehen.

3.4 Entstehung von Softwarearchitekturen

In der Softwareentwicklung werden unterschiedliche Vorgehensmodelle bzw. Entwicklungsprozesse eingesetzt, wobei sich die agilen Prozesse in den letzten Jahren durchgesetzt haben und mehr an Bedeutung gewinnen.

Unabhängig vom Entwicklungsprozess werden nach Bass et al. (2012) einige Aktivitäten beim Entwurf der Softwarearchitektur regelmäßig durchgeführt. Der gewählte Entwicklungsprozess bestimmt, wie oft und in welcher Ausprägung diese Aktivitäten durchgeführt werden müssen. Bass et al. (2012) nennen folgende Aktivitäten:

1. **Identifikation des Geschäftsszenarios für das System:** Kosten, Einsparungen, Gewinne und Risiken, die mit dem neuen System verbunden sind müssen ermittelt werden. Ebenso müssen Verknüpfungen mit anderen Systemen geprüft werden. Daraus leiten sich Anforderungen und Geschäftsziele ab.
2. **Verstehen der grundlegenden architekturellen Anforderungen:** Die funktionalen Anforderungen und Qualitätsanforderungen an das System müssen ermittelt und verstanden werden.
3. **Erstellen oder Auswählen der Architektur:** Die grundlegende Architektur des Systems muss entworfen oder eine Referenzarchitektur ausgewählt werden.
4. **Dokumentieren und Kommunizieren der Architektur:** Damit das System anhand der Architektur umgesetzt werden kann, muss die Architektur dokumentiert und die Dokumentation den Entwicklern kommuniziert werden.
5. **Analysieren oder Bewerten der Architektur:** Im Rahmen des Entwurfs entstehen mehrere Design-Alternativen, um (Qualitäts-)Anforderungen umzusetzen. Manche davon werden sofort verworfen und andere übernommen. Die Architektur muss analysiert und die Qualitätsmerkmale bewertet werden, um sicherzustellen, dass das System den erwarteten Anforderungen entspricht.
6. **Implementieren und Testen des auf der Architektur basierenden Systems:** Das System wird auf Basis der kommunizierten Architektur und dessen Vorgaben sowie Konzepte entwickelt.
7. **Überprüfen der Berücksichtigung der Architekturvorgaben in der Implementierung:** Nach Bass et al. (2000) werden nicht immer alle Vorgaben und Konzepte der Architektur von den Entwicklern berücksichtigt. Als Gründe werden hier menschliche Fehler und der fehlende Blick auf das große Ganze genannt. Die Implementierung muss deshalb auf Architekturkonformität geprüft werden.

Toth (2014) beschreibt den Prozess mit einem iterativen Ansatz, den er in Form einer Brezel darstellt und in dem sich die von Bass et al. (2012) definierten Aktivitäten wiederfinden lassen (vgl. Abbildung 3-3).

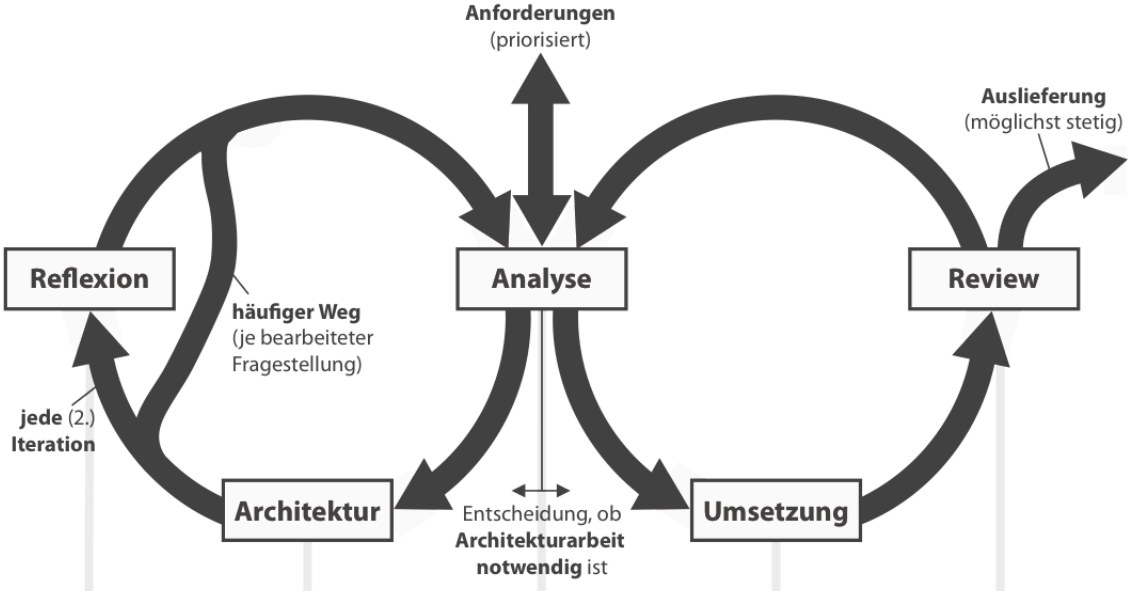

Abbildung 3-3: Architekturbrezel [Quelle: Toth (2014)]

Der Prozess beginnt mit den priorisierten *Anforderungen*. Auf Basis einer *Analyse* werden die Anforderungen für die nächste Iteration ausgewählt und entschieden, ob diese direkt umgesetzt werden können, oder Architekturarbeit notwendig ist. Architekturarbeit wäre notwendig, wenn die Anforderung eine weitreichende Entscheidung in der Umsetzung voraussetzt. Der Prozess teilt sich in einen **Architekturzyklus** (links) und einen **Implementierungszyklus** (rechts). Im Implementierungszyklus wird das eigentliche System bzw. die Software implementiert, getestet und ausgeliefert. Hier sollte nach Toth (2014) die meiste Zeit aufgewendet werden, da dies der produktive Zyklus ist und die Werte, d. h. die auslieferbare Software, erzeugt werden. Im Architekturzyklus werden teure und schwer änderbare Entscheidungen getroffen. Dazu wird zuvor je nach Anforderung eine umfangreiche Analyse der Anforderungen und der damit verbundenen Auswirkungen durchgeführt, um Sicherheit für die Entscheidung zu gewinnen und das Risiko der Fehlentscheidung zu mindern. Der Architekturzyklus stellt daher auch eine Risikominderungsmaßnahme dar.

Toth (2014) empfiehlt jede zweite Iteration eine *Reflexion* durchzuführen. Dabei sollen die vorbereiteten Entscheidungen erneut mit weiteren Stakeholdern besprochen und reflektiert werden.

3.5 Rolle des Softwarearchitekten

In Kapitel 3.1 wurde Softwarearchitektur als die wichtigen Entscheidungen über die Organisation eines Softwaresystems beschrieben. Diese Entscheidungen werden vom Softwarearchitekten oder einem Team gefällt. Auf die Besetzung der Rolle des Softwarearchitekten soll nachfolgend in Kapitel 3.5.1 näher eingegangen werden.

Die wichtigen Entscheidungen müssen oft in frühen Phasen des Projekts gefällt werden. Dazu setzen Architekten auf ihre Erfahrung und ihre Fähigkeiten zu abstrahieren, um die Entscheidung richtig zu fällen. Jedoch hat sich gezeigt, dass zu frühes Entscheiden in Projekten auch zu

höheren Kosten und nicht getroffenen Deadlines führen kann. Auf Basis dieser Einsicht und den gewachsenen Anforderungen an Auslieferung hochqualitativer Software hat sich nach Hohpe, Ozkaya, Zdun und Zi (2016) die architekturelle Entscheidungsfindung von Entwicklungsteams maßgeblich verändert. Die Rolle des Architekten hat sich verändert: Innovative Internetfirmen haben demnach wenige bis keine Architekten. Architekturentscheidungen sind dort ein kollektiver Prozess, der von Entdecken und Lernen geprägt ist, anstatt durch eine einzige Person durchgeführt zu werden. Die Rolle des Architekten ist nur noch virtuell vorhanden. Die Architekturentscheidungen dieser Teams finden sich im Quellcode wieder und können von allen eingesehen werden. Zusätzliche Dokumentation findet sich in Wiki Systemen wieder. (Hohpe et al., 2016)

Auch sind bei den genannten Internetfirmen weniger architekturelle Entscheidungen notwendig, da die Produkte und Services dieser Firmen über das Internet angeboten werden und auf fertige „Platform as a Service“-Lösungen zurückgegriffen werden kann. Heutige Frameworks und Middleware-Plattformen nehmen architekturelle Entscheidungen ab, indem sie diese Entscheidungen bereits vorab treffen. Tools nehmen dem Architekten eine Menge Arbeit ab (Hohpe et al., 2016). Fowler und Dörnenburg (2016) sind der Meinung, dass die meisten Aufgaben, die Architekten traditionell ausgeführt haben, heute von Entwicklern, von Werkzeugen oder gar nicht mehr durchgeführt werden müssen.

Anforderungen der Kunden aus dem Markt fordern flexible Systeme. Diese Systeme können aufgrund der vorherrschenden Technologien einfacher erstellt werden als noch vor zehn Jahren, als das Entwickeln von verteilten und kommunizierenden Systemen noch eine größere Herausforderung darstellte. Anstatt monolithischer Systeme werden heute dynamische Microservice-Architekturen entwickelt, die schneller aktualisiert und besser skaliert werden können. Diese bringen aber auch eine höhere Komplexität mit sich. Es lassen sich Konzepte wie „You build it, you run it“ und DevOps finden, wodurch die Aufgaben des Architekten zusätzlich um Themen rund um Betrieb und Wartung wie Monitoring und Continuous Deployment erweitert werden (Hohpe et al., 2016; Bass, Weber, & Zhu, 2015).

Durch die breite Akzeptanz von agilen Methoden hat sich die Rolle des Architekten verändert. Sherman und Hadar (2015) haben die „neue“ Rolle des Architekten analysiert. Neben den genannten zusätzlichen Themen sind auch Leadership, Mentoring sowie Training für den Architekten wichtige Aufgaben des Softwarearchitekten.

Nachfolgend soll die Besetzung der Rolle wie auch die Aufgaben des Softwarearchitekten näher erläutert werden.

3.5.1 Besetzung der Rolle

Die Rolle des Softwarearchitekten kann unterschiedlich besetzt werden. Toth (2014) beschreibt vier mögliche Formen der Besetzung der Rolle des Softwarearchitekten (vgl. Abbildung 3-4).

Abbildung 3-4: Mögliche Besetzung der Architektenrolle [Quelle: Toth (2014)]

Nach Toth (2014) sind die Formen des Architekturagenten und des unterstützenden Architekten als Alternativen zwischen dem klassischen Architekten, der alle Entscheidungen trifft und keinem benannten Architekten, also der Entscheidung durch die Entwickler, zu verstehen.

Konzeptionelle Integrität ist nach Toth (2014) der größte Vorteil des einzelnen Architekten. Jedoch können schnell Elfenbeintürme entstehen: Der Architekt fällt allein die Entwurfsentscheidungen und übergibt diese später, je nach Situation, nach langer Zeit dem Entwicklungsteam. Das Entwicklungsteam hat dann schon bereits eigene Annahmen und Entscheidungen getroffen und lehnt die Architekturentscheidungen des Architekten ab, wodurch die Konsistenz der Architektur nicht eingehalten wird (Kruchten, 2008; Ambler, 2002).

Wenn sich niemand für die Architektur verantwortlich fühlt und diese leitet entsteht eine implizite Architektur oder auch *Accidential Architecture* genannt (Booch, 2006). Diese kann nicht mehr einfach nachvollzogen werden und damit schlecht gewartet werden.

Neben Toth (2014) beschreibt Faber (2010) die Form mit einem externen Architekten, der nicht Teil des Entwicklungsteams ist, was dem unterstützenden Architekten von Toth (2014) gleicht.

Britto, Šmite und Damm (2016) beschreiben den Einsatz von dedizierten Architekturteams, deren zentrale Rolle es ist, die Entwicklungsteams zu coachen und zu unterstützen.

Posch et al. (2011) positionieren den Softwarearchitekten zwischen Projektleitung und Entwickler. Die Rolle des Softwarearchitekten soll dediziert von einer Person übernommen werden, außer es handelt sich um ein sehr kleines Projekt von maximal drei Personen. Posch et al. begründen die Besetzung darin, dass sonst wichtige Aufgaben in den unterschiedlichen Projektphasen nicht erledigt werden können. Als Beispiel nennen sie die Aufgaben des Projektmanagements während der Architekturphase sowie den Softwarearchitekten, der aufgrund ihm zugewiesener zu implementierender Arbeitspakete keine Zeit für Dokumentation und Kommunikation der Architektur aufbringen kann.

Das agile Vorgehensmodell Scrum empfiehlt cross-funktional arbeitende Teams bestehend aus sieben +/- zwei Mitgliedern, in denen alle notwendigen Fähigkeiten vorhanden sind ein Produkt zu entwerfen und auszuliefern. Alle Mitglieder des Teams übernehmen Analyse-, Test-, Design- und auch die Architekturaufgaben. Nach den agilen Prinzipien, auf denen Scrum beruht, entstehen „die besten Architekturen [...] in selbstorganisierten Teams“ (agilemanifesto.org, 2001). Teammitglieder die nur an der Architektur entwickeln stellen nach Larmann und Vodde (2010) ein Anti-Pattern dar.

Abrahamsson, Babar und Kruchten (2010) beschreiben eine Konstellation mit einem externen und einem internen Architekten („Two architects setup“). Der externe Architekt trifft große Entscheidungen, nimmt Anforderungen auf und fungiert als externer Koordinator. Der interne Architekt kümmert sich um die Kommunikation innerhalb des Teams, fungiert als Mentor und unterstützt bei der Problemlösung und der Implementierung.

Im folgenden Teil dieser Arbeit verstehen wir unter dem Softwarearchitekten alle Mitarbeiter und Führungskräfte in einem Unternehmen, welche die in dem folgenden Kapitel beschriebenen Aufgaben ausführen, außer es wird explizit eine andere Konstellation beschrieben.

3.5.2 Aufgaben des Softwarearchitekten

Ein Softwarearchitekt ist verantwortlich für Design und technische Entscheidungen innerhalb des Softwareentwicklungsprozesses (McBride, 2007). Architekturen die mit Geschäftszielen abgestimmt sind und diese Ziele erfüllen sollen lassen sich nach Bass et al. (2012) nicht nur durch das Untersuchen von bestehenden Architekturen und deren Mängeln entwerfen. Die organisatorischen und menschlichen Hintergründe, die zu diesen Architekturen geführt haben müssen ebenfalls berücksichtigt werden. Neben der Rolle als technischen Experten muss der Architekt unter anderen auch nicht-technische Fähigkeiten wie Führung, Kommunikation und Coaching besitzen (Clements, Kazman, Klein, & Verma, 2007; Bass et al., 2012). Die Aufgaben des einzelnen Softwarearchitekten oder je nach Besetzung des Softwarearchitekturteams, lassen sich demnach in technische und nicht-technische Aufgaben aufteilen.

Posch et al. (2011) teilen ein Projekt in die Projektphasen Anforderungsanalyse, Architekturdesign und Feindesign/Implementierung ein, was einem nicht-iterativen Vorgehen entspricht. Daraufhin ordnen sie die nicht phasenübergreifenden Aufgaben des Architekten den einzelnen Phasen zu. Toth (2014) hingegen beschreibt in seinem Buch „Vorgehensmuster für Softwarearchitekturen“ 29 Vorgehensmuster und zugehörige Praktiken, die den jeweiligen Aufgaben in

Architektur- und Implementierungszyklen des iterativen Prozesses (vgl. „Architekturbrezel“ in Kapitel 3.4) zugeordnet werden können.

Da agile Vorgehensmodelle wie Scrum und extreme Programming (XP) gegenwärtig einen hohen Einsatz verzeichnen (VersionOne, 2016), sollen nachfolgend die Aufgaben des Softwarearchitekten in den Phasen der Architekturbrezel nach Toth (2014) beschrieben werden. Die Aufgaben der jeweiligen Phasen werden durch die von Bass et al. (2012) genannten Aufgaben ergänzt.

3.5.2.1. Aufgaben in der Analysephase

In der Analysephase ist es die Aufgabe des Softwarearchitekten die Anforderungen aufzubereiten. Dazu müssen zuerst die Anforderungen der Kunden, Organisation und dem Geschäft aufgenommen werden, falls diese Aufgabe nicht durch einen dedizierte Requirements Engineer übernommen wird. Bei Einsatz des agilen Vorgehensmodells Scrum können die Anforderungen auch mit dem Product Owner abgestimmt werden (Schwaber & Beedle, 2002). Es gilt die grundlegenden Bedürfnisse und Erwartungen der Kunden zu verstehen und zu prüfen, ob alle notwendigen Anforderungen aus dem Geschäft als auch die Geschäftsziele berücksichtigt wurden. Das zu lösende Problem muss hinreichend verstanden werden. Dazu muss der Architekt die erhobenen funktionalen Anforderungen und die Qualitätsanforderungen analysieren und der Umfang des Projekts oder Produkts muss eingegrenzt werden (Bass et al., 2012).

Zusätzlich muss der Architekt über Kenntnisse über das Umfeld und Domänenwissen, beispielsweise über die Einflüsse der Märkte und den Kunden erlangen, damit er sich mit Stakeholdern abstimmen kann. Durch dieses Wissen kann er aktiv am Geschäftsmodell mitarbeiten und Lösungen für die Markteinflüsse erarbeiten (Sherman & Hadar, 2015).

Toth (2014) empfiehlt für das effektive Erheben von Architekturanforderungen einen initialen Anforderungs-Workshop. Der Workshop hat das Ziel eine ausreichende Liste an Architekturanforderungen zu erzeugen, so dass mit dem Projekt begonnen werden kann (Larmann & Vodde, 2010). Der Architekt muss hier nach Toth (2014) die zentralen Qualitätsanforderungen identifizieren. Je nach Domäne und Aufgabe des Systems können die Qualitätsanforderungen unterschiedlich hohe Prioritäten aufweisen. Toth (2014) nennt folgende Techniken für das Erfassen von Qualitätsanforderungen:

- **Produktkarton:** Die Teilnehmer erstellen einen Produktkarton, der alle zentralen Funktionalitäten und Merkmale des Systems aus Sicht eines potentiellen Käufers darstellt. Daraus lassen sich dann die Qualitätsanforderungen ableiten.
- **Launch-Announcement (Pressemittelung):** Durch das Anfertigen einer (nicht zur Veröffentlichung bestimmten) Pressemitteilung werden, wie auch beim Produktkarton, alle zentralen Funktionalitäten erarbeitet und daraus Qualitätsanforderungen abgeleitet.
- **Produkt-Canvas:** Der Product Canvas nach Pichler (2012) stellt eine Alternative zum traditionellen Product Backlog in Scrum-Projekten dar und gibt neben Informationen zur Zielgruppe einen Überblick über das gesamte Produkt und dessen Details.

- **Brainwriting:** Die Teilnehmer sammeln in Kleingruppen Anforderungen und Rahmenbedingungen für das Projekt. Die Ergebnisse werden anschließend in der großen Gruppe konsolidiert und gruppiert.

Für die Qualitätsanforderungen können schließlich Qualitätsszenarien entwickelt werden, mit denen die Qualitätsanforderungen gemessen oder bewertet werden können (vgl. Abbildung 3-5).

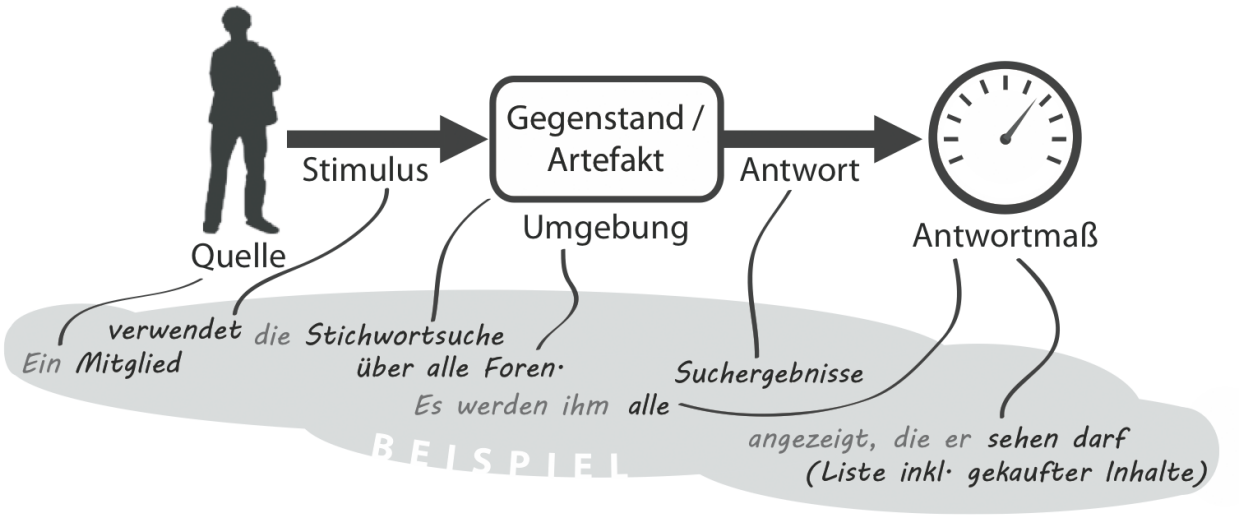

Abbildung 3-5: Qualitätsszenario inkl. Beispielszenario [Quelle: Toth (2014)]

Die erhobenen Anforderungen müssen nach Toth (2014) in folgenden Iterationen in Anforderungspflege-Workshops detailliert werden. Dabei sollen wichtige architektonische Herausforderungen gefunden werden und die Anforderungen mit Akzeptanzkriterien detailliert werden. Die Aufgabe des Softwarearchitekten ist das Miteinbeziehen der Stakeholder in den Workshops. Eine höhere Beteiligung und Miteinbeziehung der Stakeholder führte in den Untersuchungen von Abelein, Sharp und Paech (2013) zu erfolgreicheren Systemen.

Der Architekt muss die Ziele der Architektur schärfen und eine Architekturvision entwerfen. In folgenden Iterationen muss er die Architekturvision weiterentwickeln (Bass et al., 2012). Zum Start eines Scrum-Projekts muss grob klar sein, wie das Produkt aussehen und was es leisten soll. Mit der Vorstellung, welche Architekturprinzipien und Technologien eingesetzt werden sollen, kann nach Roock und Pichler (2011) die Machbarkeit des Produkts beurteilt, ein Kosten- und Zeitrahmen ermittelt und ein Team zusammenstellt werden. Roock und Pichler (2011) bezeichnen diese Vorstellung als Architekturvision, die eine zielgerichtete und nachhaltige Entwicklung sicherstellen soll. Die Architekturvision hilft die Stakeholder besser zu integrieren und dient weiterhin dem Architekturteam die gleiche Vorstellung des zu entwerfenden Systems zu vermitteln (Malan & Bredemeyer, 2000). Die Erstellung der Architekturvision erfolgt vor Beginn des ersten (Produktinkrement erzeugenden) Sprints unter Verwendung von Prototyping und wird durch Prototypen validiert („Proof it with code") (Ambler, 2002).

In der Literatur stimmen nicht alle Autoren damit überein, dass Architekten auch Aufgaben des Projektmanagements übernehmen sollen. Farenhorst, Hoorn, Lago und van Vliet (2011) sehen Projektmanagement nicht als Aufgabe des Softwarearchitekten. Nach Kruchten (1999) hingeg-

gen benötigen Manager Unterstützung durch Softwarearchitekten bei der Aufwandsschätzung und der Aufteilung von Arbeitspaketen auf mehrere Teams. Das Mitwirken von Architekten im Projektmanagement hilft den Managern die Komplexität von Problem und Lösung besser handzuhaben (McBride, 2007).

3.5.2.2. Aufgaben in der Architekturphase

„The life of a software architect is a long, and sometimes painful, succession of suboptimal decisions made partly in the dark.“
(Falessi, Cantone, Kazman, & Kruchten, 2011, S. 8)

Nach Analyse und Verstehen der Anforderungen sowie Identifikation der architekturrelevanten Fragestellungen muss der Softwarearchitekt Entscheidungen über Lösungen fällen, um die Architektur zu entwerfen. Architekturrelevante Fragestellungen erkennt man nach Toth (2014) wie folgt:

- Ist die Entscheidung später nur schwer zu ändern?
- Ist die Umsetzung der Entscheidung eher teuer?
- Werden sehr hohe, qualitative Anforderungen gestellt? (Hochsicherheit, Hochverfügbarkeit, Hochperformanz etc.)
- Lassen sich Anforderungen nur schwer in Bestehendes abbilden?
- Ist die eigene Erfahrung im Lösungsspektrum schwach?

Wie im Einstiegszitat dieses Kapitels erwähnt, ist die Entscheidung für den Architekten nicht immer einfach zu treffen. Nach Falessi et al. (2011) liegt das an folgenden Gründen:

- Anforderungen werden meist informell erfasst während Softwarearchitekturen informell spezifiziert werden. Dies führt zu einer semantischen Differenz.
- Qualitätsmerkmale sind in Modellen schwierig festzuhalten.
- Software wird oft iterativ entwickelt. Einige Anforderungen sind in den ersten Iterationen noch nicht bekannt. Manche Entscheidungen müssen unter Unsicherheit getroffen werden.
- Architekturentscheidungen sind schwer zu ändern, weil auf ihnen meist weitere Entscheidungen beruhen.
- Stakeholder haben eine andere Sicht auf die Aspekte einer Softwarearchitektur und die damit verbundene Ziele und Erwartungen. Architekten müssen die richtige Balance der Einflüsse aller Stakeholder berücksichtigen.

Bei architekturrelevanten Fragestellungen und Entscheidungen empfiehlt Toth (2014) architektonische Mittel, um das Risiko einer Fehlentscheidung zu minimieren. Architektonische Mittel sind nach Toth (2014):

- Genaue Analyse nichtfunktionaler Anforderungen
- Konzeption und Modellierung möglicher Lösungen
- Definition von Prinzipien
- Erstellung von Prototypen und Spikes
- Breite Vermittlung und Etablierung von Entscheidungen

Die Entscheidungen sollten anschließend festgehalten werden. Nach Toth (2012) besteht sogar eine Pflicht zur Dokumentation der Entscheidungen, da durch die Entscheidungen meist viele Projektbeteiligte betroffen sind und diese die Entscheidung und dessen Motivation nachvollziehen können sollten. Andernfalls entstehen sonst nach Toth (2012) inkonsistente Lösungen, die immer wieder diskutiert werden. Nach Zörner (2015) sollten neben der Fragestellung die Einflussfaktoren der Entscheidung, die getroffenen Annahmen, die betrachteten Lösungen und die Entscheidung selbst festgehalten werden.

Der Architekt sollte die entworfene Architektur zielorientiert dokumentieren, da sonst nur eine implizite Architektur entsteht, die zu einer zufälligen Architektur führen kann (Zörner, 2015). Dies kann durch Konzepte in Textform, Modelle und Diagramme erfolgen. Zörner (2015) nennt folgende Funktionen der Dokumentation:

- **Architekturarbeit unterstützen:** Die Dokumentation während der Implementierung hilft weitere Lösungen zu finden und diese im Team zu kommunizieren, wodurch Konsistenz in der Architektur erreicht wird.
- **Nachvollziehbarkeit der Architektur sicherstellen:** Neue Mitarbeiter können sich an der Dokumentation orientieren und zurechtfinden. Stakeholder wie Kunden, Produktverantwortliche, Endnutzer und Betrieb können die Architektur nachvollziehen und in Bezug auf dessen Angemessenheit bewertbar machen.
- **Umsetzung und Weiterentwicklung leiten:** Durch dokumentierte Strukturen und querschnittliche Aspekte werden die Lösungen dokumentiert. Dies führt zu Konsistenz zwischen Architektur und Implementierung bei Umsetzung und Weiterentwicklung des Systems.

Auch wenn im Projekt am Entwurf an der Architektur mehrere Personen beteiligt sind, empfiehlt Zörner (2015) eine Person zur Dokumentation abzustellen, die eine Zusammenfassung der Architektur erstellt.

3.5.2.3. Aufgaben in der Reflexionsphase

In der Reflexionsphase hat der Softwarearchitekt nach Toth (2014) die Aufgabe, die Entscheidungen bzw. Lösungen aus der Architekturphase den Stakeholdern vorzustellen und Feedback einzuholen. Toth (2014) nutzt hier den Namen Reflexion, da Bewertung mit schwergewichtigen Bewertungsverfahren wie ATAM (Architecture Tradeoff Analyses Method) und Werkzeugen verbunden wird. Die Reflexion kann in Form eines Workshops durchgeführt werden (vgl. Abbildung 3-6). Dabei können Risiken, Kompromisse oder weitere Arbeitsaufgaben entstehen (vgl.

Abbildung 3-7). Die Abbildung 3-4 zeigt, welche Stakeholder der Softwarearchitektur möglicherweise mit einbezogen werden. Zusätzlich werden die wichtigsten Qualitätsszenarien durchgesprochen.

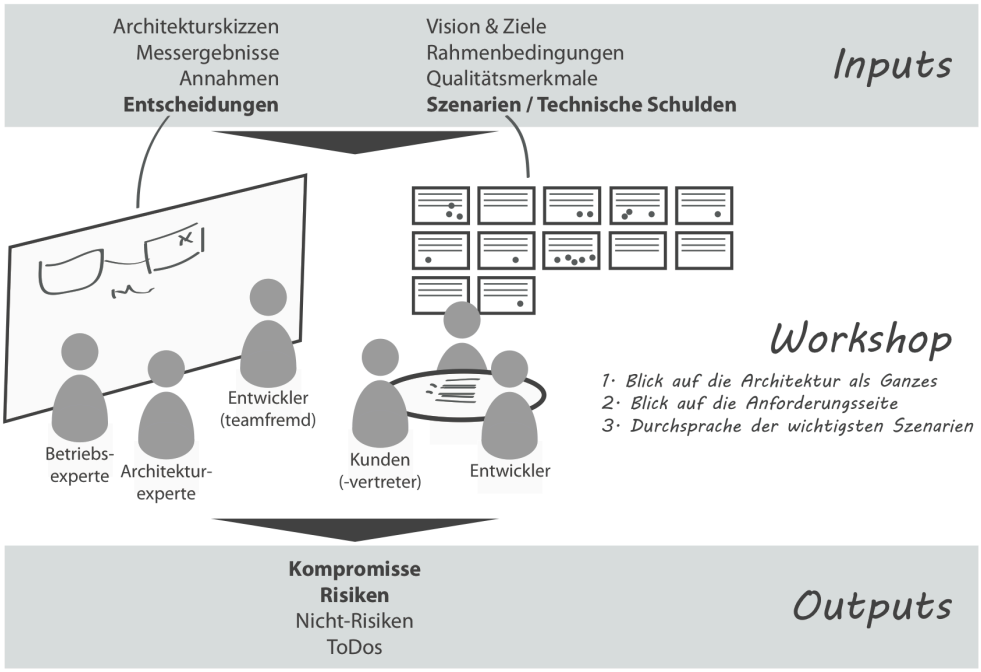

Abbildung 3-6: Architekturbewertungsworkshop (Reflexion) [Quelle: Toth (2014)]

Bass et al. (2012) nennen als Aufgabe des Architekten die Analyse und Bewertung der Architektur. Dabei soll überprüft werden, ob die Architektur den funktionalen Anforderungen und Qualitätsanforderungen genügt. Dazu gehört auch die Wahl der richtige Bewertungsmethode.

Sherman und Hadar (2015) sehen das Review der Softwarearchitektur als Möglichkeit zum Lernen für den Architekten. Durch das Review erhält der Architekt Feedback zu seiner Arbeit, wodurch er die Architektur verbessern kann.

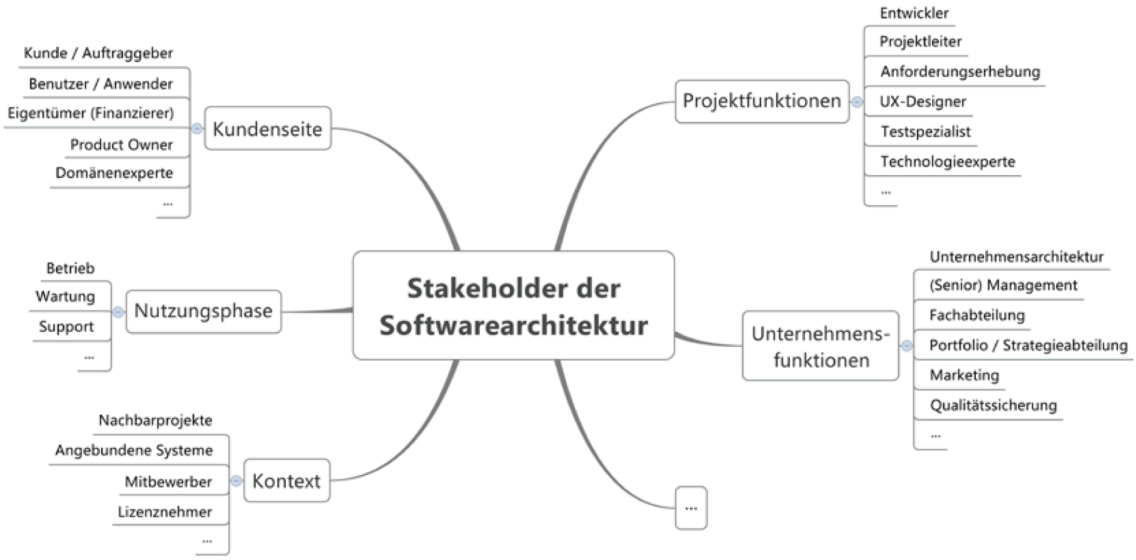

Abbildung 3-7: Stakeholder der Softwarearchitektur [Quelle: Toth (2014)]

3.5.2.4. Aufgaben in der Umsetzungsphase

In der Umsetzungsphase wird die Lösung auf Basis der Architektur hergestellt, d.h. die Software wird entwickelt und dabei die in der Architekturphase definierten Strukturen, Komponenten und Prinzipien berücksichtigt. Nach Toth (2014) kann die Umsetzung risikoreich sein, weil manche Anforderungen sich nur schwer in bestehenden Applikationen abbilden lassen oder von den Entwicklern nicht richtig verstanden werden. Die Aufgabe des Architekten in der Umsetzungsphase ist die Unterstützung des Entwicklungsteams. Während der Implementierung können Herausforderungen auftreten, die in der Analysephase nicht berücksichtigt wurden und die eine schnelle Klärung erfordern, um die Implementierung nicht zu behindern. Toth (2014) empfiehlt in diesem Fall kurzfristige Architekturworkshops einzuberufen, um darin die offenen Punkte im Team zu besprechen, Feedback einzuholen und eine Lösung zu erarbeiten.

Sherman und Hadar (2015) nennen ebenfalls Mentoring als Aufgabe des Softwarearchitekten. Sie beziehen sich dabei auf neue Technologien und Techniken wie Test-Driven-Development. Nach Sherman und Hadar (2015) fällt unter Code Reviews auch die Kommunikation mit der Qualitätssicherung bezüglich Fehler, die aufgrund des Architekturentwurfs entstanden sind, in den Aufgabenbereich des Architekten.

Im Gegensatz zu Clements et al. (2007) gehört nach Bass et al. (2012) darüber hinaus das Entwickeln von Source Code ebenfalls mit zu den Aufgaben des Architekten.

3.5.2.5. Aufgaben in der Review-Phase

In der Review-Phase wird die entwickelte Software überprüft: Die Software wird getestet und Metriken ausgewertet. Die Aufgabe des Architekten ist es, die Architekturziele und qualitativen Eigenschaften der Software zu überprüfen. Dies geschieht in dem der Architekt die entwickelte Software gegen den Architekturentwurf abgleicht, d.h. prüft, ob die in der Architekturphase definierten Strukturen, Komponenten und Prinzipien von den Entwicklern berücksichtigt wurden (Toth, 2014).

3.6 Microservice-Architekturen

Microservices sind ein Ansatz zur Modularisierung von Softwarearchitekturen (Wolff, 2016), der in den letzten Jahren stark an Interesse gewonnen hat (Google Trends, 2017). Dieses Kapitel soll ein grundsätzliches Verständnis des Ansatzes für das spätere Kapitel 4.2 vermitteln, in dem die Wirkungen der Microservice-Architektur auf die Kultur eines Unternehmens näher betrachtet werden.

Der grundlegende Ansatz von Microservice-Architekturen basiert auf der Unix-Philosophie (Wolff, 2016): Mehrere Programme, nachfolgend Microservices oder Services genannt, die für sich unabhängig sind, zusammenarbeiten, über eine universelle Schnittstelle kommunizieren und unabhängig voneinander ausgetauscht werden können. Richard Raymond beschreibt den Ansatz in „The Art of Unix Programming“ folgendermaßen (Raymond, 2008):

„The only way to write complex software that won't fall on its face is to hold its global complexity down to build it out of simple parts connected by well-defined interfaces, so that most problems are local and you can have some hope of upgrading a part without breaking the whole.“

Microservices sind nicht eindeutig definiert, lassen sich jedoch nach Newman (2015), Fowler und Lewis (2015) sowie Wolff (2016) mit den folgenden Eigenschaften beschreiben:

- **Klein:** Die Größe eines Microservices kann an unterschiedlichen Kriterien festgelegt werden:
    - Struktur der fachlichen Domäne/Business Capabilities
    - Lines of Code (LoC) der Implementierung (variiert jedoch je nach Technologie)
    - Größe des Teams, welches den Microservice entwickelt und betreut
    - Benötigte Zeit bis der Service neu implementiert werden könnte
    - Intensität der Kommunikation mit anderen Services
- **Eine Verantwortlichkeit:** Microservices sollen nur eine Aufgabe erledigen.
- **Unabhängig:**
    - **Deployment:** Microservices sind eigenständige Applikationen, die unabhängig voneinander ausgeliefert und in eigenständigen Prozessen oder virtuellen Maschinen betrieben werden können.
    - **Datenbank:** Jeder Microservice sollte eine eigene Datenbank oder ein eigenes Schema innerhalb einer gemeinsamen Datenbank nutzen. Das ermöglicht ein unabhängiges Aktualisieren der Datenstrukturen.
    - **Technologie:** Es sollte die für den Anwendungsfall am besten geeignete Technologie für Implementierung, Betrieb und Datenhaltung eingesetzt werden. Wenn sich diese später für den angedachten Anwendungsfall als nicht geeignet herausstellt, kann der einzelne Service ausgetauscht werden.
- **Verteilt:** Microservices kommunizieren untereinander über Netzwerkaufrufe. Jeder Service sollte eine API anbieten. Für den Nachrichtenaustausch können mehrere Techniken verwendet werden:
    - Request/Response: REST über HTTP
    - Event-basiert (Publish/Subscribe): Messaging über AMQP, MQTT

Um die Vorteile der Microservice-Architektur beschreiben zu können, sollen diese nachfolgend von monolithischen System abgegrenzt werden. Im anschließenden Kapitel wird auf unterstützende Techniken im Kontext von Microservice-Architekturen eingegangen.

3.6.1 Monolithische Systeme vs. Microservices

Bei Microservices-Architekturen wird anstatt einer großen Applikation, dem sogenannten Monolithen, das Softwaresystem in kleinere Applikationen aufgeteilt. Ein Monolith ist eine Applikation die aufgrund interner Abhängigkeiten schwer wartbar ist (Warner, 1982).

Um die Vorteile der Microservice-Architektur beschreiben zu können, soll vorab der Aufbau von monolithischen System erläutert werden.

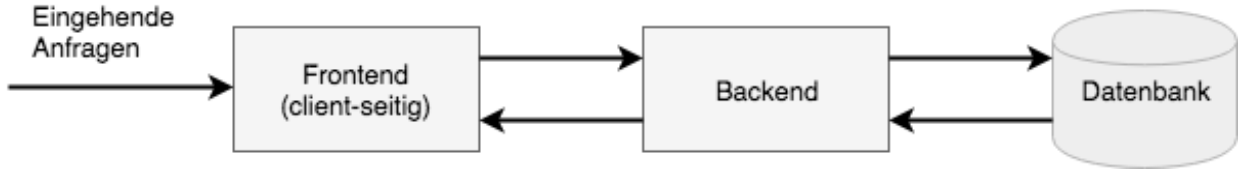

Abbildung 3-8: Drei-Schichten-Architektur [Eigene Darstellung in Anlehnung an (Fowler S. J., 2016, S. 2)]

Geschäftsanwendungen sind heute überwiegend wie in Abbildung 3-8 aufgebaut: Ein Frontend/Client in Form eines Browsers, einer Windows Applikation oder einer App. Ein Backend/Server, der die Geschäftslogik enthält und mit einer Datenbank kommuniziert als auch die Datenbank selbst (Fowler S. J., 2016). Nach Fowler und Lewis (2015) ist dies der natürliche Weg Geschäftsanwendungen zu erstellen. Das System kann oftmals auf einem Entwicklerrechner gestartet, getestet und auch in Produktion gebracht werden. Wenn die Last in Produktion ansteigt kann das System durch Vorschalten eines Load-Balancers und Hinzufügen von weiteren Instanzen der Applikation horizontal skaliert werden.

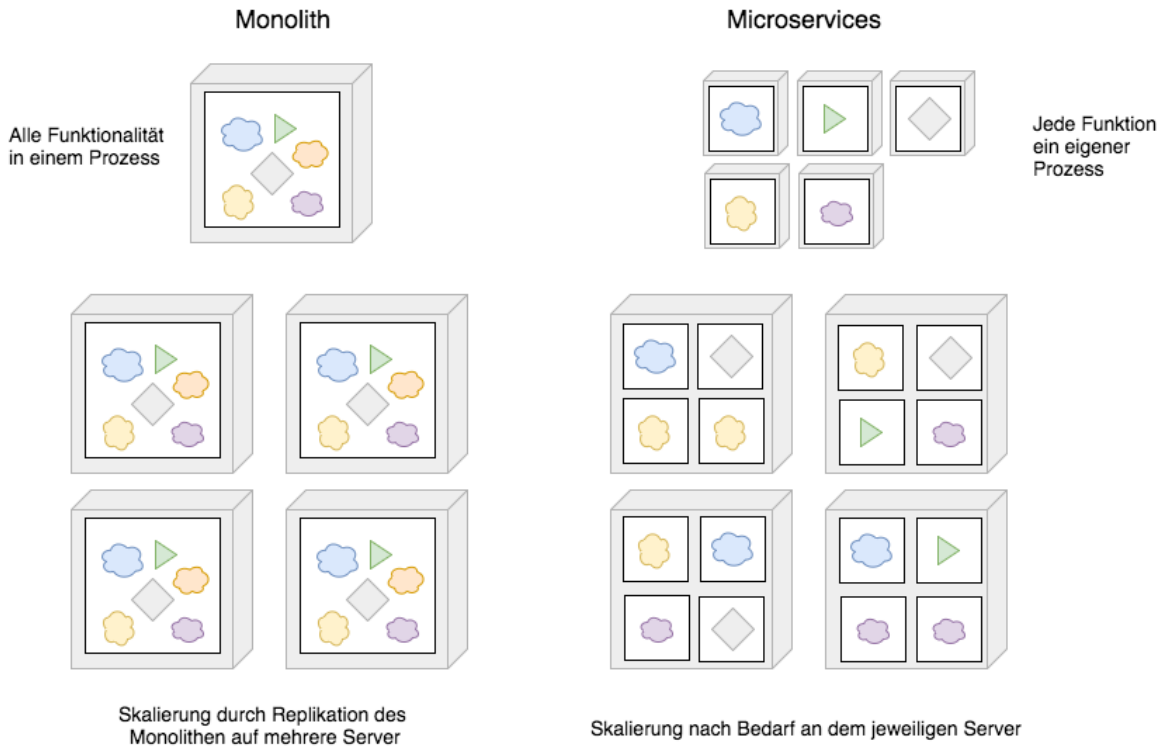

Abbildung 3-9: Monolith & Microservices [Eigene Darstellung in Anlehnung an (Fowler & Lewis, 2015)]

Je mehr Anforderungen an ein System gestellt werden, das System sich daraufhin entwickelt und verändert wird, desto mehr Zeit nimmt das Ändern, Bauen und Produktivsetzen des Systems in Anspruch. Folwer und Lewis (2015) merken an, dass es über Zeit schwierig ist eine gute modulare Struktur zu halten, so dass Änderungen immer Auswirkungen auf mehrere Module haben. Ebenso wird beim Skalieren, wie in Abbildung 3-9 dargestellt, die gesamte Anwendung skaliert anstatt einzelne Funktionen, was zu einem hohen Ressourcenverbrauch führt. Diese Probleme zeigten sich auch bei großen Online-Unternehmen wie Soundcloud, Autoscout24.de, Zalando und Otto.de, welche daraufhin Projekte starteten, um ihre monolithischen Systeme zu Microservices zu migrieren (Calçado, 2014; Hohenadl, 2015; Persa, 2015; Steinacker, 2015).

Alternativ zum radikalen Ansatz des Ersatz von monolithischen Systemen können monolitische Systeme auch inkrementell abgelöst werden. In diesem, auch unter dem Namen Strangler Pattern bekannten Verfahren, werden das monolithische System und Microservices parallel betrieben und Funktionen des monolithen schrittweise durch Microservices ersetzt, bis der Monolith außer Betrieb genommen werden kann (Fowler M. , 2004; Rook, 2016).

3.6.2 Microservices und Conways´s Law

Der Begriff Conway´s Law wurde 1995 von Frederick P. Brooks, Jr. geprägt als dieser eine Veröffentlichung von Melvin E. Conway zitierte (Brooks, 1995). Das Gesetzt von Conway lautet wie folgt:

„[...] organizations which design systems (in the broad sense used here) are constrained to produce designs which are copies of the communication structures of these organizations.“ (Conway, 1968, S. 31)

Systeme und im Kontext von Microservices besonders Softwarearchitekturen sind demnach Abbilder der Kommunikationsstrukturen der Unternehmen in denen sie entworfen werden.

Das Management in einem Unternehmen orientiert sich bei der Aufteilung von Geschäftsanwendungen nach Fowler und Lewis (2015) oft an der Technologie, was zu spezialisierten Teams und Koordination unter diesen führt (vgl. Abbildung 3-12).

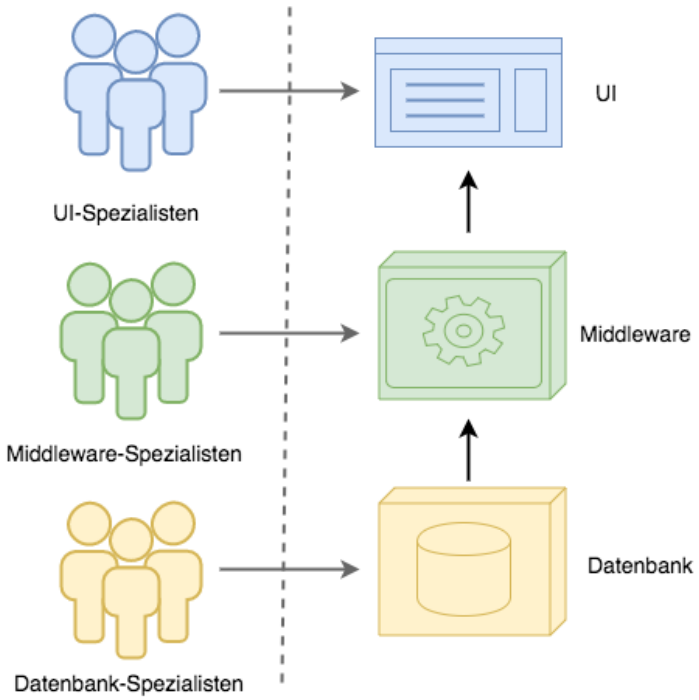

Abbildung 3-10: Conway´s Law in Aktion [Eigene Darstellung in Anlehnung an (Fowler & Lewis, 2015)]

In Microservice-Architekturen macht man sich das Conway´s Law zu Nutze und teilt Microservices nach Business Capabilities auf. Ein Team ist cross-funktional und besteht aus allen Rollen und Fähigkeiten, die benötigt werden um den Microservice zu entwickeln und zu betreiben (vgl. Abbildung 3-11). Dieses Vorgehen wird auch als Inverse Conway´s Law Maneuver bezeichnet (Fowler S. J., 2016; ThoughtWorks, Inc., 2015).

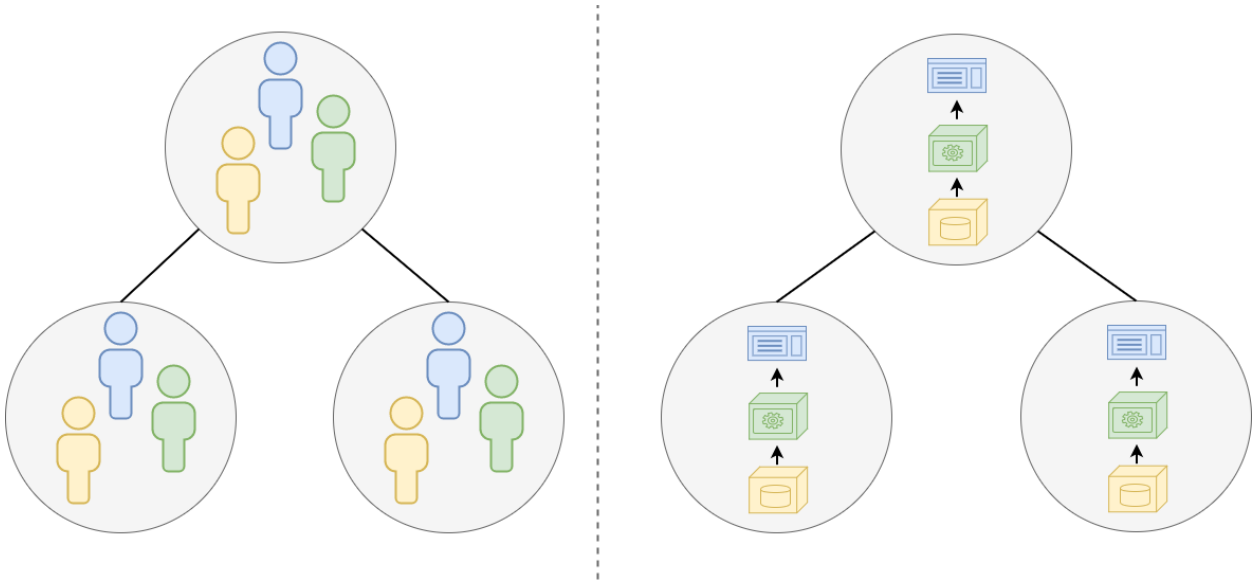

Abbildung 3-11: Inverse Conway´s Law [Eigene Darstellung in Anlehnung an (Fowler & Lewis, 2015)]

3.6.3 Unterstützende Techniken für Microservice-Architekturen

Microservices sind, wie eingangs beschrieben, ein Konzept für die Modularisierung von Softwaresystemen. Prozesse oder Vorgehensmodelle für die Migration, Einführung und den Betrieb sieht der Ansatz nicht vor. Hier kommen unterstützende Techniken zum Einsatz, die in diesem Kapitel beschrieben werden sollen.

Bei der fachlichen Aufteilung von monolithischen Systemen in kleine Microservices kann Domain Driven Design (DDD) von Eric Evans (Evans, 2003) helfen die Größen und damit die Grenzen der Microservices festzulegen. Die fachliche Aufteilung von monolithischen Systemen in kleine Microservices ist der erste Schritt bei der Migration oder Einführung von Microservices. Die Aufteilung bringt aber auch Probleme mit sich: Anstatt einem System müssen bei einem Release nun mehrere unterschiedliche Applikationen ausgeliefert werden (Deployment), im Gegensatz zu nur einer Applikation bei einem monolithischen System. Bass, Weber und Zhu betonen die Wichtig-keit des Deployment für Unternehmen:

> *„Deploying an upgrade correctly is a significant and imporant activity for an organization and, yet, one that should be done in timely fashion and minimal opperunitiy for error.“*
>
> *(Bass, Weber, & Zhu, 2015, S. 9)*

Die von Bass, Weber und Zhu genannten Fehler sollen durch Tests der Microservices in Testumgebung aufgedeckt werden. Nach erfolgreichem Test können die Microservices in Produktion gebracht werden. Klassisch werden Releases von Operations-Abteilungen durchgeführt und überwacht. Die Testumgebung stellt einige Anforderungen: Nur berechtigte Personen sollen Zugriff auf das System haben. Die Kompatibilität mit anderen Systemen, mit denen das System kommuniziert, muss gewährleistet sein. Es muss ausreichend Rechenleistung für den Testbetrieb vorhanden sein. Die Daten des Systems müssen aktuell und für andere Systeme verfügbar sein. Keine der Aktivitäten passieren aus Zufall und für alle ist Koordination zwischen den Entwicklern und dem Betrieb notwendig. Auch wenn Unternehmen dafür Vorgehensmodelle entwickelt haben, können trotzdem Fehler durch fehlende Koordination und Kommunikation auftreten (Bass, Weber, & Zhu, 2015).

An diesem Punkt setzt DevOps an: DevOps sind nach Bass, Weber und Zhu eine Reihe von Praktiken und Methoden mit der Absicht die Zeit zwischen dem Commit einer Änderung an einem System und dem Zeitpunkt bis diese Änderung in Produktion gebracht wurde zu reduzieren, bei gleichzeitiger Sicherstellung einer hohen Qualität. Der Begriff DevOps ist eine Kombination der ersten drei Buchstaben der Begriffe Development und Operations und wurde erstmalig 2009 von Patrick Debois erwähnt (Paul, 2014).

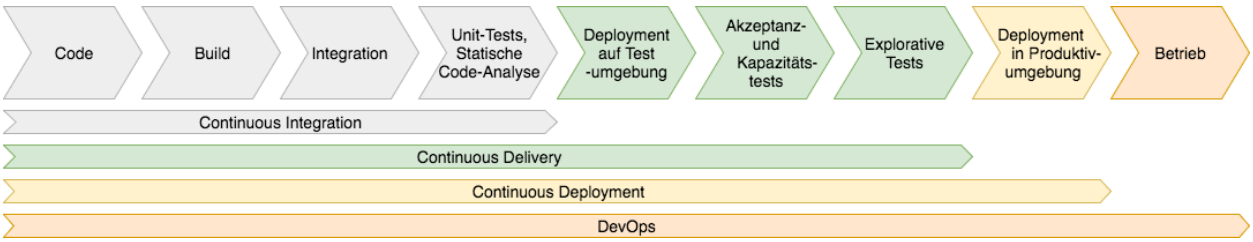

Abbildung 3-12: Zusammenhang von DevOps-Praktiken [Eigene Darstellung in Anlehnung an (Shawn, 2016)]

DevOps vereint die Praktiken Continuous Integration, Continuous Delivery und Continuous Deployment. Abbildung 3-12 stellt die Zusammenhänge der einzelnen Praktiken grafisch dar. Im Einzelnen lassen sich die Praktiken wie folgt beschreiben:

- **Continuous Integration:** Bei jedem Commit eines Entwicklers wird die Applikation gebaut. Anschließend werden Unit-Tests ausgeführt und statische Code-Analysen vorgenommen.
- **Continuous Delivery:** Die Applikation wird auf einer Testumgebung ausgeliefert, in der dann automatisierte Akzeptanz- und Kapazitätstests durchgeführt werden. Danach kann die Applikation manuell durch Domänenexperten getestet werden, falls dies notwendig ist.
- **Continuous Deployment:** Die auslieferungsfähige Applikation wird automatisiert in der Produktivumgebung in Betrieb genommen. Dazu lassen sich die Installation und Konfiguration der Applikation zählen.

Neben den beschriebenen Aktivitäten gehört zu DevOps auch das Monitoring der Applikation im Betrieb. Zu den DevOps-Praktiken kann weiterhin auch der **Infrastructure as a Code (IaaC)**-Ansatz gezählt werden. Bei diesem Ansatz werden Infrastrukturkomponenten wie beispielsweise Load-Balancer, Web-, Applikations- und Datenbankserver wie Software behandelt und automatisiert installiert, konfiguriert und getestet bereitgestellt (Morris, 2016).

4 UNTERNEHMENSKULTUR UND SOFTWAREARCHITEKTUR

In Kapitel 2 wurde der Begriff Unternehmenskultur definiert und seine Bestandteile erläutert. Ebenso wurden einige Dimensionen von Unternehmenskultur beschrieben. Anschließend wurden in Kapitel 3 der Themenkomplex Softwarearchitektur, die Rolle des Softwarearchitekten und der Architekturzyklus erläutert. In diesem Kapitel sollen diese zwei Themenkomplexe nachfolgend zusammengeführt und deren mögliche Interdependenzen von beiden Blickrichtungen betrachtet werden.

Abbildung 4-1 zeigt das Modell von Cabrera, Cabrera und Brajas (2001) zur Leistungsfähigkeit (Performance) von Unternehmen. Es basiert auf der sozio-technischen Systemtheorie, in der ein Unternehmen aus zwei komplexen und miteinander verbundenen Systemen besteht: dem technischen und dem sozialen System. Die Leistungsfähigkeit des Unternehmens hängt demnach davon ab, in welchem Maße das soziale und technische System aufeinander abgestimmt sind. Das Unternehmen verbindet Technologie und Mitarbeiter. Als Technologie betrachten werden im Rahmen dieser Arbeit IT-Systeme und deren Softwarearchitektur betrachtet. Dementgegen stehen die Mitarbeiter, welche eine Unternehmenskultur besitzen (vgl. Kapitel 2.3).

Das Modell soll im Folgenden dabei unterstützen, die Kausalitäten und Interdependenzen zwischen Unternehmenskultur und Softwarearchitektur zu erklären.

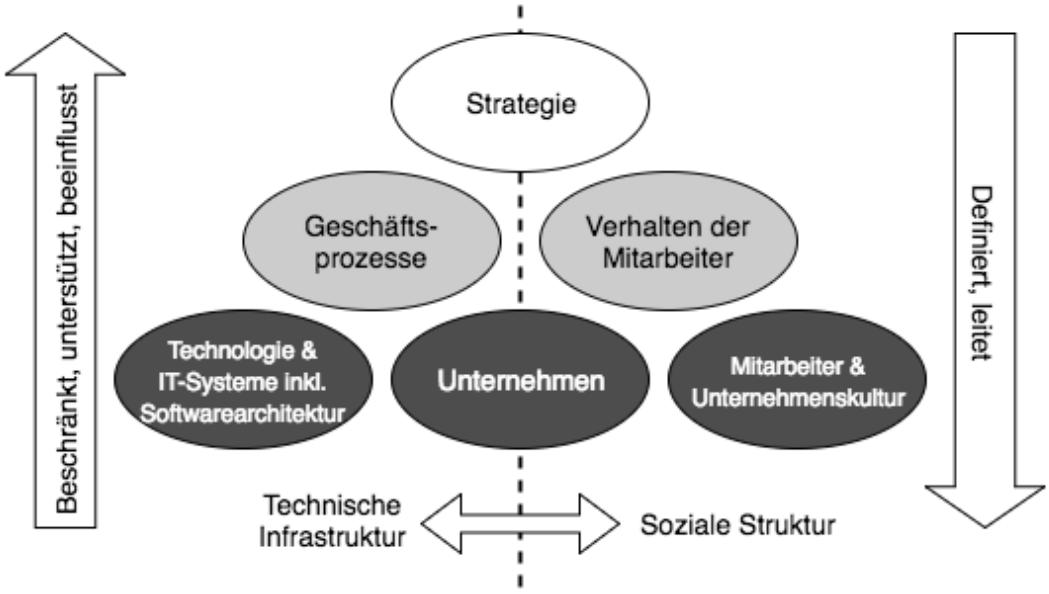

Abbildung 4-1: Modell zur Leistungsfähigkeit von Unternehmen [Eigene Darstellung in Anlehnung an (Cabrera, Cabrera, & Brajas, 2001, S. 250; Herberhold, 2016, S. 57)]

Das Modell beinhaltet drei Ebenen (Cabrera, et al., 2001, S. 250):

1. **Strategie:** Die oberste Ebene enthält die Strategie, die damit verbunden Geschäftsziele des Unternehmens und die Mission, dem Selbstbild des Unternehmens in Beziehung zu seinen Stakeholdern.
2. **Geschäftsprozesse und Verhaltensweisen:** Die mittlere Ebene bildet die Ebene der Geschäftsprozesse und Verhaltensweisen der Mitarbeter des Unternehmens. Sie setzt auf der Infrastrukturebene (unterste bzw. dritte Ebene) auf.
3. **Infrastruktur:** Die unterste Ebene beinhaltet die langlebigen Elemente des Unternehmens, bestehend aus Technologie und IT-Systemen inklusive deren Softwarearchitektur, Unternehmensstruktur, Mitarbeiter inklusive deren Kultur. Daneben enthält diese Ebene auch Managementpraktiken, die die Beziehung zwischen den Mitarbeitern und dem Unternehmen beeinflussen.

Das Erreichen der Geschäftsziele der Unternehmensstrategie hängt von den Prozessen im Unternehmen und Verhaltensweisen der Mitarbeiter ab, die selbst durch die Infrastrukturebene beeinflusst werden. Die Infrastrukturebene mit Technologie, Unternehmensstruktur und den Mitarbeitern hat somit Einfluss auf die oberen Ebenen und letztendlich auf die Fähigkeiten der Organisation und das Erreichen der strategischen Geschäftsziele. Niedrigere Ebenen entscheiden, was auf höher liegenden Ebenen möglich ist und was nicht (Cabrera et al., 2001).

Veränderungen in der Technologie haben Effekte, die sich auf andere Bereiche auswirken: Eine neue Technologie kann andere wichtige Subsysteme des Unternehmens aus dem Gleichgewicht bringen. Wenn ein Subsystem, beispielsweise eine Technologie, erfolgreich verändert werden soll, dann müssen andere Systeme das Ungleichgewicht abfangen können und sich neu ins Gleichgewicht bringen. Die Anpassung muss auf der vertikalen als auch auf der horizontalen Achse geschehen (Cabrera et al., 2001, S. 261):

- Die **vertikale Passung** ist das Gleichgewicht zwischen der Technologie, den Fähigkeit des Unternehmens und der Strategie. Es gibt keine Technologie die universell einsetzbar ist. Eine technologische Innovation wird erst einen Mehrwert für das Unternehmens bieten, wenn sie auch dazu beitragen kann, die notwendigen Geschäftsprozesse und Verhaltensweisen der Mitarbeiter im Unternehmen zu etablieren, um die Ziele des Unternehmens zu erreichen. Werte steuern das Verhalten von Mitarbeitern (vgl. Kapitel 2.4.1). Um die Verhaltensweisen zu ändern muss folglich auch die Kultur, d.h. die Werte die das Verhalten steuern, angepasst werden.
- Die **horizonale Passung** ist die Integration der technischen und sozialen Subsysteme des Unternehmens. Damit ein Unternehmen eine neue Technologie einsetzen kann, müssen die Technologie, die Unternehmensstruktur und die Mitarbeiter, bzw. deren Kultur zusammen passen oder verändert werden, so dass die Technologie richtig eingesetzt werden kann.

Nach der allgemeinen Annährung an die Interdependenzen von Technologie und Mitarbeiter im Unternehmen mit Hilfe des Modells Cabrera et al. (2001) sollen nun nachfolgend die

wechselseitigen Wirkungen von Unternehmenskultur und Softwarearchitektur aus beiden Richtungen in den folgenden zwei Kapiteln untersucht werden.

4.1 Mögliche Auswirkungen der Unternehmenskultur auf die Softwarearchitektur

Wie eingangs beschrieben muss für die Einführung und Nutzung von Technologien auch eine kompatible Kultur im Unternehmen vorherrschen – die horizontale Passung muss gewährleistet sein für die erfolgreiche Anwendung einer Technologie. Die Werte im Unternehmen schränken folglich die Anwendung einer Technologie ein. Dadurch beeinflussen die Werte und Normen auch das Verhalten der drei nachfolgenden internen Stakeholder von Softwarearchitektur im Unternehmen, die durch Ihre Aktivitäten direkt oder auch indirekt Einfluss auf die Softwarearchitektur nehmen:

- **Unternehmensführung:** Festlegung der Soll-Unternehmenskultur, der Mission, der Unternehmensstrategie und der aus der Strategie abgeleiteten Geschäftsziele
- **Softwarearchitekten:** Entwurf der Softwarearchitektur unter Berücksichtigung der Geschäftsziele und Anforderungen der Kunden
- **Softwareentwickler:** Implementierung der Softwarearchitektur

Die Wirkung der Unternehmenskultur auf die Softwarearchitektur wird in Abbildung 4-2 dargestellt. Die Unternehmensführung legt die Soll-Kultur und Strategie des Unternehmens fest. Die aus der Strategie abgeleiteten Geschäftsziele werden vom Softwarearchitekten beim Entwurf der Softwarearchitektur berücksichtigt. Die Softwareentwickler implementieren die Software unter Umsetzung der vom Softwarearchitekten entworfenen Softwarearchitektur. Die implementierte Software trägt damit zur Erreichung der Geschäftsziele bei und setzt dadurch folglich die Strategie des Unternehmens um.

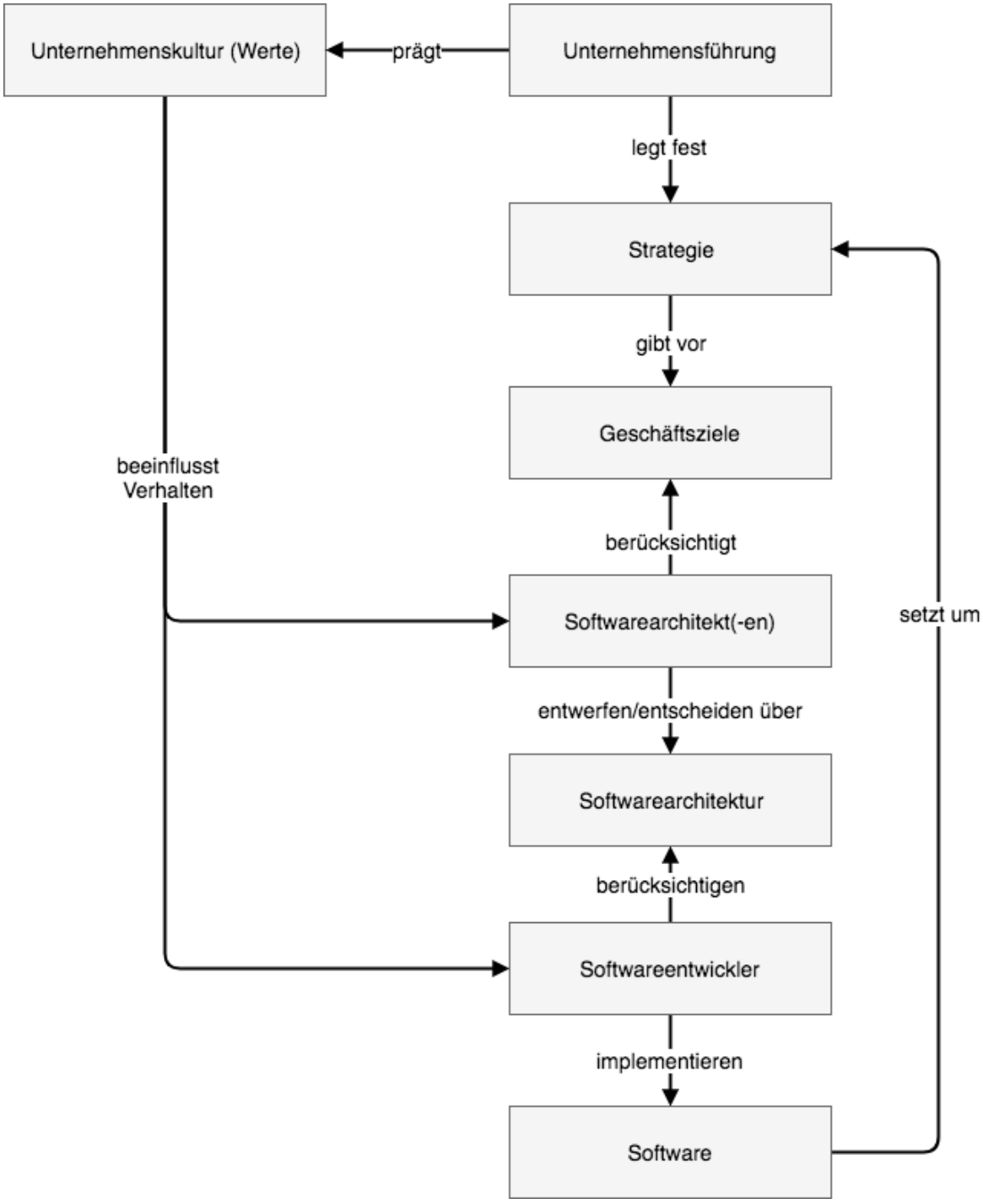

Abbildung 4-2: Wirkung der Unternehmenskultur auf die Softwarearchitektur [Eigene Darstellung]

Nachfolgend sollen die möglichen Auswirkungen der in Kapitel 2.6 beschriebenen Dimensionen von Unternehmenskultur und unter Berücksichtigung verschiedener Wertausprägungen auf den Softwarearchitekten und dadurch auch auf die Softwarearchitektur genauer erläutert und beschrieben werden. Am Ende jedes Kapitels werden die deduktiv gewonnenen Hypothesen aufgelistet.

4.1.1 Zielorientierung

Wie in Kapitel 2.6.1 beschrieben haben zielorientierte Unternehmen eine Vision und Mission, sowie eine Strategie, aus der Geschäftsziele abgeleitet sind. Aus den Geschäftszielen können sich Anforderungen an die Funktionalität des zu entwickelnden IT-Systems als auch Qualitätsanforderungen an dessen Softwarearchitektur (vgl. Kapitel 3.3.3) ableiten. Die Vision und Strategie mit den abgeleiteten Geschäftszielen unterstützen den Softwarearchitekten bei Anfertigung der Architekturvision. Die langfristigen Ziele, die mit der Architektur verfolgt werden sollen, können mit den langfristigen Zielen des Unternehmens abgeglichen werden.

Weiterhin können nicht ausreichend spezifizierte oder von Stakeholdern nicht genannte Anforderungen an das IT-System vom Softwarearchitekten durch Kenntnis der langfristigen Zielvorstellung und Strategie des Unternehmens validiert und dadurch, im Sinne der Zielerreichung, bessere Entscheidungen getroffen werden (Bass & Clements, 2011). Geschäftsziele zu erkennen und diese richtig in Softwarearchitektur umzusetzen ist wichtig, um Geschäftswerte zu erzeugen (Kazman & Bass, 2005).

Das Softwareentwicklungsteam kann sich anhand der Mission und Vision mit ihrer Arbeit, z.B. der Implementierung der Software, besser identifizieren, da sie ihren Beitrag zur Erreichung der langfristigen Ziele erkennen können. Diese Wirkung kann aber nur eintreffen, wenn Mission und Vision dem Softwareentwicklungsteam kommuniziert wurden und diese ihnen bewusst sind.

In Kapitel 2.6.1.4 wurden generische Strategien vorgestellt, die nun aufgegriffen werden sollen. Porters Strategie der Kostenführerschaft wird von Mellis (2004) für Unternehmen, die Software für den Markt entwickeln als kritisch angesehen. So müsste für einen branchenweiten Kostenvorsprung der Marktentwickler die Marktführerschaft haben, um eine Kostenführerschaft zu erreichen. Jedoch kann diese Beurteilung in heutigen Zeiten widerlegt werden, da immer mehr Software als „Software as a Service“ (SaaS) Modell angeboten und genutzt wird (Pols & Vogel, 2017). Die angebotene Software wird dann wie eine Dienstleistung durch einen Anbieter angeboten und kann von Kunden über das Internet aufgerufen werden. Ein Betreiber einer SaaS-Lösung kann, „sofern er eine große Anzahl von Kunden besitzt, Skaleneffekte und somit Kostenvorteile realisieren“ (Buxmann, Lehmann, & Hess, 2008, S. 502). Wenn der Betreiber auch gleichzeitig der Hersteller der angebotenen Software ist, dann kann dieser zusätzlich Kosten einsparen, die sonst durch Anpassung und Test hinsichtlich der Kompatibilität mit verschiedenen Betriebssystemen entstanden wären (Buxmann, Lehmann, & Hess, 2008). Ebenfalls reduziert wird die Auslieferungszeit der Anwendung bei Aktualisierungen und Änderungen. Das vorgestellte SaaS-Modell hat Auswirkungen auf die Qualitätsanforderungen an die Softwarearchitektur: Je nach Anwendungsart müssen unterschiedliche Clients berücksichtigt werden, mit denen die Kunden den Service nutzen können, um eine hohe Servicequalität sicherzustellen. Der amerikanische Video-Streaming Dienst Netflix beispielsweise kann von 800 und mehr unterschiedlichen Clients genutzt werden (Jacobson, 2012). Die Geräte variieren in ihrer Speicherkapazität, Anzahl der darzustellenden Informationen, Anbindung und Nutzerinteraktion. Das macht es notwendig, die vom Client bei den Services abgerufenen Inhalte gerätespezifisch auszuliefern.

Neben der Anforderung an Flexibilität durch unterschiedliche Clients kommen bei SaaS im Geschäftsbereich je nach Kunde unterschiedliche Anforderungen an Anpassbarkeit hinzu. Die Anforderungen variieren in der Anpassung der Funktionalität an die Geschäftsprozesse des Unternehmens. Die Gründe hierfür sind vielfältig: Konzentration auf unterschiedliche Geschäftsbereiche sowie strukturelle, prozessuale, regulatorische oder kulturelle Unterschiede. Die Anpassung an die Bedürfnisse des Kunden kann durch Konfiguration (Configuration) und Veränderung (Customization) der Software erreicht werden. Konfiguration erhöht die Komplexität des Systems. Veränderung ist nur sinnvoll, wenn eine hohe Anzahl der Kunden die gleiche Anforderung an das System stellen (Sun, Zhang, Jie Guo, Sun, & Su, 2008). Einen Ansatz für Customizing stellen Software Produktlinien dar, wie ihn Ruehl und Andelfinger (2011) präsentieren.

Auch Unternehmen die Software im Auftrag entwickeln können nach Mellis (2004) einen Kostenvorteil nur schwierig durch eine verbesserte Marktposition erreichen, da die Kosten für die Art von Entwicklungsprojekten sehr hoch sind und die Kunden „bei ihren Investitionsentscheidungen zwar eine wachsende, aber noch geringe Preissensibilität aufweisen“ (Mellis, 2004, S. 33). Auch Lang (2003) bestätigt diese Aussage und sieht die Differenzierungsstrategie als effektivere Strategie für Unternehmen aus der Software-Branche.

Nach Lang (2003) können Softwareentwicklungsvorhaben in zeit- und zuverlässigkeitsorientiert aufgeteilt werden, was der Differenzierungsstrategie nach Entwicklungszeit und nach Qualität entspricht. Bei zeitorientierten Entwicklungsvorhaben soll mit einer kurzen Entwicklungszeit ein Vorteil gegenüber den Wettbewerbern erlangt werden, weil neue und geänderte Funktionalitäten in schneller an den Kunden ausgeliefert werden als auch Anforderungen dadurch in kürzerer Zeit erfüllt werden können. Kurze Entwicklungszeiten können durch eine hohe Änderbarkeit und Wartbarkeit der der Software erreicht werden. Aber nicht nur die Änderbarkeit und Wartbarkeit der Software sind wichtig, damit daraus ein Wettbewerbsvorteil entstehen kann, denn auch die Auslieferungszeit der Software ist wichtig. Die Auslieferungszeit ist die Zeit, die benötigt wird, bis die Software installiert, konfiguriert und vom Kunden genutzt werden kann. Hier haben Anbieter von Software ein Vorteil, die auf das bereits genannte SaaS-Modell setzen, da sie nicht von externen Ressourcen, wie beispielsweise der IT-Abteilung des Kunden, angewiesen sind und somit kein zusätzlicher Koordinationsaufwand für Auslieferung, Installation und Aktualisierung entsteht. Bis ein Release nach initialer Entwicklung oder nach einer Änderung ausgeliefert werden kann, sind je nach Art der Software und Entwicklungsumgebung unterschiedliche Schritte notwendig: Die Änderungen müssen Quelltext eingearbeitet und in veränderter Version in das Quellcode-Versionsverwaltungssystem gespeichert werden. Da meist parallele Änderungen durch unterschiedliche Entwickler oder Teams an einer Software durchgeführt werden existieren mehrere Versionen des Quellcodes, die es zu integrieren gilt. Bei diesem Schritt kann es zu Komplikationen beim Zusammenführen der Änderungen kommen und die Software als Resultat nicht mehr kompiliert werden (Wolff, 2015). Mit dem in Kapitel 3.6.3 vorgestellten Continuous Integration Ansatz kann diesen Problemen durch die kontinuierliche Integration frühzeitig entgegen gewirkt werden. Im Zusammenspiel mit Continuous Deployment kann die Auslieferungszeit minimiert werden und damit ein Vorteil gegenüber dem Wettbewerb geschaffen werden.

Im Gegensatz zu zeitorientierten hat bei zuverlässigkeitsorientierten Entwicklungsvorhaben die Softwarequalität eine höhere Priorität als die Entwicklungszeit. Zu diesen Vorhaben zählen Entwicklungen von Software mit hoher Kritikalität, die sich besonders durch hohe Anforderungen an die Qualitätssicherungsmaßnahmen auszeichnen (Lang, 2003). In dieser Arbeit soll der Fokus jedoch auf zeitorientierten Projekten liegen, weshalb dieser Bereich nicht weiter ausgeführt wird.

Neben dem Entwicklungsvorhaben bestimmt die strategische Grundausrichtung des Unternehmens die Relevanz und den Stellenwert von Softwarearchitektur im Unternehmen (Bannerman, 2009). In Unternehmen mit der in Kapitel 2.6.1.4 beschriebenen Prospector-Strategie hat die Softwarearchitektur einen hohen Stellenwert, da dadurch Wettbewerbsvorteile gegenüber Mitbewerbern geschaffen werden können. Die Auswahl der Technologie, mit der die Softwarearchitektur umgesetzt wird, ist nicht auf dem Unternehmen bereits bekannten und erprobten Technologien beschränkt. Das Unternehmen ist somit frei in der Auswahl neue Technologien und Techniken einzusetzen, was dem Softwarearchitekten mehr Möglichkeiten biete, aber auch einen komplexeren Entscheidungsprozess abverlangt. Durch die schnelle Besetzung neuer Märkte und Produkte und einer hohen Innovationsorientierung können in Unternehmen mit Prospector-Charakter die Entwicklungen zu den zeitorientierten Entwicklungsvorhaben gezählt werden, so dass hier die für diese Vorhaben charakteristischen genannten Anforderungen an die Softwarearchitektur zu finden sind, die vorab genannt wurden.

Softwarearchitekten in Unternehmen mit Defender-Strategie haben die Herausforderung die Balance zwischen Festhalten an bewährten Technologien und dem schrittweisen Übergang auf neue Technologien und Architekturen zu halten. Das Festhalten an alten Technologien schränkt die Entwurfsentscheidungen des Softwarearchitekten ein. Anforderungen können nicht immer adäquat erfüllt werden, weil unter Umständen die notwendige Technologie, mit der die Softwarearchitektur umgesetzt werden soll, nicht eingesetzt werden kann.

Die Analyzer-Strategie kann sich in der Unternehmensstruktur durch getrennte Abteilungen abbilden: Eine Abteilung mit den Eigenschaften des Defenders (stabiler Betrieb und Einsatz bewährter Technologie(-n)) und eine weitere mit denen des Protectors (flexibler Betrieb und Einsatz unterschiedlicher Technologien).

Nach Bannerman (2009) hat Softwarearchitektur einen absteigenden Stellenwert in Abhängigkeit des grundlegenden Strategietyps (von am Höchsten zu am Niedrigsten): Prospector, Analyzer, Defender, Reactor.

Zusammengefasst lässt sich festhalten, dass die Strategie, die damit verbundene Art des Entwicklungsvorhabens sich auf die Qualitätsanforderungen an die Softwarearchitektur und die Entwurfsentscheidungen des Softwarearchitekten auswirken.

Für die Dimension *Zielorientierung* werden folgende acht Hypothesen aufgestellt:

- *H1.1.1.:* Wenn ein Unternehmen zielorientiert ist, dann kann der Softwarearchitekt die Architekturvision anhand der Vision, Mission und Strategie ableiten.

- *H1.1.2.:* Wenn ein Unternehmen zielorientiert ist, dann kann der Softwarearchitekt bei unklaren Anforderungen von Stakeholdern sich an der Vision, Mission und Strategie orientieren.
- *H1.1.3.:* Wenn ein Unternehmen ein SaaS-Geschäftsmodell betreibt, dann hat die Softwarearchitektur die Qualitätsmerkmale Testbarkeit und Anpassbarkeit.
- *H1.1.4.:* Wenn ein Unternehmen eine Differenzierungsstrategie nach Zeit verfolgt, dann enthält die Architektur die Qualitätsmerkmale Änderbarkeit und Testbarkeit.
- *H1.1.5.:* Wenn ein Unternehmen eine Differenzierungsstrategie nach Qualität verfolgt, dann enthält die Architektur das Qualitätsmerkmal Testbarkeit.
- *H1.1.6.:* Wenn ein Unternehmen die strategische Grundhaltung Defender verfolgt, dann ist der Softwarearchitekt in der Technologieauswahl eingeschränkt.
- *H1.1.7.:* Wenn ein Unternehmen die strategische Grundhaltung Prospector verfolgt, dann ist der Softwarearchitekt in der Technologieauswahl frei.
- *H1.1.8.:* Wenn ein Unternehmen die strategische Grundhaltung Prospector verfolgt, dann enthält die Architektur das Qualitätsmerkmal Testbarkeit.

4.1.2 Kommunikation und Information

Kommunikation hat einen hohen Anteil an den Aufgaben des Softwarearchitekten (Kruchten, 2008; Sherman & Hadar, 2015). Zunächst wird im Folgenden auf die Kommunikation in den Phasen des Architekturzyklus (vgl. Kapitel 3.4) eingegangen:

- **Analysephase**: In der Analysephase kommuniziert der Softwarearchitekt mit Stakeholdern wie dem Projektleiter, Produktmanagement, Product Owner, Requirements Engineer, Kunden und allen die Anforderungen an die Software stellen, welche relevant für den Entwurf der Softwarearchitektur sind. Der Softwarearchitekt muss die Geschäftsszenarien des Systems identifizieren und die grundlegenden architekturellen Anforderungen verstehen. Da diese nicht immer erhoben wurden und schriftlich vorliegen, muss der Architekt in den Dialog mit den genannten Stakeholdern gehen, um die notwendigen Informationen und Anforderungen zu erheben. Dabei muss er auf die unterschiedlichen Rollen und Kompetenzen des jeweiligen Stakeholders eingehen, was sich auf die Intensität und Häufigkeit der Kommunikation auswirken kann.
- **Architekturphase**: Wenn die Anforderungen geklärt sind folgt die Architekturphase, in der die Architektur ausgewählt oder erstellt und über Strukturen der zu entwickelnden Software entschieden wird. Die Entscheidungen über die Strukturen werden vom Softwarearchitekten in Form von Diagrammen oder schriftlich dokumentiert. Anschließend werden diese Dokumente an das Entwicklungsteam kommuniziert. Es zeigt sich jedoch in der Praxis, dass Architekten wenig dokumentieren (Farenhorst, Hoorn, Lago, & van Vliet, 2011).

- **Implementierungsphase**: Während der Implementierung der Software durch das Team kann es zu Problemen beim Verständnis der Softwarearchitektur und dadurch zu Rückfragen und Abstimmungen zwischen dem Team und dem Softwarearchitekten kommen.
- **Reflexionsphase**: In der Reflexionsphase stellt der Softwarearchitekt die entworfene Softwarearchitektur, d.h. die Entscheidungen bzw. Lösungen aus der Architekturphase, den Stakeholdern vor und holt Feedback dazu ein.
- **Review-Phase**: In der Review-Phase wird die vom Entwicklungsteam umgesetzte Software auf die Berücksichtigung der Architekturvorgaben geprüft. Die Abweichungen und Anmerkungen des Architekten werden dem Entwicklungsteam kommuniziert. Ebenso kann es hier zu Abstimmungen und Rückfragen zu den Design-Entscheidungen zwischen de Softwarearchitekten und den Softwareentwicklern kommen.

Neben den persönlichen Eigenschaften und sozialen Fähigkeiten des Architekten sind auch die persönlichen Werte des Architekten sowie die in der Unternehmenskultur gelebten Kommunikationswerte in den Phasen und deren Aktivitäten (vgl. Kapitel 3.4) von Bedeutung. Die Werte prägen das Verhalten des Softwarearchitekten und den Stakeholdern in ihrer Kommunikation.

Achtung, Respekt und Vertrauen sind Werte, die dazu führen, dass der Softwarearchitekt Zugang zu den Stakeholdern findet, Kontakt aufbauen und die Anforderungen abstimmen kann. Die Stakeholder müssen dem Softwarearchitekten vertrauen, damit sie die für die zu erstellende Softwarearchitektur notwendigen Informationen vom Softwarearchitekten aufgenommen und die darin enthaltenen Anforderungen analysiert werden können. Der Softwarearchitekt sollte jeden Stakeholder, unabhängig von der Position und Funktion im Unternehmen wertschätzen und anhören. Ebenso sollte Vertrauen zwischen dem Softwareentwicklungsteam und dem Softwarearchitekten bestehen, damit das Team die Lösung des Softwarearchitekten akzeptiert und auch wie geplant umsetzt. Wenn es zu Differenzen in der Implementierung der dokumentierten Softwarearchitektur kommt sollte das Softwareentwicklungsteam mit nötigem Respekt an den Softwarearchitekten herantreten und die Differenzen abstimmen. Waterman, Noble und Allan nennen (2015) die Teamkultur neben der Instabilität von Anforderungen, technischen Risiken, frühzeitiger Wertentwicklung, Agilität des Kunden und Erfahrung des Teams als Einflussfaktor der Softwarearchitektur in agilen Softwareentwicklungsprojekten. Vertrauen und Zusammenarbeit führen nach Waterman, Noble und Allan (2015) dazu, dass sich das Team weniger auf Dokumentation der Softwarearchitektur konzentrieren muss und weniger Planungsaufwand im Vorhinein notwendig ist.

Weitere Werte, die der Softwarearchitekt in der Abstimmung mit den Stakeholdern und den Softwareentwicklungsteams berücksichtigen sollte, sind **Toleranz und Rücksichtnahme**. Wenn mehr Aufwand zur Kommunikation und Abstimmung notwendig ist, weil bei den Parteien geringere Erfahrung oder Kompetenzen vorhanden sind und die vorgeschlagene Lösung nicht verstanden und daher nicht akzeptiert wird, muss der Softwarearchitekt Rücksicht auf die Stakeholder nehmen. Der Softwarearchitekt sollte dann über ein hohes Maß an **Hilfsbereitschaft** verfügen und die Parteien unterstützen.

Dabei sollte zwischen dem Softwarearchitekten und den Stakeholdern **Harmonie** herrschen. Bei auftretenden Problemen sollte eine Lösung für alle Parteien gefunden werden.

Offenheit und Mut sollten von allen Beteiligten im Softwarearchitekturzyklus berücksichtigt werden. Das beinhaltet das Zugehen auf andere sowie das Äußern eigener Meinungen und Akzeptanz fremder Meinung, wenn diese nicht mit der eigenen übereinstimmen.

Nicht jeder Stakeholder hat Zugriff auf alle Informationen, die benötigt werden, um Anforderungen formulieren zu können, weshalb **Transparenz** über Informationen wichtig für die Zusammenarbeit zwischen den Stakeholdern und dem Softwarearchitekten ist. Der Softwarearchitekt sollte allen Stakeholdern die notwendigen Informationen bereitstellen oder proaktiv liefern.

Mitarbeiter müssen zusammenarbeiten und sich austauschen, sonst ist mehr Dokumentation notwendig, um die Entwicklung zu steuern (Waterman, Noble, & Allan, 2015). Dabei unterstützt die Koordinations- und Steuerungsfunktion der Unternehmenskultur (vgl. Kapitel 2.5.5): Je identischer die Denk- und Verhaltensmuster aller Mitarbeiter sind, desto weniger andere Koordinationsmaßnahmen sind notwendig. Durch die Verhaltenssicherheit, die durch die Stabilisierungsfunktion der Unternehmenskultur (vgl. Kapitel 2.5.6) gegeben ist, muss zudem nicht jeder Vorgang neu überdacht werden.

Für die Dimension *Kommunikation und Information* werden folgende sechs Hypothesen aufgestellt:

- *H1.2.1.:* Wenn die Werte Achtung, Respekt und Vertrauen in der Kultur gelebt werden, dann kann der Softwarearchitekt einen einfachen Zugang zu den Stakeholdern der Architektur finden.
- *H1.2.2.:* Je stärker die Unternehmenskultur ist, desto weniger Koordinationsaufwand in der Analysephase des Architekturzyklus ist notwendig.
- *H1.2.3.:* Wenn die Werte Toleranz, Rücksichtnahme und Hilfsbereitschaft in der Kultur gelebt werden, dann kann der Softwarearchitekt mit unerfahrenen Stakeholdern die Anforderungen aufnehmen.
- *H1.2.4.:* Wenn der Wert Harmonie in der Kultur gelebt wird, dann kann der Softwarearchitekt bei Problemen in den Aktivitäten des Architekturzyklus die auftretenden Kommunikationsprobleme lösen.
- *H1.2.5.:* Wenn der Wert Transparenz in der Kultur gelebt wird, dann stellt der Softwarearchitekt den Stakeholdern der Architektur proaktiv alle notwendigen Informationen in der Analysephase bereit, so dass die Stakeholder die Anforderungen mit höherer Qualität nennen können.
- *H1.2.6.:* Je stärker die Unternehmenskultur ist, desto weniger Dokumentation der Architektur ist notwendig.

4.1.3 Führung und Mitarbeiterorientierung

Die Führungskräfte beeinflussen durch ihr Verhalten und die in Kapitel 2.6.3 vorgestellten Mechanismen die Werte der Mitarbeiter im Unternehmen. Es stellt sich die Frage, durch wen die Werte des Softwarearchitekten beeinflusst werden. Um diese Frage besser beantworten zu können ist es an dieser Stelle notwendig die in Kapitel 3.5.1 vorgestellten Möglichkeiten der Besetzung der Rolle des Softwarearchitekten aufzugreifen und zu berücksichtigen. Abbildung 4-3 stellt die Gedankenpfade grafisch dar. Je nach Besetzung der Rolle oder des angewandten Vorgehensmodells in der Softwareentwicklung und den daraus resultierenden Organisationsstrukturen lassen sich die möglichen Führungskräfte des oder je nach Besetzung der Rolle der Softwarearchitekten identifizieren und damit auch die mögliche Einflussquelle auf den Softwarearchitekten.

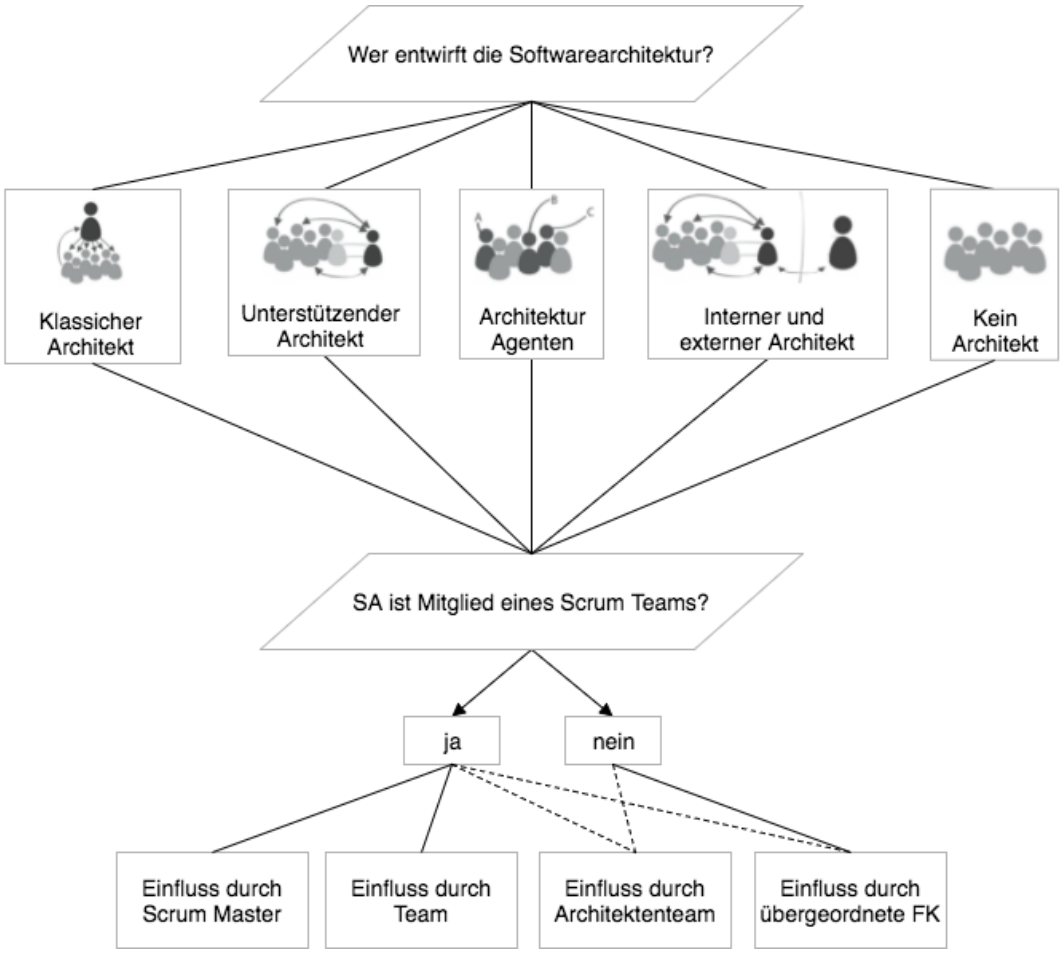

Abbildung 4-3: Führung des Softwarearchitekten [Eigene Darstellung]

Im Rahmen dieser Arbeit soll Softwarearchitektur im agilen Kontext betrachtet werden, da heute überwiegend das agile Vorgehensmodell Scrum in Unternehmen eingesetzt wird (VersionOne, 2016). Wenn der Softwarearchitekt oder die Entwickler, die Architekturaufgaben übernehmen, Mitglieder eines Scrum Teams sind, so werden diese zum einen durch den Scrum Master beeinflusst. Der Scrum Master hat in einem Scrum Team die Aufgabe, dieses zu unterstützen und

zu coachen. Insbesondere hat er die Aufgabe die Adaption von Agilität (insbesondere deren Werte und Prinzipien) und des Scrum Prozesses voranzutreiben und im Team sowie Unternehmen zu festigen (Schwaber & Sutherland, 2016). Neben dem Einfluss des Scrum Masters beeinflussen sich die Mitglieder des Teams auch gegenseitig.

Wenn der Softwarearchitekt kein Teil des Scrum Teams ist, wird er einer anderen Ebene der Unternehmenshierarchie untergeordnet sein, wie beispielsweise dem technischen Leiter des Unternehmens, der den Softwarearchitekten beeinflussen kann. Bei mehreren Softwarearchitekten werden diese zusammen ebenfalls eine Gruppe bilden und könnten sich dann ebenfalls gegenseitig beeinflussen.

In der Umsetzungsphase unterstützt der Softwarearchitekt das Entwicklungsteam bei der Implementierung durch Beantwortung offener Fragestellungen über Architekturentscheidungen. Dabei wirken die Mechanismen beschrieben in Kapitel 2.6.3.

Neben der Fragestellung, wer wen im Kontext der Softwarearchitektur führt und damit beeinflusst, stellt sich auch die Frage, wie sich konkrete Führungswerte auf den Softwarearchitekten und damit die Softwarearchitektur auswirken. In mitarbeiterorientierten Kulturen wird der Softwarearchitekt aufgrund der offenen Kommunikation mehrere Rückmeldungen von den Entwicklern erhalten und kann dadurch die Softwarearchitektur weiter verbessern.

Für die Dimension *Führung und Mitarbeiterorientierung* werden folgende sieben Hypothesen aufgestellt:

- *H1.3.1.:* Wenn der Softwarearchitekt Teil eines Scrum Teams ist, dann werden seine Werte vom Scrum Master beeinflusst.
- *H1.3.2.:* Wenn der Softwarearchitekt Teil eines Scrum Teams ist, dann werden seine Werte vom Scrum Team beeinflusst.
- *H1.3.3.:* Wenn der Softwarearchitekt nicht Teil eines Scrum Teams ist, dann werden seine Werte von der übergeordneten Führungskraft beeinflusst.
- *H1.3.4.:* Wenn der Softwarearchitekt Teil einer Gruppe von Softwarearchitekten ist, dann beeinflussen die Mitglieder der Gruppe die Werte des Softwarearchitekten.
- *H1.3.5.:* Je mitarbeiterorientierter die Unternehmenskultur, desto höher ist die Intensität der Kommunikation zwischen dem Softwarearchitekten und dem Entwicklungsteam.
- *H1.3.6.:* Je mitarbeiterorientierter die Unternehmenskultur, desto mehr Feedback erhält der Softwarearchitekt von dem Entwicklungsteam.
- *H1.3.7.:* Je mitarbeiterorientierter die Unternehmenskultur, desto höher die Qualität der Softwarearchitektur.

4.1.4 Kundenorientierung

Um auf die möglichen Auswirkungen einer kundenorientierten Unternehmenskultur auf den Softwarearchitekten und die Softwarearchitektur eingehen zu können, soll vorab geklärt wer-

den, wer in diesem Kontext den Kunden darstellt: Bei Unternehmen die Software für einen Markt herstellen ist der Kunde der Käufer der Software. Setzen Anbieter auf das Software as a Service-Modell, wie in Kapitel 4.1.1 beschrieben, so ist der Nutzer des SaaS-Angebotes der Kunde. Im Fall von SaaS ist der Kunde nicht zwangsläufig direkt identifizierbar, außer es werden personalisierte Zugänge eingesetzt. Weiterhin kann ein Kunde ein Unternehmen, ein Geschäftsbereich, eine einzelne Abteilung aber auch ein einzelner Mitarbeiter des Unternehmens sein. Der Kunde muss nicht immer von externen Unternehmen sein - bei internen Softwareentwicklungsprojekten sind die Kunden Kollegen aus anderen Abteilungen. An Softwareentwicklungsprojekten sind mehrere unterschiedliche Stakeholder auf Seite des Kunden beteiligt, mit denen sich der Softwarearchitekt im Rahmen des Architekturzyklus befassen muss. Die Abbildung 3-7 verdeutlicht dies. Auch wenn diese Personen beteiligt sind, kann es sein, dass der Softwarearchitekt nicht direkt mit dem Kunden interagiert, sondern über einen Dritten die Anforderungen und Ziele des Kunden erfasst. So kann der Kunde unter Einsatz des agilen Vorgehensmodells Scrum auch der Product Owner (PO) sein, der als Proxy-Customer fungiert, wenn der richtige Kunde nicht vor Ort, nicht erreichbar oder im Rahmen von Entwicklungen für den Markt noch nicht existiert (Paasivaara, Durasiewicz, & Lassenius, 2008). Abschließend sollen als Kunde in dieser Arbeit alle genannten Organisationen und Personen(-gruppen) als Kunden verstanden werden.

Es stellt sich die Frage wie sich eine kundenorientierte Unternehmenskultur auf die Softwarearchitektur auswirkt. Anhand des Architekturzyklus lassen sich die Phasen und deren Aktivitäten identifizieren, bei denen der Kunde involviert ist:

- **Analysephase:** Der Softwarearchitekt nimmt die Anforderungen des Kunden an das IT-System auf und stimmt diese mit dem Kunden ab.
- **Entwurfsphase:** Der Softwarearchitekt berücksichtigt alle vom Kunden genannten Anforderungen beim Entwurf der Softwarearchitektur.
- **Reflexionsphase:** Der Softwarearchitekt stellt dem Kunden die entworfene oder ausgewählte Architektur vor und beschreibt, wie die Softwarearchitektur die Anforderungen des Kunden erfüllt.

Kundenorientierung lässt sich in der Kundennähe im Interaktionsverhältnis wiederfinden: In der Analysephase kann sich eine ausgeprägte Kundenorientierung durch ausgeprägtes Requirements Engineering und intensive Kommunikation des Softwarearchitekten mit dem Kunden beim Erheben und Abstimmen der Anforderungen zeigen. Die Stärke der Kundenorientierung wird durch die Offenheit des Softwarearchitekten gegenüber dem Kunden und dessen Anregungen geprägt.

Neben der Nähe zum Kunden in der Interaktion mit diesem kann sich die Nähe auch im Leistungsangebot zeigen: Während der Entwurfsphase kann der Softwarearchitekt über die vom Kunden genannten Anforderungen heraus auch weitere Anforderungen berücksichtigen, die zu einer höheren Kundenzufriedenheit führen. Auch der Grad der Berücksichtigung der Anforderungen die vom Kunden genannt werden, wirkt sich auf die Kundenzufriedenheit aus. Das Ka-

no-Modell in der folgenden Abbildung 4-4 zeigt die Wirkung von Erfüllung und Nichterfüllung von Anforderungen auf die Kundenzufriedenheit.

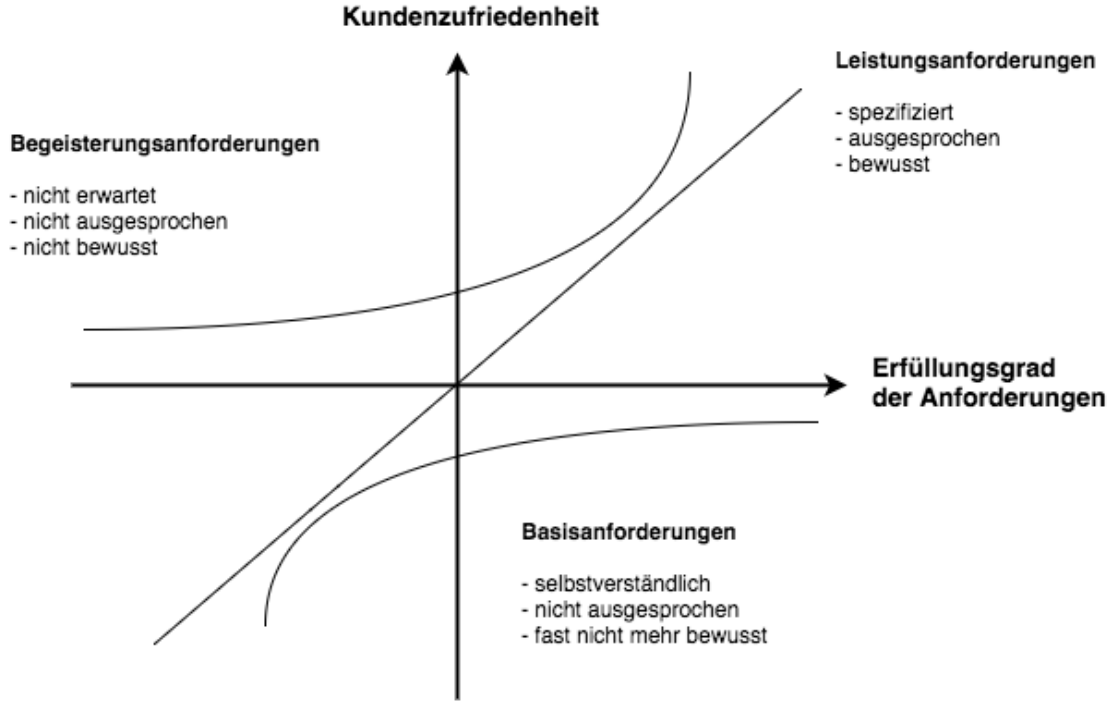

Abbildung 4-4: Kano-Modell [Eigene Darstellung in Anlehnung an (Günter & Huber, 1996, S. 250)]

Das Kano-Modell unterscheidet zwischen drei Arten von Anforderungen:

- **Basis- bzw. Grundanforderungen** sind die funktionalen Anforderungen an das IT-System und werden vom Kunden als selbstverständlich vorausgesetzt. Weil sie dem Kunden fast nicht mehr bewusst sind, werden sie meist nicht ausgesprochen. Die Erfüllung dieser Anforderungen führt nicht unbedingt zu Kundenzufriedenheit, jedoch zu Unzufriedenheit, wenn sie unerfüllt sind.
- **Leistungs- bzw. Qualitätsanforderungen** sind die von dem Kunden angesprochenen und ihm bewussten nicht-funktionalen Anforderungen an das IT-System. Je besser diese Anforderungen erfüllt sind desto höher ist die Kundenzufriedenheit.
- **Begeisterungsanforderungen** sind dem Kunden im Gegensatz zu den anderen Anforderungen nicht bewusst und werden deshalb auch nicht ausgesprochen. Ebenfalls wird ihre Umsetzung nicht erwartet, was dazu führt, dass eine Nicht-Erfüllung keine Auswirkungen hat. Wenn Begeisterungsfaktoren jedoch erfüllt sind, führt dies zu einer sehr hohen Zufriedenheit und Kunden nehmen Nicht-Erfüllen von Leistungsanforderungen unter diesen Umständen in Kauf.

In einer kundenorientierten Unternehmenskultur wird der Softwarearchitekt über die vom Kunden genannten Basis- und Leistungsanforderungen hinaus im Rahmen der Entwurfsphase auch nach Begeisterungsanforderungen suchen und danach streben diese zu berücksichtigen, um eine höhere Kundenzufriedenheit zu erreichen. Begeisterungsanforderungen, die nicht durch die Funktionalität der Software, sondern durch die Softwarearchitektur erfüllt werden könnten, wären beispielsweise eine unerwartete sehr hohe Performance beim Laden der Software oder Verarbeiten von großen Datenmengen.

Hohe Kundenzufriedenheit ist für Unternehmen ein erstrebenswerter Zustand, da diese Grundlage für langfristige Geschäftsbeziehungen ist (Zollner, 1995). Wie in Kapitel 2.6.4 beschrieben kann Kundenzufriedenheit durch eine hohe Qualität von Produkten und Service erreicht werden. Qualität ist nach der ISO 9000:2015 „der Grad, in dem ein Satz inhärenter Merkmale Anforderungen erfüllt“ (DIN EN ISO 9000:2015) oder mit anderen Worten die „realisierte Beschaffenheit einer Einheit bezüglich der Qualitätsforderung“ (Zollondz, 2011, S. 164). Das heißt das die Grund- als auch die Qualitätsanforderungen des IT-Systems erfüllt sein müssen für eine hohe Qualität, die dann zur Kundenzufriedenheit führt. Die Erfüllung der Grundanforderungen und Qualitätsanforderungen kann über verschiedene Tests sichergestellt werden, die im Testquadranten in Abbildung 4-5 im grau markierten Quadranten Q4 zu finden.

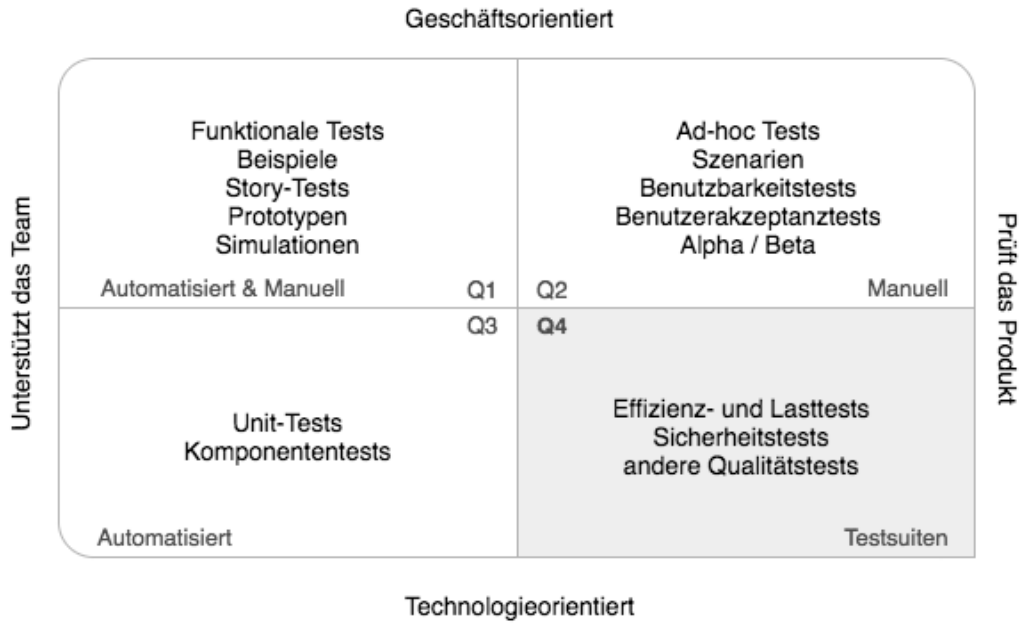

Abbildung 4-5: Agiler Testquadrant [Eigene Darstellung in Anlehnung an (Toth, 2014, S. 193)]

Die Kunden- und Innovationsorientierung eines Unternehmens ergänzen sich sehr gut. Durch Erfüllen von Begeisterungsanforderungen können Innovationen entstehen. Aber auch der Kunde kann in die Ideengenerierung für Innovationen mit integriert werden.

Für die Dimension *Kundenorientierung* werden folgende sieben Hypothesen aufgestellt:

- *H1.4.1.:* Je kundenorientierter die Unternehmenskultur, desto mehr Kontakt hat der Softwarearchitekt mit dem Kunden.

- *H1.4.2.:* Je kundenorientierter die Unternehmenskultur, desto mehr kommuniziert der Softwarearchitekt mit dem Kunden.
- *H1.4.3.:* Je kundenorientierter die Unternehmenskultur, desto offener die Kommunikation des Softwarearchitekten mit dem Kunden.
- *H1.4.4.:* Je kundenorientierter die Unternehmenskultur, desto höher die Qualität der Architektur.
- *H1.4.5.:* Je kundenorientierter die Unternehmenskultur, desto mehr Begeisterungsanforderungen erhebt der Softwarearchitekt in der Analysephase des Architekturzyklus.
- *H1.4.6.:* Je kundenorientierter die Unternehmenskultur, desto höher ist die Testbarkeit der Softwarearchitektur.
- *H1.4.7.:* Je kundenorientierter die Unternehmenskultur, desto mehr Tests der Softwarearchitektur werden erstellt.

4.1.5 Innovationsorientierung

Innovationsorientierung zeigt sich durch eine positive Einstellung gegenüber Innovationen als auch innovationsfreudige Handlungen bzw. ein innovationsorientiertes Verhalten der Mitarbeiter und Führungskräfte in einem Unternehmen. Nachfolgend soll betrachtet werden wie sich innovationsfreundliche Handlungen und innovationsorientiertes Handeln im Unternehmen auf die Softwarearchitektur auswirken können.

Um eine Innovation zum Erfolg zu bringen, braucht es erst eine oder mehrere Ideen. Diese entstehen nicht einfach, sondern sie sind Ergebnis eines Kreativitätsprozesses und müssen generiert werden. Ideen können über die Verbesserung von bestehenden oder Entwurf von neuartigen Produkten oder Prozessen handeln. Der Softwarearchitekt kann während der Analysephase des Architekturzyklus in der Kommunikation mit den Stakeholdern der Architektur, beim Erfassen und Abstimmen der Anforderungen an die Architektur Ideen generieren. Die Ideen dabei könnten sich um die Optimierung der Architektur durch Hinzufügen, Verändern oder Eliminieren von Anforderungen handeln.

Auch in der Reflexionsphase bei der Vorstellung der Architektur an die Stakeholder bietet sich in Workshops die Gelegenheit, Ideen zur Optimierung der Entwurfsentscheidungen oder des Architekturprozesses zu generieren.

Später im Architekturzyklus bei Validierung der Implementierung in der Review-Phase kann der Softwarearchitekt zusammen mit dem Entwicklungsteam ebenfalls Ideen generieren.

Sind erst einmal Ideen entstanden gilt es diese auszuprobieren und in die Tat umzusetzen. Hierfür ist je nach Intensität der Idee zum einen Risikofreude und unter Umständen auch die Unterstützung des Managements oder der Kollegen notwendig. Zumindest sollte Offenheit und Vertrauen bestehen und eine Fehlertoleranz, denn nicht jede Innovation ist erfolgreich.

Um den Erfolg der Innovation bestimmen zu können muss die Innovation validiert und getestet werden. Die Auswirkungen der Änderungen an der Architektur können mit den in Kapitel 4.1.4

vorgestellten Testmethoden erfasst werden. Dies gilt auch für die Veränderungen des Architekturprozesses, deren Auswirkungen sich im Architekturentwurf oder in der Architekturbewertung und letzten Endes in der Implementierung der Softwarearchitektur zeigen.

Innovationen fordern Flexibilität in der Softwarearchitektur, die sich in den Qualitätsanforderungen Portier- und Wartbarkeit widerspiegeln: Anpassbarkeit, Austauschbarkeit, Koexistenz, Analysierbarkeit, Änderbarkeit, Testbarkeit. Jedoch dürfen unter der Flexibilität die anderen Qualitäten Funktionalität, Zuverlässigkeit, Benutzerbarkeit und Effizienz nicht leiden. Dies gilt es durch zusätzliches Testen, Messen und Analysieren des IT-Systems über A/B-Tests, Kundenbefragungen oder Analyse des Benutzerverhaltens zu ermitteln und entsprechende Gegenmaßnahmen einzuleiten.

Für die Dimension *Innovationsorientierung* werden folgende sieben Hypothesen aufgestellt:

- *H1.5.1.:* Je innovationsorientierter die Unternehmenskultur, desto mehr Ideen für Innovationen werden während den Aktivitäten im Architekturzyklus generiert.
- *H1.5.2.:* Je innovationsorientierter die Unternehmenskultur, desto Unterstützung erhält der Softwarearchitekt bei Einsatz einer neuen Technologie oder Architektur von weiteren Mitarbeitern.
- *H1.5.2.:* Je innovationsorientierter die Unternehmenskultur, desto Unterstützung erhält der Softwarearchitekt bei Einsatz einer neuen Technologie oder Architektur vom Management des Unternehmens.
- *H1.5.3.:* Je innovationsorientierter die Unternehmenskultur, desto offener ist der Softwarearchitekt gegenüber neuen Technologien in der Entwurfsphase.
- *H1.5.4.:* Je innovationsorientierter die Unternehmenskultur, desto mehr Vertrauen wird dem Softwarearchitekt bei Einsatz neuer Technologien geschenkt.
- *H1.5.5.:* Je innovationsorientierter die Unternehmenskultur, desto höher die Toleranz gegenüber Fehlern durch Fehlentscheidung des Softwarearchitekten in der Entwurfsphase.
- *H1.5.6.:* Je innovationsorientierter die Unternehmenskultur, desto besser werden die Qualitätsanforderungen Portier- und Wartbarkeit erfüllt.
- *H1.5.7.:* Je innovationsorientierter die Unternehmenskultur, desto mehr Testverfahren werden eingesetzt, um Feedback zur Architektur zu erhalten.

4.2 Mögliche Wirkungen der Softwarearchitektur auf die Unternehmenskultur

Nach der in Kapitel 3.1 eingeführten Definition wird für diese Arbeit unter Softwarearchitektur die Menge an wichtigen Entscheidungen über die Organisation und die Struktur eines Softwaresystems verstanden. Kapitel 3.3 hat aufgezeigt, welche Einflussfaktoren den Entwurf einer Softwarearchitektur beeinflussen. Aufgrund der hohen Anzahl an Einflussfaktoren und den dar-

aus möglichen Kombinationen soll im Folgenden der mögliche Einfluss von Entscheidungen beim Entwurf einer Softwarearchitektur auf die Unternehmenskultur am Beispiel von Microservice-Architekturstils und unter Festlegung von Annahmen beschrieben werden. Daraus leitet sich die folgende Frage ab, die in diesem Kapitel beantwortet werden soll: Wie beeinflussen die Eigenschaften, Strukturen und begleitenden Techniken von Microservice-Architekten die Kultur, also die Werte und Grundannahmen der Mitarbeiter und Führungskräfte in einem Unternehmen?

Der Einfluss von Technologie auf die Kultur eines Unternehmens lässt sich aus zwei Perspektiven betrachten:

- Aus der **technikdeterministische Sichtweise** verändert der Einsatz der Technologie die Unternehmenskultur. Historisch gewachsene Unternehmenskulturen können durch die Veränderung einer Technologie in Frage gestellt werden und tief verwurzelte Annahmen und Werte müssen angepasst werden (Schein, 1995).
- In der **kulturdeterministischen Sichtweise** verändert sich die Kultur aufgrund von veränderten Rahmenbedingungen, die durch die Technologie geschaffen wurden (Goffart, 2016).

In den nachfolgenden Betrachtungen sollen diese Sichtweisen kombiniert werden. Beiden Ansichten ist gemein, dass die Veränderung einer Technologie oder Einführung einer neuen Technologie die horizontale Passung im eingangs eingeführten Modell der Leistungsfähigkeit von Unternehmen aus dem Gleichgewicht bringt, wenn die Unternehmensstruktur oder die Unternehmenskultur nicht mit der Technologie kompatibel sind. Dieses Ungleichgewicht muss wiederhergestellt werden, damit das Unternehmen weiterhin leistungsfähig bleibt.

Zur Beantwortung der Frage wird für weitere Gedankengänge eine Annahme getroffen: Eine Microservice-Architektur wurde bereits im Unternehmen eingeführt und ein Softwaresystem erfolgreich mit der Architektur umgesetzt oder ein monolithisches System inkrementell unter Anwendung des Strangler Patterns abgelöst. Erfolgreich heißt in diesem Kontext, dass die Architektur den Anforderungen, insbesondere den gestellten Qualitätsanforderungen aller ihrer Stakeholder, erfüllt. Mit der Einführung von neuen IT-Technologien geht oft Widerstand der Mitarbeiter einher und geplante Veränderungen stoßen auf Ablehnung der Belegschaft. Das Management von tiefgreifenden, geplanten Veränderungen und die Sicherstellung deren Akzeptanz wird als *Change Management* bezeichnet (Chies, 2016). An dieser Stelle soll hier nur auf die weiterführende einschlägige Literatur wie z.B. Doppler und Lauterburg (2014), Chies (2016) und Von Hehn (2016) verwiesen werden, anstatt Change Management im Einzelnen zu erläutern. Ferner soll davon ausgegangen werden, dass Change Management erfolgreich betrieben wurde und der Microservice-Architekturstil von allen Beteiligten akzeptiert und getragen wird.

Nachfolgend soll der Einfluss der Microservice-Architektur auf die in Kapitel 2.6 vorgestellten Dimensionen von Unternehmenskultur beschrieben werden. Am Ende jedes Kapitels werden die deduktiv gewonnenen Hypothesen, wie Microservice-Architekturen auf die Unternehmenskultur wirken, aufgelistet.

4.2.1 Zielorientierung

In Kapitel 2.6.1.4 wurden generische Strategien vorgestellt, die nun im Kontext von Microservice-Architekturen beschrieben werden sollen. Die Kostenführerschaft durch Einsatz des SaaS-Modells wird durch Microservice-Architekturen begünstigt. Softwaresysteme mit Microservice-Architekturen können aufgrund des Modularisierungskonzepts der Architektur, also durch die Aufteilung der Applikation in einzelne Services, einzeln getestet werden im Gegensatz zu monolithischen Systemen, was zu weniger Testaufwand bei Änderung der Software führt und damit auch zu weniger Kosten, was die Strategie der Kostenführerschaft begünstigt. Der Vorteil der verbesserten Testbarkeit macht Microservice-Kulturen auch für Unternehmen interessant sich durch die Qualität Ihrer Produkte von anderen Unternehmen differenzieren. Auf die Vorteile von Microservice-Architekturen bei Differenzierung nach Service wird nachfolgend in Kapitel 4.2.4 im Kontext der Kundenorientierung eingegangen.

Microservice-Architekturen eignen sich für Unternehmen mit strategischer Grundausrichtung mit Prospektor-Charakter. Durch das Modularisierungskonzept kann jeder Microservice mit den Technologien umgesetzt werden, welche die Anforderungen, die an den Service gestellt werden, am besten erfüllen. Ebenso fördern Microservice-Architekturen Innovationen, was nachfolgend in Kapitel 4.2.5 genauer beleuchtet werden soll.

Zusammengefasst lässt sich festhalten, dass Microservice-Architekturen Differenzierungsstrategien begünstigen.

Für die Dimension *Zielorientierung* werden folgende fünf Hypothesen aufgestellt:

- *H2.1.1.:* Wenn ein Unternehmen ein System mit Microservice-Architektur anstatt einem monolithischem System einsetzt, dann sind die Kosten für Tests geringer.
- *H2.1.2.:* Wenn ein Unternehmen ein System mit Microservice-Architektur einsetzt, dann kann es die Strategie der Kostenführerschaft umsetzen.
- *H2.1.3.:* Wenn ein Unternehmen ein System mit Microservice-Architektur einsetzt, dann kann es die Differenzierungsstrategie nach Qualität umsetzen.
- *H2.1.4.:* Wenn ein Unternehmen ein System mit Microservice-Architektur einsetzt, dann kann es die strategische Grundhaltung Prospector annehmen.
- *H2.1.5.:* Wenn ein Unternehmen ein System mit Microservice-Architektur einsetzt, dann kann es die strategische Grundhaltung Defender annehmen.

4.2.2 Kommunikation und Information

Die Aufteilung der Services in Microservice-Architekturen wird nach Business Capabilities vorgenommen. Zusammenhängende Services sollten von einem Team entwickelt und betreut werden. Hierdurch verringert sich der Kommunikationsaufwand, da Entwickler sich nur noch innerhalb ihres Teams koordinieren müssen. Durch die begleitende Technik DevOps wird Kommunikation mit zentralen Abteilungen wie z.B. Betrieb und Datenbank-Administration unnötig.

Die Aufteilung nach Business Capabilities hat den Vorteil, dass klare Verantwortungsbereiche der Teams definiert sind und diese auch transparent für alle Teams sind. Dadurch kann ein Team gezielt angesprochen werden.

Services bieten Schnittstellen, die durch andere Services genutzt werden können. Zur korrekten Nutzung müssen die Services und ihre Schnittstellen bekannt sein. Die Services und ihre Schnittstellen müssen dokumentiert und die Dokumentation muss für Service-Nutzer verfügbar gemacht werden. Microservice-Architekturen stellen somit den Anspruch an offene und transparente Kommunikation und Information innerhalb des Unternehmens.

Für die Dimension *Kommunikation und Information* werden folgende vier Hypothesen aufgestellt:

- *H2.2.1.:* Wenn ein Unternehmen ein System mit Microservice-Architektur und DevOps einsetzt, dann verringert sich die interne Kommunikation im Unternehmen.
- *H2.2.2.:* Wenn ein Unternehmen ein System mit Microservice-Architektur einsetzt und die Microservices nach Business Capabilities strukturiert hat, dann ist transparenter, welches Team für welchen Microservice kontaktiert werden muss
- *H2.2.3.:* Wenn ein Unternehmen ein System mit Microservice-Architektur einsetzt und die Microservices nach Business Capabilities strukturiert hat, dann ist eine bessere, d.h. gezieltere, Kommunikation im Unternehmen möglich.
- *H2.2.4.:* Wenn ein Unternehmen ein System mit Microservice-Architektur einsetzt und die Microservices nach Business Capabilities strukturiert hat, dann wird im Unternehmen offener kommuniziert und Wissen geteilt.

4.2.3 Führung und Mitarbeiterorientierung

Teams die Microservice-Architekturen umsetzen sollten mit einem Führungsstil, der sich durch Mitarbeiter- und Ergebnisorientierung auszeichnet, geführt werden. Dies lässt sich wie folgt begründen: Aufgabenorientierung schränkt die Möglichkeit mit Technologien und Techniken zu experimentieren ein, was sich negativ auf die Innovationskraft auswirken kann.

Microservices werden nach nach Business Capabilities aufgeteilt und dann von jeweils einem Team entwickelt und betrieben. Applikationen sind dadurch kleiner, haben einen abgegrenzten Kontext und sind weniger Komplex als eine monolithische Anwendung. Neue Mitarbeiter können dadurch einfacher ins Team ins Team integriert werden und schneller produktiv sein. Lean IX, Anbieter einer SaaS-Lösung für Architekturmanagement, bestätigt den effizienten Start von neuen Mitarbeitern und gibt eine Zeit von drei Stunden an, bis der neue Mitarbeiter seinen erste Änderung an einem Microservice in die Quellcodeverwaltung lädt (Christ, 2017).

Mitarbeiter können sich durch die DevOps-Praktiken, insbesondere durch die Verantwortung des Betriebs der Microservices, besser mit Ihrer Arbeit identifizieren, was zu höherer Motivation führt.

Microservice-Architekturen machen Unternehmen zudem für Bewerber interessanter. Entwickler möchten mit aktuellen Technologien arbeiten (Gebhart, 2016; Heusingfeld, 2017). Das Modularisierungskonzept von Microservice-Architekturen unterstützt den parallelen Einsatz von unterschiedlichen Technologien. Diese Technologien können durch das Team festgelegt werden und erfahren damit eine höhere Akzeptanz als von außen vorgegebene Technologien. Dies wiederum führt zu höherer Identifikation und Motivation der Mitarbeiter.

Für die Dimension *Führung und Mitarbeiterorientierung* werden folgende sieben Hypothesen aufgestellt:

- *H2.3.1.:* Wenn ein Unternehmen ein System mit Microservice-Architektur einsetzt und die Teams ergebnisoffen geführt werden, dann wird von den Teams mehr mit neuen Technologien und Techniken experimentiert.
- *H2.3.2.:* Je ergebnisorientiert Teams in einem Unternehmen, welches ein System mit Microservice-Architektur einsetzt, geführt werden, desto höher ist die Innovationskraft des Unternehmens.
- *H2.3.3.:* Wenn ein Unternehmen ein System mit Microservice-Architektur einsetzt und die Teams aufgabenorientiert geführt werden, dann wird von den Teams mehr mit neuen Technologien experimentiert.
- *H2.3.4.:* Wenn ein Unternehmen ein System mit Microservice-Architektur einsetzt, dann können neue Mitarbeiter schneller ins Team integriert werden und produktiv sein.
- *H2.3.5.:* Wenn ein Unternehmen ein System mit Microservice-Architektur und DevOps einsetzt, dann identifizieren sich die Mitarbeiter mit denen von Ihnen entwickelten und betriebenen Microservices.
- *H2.3.6.:* Wenn ein Unternehmen ein System mit Microservice-Architektur einsetzt, dann bekommt es mehr Bewerbungsanfragen auf offene Stellenanzeigen.
- *H2.3.7.:* Wenn ein Unternehmen ein System mit Microservice-Architektur einsetzt und die Teams die von ihnen zu verwendenden Technologien selbst auswählen können, steigt die Motivation der Mitarbeiter.

4.2.4 Kundenorientierung

Der Modularisierungsansatz in Microservice-Architekturen begünstigt die Kundenorientierung eines Unternehmens. Die Anforderungen von Kunden können in den einzelnen Services integriert und unabhängig voneinander ausgeliefert werden. Durch die Modularisierung wird zudem eine verbesserte Testbarkeit erreicht, weil die einzelnen Applikationen unabhängig voneinander getestet werden können. Die unterstützende Technik DevOps sorgt für die notwendige Qualität durch hohe Automatisierung der Tests und Auslieferung. Durch das Zusammenführen von Entwicklung und Betrieb bekommt das Team direkte Rückmeldung über die Software und die geleistete Arbeit. Abweichungen können vom Team schneller korrigiert werden, was sich positiv auf die Servicequalität auswirkt.

Ebenso bietet der Modularisierungsansatz von Microservices die Möglichkeit, für jeden Kunden eine auf die Bedürfnisse des Kunden angepasste Version eines Microservices zu betreiben, was sich positiv auf die Kundenorientierung auswirkt.

Microservices können parallel betrieben werden. Dies ermöglicht Canary Releases: Inkrementelle Umstellung auf ein neueres Release. Die Anzahl an Anfragen an einen Microservice werden, wenn keine Fehler durch Monitoring der Applikation festgestellt werden können, kontinuierlich gesteigert bis die Vorgängerversion des Microservice schlussendlich außer Betrieb genommen werden kann (Sato, 2015).

Kundenorientierung kann auch durch die im folgenden vorgestellten Ansatzpunkte für Innovationen durch Microservices erreicht werden.

Für die Dimension *Kundenorientierung* werden folgende vier Hypothesen aufgestellt:

- *H2.4.1.:* Wenn ein Unternehmen ein System mit Microservice-Architektur anstatt eines monolithischen Systems, dann ist es kundenorientierter als bei Einsatz des monolithischen Systems.
- *H2.4.2.:* Wenn ein Unternehmen ein System mit Microservice-Architektur anstatt eines monolithischen Systems, dann kann es individuelle Kundenbedürfnisse besser berücksichtigt werden als im monolithischem System.
- *H2.4.3.:* Wenn ein Unternehmen ein System mit Microservice-Architektur DevOps betreibt, dann können Bug-Fixes schneller in Produktion gebracht werden.
- *H2.4.4.:* Wenn ein Unternehmen ein System mit Microservice-Architektur DevOps betreibt, dann wirkt sich das positiv auf die Servicequalität aus.

4.2.5 Innovationsorientierung

Microservice-Architekturen bieten durch ihr Modularisierungskonzept mehrere Ansatzpunkte für Produkt- und Prozessinnovationen, die sich in der Intensität, in den Subjekten als auch in der prozessualen und normativen Dimension unterscheiden.

Durch die Aufteilung in einzelne Applikationen wird die Umsetzung einzelner Services mit den Technologien, die für den Anwendungsfall am besten geeignet sind, ermöglicht. Die Nutzung passender Technologien kann zu Produktinnovationen führen. Wie beschrieben können Microservices einzeln verändert und ausgeliefert werden, wodurch unter Nutzung von Continuous Deployment kontinuierliche Innovationen entstehen können. Kontinuierliche Innovationen können ebenfalls durch Einsatz des Strangler Patterns erreicht werden. Die Funktionen von Altsystemen können dadurch schrittweise abgelöst werden, bis diese komplett außer Betrieb genommen werden können. Aber auch radikale Innovationen können durch den kompletten Austausch einzelner Microservices erreicht werden.

Durch den Einsatz von Monitoring und A/B-Tests nach Deployment eines Release kann überprüft werden, ob es sich auch um eine normative Innovation handelt. Continuous Deployment

ermöglicht ein schnelles Ausliefern geänderter Applikationen nach Einarbeitung des Feedbacks aus vorangegangenen A/B-Tests.

Microservices werden nach den Business Capabilities aufgeteilt und jeweils von einem Team betrieben. Das ermöglicht eine differenzierte Handhabung von Technologien, von unterstützenden Techniken und von Vorgehensmodellen innerhalb der Teams. Durch die Differenzierung können in den einzelnen Teams lokale Veränderung bezüglich der genannten Differenzierungsmerkmale getestet werden. Bei Erfolg können die Innovationen anschließend in andere Teams übertragen werden.

Die genannten Ansatzpunkte für Innovationen können sich insgesamt positiv auf die Risikofreudigkeit auswirken. Microservices-Architekturen ermöglichen Neues auszuprobieren, zu testen und bei Bedarf auch wieder schnell rückgängig zu machen.

Für die Dimension *Innovationsorientierung* werden folgende vier Hypothesen aufgestellt:

- *H2.5.1.:* Wenn ein Unternehmen ein System mit Microservice-Architektur anstatt eines monolithischen Systems, dann ist es innovationsfähiger als bei Einsatz des monolithischen Systems.
- *H2.5.2.:* Wenn ein Unternehmen ein System mit Microservice-Architektur einsetzt, dann kann es inkrementelle und radikale Innovationen hervorbringen.
- *H2.5.3.:* Wenn ein Unternehmen ein System mit Microservice-Architektur anstatt eines monolithischen Systems, dann kann eine höhere Anzahl lokaler Innovationen hervorbringen.
- *H2.5.4.:* Wenn ein Unternehmen ein System mit Microservice-Architektur einsetzt, dann wirkt sich das positiv auf die Risikofreudigkeit des Unternehmens aus.

5 FAZIT

Nachfolgend werden die Ergebnisse der Arbeit zusammengefasst und diskutiert. Zum Abschluss wird ein Ausblick gegeben, bei dem auf weitere Forschungsrichtungen eingegangen wird.

5.1 Zusammenfassung

Im Rahmen der vorliegenden Arbeit wurde der Fragestellung nachgegangen, wie sich Unternehmenskultur und Softwarearchitektur gegenseitig beeinflussen. Für einen ersten Schritt in Richtung der Beantwortung dieser Frage wurden deduktiv 59 Hypothesen zur Wirkungsbeziehung generiert. Hierzu wurden zunächst die theoretischen Grundlagen durch Erläuterungen der Themenkomplexe Unternehmenskultur und Softwarearchitektur dargelegt. Anschließend wurde der mögliche Einfluss der Unternehmenskultur auf die Softwarearchitektur mittels fünf Dimensionen der Unternehmenskultur beschrieben: Zielorientierung, Kommunikation und Information, Führung und Mitarbeiterorientierung, Kundenorientierung, Innovationsorientierung.

Unternehmenskultur beeinflusst durch die gemeinsam geteilten Werte das Verhalten des Softwarearchitekten im Architekturzyklus in der Analyse- und Entwurfsphase, in der Reflexion sowie im Review der Architektur und den damit verbundenen Aktivitäten und somit schließlich auch die entworfene Softwarearchitektur. Ebenso stellt die Unternehmenskultur ausgehend von verschiedenen Dimensionen Qualitätsanforderungen an die Softwarearchitektur. Abbildung 5-1 stellt die Zusammenhänge von Dimensionen der Unternehmenskultur und der Softwarearchitektur zusammengefasst grafisch dar. Weiterhin wurden zur Wirkung von Unternehmenskultur auf Softwarearchitektur deduktiv 35 Hypothesen generiert.

Der mögliche Einfluss der Softwarearchitektur auf die Unternehmenskultur wurde am Beispiel des Microservice-Architekturstils und den anfangs genannten Dimensionen von Unternehmenskultur beschrieben. Die Einführung oder Veränderung einer Softwarearchitektur führt folglich zu einer Veränderung der Unternehmenskultur, in dem sie gegenwärtige grundlegende Annahmen und Werte in Frage stellt. Ebenso kann eine neue oder veränderte Softwarearchitektur veränderte Rahmenbedingungen in der Ablauf- und Aufbauorganisation des Unternehmens schaffen. So wurde gezeigt, dass das Modularisierungskonzept des Microservice-Architekturstils Ziel-, Kunden- und Innovationsorientierung positiv begünstigen kann (vgl. Abbildung 5-2). Zur Wirkung des Microservice-Architekturstils auf die Unternehmenskultur wurden deduktiv 24 Hypothesen generiert.

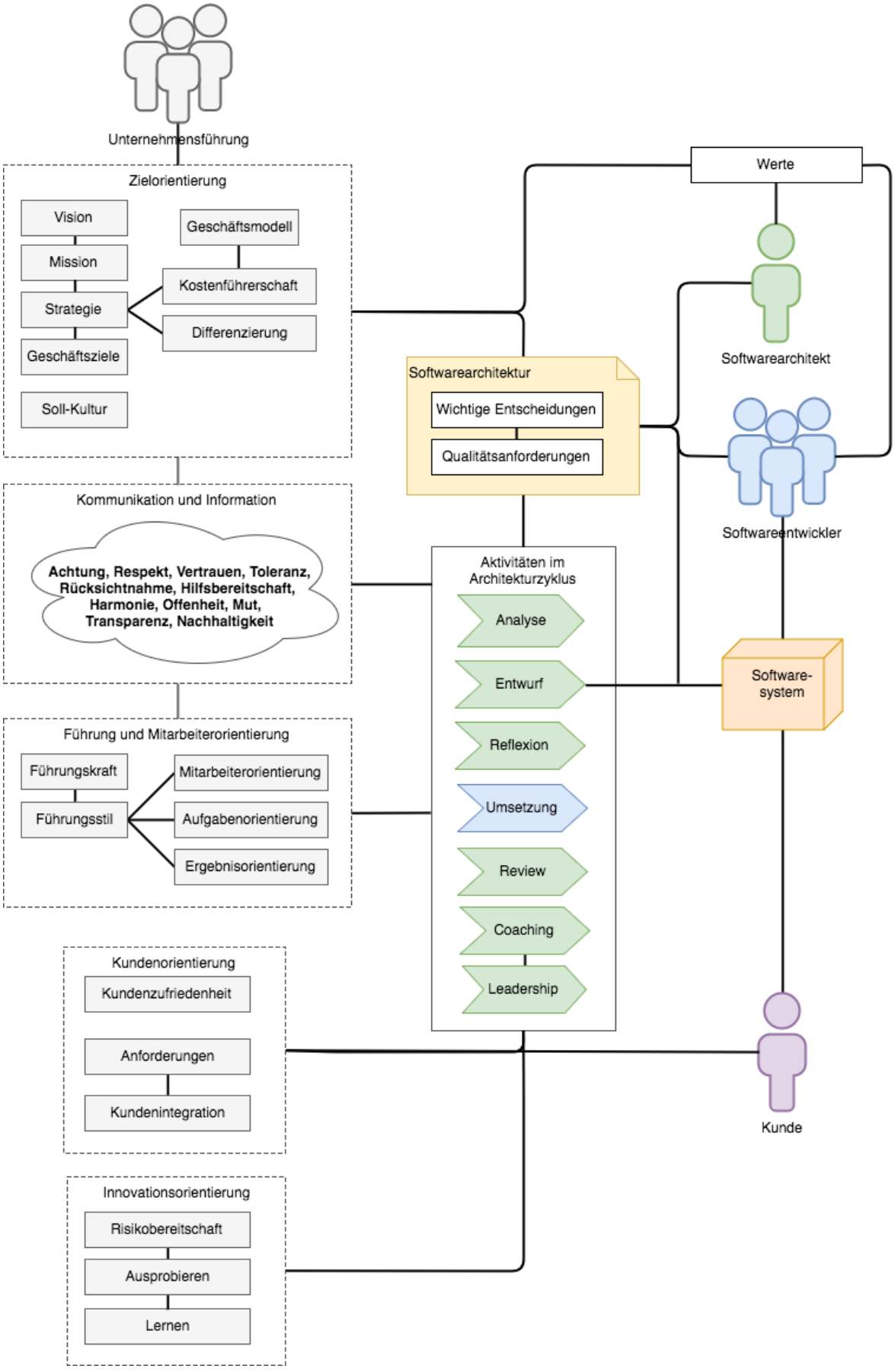

Abbildung 5-1: Zusammenfassende Übersicht über Zusammenhänge von Unternehmenskultur und Softwarearchitektur [Eigene Darstellung]

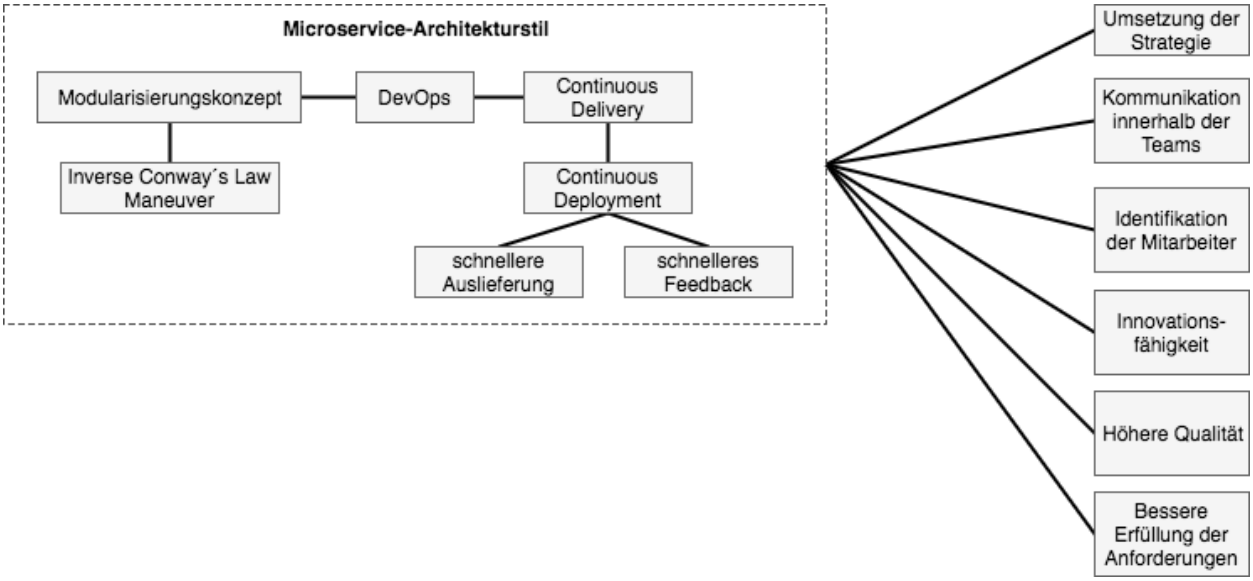

Abbildung 5-2: Zusammenfassende Übersicht über Zusammenhänge von Softwarearchitektur und Unternehmenskultur [Eigene Darstellung]

5.2 Diskussion der Ergebnisse

In der vorliegenden Arbeit konnten insgesamt 59 Hypothesen zu möglichen wechselseitigen Einflüssen von Unternehmenskultur und Softwarearchitektur auf Basis von Literatur und Praxisberichten aus der Theorie deduziert werden. Auch wenn dieses Verfahren nach Borchert, Goos und Strahler (2004) als der „Königsweg“ in der Wissenschaft bezeichnet wird, ist dieser Weg zur Erkenntnis mit Problemen verbunden, die sich auch zum Teil in dieser Arbeit widerspiegeln: „Da das deduktive Vorgehen auf bereits bekannten, grundlegenden Theorien basiert, führt dies oftmals zu Hypothesen, die den tradierten Denkgewohnheiten entsprechen, eine revolutionäre Hypothesenbildung ist nicht zu erwarten. Durch die Deduktion wird im Vergleich zur ursprünglichen Theorie oftmals kein zusätzlicher Informationsgehalt gewonnen. Dieser bleibt entweder gleich oder wird sogar noch systematisch vermindert. Selbst wenn der deduktive Schluss fehlerfrei ist, so sagt dieser nichts über die Wahrheit der Prämissen aus. Basiert die neue Hypothese auch nur auf einer einzigen falschen Prämisse, so ist es möglich, dass neben wahren auch falsche Konklusionen folgen.“ (Borchert, Goos, & Strahler, 2004, S. 15)

Die Basis für die Untersuchung bilden fünf Dimensionen von Unternehmenskultur, die aus den Untersuchungen von Unterreitmeier (2004) und Sackmann (2006) abgeleitet wurden. Unternehmenskultur ist ein weites Themengebiet mit vielfältigen Dimensionen, die jedoch im Rahmen dieser Untersuchung nicht alle Berücksichtigung finden konnten. Eine Identifikation und Auswertung weiterer Dimensionen, die im Kontext der Untersuchung unter Umständen zusätzlich relevant gewesen wären, würden über den Rahmen einer Masterarbeit hinausgehen. Ebenso konnten folgende Aspekte nicht berücksichtigt werden:

- Unternehmenskultur und Enterprise Architecture
- Differenzierung von Subkulturen innerhalb eines Unternehmens
- Einfluss nationaler Kulturen und Branchenkulturen

- Unternehmenskultur und Softwarearchitektur in Entwicklungsvorhaben von Software mit hoher Kritikalität oder Embedded Software
- Einfluss agiler Vorgehensmodelle und insbesondere deren Werte, Prinzipien und Praktiken
- Change Management bei Einführung von Softwarearchitektur(-stilen)
- Berücksichtigung von Unternehmen mit phasenorientierten im Gegensatz zu agilen Vorgehensmodellen
- Berücksichtigung von Unternehmen in regulierten Umgebungen
- Microservices bei Entwicklung „auf der grünen Wiese“
- Unternehmenskultur im Kontext des gesamten Softwareentwicklungsprozesses

Zusammenfassend trägt die vorgestellte Arbeit dazu bei, die Wirkung zwischen Unternehmenskultur und Softwarearchitektur besser zu verstehen. Die in Kapitel 1 aufgeworfenen Forschungsfragen konnten, auch wenn nicht empirisch bestätigt, beantwortet werden.

5.3 Ausblick

Die Disziplin der Softwarearchitektur und Rolle des Softwarearchitekten hat sich in den letzten Jahren gewandelt: Moderne Programmiersprachen und Frameworks nehmen viele Architekturentscheidungen ab. Die Aufgaben verlagern sich immer mehr in Richtung Leadership und Knowledge Manager (Hohpe, Ozkaya, Zdun, & Zi, 2016; Toth, 2016; Weinreich & Groher, 2016). Insbesondere im Kontext agiler Vorgehensmodelle ist die Rolle des Architekten und der Aufwand von Architektur als auch der Zeitpunkt von Architekturarbeit immer wieder Thema von Untersuchungen (Waterman, Noble, & Allan, 2015; Babar, Brown, & Mistrik, 2014).

Digitalisierung und Digitale Transformation beschäftigen Unternehmen seit einigen Jahren und stellen eine Herausforderung für sie dar: Der Fachkräftemangel im IT-Bereich erfordert effiziente Prozesse und maximalen Einsatz der vorhandenen Mitarbeiter (Bitkom Research, 2017). Der vermehrte Einsatz softwaregestützter Geschäftsmodelle und komplexere Anforderungen des Marktes an das Unternehmen stellen Anforderungen an die Flexibilität von IT-Systemen. Zusammengefasst lassen diese Aspekte darauf schließen, dass die Rolle der Softwarearchitektur in Zukunft weiter an Bedeutung gewinnt. Weiterhin muss bei Veränderungen der Softwarearchitektur, die Unternehmenskultur berücksichtigt werden.

Die vorliegende Arbeit hat gezeigt, wie Microservice-Architekturen verbesserte Rahmenbedingungen schaffen können, in denen sich Kultur entwickeln kann. In Anbetracht dessen, dass immer mehr Unternehmen den genannten Internetgiganten folgen und ihre monolithischen Systeme in der kommenden Zeit ablösen wollen (LeanIX, 2016), sollte auch in diesem Zusammenhang weiter geforscht werden. Es stellen sich folgende Fragen:

- Wie sieht eine optimale Unternehmenskultur für neue Microservice- und Migrationsprojekte aus?

- Wie kann die Unternehmenskultur für den Microservice-Architekturstil hinsichtlich einer verbesserten Kompatibilität verändert werden?
- Wie sieht ein Veränderungsprozess hin zu Microservice-Architekturen aus, der auch die Unternehmenskultur berücksichtigt?

An dieser Stelle sollte die Forschung der wechselseitigen Wirkung von Unternehmenskultur und Softwarearchitektur weiter ansetzen. Diese Arbeit konnte dazu erste Beiträge liefern, die im Folgenden empirisch validiert und gefestigt werden müssen.

ABBILDUNGSVERZEICHNIS

Abbildung 1-1: Untersuchungsgegenstand der vorliegenden Arbeit 2
Abbildung 2-1: Inhalte von Unternehmenskultur 4
Abbildung 2-2: Die drei Ebenen der Unternehmenskultur 8
Abbildung 2-3: Kulturelles Eisbergmodell 9
Abbildung 2-4: Abgrenzung der Funktionen der Unternehmenskultur 10
Abbildung 2-5: Wettbewerbskräfte innerhalb einer Branche 17
Abbildung 2-6: Anpassungskreislauf nach Miles und Snow 18
Abbildung 2-7: Zusammenhang von Strategie, Leitbild, Mission und Vision 20
Abbildung 2-8: Allgemeines Sender-Empfänger-Modell 21
Abbildung 2-9: Allgemeines Sender-Empfänger-Modell 22
Abbildung 2-10: Verankerungswerkzeuge für Kultur 23
Abbildung 2-11: Konzeptualisierung von Kundennähe 25
Abbildung 2-12: Dimensionen von Innovation 26
Abbildung 3-1: Einflüsse auf Softwarearchitektur 32
Abbildung 3-2: Kategorien für Qualitätsmerkmale nach ISO/IEC 9126 36
Abbildung 3-3: Architekturbrezel 38
Abbildung 3-4: Mögliche Besetzung der Architektenrolle 40
Abbildung 3-5: Qualitätsszenario inkl. Beispielszenario 43
Abbildung 3-6: Architekturbewertungsworkshop (Reflexion) 46
Abbildung 3-7: Stakeholder der Softwarearchitektur 46
Abbildung 3-8: Drei-Schichten-Architektur 49
Abbildung 3-9: Monolith & Microservices 49
Abbildung 3-10: Conway´s Law in Aktion 51
Abbildung 3-11: Inverse Conway´s Law 51
Abbildung 3-12: Zusammenhang von DevOps-Praktiken 53
Abbildung 4-1: Modell zur Leistungsfähigkeit von Unternehmen 54
Abbildung 4-2: Wirkung der Unternehmenskultur auf die Softwarearchitektur 57
Abbildung 4-3: Führung des Softwarearchitekten 64
Abbildung 4-4: Kano-Modell 67
Abbildung 4-5: Agiler Testquadrant 68
Abbildung 5-1: Zusammenfassende Übersicht über Zusammenhänge von Unternehmenskultur und Softwarearchitektur 78
Abbildung 5-2: Zusammenfassende Übersicht über Zusammenhänge von Softwarearchitektur und Unternehmenskultur 79

TABELLENVERZEICHNIS

Tabelle 1: Vor- und Nachteile der Koordination durch die Unternehmenskultur 12
Tabelle 3-1: Typische organisatorische Einflussfaktoren von Softwarearchitekturen 34
Tabelle 3-2: Typische technische Einflussfaktoren von Softwarearchitekturen..... 35

LITERATURVERZEICHNIS

(ISO), International organization for Standardization. (2001). *Software engineering - Product quality, ISO/IEC9126.*

(SEI), Software Engineering Institute of Carnegie Mellon University. (o.J.). *Bibliographic Software Architecture Definitions*. Abgerufen am 03. 01 2017 von Bibliographic Software Architecture Definitions: http://www.sei.cmu.edu/architecture/start/glossary/bibliographicdefs.cfm

(SEI), Software Engineering Institute of Carnegie Mellon University. (o.J.). *Classic Software Architecture Definitions*. Abgerufen am 03. 01 2017 von Classic Software Architecture Definitions: http://www.sei.cmu.edu/architecture/start/glossary/classicdefs.cfm

(SEI), Software Engineering Institute of Carnegie Mellon University. (kein Datum). *Community Software Architecture Definitions*. Abgerufen am 03. 01 2017 von Community Software Architecture Definitions: http://www.sei.cmu.edu/architecture/start/glossary/community.cfm

(SEI), Software Engineering Institute of Carnegie Mellon University. (kein Datum). *Modern Software Architecture Definitions*. Abgerufen am 03. 01 2017 von Modern Software Architecture Definitions: http://www.sei.cmu.edu/architecture/start/glossary/moderndefs.cfm

(SEI), Software Engineering Institute of Carnegie Mellon University. (o.J.). *Published Software Architecture Definitions*. Abgerufen am 03. 01 2017 von Published Software Architecture Definitions: http://www.sei.cmu.edu/architecture/start/glossary/published.cfm

Abelein, U., Sharp, H., & Paech , B. (2013). Does Involving Users in Software Development Really Influence System Success? *IEEE Software , 30* (6), 17-23.

Abolhassan, F. (2016). *Was treibt die Digitalisierung?* Wiesbaden: Gabler Verlag.

Abrahamsson, P., Babar, M. A., & Kruchten, P. (3 2010). Agility and Architecture: Can They Coexist? *IEEE Software , 27* (2), S. 16-22.

agilemanifesto.org. (2001). *Prinzipien hinter dem Agilen Manifest*. Abgerufen am 30. 01 2017 von Prinzipien hinter dem Agilen Manifest: http://agilemanifesto.org/iso/de/principles.html

Alisch, K., Arentzen, U., & Winter, E. (2004). *Gabler Wirtschaftslexikon.* Wiesbaden: Gabler Verlag.

Ambler, S. (2002). *Agile Modeling. Effective Practices for Extreme Programming and the Unified process.* New York: J. Wiley.

Amundsen, M., McLarty, M., Mitra, R., & Nadareishvili, I. (2016). *Microservice Architecture.* O'Reilly Media.

Angelov, S., Meesters, M., & Galster, M. (2016). Architects in Scrum: What Challenges Do They Face? *European Conference on Software Architecture. 9839*, S. 229-237. Cham: Springer.

Babar, M. A., Brown, A. W., & Mistrik, I. (2014). *Aligning Agile Processes and Software Architectures.* Waltham: Elsevier Inc.

Bachert, R. (2007). *Change-Management in Nonprofit-Organisationen.* Stuttgart: Schäffer-Poeschel.

Baetge, J., Schrewe, G., Schulz, R., & Solmecke, H. (27. 11 2007). Unternehmenskultur und Unternehmenserfolg: Stand der empirischen Forschung und Konsequenzen für die Entwicklung eines Messkonzeptes . *Journal für Betriebswirtschaft , 3* (57), S. 183–219.

Bannerman, P. L. (2009). Software architecture: Organizational perspectives. *LMSA '09 Proceedings of the 2009 ICSE Workshop on Leadership and Management in Software Architecture* (S. 37-42). DC, USA: IEEE Computer Society Washington.

Bass, L., & Clements, P. (2011). Business Goals and Architecture. In P. Avgeriou, J. H. Grundy, L. P., & M. I., *Relating Software Requirements and Architectures* (S. 183-195). Berlin, Heidelberg: Springer Verlag.

Bass, L., Clements, P., & Kazmann, R. (2012). *Software Architecture in Practice* (Third Edition Ausg.). Upper Saddle River, NJ: Addison-Wesley.

Bass, L., Weber, I., & Zhu, L. (2015). *DevOps: A Software Architect's Perspective.* Old Tappan, New Jersey: Addison Wesley.

Beck, K. (2005). *Extreme Programming Explained.* Addison Wesley.

Bitkom Research. (11. 07 2017). *Digitale Transformation der Wirtschaft (2. Auflage).* Abgerufen am 27. 07 2017 von https://www.bitkom-research.de/Digitale-Transformation-der-Wirtschaft

Bjarnason, E., Wnuk, K., & Regnell, B. (2011). Gaps, Requirements Are Slipping Through the Gaps. *19th IEEE International Requirements Engineering Conference (RE).* Piscataway, NJ: IEEE.

Bleicher, K. (2004). *Das Konzept Integriertes Management.* New York: Campus Verlag.

Bleicher, K. (1994). *Normatives Management.* Frankfurt: Campus-Verlag.

Bonsen, M. (1994). *Führen mit Visionen.* Wiesbaden: Gabler Verlag.

Booch, G. (2006). The Accidental Architecture. *IEEE SOFTWARE , 23* (3), 9-11.

Borchert, J., Goos, P., & Strahler, B. (2004). *Forschungsansätze.* Abgerufen am 27. 07 2017 von Arbeitsberichte des Instituts für Wirtschaftsinformatik, Professur für Anwendungssysteme und E-Business: http://webdoc.sub.gwdg.de/ebook/serien/lm/arbeitsberichte_wi2/2004_25.pdf

Bosch, J. (2000). *Design and Use of Software Architectures.* Amsterdam: Pearson Education.

Boston Consulting Group. (1988). Vision. *Die 34. Kronberger Konferenz.* München.

Britto, R., Šmite, D., & Damm, L.-O. (2016). Software Architects in Large-Scale Distributed Projects. *IEEE SOFTWARE , 3* (6), 48-55.

Brooks, F. P. (1995). *The mythical man-month: essays on software engineering.* Addison-Wesley.

Buxmann, P., Lehmann, S., & Hess, T. (2008). Software as a Service. *Wirtschaftinformatik , 6* (50), 500–503.

Cabrera, A., Cabrera, E. F., & Brajas, S. (2001). The key role of organizational culture in a multi-system view of technology-driven change. *International Journal of Information Management* (21), 245-261.

Calçado, P. (11. 04 2014). *Backstage Blog - Building Products at SoundCloud - Part I: Dealing with the Monolith*. Abgerufen am 21. 07 2017 von https://developers.soundcloud.com/blog/building-products-at-soundcloud-part-1-dealing-with-the-monolith

Cameron, K. S., & Freeman, S. J. (1985). *Cultural congruence, strength, and type: relationships to effectiveness.* Michigan: School of Business Administration, University of Michigan.

Chandler, A. D. (1962). *Strategy and structure.* Cambridge, Massachusetts: Harvard University Press.

Chies, S. (2016). *Change Management bei der Einführung neuer IT-Technologien.* Wiesbaden: Springer Verlag.

Christ, A. (14. 06 2017). *Wie Sie Ihre IT-Architektur neu erfinden.* Abgerufen am 03. 07 2017 von https://www.leanix.net/de/download/Reinvent-your-IT-architecture

Clarke, P., & O'Connor, R. V. (21. 12 2012). The situational factors that affect the software development process: Towards a comprehensive reference framework . *Information and Software Technology* (54).

Clements, P., Kazman, R., Klein, M., & Verma, P. (1 2007). The Duties, Skills, and Knowledge of Software Architects. (IEEE, Hrsg.) *2007 Working IEEE/IFIP Conference on Software Architecture (WICSA'07)* .

Conway, M. E. (4 1968). How do committees invent. *Datamation* , S. 28-31.

Dasanayake, S., Markkula, J., Aaramaa, S., & Oivo, M. (2016). An Empirical Study on Collaborative Architecture Decision Making in Software Teams. *Software Architecture ECSA 2016. 9839*, S. 238-246. Cham: Springer.

Deal, T., & Kennedy, A. (1982). *Corporate cultures: The rites and rituals of corporate life.* Reading, Mass: Addison-Wesley.

Derenthal, K. (2009). *Innovationsorientierung von Unternehmen.* Wiesbaden: Gabler Verlag.

DIN EN ISO 9000:2015. (2015). *Qualitätsmanagementsysteme – Grundlagen und Begriffe.* Berlin.

Doppler, K., & Lauterburg, C. (2014). *Change Management.* Campus Verlag.

Dormayer, H.-J., & Kettnern, T. (1987). Kulturkonzepte in der allgemeinen Kulturforschung. In E. Heinen, *Unternehmenskultur* (S. 49-66). München: Oldenbourg.

Drucker, P. F. (1954). *The Practice of Management.* New York.

Evans, E. J. (2003). *Domain-Driven Design.* Addison Wesley.

Faber, R. (3 2010). Architects as Service Providers. *IEEE Software , 27* (2), S. 33-40.

Falessi, D., Cantone, G., Kazman, R., & Kruchten, P. (2011). Decision-Making Techniques for Software Architecture Design: A Comparative Survey. *ACM Computing Surveys , 43* (4).

Farenhorst, R., Hoorn, J. F., Lago, P., & van Vliet, H. (2011). The Lonesome Architect. *Journal of Systems and Software , 84* (9), 1424-1435 .

Fischel, N. (2008). *Unternehmenskultur und radikale Innovation.* Lohmar: JOSEF EUL.

Fleig, J. (10. 05 2016). *business-wissen.de*. Abgerufen am 26. 05 2017 von Vision und Mission: https://www.business-wissen.de/hb/was-vision-und-mission-im-unternehmen-bewirken/

Fowler, M. (5 2004). *Growing an Architecture*. Abgerufen am 12. 1 2017 von Growing an Architecture: https://www.martinfowler.com/articles/designDead.html#GrowingAnArchitecture

Fowler, M. (29. 06 2004). *StranglerApplication*. Abgerufen am 26. 07 2017 von https://www.martinfowler.com/bliki/StranglerApplication.html

Fowler, M., & Dörnenburg, E. (29. 4 2016). *Architecture without architects*. Abgerufen am 2. 2 2017 von Architecture without architects: http://www.ustream.tv/recorded/86152575

Fowler, M., & Lewis, J. (25. 03 2015). *Microservices*. Abgerufen am 21. 07 2017 von A definition of this new architectural term: https://martinfowler.com/articles/microservices.html

Fowler, S. J. (2016). *Production-Ready Microservices.* O'Reilly Media.

Günter, B., & Huber, O. (1996). Beschwerdemanagement als Instrument der Customer Integration. In M. Kleinaltenkamp, S. Fließ, & F. Jacob, *Customer Integration* (S. 245-257). Wiesbaden: Gabler Verlag.

Gebhart, M. (01. 06 2016). *Microservices, ReactJS und Co: Warum attraktive Software-Architekturen und moderne Technologien wirklich wichtig sind.* Abgerufen am 26. 07 2017 von https://www.iteratec.de/tech-blog/artikel/news/microservices-reactjs-und-co-warum-attraktive-software-architekturen-und-moderne-technologien-wir/

Glicken, J. (2000). Getting stakeholder participation ‘right’: a discussion of participatory processes and possible pitfalls. *Getting stakeholder participation ‘right’: a discussion of participatory processes and possible pitfalls , 3* (6), 305–310.

Goffart, A. (2016). Beeinflusst HR-Technologie die Unternehmenskultur? *AuA* (10), S. 614-617.

Google Trends. (21. 07 2017). *Google Trends*. Abgerufen am 21. 07 2017 von Stichwort: Microservices: https://trends.google.com/trends/explore?date=today%205-y&q=Microservices

Gopal, A., Espinosa, J. A., Gosain, S., & Darcy, D. P. (2011). Coordination and Performance in Global Software Service Delivery: The Vendor's Perspective. (IEEE, Hrsg.) *IEEE Transactions on Engineering Management , 4* (58), 772 - 785.

Haase, M., Jöhnk, J., Lipowsky, S., & Urbach, N. (2017). Der Einfluss des Agilitätsgrads auf den Erfolg von Softwareentwicklungsprojekten unter Berücksichtigung der Unternehmenskultur.

Tagungsband der 13. Internationalen Tagung Wirtschaftsinformatik (WI), (S. 454-468). St. Gallen, Schweiz.

Hans, N. (2003). *Aufbruch im Mittelstand.* Wiesbaden: Gabler Verlag.

Hauff, V. (1987). *Unsere gemeinsame Zukunft - Der Brundtland-Bericht der Weltkommission für Umwelt und Entwicklung.* Greven: Eggenkamp Verlag.

Haug, A. (2012). *Multisensuelle Unternehmenskommunikation.* Wiesbaden: Gabler Verlag.

Hauschildt, J., & Salomo, S. (2007). *Innovationsmanagement.* München: Vahlen.

Herberhold, C. (2016). *Unternehmenskultur zum Enterprise Resource Planning.* Hamburg: Verlag Dr. Kovač.

Herzwurm, G. (1998). Kundenorientierung in der Softwareentwicklung. *Information Management for Business and Competitive Intelligence and Excellence* (S. 281-297). Braunschweig, Wiesbaden: Vieweg Verlag.

Heusingfeld, A. (21. 06 2017). *Microservices: Warum wollen Sie das?* Abgerufen am 26. 07 2017 von https://jaxenter.de/microservices-warum-57795

Hofmeister, C., Nord, R., & Soni, D. (2000). *Applied Software Architecture.* Boston: Addison-Wesley.

Hofstede, G. (1980). Kultur und Organisation. In E. Grochla, *HWO* (2. Auflage Ausg., S. 1168-1182). Stuttgart: Poeschel.

Hohenadl, S. (04. 01 2015). *Why AutoScout24 Changes its technology*. Abgerufen am 27. 07 2017 von http://techblog.scout24.com/2015/01/autoscout24-changes-technology/

Hohpe, G., Ozkaya, I., Zdun, U., & Zi, O. (2016). The Software Architect's Role in the Digital Age. *IEEE Software , 33* (6), 30-39.

Homburg, C. (1998). *Kundennähe von Industriegüterunternehmen.* Wiesbaden: Gabler Verlag.

Homma, N., Bauschke, R., & Hofmann, L. M. (2014). *Einführung Unternehmenskultur.* Wiesbaden: Springer Gabler.

Hungenberg, H. (2014). *Strategisches Management in Unternehmen.* Wiesbaden: Gabler Verlag.

Jacobson, D. (2012. 06 2012). *Netflix Technology Blog*. Abgerufen am 20. 07 2017 von Embracing the Differences : Inside the Netflix API Redesign.

Kühn, R. (1992). Methodische Überlegungen zum Umgang mit der Kundenorientierung im Marketing-Management. *Marketing ZFP* (2), S. 97-107.

Kazman, R., & Bass, L. (2005). *Categorizing Business Goals for Software Architectures.* Software Engineering Institute.

Keller, W. (2007). *IT-Unternehmensarchitektur.* Heidelberg: dpunkt.verlag GmbH.

Koch, R. (20. 6 2016). *My-Leadership*. Abgerufen am 11. 7 2017 von Warum Ihr Unternehmen ein Führungsleitbild benötigt: https://www.my-leadership.de/2016/06/20/warum-ihr-unternehmen-ein-f%C3%BChrungsleitbild-ben%C3%B6tigt/

Kroeber, A. L., & Kluckhohn, C. (1952). *Culture − a Critical Review of Concepts and Definitions.* New York: Vintage Books.

Kruchten, P. (1995). The 4+1 View Model of Architecture. *IEEE Software , 12* (6), 42-50.

Kruchten, P. (1999). The Software Architect, and the Software Architecture Team. In P. Donohoe, *Software Architecture; First IFIP Conference on Software Architecture (WICSA1)* (S. 565-583). Boston: Kluwer Academic Publishers.

Kruchten, P. (2008). What do software architects really do? . *The Journal of Systems and Software , 81*, 2413–2416 .

Krulis-Randa, J. (1990). *Die Unternehmenskultur: Ihre Grundlagen und ihre Bedeutung für die Führung der Unternehmung.* Heidelberg: Physica-Verlag.

Lang, C. (2003). *Organisation der Software-Entwicklung.* Wiesbaden: Dt. Univ.-Verl.

Larmann, C., & Vodde, B. (2010). *Scaling Lean and Agile Development: Thinking and Organizational Tools for Large-Scale Scrum: Successful Large, Multisite and Offshore Products with Large-scale Scrum.* Longman: AddisonWesley.

LeanIX. (23. 12 2016). *Microservice Studie: Was kommt nach agil?* Abgerufen am 05. 07 2017 von Es wird Zeit, Microservices zu adoptieren: https://www.leanix.net/de/download/Microservices-Study

Lippold, A. (2007). *Die Innovationskultur.* Göttingen: Cuvillier.

Müller, R. (o.J.). *Vision Unternehmen*. Abgerufen am 20. 06 2017 von http://www.sunternehmensentwicklung.de/vision-unternehmen/allgemein/vision-unternehmen.html

Müller-Stewens, G., & Lechner, C. (2016). *Strategisches Management.* Stuttgart: Schäffer-Poeschel Verlag.

Malan, R., & Bredemeyer, D. (25. 7 2000). *Creating An Architectural Vision: Collecting Input.* Abgerufen am 2. 2 2017 von Creating An Architectural Vision: Collecting Input: http://www.bredemeyer.com/pdf_files/vision_input.pdf

Mast, C. (2004). Kommunikation. In G. Schreyögg, & A. Von Werder, *Handwörterbuch Unternehmensführung und Organisation* (S. 596-606). Stuttgart: Schäffer-Poeschel.

Mast, C. (2016). *Unternehmenskommunikation.* Konstanz, München: UKV Vertragsgesellschaft.

Matje, A. (1996). *Unternehmensleitbilder als Führungsinstrument.* Gabler Verlag.

McBride, M. R. (2007). The software architect. *Communications of the ACM - ACM at sixty: a look back in time , 50* (5), 75-81.

Mellis, W. (2004). *Projektmanagement der SW-Entwicklung.* Wiesbaden: Vieweg Verlag.

Menzenbach, J. (2012). *Visionäre Unternehmensführung.* Wiesbaden: Gabler Verlag.

Miles, R. E., Snow, C. C., Meyer, A. D., & Coleman, H. J. (1978). Organizational Strategy, Structure, and Process. *The Academy of Management Review , 3* (3), S. 546-562.

Miles, R., & Snow, C. (2003). *Organizational Strategy, Structure, and Process.* Stanford, Kalifornien: Stanford Business Books.

Morris, K. (2016). *Infrastructure as a Code.* O'Reilly Media.

Newman, S. (2015). *Building Microservices.* O'Reilly Media, Inc.

Nicolai, C. (2012). *Grundlagen der Unternehmensorganisation.* Berlin: Berliner Wissenschafts-Verlag.

Paasivaara, M., Durasiewicz, S., & Lassenius, C. (2008). Distributed Agile Development: Using Scrum in a Large Project. *2008 IEEE International Conference on Global Software Engineering* (S. 87-95). Bangalore, India: IEEE.

Paul, F. (16. 05 2014). *New Relic Tech Topics*. Abgerufen am 22. 07 2017 von The Incredible True Story of How DevOps Got Its Name: https://blog.newrelic.com/2014/05/16/devops-name/

Persa, D. (16. 01 2015). *From Jimmy to Microservices: Rebuilding Zalando's Fashion Store*. Abgerufen am 27. 07 2017 von https://jobs.zalando.com/tech/blog/from-jimmy-to-microservices-rebuilding-zalandos-fashion-store/?gh_src=4n3gxh1

Peters, T., & Waterman, R. (1982). *In search of excellence : lessons from America's best-run companies.* New York: Harper & Row.

Pichler, R. (16. 06 2012). *The Product Canvas*. Abgerufen am 01. 02 2017 von The Product Canvas: http://www.romanpichler.com/blog/the-product-canvas/

Pittrof, M. (2011). *Die Bedeutung der Unternehmenskultur als Erfolgsfaktor für Hidden Champions.* Wiesbaden: Gabler.

Pohl, J. (29. 3 2012). *Schöne Aussichten – Hintergrundpapier.* Abgerufen am 20. 6 2017 von http://www.fortschrittszentrum.de/dokumente/2012-03_SA_Hintergrundpapier.pdf

Pols, A., & Vogel, M. (14. 03 2017). *Bitkom.* Abgerufen am 20. 07 2017 von Cloud Monitor 2017: https://www.bitkom.org/Presse/Anhaenge-an-PIs/2017/03-Maerz/Bitkom-KPMG-Charts-PK-Cloud-Monitor-14032017.pdf

Porter, M. E. (1999). *Wettbewerbsstrategie.* Frankfurt am Main: Campus Verlag.

Posch, T., Birken, K., & Gerdom, M. (2011). *Basiswissen Softwarearchitektur* (3., aktualisierte und erweiterte Auflage Ausg.). Heidelberg: dpunkt.verlag GmbH.

Preissler, J., & Tigges, O. (2015). *Docker - perfekte Verpackung von Microservices*. Abgerufen am 21. 07 2017 von https://www.sigs-datacom.de/uploads/tx_dmjournals/preissler_tigges_OTS_Architekturen_15.pdf

Röhner, J., & Schütz, A. (2012). *Psychologie der Kommunikation.* Wiesbaden: Springer Fachmedien.

Raymond, E. S. (2008). *The art of UNIX programming.* Addison-Wesley.

Roock, S., & Pichler, R. (2011). Die Architekturvision in Scrum: Vorausplanung und emergentes Design balancieren. *OBJEKTspektrum* (4), S. 46-49.

Rook, M. (20. 11 2016). *Michiel Rook's blog*. Abgerufen am 26. 07 2017 von The Strangler pattern in practice: https://www.michielrook.nl/2016/11/strangler-pattern-practice/

Ruehl, S. T., & Andelfinger, U. (2011). Applying software product lines to create customizable software-as-a-service applications. *SPLC '11 Proceedings of the 15th International Software Product Line Conference.* München: ACM.

Rupp, C., & Sophist Group. (2001). *Requirements-Engineering und -Management.* Hanser.

Sackmann, S. (14. 09 2006). Abgerufen am 03. 11 2016 von Betriebsvergleich Unternehmenskultur: Welche kulturellen Faktoren beeinflussen den Unternehmenserfolg?: https://www.dgfp.de/wissen/personalwissen-direkt/dokument/84192/herunterladen

Sackmann, S. (2004). *Erfolgsfaktor Unternehmenskultur.* Wiesbaden: Gabler.

Sato, D. (25. 06 2015). *CanaryRelease*. Abgerufen am 26. 07 2017 von https://martinfowler.com/bliki/CanaryRelease.html

Schein, E. H. (2010). *Organizational Culture and Leadership.* San Francisco: John Wiley and Sons.

Schein, E. H. (2017). *Organizational Culture and Leadership.* Hoboken, New Jersey: John Wiley & Sons, Inc.

Schein, E. H. (2006). *The Ed Schein Corporate Culture Survival Guide* (2. Auflage Ausg.). (G. Fatzer, Hrsg.) Bergisch Gladbach: EHP Edition Humanistische Psychologie.

Schein, E. H. (1995). *Unternehmenskultur.* Frankfurt am Main, New York: Campus Verlag.

Schiffelholz, A. (2014). *Stabilität und Wechsel bei Miles-und-Snow-Strategietypen.* München und Mering: Rainer Hampp Verlag.

Schlegelmilch, G. (1999). *Management strategischer Innovationsfelder.* Deutscher Universitätsverlag.

Scholz, C. (9 1992). Visionäres Personalmanagement als strategische Chance. *Karriereberater* , 33ff.

Scholz, C., & Hofbauer, W. (1990). *Organisationskultur.* Wiesbaden: Gabler.

Schrader, S. (1995). *Spitzenführungskräfte, Unternehmensstrategie und Unternehmenserfolg.* Tübingen: Mohr.

Schulz von Thun, F. (2006). *Miteinander reden: Kommunikationspsychologie für Führungskräfte.* Hamburg: Rowohlt Verlag.

Schumpeter, J. A. (1939). *Business Cycles.* New York, London: McGraw-Hill.

Schwaber, K., & Beedle, M. (2002). *Agile Software Development with Scrum.* Upper Saddle River, NJ: Prentice Hall.

Schwaber, K., & Sutherland, J. (06 2016). *The Scrum Guide.* Abgerufen am 17. 07 2017 von http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf

Schwegler, R. (2009). *Moralisches Handeln von Unternehmen.* Wiesbaden: Gabler Verlag.

Shannon, C. E., & Weaver, W. (1949). *The mathematical theory of communication.* Urbana: University of Illinois Press.

Shawn, J. (13. 06 2016). *Follow the Simple Path of Nirvana to DevOps*. Abgerufen am 18. 07 2017 von https://devops.com/follow-simple-path-nirvana-devops/

Sherman, S., & Hadar, I. (2015). Toward Defining the Role of the Software Architect. *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering* (S. 71-76). IEEE.

Slater, S. F., Olson, E. M., & Hult, G. T. (2006). The moderating influence of strategic orientation on the strategy formation capability–performance relationship. *Strategic Management Journal , 27* (12), 1123–1233.

Starke, G. (2011). *Effektive Software-Architekturen.* München: Carl Hanser Verlag.

Steinacker, G. (03. 06 2015). *Von Monolithen und Microservices*. Abgerufen am 27. 07 2017 von https://dev.otto.de/2015/06/03/von-monolithen-und-microservices/

Sun, W., Zhang, X., Jie Guo, C., Sun, P., & Su, H. (2008). Software as a Service: Configuration and Customization Perspectives . *Congress on Services Part II, 2008. SERVICES-2. IEEE.* Beijing, China: IEEE.

Sztuka, A. (2017). *Normatives Management: Vision, Mission und strategische Ziele*. Abgerufen am 20. 6 2017 von http://www.manager-wiki.com/strategie-grundlagen/5- normatives-management-vision-mission-und-strategische-ziele

ThoughtWorks, Inc. (01 2015). *Inverse Conway Maneuver*. Abgerufen am 26. 07 2017 von https://www.thoughtworks.com/radar/techniques/inverse-conway-maneuver

Tofan, D., Galster, M., & Avgeriou, P. (2013). Difficulty of Architectural Decisions – A Survey with Professional Architects. *Software Architecture ECSA 2013. 7957*, S. 192-199. Berlin, Heidelberg: Springer.

Toth, S. (2014). *Vorgehensmuster für Softwarearchitektur.* München: Carl Hanser Verlag .

Toth, S. (03. 02 2016). *Vortrag OOP 2016: No Bullshit – Architekturarbeit neu gedacht.* Abgerufen am 26. 07 2017 von http://www.embarc.de/wp-content/uploads/2016/02/NoBullshit_ST_fin.pdf

Unterreitmeier, A. (2004). *Unternehmenskultur bei Mergers & Acquisitions.* Wiesbaden: Gabler.

Urbach, N., & Ahlemann, F. (2017). *Die IT-Organisation im Wandel.* Abgerufen am 27. 07 2017 von Implikationen der Digitalisierung für das IT-Management: http://www.fim-rc.de/Paperbibliothek/Veroeffentlicht/677/wi-677.pdf

Vahs, D., & Schäfer-Kunz, J. (2005). *Einführung in die Betriebswirtschaftslehre.* Stuttgart: Schäffer-Poeschel.

Venzin, M., Rasner, C., & Mahnke, V. (2010). *Der Strategieprozess.* Frankfurt a.M., New York: Campus Verlag.

VersionOne. (2016). *The 10th Annual State of Agile Report.* Abgerufen am 3. 2 2017 von The 10th Annual State of Agile Report: https://versionone.com/pdf/VersionOne-10th-Annual-State-of-Agile-Report.pdf

Vigenschow, U. (2015). *APM - Agiles Projektmanagement.* Heidelberg: dpunkt.verlag GmbH.

Von Hehn, S. (2016). *Kulturwandel in Organisationen.* Berlin, Heidelberg: Springer Verlag.

Von Heynitz, H. (20. 03 2017). *Fertigungsindustrie: Der Kunde mag es individuell.* Abgerufen am 27. 07 2017 von https://klardenker.kpmg.de/optimieren/transformation/fertigungsindustrie-der-kunde-mag-es-individuell/

Warner, W. P. (12 1982). *What is Software Engineering Environment (SEE)? (Desired Characteristics).* Abgerufen am 26. 07 2017 von http://www.dtic.mil/dtic/tr/fulltext/u2/a124280.pdf

Waterman, M., Noble, J., & Allan, G. (2015). How Much Up-Front? A Grounded Theory of Agile Architecture. *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. 10027*, S. 432. IEEE/ACM.

Watzka, K. (2017). *Zielvereinbarungen in Unternehmen.* Wiesbaden: Gabler Verlag.

Weinreich, R., & Groher, I. (28. 10 2016). The Architect's Role in Practice: From Decision Maker to Knowledge Manager? *IEEE Software* , S. 63-69.

Welge, M. K. (2008). *Strategisches Management.* Wiesbaden: Gabler.

Wenninger, G. (2000). *Lexikon der Psychologie: Verhalten*. Abgerufen am 21. 10 2016 von http://www.spektrum.de/lexikon/psychologie/verhalten/16243

Wheatley, M., & Frieze, D. (2006). *The Berkana Institute.* Abgerufen am 20. 6 2017 von Using Emergence to Take Social Innovation to Scale: http://www.margaretwheatley.com/articles/using-emergence.pdf

Wieland, J. (1999). Ethik im Unternehmen – Widerspruch in sich selbst? *Personalführung* (8), S. 18-23.

Wilson, I. (10 1992). Realizing the power of strategic vision. *Long Range Planning , 25* (5), S. 18-28.

Wolff, E. (2015). *Continuous Delivery.* Heidelberg: dpunkt.verlag.

Wolff, E. (2016). *Microservices.* Heidelberg: dpunkt.verlag.

Zörner, S. (2015). *Softwarearchitekturen dokumentieren und kommunizieren* (2. Auflage Ausg.). München: Carl Hanser Verlag.

Zollner, G. (1995). *Kundennähe in Dienstleistungsunternehmen.* Wiesbaden: Deutscher Universitätsverlag.

Zollondz, H.-D. (2011). *Grundlagen Qualitätsmanagement.* München: Oldenbourg Wissenschaftsverlag.