EUROPA-UNIVERSITÄT
VIADRINA
FRANKFURT (ODER)

# Systematisierung und Bewertung der Customizing- und Modifikationsmöglichkeiten von cloudbasierten ERP-Systemen

I n a u g u r a l d i s s e r t a t i o n

zur Erlangung des akademischen Grades
„Doktor der Wirtschaftswissenschaften“
(Dr. rer. pol.)

eingereicht an der

Wirtschaftswissenschaftlichen Fakultät
der Europa-Universität Viadrina
in Frankfurt (Oder)
05.05.2017

von

Dipl.-Kfm. Dawid Nowak

Frankfurt (Oder)

Erstgutachter: Prof. Dr. Karl Kurbel
Zweitgutachter: Prof. Dr. Achim Koberstein

Wirtschaftswissenschaftliche Fakultät
der Europa-Universität Viadrina, Frankfurt (Oder)

Tag der Disputation: 26.09.2017

# Danksagung

Ich danke Herrn Prof. Dr. Karl Kurbel, meinem Doktorvater, für seine kompetente Betreuung und professionelle Begleitung sowie für seine außerordentlich sachkundige, erfahrene und wertvolle Unterstützung für die vorliegende Dissertation.

Insbesondere danke ich Herrn Prof. Dr. Karl Kurbel für seine konstruktiven Empfehlungen und hilfreichen Ratschläge sowie die Ermutigung und Motivation bei der Realisierung meines Promotionsvorhabens.

Mein großer Dank geht ebenfalls an Herrn Prof. Dr. Achim Koberstein für die hilfreiche und wissenschaftliche Betreuung als Zweitgutachter.

Ich möchte mich auch bei allen meinen Kolleginnen und Kollegen bedanken, mit denen ich eine gute und produktive Zeit am Lehrstuhl für Wirtschaftsinformatik der Europa-Universität Viadrina verbracht habe. Mein besonderer Dank dabei gilt Frau Olga Stawnicza für die zahlreichen konstruktiven Gespräche und ihre wertvollen Empfehlungen während meines Forschungsprojektes.

Des Weiteren gilt mein besonderer Dank Frau Elvira Fleischer für ihre Hilfsbereitschaft bei den administrativen Aufgaben als auch für ihre wertvollen Ratschläge bei der Formatierung des Manuskriptes. Bedanken möchte ich mich auch bei Herrn Steffen Kern für seine stets kompetente IT-Unterstützung im Laufe des ganzen Dissertationsprojektes.

An dieser Stelle möchte ich mich auch bei allen bedanken, die an der Korrektur meiner Dissertationsschrift mitgewirkt haben.

Nicht zuletzt geht mein außerordentliches Dankeschön an meine Frau Aleksandra, meine Eltern und meine Schwester für ihre unendliche Geduld, das Verständnis und die stets liebvolle Unterstützung.

Frankfurt (Oder), 05.05.2017

# Abstract

In der Vergangenheit wurden die ERP-Systeme typischerweise auf den Servern installiert, welche sich im Bereich des Anwenderunternehmens befanden. Zwischenzeitlich sind neue Bereitstellungskonzepte entstanden, die es den Systemanbietern erlauben, eine größere Kundenanzahl effizienter zu erreichen. Aufbauend auf den Prinzipien des Cloud-Computing-Paradigma kann die ERP-Standardsoftware heutzutage dem Anwenderunternehmen auch in Form einer Dienstleistung über das Internet zur Verfügung gestellt werden. Die Anforderungen dieses Bereitstellungsmodells bleiben jedoch nicht ohne Auswirkung auf die Systemarchitektur und die bereitgestellte Systemflexibilität.

Die vorliegende Arbeit thematisiert die Problematik der Adaptierbarkeit der cloudbasierten ERP-Systeme und widmet sich einer vergleichenden Analyse der Customizing-Möglichkeiten, welche auf die Erarbeitung eines architekturübergreifenden Systematisierungsrahmens der bestehenden Customizing-Konzepte abzielen.

Die Validierung der bereitgestellten Systemflexibilität erfolgt anhand konkreter praxisnaher Szenarien und trägt systematisch zur Bewertung der Anpassungs- und Modifikationsmöglichkeiten der cloudbasierten ERP-Systeme bei. Trotz der Tatsache, dass die cloudbasierten ERP-Systeme vergleichend zu konventionellen On-Premise ERP-Systemen in der Regel weniger tiefgreifende Adaptionen zulassen, lassen sich diese Aussagen nicht im gleichen Ausmaß auf alle cloudbasierten ERP-Systeme verallgemeinern. Die Erkenntnisse der durchgeführten Machbarkeitsstudien weisen auf die Notwendigkeit einer differenzierten Betrachtung der Customizing- und Modifikationsmöglichkeiten von cloudbasierten ERP-Systemen hin, die durch unterschiedliche Konformität mit dem SaaS-Modell gekennzeichnet sind.

Die Einbeziehung der architektonischen Aspekte in die Analysen und die darauffolgende statistische Auswertung der empirischen Daten bilden die Grundlage für die Identifizierung der systemübergreifenden Tendenzen hinsichtlich der bereitgestellten Adaptierbarkeit und der architekturbedingten Differenzen in dem bereitgestellten Flexibilitätsumfang. Durch die Systematisierung der gewonnenen Erkenntnisse entlang der drei Betrachtungsebenen, welche auf die Customizing-Ansätze, adaptierbare Systemelemente und bereitgestellte Adaptionswerkzeuge ausgerichtet sind, schafft diese Arbeit einen Bezugsrahmen, der als praktische Hilfestellung bei der Bewertung der Systemflexibilität differenzierter Cloud-ERP-Systeme verwendet werden kann.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# 1 Einführung

Im Rahmen dieses Kapitels werden ausgehend von der Problemstellung und der Motivation, welche der vorliegenden Untersuchung zugrunde liegen, deren zentrale Begriffe erörtert. Darauf aufbauend werden die Ziele für die Untersuchung aufgestellt und die Methodik zum Erreichen dieser Ziele präsentiert. Dieser Abschnitt endet mit einer Präsentation des Aufbaus der Dissertation.

## 1.1 Problemstellung und Motivation

Ein Enterprise-Resource-Planning-System (ERP-System) stellt eine integrierte Standardsoftware dar, die entwickelt wurde, um unternehmerische Prozesse zu unterstützen (vgl. Mijač et al. 2013, S. 132). Die ERP-Systeme *„*[...] *bestehen aus einem Basissystem und verschiedenen funktionsbezogenen Komponenten für unterschiedliche betriebliche Anwendungsgebiete. Diese Komponenten basieren auf einer einheitlichen Datenbank und sind funktional so angelegt, dass auch unternehmensübergreifende Geschäftsprozesse abgebildet werden können.“* (Stahlknecht und Hasenkamp 2006, S. 239 f.). Das Konzept einer integrierten berieblichen Anwendungssoftware hat sich evolutionär seit den sechziger Jahren des zwanzigsten Jahrhunderts entwickelt (mehr dazu Kurbel 2016, S. 1 ff.). Heutzutage stellen die ERP-Systeme das Rückgrat der meisten Unternehmen dar.

Während in der Vergangenheit die ERP-Systeme üblicherweise auf der Seite des Kundenunternehmens als sogenannte On-Premise-Software installiert wurden, sind zwischenzeitlich alternative Betreibermodelle entstanden, mit deren Hilfe das ERP-System dem Kunden zur Verfügung gestellt werden kann.

Das neue Paradigma, welches seit einigen Jahren mit dem Cloud-Computing entsteht, hat die neue Tendenz entwickelt, dem Anwender die Betriebssoftware in Form einer Dienstleistung bedarfsbezogen, also *on demand* über das Internet bereitzustellen. Das Software-as-a-Service-Betreibermodell (SaaS) erfreut sich eines zunehmenden Interesses seitens der Hersteller der betrieblichen Anwendungssoftware im Allgemeinen und seitens der ERP-Systemhersteller im Besonderen. Es werden immer mehr ERP-Systeme unter dem Label „Cloud“ bzw. „on demand“ vermarktet.

Die sogenannten On-Demand-ERP oder ERP-as-a-Service-Systeme (ERPaaS) werden üblicherweise außerhalb des Anwenderunternehmens installiert und dem Kunden als eine Zusammenstellung von Dienstleistungen angeboten. Im Gegensatz zu den On-Premise-ERP-Systemen teilen die Anwenderunternehmen von ERPaaS die von dem System-Anbieter verwaltete Hardware- und Software-Infrastruktur miteinander. Infolge der gemeinsamen Ressourcennutzung entstehen viele Vorteile sowohl für die Anbieter als auch für die Kunden, unter welchen die Kostensenkung und die Realisierung der Wettbewerbsvorteile besonders nennenswert sind (vgl. Duan et al. 2013, S. 1). Das Erreichen dieser Vorteile erfolgt jedoch in der Regel auf Kosten der erweiterten Standardisierung der cloudbasierten ERP-Systeme (vgl. Mijač et al. 2013, S. 133), die einen Einfluss auf die erzielbare Flexibilität haben kann. Die Flexibilität eines Systems ist als die systemeigene interne Eigenschaft zu interpretieren, welche beschreibt, wie gut das System an die Anforderungen des Kunden angepasst werden kann (vgl.

Kassem und Schult 2008, S. 4). Zur Sicherstellung einer entsprechenden Flexibilität muss diese ab den frühen Entwicklungsphasen eingeplant und entsprechend im architektonischen Konzept verankert werden (vgl. Kabbedijk 2014, S. 5).

Unabhängig davon, wie das ERP-System dem Kunden zur Verfügung gestellt wird, sollte es entsprechend flexibel sein, um möglichst gut an die Anforderungen des Kundenunternehmens angepasst werden zu können. Die gezielte Anpassung des Systems an die unternehmerischen Anforderungen wird in der Literatur als *Customizing* bezeichnet (vgl. Kurbel 2008, S. 2). Trotz der Tatsache, dass viele Autoren auf die Wichtigkeit des Customizing im Kontext der cloudbasierten ERP-Systeme aufmerksam gemacht haben, wurde diese Thematik bisher noch unzureichend untersucht. Insbesondere fehlen zurzeit empirische Untersuchungen, welche die Bewertung der Adaptierbarkeit der ERPaaS anhand konkreter Fallstudien validieren bzw. Customizing-Möglichkeiten mehrerer Cloud-ERP-Systeme vergleichen.

Zusätzlich weisen einige Studien auf die Diskrepanzen zwischen Akademikern und Praktikern hinsichtlich der Bewertung der Adaptierbarkeit der cloudbasierten ERP-Systeme hin sowie auf die fehlenden Vergleichs- bzw. Bewertungsrahmen, sodass eine Evaluierung der Anpassungs- und Modifikationsmöglichkeiten der On-Demand-ERP-Systeme bisher nicht möglich ist (vgl. Mijač et al. 2013, S. 139). Die Autoren, welche ihre Untersuchungen auf die Analyse der Einflussfaktoren für die Adaption der cloudbasierten ERP-Systeme fokussieren, nennen in der Regel das Customizing als eine der wichtigsten Herausforderungen der cloudbasierten ERP-Systeme (vgl. Hao et al. 2012, S. 417; Saeed et al. 2012, S. 433). Die begrenzte Adaptierbarkeit und Integrationsfähigkeit werden in der Literatur oft als Schwachstellen des SaaS-Modells bezeichnet (vgl. Duan et al. 2013, S. 8). Während einige Quellen auf die mangelnden Customizing-Möglichkeiten der cloudbasierten ERP-Systeme hindeuten, versichern anderseits die Systemhersteller bzw. -anbieter die umfangreiche Adaptierbarkeit und hohe Flexibilität ihrer Systeme (vgl. Saeed et al. 2012). Die empirischen Studien, die sich auf die Adaptierbarkeit der ERPaaS beziehen, sind immer noch rar. Sie weisen jedoch auf alternative und teilweise innovative Customizing-Konzepte und -Werkzeuge hin, welche oftmals über den von den On-Premise-Systemen bekannten Customizing-Rahmen hinausgehen (vgl. Kurbel und Nowak 2013a; Ortner und Krenn 2016; Chen et al. 2015). Die Evaluation der bestehenden Anpassungs- und Modifikationsmöglichkeiten der cloudbasierten ERP-Systeme ist allerdings trotz hoher Relevanz noch wenig erforscht.

Während die bestehende Forschung unzureichend ist, ist aber auch anzumerken, dass eine korrekte Interpretation der aktuellen Forschungsergebnisse durch das anhaltende Marktwachstum und die Vielfalt der gängigen Service-Betreiber- und Cloud-Bereitstellungsmodelle zusätzlich erschwert wird. Nicht alle auf dem ERP-Markt angebotenen cloudbasierten ERP-Systeme stellen neue, vollständig für das SaaS-Modell konzipierte Lösungen dar. Es sind auch Systeme auf dem Markt verfügbar, welche noch vor einigen Jahren ausschließlich als lokal installierbare Software angeboten wurden und heute sowohl konventionell – on premise – betrieben als auch in einem Cloud-Modell bereitgestellt werden können. Immer häufiger entscheiden sich einige ERP-Anbieter für eine Adaption ihrer ursprünglich für On-Premise-Modell konzipierten Systeme, um von der cloudbasierten Bereitstellung profitieren zu können (vgl. z. B. Johansson und Ruivo 2013, S. 95; Grubisic 2014, S. 67). Die Architekturen dieser ERP-Systeme unterliegen dabei auch einer Weiterentwicklung, um die aus dem SaaS-konformen Betrieb resultierenden Vorteile möglichst weit auszuschöpfen. Die ständige Entwicklung des Marktes und die unzureichende Transparenz der Bereitstellungskonzepte erschwert dabei eine eindeutige Bewertung bzw. Systematisierung der Customizing-Möglichkeiten der cloudbasierten ERP-Systeme.

Der geringe Erkenntnisstand und die hohe Relevanz der Problematik der Adaptierbarkeit von cloudbasierten ERP-Systemen stellen Gründe dafür dar, dass weitere Forschung auf diesem Gebiet notwendig ist. Hierdurch motiviert widmet sich die vorliegende Dissertation einer vergleichenden Analyse der Customizing-Möglichkeiten der cloudbasierten ERP-Systeme und der Erarbeitung eines architekturübergreifenden Systematisierungsrahmens der bestehenden Customizing-Konzepte.

## 1.2 Terminologie

Dieser Abschnitt enthält eine Übersicht der Definitionen und Konzepte, welche in der vorliegenden Dissertation verwendet werden. Besondere Aufmerksamkeit wird dabei auf die Begrifflichkeiten des *Cloud-Computing*, des *cloudbasierten ERP-Systems* sowie des *Customizing* gelegt, welche die Grundlage zum Verständnis des Forschungsfeldes und der angestrebten Forschungsziele bilden.

### 1.2.1 Cloud-Computing

Die Literatur bietet, trotz mehrerer Definitionsversuche (vgl. z. B. Vaquero et al. 2009; Yang und Tate 2012; Buyya et al. 2009), noch keine eindeutige Definition des Cloud-Computing (vgl. Repschläger 2013, S. 18 f.). Im Rahmen dieser Dissertation wird die etablierte und meist zitierte Definition des National Institute of Standards and Technology (NIST) verwendet:

> *„Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.“* (Mell und Grance 2011, S. 2)

Zur genaueren Spezifizierung des Cloud-Computing-Ansatzes und seiner eventuellen Differenzierung von den anderen Ansätzen werden vom NIST fünf Kernmerkmale von Cloud-Computing definiert:

- *On-demand-Selfservice* (Selbstbedienung nach Bedarf): Die benötigten Ressourcen können vom Benutzer eigenständig angefordert werden und werden ihm automatisch (ohne Interaktion mit dem Service-Provider) zugewiesen.
- *Broad Network-Access* (breitbandiger Netzwerkzugang): Die Bereitstellung der Ressourcen erfolgt über das Netzwerk mittels Inanspruchnahme der standardisierten Technologien und Schnittstellen.
- *Resource-Pooling* (Zusammenlegung der Ressourcen in Pools): Von den physikalischen Ressourcen wird abstrahiert. Die einzelnen Hardwareressourcen (z. B. Rechenleistung, Speicherplatz etc.) werden in Pools zusammengelegt und automatisch nach dem aktuellen Bedarf für die verschiedenen Aufgaben der einzelnen Anwender zugewiesen. Der Benutzer kennt die physische Hardware nicht, welche für die Bearbeitung seiner Aufgabe in Anspruch genommen wird.
- *Rapid Elasticity* (sofortige Elastizität): Die Dienste können flexibel auf die Lastschwankungen reagieren, indem die einzelnen Ressourcen zugewiesen bzw. losgelöst werden. Aus der Perspektive des Benutzers scheinen die Ressourcen unbegrenzt zu sein.
- *Measured Service* (gemessene Dienstnutzung): Der Dienst- und Ressourcenverbrauch ist messbar und bildet die Grundlage der Benutzerabrechnung.

Neben den fünf zentralen Eigenschaften kann das Cloud-Computing durch drei Betreibermodelle und vier Bereitstellungsmodelle gekennzeichnet werden. Die Zusammenfassung der Charakteristika von Cloud-Computing wurde in der Abbildung 1.1 vorgenommen.

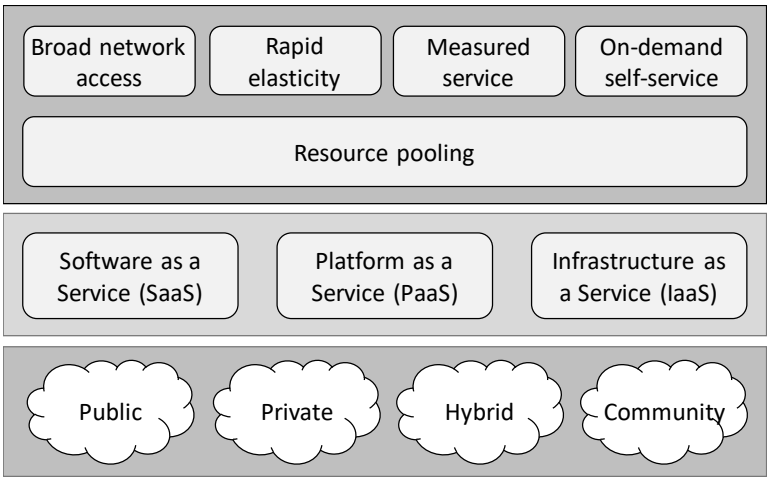



**Abbildung 1.1:** Charakteristika des Cloud-Computing in Anlehnung an Mell und Grace (2011)

Im Rahmen des Cloud-Computing können einem Kunden drei Betreibermodelle zur Verfügung gestellt werden:

- *Software-as-a-Service* (SaaS): Im SaaS-Modell wird die Anwendung von dem Dienstanbieter gehostet und dem Kunden in Form eines Dienstes über das Internet bereitgestellt. Der Kunde erhält einen Zugriff auf die Anwendung. Er kann die zugrunde liegenden infrastrukturellen Ressourcen (z. B. Netzwerk, Server, Datenspeicher und Betriebssysteme) aber nicht verwalten bzw. beeinflussen. In der Regel greift der Anwender auf den Service über einen Webbrowser zu und verfügt über die in der Anwendung enthaltenen Anpassungswerkzeuge.
- *Plattform-as-a-Service* (PaaS): Im PaaS-Modell wird dem Kunden eine Plattform für die Entwicklung, Bereitstellung und Verwaltung von Anwendungen zur Verfügung gestellt. Mithilfe der in der Plattform enthaltenen Werkzeugen und Dienste können die selbst entwickelten bzw. fremdbeschafften Anwendungen in der Cloud-Infrastruktur bereitgestellt werden. In diesem Betreibermodell befindet sich die Anwendung in der Regel unter vollständiger Kontrolle des Kunden, welcher neben der Adaption der Anwendung auch einige Konfigurationen der Hosting-Umgebung durchführen kann. Die zugrunde liegenden Ressourcen der Cloud-Infrastruktur sind weiterhin unter der Kontrolle des Service-Anbieters.
- *Infrastructure-as-a-Service* (IaaS): In diesem Betreibermodell werden dem Anwender die entsprechenden infrastrukturellen Ressourcen zur Verfügung gestellt, welche ihm eine Bereitstellung und Verwaltung der Betriebssysteme und anderer Anwendungen ermöglicht. Der Anwender kann die zugrunde liegende Infrastruktur nicht vollständig verwalten, hat aber Einfluss auf einige ihrer Bestandteile, wie auf installierte Anwendungen, Betriebssysteme, Datenspeicher und gelegentlich auf bestimmte Netzwerkkomponenten (vgl. Mell und Grance 2011).

Zusätzlich zu den drei Betreibermodellen können vier Cloud-Bereitstellungsmodelle voneinander abgegrenzt werden:

- *Private Cloud*: In diesem Cloud-Bereitstellungsmodell wird die Cloud-Infrastruktur ausschließlich einer einzigen Organisation über das Internet zur Verfügung gestellt wird.
- *Public Cloud*: In diesem Cloud-Bereitstellungsmodell werden die infrastrukturellen Ressourcen einer breiten Öffentlichkeit zur gemeinsamen Nutzung über das Internet bereitgestellt. Die einzelnen Anwender werden von dem Dienstanbieter am Ende einer Abrechnungsperiode mit den Kosten für die tatsächlich genutzten Ressourcen belastet (Pay-per-Use-Modell). Der Dienstanbieter ist neben der Verwaltung der Cloud-Infrastruktur auch für die Sicherstellung der Sicherheit und Unabhängigkeit der einzelnen Anwender verantwortlich.
- *Hybrid Cloud*: Dieses Konzept entspricht einer Zusammenstellung von zwei oder mehreren Cloud-Bereitstellungsmodellen, welche in Abhängigkeit von den Anforderungen des Anwenders miteinander kombiniert werden.
- *Community Cloud*: Die Community Cloud ist ein Bereitstellungsmodell, in welchem die Cloud-Infrastruktur einer Gemeinschaft von Organisationen zugänglich gemacht wird, wobei die Organisationen die zum Zweck der Realisierung gemeinsam angestrebter Ziele die bereitgestellten Ressourcen zusammen nutzen (vgl. Mell und Grance 2011; Lenart 2011).

### 1.2.2 Cloudbasierte ERP-Systeme

Die ERP-Systeme in der Cloud realisieren, unter Beibehaltung der Prinzipien des Cloud-Computing, dieselben Aufgaben wie die konventionellen, lokal installierbaren ERP-Systeme und stellen die gleiche Funktionalität zur Verfügung (vgl. Grobman 2008, S. 19; Konstantinidis et al. 2012, S. 37; Kapitel 1.2.1).

Im Gegensatz zu den On-Premise ERP-Systemen, welche in der Regel auf der Seite des Kundenunternehmens installiert werden, werden die cloudbasierten ERP-Systeme auf der Anbieterseite betrieben und mehreren unabhängigen Anwenderunternehmen (sogenannten Mandanten) über das Internet in Form einer Zusammenstellung mehrerer Dienstleistungen zur Verfügung gestellt.

Der Begriff *cloudbasiert* bedeutet in diesem Zusammenhang nichts anderes, als dass dem System ein Cloud-Modell zugrunde liegt, welches als Basis der Dienstleistung verwendet wird (vgl. Hao et al. 2012, S. 410). Erst von den Charakteristika einzelner Cloud-Bereitstellungsmodelle und Betreibermodelle abgeleitet ergeben sich auch die Charakteristika der ERP-bezogenen Dienstleistung.

Die ERP-Systeme, bei welchen die Cloud als Grundlage des Bereitstellungsmodells verwendet wird, werden zunehmend abgekürzt als *Cloud-ERP* bezeichnet bzw. dem Trend Everything-as-a-Service (XaaS) folgend als *ERP-as-a-Service* (ERPaaS) benannt (vgl. Hao et al. 2012, S. 410). Alternativ zu den zwei Stichwörtern wurde die andere populäre Bezeichnung – *On-Demand-ERP-Systeme* – von der Eigenschaft des Cloud-Computing abgeleitet. Der Begriff *on demand* bedeutet im Kontext, dass das System dem Kunden „bei Bedarf“ sofort zur Verfügung steht. Der Systemanbieter übernimmt damit die Verantwortung zur Sicherstellung der entsprechenden Systemverfügbarkeit, um eine bedarfsgerechte Nutzung des cloudbasierten ERP-Systems zu ermöglichen (vgl. Grobman 2008, S. 16). Trotz diverser Stichwörter, welche von den Wissenschaftlern und Praktikern zur Abgrenzung der cloudbasierten ERP-Systeme von den konventionellen Lösungen benutzt werden, ist damit eine ERP-Standardsoftware gemeint, welche von den Eigenschaften des Cloud-Computing profitiert. Im Rahmen dieser Dissertation werden die drei Begriffe Cloud-ERP, ERPaaS sowie On-Demand-ERP gleichgesetzt und synonymisch benutzt.

Das Cloud-Computing-Paradigma initiierte ein neues Segment für die cloudbasierten ERP-Systeme des ERP-Software-Markts, welches sich eines anhaltenden Wachstums erfreut. Die Marktstudien zeigen, dass sich der Umsatz des Cloud-ERP-Sektors in den letzten fünf Jahren verdoppelt hat und prognostizieren eine ähnliche Marktentwicklung für die nächsten Jahre (vgl. Becker et al. 2015, S. 92; Henkes 2016; Hausen 2016). Daneben sind auch immer mehr ERP-Anbieter auf dem Markt zu finden, welche ihr ERP-Portfolio um die cloudbasierten Systeme erweitern (vgl. Eggert et al. 2013; Hamerman et al. 2017; Hedman und Xiao 2016). Die Marktstudien bestätigen auch das systematisch wachsende Interesse an einem cloudbasierten ERP-System bei den Kunden (vgl. KPMG und BITKOM 2016; Kimberling 2016; Bento et al. 2015). Dieses ist auf die aus dem cloudbasierten Betrieb resultierenden Vorteile zurückzuführen, welche für viele – insbesondere kleine und mittelständische Unternehmen – attraktiv wirken. Die wichtigsten Vorteile aus Kundensicht, abgeleitet von den Untersuchungen von Zhao und Kirche (2013), sind niedrige Betriebskosten, räumliche Flexibilität, kurze Implementierungszyklen und automatische Aktualisierungen. Diesen Vorteilen stehen oftmals jedoch die Nachteile gegenüber, unter welchen die mangelnde Adaptierbarkeit und Integrationsfähigkeit sowie die erschwerte Sicherstellung der erforderlichen Datensicherheit besonders ausgeprägt sind (vgl. Duan et al. 2013). In welchem Umfang die konkreten Vorteile bzw. Nachteile zu spüren sind, hängt in der Regel von den Charakteristika des konkreten Systems und von dem Ausmaß ab, in dem es die Prinzipien des Cloud-Computing realisiert (vgl. Scavo et al. 2012).

Aufgrund der Tatsache, dass das ERP-System eine Software darstellt, wird es in der Regel in dem SaaS-Betreibermodell angeboten (vgl. Al-Shardan und Ziani 2015). Uppström et al. (2015) weisen jedoch darauf hin, dass auch die anderen Betreibermodelle, d. h. die PaaS und IaaS, immer häufiger als eine Erweiterung des typischen SaaS-Betreibermodells angeboten werden (vgl. Uppström et al. 2015, S. 4222 f.). Aufgrund der unterschiedlichen Cloud-Bereitstellungsmodelle ergeben sich diverse Bereitstellungskonzepte, in welchen das cloudbasierte ERP-System betrieben werden kann (vgl. Zhao und Kirche 2013, S. 494).

Aufgrund dessen, dass das cloudbasierte ERP-System in unterschiedlichem Umfang die Prinzipien des Cloud-Computing realisieren kann (vgl. Scavo et al. 2012), wird zur Sicherstellung der erforderlichen Vergleichbarkeit eine Differenzierung der cloudbasierten Anwendungen vorgeschlagen. Eine detailliertere Unterteilung der cloudbasierten ERP-Systeme hinsichtlich der Bereitstellungsstrategie bzw. des architektonischen Konzeptes wird im dritten Kapitel vorgenommen.

### 1.2.3 Customizing

Die auf dem Markt verbreiteten On-Demand- und On-Premise-ERP-Systeme stellen Standardsoftware dar (vgl. Mijač et al. 2013, S. 133). Da Standardsoftware für vergleichbar auftretende (und dadurch standardisierbare) Aufgabestellungen mehrerer Unternehmen entwickelt wird, erfüllt sie in der Regel nur in einem bestimmten Umfang die Anforderungen eines konkreten Anwenderunternehmens (vgl. Umble et al. 2003, S. 248). Die Sicherstellung entsprechender Übereinstimmung zwischen dem Unternehmen und der Standardsoftware erfolgt unter anderem mittels Anpassung bzw. Modifikation, welche oft unter dem Oberbegriff *Customizing* zusammengefasst werden.

Das Wort *Customizing* bezeichnet eine gezielte Adaption des ERP-Systems an die individuellen Anforderungen eines Unternehmens (vgl. Kurbel 2016, S. 316). Im Zusammenhang mit der ERP-Standardsoftware bedeutet dies, dass das System nicht in der Standardform durch das Unternehmen übernommen wird, sondern dass die im System enthaltenen Prozesse und Strukturen an die Anforderungen des Anwenderunternehmens angepasst werden. Das Customizing endet in der Regel nicht mit der Einführung des ERP-Systems, sondern es ist auch im Laufe des Systembetriebs weiterhin notwendig.

Die Bedeutung des Customizing wird in vielen Quellen beschrieben (z. B. Haines 2009; Davis 2005; Rothenberger und Srite 2009), welche auch auf die unterschiedlichen Ausmaße der erforderlichen Anpassungen bei Kundenunternehmen hinweisen. Laut Leyh (2011) stellt die Systemanpassung einen integralen Teil nahezu jeder Systemimplementierung dar. Die Marktstudie von Kimberling (2016) zeigt dabei, dass lediglich bei ca. 30 % der Implementierungen kleine Anpassungen des ERP-Systems ausreichend sind. Die überwiegende Mehrheit (ca. 60 %) der Implementierungsprojekte erfordert weitgehende bis sogar vollständige Adaptionen des zu implementierenden Systems. Die cloudbasierten ERP-Systeme müssen, genauso wie die konventionellen On-Premise Gegenstücke, entsprechende Anpassungen bzw. Modifikationen ermöglichen, um an die individuellen Anforderungen mehrerer Kundenunternehmen angepasst werden zu können.

In diesem Zusammenhang wird oft der Begriff der *Adaptierbarkeit* verwendet, welcher beschreibt, wie gut bzw. in welchem Umfang das Customizing im Systemrahmen unterstützt wird (vgl. Jadhav und Sonar 2009, S. 560; Johansson und Sudzina 2008, S. 653).

Customizing kann auf unterschiedlichen Software-Ebenen und mittels unterschiedlicher Ansätze durchgeführt werden. Unter einer *Ebene* wird die architektonische Schicht verstanden. Ein Dreischichtenmodell betrachtend, kann das Customizing entsprechend auf der Daten-, Anwendungs- bzw. Präsentationsschicht vorgenommen werden.

Der *Customizing-Ansatz* soll als eine Methode bzw. ein Konzept verstanden werden, welches in das architektonische Konzept zwecks der Sicherstellung der Adaptierbarkeit des Systems integriert wurde. Die anwendbaren Customizing-Ansätze erstrecken sich vom Setzen der vordefinierten Parameterwerte (Customizing im engeren Sinn) bis hin zur Modifikation des Quellcodes des Systems (Customizing im weiteren Sinn).

Eine detaillierte Beschreibung des Customizing im Kontext der ERP-Systeme unter Einbeziehung der unterschiedlichen Customizing-Aspekte, welche für die nachfolgende Untersuchung von Bedeutung sind, wird im zweiten Kapitel vorgenommen.

## 1.3 Zielsetzung

Die vorliegende Arbeit thematisiert die Problematik der Adaptierbarkeit der cloudbasierten ERP-Systeme. Das Ziel der Untersuchung besteht in der Systematisierung der bestehenden Anpassungs- und Modifikationsmöglichkeiten der cloudbasierten ERP-Systeme sowie in der Bewertung der dafür von den Systemherstellern bzw. -Anbietern bereitgestellten Werkzeuge und Konzepte.

Die Systematisierung der Adaptionsmöglichkeiten soll anhand einer vergleichenden Analyse der bestehenden Customizing-Strategien mehrerer cloudbasierter ERP-Systeme vorgenommen werden, wobei sich die Untersuchung auf die Identifizierung der anwendbaren Customizing-Ansätze und die Ermittlung des adaptierbaren Systemumfangs fokussiert. Parallel dazu soll eine Bewertung der Umsetzbarkeit erfolgskritischer Systemadaptionen anhand vordefinierter Szenarien durchgeführt werden, in deren Rahmen die Effektivität der ausgewählten Customizing-Konzepte und -Werkzeuge validiert werden kann.

Den Untersuchungen des vorliegenden Dissertationsprojektes werden zwei Hauptforschungsfragen (HF) zugrunde gelegt, deren Beantwortung den Einsatz diverser Untersuchungsansätze (vgl. Kapitel 1.4) erfordert.

Die erste Hauptforschungsfrage, welche im Rahmen dieser Ausarbeitung beantwortet werden soll, lautet:

HF 1: *Wie können die cloudbasierten ERP-Systeme adaptiert werden?*

Die Beantwortung dieser Frage erfordert eine vergleichende Analyse der Anpassungskonzepte der auf dem Markt verfügbaren cloudbasierten ERP-Systeme. Das Verständnis über das jeweilige Customizing-Konzept erfordert eine Analyse der bereitgestellten Customizing-Werkzeuge, der ihnen zugrunde liegenden Adaptionsansätze sowie des Umfangs ihrer Anwendung. Daraus ableitend werden drei weitere Forschungsfragen (FF) aufgestellt und diskutiert:

FF 1.1: *Welche Customizing-Ansätze werden zur Sicherstellung der Adaptierbarkeit der cloudbasierten ERP-Systeme eingesetzt?*

In Hinblick auf diese Untersuchungsfrage muss zunächst das Spektrum der potenziellen Customizing-Ansätze ermittelt werden, welches die entsprechende Untersuchungsbasis schafft. Folgend soll untersucht werden, welche dieser Ansätze zwecks der Adaption der cloudbasierten ERP-Systeme angewendet werden können und welche Bedeutung jedem dieser Ansätze bei der Umsetzung der erfolgskritischen Systemadaptionen zugeschrieben wird. Obwohl die einschlägige Literatur hauptsächlich die Parametrisierung als bevorzugte Adaptionstechnik im Cloud-Umfeld nennt, weisen einige Quellen auch auf andere gängige Konzepte hin, welche bei den cloudbasierten Anwendungen verwendbar sind. Die bevorstehende Untersuchung soll eine Antwort auf die gestellte Forschungsfrage liefern und die Bedeutung der einzelnen Adaptionsansätze für das anwenderseitige Customizing aufzeigen.

FF 1.2: *Welche Elemente eines cloudbasierten ERP-Systems können an die Anforderungen des Anwenderunternehmens adaptiert werden?*

Zur Schaffung eines erforderlichen Bezugsrahmens für die Untersuchung müssen zuerst die häufig auftretenden Adaptionsanforderungen ermittelt werden. Darauf folgend soll die Adaptierbarkeit einzelner Systemkomponenten bei mehreren cloudbasierten Systemen, bezogen auf die mit ihnen korrespondierenden Adaptionsanforderungen, validiert und verglichen werden. Durch die Gegenüberstellung der gewonnenen Ergebnisse sollen generalisierbare Erkenntnisse hinsichtlich der Flexibilität einzelner Systemelemente abgeleitet werden und der adaptierbare Systemumfang soll ermittelt werden, welcher im Rahmen eines anwenderseitigen Customizing angepasst werden kann.

FF 1.3: *Welche Customizing-Werkzeuge werden zur Durchführung der Adaptionsprozesse von den Systemanbietern bereitgestellt?*

Die Umsetzung unterschiedlicher Customizing-Strategien macht die Entwicklung und Bereitstellung von unterschiedlichen Customizing-Werkzeugen erforderlich, mit deren Hilfe das cloudbasierte ERP-System an die Anforderungen des Anwenderunternehmens angepasst werden kann. Basierend auf einer Analyse differenzierter Customizing-Konzepte mehrerer Cloud-ERP-Systeme soll die Gegenüberstellung der identifizierten Customizing-Werkzeuge in Abhängigkeit von dem Integrationsgrad mit der Basisanwendung vorgenommen werden (vgl. Weidenhaupt 2001, S. 44). Daneben soll untersucht werden, in welchem Umfang die anwenderseitigen Administratoren die Adaptionen eigenständig im Rahmen des On-Demand-Selfservices vornehmen können bzw. welche Mechanismen der Anwenderunterstützung in den Werkzeug-Konzepten verankert wurden.

Die Antworten auf die gestellten Forschungsfragen (FF 1.1-3) sollen systematisch zur Beantwortung der ersten Hauptforschungsfrage (HF 1) führen und eine Bewertung und Systematisierung der Anpassungs- und Modifikationsmöglichkeiten der cloudbasierten ERP-Systeme ermöglichen.

Die gewonnenen Erkenntnisse hinsichtlich der einzelnen Forschungsfragen sollen weiterhin einen Ausgangspunkt für die anschließende Untersuchung bilden, welche auf die Beantwortung der folgenden Hauptforschungsfrage (HF 2) abzielt:

HF 2: *Wie wirkt sich das architektonische Konzept eines cloudbasierten ERP-Systems auf die bereitgestellte Systemflexibilität aus?*

Zur Beantwortung dieser Forschungsfrage soll bei der Untersuchung eine Differenzierung der cloudbasierten ERP-Systeme anhand ihrer architektonischen Konzepte vorgenommen werden. Der architekturübergreifende Vergleich der Customizing-Strategien unterschiedlicher ERP-Systeme, welche in differenzierten Cloud-Bereitstellungs- und Service-Modellen bereitgestellt werden, ermöglicht es, die systemübergreifenden Tendenzen hinsichtlich der sichergestellten Adaptierbarkeit zu identifizieren und die architekturbedingten Differenzen in dem bereitgestellten Flexibilitätsumfang aufzuzeigen. Allerdings muss in diesem Kontext zunächst die Frage bezüglich der Signifikanz einzelner Zusammenhänge zwischen der Systemarchitektur und der erzielbaren Adaptierbarkeit beantwortet werden. Um die entsprechende Generalisierbarkeit der Erkenntnisse sicherzustellen, sollen die einzelnen Relationen zwischen dem architektonischen Konzept und der Customizing-Strategie statistisch nachgewiesen werden.

Auf Basis der Antworten auf alle der gestellten Forschungsfragen soll dem Ziel der vorliegenden Dissertation nachgekommen werden. Die gewonnenen Erkenntnisse sollen zur eindeutigen Verifizierung der bestehenden Differenzen in der Bewertung der Anpassungs- und der Modifikationsmöglichkeiten der cloudbasierten ERP-Systeme beitragen. Daneben kann die vorgenommene Systematisierung als Vergleichsbasis für die Kunden bzw. Anbieter dienen, anhand derer die Anpassungs- und Modifikationsmöglichkeiten der anderen cloudbasierten ERP-Systeme bewertet werden können.

## 1.4 Methodologie

In der vorliegenden Ausarbeitung wird ein behavioristischer Forschungsansatz verfolgt, der systematisch durch eine konstruktionsorientierte Forschung komplementiert wird. Während der erste Ansatz hauptsächlich auf das Testen von Theorien zur Ermittlung ihres Wahrheitsgehaltes ausgerichtet ist, dient der zweite Ansatz einer Bewertung bzw. praxisbezogenen Validierung neuentwickelter wissenschaftlicher Artefakte (vgl. Hevner et al. 2004, S. 79).

Diese forschungsbedingte Aufteilung entspricht den Untersuchungszielen, welche aus der Konstruktion eines architekturübergreifenden Systematisierungsmodells zur Adaptierbarkeit von cloudbasierten ERP-Systemen und aus der Bewertung der Umsetzbarkeit von einzelnen Systemadaptionen mittels verfügbarer Customizing-Werkzeuge bestehen. Dabei sollen zwei, der Ausarbeitung zugrunde liegenden Fragen beantwortet werden: Wie werden die cloudbasierten ERP-Systeme angepasst und welche Zusammenhänge bestehen zwischen der Systemarchitektur und der bereitgestellten Customizing-Strategie?

Die Untersuchung erfolgt in zwei aufeinanderfolgenden Phasen. Die erste Phase (Kapitel 2 bis 4) widmet sich hauptsächlich der Entwicklung einer wissenschaftlichen Fundierung für die Systematisierung der Customizing-Möglichkeiten der ERPaaS. Darauf aufbauend fokussiert sich die zweite Phase (Kapitel 5 und 1) auf die praxisbezogene Validierung der einzelnen Adaptionskonzepte und die Systematisierung der bereitgestellten Systemflexibilität.

Sowohl für die erste als auch für die zweite Untersuchungsphase kommen unterschiedliche Untersuchungsmethoden zur Anwendung, welche in der Literatur bezüglich des Methodenspektrums der Wirtschaftsinformatik dargestellt werden (vgl. Becker et al. 2003; Wilde und Hess 2007).

Im Rahmen der vorliegenden Dissertation werden sowohl qualitative als auch quantitative Forschungsmethoden verwendet, welche gemäß der angestrebten Forschungsziele einzelner Untersuchungsphasen ausgewählt und miteinander kombiniert werden. Die auf mehreren Methoden basierende Untersuchung wurde gewählt, da der Untersuchungsgegenstand neu ist und einer dynamischen Weiterentwicklung unterliegt. In diesem Kontext kann durch die Anwendung differenzierter Methoden und Sichtweisen auf die Themenstellung eine höhere Validität der Forschungsergebnisse erreicht und systematische Fehler können verringert werden (vgl. Johnson et al. 2007, S. 127 ff.).

Die erste Untersuchungsphase baut auf der qualitativen strukturierten Literaturanalyse als Hauptmethode auf, die zur Erhebung der Sekundärdaten angewendet wird (vgl. Arbnor und Bjerke 2008, S. 180 ff.). Die Sekundärdaten schaffen einen theoretischen Hintergrund für die bevorstehende Untersuchung und helfen, bestimmte Annahmen aufzustellen. Basierend auf einer Literaturauswertung wird der aktuelle Stand der Forschung diskutiert und deduktiv werden empirisch prüfbare Hypothesen bezüglich der Zusammenhänge zwischen der Anpassungsfähigkeit und dem architektonischen Konzept der cloudbasierten ERP-Systeme hergeleitet.

Anschließend werden in der zweiten Untersuchungsphase die Primärdaten gewonnen, welche eine Grundlage für die Verifikation der vom Autor vorgenommenen Annahmen bilden. Zum Zweck einer vielseitigen und tiefgreifenden Analyse der Adaptierbarkeit von Cloud-ERP wird eine intensive explorative Forschung anhand mehrerer cloudbasierten ERP-Systeme vorgenommen.

Im Hinblick auf die angestrebte Validierung und Bewertung der bereitgestellten Systemflexibilität werden im Rahmen des Studiums Elemente der konstruktionsorientierten Forschung eingesetzt. Neue Erkenntnisse werden durch die Entwicklung und Implementierung von neuen Konzepten gewonnen (vgl. Wilde und Hess 2007, S. 281; Kurbel et el. 2011, S. 74). Dieser hier gewählte Ansatz umfasst die Entwicklung von Methoden, mit deren Hilfe die Umsetzungsfähigkeit des zugrunde liegenden Konzeptes belegt wird (vgl. Krcmar 1995; Wilde und Hess 2007, S. 281). In dieser Untersuchung werden diverse Systemadaptionen (als Methode zur Validierung der Systemadaptierbarkeit) konzipiert und ihre Umsetzbarkeit wird im Rahmen differenzierter cloudbasierter ERP-Systeme beschrieben. Es werden die Machbarkeitsstudien definiert, in denen die Umsetzungsfähigkeit konkreter Adaptionsanforderungen validiert wird.

Gleichzeitig zu den Machbarkeitsstudien wird eine empirische Forschung vorgenommen, welche auf einer strukturierten Umfrage als Datenerhebungsmethode aufbaut. Die Auswahl dieser Methode ist dadurch begründet, dass die Untersuchung die Zusammenhänge zwischen der Customizing-Strategie und dem architektonischen Konzept der cloudbasierten ERP-Systeme analysieren und das Vorhandensein eines allgemeingültigen Korrelationsmusters verifizieren soll. Die statistische Auswertung der quantitativen Umfragedaten ermöglicht es zusätzlich, eine Validierung bzw. Verallgemeinerung der Erkenntnisse der qualitativen Untersuchung vorzunehmen, wodurch die Aussagekraft des Gesamtergebnisses zusätzlich erhöht werden kann.

Bei der Zusammenstellung der Ergebnisse der explorativen Untersuchung und der empirischen Forschung wird eine architekturübergreifende Systematisierung der Anpassungs- und Modifikationsmöglichkeiten vorgenommen und ein einheitliches Customizing-Rahmenwerk für die cloudbasierten ERP-Systeme erarbeitet, welches eine adäquate Typologisierung der Customizing-Ansätze bietet und die Potenziale der einzelnen Anpassungskonzepte bzw. Customizing-Werkzeuge zur Durchführung konkreter Systemadaptionen zeigt.

Die genauere Beschreibung der angewendeten Untersuchungsmethoden und des Verlaufs der einzelnen Untersuchungen befindet sich jeweils in dem Kapitel, welches sich der spezifischen Untersuchungsphase widmet.

## 1.5 Aufbau

Die vorliegende Arbeit besteht aus sieben Kapiteln, die systematisch zur Bearbeitung der zugrunde liegenden Problemstellung beitragen. Die Abbildung 1.2 stellt die Struktur der Arbeit dar.

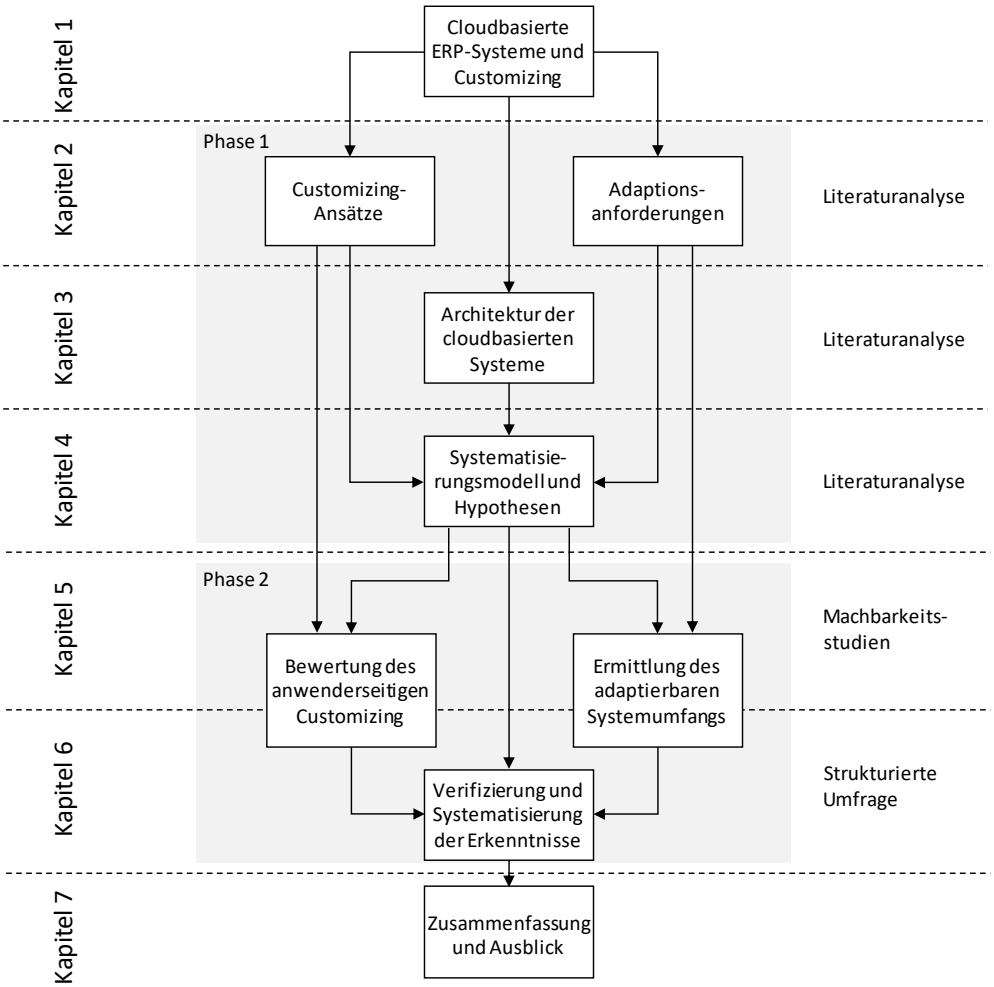



**Abbildung 1.2:** Struktur der Arbeit und Verlauf der Untersuchung

Das erste Kapitel enthält die Einleitung in das Thema des Customizing von cloudbasierten ERP-Systemen und fasst die grundlegenden Konzepte und Definitionen zusammen, die eine konzeptionelle Basis für die Analysen der nachfolgenden Kapitel liefern. Des Weiteren werden in diesem Abschnitt die angestrebten Untersuchungsziele und die für deren Erreichung angewandten Forschungsmethoden beschreiben. Das Kapitel schließt mit der Darstellung des Aufbaus der Dissertation ab.

Das zweite Kapitel fokussiert sich auf die Erarbeitung eines einheitlichen Begriffsrahmens von *Customizing*, sodass eine adäquate Grundlage für die bevorstehenden Untersuchungen und Analysen geschaffen wird. Dabei werden die in der Literatur etablierten Aufteilungen der Customizing-Ansätze gegenübergestellt, die einzelnen Customizing-Ansätze differenziert sowie die häufig auftretenden Adaptionsanforderungen hinsichtlich einzelner Systemkomponenten ermittelt.

Im dritten Abschnitt werden die diversen Bereitstellungskonzepte für das cloudbasierte ERP-System diskutiert und eine Differenzierung der cloudbasierten ERP-Systeme wird vorgenommen. Aus einer vergleichenden Analyse der gut etablierten SaaS-Reifegradmodelle werden die wesentlichen Architekturmerkmale der cloudbasierten Anwendungen abgeleitet, welche einen Einfluss auf die erzielbare Adaptierbarkeit haben können. Anschließend wird die potenzielle Auswirkung der Realisierung einzelner architektonischer SaaS-Merkmale auf die bereitgestellte Systemflexibilität separat analysiert und die identifizierten Zusammenhänge zusammengestellt.

Das vierte Kapitel beginnt mit der Vorstellung des Systematisierungsmodells, welches die für die ganzheitliche und transparente Analyse der Untersuchungsproblematik erforderlichen Betrachtungsebenen zusammenstellt. Im Anschluss wird ein strukturiertes Literaturstudium vorgenommen, welches die Erkenntnisse der bisherigen Forschung entlang der einzelnen Modellebenen erfasst. Basierend auf der intensiven Ergebnisanalyse der durchgeführten Literaturauswertung werden die Hypothesen für die bevorstehende explorative Untersuchung aufgestellt.

Im Rahmen des fünften Kapitels wird eine Vergleichbarkeitsstudie der Adaptierbarkeit diverser cloudbasierter ERP-Systeme vorgenommen. Dieser Abschnitt beginnt mit der Beschreibung des Untersuchungskonzeptes und der Charakteristika der Untersuchungsgegenstände, welche für die explorative Untersuchung ausgewählt wurden. Die nachfolgenden Unterabschnitte beziehen sich auf die Definition der bestimmten Anwendungsszenarien und auf die ausführliche Beschreibung der Machbarkeitsstudien, welche zwecks der Validierung der bereitgestellten Systemflexibilität sowie der Bewertung der Effektivität verfügbarer Customizing-Werkzeuge bzw. -Konzepte vorgenommen wurden. Abschließend werden die Erkenntnisse der Machbarkeitsstudien entlang der einzelnen Betrachtungsebenen des Systematisierungsmodells zusammengestellt und diskutiert.

Der sechste Abschnitt beinhaltet die Beschreibung der durchgeführten empirischen Studie und präsentiert die gewonnenen Erkenntnisse, welche die Ergebnisse des fünften Kapitels ergänzen bzw. validieren. Als Einleitung werden das Umfragekonstrukt und die Methodologie der Studie vorgestellt. In den nachfolgenden Unterabschnitten werden die Ergebnisse der durchgeführten Expertenbefragungen ausgewertet und diskutiert sowie die Verifizierung der aufgestellten Hypothesen vorgenommen. Die Analysen der einzelnen Unterabschnitte tragen systematisch zur Beantwortung der aufgestellten Forschungsfragen bei. Das Kapitel endet mit einer architekturübergreifenden Systematisierung der Anpassungs- und Modifikationsmöglichkeiten der cloudbasierten ERP-Systeme, welche aus den Erkenntnissen der Machbarkeitsstudien und der empirischen Untersuchung abgeleitet wurden.

Das siebte und abschließende Kapitel fasst den Erkenntnisgewinn zusammen und diskutiert die Implikationen für die Forschung und Praxis. Es werden die Limitationen der vorgenommenen Studie beschrieben und die Ansatzpunkte für die Anschlussforschung definiert.

# 2 Customizing von ERP-Standardsoftware

Die auf dem Markt verbreiteten On-Premise- und On-Demand-ERP-Systeme stellen Standardsoftware dar, die zur Realisierung vergleichbar auftretender Aufgabenstellungen mehrerer Unternehmen entwickelt wurde (vgl. Mijač et al. 2013, S. 133). Als standardisierte Software erfüllen diese Systeme in ihrem Lieferzustand die Anforderungen eines konkreten Anwenderunternehmens nur in einem bestimmten Umfang (vgl. Umble et al. 2003, S. 248). Die Sicherstellung fehlender Übereinstimmung zwischen der ERP-Standardsoftware und den unternehmerischen Strukturen und Prozessen erfolgt meist mittels *Customizing*.

Im Hinblick auf die herrschenden Diskrepanzen in der wissenschaftlichen Definition des Customizing-Begriffs widmet sich das vorliegende Kapitel der Erstellung eines einheitlichen begrifflichen Rahmens, welcher den Untersuchungen und Analysen im Rahmen dieser Ausarbeitung zugrunde gelegt wird.

Durch Abgrenzung der einzelnen Customizing-Ansätze, die Analyse der Herausforderungen der Customizing-Prozesse und die Gegenüberstellung der Adaptionsanforderungen wird im Rahmen dieses Kapitels eine Basis erstellt, welche für die korrekte Interpretation der Untersuchungsergebnisse entscheidend ist.

## 2.1 Begriffsrahmen

Das Wort *Customizing* stammt aus dem Englischen und bedeutet eine gezielte Anpassung eines Produktes oder einer Dienstleistung an die individuellen Anforderungen eines Kunden (vgl. Scholze-Stubenrecht 2011; Krajewski et al. 2016).

Die Literaturstudien bieten verschiedene Konzeptualisierungen vom ERP-System-Customizing einschließlich verwandter Begriffe wie *Zuschnitt* (engl. tailoring; Brehm et al. 2001, S. 8017), *Modifikation* (Rothenberger und Srite 2009) und andere Begrifflichkeiten, welche oft als Synonyme verwendet werden.

Die Tabelle 2.1 bietet eine Übersicht über die in der ERP-bezogenen Literatur verbreiteten Customizing-Definitionen bzw. Interpretationen, sowie über die Synonyme und verwandten Begriffe, welche alternativ von den Autoren benutzt werden.

Kurbel (2013) definiert Customizing als einen Prozess, in dem der Zuschnitt des standardisierten ERP-Systems an die individuellen Anforderungen vorgenommen wird (vgl. Kurbel 2013, S. 167). Das „Zuschnitt"-Konzept wird auch von Brehm et al. (2001) verwendet, die unter dem Begriff alle Customizing-Formen zusammenfassen, mit deren Hilfe eine Anpassung der standardisierten ERP-Software an die Anforderungen des Kundenunternehmens erfolgt (vgl. Brehm et al. 2001, S. 8017). Diese prozessorientierte Perspektive ist auch bei anderen Konzepten zu erkennen und wird in der Literatur stark vertreten.

Laut Kanchymalay et al. (2013) stellt Customizing einen Prozess dar, der den Zweck verfolgt, die Ausrichtungsfehler zwischen der ERP-Standardsoftware und den unternehmerischen Strukturen zu beseitigen (vgl. Kanchymalay et al. 2013, S. 1788). Die Modifikation, die gemäß den Autoren mit dem Customizing gleichbedeutend ist, kann die Funktionalität oder andere Aspekte des ERP-Systems betreffen. Im Kontrast dazu wird von Light (2001) ein Konzept angeboten,

welches unter Customizing lediglich die Tätigkeiten versteht, die bestehende Funktionalität eines ERP-Systems zu verändern bzw. zu ergänzen (vgl. Light 2001, S. 417).

**Tabelle 2.1:** Übersicht der Customizing-Definitionen und Interpretationen in der ERP-Literatur

| Customizing-Termini | Definition | Referenz |
|---|---|---|
| Tailoring<br>Modification<br>Configuration | "We use the word *tailoring* to encompass both configuration and modification and a range of options in between." | Brehm et al. 2001, S. 8017 |
| Customization<br>Code change | "[…] Customization is a *code change* put into place because the ERP business process does not mirror the 'desired' business process." | Davis 2005, S. 250 |
| Modification | "[…] ERP customization refers to the *modification* of the ERP package or its functionality […]." | Kanchymalay et al. 2013, S. 1788 |
| Tailoring | "The term 'customizing' describes the process of *tailoring* a standard software system to individual requirements" | Kurbel 2013, S. 167 |
| Change<br>Addition<br>Customisation | "[…] customisation is meant to describe *changes* or *additions* to the functionality available in the standard ERP software." | Light 2001, S. 417 |
| Technical customization | "The primary goal of customization in ERP […] is to achieve a fit between the ERP system and the process that the system supports. […] When the system is customized to fit the process, we refer to this kind of customization as *technical customization*." | Luo und Strong 2004, S. 324 |
| Customization<br>Configuration | "We define customization (or *configuration*) as effort to configure the ERP system using switches/tables provided by the vendor, in order to adapt part of the system to support an organizations preferred business processes, practices and requirements […]" | Ng et al. 2002, S. 88 |
| Code change | "[…] customization is defined as building custom features by using standard programming languages or the ERP system's language, changing the ERP system code, and/or including third-party packages that require some degree of programming to implement." | Rothenberger und Srite 2009, S. 664 |

Eine noch restriktivere Ansicht bezüglich der ERP-Systemanpassung wird von Rothenberger und Srite (2009) sowie Zach und Munkvold (2012) vorgestellt, die die Konfiguration vom Customizing-Begriff ausschließen, weil diese keine tiefgreifenden Systemänderungen ermöglicht (vgl. Rothenberger und Srite 2009, S. 664; Zach und Munkvold 2012, S. 469). Als Customizing werden somit nur Änderungen oder Erweiterungen der vorhandenen System-Funktionalität verstanden.

Im Gegensatz dazu verstehen Ng et al. (2002) unter dem Begriff Customizing nur die Systemanpassungen, die ohne jegliche Modifikation des System-Quellcodes erfolgen. Dabei definiert Customizing vor allem die modulare Systemzusammensetzung und die darauffolgenden Tabellenkonfigurationen (vgl. Ng et al. 2002, S. 88). Zugleich wird von einer auf einem Quellcode basierenden Systemanpassung abgesehen, die über den angenommenen Customizing-Rahmen hinausgeht.

Trotz der Differenzen in den gegenübergestellten Konzeptualisierungen sind die Autoren sich darüber einig, dass das primäre Ziel des Customizing in einer möglichst genauen Anpassung

des ERP-Systems an die Anforderungen eines Unternehmens besteht. Customizing von ERP-Standardsoftware ist notwendig und bei den meisten Unternehmen unabdingbar, um die vorliegenden, unternehmerischen Organisationsstrukturen und Prozesse durch die Software korrekt abzudecken (vgl. z. B. Davenport 1998, S. 125; Wong et al. 2005, S. 499; Mertens et al. 2010, S. 140; Luo und Strong 2004, S. 322).

Im Hinblick auf die Untersuchungsfragen (vgl. Kapitel 1.3) werden die Anpassungskonzepte der cloudbasierten ERP-Standardsysteme im Rahmen dieser Ausarbeitung aus differenzierten Perspektiven betrachtet. Um eine einheitliche begriffliche Grundlage für die Untersuchungen und Analysen der Folgekapitel zu schaffen, wird in der vorliegenden Dissertation die folgende Definition von Customizing aufgestellt:

> *Customizing fasst alle Aktivitäten zusammen, die den Zweck verfolgen, das ERP-System an die Anforderungen des Anwenderunternehmens anzupassen.*

Das Wort *Customizing* wird mit den anderen Begriffen wie *Anpassung*, *Änderung* und *Adaption* gleichgesetzt und als Synonym benutzt.

Die anderen Termini, wie z. B. „Modifikation“ oder „Konfiguration“ bzw. „Erweiterung“, welche eine bestimmte Vorgehensweise bei der Durchführung der Anpassungsprozesse implizieren bzw. eine bestimmte Customizing-Form darstellen, werden im nachfolgenden Kapitel genauer untersucht und voneinander abgegrenzt.

## 2.2 Customizing-Ansätze

Die in der eingeführten Customizing-Definition erwähnten Aktivitäten können unter Inanspruchnahme von unterschiedlichen Customizing-Ansätzen durchgeführt werden, deren Einsetzbarkeit von dem anzupassenden Systemelement bzw. dem vorgesehenen Anpassungsumfang abhängig ist.

Das Customizing kann in Abhängigkeit von der Systemarchitektur auf unterschiedlichen architektonischen Ebenen und in unterschiedlichem Ausmaß vorgenommen werden. Gerade in der Systementwurfsphase werden die einzelnen Customizing-Ansätze von dem Systemhersteller gezielt ausgewählt und in einer kohärenten Customizing-Strategie zusammengestellt (vgl. Dittrich et al. 2009). Die Customizing-Strategie definiert, in welchem Umfang und mithilfe welcher Customizing-Ansätze die einzelnen Systemelemente adaptiert werden können (vgl. Sun et al. 2008, S. 22 f.).

Im Rahmen der aufgenommenen Customizing-Definition wird im Folgenden eine genauere Unterteilung der Customizing-Ansätze vorgenommen, die in der Literatur bezüglich der Anpassung von ERP-Standardsoftware am meisten vertreten sind.

Um einen Überblick über die in der Literatur verbreitete Aufteilung der Customizing-Ansätze zu schaffen, wurde eine tabellarische Darstellung vorgenommen (vgl. Tabelle 2.2), die eine bessere Analyse der Gemeinsamkeiten und Differenzen der Typologien ermöglicht.

Da im Rahmen dieser Ausarbeitung, insbesondere bei der praxisorientierten Untersuchung, eine entsprechende Detaillierung und Abgrenzung der einzelnen Konzepte und Customizing-Ansätze erforderlich ist, werden die von Luo und Strong (2004), Rothenberger und Srite (2009) sowie Haines (2009) definierten Rahmenwerke aufgrund ihres hohen Generalisierungsgrades lediglich zu einem gewissen Ausmaß einsetzbar.

**Tabelle 2.2:** Customizing-Typologien im Überblick

| Typologie | Customizing-Ansatz | Beschreibung |
|---|---|---|
| Brehm et al. (2001) | Configuration | „Setting of parameters to choose between different executions of processes and functions“ |
| | Bolt-Ons | „Implementation of third-party package designed to work with ERP system and provide industry-specific functionality“ |
| | Screen masks | „Creating new screen masks for data in- and output“ |
| | Extended reporting | „Programming of extended data output and reporting options“ |
| | Workflow programming | „Creating of non-standard workflows“ |
| | User exits | „Programming of additional software code in an open interface“ |
| | ERP programming | „Programming of additional applications, without changing the source code (in vendor’s computer language)“ |
| | Interface development | „Programming of interfaces to legacy systems or thirdparty products” |
| | Package code modification | „Changing the source-codes ranging from small changes to change whole modules“ |
| Luo und Strong (2004) | Module selection | „In module selection, companies choose to implement one or more modules using the default configuration set by ERP vendors.“ |
| | Table customization | „[...] select(ing) configuration options in the tables so that the system fits organizational needs“ |
| | Code customization | „[...] the source code of the ERP system is changed, the functionality is augmented, or a new interface is developed [...]“ |
| Haines (2009) | Configuration | „Configuration activities [...] usually amount to changing entries in tables or configuration files [...].“ |
| | Extension | „[...] ES vendors [...] allow extension of their systems by supporting common interfaces (also known as user-exits) [...].“ |
| | Modification | „Modification is an alteration that is usually not supported by the vendor. This includes code changes and other more invasive alterations.“ |
| Kurbel (2016) | Parametrisierung | „[...] die Anpassung des Systems mit Hilfe von Einstellungen [...]“ |
| | User Exits | „User Exits [...] sind vordefinierte Stellen im Programmcode, an denen externe Programme aufgerufen werden können.“ |
| | Application Programming Interfaces (APIs) | „Über APIs können von Programmen aus, die das Anwenderunternehmen individuell entwickelt, die vom Systemhersteller vorgefertigten Module aufgerufen werden.“ |
| | Individualentwicklung | „Individualentwicklung außerhalb des ERP-Systems ist ein Weg, für Probleme, die das ERP-System nicht abdeckt, eigene Lösungen zu erstellen.“ |
| | Modellbasierte Generierung | „Generierbare Systeme zeichnen sich dadurch aus, dass ein Informationsmodell zugrunde liegt, auf dessen Basis die Systemkomponenten automatisch erzeugt werden.“ |
| | Componentware | „[...] Verwendung von Komponenten zur Erzeugung eines Informationssystems [...]“ |
| | Änderungen des Programmcode | „Änderungen direkt in den Programmen des ERP-Systems [...]“ |
| Rothenberger und Srite (2009) | Configuration/Selection | „Modification of an ERP software package [...] by selecting appropriate system components and by setting parameters [...].“ |
| | System change | „[...] the ERP system code (is) modified to fit an organization’s needs.“ |
| | Bolt-Ons | „[...] additional program code development, but it does not require a modification of existing system code.“ |

Das von Brehm et al. (2001) entwickelte Rahmenwerk schlägt eine sehr detaillierte Aufteilung der Customizing-Ansätze vor – ausgehend von einer einfachen Konfiguration bis hin zur tiefgreifenden Modifikation des Quellcodes (vgl. Brehm et al. 2001, S. 8020 ff.). Es kann jedoch der Tabelle 2.2 leicht entnommen werden, dass Brehm und Koautoren im Vergleich zu den anderen Wissenschaftlern eine funktionsorientierte Typologie vorschlagen, welche teilweise die Systemanpassungen mit den dazu verwendeten Maßnahmen-Konzepten kombiniert. Die anderen Autoren fokussieren sich dagegen hauptsächlich auf die Ansätze, mit denen das Customizing der Standardsoftware realisiert werden kann. Des Weiteren werden in den aktuellen Studien bezüglich der Anpassbarkeit von ERP-Systemen einige der von Brehm et al. (2001) postulierten Anpassungsansätzen als nicht mehr aktuell angesehen (vgl. Uppström et al. 2015, S. 4225; Usman et al. 2014, S. 4). Basierend auf zahlreichen Experteninterviews wurde von Üppström et al. (2015) festgestellt, dass aufgrund eines technologischen Wandels einige Anpassungskonzepte – vor allem die Entwicklung von dedizierten Schnittstellen, Bildschirmmasken bzw. die Programmierung der Geschäftsflüsse – an Aktualität verloren haben (vgl. Uppström et al. 2015, S. 4225). Daneben fehlen dem Rahmenwerk von Brehm die Ansätze, die insbesondere bei der Anpassung der cloudbasierten ERP-Systeme eine Rolle spielen könnten, d. h. die komponentenbezogene Systemzusammenstellung und die modellgetriebene Codegenerierung.

Diese Lücken der bisherigen Rahmenwerke werden in der von Kurbel (2016) angebotenen Typologie gefüllt. Diese Typologie beinhaltet sowohl die altbewährten Ansätze als auch die innovativeren Adaptionskonzepte, welche insbesondere bei den web- und cloudbasierten Architekturen eine größere Rolle spielen können. Aus diesem Grund wird im Rahmen der vorliegenden Arbeit eine Aufteilung der Customizing-Ansätze in Anlehnung an Kurbel verwendet. Um den unterschiedlichen Interpretationen der einzelnen Customizing-Konzepte entgegenzuwirken, werden sie in den nachfolgenden Unterabschnitten näher dargestellt.

### 2.2.1 Komposition

Die *Komposition* bezeichnet die Zusammenstellung einer ERP-Standardsoftware aus kleineren Komponenten, die ausgewählt und miteinander verknüpft werden müssen (vgl. Hahn 2006, S. 29). Die Grundlage dieses Ansatzes bilden die modulare Struktur des Systems und die eindeutig definierten Schnittstellen der einzelnen Komponenten, welche die Verbindung der Systembausteine ermöglichen (vgl. Turowski 2002, S. 1).

Als Softwarekomponente kann eine Softwareeinheit verstanden werden, die eine bestimmte Funktionalität bereitstellt und über Schnittstellen verfügt, welche die Wiederverwendbarkeit der Komponente sicherstellen. Obwohl eine Softwarekomponente auf unterschiedlichen Abstraktionsebenen definiert werden kann (vgl. Kurbel 2016, S. 320), wird der Begriff im Rahmen dieser Ausarbeitung immer mit einem Bezug auf einen bestimmten Webdienst (als eine über das Internet bereitgestellte Funktionalität; vgl. W3C 2017b) oder auf ein bestimmtes Softwaremodul (Zusammenstellung mehrerer Webdienste) verwendet.

Die Bedeutung der komponentenbasierten Systemzusammenstellung, als einer Vorgehensweise des Customizing, wächst mit steigendem Einsatz der serviceorientierten Softwarearchitektur (SOA) und cloudbasierten Softwarelösungen (vgl. Hufgard und Krüger 2012, S. 29 f.). Gemäß dem SOA-Paradigma stellen die Systeme keine monolithischen Strukturen dar, sondern bestehen aus losgekoppelten Webdiensten, die zur Realisierung einer bestimmten Aufgabe zusammengestellt werden können (vgl. Kurbel 2008, S. 105 f.).

Abbildung 2.1 bietet eine Veranschaulichung der komponentenbasierten Zusammenstellung eines ERP-Systems. Das Unternehmen stellt sich das System aus den Komponenten (Modulen) zusammen, die sich im Angebot des Systemherstellers befinden. Dabei konzentriert sich das Unternehmen lediglich auf die Komponenten, welche für die Abbildung seiner Geschäfts-

prozesse entscheidend sind (vgl. Davenport 1998, S. 125). Können die unternehmerischen Prozesse im Rahmen des vom Systemherstellers bereitgestellten Komponentenumfangs nicht komplett abgedeckt werden, muss die Systemzusammenstellung durch die unternehmensdedizierten Komponenten ergänzt werden, die sich entweder im Angebot der Drittanbieter befinden (Branchenlösungen und Partnerlösungen) oder eigenentwickelt werden müssen.

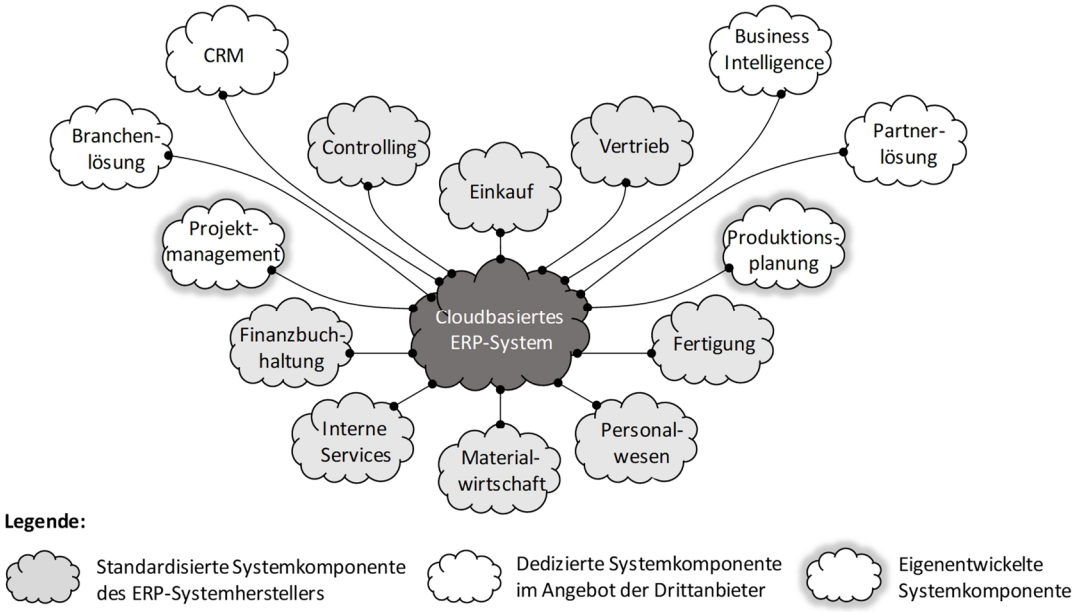



**Abbildung 2.1:** Visualisierung der Zusammenstellung eines ERP-Systems in Anlehnung an Frick (2008)

Anschließend werden die Zusammenhänge zwischen den ausgewählten Systembestandteilen auf Basis der vorgegebenen Komponenten-Schnittstellen definiert, um eine erforderliche Funktionsintegration des ERP-Systems sicherzustellen.

Werden in dem unternehmerischen Lebenszyklus zusätzliche Komponenten benötigt, können diese in der Regel auch in der Betriebsphase in die bestehende Systemzusammenstellung nachträglich integriert werden (vgl. Beheshti 2006, S. 191). Wird nach der System-Inbetriebnahme eine Änderung der bestehenden Systemkomposition erforderlich, können die individuellen Komponenten ausgetauscht werden.

### 2.2.2 Parametrisierung

*Parametrisierung* stellt einen Customizing-Ansatz dar, bei dem die Anpassung der Komponenten eines ERP-Systems durch das Setzen von *Parametern* erfolgt. Ein *Parameter* in der ERP-Systemanpassung bezeichnet ein Datenfeld bzw. Attribut, welches innerhalb einer Systemkomponente zwecks geplanter Anpassung von deren Struktur oder Verhalten vom Systemhersteller vordefiniert wurde (vgl. Ackermann und Turowski 2006, S. 344). Da die Parameter oft in einer tabellarischen Form abgespeichert werden (vgl. Brehm et al. 2001, S. 8018; Ackermann und Turowski 2006, S. 343), wird dieser Ansatz in der Literatur auch als tabellenbasiertes Customizing bzw. tabellarische Konfiguration bezeichnet (vgl. Luo und Strong 2004, S. 324).

Da die standardisierten ERP-Systeme für eine Vielzahl von Unternehmen konzipiert werden, welche durch unterschiedliche Strukturen und Prozesse gekennzeichnet sind, enthalten sie meistens mehr Funktionen, als das einzelne Unternehmen für die Realisierung seiner Kernaufgaben benötigt (vgl. Sekatzek und Krcmar 2009, S. 237). Damit das Anwenderunternehmen

sich den erforderlichen Funktionsumfang aus der zur Auswahl stehenden Optionen zusammenstellen kann, werden durch den Systemhersteller die Parameter definiert.

Die Parametrisierung charakterisiert sich durch das Zusammenstellen eines erforderlichen Funktionsumfangs bzw. die Festlegung eines gewünschten Programmablaufs mittels Zuweisung der ausgewählten, vordefinierten Parameterwerte zu den System-Parametern (vgl. Kurbel 2016, S. 317). Die Parameter können dabei eine komponenteninterne bzw. -übergreifende Auswirkung haben (vgl. Ackermann und Turowski 2006, S. 350).

Die Parametrisierung stellt einen vergleichsweise wenig aufwendigen Customizing-Ansatz dar, da die gewünschte Funktionalität im System enthalten ist und lediglich durch die Auswahl der zulässigen Parameterwerte aktiviert bzw. konfiguriert werden muss. Das Customizing mittels Parametrisierung kann somit ohne jegliche Modifikation bzw. Erweiterung der bestehenden Komponenten erfolgen, welche normalerweise entsprechende Programmierkenntnisse voraussetzen würden (siehe Kapitel 2.2.5 bzw. 2.2.7). Die vorgenommene parameterbasierte Systemkonfiguration lässt sich – gemäß der geänderten unternehmerischen Anforderungen der Betriebsphase – durch Vergabe neuer Parameterwerte ändern (vgl. Ackermann und Turowski 2006, S. 343).

In der Praxis sind jedoch auch signifikante Schwachstellen dieses Ansatzes sowohl aus der Systemhersteller- als auch der Systemanwendersicht zu erkennen. Vor allem können bei der Konfiguration lediglich jene Einstellungen vorgenommen sind, die vom Systemhersteller explizit vorgesehen werden. Erhöht der Systemhersteller die Anzahl der wählbaren Funktionen durch Hinzufügen weiterer Parameter, so steigt meist neben dem Systemumfang auch die Systemkomplexität. Aufgrund der hohen Anzahl unterschiedlicher Parameter und der Wechselbeziehungen zwischen einzelnen Parametern steigt die Komplexität der Parametrisierung, da die Übersichtlichkeit aller Ursache-Wirkungs-Beziehungen stark beeinträchtigt wird (vgl. Ackermann und Turowski 2006, S. 350; Brehm et al. 2001, S. 8018; Kurbel 2016, S. 317 f.).

### 2.2.3 User-Exits

Neben den Parametern, die als Datenfelder bzw. Attribute definiert werden, können von dem Systemhersteller sogenannte ausführbare Parameter zur Anpassung der Komponenten eingesetzt werden. Bei diesem Customizing-Ansatz stellt der Parameter eine Schnittstelle dar, welche als Input eine ausführbare Programmeinheit (meistens einen Quellcode) erwartet (vgl. Ackermann und Turowski 2006, S. 343). Diese Schnittstellen werden in der Literatur als *User-Exits* bezeichnet (vgl. Haines 2009, S. 184; Brehm et al. 2001, S. 8020).

Im Allgemeinen kann ein User-Exit als eine Stelle im Quellcode (sogenannter Platzhalter) einer Komponente verstanden werden, über welche benutzerspezifische Prozeduren angebunden und in der Programmlaufzeit aufgerufen werden können (vgl. Kurbel 2008, S. 450). In Abhängigkeit von der durch den Systemhersteller verfolgten Strategie, können dem Anwender im Rahmen eines User-Exits die Möglichkeit einer Integration individueller Programmeinheiten bzw. eine Modifikation der bestehenden, standardisierten Codestrukturen bereitgestellt werden (vgl. SAP 2006).

Die User-Exits werden durch den Hersteller in Komponenten verankert, bei denen (geprägt durch die unternehmerische Vielfalt) eine sehr große Variabilität erforderlich ist, so dass die Berücksichtigung aller potenziell zu erwartenden Lösungskonzepte im Rahmen einer Parametrisierungsstrategie nicht möglich ist bzw. sich nicht mehr lohnt (vgl. Kurbel 2016, S. 318).

Die Inanspruchnahme der vordefinierten User-Exits erfordert von dem Anwender neben den entsprechenden Programmierkenntnissen auch ein Verständnis der vordefinierten, anwendba-

ren Schnittstellenstrukturen. Aus diesem Grund muss neben dem User-Exit auch eine technische Dokumentation vorliegen.

Die mittels User-Exits vorgenommenen Erweiterungen bzw. Änderungen müssen entsprechend getestet werden, bevor sie im ERP-System implementiert werden können. Die über User-Exits angebundenen Programmteile bzw. Datenobjekte können auch nach der System-Inbetriebnahme modifiziert werden. Im Vergleich zu den zwei ursprünglich beschriebenen Anpassungsansätzen ist Customizing mittels User-Exits mit größerem Aufwand verbunden, der sowohl durch die Implementierung als auch durch die Wartung und Instandhaltung verursacht werden kann.

### 2.2.4 Programmierschnittstellen

Die *Programmierschnittstellen* stellen einen Customizing-Ansatz dar, welcher es den externen Softwareentwicklern ermöglicht, systemdedizierte Komponenten zu entwickeln, welche die von dem Systemhersteller vordefinierten Systemkomponenten auf alternative Art und Weise wiederverwenden (vgl. Kurbel 2016, S. 318).

Die Programmierschnittstellen – oft abgekürzt *APIs* (*Application Programing Interfaces*) genannt – erlauben es, die unternehmensangepassten Systemerweiterungen unabhängig von dem Systemhersteller zu entwickeln. Aufgrund der Wiederverwendbarkeit der bereits existierenden standardisierten Komponenten sind die neuen Entwicklungen standardkonform und können in das ERP-System integriert werden.

Die Bereitstellung einer Programmierschnittstelle ist stark mit der Geschäftsstrategie des Systemherstellers verbunden. In der Regel werden die APIs lediglich den von dem Systemhersteller zertifizierten Softwareentwicklern gegen Zahlung einer Entwicklerlizenz zur Verfügung gestellt. Damit sichern die Systemhersteller die Qualität der Systemerweiterungen und die Kompatibilität der Entwicklungen ab. Bei nichtkommerziellen ERP-Systemen (z. B. Odoo; Odoo S. A. 2016a) werden die APIs, sofern diese vorhanden sind, frei verfügbar und über das Internet bereitgestellt.

Neben der Bereitstellungsstrategie stellen der Umfang und die Stabilität einer API die Faktoren dar, welche für die Anwendbarkeit dieses Customizing-Ansatzes entscheidend sein können. Durch den Umfang einer API legt der Systemhersteller fest, welche Komponenten bzw. Funktionalitäten oder Strukturen im Rahmen des Customizing wiederverwendet werden können. Die API-Stabilität ist ein Garant für die Kompatibilität der Neuentwicklung mit den künftigen Versionen des Standardsystems und bestimmt, ob oder wie häufig eine Instandhaltung bzw. Umprogrammierung der mittels APIs vorgenommenen Entwicklungen erforderlich ist. Eine umfangreiche, gut dokumentierte und stabile Programmierschnittstelle stellt eine geeignete Basis für die unternehmensindividuelle Erweiterung des standardisierten ERP-Systems dar.

Die APIs gewinnen durch steigende Popularität der serviceorientierten Softwarearchitektur an Bedeutung, in deren Rahmen durch lose Kopplung unabhängiger Webdienste eine neue Funktionalität entsteht (vgl. Iqbal et al. 2012, S. 390).

### 2.2.5 Systemerweiterungen

Eine *Erweiterung* wird in der ERP-Literatur als eine externe Komponente definiert, die – nach einer erfolgreichen Integration in das ERP-Standardsystem – eine zusätzliche, im Standard nicht enthaltene Funktionalität bereitstellt (vgl. Kurbel 2016, S. 319). Neben dem Begriff Erweiterung werden oft auch andere Bezeichnungen, wie Add-on, Plug-in oder Bolt-on (vgl. Zach und Munkvold 2012, S. 463; Brehm et al. 2001, S. 8020) synonymisch verwendet.

Ohne Erweiterungsprogrammierung können die standardisierten ERP-Systeme, trotz komponentenbasierter Systemstruktur und großer Anzahl an wählbare Parameter, an die individuellen Anforderungen der Anwenderunternehmen oftmals nicht genauer angepasst werden, als das im Fall der individuell an das Anwenderunternehmen zugeschnittenen Lösungen möglich ist (vgl. Mertens et al. 2010, S. 141).

Die Systemhersteller bieten zu dem standardisierten ERP-System zusätzlich auch die dedizierten Entwicklertools bzw. -umgebungen an, welche die Programmierung individueller Softwarekomponenten und ihre Integration in das ERP-System vereinfachen (vgl. Luo und Strong 2004, S. 324 f.).

Die durch externe Programmierer entwickelten systemdedizierten Erweiterungen werden oft als komplementäre Software zu den ERP-Systemen auf dem Markt angeboten. Die Entwicklung und Vermarktung der ergänzenden Systemkomponenten bzw. Funktionalitäten bringt positive Effekte mit sich, nicht nur für die Entwickler, sondern auch für den Systemhersteller (vgl. Sarker et al. 2012, S. 321). Durch die Arbeit der unabhängigen Softwareentwickler können die funktionalen Mängel eines Standardsystems in konkreten unternehmerischen Bereichen abgedeckt werden. Aus diesem Grund wird die Entwicklung der Erweiterungen durch die Anbieter der standardisierten Systeme im Rahmen der Zertifizierungsprogramme häufig unterstützt (vgl. Magnusson und Nilsson 2013, S. 227 ff.).

Die individuell entwickelten Systemkomponenten können nicht nur als reine Erweiterung des Funktionalitätsumfangs in die Standardsoftware eingebettet werden, sondern auch zur Veränderung der vordefinierten Systemabläufe, Strukturen bzw. Geschäftsdatenobjekte eingesetzt werden.

### 2.2.6 Modellbasierte Generierung

Die *modellbasierte Generierung* stellt einen Customizing-Ansatz dar, bei dem die Adaption der Komponente durch Änderung der Komponentenspezifikation erfolgt, anhand welcher ein adaptierter Komponentencode automatisch generiert wird.

Die Voraussetzung für die Anwendung dieses Ansatzes ist ein *generierbares System*, welches nach den Prinzipien der modellgetriebenen Softwareentwicklung (vgl. Stahl et al. 2007) entwickelt wurde. Das System bzw. die Systemkomponente wird als generierbar bezeichnet, wenn dessen Code durch die Zusammenstellung der wiederverwendbaren Bausteine und durch eine automatische Generierung erzeugt wird.

Die Grundlage einer generierbaren Systemkomponente stellt ein Informationsmodell dar, das die Komponente in einer formellen Notation beschreibt und auf dessen Basis eine automatische Generierung der Komponente erfolgen kann (vgl. Stahl et al. 2007, S. 11 ff.). In Abhängigkeit von dem durch den Systemhersteller verfolgten Konzept können die Modelle zur Abbildung und Erzeugung der Datenstrukturen, Funktionen und Geschäftsabläufe eingesetzt werden (vgl. Kurbel 2016, S. 319).

Dabei werden für die unterschiedlichen Ebenen unterschiedliche Modellierungstechniken verwendet. Während die Datenmodelle als Entity-Relationship-Modelle entwickelt werden, werden die Geschäftsabläufe unter Einsatz einer prozessorientierten Modellierungstechnik entworfen.

Die Anpassungsprozesse erfolgen durch eine anwenderkonforme Adaption des formalen Modells, welches der zu adaptierenden Struktur bzw. Funktionalität zugrunde liegt. Das Customizing wird dabei durch entsprechende Modellierungswerkzeuge unterstützt, welche die Umsetzung einer Modellveränderung vereinfachen (vgl. Stahl et al. 2007, S. 23). Da die Modelle in einer formalen Modellierungstechnik (unter Einhaltung der vorgegebenen Notationsregel)

erstellt werden, erfordert der modellbasierte Customizing-Ansatz keine Programmierkenntnisse und kann im Rahmen einer anwenderseitigen Systemadaption eingesetzt werden.

Die vorgenommenen Änderungen werden im Repository – einem zur Speicherung und zur Verwaltung der Modelle dediziertem Verzeichnis – gespeichert. Von dort werden die in der Systemlaufzeit benötigten Komponenten durch die automatische, modellbasierte Erzeugung ihrer Codes generiert (vgl. Kurbel 2013, S. 169).

Der modellgetriebene Anpassungsansatz erleichtert durch die hohe Transparenz der Komponentenmodelle die Durchführung von Anpassungen und trägt zur Reduktion der mit der Durchführung von Customizing verbundenen Fehlerquote bei.

### 2.2.7 Codebasierte Modifikation

Einige Systemhersteller erlauben es ihren Kunden mithilfe der bereitgestellten Entwicklungsumgebung, neben einer Erweiterung auch eine Modifikation der bestehenden Systemkomponenten vorzunehmen. Im Rahmen dieser Ausarbeitung stellt die *codebasierte Modifikation* einen Customizing-Ansatz dar, bei welchem die Adaption des ERP-Systems an die unternehmerischen Gegebenheiten durch eine Veränderung des Quellcodes erfolgt. Als Quellcode wird ein Text in der dem System zugrunde gelegten Programmiersprache verstanden.

Zwecks der Systemmodifikation benötigen die Programmierer einen Zugriff auf den Quellcode des ERP-Systems und auf die entsprechenden Entwickler-Berechtigungen. Die Modifikation der standardisierten Systemkomponenten stellt eine komplexe und aufwendige Aufgabe dar (vgl. Zach und Munkvold 2012, S. 469). Zwecks der korrekten Interpretation der zu adaptierenden Codestrukturen muss der Entwickler neben den erforderlichen Programmierkenntnissen auch ein systembezogenes Know-how und Verständnis der Systemarchitektur besitzen.

Bei einer codebasierten Modifikation der standardisierten Systemkomponente besteht die Gefahr, dass die vorgenommene Veränderung des Quellcodes einige unerwünschte Nebeneffekte in den verbundenen Systemelementen verursachen kann. Aus diesem Grund werden die Modifikationen zuerst an einem Test- bzw. Referenzsystem vorgenommen, und erst nach Beseitigung aller Fehler in das produktive ERP-System des Kundenunternehmens implementiert (vgl. Davis 2005, S. 250). Dabei ist zu beachten, dass jede Fehlerstelle die erfolgreiche Inbetriebnahme des Systems verhindert und Verzögerungen der Systemeinführung verursachen kann. Die Literatur nennt zahlreiche Beispiele für einen Misserfolg der ERP-Implementierungsprojekte, welcher durch eine zu tiefgreifende Modifikation der Standardkomponenten verursacht wurde (vgl. Zach und Munkvold 2012, S. 466; Wong et al. 2005, S. 500). Die mit der Modifikation des Standards verbundenen Probleme enden meistens nicht nach der Implementierung des ERP-Systems, sondern setzen sich auch während der Betriebsphase fort (mehr dazu in Kapitel 2.3.3). Im Hinblick auf diese Erfahrungen empfehlen die Systemhersteller und externen Berater, die Modifikation des ERP-Standardsystems zu vermeiden bzw. auf das absolute Minimum zu reduzieren (vgl. Kurbel 2016, S. 317).

## 2.3 Customizing im ERP-Lebenszyklus

Der Lebenszyklus eines ERP-Systems umfasst die zeitliche Entwicklung ausgehend von der Entscheidung über Eigenerstellung oder Beschaffung des Systems, über den Entwurf, die Inbetriebnahme und die nachfolgende Wartung bzw. Weiterentwicklung bis hin zur Abschaffung (vgl. Krcmar 2015, S. 203). Der Lebenszyklus einer Standardsoftware kann mithilfe eines Lebenszyklusmodells beschrieben werden, welches ermöglicht, die ERP-Systeme in Abhängigkeit von ihrer aktuellen Entwicklungsphase zu betrachten und die Aktivitäten und Aufgaben entsprechend der Lebenszyklusphasen zu definieren.

Obwohl unterschiedliche Lebenszyklusmodelle für die Anwendungssoftware entwickelt wurden, die teilweise unterschiedliche Phaseneinteilungen vorschlagen (vgl. Ross und Vitale 2000, S. 235; Esteves und Pastor 1999, S. 362 f.; Markus und Tanis 2000, S. 190 ff.), kann der Lebenszyklus eines ERP-Systems im Hinblick auf das Customizing grundsätzlich in zwei Phasen – Implementierungsphase und Postimplementierungsphase – unterteilt werden (vgl. Willis und Willis-Brown 2002, S. 35; Brehm et al. 2001, S. 8024).

Die beiden Phasen werden in der ERP-bezogenen Literatur oft auch unterschiedlich bezeichnet – die Implementierungsphase wird oft als *Einführungsphase* bzw. *vor Going-Live* genannt, für die Postimplementierungsphase werden *Betriebsphase* und *nach Going-Live* als Bezeichnungen verwendet (vgl. Zach und Munkvold 2012).

### 2.3.1 Customizing bei der Systemimplementierung

Customizing-Aufgaben bilden einen integralen Bestandteil der ERP-Systemeinführung, der sicherstellt, dass die unternehmerischen Anforderungen in dem ERP-System abgebildet werden. Die Entscheidung über den Umfang der Systemanpassungen und die Auswahl der geeigneten Customizing-Ansätze stellen wichtige Faktoren der Implementierungsphase dar, die den gesamten Systemlebenszyklus betreffen (vgl. Brehm et al. 2001, S. 8024).

Die komponentenbasierte Systemzusammenstellung und die nachfolgende Parametrisierung stellen die Customizing-Ansätze dar, welche im Rahmen der Systemeinführung bevorzugt werden. Kann das ERP-System mithilfe dieser Ansätze nicht vollständig an die unternehmerischen Anforderungen angepasst werden, kommen die anderen Customizing-Ansätze zum Einsatz. Dabei ist jedoch zu beachten, dass zu viel Individualisierung in Form von Eigenentwicklung, Erweiterung und Modifikation auch mit negativen Effekten verbunden sein kann.

Die Systemadaptionen können sowohl zur Verzögerung der Inbetriebnahme des ERP-Systems führen als auch die Überschreitung des für die Implementierung vorgesehenen Budgets zur Folge haben. Die codebasierten Adaptionen erfordern in der Regel während der Betriebsphase eine zusätzliche Wartung und Instandhaltung, welche eine Steigerung der System-Betriebskosten zur Folge haben (vgl. Ng et al. 2003, S. 1; Haines 2009, S. 182 f.). Durch die mangelnde Qualität der Eigenentwicklungen, ungelöste Codefehler und unzureichende Tests der vorgenommenen Systemadaptionen kann das System unzuverlässig werden (vgl. Brehm et al. 2001, S. 8022; Wong et al. 2005, S. 500; Davis 2005, S. 250; Keckeis et al. 2016).

Im Hinblick auf diese Probleme empfehlen viele Autoren, die sich mit der Problematik der Implementierung von ERP-Systemen befassen, die codebasierte Modifikation des Standardsystems möglichst zu vermeiden und gegebenenfalls die unternehmerischen Strukturen bzw. Geschäftsprozesse an die im ERP-System enthaltenen Standards anzupassen (vgl. Arif et al. 2010; Kumar et al. 2010; Chen et al. 2009; Gargeya und Brady 2005).

Die in der Standardsoftware enthaltenen Geschäftsprozesse stellen die sogenannte *Best Practice* dar, die vom Softwareanbieter anhand der Erfahrungen von bisherigen Installationen entwickelt wurde (vgl. Davenport 1998, S. 125; Kurbel 2016, S. 317). Die Anpassung an die Standards erlaubt es dem Unternehmen von dem im ERP-System eingebetteten Know-how zu profitieren. Dabei ist jedoch zu beachten, dass die unterschiedlichen Anwendungsbereiche eines Unternehmens in unterschiedlichem Ausmaß einer Standardisierung unterzogen werden können (vgl. Luo und Strong 2004, S. 325; Subramoniam et al. 2009, S. 663).

Ist das Unternehmen durch individuelle Geschäftsprozesse gekennzeichnet, die direkt mit der Erbringung von Wettbewerbsvorteilen verbunden sind, ist eine Abbildung der unternehmerischen Geschäftsprozesse im ERP-System empfehlenswert (vgl. Haines 2009, S. 184; Zach und Munkvold 2012, S. 470).

Während das Customizing eine bessere Anpassung des ERP-Systems an die unternehmerischen Anforderungen bietet, übt es auch direkten Einfluss auf das mit der ERP-Systemimplementierung verbundene Budget und auf den für die Implementierung vorgesehenen Zeitraum aus (vgl. Beijsterveld und Groenendaal 2015, S. 379; Themistocleous et al. 2001; Nah und Delgado 2006, S. 101). Die Suche nach einer optimalen Lösung dieses Trade-offs stellt einen sensiblen Prozess dar, dem eine Analyse der bestehenden Lösungsalternativen vorausgegangen sein muss (vgl. Parthasarathy und Daneva 2016; Pajk und Kovačič 2013).

Die Bestimmung der minimalen erforderlichen Anpassung ist schwierig und hängt weitgehend von dem Unternehmen ab (vgl. Davis 2005, S. 251 ff.). Die meisten Unternehmen erwarten, dass das ERP-System die tägliche Arbeit des Endbenutzers soweit wie möglich vereinfacht, um Zeit zu sparen und die Effektivität der Mitarbeiter zu steigern (vgl. Haines 2009, S. 190). Diese unternehmerischen Wünsche spiegeln sich oft in zusätzlichen Customizing-Anforderungen wider. Allerdings kann die Implementierung von zu vielen System-Modifikationen zum Anstieg der Betriebs- und Wartungskosten des ERP-Systems führen und in Extremfällen sogar zum Misserfolg der Systemimplementierung beitragen (vgl. z. B. Zach und Munkvold 2012, S. 466; Arif et al. 2010, S. 374; Callejas und Terzi 2013, S. 8).

Die zulässige Obergrenze des Customizing ist schwierig zu bestimmen. Einige Autoren, basierend auf den durchgeführten Untersuchungen und Analysen zahlreicher ERP-Implementierungsprojekte, legen die zulässige Grenze des Customizing bei ca. 20 % bis 30 % der ursprünglichen Systemfunktionalität fest. Obwohl die vorgeschlagenen Prozentwerte als Referenz verwendet werden können, ist das Ausmaß der erforderlichen Systemadaption durch sehr viele Faktoren bedingt und stellt in der Regel ein individuelles Merkmal jeder ERP-Systemeinführung dar (vgl. Parthasarathy und Daneva 2016).

Werden jedoch im Rahmen der Systemimplementierung zu viele Customizing-Aufgaben abgelehnt, können aufgrund fehlender Übereinstimmung zwischen den unternehmerischen Strukturen und den im ERP-System abgebildeten Prozessen weitere Probleme beim Systembetrieb entstehen (vgl. Scherrer-Rathje und Boyle 2008, S. 96). Die Literatur weist in diesem Zusammenhang z. B. auf Performance-Probleme bei der Bearbeitung der täglichen Aufgaben und auf allgemeine Probleme der Systemakzeptanz hin (vgl. Hohmann et al. 2013, S. 254).

In den meisten Fällen können die Fahrlässigkeiten bzw. Auslassungen der Systemimplementierung erst bei der Arbeit am System entdeckt werden (vgl. z. B. Soh et al. 2003). Die während der Betriebsphase sichtbaren Diskrepanzen zwischen den unternehmerischen Strukturen bzw. Prozessen und den in dem System implementierten Lösungen können nachträglich im Rahmen der Änderungsprojekte in der Betriebsphase korrigiert werden (vgl. Gattiker und Goodhue 2002; Zach und Munkvold 2012, S. 474).

### 2.3.2 Customizing während der Postimplementierungsphase

Im Zusammenhang mit der Postimplementierungsphase wird das Wort Customizing selten benutzt, viel öfter werden in der Literatur die in der Betriebsphase anfallenden Anpassungsaufgaben als Änderungsprojekte (engl. change projects) oder Wartung (engl. maintenance) bezeichnet (vgl. Kurbel 2013, S. 167).

Die Literatur bietet unterschiedliche Aufteilungen der Postimplementierungsaufgaben, welche in den Studien von Nah et al. (2001), Brehm et al. (2001) und Ng et al. (2002) näher betrachtet werden. Im Hinblick auf die Customizing-Prozesse können von den vorgeschlagenen Kategorisierungen drei Hauptaufgaben – Fehlerbehebung, funktionale Änderung und Erweiterung, Releasewechsel – abgeleitet werden, die entsprechend einen korrigierenden, adaptierenden bzw. aktualisierenden Charakter haben (vgl. Ng et al. 2002, S. 105 ff.).

Im Rahmen der Fehlerbehebung werden die übersehenen Fehler der Systemkonfiguration sowie Fehlerquellen in den Systemadaptionen und -erweiterungen beseitigt. In Abhängigkeit von dem zur Adaption des ERP-Systems verwendeten Customizing-Ansatz kann die Verantwortlichkeit für die Fehlerbeseitigung auf der Systemanbieterseite oder auf der -anwenderseite liegen (vgl. Haines 2009, S. 184).

Ist der Fehler durch den nicht modifizierten Quellcode des Basissystems verursacht, nimmt der Systemhersteller die Verantwortung für die Fehlerbehebung auf sich (vgl. Brehm 2001, S. 8206). Die bestehenden Fehlerstellen können von dem Systemhersteller unter anderem mittels Einspielen entsprechender Korrekturen (z. B. Patches) repariert werden. Ist der Fehler durch eine codebasierte Individualentwicklung bzw. Modifikation entstanden, muss in der Regel ein korrigierender Eingriff in die ursprünglich implementierten Individuallösungen seitens der Entwickler erfolgen (vgl. Nah et al. 2001, S. 406).

Daneben finden in der Postimplementierungsphase die funktionalen Änderungen des ERP-Systems statt, die aufgrund geänderter unternehmerischer, technischer oder legislativer Gegebenheiten vorgenommen werden müssen. Im Rahmen der Änderungsprojekte der Betriebsphase wird die ursprüngliche Konfiguration des ERP-Systems geändert bzw. sein Funktionsumfang erweitert.

Aufgrund der Tatsache, dass Unternehmen eine flexible Form darstellen und deren Gestalt variieren kann, können auch in der Postimplementierungsphase einige Änderungen der unternehmerischen Aufbau- und Ablauforganisation erforderlich sein. Neben der Systemadaption, die aufgrund der vom Anwenderunternehmen geplanten Änderungen der unternehmerischen Organisationsstrukturen bzw. Geschäftsprozesse entstehen, können auch einige unvorhersehbare Ereignisse eintreten, die in zusätzlichen Customizing-Aufgaben resultieren.

Während der alltäglichen Arbeit am System können einige Sonderfälle auftreten, die im bestehenden Systemrahmen nicht vorhergesehen wurden. Diese individuellen Geschäftsvorfälle stellen Ausnahmen von den im System angenommenen Standards dar und können als eine Prämisse zur Durchführung zusätzlicher Customizing-Aufgaben angesehen werden, um den bisherigen Lösungsumfang des ERP-Systems um die Möglichkeit der Realisation dieses Sonderfalls zu erweitern. Treten die Sonderfälle selten auf und können im bestehenden Systemrahmen (oder außerhalb des ERP-Systems; vgl. Dittrich et al. 2009) gesondert abgewickelt werden, kann die Durchführung der Systemanpassungen nochmals abgewogen werden (vgl. Seethamraju und Sundar 2013, S. 146; Beijsterveld und Groenendaal 2015, S. 379 f.). Ist mindestens eine dieser Konditionen nicht erfüllt, stellt eine Systemadaption eine unabdingbare Lösung dar.

Hierbei muss darauf hingewiesen werden, dass einige Änderungen aufgrund der bestehenden Wechselbeziehungen weitere Anpassungen der verbundenen Komponenten erforderlich machen können (siehe Kapitel 2.4.3).

Im Gegensatz zu den neu entstandenen Anforderungen der Betriebsphase, die eine entsprechende Systemanpassung bzw. -erweiterung erfordern, können im Laufe der Zeit auch einige der ursprünglich vorgenommenen Systemadaptionen von den Systembenutzern nicht mehr benötigt werden (vgl. Sekatzek und Krcmar 2009, S. 234). Werden die unbenutzten Systemadaptionen nicht frühzeitig identifiziert und aus dem System entfernt, können sie die kommenden Aktualisierungsprozesse verlängern und zusätzliche Wartungskosten generieren.

Da die Aktualität des ERP-Systems, seines Funktionsumfangs sowie die Kontinuität des Supports seitens des Softwareanbieters im ganzen Software-Lebenszyklus gewährleistet werden müssen, muss das ERP-System regelmäßigen *Upgrades* unterzogen werden (vgl. Beatty und Williams 2006, S. 106; Nah und Delgado 2006, S. 99; Haines 2009, S. 183). Ein System-Upgrade bezeich-

net einen Aktualisierungsprozess des bisherigen Systems auf eine höherwertige Systemversion (vgl. Ng et al. 2002, S. 91). Die Upgrades stellen eine der Hauptaufgaben der Postimplementierungsphase eines ERP-Systems dar (vgl. Nah und Delgado 2006, S. 103; Olson und Zhao 2007, S. 129; Brehm et al. 2001, S. 8019).

ERP-Upgrades zielen hauptsächlich darauf ab, dem Kunden die neuen technologischen Vorteile bzw. Geschäftsstrategien zu liefern und dadurch eine Unternehmensentwicklung in Übereinstimmung mit den aktuellen Trends im Bereich des Business Development zu ermöglichen (vgl. Brehm et al. 2001, S. 8019; Haines 2009, S. 183; Olson und Zhao 2007, S. 130).

Da infolge eines System-Upgrades einige Inkompatibilitäten zwischen der neuen Systemversion und den ursprünglich implementierten Individuallösungen entstehen können, müssen in der Regel nach einem Versionswechsel die Wartung und gegebenenfalls Wiederherstellung der Kompatibilität vorgenommen werden. Der Einfluss eines System-Upgrades auf die Kompatibilität der Systemadaptionen wird im Kapitel 2.3.3 thematisiert.

Obwohl ein Versionswechsel in der Regel mit zusätzlichen Anpassungsaktivitäten verbunden ist, muss bei jedem Upgrade auch überprüft werden, ob manche der ursprünglich individuell entwickelten Funktionalitäten vom Systemhersteller mit der neuen Systemversion standardmäßig mitgeliefert werden (vgl. Beatty und Williams 2006, S. 108; Seethamraju und Sundar 2013, S. 143). In diesem Fall kann das Unternehmen bei dem Umstieg auf die neue Systemversion auf einige der Individuallösungen verzichten und damit die in der Zukunft anfallenden Test- und Reimplementierungskosten einsparen (vgl. Haines 2009, S. 191; Sekatzek und Krcmar 2009, S. 237).

### 2.3.3 Stabilität der Systemanpassungen im Systemlebenszyklus

Die bei dem System-Upgrade entstehenden Arbeiten werden stark von den eingesetzten Customizing-Ansätzen beeinflusst (vgl. Light 2001, S. 425; Brehm et al. 2001, S. 8023; Callejas und Terzi 2013, S. 6; Rothenberger und Srite 2009, S. 664).

Im Fall von Anpassungen, die im Rahmen der Systemkonfiguration unter Verwendung der Parametrisierung vorgenommen wurden, sind die System-Upgrades nicht mit einem Verlust der bisherigen Einstellungen verbunden (vgl. Brehm et al. 2001, S. 8023; Nah und Delgado 2006, S. 110). Der Systemhersteller übernimmt in diesem Fall die volle Verantwortung für die Sicherstellung der Kompatibilität aller am System vorgenommenen Einstellungen (vgl. Haines 2009, S. 184).

Beim Einsatz der modellgetriebenen Systemanpassungen bleiben, wie im Fall der Parametrisierung, die vorgenommenen Adaptionen vorwiegend mit der neuen Systemversion kompatibel. Da die Änderungen in einer Modellform gesichert werden, die erst in der Systemlaufzeit zur automatischen Generierung des Komponentencodes verwendet wird, beeinflussen sie die Standardsoftware nicht direkt. Soweit der Systemversionswechsel nicht mit einem Wechsel der Modellnotation bzw. mit einer Änderung der durch das Modell angesprochenen Komponentenzusammenstellung verbunden ist, werden die vorgenommenen modellgetriebenen Systemanpassungen nicht tangiert. In anderen Fällen wird eine Adaption der durch den Versionswechsel betroffenen Modelle erforderlich.

Abgesehen von den parameterbasierten Systemanpassungen kann das ERP-System mithilfe von User-Exits erweitert bzw. adaptiert werden. Da User-Exits vom Softwareanbieter bereitgestellt werden, entsteht in der Regel keine Gefahr eines Funktionalitätsverlustes der einzelnen Anpassungen bzw. Erweiterungen bei einem Systemversionswechsel (vgl. Haines 2009, S. 184). Die Softwareanbieter verpflichten sich im Lizenzvertrag, die Kompatibilität der Systemschnittstellen bei künftigen ERP-System-Upgrades sicherzustellen.

Ausnahme von dieser Regel kann jedoch ein technologisches Upgrade sein, bei dem ein Umstieg auf eine völlig neue bisher durch das System nicht genutzte Technologie stattfindet. In diesem Fall müssen zusätzliche Tests durchgeführt werden, welche die Kompatibilität der einzelnen Erweiterungen mit der neuen Systemversion überprüfen (vgl. Sekatzek und Krcmar 2009, S. 234). Werden bei der Untersuchung einige Diskrepanzen entdeckt, so dass eine reale Gefahr der Inkompatibilität besteht, ist die Reimplementierung der einzelnen Erweiterungen unabdingbar (vgl. Ng et al. 2003, S. 7).

Die bereitgestellten Programmierschnittstellen sind ebenfalls relativ stabil (vgl. Kurbel 2016, S. 321). Ist die Langzeitstabilität einer API von dem Systemhersteller gewährleistet, so muss die mittels API bereitgestellte Funktionalität nach einem Systemversionswechsel meist nicht gesondert gewartet werden. Wie die Studie von Doedt und Steffen (2011) zeigt, kann jedoch die API-Stabilität von der Systemreife abhängig sein. Entscheidet sich der Systemhersteller für einige Änderungen bzw. Erweiterungen der APIs, sind die Entwickler der systemdedizierten Zusatzanwendungen zu Kompatibilitätstests gezwungen. Sind die Systemerweiterungen mit der neuen Systemversion nicht mehr kompatibel, müssen die Programmierarbeiten zwecks der Wiederherstellung der Kompatibilität vorgenommen werden.

Um diesen Situationen entgegenwirken zu können, werden die bevorstehenden API-Änderungen häufig an die in der Entwicklung von Zusatzlösungen involvierten Drittanbieter noch vor der Einführung der neuen Systemversion bekanntgegeben. Einige Systemhersteller verpflichten sich im Voraus, die API-Stabilität für eine bestimmte Anzahl an Systemversionen zu garantieren (vgl. z. B. Konstantinidis et al. 2012, S. 625).

Beim Einsatz der Systemerweiterungen, die eigenentwickelt bzw. fremdbeschafft wurden, besteht oft die Gefahr eines Upgrade-bedingten Kompatibilitätsverlustes. Aus diesem Grund müssen die Systemerweiterungen nach jedem Versionswechsel erneut getestet und gegebenenfalls gewartet bzw. umprogrammiert werden.

Verwendet das Unternehmen einige Systemkomponenten (z. B. Spezialmodule, Funktionalitätserweiterungen etc.), die von Drittanbieter bereitgestellt werden, so sind diese auch für die Sicherstellung der Kompatibilität ihrer Lösung mit der neuen Systemversion verantwortlich. Im Rahmen des Lizenzvertrages werden häufig von Lösungsanbietern die entsprechenden Patches installiert, die die verlorene Kompatibilität wiederherstellen. Da die erforderlichen Patches von dem Drittanbieter in den meisten Fällen erst nach Ausgabe der neuen Softwareversion vom Systemhersteller erstellt, getestet und dem Kunden zur Verfügung gestellt werden, kann eine Verzögerung entstehen, bis die volle Systemfunktionalität nach dem Upgrade sichergestellt wird. Die beim Versionswechsel bedingte Verzögerung der Ausgabe von Aktualisierungen für Drittparteilösungen wird in der Literatur als *Release Lag* bezeichnet (vgl. Brehm et al. 2001, S. 8023). Die Lösung des oben beschriebenen Problems hängt jedoch stark von der Art der Kooperation zwischen den beiden Software-Anbietern ab und kann im Rahmen der Zusammenarbeit noch im Stadium der Entwicklungsarbeiten an der neuen Systemversion vorbereitet werden (vgl. Magnusson und Nilsson 2013, S. 234 f.; Peng und Gala 2014, S. 26).

Die codebasierten Systemmodifikationen werden durch einen Releasewechsel am ehesten gefährdet (vgl. Brehm et al. 2001, S. 8023 f.). Alle Änderungen des Systemcodes werden in der Regel vom Systemhersteller nicht unterstützt und aus jeglichem Support ausgeschlossen (vgl. Haines 2009, S. 184; Beatty und Williams 2006, S. 106). Dies bedeutet für das Anwenderunternehmen, dass die Reimplementierung der individuellen Modifikationen in der neuen Systemversion vorgenommen werden muss (vgl. Light 2001, S. 419 ff.; Olson und Zhao 2007, S. 135). Die komplette Übertragung, Tests bzw. Anpassung aller codebasierten Systemmodifikationen werden in der Regel bei jedem neuen System-Upgrade erforderlich (vgl. Ng et al. 2003, S. 7; Beatty und Williams 2006, S. 108 f.; Bhamangol et al. 2011, S. 5). Jede einzelne Systemmodifi-

kation, die in die neue Systemversion übertragen, auf Kompatibilität getestet und gegebenenfalls umprogrammiert werden muss, verursacht zusätzlichen Aufwand und verlängert die für das Upgrade benötigte Zeit.

Zusammenfassend werden die Systemanpassungen, die nicht mit codebasierten Änderungen verbunden sind, durch das System-Upgrade nicht tangiert (vgl. Sekatzek und Krcmar 2009, S. 237). Die Individualentwicklungen bzw. Modifikationen gestalten sich im Gegenteil dazu sehr aufwendig.

## 2.4 Customizing-Aufgaben

Die Ursache für Customizing ist häufig eine fehlende Übereinstimmung zwischen den unternehmerischen Anforderungen und den im ERP-System enthaltenen Standards. Die von Soh et al. (2000) vorgeschlagene Typologisierung unterscheidet drei Arten von Inkompatibilitäten – *Daten*-, *Funktionalitäts*- und *Präsentations- bzw. Output-Inkompatibilität*:

- Die *Inkompatibilität der Daten* entsteht, wenn die sytembezogenen Datenformate bzw. die Relationen zwischen den Datenobjekten den Anforderungen des Anwenderunternehmens nicht entsprechen bzw. wenn die erforderlichen Datenobjekte im System nicht vorhanden sind.
- Die *Inkompatibilität der Funktionalität* entsteht, wenn die vom Anwenderunternehmen benötigte Funktionalität im System nicht enthalten ist bzw. wenn die unternehmerischen Prozesse und Prozeduren mit den im System vorhandenen Standards nicht übereinstimmen.
- Die *Inkompatibilität der Präsentation* entsteht, wenn der Informationsinhalt oder seine Darstellungsform im System den unternehmerischen Bedürfnissen nicht entsprechen (vgl. Soh et al. 2000, S. 48 ff.).

Die vorgeschlagene Aufteilung weist darauf hin, dass die Inkompatibilitäten unterschiedliche, architektonische Ebenen eines ERP-Systems betreffen können. Der Aufteilung fehlt jedoch eine entsprechende Detaillierung, aus welcher konkrete Customizing-Aufgaben abgeleitet werden können. Zur Identifizierung der meist auftretenden Customizing-Aufgaben bzw. -Anforderungen wurde die Literaturanalyse vorgenommen.

Die Identifizierung der häufigsten Customizing-Anforderungen war für die bevorstehenden Untersuchungen der Folgekapitel (insbesondere Kapitel 5 und 1) von besonderer Bedeutung. Aufgrund der unternehmerischen Vielfalt wäre es nicht möglich, im Rahmen der Ausarbeitung alle potenziell auftretenden Customizing-Aufgaben zu betrachten. Dem Häufigkeitsprinzip folgend, soll die vorliegende Ausarbeitung auf die Adaptionsanforderungen fokussiert werden, die für mehrere Anwenderunternehmen häufig relevant sind.

### 2.4.1 Adaptionsanforderungen

Im Rahmen des Studiums wurden diejenigen Systembereiche bzw. -elemente identifiziert, welche am häufigsten an die Anforderungen der Anwenderunternehmen angepasst werden müssen. Die Untersuchung wurde anhand einer quantitativen Inhaltsanalyse vorgenommen.

Die quantitative Inhaltsanalyse wird mit dem Zweck durchgeführt, mittels Quantifizierung der Inhalte ein allgemein gültiges Muster zu entdecken (vgl. Bortz und Döring 2006, S. 149). Der Einsatz dieser Untersuchungsmethode wird wegen der unzureichenden Berücksichtigung des Kontextes und der Bedeutung der Textinhalte manchmal kritisiert (vgl. Klammer 2005, S. 256 f.). Trotzdem zeigen die anderen Studien (vgl. z. B. Riedl und Roithmayr 2006; Kabbedijk et al. 2015), dass diese Technik zur Beantwortung der deskriptiv-statistischen Fragestellungen der Wirtschaftsinformatik erfolgreich eingesetzt werden kann.

Im Rahmen der Ausarbeitung wurden die Regeln einer quantitativen Inhaltsanalyse nach Bortz und Döring (2006, S. 149 ff.) beachtet, um die häufigsten Adaptionsanforderungen eines ERP-Systems zu identifizieren. Es wurde dabei von der These ausgegangen, dass durch die Gegenüberstellung der durch mehrere Studien dokumentierten Systemadaptionen auf die am zahlreichsten auftretenden Adaptionsanforderungen geschlossen werden kann.

Den Untersuchungsprinzipien nach Brereton et al. (2007, S. 577 f.) folgend, wurden für die Literaturrecherche die Online-Literaturdatenbanken ausgewählt, welche die relevantesten Konferenzen und Zeitschriften aus dem Themenbereich „Enterprise Resource Planning“ beinhalten (vgl. Schlichter und Kraemmergaard 2010, S. 493). Um die Anzahl der Suchergebnisse möglichst abzugrenzen, wurde eine Suche anhand der Metadaten (Titel, Abstract, Schlagwörter) vorgenommen. Die Suchanfrage wurde durch die Zusammensetzung der Terme „Customizng“ und „Enterprise Resource Planning“ gebildet.

Die metadatenbasierte Literatursuche der oben genannten Literaturdatenbanken lieferte 1.497 Veröffentlichungen. Nach der Entfernung der sich wiederholenden Beiträge verblieben 1.156 Quellen, die einer manuellen und inhaltlichen Analyse der Abstracts unterworfen wurden. Es wurden 924 Beiträge aufgrund unzureichender Relation mit dem Untersuchungsthema von der weiteren Analyse entfernt. Die verbliebenen 232 Veröffentlichungen bildeten die Basis für die inhaltliche Analyse.

Im Hinblick auf den quantitativen Charakter der Untersuchung wurden im ersten Schritt der Inhaltsanalyse die quantitativen Studien ermittelt, welche präzise Angaben über den untersuchten Stichprobenumfang beinhalten. Die Veröffentlichungen, welche über qualitativen Untersuchungen berichten, wurden nur dann in die weitere Analyse einbezogen, wenn der Umfang der analysierten Fallstudien bzw. die Anzahl der untersuchten ERP-Systeme im Text explizit vorgegeben war. Diese Phase der Inhaltsanalyse endete mit einer Gesamtheit von 72 Veröffentlichungen, aus der 18 (25%) als Stichprobe nach dem Zufallsprinzip ausgewählt und für die weitere Analyse verwendet wurden.

Der nächste Schritt der Inhaltsanalyse wurde mit der Absicht verfolgt, die dokumentierten Systemadaptionen bzw. Adaptionsanforderungen herauszufiltern und zu quantifizieren. Es wurde für jede der analysierten Veröffentlichungen eine Liste der Systemadaptionen erstellt. Durch Zusammenstellen der einzelnen Listen ergab sich eine Kreuztabelle (siehe Anhang A.1) mit den Veröffentlichungen (vertikal) und den identifizierten Systemadaptionen (horizontal). Danach wurden für jede Gruppe der identifizierten Systemadaptionen die Häufigkeiten aufsummiert und die prozentualen Werte berechnet. Das Ergebnis der vorgenommenen Auswertung stellt die Tabelle 2.3 dar.

Die vorgenommene quantitative Inhaltsanalyse anhand der 18 Veröffentlichungen hat als häufig auftretende Customizing-Anforderungen die Definition neuer Systemfunktionalität und das Anlegen individuell angepasster Berichte und Dashboards identifiziert. Diese Adaptionsanforderungen wurden bei 77,8% der analysierten Studien angezeigt. Diesen Anpassungsaktivitäten folgen Änderungen der Parametereinstellungen (61,1 %) und Adaptionen des Layouts einzelner Systemseiten (61,1 %).

Die Literatur weist weiterhin auf den Bedarf der Integration systemexterner Anwendungen (55,6 %) und der Adaption der Benutzeroberflächen mittels Ein- und Ausblenden einzelner Oberflächenkomponenten (38,9 %) hin. In jeder dritten Veröffentlichung wurden die Anpassungen des Systemumfangs durch die Zusammenstellung der erforderlichen Systemkomponenten und die weitgehenden Adaptionen der standardisierten Geschäftsprozesse dokumentiert.

Daneben benötigen gemäß der Literaturstudium die Anwenderunternehmen entsprechende Adaptionen der Prozedurabläufe (27,8 %), die Definition individuallisierter Dokumentenvor-

lagen (27,8 %) sowie die Integration externer Komponenten (z. B. Add-ons), die sich im Angebot der Drittanbieter befinden (27,8 %). Des Weiteren wurden bei 27,8 % der analysierten Veröffentlichungen zusätzliche Anforderungen hinsichtlich der Adaption der standardisierten Datenstrukturen – insbesondere der Datenmodelle und der Datenformate ermittelt.

**Tabelle 2.3:** Ergebnisse der quantitativen Literaturstudie (n = 18)

| Adaptionsanforderung | Anzahl | Anteil | Adaptionsanforderung | Anzahl | Anteil |
|---|---|---|---|---|---|
| Definition neuer Systemfunktionalität | 14 | 77,8 % | Adaption der standardisierten Datenmodelle | 5 | 27,8 % |
| Definition neuer Berichte und Dashboards | 14 | 77,8 % | Adaption der standardisierten Datenformate | 5 | 27,8 % |
| Änderung der Parametereinstellungen | 11 | 61,1 % | Änderung der standardisierten Nomenklatur und Terminologie | 4 | 22,2 % |
| Adaption des Layout der Systemseiten | 11 | 61,1 % | Definition neuer Kennzahlen | 3 | 16,7 % |
| Integration externer Anwendungen | 10 | 55,6 % | Definition neuer Datenfelder | 3 | 16,7 % |
| Ein- bzw. Ausblenden der Oberflächenkomponenten | 7 | 38,9 % | Adaption der Organisationsstruktur | 3 | 16,7 % |
| Änderung der Zusammenstellung von Systemkomponenten | 6 | 33,3 % | Adaption der rollenbasierten Berechtigungen und Zuständigkeiten | 2 | 11,1 % |
| Individualisierung der Geschäftsprozesse | 6 | 33,3 % | Definition neuer Datenquellen für Berichte und Diagramme | 2 | 11,1 % |
| Anlegen individuallisierter Dokumentenvorlagen | 5 | 27,8 % | Adaption der Dokumentenflüsse | 2 | 11,1 % |
| Adaption der Prozedurabläufe | 5 | 27,8 % | Adaption der Formularansicht an das Corporate-Design | 2 | 11,1 % |
| Integration externer Systemkomponenten | 5 | 27,8 % | Sonstige Adaptionsanforderungen | 6 | 33,3 % |

Einige Studien deuten auf eine Änderung der bestehenden Terminologie (22,2 %), die Definition neuer Kennzahlen (16,7 %) und andere wiederum auf die Erweiterung der Geschäftsobjekte mit neuen Datenfeldern (16,7 %) sowie die Anpassung der Organisationsstruktur (16,7 %) hin.

Weniger häufig wurde in der analysierten Literatur auf die Anforderungen hinsichtlich der Adaption der rollenbasierten Systemberechtigungen (11,1 %) und der Definition neuer Datenquellen für Berichte und Diagramme (11,1 %) hingewiesen. Neben den oben aufgelisteten Adaptionsanforderungen sind bei 11,1 % der Anwenderunternehmen weitere Adaptionen der Dokumentenflüsse und visuelle Anpassungen der Formulare notwendig.

Darüber hinaus wurden in der untersuchten Stichprobe auch die weiteren Systemadaptionen identifiziert, die unternehmensspezifischer sind und deshalb entsprechend seltener vorkommen. Diese Adaptionen wurden in der letzten Zeile unter „Sonstige Adaptionsanforderungen“ zusammengefasst.

### 2.4.2 Häufigkeit des Customizing

Aufgrund der sich ändernden unternehmerischen Bedürfnisse, welche aus den unternehmensabhängigen Ursachen (z. B. Änderungen des Unternehmens, unternehmerischen Geschäftsum-

felds, der Strategie und Technologie) bzw. unabhängigen Gründen (z. B. Änderung der rechtlichen Vorschriften etc.) resultieren können (vgl. z. B. Hecht et al. 2011; Cao 2010, S. 1601), müssen die standardisierten ERP-Systeme an die aktuellen Anforderungen angepasst werden (vgl. Zach und Munkvold 2012, S. 472 f.; Sekatzek und Krcmar 2009, S. 234; Ross und Vitale 2000, S. 240).

Aus der zeitlichen Perspektive charakterisieren sich die Customizing-Aufgaben unter anderem durch ihre *Wiederholungshäufigkeit*. Die Wiederholungshäufigkeit definiert, wie häufig eine Customizing-Aufgabe im Systemlebenszyklus vorgenommen wird. Aufgrund der unternehmerischen Unterschiede können in der Regel keine allgemeingültigen absoluten Häufigkeiten für die einzelnen Customizing-Aufgaben ermittelt werden. In diesem Kontext versuchen Hufgard und Krüger (2012) einige der typischen Customizing-Aufgaben auf einer ordinal skalierten Zeitachse anzuordnen. Obwohl die Auftrittshäufigkeit der einzelnen Aufgaben von unterschiedlichen Faktoren abhängig ist, bietet die Abbildung 2.2 einen generalisierten Überblick über die Wiederholungshäufigkeit der einzelnen Systemanpassungen im ERP-Systemlebenszyklus.

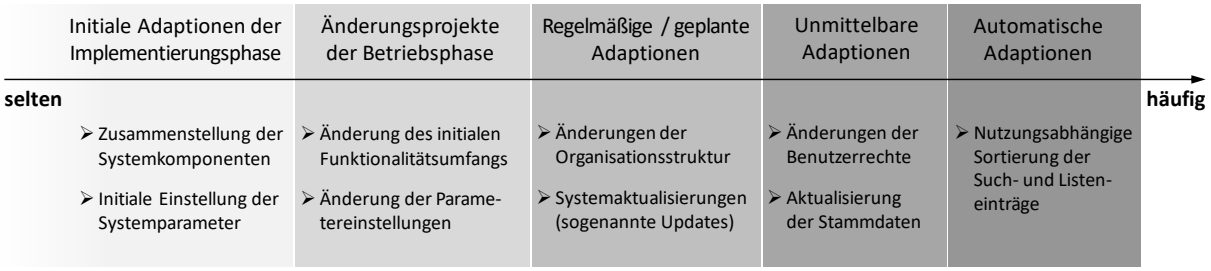



**Abbildung 2.2:** Wiederholungshäufigkeit der Systemanpassungen in Anlehnung an Hufgard und Krüger (2012, S. 220)

Die Systemadaptionen der Implementierungsphase – die initiale Zusammenstellung der Systemkomponenten bzw. die initiale Parametrisierung der einzelnen Systembestandteile – haben einen einmaligen Charakter. Ein bisschen öfter können die Customizing-Prozesse durchgeführt werden, die mit dem Zweck der Änderung der initialen Funktionalitätsumfangs erfolgen. Die Systemanpassungen, welche eine Erweiterung oder eine Modifikation der Geschäftsprozesse oder der Systemfunktionalität zur Folge haben, treten jedoch grundsätzlich selten ein. Dies trifft auch auf die Integrationsprozesse zu, welche nur einmalig bei der Implementierung bzw. selten beim Erwerb oder bei der Aktualisierung von externen Anwendungen durchgeführt werden müssen.

Die Studie von Zach und Munkvold (2012) zeigt dabei, dass die Wiederholungshäufigkeit einiger Customizing-Aufgaben u. a. von der Unternehmensreife abhängen kann. Während Unternehmen, welche bereits länger auf dem Markt agieren, meist gut etablierte Geschäftsprozesse besitzen, können bei neuen klein- und mittelständischen Unternehmen die Betriebsprozesse häufiger Änderungen unterliegen (vgl. Zach und Munkvold 2012, S. 473).

Demgegenüber werden einige Systemanpassungen auf einer regelmäßigen bzw. vorhersehbaren Basis vorgenommen. Als Beispiele können hier die Aktualisierung der buchhalterischen Vorschriften bzw. Änderungen des Beitragssatzes bei der gesetzlichen Krankenversicherung genannt werden (vgl. Ackermann und Turowski 2006, S. 355 f.). Die erforderlichen Anpassungen können entweder seitens der Systemhersteller in Form einer dedizierten Systemaktualisierung (sogenanntes Update) erfolgen oder vom Anwender durch die Änderung der Systemparameter vorgenommen werden.

Neben den von dem Anwenderunternehmen unabhängigen Anpassungen finden in der Betriebsphase auch einige geplante bzw. periodische Customizing-Aufgaben statt, die die unternehmerischen Veränderungen widerspiegeln (vgl. Scherrer-Rathje und Boyle 2008, S. 90). Hierzu zählen z. B. die Veränderungen der organisatorischen Strukturen, die durch Reorganisationsprozesse bzw. Einstellung von neuen Mitarbeitern verursacht werden (vgl. Hufgard und Krüger 2012, S. 220).

Im Tagesgeschäft können auch einige unvorhersehbare Customizing-Aufgaben entstehen, die wegen direkter Auswirkungen auf bestimmte Geschäftsprozesse bzw. -abläufe unverzüglich erfolgen müssen. Die Adaption der Geschäftsabläufe zwecks der Realisation eines variablen Geschäftsvorfalls im Systemrahmen stellt eine relativ häufige Customizing-Aufgabe dar, die unmittelbar umgesetzt werden soll. Die unmittelbaren Anpassungen kommen häufig vor und müssen dann mit der hohen Priorität durchgeführt werden. Zu den unmittelbaren Anpassungen gehören unter anderem Aktualisierungen von Stammdaten für Produkte und Vertragspartner, Veränderung der Zugriffsrechte der Mitarbeiter etc. (vgl. Scherrer-Rathje und Boyle 2008, S. 91 f.).

Wenn die Systemänderung oder -anpassung nicht mit der Notwendigkeit der Durchführung von zusätzlicher Adaptionsarbeiten in Form von größeren Customizing-Projekten verbunden ist (und demnach keine Seiteneffekte und Nebenwirkungen für die anderen Systemelemente mit sich bringt), sollten diese Aufgaben in Form von unmittelbaren Anpassungen durchsetzbar sein (vgl. Scherrer-Rathje und Boyle 2008, S. 91).

Bei modernen ERP-Systemen gibt es auch Anpassungen, welche automatisch (durch das System) durchgeführt werden. Die automatischen Adaptionen werden größtenteils anhand der systembezogenen Analyse des Nutzerverhaltens durchgeführt. Als Beispiel hierfür kann die Zusammenstellung der am häufigsten verwendeten Systemfunktionen bzw. die Sortierung der einzelnen Sucheinträge in Abhängigkeit von deren Nutzungsintensität genannt werden (vgl. Hufgard und Krüger 2012, S. 220). Einige Forscher sehen in der Automatisierung des Customizing (z. B. UI-Anpassung etc.) das Potenzial für eine Weiterentwicklung von ERP-Systemen (vgl. Lucas und Babaian 2009; Eichler und Dostál 2012; Kassem und Schult 2008; Wölfel 2015).

Das ERP-System muss sowohl die erforderliche Anpassungsflexibilität für das Tagesgeschäft sicherstellen als auch die Durchführung von umfangreicheren Änderungsprojekten ermöglichen sowie eine Änderung der bestehenden Systemzusammenstellung oder Erweiterung der Systemfunktionalität erlauben (vgl. Scherrer-Rathje und Boyle 2008, S. 83 f.).

### 2.4.3 Wechselwirkungen und Seiteneffekte beim Customizing

Während der Customizing-Prozesse spielen die Wechselwirkungen und Abhängigkeiten zwischen den einzelnen Komponenten des ERP-Systems eine sehr wichtige Rolle (vgl. Dittrich et al. 2009, S. 46).

Vor der Ausführung jeder Adaption des Standardsystems ist daher eine genaue Analyse des Einflusses auf andere Systemkomponenten empfehlenswert (vgl. Haines 2009, S. 186 f.). Die frühzeitige Beurteilung des Umfangs möglicher Wechselwirkungen ermöglicht eine genauere Planung der einzelnen Adaptionsprozesse – insbesondere der Auswahl der geeigneten Customizing-Werkzeuge sowie eine genauere Einschätzung der erforderlichen Zeit (vgl. Nah und Delgado 2006, S. 110).

Die Abhängigkeiten zwischen den einzelnen Systemkomponenten steigern die Komplexität der Customizing-Aufgaben. Die Untersuchungen von Wei et al. (2005), Dittrich et al. (2009) und Brehm et al. (2001) bestätigen, dass viele Customizing-Aufgaben weitere Anpassungsaktivitäten

zur Folge haben. Ein illustratives Beispiel dafür stellt die Adaption bestehender Geschäftsabläufe dar, die eine Definition zusätzlicher Berichte bzw. die Definition neuer Kennzahlen erforderlich machen kann (vgl. Scherrer-Rathje und Boyle 2008, S. 91; Light 2001, S. 423).

Weiterhin könnte der Fall eintreten, dass Änderungen der Aufbauorganisation vorgenommen werden müssen, die eine Vergabe der Zugriffsrechte und Verantwortlichkeiten, aber manchmal auch das Umgestalten der bestehenden Geschäftsabläufe oder sogar die Definition von neuen Prozeduren nach sich ziehen können (vgl. Seethamraju und Sundar 2013, S. 146).

Im Fall der tiefgreifenden Systemänderungen, in deren Rahmen eine Implementierung der zusätzlichen Systemkomponenten (z. B. neue Geschäftsprozesse, funktionale Systemerweiterungen) erfolgt, kann eine Adaption der ursprünglichen Systemkonfiguration erforderlich sein (vgl. Zach und Munkvold 2012, S. 473).

Die Abbildung 2.3 stellt die möglichen Abhängigkeiten zwischen den Customizing-Aufgaben dar. Die Pfeile zeigen die Richtung einer Wechselbeziehung zwischen einzelnen Systemadaptionen an.

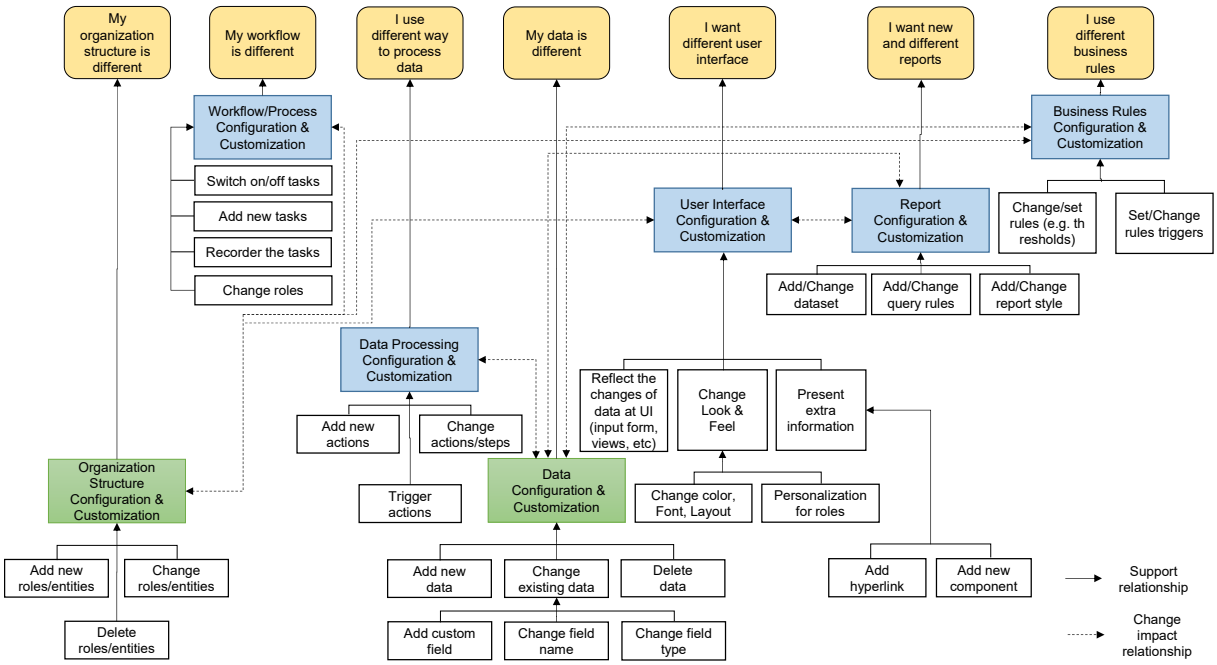



**Abbildung 2.3:** Wechselbeziehungen zwischen Customizing-Aufgaben nach Guo et al. (2011)

Die Abbildung hat lediglich einen illustrativen Charakter und weist auf mögliche Wechselbeziehungen hin. Da verschiedene ERP-Systeme durch eine unterschiedliche Architektur bzw. Komponentenzusammensetzung gekennzeichnet sind, stellt die Abbildung kein allgemeingültiges Muster dar.

In Abhängigkeit vom Umfang der Wechselwirkungen und Seiteneffekte kann die einzelne Systemadaption entweder direkt in dem laufenden System durchgeführt werden oder es sind tiefgreifende und zeitintensive Systemänderungen erforderlich, so dass weitreichende Veränderungen der ursprünglichen Systemkonfiguration vorgenommen werden müssen.

Aufgrund von Wechselwirkungen und Seiteneffekten zwischen den verschiedenen Systemkomponenten kann die Umsetzung einiger Adaptionen manchmal arbeitsintensiver als ursprünglich geplant sein.

Einfache Systemanpassungen, die durch auftretende Sonderfälle bzw. durch individuelle Geschäftsvorfälle verursacht werden, können größtenteils in Form einzelner Anpassungen am laufenden System vorgenommen werden. Tiefgreifende Systemadaptionen aufgrund umfangreicher Seiteneffekte erfordern eine Durchführung von zusätzlichen Änderungen in den direkt verbundenen Systemkomponenten (vgl. Brehm et al. 2001, S. 8018). In diesem Fall werden die mit den Seiteneffekten verbundenen Systemadaptionen in Form von Änderungsprojekten durchgeführt. Aufgrund ihrer Komplexität können die Änderungsprojekte in der Regel nicht ad hoc auf dem laufenden System vorgenommen werden.

# 3 Architekturmerkmale der cloudbasierten ERP-Systeme

Das Cloud-Computing setzt keine feste Architektur voraus bzw. nennt keine Architekturkonzepte, denen eine Architektur unbedingt folgen muss. In der Literatur werden jedoch einige allgemeine Merkmale definiert, welche eine Anwendung in der Cloud charakterisieren (vgl. Kapitel 1.2.1). Im Rahmen dieses Kapitels werden die architektonischen Hauptmerkmale einer in der Cloud betriebenen Anwendung abgeleitet und deren Einfluss auf die Anpassungsfähigkeit des cloudbasierten ERP-Systems untersucht.

## 3.1 Cloudbasierte ERP-Systeme

Die ERP-Systeme in der Cloud stellen die gleiche Funktionalität wie die konventionellen On-Premise-ERP-Systeme bereit, unter Beibehaltung der Prinzipien des Cloud-Computing (vgl. Kapitel 1.2.1). Laut der über die SaaS-bezogene Literatur etablierten Definition von Chong und Carraro (2006) bezeichnet der Begriff *cloudbasiert* eine Anwendung, die in der *Cloud* betrieben und einem Anwender über das Internet in Form einer Dienstleistung zur Verfügung gestellt wird (vgl. Chong und Carraro 2006).

Folglich werden die ERP-Systeme, bei welchen die Cloud als „Basis“ des Bereitstellungsmodells verwendet wird, zunehmend abgekürzt als *Cloud-ERP* oder *ERP-as-a-Service* (ERPaaS) bezeichnet (vgl. Hao et al. 2012, S. 410). Im Rahmen der aufgestellten Definition eines cloudbasierten ERP-Systems, können unterschiedliche Konzepte zur Sicherstellung des Cloud-Bereitstellungsmodells genutzt werden.

Zur Schaffung einer erforderlichen konzeptionellen Basis für die Untersuchungen der Folgekapitel sollen zunächst die alternativen Konzepte zur Bereitstellung eines Systems im Cloud-Modell gegenübergestellt werden. Dies ermöglicht eine adäquate Differenzierung des Cloud-ERP-Begriffs und bildet eine Grundlage für die darauffolgende Analyse der Auswirkungen einzelner Architekturmerkmale auf die systembezogene Adaptierbarkeit.

### 3.1.1 Bereitstellungskonzepte

Das Cloud-Computing-Paradigma hat bei den Systemherstellern zu der Tendenz geführt, zunehmend die Cloud für die Bereitstellung ihrer ERP-Systeme anzuwenden (vgl. Duan et al. 2013; Scavo et al. 2012; Hamerman et al. 2017; Hedman und Xiao 2016). Es sind unterschiedliche Konzepte entstanden, die eine Bereitstellung eines cloudbasierten Systems ermöglichen.

Während aus der Sicht eines Anwenders das Bereitstellungskonzept des cloudbasierten ERP-Systems meistens vollständig transparent bleibt, ist aus der Perspektive eines Systemanbieters die Auswahl des geeigneten Bereitstellungsansatzes von besonderer Bedeutung (vgl. Saeed et al. 2012, S. 431). Welcher Ansatz von dem Systemanbieter in Betracht gezogen wird, hängt von vielen Faktoren ab, unter denen die Charakteristik des Kundensegments, die Architektur der gerade vorhandenen Lösungen und die Strategie besonders nennenswert sind (vgl. Sun et al. 2008).

Die grundlegenden Ansätze, mit deren Hilfe ein ERP-System in einem Cloud-Modell bereitgestellt werden kann, wurden in der Abbildung 3.1 visualisiert.

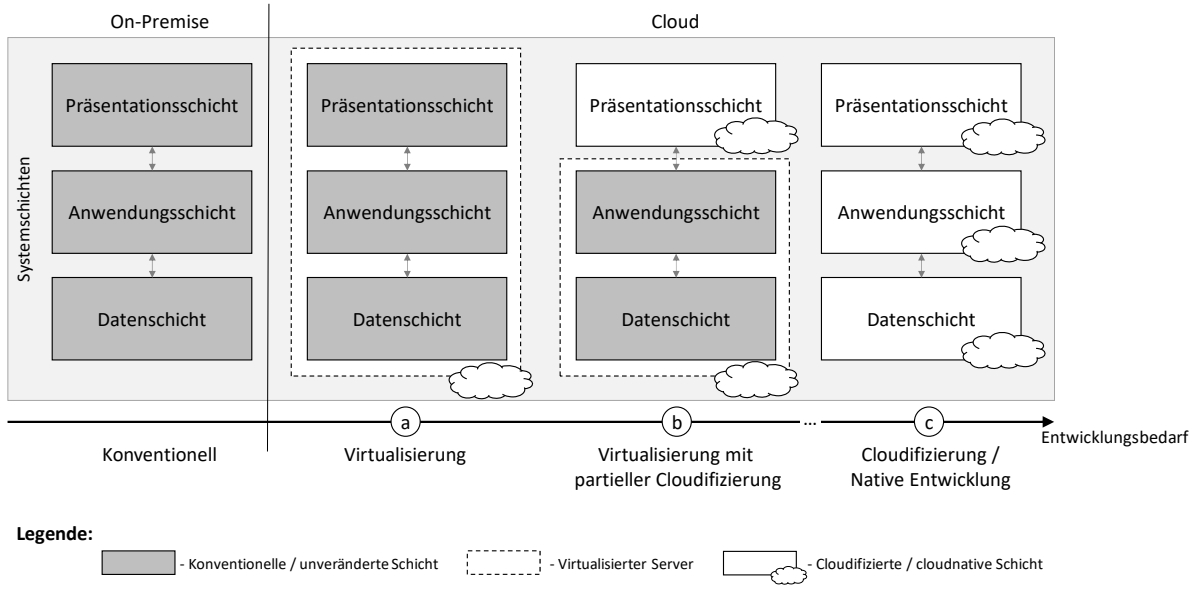



**Abbildung 3.1:** Bereitstellungskonzepte einer cloudbasierten Anwendung in Anlehnung an Jamshidi et al. (2013, S. 150)

*Cloudnative Entwicklung*

Die Neuentwicklung eines cloudbasierten ERP-Systems (siehe Abbildung 3.1, c) stellt vorwiegend ein erstrebenswertes Konzept zum Erreichen einer möglichst weitreichenden Übereinstimmung mit dem SaaS-Betreibermodell dar. Dieser Ansatz erfordert es, bereits in der Entwurfsphase des ERP-Systems ein architektonisches Konzept gemäß der Prinzipien des Cloud-Computing (vgl. Mell und Grance 2011) zu entwickeln. Die neuentwickelten Cloud-ERP sind in der Regel für die öffentlichen Clouds gedacht, die eine Grundlage des SaaS-Modells bilden (vgl. Duan et al. 2013).

Die cloudnative Entwicklung wird von den ERP-Anbietern ausgewählt, die ihr cloudbasiertes System von Anfang an entwickeln wollen, um die Vorteile des SaaS-Modells – insbesondere die Skaleneffekte – in möglichst vollem Umfang ausschöpfen zu können. Im Vergleich zu den anderen Bereitstellungskonzepten ist dieser Ansatz mit einem großen Entwicklungsaufwand verbunden. Er ermöglicht es aber, die künftigen Instandhaltungskosten des Systems zu senken und mit dem Nachfrageanstieg größere Gewinne zu erwirtschaften (vgl. Sun et al. 2008, S. 23).

*Virtualisierung*

Die Untersuchungen von Scavo et al. (2012) zeigen, dass auch die konventionellen, lokal installierbaren ERP-Systeme für einen Cloud-Betrieb adaptiert werden. Das steigende Interesse der ERP-Systemanbieter an der Adaption ihrer altbewährten ERP-Systeme zum Cloud-Betrieb wurde auch durch Marktstudien bewiesen (vgl. Eggert et al. 2013, S. 24 ff.; Grubisic 2014, S. 67).

Der relativ simple Ansatz für eine Umstellung auf das Cloud-Modell stellt die *Virtualisierung* dar, welche als eine Migration der gesamten Anwendung auf einen oder mehrere in der Cloud laufenden, virtualisierten Servern definiert werden kann (siehe Abbildung 3.1, a). Dabei stellt ein virtueller Server (eine sogenannte virtuelle Maschine) die Nachbildung einer Serverarchitektur dar, die auf einem physikalischen Server läuft (vgl. Meinel et al. 2011, S. 18 ff.). Die physikalischen Server können eine Infrastruktur des SaaS-Anwendungsdienstleisters (z. B. ein Rechenzentrum) sein oder von ihm selbst im Rahmen eines Infrastruktur-as-a-Service von einem unabhängigen IaaS-Anbieter bezogen werden (vgl. Schubert und Adisa 2011, S. 14). Daneben

existieren mehrere Virtualisierungskonzepte, die auf unterschiedlichen Ebenen – Hardware, Betriebssystem, Middleware – realisiert werden können. Diese werden vertiefend von Meinel et al. (2011) beschrieben.

Die Virtualisierung ist meistens mit deutlich geringeren Aufwendungen seitens des Systemanbieters verbunden als die Neuentwicklung eines cloudbasierten Systems (vgl. Andrikopoulos et al. 2013, S. 526). Daneben fördert die gerade vorhandene webbasierte Systemoberfläche oft diese Form der Migration.

Die virtualisierten ERP-Systeme erfüllen die SaaS-Eigenschaften nicht zu dem gleichen Grad wie eine für das öffentliche Cloud-Modell konzipierte Anwendung und erlauben dem Systemanbieter von Skaleneffekten lediglich in einem begrenzten Umfang zu profitieren.

*Cloudifizierung*

Die alternative Migrationsoption besteht in einer kompletten Umgestaltung der On-Premise Anwendung bis hin zum Erreichen der vollständigen Übereinstimmung mit dem SaaS-Modell auf allen Anwendungsebenen (siehe Abbildung 3.1, c). Der Prozess in dessen Rahmen eine ursprünglich nicht cloudfähige Anwendung vollständig an das SaaS-Modell angepasst wird, wird in der Literatur oft als *Cloudifizierung* bezeichnet (*cloudification*; vgl. Andrikopoulos et al. 2013, S. 498).

Dieser Ansatz stellt die aufwendigste Migrationsoption dar, welche mit der Rekonzeptualisierung, Umgestaltung und dem Umprogrammieren der einzelnen Systemkomponenten eines konventionellen On-Premise-Systems verbunden ist (vgl. Hohmann et al. 2013, S. 256).

Aufgrund der architektonischen Herausforderungen, welche die Cloudifizierung mit sich bringt, entscheiden sich die Systemhersteller für eine stufenweise Migration ihrer Systeme in die Cloud (vgl. Ragusa und Puliafito 2011). Zunächst werden die einzelnen Systemkomponenten virtualisiert und danach schrittweise cloudifiziert (siehe Abbildung 3.1, a→b→c), bis hin zum Erreichen der angestrebten Übereinstimmung mit den Voraussetzungen des SaaS-Modells.

Daneben existieren auch andere Konzepte für eine partielle Migration einer konventionellen On-Premise Anwendung in die Cloud, die getrennt nach den architektonischen Schichten vorgenommen wird (mehr dazu Jamshidi et al. 2013, S. 150). Im Rahmen dieser Ausarbeitung wird gemäß der aufgestellten Definition eines cloudbasierten Systems ausschließlich auf die vollständig cloudbasierten Architekturen eingegangen, deren Schichten alle in einer Cloud bereitgestellt werden.

### 3.1.2 Cloudnativität und Cloudfähigkeit

Aufgrund der Unterschiede in den Bereitstellungskonzepten der cloudbasierten ERP-Systeme soll im Hinblick auf die bevorstehende Untersuchung eine Unterscheidung der cloudbasierten ERP-Systeme in *cloudnative* und *cloudfähige* ERP-Systeme vorgenommen werden.

In der vorliegenden Ausarbeitung bezeichnet das Adjektiv *cloudnativ* eine Anwendung bzw. Komponente, die von Anfang an konform für den Cloud-Betrieb entworfen wurde und deren Architektur alle Eigenschaften einer SaaS-Anwendung erfüllen kann (vgl. Andrikopoulos et al. 2013, S. 494).

Eine *cloudfähige* Anwendung bzw. Komponente stellt dagegen eine Software dar, welche ursprünglich als On-Premise-Software angeboten wurde und danach an den Betrieb im SaaS-Modell angepasst wurde. Diese Anwendung wird als cloudfähig bezeichnet, weil sie sowohl als konventionelle lokal installierbare Software als auch in einem Cloud-Betreibermodell angeboten werden kann (vgl. Scavo et al. 2012, S. 11). Die cloudfähigen Systeme realisieren meist

in unterschiedlichem Umfang die architektonischen Eigenschaften einer cloudnativen SaaS-Anwendung (vgl. Scavo et al. 2012, S. 3).

Bei einer kritischen Betrachtung der auf dem Markt verfügbaren ERP-Systeme, die laut ihren Anbietern cloudbasiert sind, zeigt sich in den meisten Fällen, dass nicht alle Eigenschaften eines SaaS-Betreibermodells vollständig erfüllt werden. Die Untersuchungen von Scavo et al. (2012) sowie von Zhao und Kirche (2013) zeigen, dass die Anbieter der cloudnativen ERP-Systeme teilweise auch auf einige Voraussetzungen des Cloud-Computing unter dem Nachfragedruck verzichten. Hao et al. (2012) berichten anhand von Experteninterviews, dass sich einige ERP-Systemanbieter lediglich auf die Bereitstellung ihrer Systeme in privaten Clouds konzentrieren, weil ihre Kunden kein Interesse an den öffentlichen Clouds haben. Diese Tendenz ist sogar bei den größten Anbietern erkennbar (vgl. Eggert et al. 2013, S. 25 f.), deren ERP-Systeme alle SaaS-Merkmale erfüllen. Daneben beeinflussen auch einige legislative Regelungen die Umsetzungskonzepte mancher Eigenschaften des SaaS-Modells, so dass diese nur in einem gewissen Grad realisiert werden können (vgl. Bezemer und Zaidman 2010a, S. 3). Dies betrifft z. B. die Speicherung der Betriebsdaten der Cloud-Anwender, die gemäß der internen Regelungen einiger Länder im Inland erfolgen muss (z. B. Patriot Act, vgl. Peng und Gala 2014, S. 27). Es wirkt sich hemmend auf die Umsetzung der Cloud-Computing-Prinzipien aus, gemäß denen bei der cloudbasierten Dienstleistung keine Transparenz der verwendeten physischen Hardware für die User ausgewiesen werden soll (vgl. Saeed et al. 2012, S. 432).

Diese Erkenntnisse deuten darauf hin, dass der Cloud-Markt noch nicht ausgereift ist und der Grad der Umsetzung einzelner Cloud-Computing-Prinzipien bei unterschiedlichen cloudbasierten Systemen variieren kann. Aufgrund der Differenzen zwischen den Bereitstellungskonzepten der cloudbasierten Systeme sind auch Differenzen in der Realisierung einzelner architektonischer Merkmale zu erwarten, welche wiederum einen Einfluss auf die erzielbare Adaptierbarkeit des cloudbasierten ERP-Systems haben können.

Aus diesem Grund müssen im Hinblick auf die bevorstehende Analyse zunächst die Hauptmerkmale der SaaS-Anwendung abgeleitet und entsprechend systematisiert werden. Dies wird anhand der Analyse der etablierten *Reifegradmodelle* der SaaS vorgenommen.

Die Reifegradmodelle wurden als Ausgangspunkt der Analyse ausgewählt, weil sie das gesamte Spektrum der cloudbasierten Systeme – von virtualisierten, cloudfähigen Anwendungen bis hin zu cloudnativen Systemen – betrachten und zugleich einen Bezug auf die Umsetzung der Anpassbarkeit sicherstellen.

## 3.2 Reifegradmodelle für SaaS

Ein *Reifegradmodell* kann als ein allgemeingültiges, wiederverwendbares Modell definiert werden, welches einen Entwicklungsprozess in einer Domäne – vom Begin bis hin zum Erreichen einer vollständigen Reife – anhand der aufeinander aufbauenden Entwicklungsstufen (Reifegrade) beschreibt (vgl. Becker et al. 2009, S. 249).

Im Kontext der SaaS beschreiben die SaaS-Reifegradmodelle die einzelnen Stadien eines Adaptationsprozesses einer Anwendung für den SaaS-Modell-konformen Cloud-Betrieb. Jeder SaaS-Reifegrad wird durch eine Zusammenstellung von Eigenschaften charakterisiert, welche eine Anwendung erfüllen muss, um eine bestimmte Reife zu erreichen.

Obwohl die SaaS-Reifegradmodelle für eine nicht konkretisierte (anonyme) Anwendung entwickelt wurden, sind sie per Definition des Reifegradmodells auch für die cloudbasierten ERP-Systeme anwendbar, die selbst eine Anwendung bzw. Zusammenstellung mehrerer Anwendungen (SOA; vgl. Jouanne-Diedrich et al. 2012) darstellen.

Im Rahmen dieses Unterkapitels wird eine komparative Analyse der wissenschaftlich etablierten SaaS-Reifegradmodelle durchgeführt, welche der Ableitung der architektonischen Hauptmerkmale einer SaaS-Anwendung dient. Obwohl die Identifizierung der SaaS-Merkmale den Kernaspekt im Hinblick auf die bevorstehende Analyse der Anpassungsmöglichkeiten der Cloud-ERP-Systeme darstellt, ist auch die Reihenfolge des Hinzufügens einzelner Merkmale bis hin zum Erreichen einer vollständigen SaaS-Reife von Bedeutung.

### 3.2.1 Chong-Carraro-Reifegradmodell

Das von Chong und Carraro im Jahr 2006 veröffentlichte SaaS-Reifegradmodell stellt eine erste in der wissenschaftlichen Community gut etablierte Aufteilung der cloudfähigen Architekturen dar (vgl. Brown und Nyarko 2012, S. 58). Dank seines direkten Bezugs auf die Umsetzung der Anpassungsfähigkeiten bildet das Modell die Grundlage zahlreicher Veröffentlichungen im Bereich des Customizing von Cloud-Systemen. Das Modell setzt voraus, dass eine cloudbasierte Anwendung auf unterschiedlichen SaaS-Reifestufen angeboten werden kann, wobei Letztere von der Realisierung der einzelnen Architektureigenschaften abhängig sind.

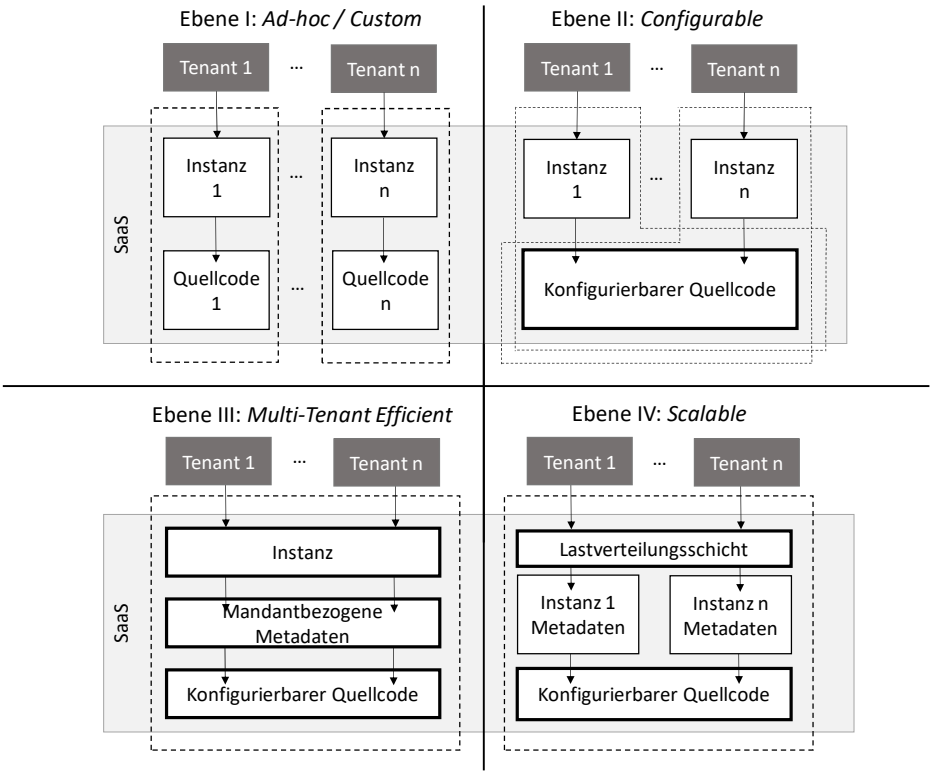



**Abbildung 3.2:** Anpassungskonzepte der SaaS-Reifestufen in Anlehnung an Chong und Carraro (2006)

Die Transformation einer nicht ausgereiften zu einer ausgereiften Cloud-Anwendung wird im Modellrahmen als ein Entwicklungsprozess anhand der vier aufeinander folgenden Reifestadien dargestellt. Abbildung 3.2 visualisiert die Anpassungskonzepte der einzelnen Reifestufen. Die vier Stadien im SaaS-Reifegradmodell von Chong und Carraro (2006) werden wie folgt definiert:

- *Ebene I – Ad hoc / Custom*: Auf der ersten Reife-Ebene wird die Anwendung jedem Anwenderunternehmen (sogenannten Mandanten) als eine separate Instanz auf einem dedizierten (virtualisierten) Anwendungsserver bereitgestellt. Die einzelnen Anwendungsinstanzen werden individuell an die Anforderungen der Anwenderunternehmen angepasst und vollstän-

dig unabhängig voneinander betrieben. Dieses Modell ähnelt somit sehr dem standardisierten On-Premise-Modell, mit dem Unterschied, dass sich die Infrastruktur auf der Service-Anbieterseite und nicht auf der Kundenseite befindet (vgl. Kulkarni et al. 2012, S. 14). Dies ermöglicht es, Einsparungen durch die Zentralisierung der Anwendungsbereitstellung und -verwaltung sowie die effizientere Nutzung von Hardware-Ressourcen zu erreichen (vgl. Sääksjärvi et al. 2005, S. 180). Dieser Ansatz stellt die einfachste Möglichkeit dar, um von dem On-Premise- auf das On-Demand-Modell umzustellen. Es ist dabei keine Transformation bzw. Anpassung der bestehenden Anwendung erforderlich. Lediglich die Art und Weise der Implementierung und Bereitstellung wird geändert. Dieser Ansatz bietet einige Vorteile, insbesondere in den frühen Phasen der Umstellung auf das SaaS-Modell, wenn die genaue Service-Nachfrage noch nicht bekannt ist.

- *Ebene II – Configurable*: Die zweite Ebene des Reifegradmodells setzt voraus, dass die Anwendung weiterhin dediziert als völlig unabhängige Software-Instanz bereitgestellt wird, zugleich aber Konfigurationsmöglichkeiten anbietet. Statt jede Instanz individuell an die Anforderungen einzelner Mandanten anzupassen, stellt der SaaS-Anbieter seinen Kunden die Möglichkeit bereit, mithilfe der Parametrisierung die Software-Instanz selbstständig zu konfigurieren. Da die Konfigurationsmechanismen direkt in der Anwendungslogik eingebettet sind, kann der SaaS-Anbieter alle Anwenderunternehmen mit dem gleichen Anwendungscode bedienen (vgl. Kulkarni et al. 2012, S. 14). Die vorgenommenen Anpassungen werden als mandantenspezifische Metadaten gesichert, so dass der Quellcode der Software weiterhin unbeeinträchtigt bleibt. Obwohl die Instanzen der einzelnen Mandanten auf der Code-Ebene gleich sind, sind sie weiterhin völlig voneinander isoliert.

  Der Übergang auf die zweite Reifegrad-Ebene ist in der Regel schon mit großen Veränderungen der Softwarearchitektur verbunden. Dies ist insbesondere dann so, wenn die Anwendung für die individuelle codebasierte Anpassung anstelle der für die metadatenbasierten Konfigurationen konzipiert wurde. Anderseits ermöglicht aber eine einheitliche Codebasis eine deutliche Reduzierung des Verwaltungsaufwands bei den Updates und Upgrades. Während auf der ersten Ebene des Reifegradmodells alle kundenangepassten Instanzen separat aktualisiert werden müssen, können die Instanzen dieser Ebene, aufgrund des gleichen Quellcodes gleichzeitig aktualisiert werden (vgl. Mietzner et al. 2011, S. 63).

- *Ebene III – Multi-Tenant-Efficient*: Auf der dritten Reifegrad-Ebene unterstützt eine Anwendungsinstanz mehrere Mandanten gleichzeitig und ermöglicht dabei eine dynamische Konfiguration anhand der Metadaten. Für die Anwender ist der Wechsel zwischen der zweiten und der dritten Stufe des Reifegradmodells nicht wahrnehmbar. Es ist nicht erkennbar, dass die gleiche Anwendungsinstanz mit anderen Mandanten gemeinsam benutzt wird. Auf der Anwendungsebene müssen jedoch entsprechende Mechanismen eingebaut werden, um diese Transparenz zu gewährleisten – insbesondere Mechanismen der Isolation der einzelnen Mandanten sowie Autorisierungs- und Sicherheitsmechanismen.

  Da eine einzelne Software-Instanz viele Mandanten gleichzeitig bedienen kann, ist eine weniger umfangreiche Hardwarearchitektur (im Vergleich zur zweiten Modellstufe) erforderlich. Dies spiegelt sich direkt in den reduzierten Wartungskosten wider. Ein wesentlicher Nachteil dieses Ansatzes ist die begrenzte Skalierbarkeit (vgl. Kulkarni et al. 2012, S. 14). Wenn die Anzahl der zusätzlichen Mandanten wächst, reicht eine Software-Instanz nicht aus, um alle eingehenden Mandanten-Anfragen effektiv zu verarbeiten. Der einzige Weg, um die Leistung der Anwendung zu verbessern, besteht in der vertikalen Skalierung der vorhandenen Hardwarearchitektur, d. h. in einem Umstieg auf einen leistungsfähigeren Server. Diese Skalierung hat aber auch ihre Grenze, ab welcher die Serverleistung nicht mehr kosteneffizient erhöht werden kann.

- *Ebene IV – Scalable*: Auf der höchsten Ebene des SaaS-Reifegradmodells ist die Anwendung konfigurierbar und mandantenfähig, genau wie in der vorherigen Ebene. In diesem Fall besitzt sie aber zusätzlich die Lastverteilungsmechanismen, welche für die dynamische Erzeugung der neuen Softwareinstanzen und für die Zuweisung der erforderlichen Ressourcen zu den einzelnen Mandanten-Aufgaben verantwortlich sind. Ein SaaS-System ist auf eine beliebig große Anzahl von Anwenderunternehmen skalierbar, da die Anzahl der Server und Anwendungsinstanzen, ohne Neugestaltung der Softwarearchitektur, an die bestehende Nachfrage anpasst werden kann (vgl. Chong und Carraro 2006). Die einzelnen mandantenbezogenen Daten und die konfigurierbaren Metadaten werden getrennt gesichert (vgl. Kulkarni et al. 2012, S. 15). Die Aktualisierungen bzw. Versionswechsel können für alle Mandanten der gleichen Instanz gleichzeitig durchgeführt werden (vgl. Koziolek 2011, S. 324).

Obwohl das dargestellte Reifegradmodell aufgrund der klar abgegrenzten, inkrementellen Struktur ein gut etabliertes Modell ist, weisen einige Wissenschaftler auf Mängel des Modells hin. Die meisten Kritikpunkte betreffen die unzureichende Detaillierung der architektonischen Konzepte einzelner SaaS-Reifegrade und die fehlende Differenzierung der Systemschichten (vgl. Kang et al. 2010). Im Gegensatz zu den anderen Reifegradmodellen konkretisiert das Modell von Chong und Carraro (2006) die Anpassungskonzepte einzelner SaaS-Reifen auf der Anwendungsebene, wodurch auch die Relevanz dieses Modells für die vorliegende Dissertation gegeben ist.

### 3.2.2 Ried-Reifegradmodell

Das Reifegradmodell nach Ried et al. (2008) differenziert die SaaS-Anwendungen in sechs Reifegrade – ausgehend von *Outsourcing (keine SaaS)* bis hin zu einer *dynamischen Geschäftsanwendungsplattform* (vgl. Ried et al. 2008). Dem Modell liegt die Annahme zugrunde, dass die Reife einer SaaS-Anwendung im Zeitablauf systematisch erhöht werden kann. Der Entwicklungsprozess einer SaaS gemäß dem Konzept von Ried wurde in der Abbildung 3.3 dargestellt.

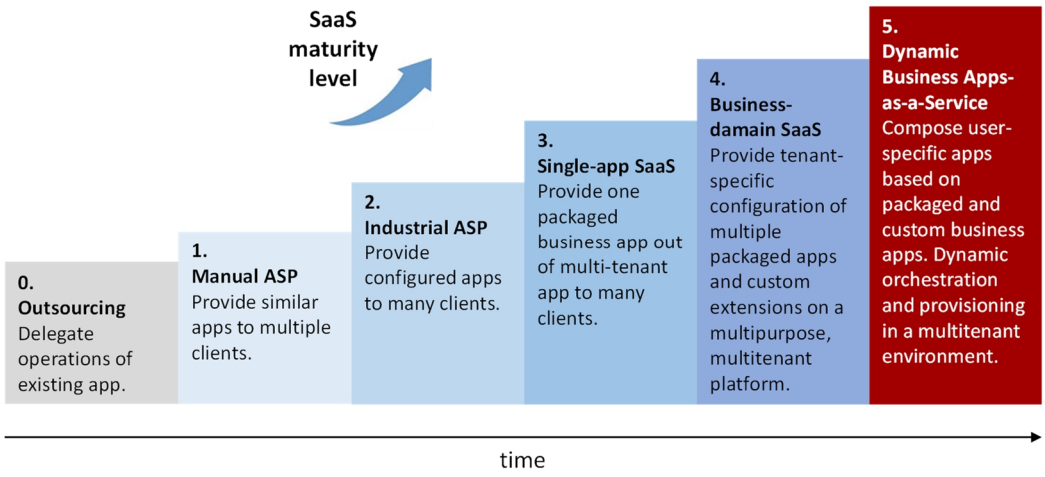



**Abbildung 3.3:** Entwicklungsprozess einer SaaS gemäß dem Reifegradmodell nach Ried et al. (2008)

Die detaillierten Charakteristika der einzelnen Reifegrad-Ebenen werden von Ried et al. wie folgt definiert:

- *Ebene 0 – Outsourcing* (keine SaaS): Im Rahmen des Outsourcing betreibt der Dienstleister eine Anwendung oder eine dedizierte Anwendungslandschaft für ein einziges Anwender-

unternehmen. Aufgrund der Tatsache, dass diese Anwendung für keinen anderen Kunden bereitgestellt werden kann, kann Outsourcing nicht als eine Form der SaaS betrachtet werden (vgl. Mell und Grance 2011).

- *Ebene 1 – Manual ASP*: Auf der ersten Reifegrad-Ebene bietet ein Hosting-Anbieter die Standardanwendungen für mehrere Anwenderunternehmen an. In der Regel wird für jedes Anwenderunternehmen ein dedizierter Server eingerichtet, auf dem seine Anwendungsinstanz installiert ist. Wie im Fall des On-Premise Betreibermodells kann die dedizierte Anwendungsinstanz in vollem Umfang an die Anforderungen des Anwenderunternehmens angepasst werden.
- *Ebene 2 – Industrial ASP*: Auf der zweiten Ebene verwendet ein Anwendungsdienstleister (ASP) eine komplexe IT-Management-Software, um identische Anwendungssoftwarepakete mit anwenderspezifischen Konfigurationen mehreren Anwenderunternehmen bereitzustellen. Die Anwendung stellt noch immer die gleiche Software dar, welche für das konventionelle On-Premise-Betreibermodell erstellt wurde.
- *Ebene 3 – Single-app SaaS*: Die Geschäftsanwendungen der dritten Reifegrad-Ebene werden durch die eingebauten SaaS-Fähigkeiten gekennzeichnet. Das webbasierte Konzept der Benutzeroberfläche (UI) und die Fähigkeit mehrere Anwender mit einer skalierbaren Infrastruktur zu bedienen, stellen die wichtigsten Merkmale dar. Die Anpassbarkeit ist auf die Konfiguration beschränkt.
- *Ebene 4 – Business-domain SaaS*: Auf der vierten Reifegrad-Ebene stellt der SaaS-Anbieter neben einer Geschäftsanwendung auch eine Plattform für zusätzliche Geschäftslogik bereit. Dies ermöglicht es, die alleinstehende Anwendung der vorherigen Ebene mit den SaaS-Lösungen der Drittanbieter und kundeneigenen Erweiterungen zu ergänzen. Das Modell erfüllt auch die Anforderungen der Großunternehmen, einen kompletten Geschäftsbereich in die Cloud migrieren zu können.
- *Ebene 5 – Dynamic Business Apps-as-a-Service*: Auf der fünften Ebene stellen die SaaS-Anbieter eine umfassende On-Demand-Anwendungs- und Integrationsplattform bereit, in der vorab bereits Geschäftsanwendungen und -dienste vorhanden sind. Die verfügbaren Komponenten können vom Anwender zu einem mandantendedizierten System zusammengestellt werden. Die daraus resultierende Anpassungsfähigkeit macht diese Lösung für alle, einschließlich großer Anwenderunternehmen, attraktiv (vgl. Ried et al. 2008).

Das sechsstufige Reifegradmodell nach Ried et al. (2008) ähnelt dem Modell von Chong und Carraro, dessen ausgereifte Stadien um die Funktionalität der PaaS erweitert wurden. Obwohl das Modell weit verbreitet und anerkannt ist, wird sein ausschließlicher Fokus auf die Anwendungsschicht bzw. die Vernachlässigung der anderen architektonischen Schichten als Nachteil bewertet (vgl. Kang et al. 2010). Im Unterschied zu den anderen Reifegradmodellen weist das Modell auch auf andere Anpassungskonzepte im Cloud-Bereich hin – insbesondere auf die komponentenbasierte Zusammenstellung und Erweiterung.

### 3.2.3 Kang-Reifegradmodell

Das Reifegradmodell nach Kang et al. (2010) kann als eine Zusammenstellung der zwei ursprünglich dargestellten SaaS-Reifegradmodelle gesehen werden, in deren Rahmen die zusätzliche Untergliederung der Service-Komponenten – System, Datenbank, Integration, Geschäftsmodell – vorgenommen wurde.

Die Abbildung 3.4 veranschaulicht das Entwicklungskonzept einer SaaS anhand einer Matrix, in welcher die aufeinanderfolgenden Entwicklungsstadien jeder Service-Komponente spalten-

weise aufsteigend dargestellt werden. Jede Zeile entspricht dabei einer Reifegrad-Ebene, welche aus vier Service-Bestandteilen besteht.

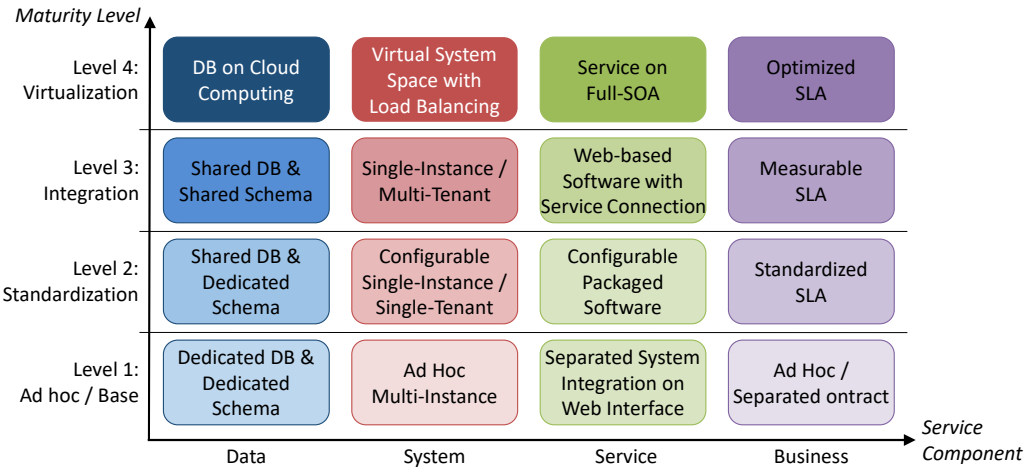



**Abbildung 3.4:** SaaS-Reifestufen gemäß dem Reifegradmodell nach Kang et al. (2010, S. 344)

Die vier in der Abbildung 3.4 zeilenweise dargestellten Reifestufen werden von Kang et al. (2010) wie folgt charakterisiert:

- *Ad hoc-Reife* bezeichnet eine Anwendung, die ad hoc in die private Cloud übernommen wurde, wobei der Anwender für das Hosting, die Administration der Software und die Benutzerbetreuung einen Anwendungsdienstleister auf Basis eines individuellen Mietvertrages bezahlt. Diese Reife charakterisiert sich durch eine dedizierte Anwendungsinstanz und eine dedizierte Datenbank mit einem dedizierten Schema, die für jeden Mandanten bereitgestellt werden. Die Integrationsfähigkeit der Anwendung erfolgt größtenteils auf der Präsentationsebene.
- *Standardisierung-Reife* erfordert, dass die Anwendung jedem Mandanten als eine unabhängige, identische, konfigurierbare Anwendungsinstanz bereitgestellt werden kann. Im Rahmen dieses Ansatzes verwenden die Mandanten ein dediziertes Datenschema, welches in einer mandantenübergreifenden Datenbank angelegt ist. Die einzelnen Systemkomponenten können ausgewählt und zusammengestellt werden. Die Abrechnung erfolgt anhand einer individuellen Dienstleistungsvereinbarung für die ausgewählten Systemkomponenten.
- *Integration-Reife* charakterisiert eine Anwendung, die von mehreren unabhängigen Mandanten gemeinsam genutzt wird. In diesem Ansatz wird von allen Mandanten eine einzige Systeminstanz gemeinsam genutzt, welche jeweils separat an die individuellen Anforderungen getrennt angepasst werden kann. Der Anwendung liegt eine gemeinsame Datenbank mit einem gemeinsamen Datenschema zugrunde, in welcher die Daten aller Mandanten gespeichert werden. Die Erweiterung der Systemfunktionalität kann durch die Integration der externen Webdienste vorgenommen werden. Die Abrechnung erfolgt auf Basis einer messbaren, standardisierten Dienstleistungsvereinbarung.
- *Virtualisierung-Reife* kennzeichnet eine Anwendung, bei der die Anwendungsinstanz und die ihr zugrunde liegende Datenbank mit einem Datenschema von allen Mandanten gemeinsam betrieben werden. Dabei folgen die Daten- und Anwendungsschicht den Prinzipien einer verteilten Systemarchitektur. Die benötigten Funktionalitäten können von den einzelnen Mandanten ausgewählt, zusammengestellt und entsprechend ihrer Anforderungen konfiguriert werden. Im Vergleich zu der Integration-Reife ist dieser Reifegrad mit den

entsprechenden Mechanismen ausgestattet, welche die Rechenleistung bzw. den virtuellen Speicherplatz dem Mandanten dynamisch und anhand des aktuellen Bedarfs zuweisen. Die Abrechnung erfolgt anhand der tatsächlich genutzten Dienstleistung, welche quantifiziert werden kann (vgl. Kang et al. 2010).

Das dargestellte Reifegradmodell nach Kang et al. (2010) liefert mehr Bezug auf die Integrationsfähigkeit und die Anpassungsfähigkeit der einzelnen Reifegrade. Es ergänzt die bisherigen reifebezogenen Charakteristika um die Eigenschaften der Datenschicht und deckt damit die Lücken der bisherigen Modelle ab.

### 3.2.4 Kernmerkmale der cloudbasierten Systeme

Trotz der Tatsache, dass die gut etablierten Reifegradmodelle den Transformationsprozess einer nicht ausgereiften in eine ausgereifte SaaS-Anwendung anhand der unterschiedlichen Anzahl der Reifegrade beschreiben und deren Fokus oft auf unterschiedliche Aspekte gelegt wird, sind die architektonischen Kernmerkmale einzelner Reifegrade gut identifizierbar.

Die Tabelle 3.1 stellt das Ergebnis der komparativen Analyse der drei Reifegradmodelle dar. Die einzelnen Reifestufen wurden für jedes der Modelle aufsteigend von der am wenigsten bis hin zu der am meisten ausgereiften Anwendungsarchitektur sortiert.

Die Reifegrade von Modellen, welche ähnliche Charakteristika aufweisen, wurden zeilenweise gegenübergestellt. Die ersten drei Spalten enthalten die Bezeichnung der nachfolgenden, modellbezogenen Reifestufen, während die vier weiteren Spalten auf das Vorhandensein konkreter, reifebezogener Architekturmerkmale hindeuten.

**Tabelle 3.1:** SaaS-Reifegradmodelle im Überblick

| **Bezeichnung der SaaS-Reife** | | | **Architekturmerkmale** | | | |
|---|---|---|---|---|---|---|
| Ried et al. (2008) | Chong und Carraro (2006) | Kang et al. (2010) | Cloud-fähigkeit | Konfigurier-barkeit | Mandanten-fähigkeit | Skalier-barkeit |
| Outsourcing | — | — | — | — | — | — |
| Manual ASP | Ad hoc / Custom | Ad hoc | ✓ | — | — | — |
| Industrial ASP | Configurable | Standardization | ✓ | ✓ | — | — |
| Single-app SaaS | Multi-Tenant-Efficient | Integration | ✓ | ✓ | ✓ | — |
| Business-domain SaaS | Scalable | — | ✓ | ✓ | ✓ | ✓ |
| Dynamic Business Apps-as-a-Service* | — | Virtualization* | ✓ | ✓* | ✓ | ✓ |

**Legende:** ✓ Merkmal vorhanden; — Merkmal nicht vorhanden; * Merkmal auf der SaaS und PaaS vorhanden

Basierend auf der Vergleichsanalyse einzelner Reifestufen der drei ursprünglich beschriebenen SaaS-Reifegradmodelle können vier grundlegende Merkmale der Architektur einer SaaS-Anwendung identifiziert werden:

- *Cloudfähigkeit* stellt die grundlegende Eigenschaft einer cloudbasierten Anwendung dar, welche sicherstellt, dass diese Anwendung in einem Cloud-Bereitstellungsmodell unter Berücksichtigung der Voraussetzungen des Cloud-Computing-Ansatzes betrieben werden kann (vgl. Kapitel 3.1.1).
- *Konfigurierbarkeit* ist eine Softwareeigenschaft, welche es dem System erlaubt, an die Bedürfnisse und Anforderungen der Anwender mittels Einstellung der vordefinierten Parameter angepasst zu werden. Im Fall der Kang- und Ried-Reifegradmodelle wird die Konfigurierbarkeit der jeweils höchsten Reifegrad-Ebene auch auf der Plattformebene sichergestellt und bezieht sich in beiden Fällen auf eine anwenderindividuelle Zusammenstellung einzelner Systemfunktionalitäten, welche vom Systemanbieter als web-basierte Dienste zur Verfügung gestellt werden.
- *Mandantenfähigkeit* wird als eine Eigenschaft der SaaS-Anwendung definiert, welche sicherstellt, dass mehrere unabhängige Anwender (Mandanten) mit den gemeinsam genutzten infrastrukturellen Ressourcen gleichzeitig unterstützt werden können.
- *Skalierbarkeit* stellt die Softwareeigenschaft dar, welche eine Möglichkeit der dynamischen Anpassung der Systemleistung im Fall der Auslastungsschwankungen sicherstellt, sodass den aktuellen Leistungsanforderungen ohne Neugestaltung der Anwendung entsprochen werden kann (vgl. Chong und Carraro 2006; Shao 2011, S. 20).

Die identifizierten SaaS-Merkmale treten bei allen analysierten Reifegradmodellen in gleicher Reihenfolge auf und tragen zum Erreichen der vollständigen Reife bei. Im Hinblick auf die Tabelle 3.1 können somit die Anwendungen in der Cloud als nicht ausgereift bzw. ausgereift bezeichnet werden.

Eine ausgereifte SaaS-Anwendung ist diejenige, deren Architektur zugleich alle Merkmale der SaaS-Architektur erfüllt. Werden einige SaaS-Eigenschaften nicht bzw. nicht komplett erfüllt, so ist die Anwendung als eine nicht ausgereifte SaaS-Anwendung einzustufen.

Die Sicherstellung der identifizierten Merkmale im Rahmen des architektonischen Konzeptes eines cloudbasierten ERP-Systems kann Einfluss auf die anwendbaren Customizing-Konzepte haben und dadurch auf die erzielbare Systemanpassbarkeit einwirken. Im nächsten Kapitel werden die Zusammenhänge zwischen den identifizierten Architekturmerkmalen und der Anpassbarkeit der cloudbasierten ERP-Systeme untersucht.

## 3.3 Einfluss der SaaS-Merkmale auf das Customizing

Die möglichen Auswirkungen der einzelnen architektonischen Merkmale einer dem SaaS-Modell ausgesetzten Anwendung auf die erzielbare Anpassungsfähigkeit werden getrennt im Rahmen der nachfolgenden Unterkapitel untersucht.

### 3.3.1 Cloudfähigkeit

Im Rückblick auf die vorgenommene Analyse der Reifegradmodelle soll die *Cloudfähigkeit* als die grundlegende Eigenschaft jeder SaaS verstanden werden. Die Cloudfähigkeit stellt eine Softwareeigenschaft dar, welche es der Anwendung erlaubt, in der Cloud betrieben und dem Anwender in Form einer Dienstleistung über das Internet bereitgestellt zu werden. Eine Anwendung kann dabei in der Cloud unter Verwendung unterschiedlicher Cloud-Bereitstellungsmodelle und Service-Betreibermodelle ausgeführt und betrieben werden (vgl. Kapitel 1.2.1). Jedes der genannten Modelle kann Einfluss auf die Anwendbarkeit der einzelnen Customizing-Ansätze haben.

*Cloud-Bereitstellungsmodelle*

Unabhängig von dem Cloud-Bereitstellungsmodell wird die Software, welche die Grundlage des SaaS-Dienstes bildet, auf der Anbieterseite installiert. Aus diesem Grund erhalten die kundeninternen Administratoren (die sogenannten Anwendungsexperten) im Fall der cloudbasierten Systeme meist keinen direkten Zugang zum Backend-Bereich – im Gegensatz zu den konventionellen On-Premise-Systemen.

Die öffentlichen Clouds werden zwecks Sicherstellung einer gemeinsamen Nutzung der infrastrukturellen Ressourcen durch mehrere unabhängige Kundenunternehmen bereitgestellt. Dieses Cloud-Bereitstellungsmodell bildet in der Regel eine Grundlage der ausgereiften, hoch standardisierten Systeme, welche durch weitere SaaS-Eigenschaften, insbesondere durch die Mandantenfähigkeit, gekennzeichnet sind.

Die Sicherstellung einer gemeinsamen Ressourcennutzung in einer öffentlichen Cloud wirkt sich negativ auf die Realisierung unterschiedlicher Systemanpassungen mehrerer Anwenderunternehmen aus (vgl. Hofmann und Woods 2010, S. 93). Aufgrund der Tendenz zur Sicherstellung einer einheitlichen Codebasis ist der Einsatz von codebasierten Customizing-Ansätzen oftmals limitiert bzw. erfordert eine Anwendung von alternativen Konzepten. In diesem Kontext erweitern die SaaS-Anbieter ihr Angebot um das PaaS-Modell, um zusätzliche Adaptionsmöglichkeiten ihrer Systeme in öffentlichen Clouds sicherzustellen.

Im Gegensatz zu dem öffentlichen Cloud-Bereitstellungsmodell werden die privaten Clouds durch ein einzelnes Kundenunternehmen verwendet (vgl. Mell und Grance 2011, S. 2). Dies ermöglicht eine wesentlich größere Flexibilität beim Einsatz unterschiedlicher Anpassungstechniken und -konzepte, da die Adaption der Anwendung ohne Auswirkung auf andere Mandanten bleibt. Die privaten Clouds lassen die Anwendung von codebasierten Customizing-Ansätzen zu, mit deren Hilfe die Codebasis modifiziert bzw. erweitert werden kann. Folglich kann auch das bereitgestellte System extensiver als in der öffentlichen Cloud an die Anforderungen des Anwenderunternehmens angepasst werden.

Obwohl die Systeme in den privaten Clouds in der Regel eine umfangreichere Anpassbarkeit bereitstellen können, müssen die dafür vorgesehenen Customizing-Werkzeuge nicht notwendigerweise in Form der On-Demand-Selfservice dem Anwenderunternehmen zur Verfügung gestellt werden. In Abhängigkeit von der verfolgten Customizing-Strategie des Systemanbieters können jedoch einige Kompetenzen hinsichtlich der Systemanpassung auf die Anwender transferiert werden (vgl. Seethamraju 2015, S. 479), soweit die dafür verfügbaren Werkzeuge vorhanden sind.

Daneben werden auch die Mischformen der beiden Cloud-Bereitstellungsmodelle erlaubt, welche unter dem Begriff *Hybrid Cloud* zusammengefasst werden. Im Rahmen eines hybriden Bereitstellungsmodells werden die einzelnen Komponenten des ERP-Systems in diversen Cloud-Bereitstellungsmodellen angeboten. Die Differenzierung der Betreibermodelle einzelner Komponenten kann auch unterschiedlich erfolgen. Es können z. B. unternehmenskritische Komponenten in einer privaten Cloud betrieben werden, während die anderen Systembestandteile in einer öffentlichen Cloud bereitgestellt werden (vgl. Johansson et al. 2014, S. 10). Eine andere Option stellt die Differenzierung nach den architektonischen Schichten dar: die Anwendungsschicht in einer Private Cloud und die Datenschicht in einer Public Cloud bzw. umgekehrt (vgl. Brown und Nyarko 2012, S. 54; Jamshidi et al. 2013, S. 150).

Die hybriden Bereitstellungsmodelle stellen eine flexible Zwischenlösung dar, welche es dem Anwenderunternehmen erlaubt, von den Eigenschaften des öffentlichen als auch des privaten Bereitstellungsmodells zu profitieren. Die hybriden Clouds ermöglichen es, auch einige Begrenzungen im Customizing der öffentlichen Clouds zu umgehen. Es können z. B. die stan-

dardisierteren Anwendungskomponenten weiterhin in einer Public Cloud betrieben werden, während die anderen individualisierten Komponenten in einer Private Cloud bereitgestellt werden können.

Die Elemente einer Anwendung in einer hybriden Cloud können zumeist in unterschiedlichem Umfang an die Anforderungen eines Anwenderunternehmens angepasst werden. Während die Komponenten in der öffentlichen Cloud in der Regel weniger individuelle Anpassung ermöglichen, können die Systemkomponenten in einer Private Cloud in der Regel besser an die Anforderungen des Anwenderunternehmens adaptiert werden.

*Service-Betreibermodelle*

Die drei grundlegenden Service-Betreibermodelle (vgl. Kapitel 1.2.1) können Einfluss auf die anwendbaren Anpassungskonzepte ausüben. Die Abhängigkeit der systembezogenen Anpassbarkeit von dem Service-Betreibermodell wurde in der Abbildung 3.5 visualisiert.

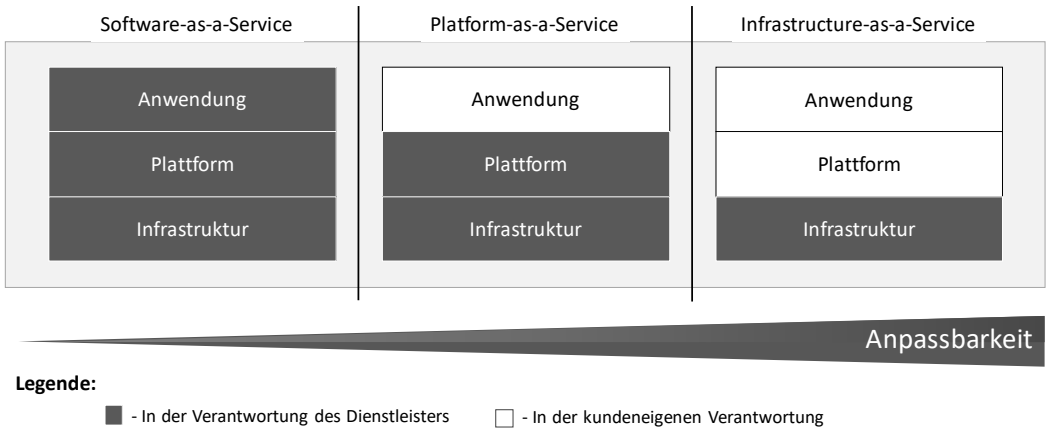



**Abbildung 3.5:** Anpassbarkeit einer SaaS-Anwendung in Abhängigkeit von dem Betreibermodell in Anlehnung an Jouanne-Diedrich et al. (2012, S. 56)

Beim IaaS-basierten Customizing können fast alle potenziell auftretenden Adaptionen umgesetzt werden (vgl. Schmid und Rummler 2012, S. 166). Customizing auf der IaaS-Ebene ermöglicht jedem Anwenderunternehmen, eine dedizierte Lösung basierend auf den kundenangepassten virtuellen Maschinen zur Verfügung zu stellen.

Das IaaS-Betreibermodell kann besonders dann von Vorteil sein, wenn mehrere extrem individualisierte Systeme mit stark differenzierter Funktionalität unterschiedlichen Anwenderunternehmen bereitgestellt werden müssen. Dies gilt auch, wenn einige umfangreiche Installations- bzw. Integrationsprozesse weitere dedizierte Anpassungen an der dem Service zugrunde liegenden Plattform erforderlich machen.

Die PaaS ermöglicht im Vergleich zu IaaS keine Adaptionen der virtualisierten Hardware (z. B. Databank- oder Anwendungsservers). Sie stellt aber weiterhin umfangreiche Customizing-Möglichkeiten zur Verfügung. Die in einer PaaS bereitgestellten Anwendungen können unter anderem mittels Inanspruchnahme der codebasierten Customizing-Ansätze modifiziert bzw. erweitert werden. Somit kann jedem Anwenderunternehmen eine individuell angepasste Anwendung zur Verfügung gestellt werden.

In einem SaaS-Betreibermodell kann eine bereitgestellte Anwendung mithilfe bereits eingebauter Werkzeuge angepasst werden. Im Vergleich zu den zwei ursprünglich genannten Ansätzen sind die Anpassungsmöglichkeiten im SaaS-Modell sehr begrenzt.

Zusammenfassend kann man sagen, dass je tiefer die durch Customizing betroffene Ebene in der Struktur der Betreibermodelle liegt (IaaS stellt die tiefste Ebene dar), desto umfangreichere Customizing-Aufgaben können umgesetzt werden und desto mehr potenzielle Anforderungen des Kundenunternehmens können erfüllt werden (vgl. Bedner 2013, S. 51; Schmid und Rummler 2012, S. 169).

Die Bereitstellung umfangreicher Customizing-Möglichkeiten, die z. B. auf der Infrastruktur- bzw. Plattformebene erfolgt, steht oft im Widerspruch mit den Geschäftsstrategien der SaaS-Anbieter. Aus der Perspektive eines SaaS-Anbieters stellt die Bereitstellung einer standardisierten Lösung an eine möglichst große Anzahl von Anwenderunternehmen die gewünschte Option dar, die es ihm erlaubt, die Infrastruktur-Ressourcen durch Nutzung der Skaleneffekte besser als bei den stark individualisierten Lösungen auszuschöpfen. Daneben ermöglicht es eine einheitliche Codebasis, die künftigen Wartungsarbeiten effizienter durchzuführen. Aus Sicht des SaaS-Anbieters können größere Skaleneffekte erreicht werden, je tiefer die Standardisierung geht.

Bemerkenswert bleibt jedoch, dass unabhängig von dem Bereitstellungskonzept stets ein bestimmter Flexibilitätsumfang des ERP-Systems seitens der Kundenunternehmen erwartet wird (vgl. Lechesa et al. 2012). Wie intensiv die Individualisierung durch den SaaS-Systemanbieter zugelassen wird, hängt von dessen Geschäftsstrategie und dem entsprechenden Marktsegment ab (vgl. Guo et al. 2007). Auf den Trade-off zwischen der Standardisierung und der Sicherstellung einer individuellen Systemadaption wird in der Literatur zu SaaS oft eingegangen.

Im Hinblick auf die mögliche Zusammenstellung grundlegender Betreiber- und Bereitstellungsmodelle ergeben sich mehrere gängige Bereitstellungskonzepte, in welchen eine cloudbasierte Anwendung bereitgestellt werden kann (vgl. Münzl et al. 2009, S. 36).

### 3.3.2 Konfigurierbarkeit

Eine konfigurierbare Anwendung stellt eine Software dar, bei welcher die anwenderkonformen Anpassungen in der Laufzeit mittels der Parametrisierung umgesetzt werden können. Die Parametrisierung wird in der Literatur oft als der bevorzugte Customizing-Ansatz der cloudbasierten ERP-Systeme präsentiert, da sie im Gegensatz zu den codebasierten Anpassungstechniken die Realisierung anderer architektonischen Eigenschaften des SaaS-Modells nicht beeinträchtigt.

Das Konzept zum Erlangen einer ausgereiften gut skalierbaren SaaS-Anwendung besteht darin, eine möglichst hohe Systemadaptierbarkeit durch eine umfangreiche Parametrisierbarkeit sicherzustellen. Die Parametrisierbarkeit kann somit als eine Maßnahme zur Erreichung der erforderlichen Standardisierung einer SaaS-Anwendung interpretiert werden. Die Sicherstellung eines standardisierten Quellcodes ermöglicht es dem Systemanbieter, einen Versionswechsel bzw. die Wartungs- und Aktualisierungsarbeiten effektiver als bei mehreren individuell angepassten Anwendungsinstanzen durchzuführen. Eine ideale Lösung aus Sicht des SaaS-Modells stellt eine vollständig standardisierte Anwendung dar, die an alle oft gegensätzlichen Anforderungen mehrerer Anwenderunternehmen mittels Parametrisierung angepasst werden kann.

Die Sicherstellung umfangreicher Konfigurierbarkeit erfordert die Implementierung mehrerer Varianten der Systemfunktionalität im Quellcode und kann zur Steigerung des Umfangs und der Komplexität einer Anwendung führen (vgl. Kurbel 2016, S. 317 f.). Die Umsetzung aller potenziell auftretenden Systemanpassungen in Form eines parametrisierbaren Systemumfangs ist meist nicht möglich. Laut Guo et al. (2007) muss der Bereitstellung eines konfigurierbaren Systemumfangs eine Analyse der Customizing-Anforderungen des gezielten Kundensegments vorausgehen. Danach muss entschieden werden, welche der potenziell auftretenden Customizing-Aufgaben in dem Anwendungscode als Parameter implementiert werden können.

Die Untersuchungen von Sun et al. (2008) bestätigen, dass je umfangreicher der Parametrisierungsumfang der Anwendung ist, desto weniger müssen andere Anpassungstechniken angewendet werden und eine umso umfangreichere Standardisierung des cloudbasierten Systems kann erreicht werden (vgl. Sun et al. 2008, S. 19).

Die Sicherstellung umfangreicher Parametrisierbarkeit bei einer ursprünglich für den On-Premise-Betrieb gedachten Anwendung kann allerdings mit einigen schwerwiegenden Aufgaben zur Umgestaltung der Systemarchitektur verbunden sein, insbesondere dann, wenn für bestimmte Anpassungsaufgaben ursprünglich andere Customizing-Ansätze vorgesehen wurden (vgl. Chong und Carraro 2006).

Obwohl bereits die Umgestaltung einiger Anpassungskonzepte bei nicht ausgereiften Cloud-Systemen problematisch sein kann, stellt die Sicherstellung einer entsprechenden Flexibilität bei den mandantenfähigen Anwendungen eine noch größere Herausforderung dar. In einem mandantenbasierten Szenario müssen zusätzlich entsprechende Mechanismen zur Trennung der Systemeinstellungen unabhängiger Mandanten gewährleistet werden. Der oft in der Praxis verwendete Ansatz besteht in der Sicherstellung der entsprechenden Konfigurierbarkeit, die in der Laufzeit anhand der mandantendedizierten Metadaten erfolgt. Laufzeitbezogen bedeutet dies, dass die Einstellungen einzelner Mandanten erst bei Bedarf (z. B. Aktivierung einer angepassten Systemseite etc.) aus den Metainformationen in der Anwendungslaufzeit eingelesen werden.

Im Hinblick auf das cloudbasierte System-Betreibermodell ermöglicht die Konfigurierbarkeit eine zusätzliche Unabhängigkeit des Anwenderunternehmens von dem SaaS-Anbieter bei der Durchführung einiger Systemanpassungen. Bei konfigurierbaren Systemen können die Systemanpassungen im Rahmen der anwenderseitigen Anpassungsprozesse vorgenommen werden und es muss theoretisch keine Inanspruchnahme des Supports seitens der Systemanbieter erfolgen. Auf diese Art und Weise wird dem On-Demand-Selfservice, als kennzeichnende Eigenschaft des Cloud-Computing-Ansatzes, weiter nachgekommen.

### 3.3.3 Mandantenfähigkeit

Mandantenfähigkeit ist eine wesentliche Architektureigenschaft der SaaS-Anwendung, welche ihr erlaubt, mehrere unabhängige Mandanten mit den gemeinsam benutzten infrastrukturellen Ressourcen gleichzeitig zu unterstützen (vgl. Guo et al. 2007, S. 551). Als Mandant wird jedes Anwenderunternehmen verstanden, das die Dienste von einem SaaS-Anbieter bezieht (vgl. Kabbedijk und Jansen 2011, S. 153).

Die Mandantenfähigkeit ist nicht mit konkreten Systemressourcen verbunden und kann auf unterschiedlichen architektonischen Ebenen sichergestellt werden (vgl. Krebs et al. 2012, S. 426 f.; Osipov et al. 2009). Aus Sicht eines cloudbasierten ERP-Systems erlangen jedoch zwei Architekturschichten besondere Bedeutung: die Anwendungsschicht und die Datenschicht (vgl. Kabbedijk et al. 2014, S. 204 f.). Auf diesen kann die Mandantenfähigkeit unabhängig voneinander und mithilfe von unterschiedlichen Ansätzen gewährleistet werden (vgl. z. B. Kwok und Mohindra 2008; Schiller et al. 2011).

*Mandantenfähigkeit der Anwendungsschicht*

Die Literatur bietet drei unterschiedliche Konzepte an, welche zur Sicherstellung der Mandantenfähigkeit für die Anwendungsschicht benutzt werden können. Diese sind:

- *Mandantendedizierter Anwendungsserver*: stellt einen Ansatz dar, in dem jedem Mandanten eine dedizierte Anwendungsinstanz bereitgestellt wird, welche auf einem dedizierten (virtualisierten) Anwendungsserver läuft (vgl. Momm und Krebs 2011, S. 142).

- *Mandantenübergreifender Anwendungsserver*: bildet die Grundlage eines Ansatzes, in dessen Rahmen jedem Mandanten eine dedizierte Anwendungsinstanz auf einem mandantenübergreifenden Anwendungsserver zur Verfügung gestellt wird (vgl. Kabbedijk et al. 2014; Walraven et al. 2016, S. 3).
- *Mandantenübergreifende Anwendungsinstanz*: stellt ein Konzept dar, in welchem auf einem Anwendungsserver eine einzige Anwendungsinstanz läuft, die gemeinsam durch mehrere unabhängige Mandanten benutzt wird (vgl. Chong und Carraro 2006).

Die Darstellung der oben genannten Ansätze zur Sicherstellung der Mandantenfähigkeit auf der Anwendungsschicht fasst die Abbildung 3.6 zusammen.

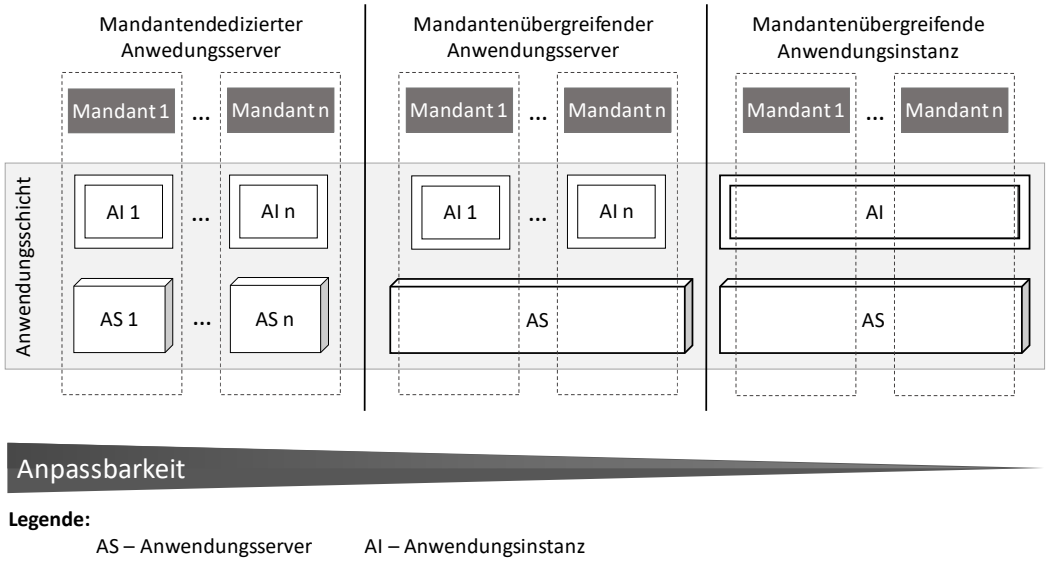



**Abbildung 3.6:** Ansätze zur Sicherstellung der Mandantenfähigkeit der Anwendungsschicht in Anlehnung an Walraven et al. (2011, S. 372)

Das Konzept eines mandantendedizierten (virtuellen) Anwendungsservers (siehe Abbildung 3.6, links) erfordert keinerlei Umgestaltung der konventionellen On-Premise Anwendung in Bezug auf die Mandantenfähigkeit. Die Virtualisierung erlaubt es jedoch nicht, die infrastrukturellen Ressourcen effizient zu nutzen. Aufgrund der physischen Trennung der mandantendedizierten Anwendungsinstanzen auf der Server-Ebene kann jede Instanz an die Anforderungen des Anwenderunternehmens in der Regel in vollem Umfang angepasst werden.

Der nächste Ansatz entsteht durch die Sicherstellung der Mandantenfähigkeit direkt auf der Anwendungsserver-Ebene (siehe Abbildung 3.6, Mitte). Die Anwendungsinstanzen einzelner Mandanten laufen unabhängig voneinander auf einem mandantenfähigen Anwendungsserver und sind weiterhin vollständig mandantenkonform anpassbar. Obwohl dieser Ansatz in der Regel keine Umgestaltung einer Basisanwendung erfordert, müssen einige Serverdienste überarbeitet werden, um mehrere Anwendungsinstanzen parallel unterstützen zu können. Dabei sind die Herausforderungen in Bezug auf die Mandanten-Isolation von besonderer Bedeutung.

Die Mandantenfähigkeit auf der Ebene der Anwendungsinstanz (siehe Abbildung 3.6, rechts) stellt die größte Herausforderung bei der Entwicklung von cloudbasierten Systemen dar. Das Konzept besteht in der Integration von den Isolationsmechanismen direkt in der Anwendung, so dass alle Mandanten auf einer einzigen Anwendungsinstanz betrieben werden können (vgl. Fiaidhi et al. 2012, S. 16).

Das Modell, in welchem die Mandantenfähigkeit mithilfe einer Virtualisierungstechnik gewährleistet wird, kann eher als ein Zwischenstand bei der Entwicklung einer „echten“ mandantenfähigen Anwendung betrachtet werden (vgl. Krebs et al. 2012, S. 427).

Die Durchführbarkeit bestimmter Systemanpassungen bzw. die Anwendbarkeit einzelner Customizing-Ansätze ist durch die Möglichkeit der Sicherstellung erforderlicher Isolation der individuell anzupassenden Systembestandteile einzelner Mandanten bedingt. Die Untersuchungen von Kabbedijk et al. (2014) bestätigen, dass mit steigender Isolation der Mandanten die Anpassbarkeit der cloudbasierten Systeme ansteigen kann.

Anwendungen, deren dedizierte Instanzen die einzelnen Mandanten bedienen, können in der Regel extensiver an die individuellen Anforderungen angepasst werden (vgl. Mietzner et al. 2011, S. 63 ff.). Aufgrund der physischen Isolation der mandantendedizierten Anwendungsinstanzen können die individuellen Anpassungen direkt im Quellcode vorgenommen werden. Bei den mandantenfähigen Anwendungen sind die codebasierten Customizing-Ansätze nur schwer anwendbar, da der standardisierte Quellcode mehrere Mandanten unterstützt (vgl. Chong und Carraro 2006).

Daher müssen von den SaaS-Anbietern andere Mechanismen bereitgestellt werden, um eine mandantenfähige Anwendungsinstanz für die einzelnen Anforderungen unterschiedlicher Mandanten simultan anpassen zu können. Die Simultanität ist hier besonders entscheidend und bedeutet, dass die Anwendungsinstanz in ihrer Laufzeit für verschiedene Mandanten unterschiedliche Zustände (mandantenangepasste Ansicht und Funktionalität) gleichzeitig annehmen muss. Mit anderen Worten, jeder Mandant sollte die Vorstellung haben, dass er an einem dedizierten System arbeitet, während tatsächlich alle Mandanten eine einzige Anwendungsinstanz gemeinsam verwenden (vgl. Schmid und Rummler 2012, S. 168). Die Literatur bietet unterschiedliche Konzepte an, welche das Problem des laufzeitbasierten Customizing lösen sollen (vgl. z. B. Walraven et al. 2011; Mietzner et al. 2011; Al-Shardan und Ziani 2015, Arya et al. 2010). Sie weist aber unzureichend auf die Umsetzbarkeit dieser Lösungsvorschläge bei den hochkomplexen Systemen wie z. B. den ERP-Systemen hin.

*Mandantenfähigkeit der Datenschicht*

Ein mandantenfähiges System muss neben der Mandantenfähigkeit auf der Anwendungsebene auch die Mandantenfähigkeit der Datenschicht sicherstellen. Trotz der Tatsache, dass alle Mandanten die gleiche Software verwenden, kann die Nachfrage nach den gespeicherten Dateninhalten aufgrund der unterschiedlichen Geschäftsprozesse und Organisationsstrukturen voneinander abweichen. Daher muss die Datenbank entsprechende Erweiterungsmöglichkeiten anbieten, welche eine effiziente Organisation der mandantenrelevanten Geschäftsdaten gewährleisten. Die Modifikation des Datenmodells durch einen Mandanten soll die Datenmodelle der anderen Anwenderunternehmen nicht beeinflussen. Die Umsetzung der erforderlichen Isolation zwischen den Mandanten und zugleich die Sicherstellung der Flexibilität des Datenmodells in einem mandantenfähigen Szenario stellt eine zusätzliche Herausforderung im Vergleich zu den dedizierten nicht mandantenfähigen Systemen dar.

Die Mandantenfähigkeit auf der Datenbankebene kann mittels unterschiedlicher Ansätze realisiert werden, welche jeweils ein unterschiedliches Niveau der Datenisolation sicherstellen:

- *mandantendedizierte Datenbank*: Im Rahmen eines einzelnen mandantenübergreifenden Datenbankservers wird für jeden Mandanten eine getrennte Datenbank erstellt,
- *mandantendediziertes Datenbankschema*: Dieses stellt ein Konzept dar, in dem auf einem Datenbankserver eine mandantenübergreifende Datenbank angelegt wird, die gemeinsam durch mehrere unabhängige Mandanten benutzt wird,

- *mandantenübergreifendes Datenbankschema*: Bei diesem Konzept wird auf einem Datenbankserver eine einzige Datenbank mit einem mandantenübergreifenden Datenbankschema angelegt. Die Trennung der Mandantendaten erfolgt auf der Ebene der Datenbanktabelle (vgl. Bezemer und Zaidman 2010b, S. 89; Jacobs und Aulbach 2007, S. 517 ff.).

Die Abbildung 3.7 stellt die Visualisierung der oben genannten Ansätze zur Sicherstellung der Mandantenfähigkeit auf der Datenschicht dar.

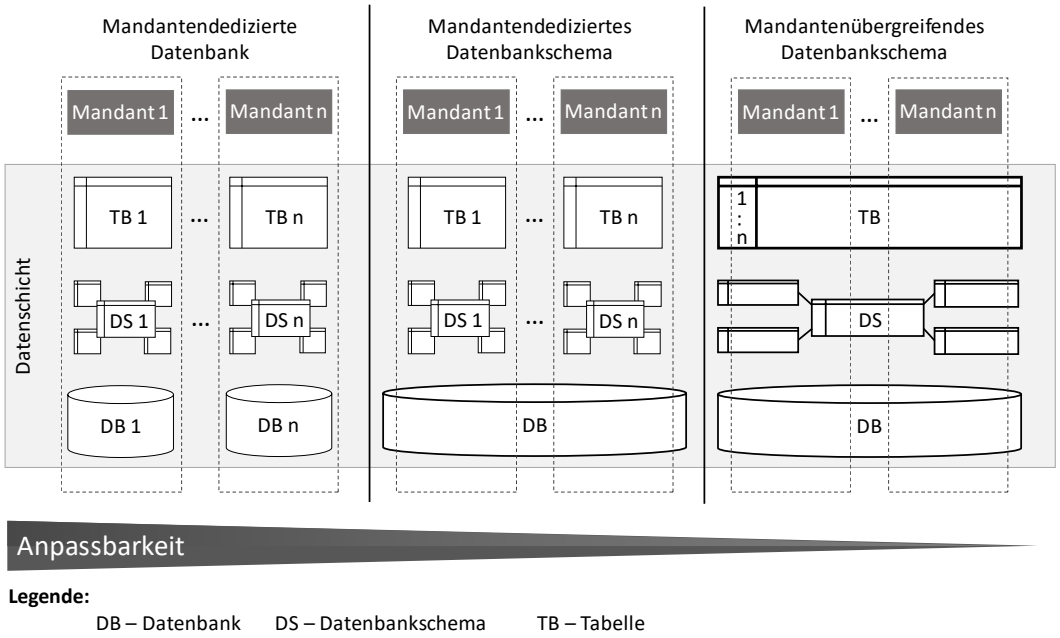



**Abbildung 3.7:** Ansätze zur Sicherstellung der Mandantenfähigkeit der Datenschicht in Anlehnung an Chong et al. (2008)

Das Konzept einer *mandantendedizierten Datenbank* bietet vergleichend die größte Anpassbarkeit der Datenebene aus Sicht eines Mandanten und die zugleich höchste Datenisolation (siehe Abbildung 3.7, links). Innerhalb dieses Ansatzes sind sowohl die Tabellen als auch die Datenschemata einzelner Mandanten frei anpassbar bzw. erweiterbar (vgl. Chong et al. 2008). Aufgrund der physischen Isolierung auf der Datenebene können die Backup- und Wiederherstellungsprozesse bei einzelnen Mandanten unabhängig voneinander durchgeführt werden.

Trotz seiner relativ einfachen Umsetzbarkeit – keine Quellcode-Änderungen erforderlich im Vergleich zu dem On-Premise-Modell – ermöglicht die mandantendedizierte Datenbank keine effiziente Nutzung der infrastrukturellen Ressourcen (vgl. Heng et al. 2012, S. 199). Da die Anzahl der Datenbanken, die auf einem einzelnen Datenbankserver laufen können, begrenzt ist, erhöht eine zunehmende Mandantenzahl die Anzahl der erforderlichen Server. Dies spiegelt sich in den steigenden Wartungskosten des gesamten Systems wider.

Eine *gemeinsame Datenbank* für alle Mandanten stellt einen anderen Ansatz zur Sicherstellung der Mandantenfähigkeit auf der Ebene des Datenbankservers dar. Bei der Umsetzung dieses Konzeptes sind zwei Szenarien denkbar: getrennte, *mandantendedizierte Datenbankschemata* oder ein gemeinsames, *mandantenübergreifendes Schema*.

Das Konzept der getrennten Datenbankschemata ermöglicht es, die Datenbestände unterschiedlicher Mandanten in separaten, mandantendedizierten Tabellen innerhalb einer gemeinsam be-

nutzten Datenbank zu speichern (siehe Abbildung 3.7, Mitte). Die Datensicherheit kann in diesem Ansatz direkt auf der Datenbankebene durch die Einstellung der mandantendedizierten Zugriffsrechte zu einem bestimmten Schema gewährleistet werden. Die Isolation der mandantenbezogenen Daten in diesem Ansatz ist jedoch geringer als im Fall der physisch getrennten Datenbanken. Die Anwendung eines Standardschemas bietet die Möglichkeit, in diesem Modell auch standarisierte SQL-Abfragen für alle Mandanten anzuwenden.

Dieser Ansatz ähnelt dem Konzept der dedizierten Datenbanken hinsichtlich der Einfachheit der Realisierung und bezüglich der Flexibilität des Datenmodells, welches zusätzlich um weitere Tabellen bzw. Tabellenfelder erweitert werden kann (vgl. Heng et al. 2012, S. 199). Es wird jedoch auch die Portierung einer höheren Mandantenzahl auf einem Datenbankserver ermöglicht. Dies wiederum reduziert die modellbezogenen Wartungskosten. Bemerkenswert bleibt jedoch, dass die schwächere Isolation und damit auch die niedrigere Datensicherheit durch dieses Konzept entstehen, was durch die Implementierung zusätzlicher Sicherheitsmechanismen auf der Anwendungsebene kompensiert werden müssen.

Das Konzept eines gemeinsamen Datenbankschemas bietet vergleichbar geringe Anpassungsmöglichkeiten an, welche lediglich eine Erweiterung der bestehenden Tabellen um eine vordefinierte bzw. konfigurierbare Anzahl der mandantendefinierbaren Felder ermöglichen (vgl. Bezemer und Zaidman 2010b, S. 89). Jede mandantenübergreifende Tabelle wird um eine zusätzliche Spalte erweitert, in welcher die Mandantennummer gespeichert wird (vgl. Chong und Carraro 2006). Die Daten-Tupel des einzelnen Mandanten sind durch diese Spalte eindeutig identifizierbar (siehe Abbildung 3.7, rechts).

Das Konzept des gemeinsamen Datenbankschemas ist durch die effektivste Ressourcennutzung im Vergleich zu den anderen Ansätzen gekennzeichnet (vgl. Schiller et al. 2011, S.125 ff.). Obwohl das Konzept eine Effizienzsteigerung der Ressourcennutzung ermöglicht, erfordert es sowohl einen zusätzlichen Programmieraufwand zur Sicherstellung erforderlicher Datenisolation- und Sicherheitsmechanismen (vgl. Chong et al. 2008) als auch die Implementierung dedizierter Backup- und Wiederherstellungskonzepte (vgl. Heng et al. 2012, S. 199).

Im Hinblick auf die konventionellen On-Premise-Systeme erfordert dieser Ansatz in der Regel eine umfangreiche Umgestaltung des Anwendungscodes zur Erreichung einer Übereinstimmung mit dem mandantenfähigen Datenschema. Aufgrund dessen wird dieser Ansatz hauptsächlich bei ausgereiften Systemen verwendet, die von Anfang an gemäß den Anforderungen des gemeinsamen Datenbankschemas entwickelt wurden.

### 3.3.4 Skalierbarkeit

Die Skalierbarkeit stellt ein Architekturmerkmal einer mandantenfähigen SaaS-Anwendung dar. Dieses Merkmal stellt sicher, dass die zusätzlichen Systemressourcen proportional zu den Anforderungen einzelner Mandanten bereitgestellt werden, so dass für die Bearbeitung der eingehenden Anfragen einzelner Mandanten ein akzeptables Leistungsniveau gewährleistet werden kann (vgl. Shao 2011, S. 20).

Die Systemleistung kann mittels *vertikaler* und *horizontaler* Skalierung gesteigert werden. Die *vertikale Skalierung* stellt ein Verfahren dar, in dem zusätzliche Hardware-Ressourcen (z. B. Datenträger, Prozessoren, Arbeitsspeicher etc.) zu einer bestimmten Systemkomponente (Server) hinzugefügt werden. Dieser Ansatz erfordert keinerlei Umgestaltung der Anwendung. Er ist aber stark durch die Verfügbarkeit der erforderlichen Hardware-Komponenten bzw. deren Preise beschränkt. Im Gegensatz dazu wird bei der *horizontalen Skalierung* die Systemleistung durch Hinzufügen weiterer Systemkomponenten (Server) gesteigert. Dieser Ansatz hat keine Hardware-bezogenen Beschränkungen. Er erfordert jedoch eine entsprechende Implementierung der Mechanismen der dynamischen Lastverteilung im Anwendungscode.

Bei den cloudbasierten Anwendungen stellt die horizontale Skalierung in der Regel die beste Option zur Sicherstellung der erforderlichen Systemleistung bei den Auslastungsschwankungen dar (vgl. Chong und Carraro 2006). Das erreichbare Niveau der Skalierbarkeit hängt von den architektonischen Konzepten ab, die bei der Entwicklung der Anwendung implementiert werden. Daneben spielen auch die umgesetzten Customizing-Konzepte und -Strategien eine Rolle. Tabelle 3.2 stellt eine Übersicht der unterschiedlichen Customizing-Strategien und der erzielbaren Skalierbarkeit des cloudbasierten Systems dar.

**Tabelle 3.2:** Zusammenhänge zwischen den Customizing-Strategien und der Skalierbarkeit im SaaS-Modell nach Guo et al. (2007)

| Bezeichnung | Customizing-Strategie | Skalierbarkeit |
|---|---|---|
| Modell A: *Native Design* | Der Anbieter stellt für den Mandanten die erforderlichen webbasierten Anpassungswerkzeuge und Programmierschnittstellen (APIs) zur Ermöglichung selbstständiger Systemanpassung sicher.<br>Der SaaS-Anbieter führt keine mandantendedizierten Änderungen des Quellcodes durch. | Sehr groß |
| Modell B: *Smooth Evolvement* | Der Anbieter modifiziert den Anwendungscode gemäß den individuellen Anforderungen einzelner Mandanten.<br>Der Anbieter benutzt entsprechende Werkzeuge zur Verwaltung der Systemanpassungen, um die Anpassungskosten pro Mandant zu reduzieren. | Gering bis mittelmäßig |
| Modell C: *Pulse Evolvement* | Der Anbieter modifiziert den Anwendungscode, wenn die Anpassungen durch entsprechend große Mandantenzahl benötigt werden. | Mittelmäßig bis groß |
| Modell D: *Failure Management* | Der Anbieter modifiziert den Anwendungscode gemäß den individuellen Anforderungen einzelner Mandanten.<br>Der Anbieter verfügt über keine Werkzeuge zur effektiven Verwaltung der Systemanpassungen, um die Anpassungskosten pro Mandanten zu reduzieren. | Gering |

Eine hohe Skalierbarkeit kann bei einer stark standardisierten Anwendung erreicht werden, welche die Wiederverwendbarkeit einzelner Komponenten durch mehrere Mandanten sicherstellt und sich gut parallelisieren lässt (vgl. Andrikopoulos et al. 2013, S. 522). Allerdings soll eine mandantenfähige SaaS-Anwendung zugleich die erforderliche Flexibilität bereitstellen, welche die mandantenindividuelle Anpassung der Anwendungsinstanz ermöglicht. Die mandantendedizierten Anpassungen müssen dabei ausschließlich für ein bestimmtes Anwenderunternehmen wirksam sein und dürfen somit die Einstellungen der anderen Mandanten nicht beeinflussen. Dies macht die Implementierung entsprechender Isolationsmechanismen nötig, welche eine Trennung der Adaptionen einzelner Mandanten sicherstellen.

Wie stark die Isolation sein muss, hängt von den einzelnen Adaptionsansätzen ab, welche für die Durchführung konkreter Customizing-Aufgaben vorgesehen sind. Während im Fall der Parametrisierung eine Isolation der mandantenabhängigen Systemeinstellungen auf der Tabellenebene meist ausreichend ist, erfordern die anderen Customizing-Ansätze die Implementierung von zusätzlichen Isolationsmechanismen auf der Dienst- bzw. Plattformebene (vgl. Schmid und Rummler 2012, S. 168).

Wie Guo et al. (2011, S. 19 f.) sowie Brown und Nyarko (2012, S. 68) zeigen, verringert sich die Möglichkeit einer effektiven Skalierung der SaaS-Anwendung mit dem Anstieg der individuellen Anpassungen, welche für den einzelnen Mandanten direkt im Anwendungscode vorgenommen werden. Durch die codebasierte Systemmodifikation steigt der Umfang der nicht wiederverwendbaren Systembestandteile, der entsprechend von dem standardisierten und wiederverwendbaren Anwendungscode isoliert werden muss. Mit steigenden Mandantenzahlen stellt die Sicherung entsprechender Skalierbarkeit und mandantendedizierter Anpassungsfähigkeit im SaaS-Modell eine besondere Herausforderung dar. Es besteht somit aus Sicht einer mandantenfähigen Anwendungsinstanz ein Trade-off zwischen der Anpassungsfähigkeit und der Skalierbarkeit.

## 3.4 Bedeutung der architektonischen Einflussfaktoren

Im Hinblick auf die aufgestellte Hauptforschungsfrage HF 2 wurde in diesem Kapitel gezeigt, dass die Architektur einer cloudbasierten Anwendung durch vier grundlegende Eigenschaften – Cloudfähigkeit, Konfigurierbarkeit, Mandantenfähigkeit und Skalierbarkeit – gekennzeichnet werden soll (vgl. Kapitel 3.2.4). Die einzelnen SaaS-Merkmale können im Rahmen des architektonischen Konzeptes mittels diverser Ansätze und in unterschiedlichem Umfang umgesetzt werden. Im Hinblick auf die Reifegradmodelle kann das Cloud-ERP-System in Abhängigkeit von den realisierten Merkmalen als eine ausgereifte bzw. nicht ausgereifte SaaS betrachtet werden.

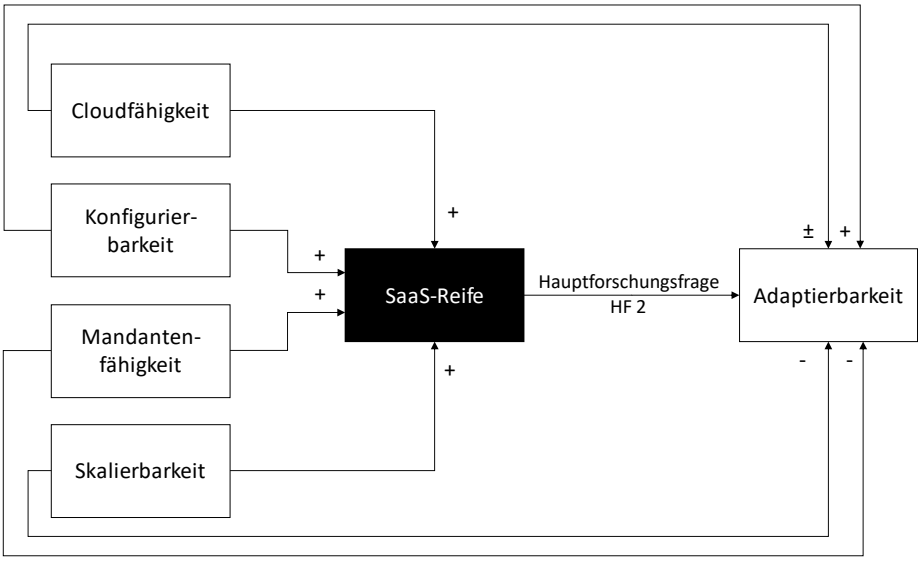
Der Reifeaspekt ermöglicht es, eine Vergleichbarkeit architektonisch unterschiedlicher, cloudbasierter ERP-Systeme sicherzustellen. Dabei sei angemerkt, dass die Zielausrichtung auf den höchsten Reifegrad keine ideale Strategie für jeden Systemhersteller bzw. -anbieter sein muss (vgl. Brown und Nyarko 2012, S. 58; Chong und Carraro 2006; Kapitel 3.1.2). Vielmehr lassen die Forschungsergebnisse von Scavo et al. (2012) einige Differenzen in der Realisierung einzelner Merkmale durch die cloudbasierten ERP-Systeme erwarten.

Die identifizierten SaaS-Architekturmerkmale können einen direkten bzw. indirekten Einfluss auf die systembezogene Adaptierbarkeit ausüben. Im Rahmen der vorgenommenen Analyse wurden die Vorzeichen der untersuchten Zusammenhänge bzw. Auswirkungen auf die Adaptierbarkeit der cloudbasierten Systeme ermittelt.

Die Abbildung 3.8 stellt die im Rahmen des vorliegenden Kapitels identifizierten Zusammenhänge zwischen den einzelnen Architekturmerkmalen und der Flexibilität der cloudbasierten ERP-Systeme dar. Die einzelnen Beziehungen wurden als gerichtete Pfeile dargestellt. Das an der Pfeilspitze stehende Vorzeichen (+ oder –) bezieht sich auf die Richtung des Zusammenhanges.

Die Sicherstellung weitgehender Skalierbarkeit eines cloudbasierten Systems wirkt sich in der Regel negativ auf die Anwendbarkeit der codebasierten Customizing-Konzepte aus und beeinträchtigt somit die erzielbare Systemadaptierbarkeit. Ein ähnlich negativer Zusammenhang zeigt sich bei der Mandantenfähigkeit. Im Gegensatz dazu ermöglicht der Anstieg der Konfigurierbarkeit eine bessere Anpassung an die Anforderungen des Anwenderunternehmens. Der Einfluss der Cloudfähigkeit auf die Adaptierbarkeit ist auf das Zusammenspiel zwischen dem Service-Betreibermodell und dem Cloud-Bereitstellungsmodell zurückzuführen. Daraus können sowohl Einbußen als auch der Anstieg der Adaptierbarkeit resultieren.

Die einzelnen architektonischen Merkmale führen gemäß der Reifegradmodelle (vgl. Kapitel 3.2) zum Erreichen einer vollständigen Übereinstimmung mit dem SaaS-Modell, weshalb ein positiver Zusammenhang zwischen jedem Merkmal und der SaaS-Reife entsteht.




**Legende:** + positive Relation, - negative Relation, ± keine eindeutige Relation

**Abbildung 3.8:** Zusammenhänge zwischen den SaaS-Architekturmerkmalen und der Adaptierbarkeit eines cloudbasierten Systems

Obwohl die vorgenommene literaturbasierte Einflussanalyse auf Zusammenhänge zwischen den vier architektonischen Merkmalen und der systembezogenen Adaptierbarkeit hindeutet, erlaubt es die getrennte Untersuchung einzelner Zusammenhänge noch nicht, allgemeingültige und weitreichende Erkenntnisse im Hinblick auf die Adaptierbarkeit der cloudbasierten ERP-Systeme aufzustellen.

Die Architektur eines Cloud-ERP-Systems kann die vorgestellten Architekturmerkmale in unterschiedlichem Ausmaß realisieren. Aus diesem Grund ist es notwendig, ergänzend zu der vorgenommenen separaten Untersuchung einzelner Merkmale die Auswirkung des gesamten architektonischen Konzepts auf die erzielbare Adaptierbarkeit der cloudbasierten ERP-Systeme zu untersuchen.

# 4 Adaptierbarkeit der cloudbasierten ERP-Systeme

Die Analysen der vorigen Kapitel haben gezeigt, dass eine adäquate Bewertung und Systematisierung der Adaptierbarkeit von cloudbasierten ERP-Systemen eine komplexe Aufgabe darstellt, die eine vielseitige Betrachtung differenzierter Systemaspekte erfordert. Im Rahmen dieses Kapitels wird ein Systematisierungsmodell erstellt, welches die für die ganzheitliche und transparente Analyse der Untersuchungsproblematik erforderlichen Betrachtungsebenen zusammenstellt und ihren Detaillierungsgrad definiert. Im Anschluss wird ein strukturiertes Literaturstudium vorgenommen, das den aktuellen Forschungstand auf dem Gebiet der Flexibilität der cloudbasierten ERP-Systeme widrgibt und die Charakteristika und Defizite der bisherigen Untersuchungen im Kontext der Forschungsziele dieser Dissertation aufzeigt. Basierend auf der intensiven Inhaltsanalyse werden die wichtigsten Erkenntnisse der bisherigen Studien entlang der einzelnen Betrachtungsebenen des konzipierten Systematisierungsrahmens zusammengestellt und die Hypothesen für die bevorstehende explorative Untersuchung aufgestellt.

## 4.1 Systematisierungsrahmen

Das Hauptziel dieser Dissertation besteht in der Systematisierung und Bewertung der Flexibilität von cloudbasierten ERP-Systemen. Im Rahmen der aufgestellten Customizing-Definition (siehe Abschnitt 2.1) werden hinsichtlich der aufgestellten Forschungsfragen die Customizing-Strategien mehrerer cloudbasierten ERP-Systeme untersucht und die Zusammenhänge zwischen der bereitgestellten Flexibilität und dem architektonischen Systemkonzept aufgezeigt.

Um eine ganzheitliche Betrachtung der untersuchten Problematik sicherstellen zu können, werden in dem Forschungsprojekt differenzierte Betrachtungsebenen eingesetzt und miteinander kombiniert. Zur Sicherstellung der erforderlichen Transparenz bei den Analysen und Untersuchungen der Folgekapitel wurde ein Systematisierungsmodell entwickelt, das alle Betrachtungsebenen, welche für die Beantwortung der aufgestellten Hauptforschungsfragen nötig sind, gemäß dem aufgestellten Untersuchungskonzept einordnet und ihren Detaillierungsgrad eindeutig definiert. Die Abbildung 4.1 präsentiert eine schematische Darstellung des entwickelten Systematisierungsmodells und seiner einzelnen Betrachtungsebenen.

Der in der Mitte der Abbildung 4.1 dargestellte Kreis repräsentiert den Untersuchungsgegenstand, also die Flexibilität eines ERP-Systems. Der bereitgestellte Flexibilitätsumfang ist durch die Customizing-Strategie des Systemherstellers bestimmt. Die Customizing-Strategie wird in der Entwurfsphase des ERP-Systems festgelegt und beschreibt, wie das cloudbasierte System an die Anforderungen des Anwenderunternehmens angepasst werden kann. Sie definiert u. a., in welchem Umfang die einzelnen Systemelemente adaptiert werden können, mit welchen Customizing-Ansätzen die Flexibilität sichergestellt wird und welche Adaptionswerkzeuge zur Umsetzung der Adaptionsanforderungen bereitgestellt werden. Im Hinblick auf die drei Betrachtungsebenen, welche entsprechend auf die Adaptionsansätze, adaptierbare Systemelemente und Adaptionswerkzeuge ausgerichtet sind, bietet das konzipierte Modell entsprechende Typologien bzw. Kategorisierungen, die eine detaillierte und mehrdimensionale Systematisierung der gewonnenen Erkenntnisse bezüglich einzelner Customizing-Strategien sicherstellen. Dadurch leistet das Systematisierungskonzept einen besonderen Beitrag zur Steigerung der

Analysegenauigkeit und ermöglicht einen präziseren Vergleich differenzierter Customizing-Strategien mehrerer cloudbasierter ERP-Systeme.

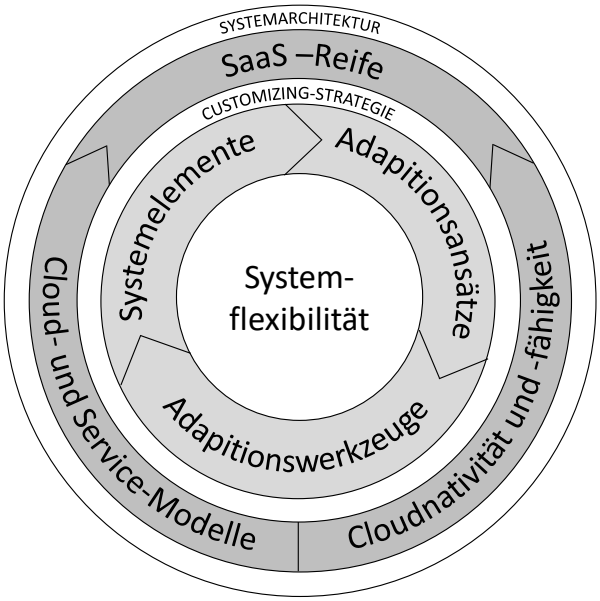



**Abbildung 4.1:** Systematisierungsmodell

Wie bereits im dritten Abschnitt gezeigt wurde, ist die bereitgestellte Flexibilität einer cloudbasierten Anwendung zusätzlich durch das ausgewählte architektonische Systemkonzept beeinflusst. Die Analyse der Zusammenhänge zwischen der Systemarchitektur und -flexibilität stellt einen weiteren wichtigen Aspekt der Untersuchung dar. In diesem Kontext wird durch die Einbeziehung der SaaS-Reife, welche die architekturbezogene Konformität mit den Anforderungen des SaaS-Modells beschreibt, in das Systematisierungsmodell eine Grundlage für die Auswertung der Zusammenhänge zwischen dem architektonischen Konzept und der Customizing-Strategie mithilfe der quantitativen Methoden geschaffen. Die Verwendung unterschiedlicher Betrachtungsebenen in Verbindung mit dem Reifekonzept ermöglicht es, die systemübergreifenden Tendenzen hinsichtlich der bereitgestellten Adaptierbarkeit zu identifizieren und die architekturbedingten Differenzen in dem bereitgestellten Flexibilitätsumfang aufzuzeigen.

Die einzelnen Betrachtungsebenen des dargestellten Systematisierungsmodells werden in den nachfolgenden Unterabschnitten dieses Kapitels näher beschrieben.

### 4.1.1 Adaptionsansätze

Die implementierten Adaptionsansätze stellen einen der wesentlichen Aspekte der Customizing-Strategie dar. Sie determinieren unter anderem, wie und mit welchem Aufwand die Individualisierung einzelner Systemelemente erfolgt, und bestimmen z. B., ob eine Änderung der Systemeinstellungen oder eine Neuentwicklung erforderlich ist.

Obwohl das Customizing im Allgemeinen in codefreie und codebasierte Adaptionen unterteilt werden kann, erfordert eine umfassende Analyse der bereitgestellten Customizing-Strategien eine viel detailliertere Betrachtung der bereitgestellten Adaptionsansätze. Eine solche Untersuchung kann eine ausführliche Antwort auf die gestellte Forschungsfrage (FF 1.1) liefern.

Aus diesem Grund wird den Untersuchungen dieses Forschungsprojektes eine erweiterte Typologie der Customizing-Ansätze in Anlehnung an Kurbel (2016) zugrunde gelegt (vgl. Kapitel 2.1), die eine detailliertere Betrachtung der differenzierten Customizing-Formen ermöglicht.

Die einzelnen Customizing-Ansätze wurden in der Tabelle 4.1 zusammengestellt.

**Tabelle 4.1:** Customizing-Ansätze in Anlehnung an Kurbel (2016)

| Ansatz | Beschreibung | Codierung |
|---|---|---|
| Komposition | Zusammenstellung eines individuellen Satzes von vordefinierten Systemkomponenten, die zusammen eine individuelle Lösung bilden. | CA1 |
| Parametrisierung | Anpassung der Systemfunktionalität und der Datenstrukturen durch Zuweisung von Parameterwerten zu den vorgegebenen Systemparametern. | CA2 |
| Modellbasierte Generierung | Automatische Erzeugung von Programmcode- und / oder Datenbankschemata auf Basis eines Informationsmodells. Anpassungen erfolgen auf der Ebene des Informationsmodells, welches als Eingabe für die Codeerzeugung dient. | CA3 |
| User-Exits | Erweiterung der Systemfunktionalität durch Einbetten eines benutzerdefinierten Codes an den vordefinierten Stellen im Systemcode eines ERP-Systems (sogenannte User-Exits – auch als Erweiterungspunkte bezeichnet). | CA4 |
| Programmierschnittstellen (API) | Codebasierte Entwicklung von Systemerweiterungen, welche die vordefinierten Systemkomponenten auf alternative Art und Weise wiederverwenden. | CA5 |
| Codebasierte Erweiterung | Codebasierte Entwicklung neuer Komponenten außerhalb des ERP-Systems. Die neuentwickelten Komponenten stellen die im Standardsystem nicht enthaltene Funktionalität zur Verfügung. | CA6 |
| Codebasierte Modifikation | Direkte Veränderung des Systemquellcodes. | CA7 |

Die Spalten der Tabelle beinhalten aufeinanderfolgend die Bezeichnung und Beschreibung des Customizing-Ansatzes sowie die Codierung, die als Referenz in weiteren Abschnitten der Arbeit verwendet wird.

### 4.1.2 Adaptierbare Systemelemente

Die Flexibilität eines ERP-Systems ist durch die Adaptierbarkeit einzelner Systemelemente bestimmt. Im Hinblick auf die Bewertung des bereitgestellten Flexibilitätsumfangs der cloudbasierten ERP-Systeme ist daher eine umfassende Analyse der Adaptionsmöglichkeiten einzelner Systemelemente erforderlich.

Aufgrund der unternehmerischen Vielfalt wäre es nicht möglich, im Rahmen dieses Forschungsprojektes alle potenziell geforderten Systemadaptionen zu betrachten. Stattdessen wird der Fokus auf die Adaptionsanforderungen gelegt, die oftmals bei der Implementierung eines ERP-Systems entstehen.

Um einen entsprechenden Detaillierungsgrad und zugleich eine erforderliche Transparenz der Untersuchung auf dieser Betrachtungsebene sicherzustellen, wurde aufbauend auf den Erkenntnissen des Kapitels 2.4 der Anforderungskatalog erstellt, in dem die häufig auftretenden Adaptionsanforderungen zusammenstellt sind. Die einzelnen Adaptionsanforderungen wurden zunächst anhand des betroffenen Systemelements in neun Kategorien gruppiert (siehe Anhang A.2).

Als Ergebnis der vorgenommenen Kategorisierung lassen sich die in der Tabelle 4.2 aufgelisteten Systemelemente konkretisieren, die eine Betrachtungsgrundlage für die Untersuchungen der Folgekapitel schaffen.

**Tabelle 4.2:** Systemelemente und Adaptionsanforderungen

| Systemelement | Adaptionsanforderungen | Codierung |
|---|---|---|
| Zusammenstellung der Systemmodule | Zusammenstellen eines individuellen Systemumfangs aus den vordefinierten Systemkomponenten (Module). | SE1 |
| Parametereinstellungen | Anpassung der visuellen und funktionalen Systemaspekte durch Änderung der Parameterwerte der vordefinierten Systemparameter. | SE2 |
| Benutzeroberfläche | Adaption der Benutzeroberfläche entsprechend den Anforderungen des Anwenderunternehmens. Dies umfasst u. a. die Anpassung an das Corporate Design und die betriebliche Nomenklatur, Neupositionierung oder Aktivierung / Deaktivierung von Eingabefeldern und anderen Oberflächenkomponenten. | SE3 |
| Geschäftsobjekte | Definition individueller Datenstrukturen und / oder Erweiterung der standardisierten Geschäftsobjekte mit neuen Datenfeldern. | SE4 |
| Analytische Komponenten | Definition der neuen Kennzahlen, Diagramme und Berichte. Erweiterung der standardisierten Übersichtsseiten, Dashboards und der vordefinierten Datenquellen. | SE5 |
| Organisationsstruktur | Anpassung der systeminternen Aufbauorganisation an die bestehende Unternehmensstruktur des Anwenderunternehmens. | SE6 |
| Geschäftsprozesse | Adaption der standardisierten Geschäftsabläufe, Geschäftsprozesse und Prozeduren gemäß den Anforderungen des Anwenderunternehmens. | SE7 |
| Anwendungsintegration | Integration von Geschäftsanwendungen und Webdiensten in das ERP-System bzw. die Ermöglichung eines beidseitigen Datenaustausches. | SE8 |
| Kundenspezifische Funktionalität | Entwicklung individueller funktionaler Erweiterungen und ihre Integration in das Basissystem. | SE9 |

Die zwei ersten Spalten der Tabelle beinhalten die Bezeichnung des Systemelements und die Zusammenstellung der wichtigsten Adaptionsanforderungen, die das jeweilige Element betreffen. Die letzte Spalte zeigt die Codierung, über die auf das einzelne Systemelement bzw. die mit ihm korrespondierende Zusammenstellung der Adaptionsanforderungen in den weiteren Abschnitten der Arbeit referenziert wird.

### 4.1.3 Adaptionswerkzeuge

Die Adaptionswerkzeuge stellen das zentrale Element dar, welches determiniert, wie der Anwender das ERP-System eigenständig adaptieren kann. Im Hinblick auf die Forschungsfrage FF 1.3 ist nicht nur interessant, wie die einzelnen Werkzeuge den Anwendungsexperten bereitgestellt werden, sondern auch, wie effizient die Adaptionen mithilfe dieser Werkzeuge umgesetzt werden können.

Die einzelnen Werkzeuge sind zur Durchführung konkreter Adaptionen vorgesehen. Die Adaptionsmöglichkeiten einzelner Werkzeuge und deren Verfügbarkeit werden durch die verfolgte Customizing-Strategie des Systemanbieters bestimmt. Die Systemanbieter stellen oft unter-

schiedliche Werkzeuge an differenzierte Anwendergruppen bereit, in Abhängigkeit z. B. von den vorhandenen Programmierkenntnissen oder dem systembezogenen Know-how. Die vorliegende Arbeit fokussiert die anwenderseitigen Administratoren bzw. Anwendungsexperten als Zielgruppe und ist somit auf die Analyse der für diese Anwendergruppe bereitgestellten Werkzeuge ausgerichtet.

Im Kontext der cloudbasierten Anwendungen sind bei der Bereitstellung der Customizing-Werkzeuge unterschiedliche Konzepte denkbar. Durch die Analogie zur Werkzeug-Typologie von Weidenhaupt (2001) können die Adaptionswerkzeuge unter anderem nach dem Integrationsgrad mit dem Basissystem unterteilt werden. Die vorgeschlagene Unterteilung ist in der Tabelle 4.3 dargestellt.

**Tabelle 4.3:** Adaptionswerkzeuge

| Werkzeug-Kategorie | Charakteristik | Codierung |
|---|---|---|
| Integrierte Adaptionswerkzeuge | Adaptionswerkzeuge sind direkt im Systemrahmen vorhanden und über die Benutzeroberfläche zugänglich. Sie stellen einen integralen untrennbaren Systembestandteil dar. | AW1 |
| Externe lokal installierbare Adaptionswerkzeuge | Systemexterne Customizing-Werkzeuge werden oft optional als komplementäre Software bereitgestellt. Die Installation ist auf einem lokalen Rechner erforderlich. | AW2 |
| Externe cloudbasierte Adaptionswerkzeuge | Systemexterne Adaptionswerkzeuge werden ähnlich wie das Basissystem in der Cloud bereitgestellt. | AW3 |

Hinsichtlich der Bereitstellung systemexterner Customizing-Werkzeuge sind einige Differenzen zwischen der typischen On-Demand- und On-Premise-Software zu erkennen (vgl. Weidenhaupt 2001, S. 44).

Bedingt durch die Architektur der cloudbasierten Anwendung kann ein systemexternes Werkzeug entweder als PaaS-basiertes Werkzeug oder als lokal installierbares Werkzeug bereitgestellt werden. Während die PaaS-basierten systemexternen Tools über das Internet zur Verfügung gestellt werden, werden die lokalen systemexternen Werkzeuge heruntergeladen und auf einem lokalen Rechner des Anwenders installiert.

Für eine effiziente Durchführung der Systemadaptionen spielt jedoch nicht der Integrationsgrad mit dem Basissystem, sondern viel mehr die im Werkzeug integrierten Konzepte der Anwenderunterstützung eine wichtige Rolle.

In Tabelle 4.4 findet sich diesbezüglich eine Übersicht solcher Mechanismen der Anwenderunterstützung, welche oft im Konzept eines Werkzeuges verankert sind.

Die dargestellten Konzepte der Klassifikation der anwenderseitigen Adaptionswerkzeuge werden als Grundlagen für die Analysen verwendet, die systematisch zur Beantwortung der Forschungsfrage FF 1.3 beitragen.

**Tabelle 4.4:** Mechanismen der Anwenderunterstützung in Anlehnung an Weidenhaupt (2001)

| Konzept | Beschreibung | Codierung |
|---|---|---|
| Passive Beratung | Der Benutzer erhält auf Anfrage zusätzliche Informationen über den aktuellen Zustand des Adaptionsprozesses und gegebenenfalls kontextabhängige Ratschläge zur Beibehaltung der Konformität mit den Prozessvorgaben. | AU1 |
| Aktive Prozessanleitung | Bei der Durchführung einzelner Adaptionsaktivitäten werden dem Anwendungsexperten bei den einzelnen Prozessschritten automatisch kontextabhängige Ratschläge gegeben und relevante Informationen zur Verfügung gestellt. | AU2 |
| Prozesslenkung | Die Ausführung des Customizing-Prozesses erfolgt in voller Konformität mit den Vorgaben der unterstützenden Mechanismen. In Abhängigkeit von dem aktuellen Prozesszustand erhält der Benutzer Zugriff auf eine relevante Teilmenge der verfügbaren Werkzeugdienste und Datenobjekte, mit deren Hilfe die nächste Benutzeraktion vorgenommen werden soll. | AU3 |
| Prozessautomation | Eine automatische Durchführung eines Teils des Customizing-Prozesses erfolgt ohne die direkte Intervention des Benutzers. | AU4 |

### 4.1.4 SaaS-Reife

Der Aspekt der Systemarchitektur ist im Hinblick auf die Hauptforschungsfrage HF 2 besonders signifikant. Wie bereits im dritten Kapitel angedeutet wurde, können die architektonischen Merkmale einer SaaS-Anwendung erhebliche Auswirkungen auf die Anwendungsmöglichkeiten der einzelnen Customizing-Konzepte und -Ansätze haben. Aus diesem Grund soll bei der Untersuchung eine Differenzierung der cloudbasierten ERP-Systeme anhand ihrer architektonischen Konzepte vorgenommen werden.

Im Hinblick auf die Hauptforschungsfrage HF 2 stellt das Reifekonzept, welches die architekturbezogene Konformität mit den Anforderungen des SaaS-Modells beschreibt, eine adäquate Grundlage für eine architekturbezogene Differenzierung der cloudbasierten ERP-Systeme dar.

Abgeleitet von der vorgenommenen Analyse der Reifegradmodelle für SaaS-Anwendungen (siehe Kapitel 3.2) wird im Rahmen des Untersuchungsprojekts eine Differenzierung in vier architektonische SaaS-Reifegrade vorgenommen. Die Aufteilung und Charakteristika einzelner Reifeebenen sind in der Tabelle 4.5 veranschaulicht.

Da die vorliegende Dissertation als Schwerpunkt das Customizing von ERPaaS thematisiert, werden alle ERP-Systeme, die nicht cloudfähig sind, von der weiteren Analyse ausgeschlossen. Den cloudbasierten ERP-Systemen, die im Rahmen eines On-Demand-Selfservices nicht konfigurierbar sind, wird der erste SaaS-Reifegrad der vierwertigen Reifegradskala zugewiesen. Diejenigen konfigurierbaren, cloudbasierten Systeme, welche keine Mechanismen der Mandantenfähigkeit besitzen, bzw. nur mandantenfähig auf der Anwendungsserver-Ebene sind (sogenannte Semi-Mandantenfähigkeit, vgl. Bezemer und Zaidman 2010b, S. 89; Abbildung 3.6, Mitte), werden durch den zweiten SaaS-Reifegrad gekennzeichnet.

Cloudbasierte, mandantenfähige ERP-Systeme, welche eine unabhängige Konfiguration einzelner Mandanten zulassen, aber zugleich begrenzte vertikale Skalierbarkeit sicherstellen, erhalten den dritten Reifegrad. Der vierte Reifegrad ist für cloudbasierte, vollständig mandantenfähige, konfigurierbare ERP-Systeme vorgesehen, welche zugleich die dynamische horizontale Skalierbarkeit der gemeinsam benutzten infrastrukturellen Ressourcen sicherstellen.

Neben dem SaaS-Reifegrad können innerhalb der Untersuchung auch andere Perspektiven auf die Betrachtungsebene der Systemarchitektur angewendet werden. Insbesondere kann bei der Datenauswertung eine Differenzierung anhand einzelner Cloud-Bereitstellungs- und Service-Modelle (vgl. Kapitel 1.2.1) notwendig sein, bzw. eine Unterscheidung zwischen den cloud-nativen und -fähigen Systemen (vgl. Kapitel 3.1.2) vorgenommen werden.

**Tabelle 4.5:** SaaS-Reifegrade

| Reifegrad | Charakteristika | Codierung |
|---|---|---|
| I | Cloudfähiges System ohne eigebaute Mechanismen zur anwenderseitigen Systemkonfiguration; keine Mechanismen der Mandantenfähigkeit vorhanden bzw. Mandantenfähigkeit realisiert auf der Infrastrukturebene (siehe Abbildung 3.6, links); vertikale Skalierbarkeit | R1 |
| II | Cloudfähiges System mit eingebauten Tools zur Auswahl und Konfiguration des Lösungsumfangs im Rahmen eines On-Demand-Selfservices; keine Mechanismen der Mandantenfähigkeit vorhanden oder Mandantenfähigkeit realisiert auf der Server-Ebene (siehe Abbildung 3.6, Mitte); vertikale Skalierbarkeit | R2 |
| III | Cloudfähiges, konfigurierbares System; Mechanismen der Mandantenfähigkeit vorhanden auf der Ebene der Anwendungsinstanz; die mandantenübergreifenden Hardware-Ressourcen können vertikal skaliert werden | R3 |
| IV | Cloudfähiges, konfigurierbares System mit eingebauten Mechanismen der Mandantenfähigkeit; dynamische horizontale Skalierbarkeit der gemeinsam benutzten Ressourcen | R4 |

Die Einbeziehung der architektonischen Aspekte in das Systematisierungsmodell schafft die erforderliche Transparenz für die bevorstehenden Analysen und trägt wesentlich zum Verständnis der Customizing-Strategien differenzierter cloudbasierten ERP-Systeme bei. Die Berücksichtigung architektonischer Aspekte ermöglicht des Weiteren eine statistische Auswertung der Zusammenhänge zwischen dem architektonischen Konzept und der Customizing-Strategie, wodurch ein besonderer Mehrwert geschaffen wird.

## 4.2 Literaturrecherche zur Anpassung von ERPaaS

Dieses Kapitel bietet einen Überblick über bestehende Studien zum Thema des Customizing der cloudbasierten ERP-Systeme. Um ein klares Bild über den Stand der wissenschaftlichen Forschung zu schaffen, wurde eine strukturierte Literaturanalyse vorgenommen. Im Rahmen dieses Abschnitts werden die Charakteristika der bisherigen Untersuchungen dargestellt und die Limitierungen der Forschung analysiert.

### 4.2.1 Methodik

In der vorliegenden Ausarbeitung wurde eine strukturierte Literaturauswertung vorgenommen. Um möglichst systematische bzw. objektive Ergebnisse zu erzielen, wurde der Leitfaden für eine strukturierte Literaturauswertung nach Kitchenham (2007) durch die Prinzipien für eine qualitative Inhaltsanalyse nach Mayring und Brunner (2009) ergänzt.

Die Literaturrecherche basierte auf englischsprachiger Fachliteratur. Für die Literatursuche wurden zunächst die folgenden wissenschaftlichen Datenbanken ausgewählt: IEEE Xplore, Elsevier ScienceDirect, SpringerLink und CiteSeer. Diese sind laut Brereton et al. (2007, S. 577 f.) für dieses Forschungsgebiet am relevantesten. Darüber hinaus wurde in drei weiteren Literaturdatenbanken – EBSCO, AISeL, ACM Digital Library – nach Konferenzbeiträgen recherchiert (vgl. Kitchenham 2007, S. 17; Schlichter und Kraemmergaard 2010, S. 493).

Die Literatursuche wurde mittels einer webbasierten Suchmaschine der jeweiligen Literaturdatenbank durchgeführt. Dabei wurde immer eine erweiterte Suche bevorzugt, bei der neben den Booleschen Ausdrücken auch die Stellvertretersymbole benutzt werden konnten. Die Metadaten wie Titel, Schlagwörter und Abstract wurden auf das Vorhandensein der Suchbegriffe „Cloud“, „ERP“ und „Customizing“ überprüft.

Zum Erstellen einer eindeutigen Abfrage wurden sowohl die Suchbegriffe als auch ihre Synonyme bzw. gleichbedeutende Wörter benutzt:

- Cloud, On-demand, Software-as-a-Service, SaaS
- Enterprise Resource Planning, ERP
- Customizing, configuration, tailoring, modification, adaptation etc. (siehe Tabelle 2.1).

Da die Suchbegriffe in unterschiedlichen Formen auftreten können, wurde dort, wo es nötig war, das *-Zeichen als Stellvertreteroperator angewendet, um alle möglichen Konjunktionen des Suchbegriffes zu berücksichtigen. Der Umfang der Literaturrecherche wurde auf die Veröffentlichungen aus dem Informatikbereich (Computer Science), die nach dem Jahr 2006 erschienen sind, beschränkt. Das Jahr 2006 wurde als untere Grenze der Suchperiode gesetzt, da es als Einführungsjahr der Begriffe Cloud, Cloud-Computing und Software-as-a-Service gilt (vgl. Yang und Tate 2012, S. 45).

### 4.2.2 Verlauf der Literaturrecherche und Charakteristika bisheriger Forschung

Der Verlauf der Literaturrecherche wurde in der Tabelle 4.6 zusammengefasst. Die erste Spalte gibt einen Überblick über die ausgewählten Literaturdatenbanken, die weiteren Spalten stellen die nachfolgenden Schritte der Literaturauswertung dar. Die Zeilen geben die Anzahl der in der jeweiligen Datenbank gefundenen Veröffentlichungen an, welche nach dem entsprechenden Schritt der Literaturrecherche als relevant eingestuft wurden.

**Tabelle 4.6:** Verlauf der Literaturrecherche

| | Phase 1 | | | Phase 2 | | |
|---|---|---|---|---|---|---|
| **Literatur-datenbank** | **Suchmaschinen-anfrage** (1) | **Metadaten-analyse** (2) | **Inhalts-analyse** (3) | **Vorwärts- und Rückwärtssuche** (4) | **Google Scholar** (5) | **Summe** (3) + (4) + (5) |
| ACM | 34 | 3 | 2 | 2 | - | **4** |
| AISeL | 25 | 7 | 1 | - | - | **1** |
| CiteSeer | 25 | 6 | 1 | - | - | **1** |
| EBSCO | 6 | 4 | 1 | - | - | **1** |
| Emerald | 26 | 5 | 2 | - | - | **2** |
| IEEE | 48 | 11 | 3 | 1 | - | **4** |
| ScienceDirect | 19 | 7 | 3 | - | - | **3** |
| SpringerLink | 35 | 12 | 6 | - | - | **6** |
| Sonstige | - | - | - | 5 | 5 | **10** |
| **Summe** | **218** | **55** | **19** | **8** | **5** | **32** |

Die Suchmaschinen der ausgewählten Datenbanken lieferten insgesamt 218 Quellen. Davon wurden nach einer detaillierten Überprüfung des Titels und des Abstracts 23 Publikationen als relevant und weitere 35 als teilweise relevant eingestuft. Die weitere Fachliteratur erwies sich

als zu oberflächlich, bzw. passte nicht genau zum Thema. Nach Entfernung der Duplikate sind 55 Publikationen verblieben, die einer exakten Analyse des Inhaltes unterworfen wurden. Im Rahmen der Inhaltsanalyse wurden 19 Veröffentlichungen ermittelt, welche die Thematik aus Sicht der vorliegenden Arbeit hinreichend untersuchten.

Aufgrund der geringen Anzahl an gefundenen Quellen wurde eine zweite Phase der Literaturrecherche mithilfe der Rückwärts- und Vorwärtssuche vorgenommen (vgl. Webster und Watson 2002, S. 16). Bei der Rückwärtssuche wurden die Literaturverzeichnisse der durch die erste Phase der Literaturrecherche ermittelten Veröffentlichungen nach weiteren Artikeln durchsucht. Im Gegensatz dazu wurden in der Vorwärtssuche weitere Veröffentlichungen betrachtet, in denen die ursprünglich ermittelten Quellen zitiert wurden. Die Rückwärts- und Vorwärtssuche lieferten insgesamt 21 weitere Quellen, von denen nach einer detaillierten Inhaltsanalyse 8 als relevant eingestuft wurden.

Zur Erweiterung der immer noch kleinen Anzahl an Suchergebnissen wurde eine internetbasierte Suche mittels der Google-Scholar-Suchmaschine vorgenommen, bei welcher die gleiche Abfrage wie bei der wissenschaftlichen Literaturdatenbanken durchgeführt wurde. Es wurden insgesamt fünf zusätzliche Publikationen gefunden, in welchen das Thema hinreichend abgedeckt wurde.

Insgesamt 32 Veröffentlichungen wurden einer inhaltlichen Analyse unterworfen, die mit einer Kategorisierung der Literaturergebnisse hinsichtlich der untersuchten Customizing-Themen endete. Die Ergebnisse der Literaturauswertung wurden in der Tabelle 4.7 zusammengestellt.

**Tabelle 4.7:** Literaturübersicht zum Customizing von ERPaaS (2006-2016)

| | **Behandelte Themenschwerpunkte** | Al-Shardan und Ziani (2015) | Arya et al. (2010) | Bezemer und Zaidman (2010b) | Borovskiy und Zeier (2009) | Chen et al. (2015) | Duan et al. (2013) | Elragal und Kommos (2012) | Gerhardter und Ortner (2013) | Grubisic (2014) | Hao et al. (2012) | Iqbal et al. (2012) | Johansson und Ruivo (2013) | Kang et al. (2011) | Kurbel und Nowak (2013a) | Kurbel und Nowak (2013b) | Link (2013) | Link und Back (2013) | Magnusson et al. (2012) | Mijač et al. (2013) | Müller et al. (2009) | Nitu (2009) | Ortner und Krenn (2016) | Parthasrathy (2013) | Peng und Gala (2014) | Rabay'a et al. (2013) | Saeed et al. (2012) | Scavo et al. (2012) | Schmid und Rummler (2012) | Schubert und Adisa (2011) | Seethamraju (2015) | Sun et al. (2008) | Uppström et al. (2015) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **System-Flexibilität** | Adaptierbarkeit der Cloud-ERP-Systeme | X | + | + | + | X | + | + | + | + | + | + | + | + | X | + | + | + | + | X | + | + | X | + | + | + | + | + | X | + | + | + | X |
| | Flexibilitätskonzepte und ihre Machbarkeit | X | X | + | X | X | – | + | + | – | + | – | – | X | X | X | – | – | – | + | X | X | X | – | – | – | – | – | X | – | + | X | – |
| | Herausforderungen der Systemadaption | X | X | X | + | + | + | – | – | + | + | + | + | + | + | + | + | + | – | X | + | + | + | + | – | + | + | + | X | + | + | + | + |
| **System-Architektur** | Cloudnativität und Cloudfähigkeit | – | – | + | – | – | X | – | – | – | – | – | – | – | – | – | – | – | + | + | – | – | – | – | + | – | – | X | + | – | – | – | – |
| | Cloud-Bereitstellungsmodelle | – | – | – | – | – | + | – | – | + | + | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | X | + | + | – | – | + |
| | Service-Betreibermodelle | – | – | – | – | + | – | – | – | + | + | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | X | + | – | – | + |
| | SaaS-Reife | – | – | + | – | + | + | – | – | – | – | + | – | – | – | – | – | – | – | + | – | – | – | – | + | – | + | + | + | – | + | – | – |
| **Customizing-Strategie** | Adaptionsansätze | X | X | X | + | + | – | + | X | – | + | – | + | X | + | + | + | + | – | X | X | X | + | + | – | – | – | + | X | + | + | X | X |
| | Adaptierbare Systemelemente | + | X | X | – | – | + | – | – | – | – | – | + | X | X | + | – | – | – | + | + | X | + | – | – | + | – | + | X | + | + | + | + |
| | Adaptionswerkzeuge | + | – | – | + | X | – | + | + | – | – | – | – | + | X | X | – | + | – | + | – | – | X | + | – | – | – | – | X | + | – | + | – |

**Legende:** X Hauptfokus, + Nebenthema, - nicht betrachtet

Die vorgenommene Literaturauswertung lieferte lediglich eine kleine Anzahl an Veröffentlichungen (n = 7), die inhaltlich ausschließlich dem Thema der Adaptierbarkeit von cloudbasierten ERP-Systemen gewidmet sind. Weitere Quellen (n = 25) stimmen nur teilweise mit dem Forschungsthema überein, bzw. weisen lediglich auf die Problematik der Anpassung von mandantenfähigen On-Demand ERP-Systemen hin.

Über den Mangel an Fachliteratur zu diesem Thema wurde auch von anderen Forscher berichtet (vgl. Hao et al. 2012, S. 415; Peng und Gala 2014, S. 23; Mijač et al. 2013, S. 133). Die Unterrepräsentierung der für das Forschungsthema relevanten Veröffentlichungen in der Gesamtheit aller Publikationen über ERPaaS ist ein Grund dafür, dass weitere Forschung auf diesem Gebiet erforderlich ist.

Abbildung 4.2 veranschaulicht die zeitliche Verteilung der aus Sicht dieser Arbeit relevanten Veröffentlichungen im Vergleich zu der Gesamtheit aller Veröffentlichungen zum Thema der cloudbasierten ERP-Systeme. Die Anzahl der ausgewählten Veröffentlichungen ist auf der Primärachse (links) dargestellt. Die Anzahl der Publikationen, die das Thema der „cloudbasierten ERP-Systeme“ und der „Flexibilität der Cloud-ERP-Systeme“ berühren, ist auf der Sekundärachse (rechts) angegeben.

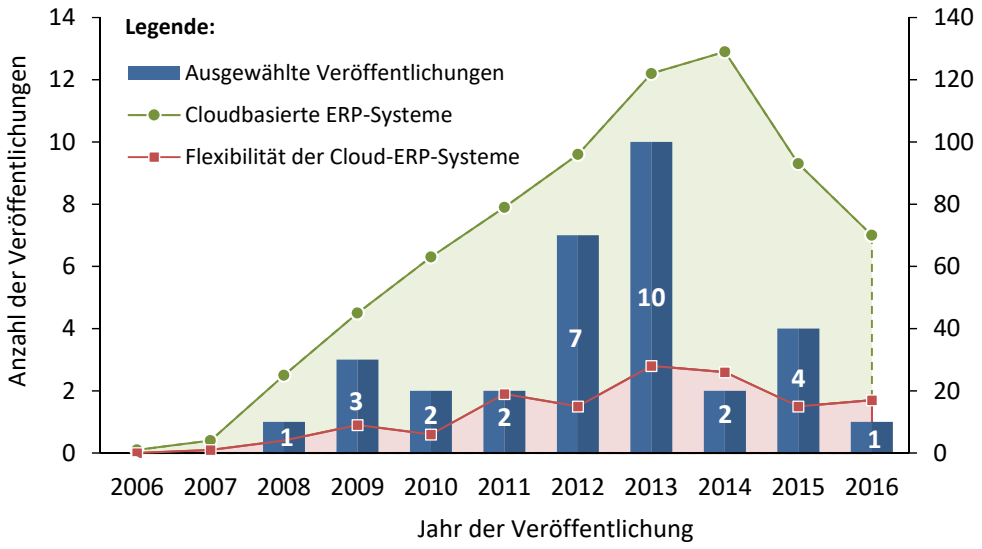


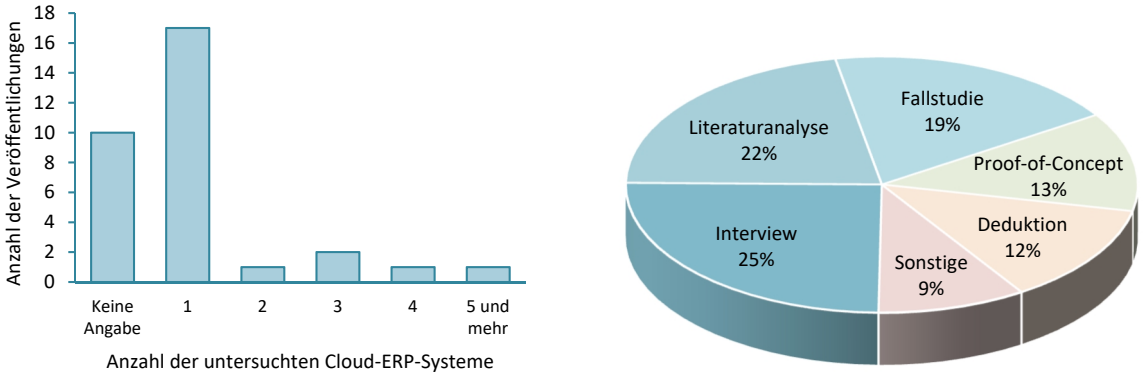

**Abbildung 4.2:** Zeitliche Verteilung der analysierten Veröffentlichungen

Die Graphik verdeutlicht, dass trotz des allgemeinen Interesses an der Thematik der cloudbasierten ERP-Systeme lediglich etwa fünf Prozent der erschienenen Quellen die für die vorliegende Ausarbeitung relevanten Aspekte der Systemflexibilität fokussieren.

Nach einer detaillierten Inhaltsanalyse sei anzumerken, dass die Mehrheit der Studien vor dem Jahr 2012 einer Analyse der Herausforderungen an das Customizing der cloudbasierten ERP-Systeme bzw. der Entwicklung der Customizing-Konzepte gewidmet ist. Erst im Jahr 2012 sind erste Untersuchungen erschienen, welche die Flexibilität konkreter ERP-Systeme untersuchen (vgl. z. B. Scavo et al. 2012).

Im Hinblick auf den Umfang der analysierten Studien kann festgestellt werden, dass viele Autoren sich bei ihren Untersuchungen bezüglich des cloudbasierten Customizing hauptsächlich auf ein ausgewähltes ERP-System bzw. auf ein bestimmtes Cloud-Betreibermodell fokussieren (siehe Abbildung 4.3a; vgl. z. B. Elragal und Kommos 2012; Seethamraju und Sundar 2013; Hao et al.

2012). Während einige Forscher ausgereifte, cloudbasierte ERP-Systeme analysieren (z. B. Duan et al. 2013, S. 5), erweitern andere Autoren ihre Untersuchung auf Systeme in gehosteten Clouds, die mithilfe der Virtualisierungstechniken bereitgestellt werden (vgl. z. B. Grubisic 2014; Rabay'a et al. 2013). Darüber hinaus fehlen den bisherigen Studien oft die präzisen Angaben über die Architektur der untersuchten ERP-Systeme. Die ständige Entwicklung des Marktes und die unzureichende Transparenz der analysierten Bereitstellungskonzepte erschweren dabei eine korrekte Interpretation der Studienergebnisse.




(a) Umfang der Studien (b) Angewandte Untersuchungsmethoden

**Abbildung 4.3:** Charakteristika der analysierten Veröffentlichungen – Umfang der Studien (a); angewandte Untersuchungsmethoden (b)

Bei Auswertung der selektierten Literatur hinsichtlich der angewandten Untersuchungsmethoden wurde festgestellt, dass die Mehrheit (n = 21) der Veröffentlichungen die qualitativen Studien bilden, die ein Interview, eine Literaturanalyse oder eine Fallstudie als Forschungsmethode einsetzen (siehe Abbildung 4.3b). Unterrepräsentiert bleiben zurzeit insbesondere explorative Untersuchungen (n = 5), welche die Adaptierbarkeit der cloudbasierten ERP-Systeme anhand konkreter praxisnahen Szenarien validieren, bzw. Customizing-Möglichkeiten mehrerer Cloud-ERP-Systeme vergleichen.

## 4.3 Diskussion und Hypothesen-Bildung

Die in der Literaturrecherche ermittelten Veröffentlichungen wurden in Hinblick auf die gestellten Untersuchungsfragen (vgl. Kapitel 1.3) analysiert, wobei ein besonderer Fokus auf mögliche Zusammenhänge zwischen der Reife eines cloudbasierten ERP-Systems und der bereitgestellten Adaptierbarkeit gelegt wurde.

Die nachfolgenden Unterkapitel zielen auf eine ausführliche Diskussion der identifizierten Adaptierbarkeitsaspekte der Cloud-ERP ab und dienen der Aufstellung von Hypothesen, welche im Rahmen dieser Ausarbeitung in einer empirischen Untersuchung validiert werden und letztendlich zur Beantwortung der zweiten Hauptforschungsfrage (HF 2) beitragen.

### 4.3.1 Übersicht der Customizing-Ansätze

Gemäß dem Reifegradmodell für cloudbasierte Anwendungen, in welchem eine cloudfähige On-Premise-Anwendung ihre SaaS-Reife durch schrittweises Hinzufügen von weiteren SaaS-Eigenschaften erreicht, sind die wenig ausgereiften Cloud-Anwendungen dem On-Premise-Modell am ähnlichsten (vgl. Chen et al. 2011, S. 392). Diese Ähnlichkeiten werden laut Magnusson et al. (2012) insbesondere im Falle des bereitgestellten Abrechnungsmodells und der bereitgestellten Customizing-Strategie ersichtlich (vgl. Magnusson et al. 2012, S. 72).

Die gezielte Auswahl und Zusammenstellung der einzelnen Customizing-Ansätze in eine systemindividuelle Customizing-Strategie ist im Fall der cloudbasierten, mandantenfähigen ERP-Systeme stärker architektonisch bedingt als bei den konventionellen On-Premise-ERP-Systemen. Die architektonischen Herausforderungen, welche die Auswahl der anwendbaren Adaptionsansätze begrenzen, haben einen Einfluss auf den Entwurf einer Customizing-Strategie und auf die Festlegung eines adaptierbaren Systemumfangs eines ERPaaS.

Sun et al. (2008) postulieren, dass die cloudbasierten Systeme lediglich Konfigurationsmechanismen anbieten, welche vor allem die Anpassung der Benutzeroberfläche und nur einen begrenzten Anteil der Funktionalität in einem vordefinierten Bereich erlauben. Eine ähnliche Meinung wird auch von Seethamraju (2015) und Hohmann et al. (2013) vertreten. Andere Studien bestätigen dagegen, sowohl im Falle der weniger als auch im Falle der mehr ausgereiften cloudbasierten ERP-Systeme die Möglichkeit der Anwendung von einigen Adaptionsansätzen, welche außerhalb des Konfigurationsrahmens liegen (vgl. Borovskiy und Zeier 2009; Saeed et al. 2012; Duan et al. 2013; Iqbal et al. 2012). In diesem Zusammenhang wird von den Autoren insbesondere auf die Programmierschnittstellen, Add-ons und Kompositanwendungen als mögliche Adaptionsansätze hingewiesen (vgl. auch Johansson und Ruivo 2013, S. 97 f.).

Uppström et al. (2015) wiesen in diesem Kontext auf die Existenz wesentlicher Unterschiede bezüglich der Anwendbarkeit von einzelnen Adaptionsansätzen zwischen konventionellen On-Premise-ERP-Systemen und ihren cloudbasierten Gegenstücken hin. Die Ergebnisse der durchgeführten Anbieter-Interviews deuten auf eine durch das Bereitstellungsmodell bedingte Veränderung der bereitgestellten Customizing-Konzepte hin. Einige Systemadaptionen, welche bei den On-Premise-Systemen eine Modifikation des Systemcodes erfordern, können laut den Autoren bei den ERPaaS mittels der Parametrisierung vorgenommen werden (vgl. Uppström et al. 2015, S. 4224).

Aus diesem Grund lassen sich auch einige Unterschiede beim Einsatz der einzelnen Customizing-Ansätze an unterschiedlichen Stufen des SaaS-Reifegradmodells erwarten. Um diese Behauptung verifizieren zu können, wird die folgende Hypothese aufgestellt:

H1: *Die Bedeutung der einzelnen Customizing-Ansätze im Rahmen der gesamten Customizing-Strategie des Systemanbieters hängt von dem SaaS-Reifegrad des cloudbasierten ERP-Systems ab*

In Übereinstimmung mit den Forschungsergebnissen von Nitu (2009) sowie von Bezemer und Zaidman (2010b) können die durch die Mandantenfähigkeit gekennzeichneten Systeme vor allem mittels codefreien Customizing-Ansätzen adaptiert werden. Diese Erkenntnis würde eine positive Beziehung zwischen dem SaaS-Reifegrad des ERP-Systems und dem Anwendungsumfang der Parametrisierung bzw. der komponentenbasierten Zusammenstellung implizieren. Daraus ableitend, sollen die ausgereiften Cloud-ERP weniger codebasierte Customizing-Ansätze zur Verfügung stellen als die weniger ausgereiften ERP-Systeme. Mit anderen Worten, je ausgereifter das cloudbasierte ERP-System ist, umso größer soll auch der Anwendungsumfang der parameterbasierten Anpassungsmethoden und desto kleiner der Einsatz von codebasierten Systemmodifikationen sein.

Die Literatur bezüglich des Customizing von cloudbasierten Anwendungen weist auf weitere Ansätze hin, welche die Anpassungsprobleme der cloudbasierten Software lösen können (vgl. Arya et al. 2010; Kang et al. 2011). Einige Forscher entwickelten weitere Customizing-Konzepte, welche die Erweiterung und Anpassung von cloudbasierten ERP-Systemen ermöglichen (vgl. Chen et al. 2015; Al-Shardan und Ziani 2015). Die vorgenommene Literaturauswertung stellt

jedoch kaum einen Beweis für die Umsetzbarkeit dieser Adaptionskonzepte in der Praxis dar. Aus diesem Grund ist neben der vorgenommenen Literaturauswertung eine weitere praxisorientierte Analyse der auf dem ERPaaS-Markt verfügbaren Customizing-Konzepte notwendig.

### 4.3.2 Adaptierbarkeit der Systemelemente

Die analysierten Quellen weisen hauptsächlich auf die Konfigurierbarkeit bzw. Parametrisierbarkeit von mehreren Elementen der cloudbasierten ERP-Systeme hin (vgl. Uppström et al. 2015, S. 4224). Einige Wissenschaftler, die sich mit der Customizing-Problematik der cloudbasierten ERP-Anwendungen befassen, nennen zahlreiche Konzepte für die unternehmensbezogene Individualisierung der Systemoberfläche (Kurbel und Nowak 2013a, S. 293; Müller et al. 2009, S. 68), die Anpassungen der Geschäftsabläufe (Arya et al. 2010, S. 211; Bezemer und Zaidman 2010b, S. 91), die Adaptionen bzw. Erweiterungen des Datenmodells um kundenspezifische Geschäftsobjekte (Al-Shardan und Ziani 2015; Nitu 2009, S. 22) sowie für eine betriebswirtschaftliche Konfiguration des bestehenden Lösungsumfangs und die Integration mit anderen Anwendungen des Geschäftsumfelds (Gerhardter und Ortner 2013, S. 173; Kurbel und Nowak 2013b, S. 423 ff.; Nowak und Kurbel 2014, S. 433 ff.). Die meisten der genannten Referenzen liefern jedoch keine klare Antwort auf die Frage bezüglich des bereitgestellten Flexibilitätsumfangs der untersuchten Systemelemente, welcher durch das eingesetzte Adaptionskonzept sichergestellt wird.

Allerdings weisen Schmid und Rummler (2012) in diesem Kontext auf die Existenz architekturbedingter Differenzen im Umfang der bereitgestellten Flexibilität einzelner Systemelemente hin, welche unter anderem aus dem ausgewählten Service-Modell resultieren können (vgl. Schmid und Rummler 2012, S. 166). Für die detaillierte Untersuchung möglicher Zusammenhänge zwischen dem architektonischen Konzept eines cloudbasierten ERP-Systems und dem bereitgestellten Flexibilitätsumfang einzelner Systemelemente wurde die folgende Hypothese aufgestellt:

H2: *Die bereitgestellte Flexibilität einzelner Systemelemente ist von der SaaS-Reife des cloudbasierten ERP-Systems abhängig*

Die statistische Validierung der aufgestellten Hypothese anhand der Daten mehrerer cloudbasierter ERP-Systeme soll eine Antwort auf das Vorhandensein generalisierbarer Zusammenhänge zwischen dem architektonischen Konzept eines Cloud-ERP und dem bereitgestellten Flexibilitätsumfang einzelner Systemelemente geben.

Laut einigen Forschern (z. B. Nitu 2009; Link und Back 2013) schöpfen die mittels der Konfiguration erreichbaren Systemadaptionen die Adaptionsmöglichkeiten der ausgereiften, mandantenfähigen Software-Architektur aus. Dabei ist der adaptierbare Systemumfang auf die vom Systemhersteller vorgesehenen Konfigurationsvarianten begrenzt (vgl. Al-Shardan und Ziani 2015, S. 97 f.; Sun et al. 2008, S. 22 f.; Gerhardter und Ortner 2013, S. 173).

Scavo et al. (2012) behaupten, dass ausgereifte, cloudbasierte ERP-Systeme im Gegensatz zu den cloudfähigen ERP-Systemen in gehosteten Clouds keine umfassende Systemadaption bzw. Integration mit den externen Webdiensten und Systemen der Drittanbieter zulassen (vgl. Scavo et al. 2012, S. 12 f.). Diese Skepsis wird auch durch weitere Studien vertreten (vgl. Link 2013, S. 271; Johansson et al. 2014, S. 9 f.; Peng und Gala 2014, S. 28). Bezemer und Zaidman (2010b) begründen, dass ein großer Flexibilitätsumfang im Fall der ausgereiften, mandantenfähigen Cloud-ERP-Systeme zu Problemen bei der effektiven Anwendungswartung führen könnte (vgl. Bezemer und Zaidman 2010b, S. 89 ff.).

Sind die von Scavo und Bezemer dargestellten Erkenntnisse generalisierbar, sollen sie auch durch eine statistische Analyse der Zusammenhänge zwischen der SaaS-Reife und der Flexibilität einzelner Systemelemente nachgewiesen werden. Diese Erkenntnisse würden eine negative Korrelation zwischen dem SaaS-Reifegrad und dem bereitgestellten Flexibilitätsumfang im Falle mindestens eines Systemelementes implizieren.

### 4.3.3 Bereitgestellte Customizing-Werkzeuge

Scavo et al. (2012) betonen, dass im Fall der ERP-Systeme in der gehosteten Cloud ein Großteil der Systemanpassungen (einschließlich Parametrisierung) durch Anbieter- bzw. Partnerunternehmen durchgeführt wird. Die Anbieter der gehosteten ERP-Systeme erlauben oftmals keine Systemkonfiguration bzw. Aktivierung von neu entwickelten Systemfunktionalitäten, ohne eine Involvierung der Anbieterseite vorzunehmen (vgl. Scavo et al. 2012, S. 6). Die ausgereiften ERP-Systeme stellen dagegen den Anwendungsexperten geeignete Konfigurationswerkzeuge zur Verfügung, welche eine eigenständige Softwareanpassung ermöglichen (vgl. Kurbel und Nowak 2013a, S. 292).

Laut der Studie von Iqbal et al. (2012) bestätigen einige Systemanbieter, dass sie ihren Kunden auch zusätzliche, nicht in der ERP-Software integrierte Customizing-Werkzeuge bereitstellen können, welche eine weitergehende Adaption der ERPaaS an die kundenindividuellen Anforderungen mittels Erweiterungsprogrammierung sicherstellen (vgl. Iqbal et al. 2012, S. 398). Eine ähnliche Meinung wird auch von Scavo et al. (2012) vertreten. Die Autoren stellen fest, dass einige Anbieter der ausgereiften Cloud-ERP entsprechende Werkzeuge bereitstellen, die eine Anpassung bzw. Erweiterung der standardisierten Geschäftslogik von der Anwenderseite her ermöglichen (vgl. Scavo et al. 2012, S. 15). Ihrer Meinung nach sind die bereitgestellten Adaptionsmöglichkeiten der ERPaaS jedoch weniger umfangreich als diejenigen, die von den wenig ausgereiften, gehosteten ERP-Systemen angeboten werden (vgl. Scavo et al. 2012, S. 12 f.). Andere Studien weisen hingegen auch auf proprietäre systemexterne Werkzeuge hin, welche den Kunden vom Systemanbieter zur Verfügung gestellt werden. Dazu zählen unter anderem systemdedizierte webbasierte Schnittstellen und PaaS-bezogene Dienste, die eine Adaption von Integrationsmechanismen und Mechanismen der B2B-Kommunikation ermöglichen (vgl. Parthasarathy 2013, S. 190; Schubert und Adisa 2011, S. 16; Johansson und Ruivo 2013, S. 97 f.).

Link und Back (2013) weisen in ihrer Studie darauf hin, dass die Anbieter der cloudbasierten ERP-Systeme einerseits einige eingebaute Konfigurationswerkzeuge bereitstellen, die z. B. visuelle Systemadaptionen erlauben, anderseits aber auch vorkonfigurierte Schnittstellen anbieten, welche der Integration von systemexterner betrieblicher Software dienen (vgl. Link und Back 2013, S. 14). Hao und Koautoren (2012) weisen in ihrer Studie auf Adaptionsmöglichkeiten der cloudbasierten ERP-Systeme hin, welche über den Rahmen einer Parametrisierung hinausgehen. Die Forscher argumentieren jedoch, dass das Bereitstellen der Customizing-Werkzeuge hauptsächlich von der Customizing-Strategie des Systemanbieters abhängt (vgl. Hao et al. 2012, S. 420 f.).

Während einige Studien auf die Verfügbarkeit differenzierter Customizing-Werkzeuge deuten, weisen die Untersuchungen von Mijač et al. (2013) im Gegensatz auf Mängel an den systemexternen Werkzeugen, insbesondere an den Entwicklungsumgebungen, hin. Die fehlenden Entwicklertools werden von den Autoren als direkte Ursache für die begrenzten Customizing-Möglichkeiten der meisten cloudbasierten ERP-Systeme angesehen (vgl. Mijač et al. 2013, S. 139).

Zusammenfassend kann aus den Ergebnissen der Literaturauswertung keine klare Tendenz hinsichtlich der Bereitstellung einzelner Kategorien von Adaptionswerkzeugen abgeleitet werden.

Viele Autoren sind sich darüber einig, dass viele erfolgskritische Adaptionen mithilfe integrierter Werkzeuge realisiert werden können. Im Hinblick auf die Verfügbarkeit der systemexternen Adaptionswerkzeuge herrscht jedoch zwischen den Autoren keine Übereinstimmung mehr. Um die identifizierten Unstimmigkeiten aufzuklären, wird im Rahmen der vorliegenden Dissertation eine detaillierte Analyse der bereitgestellten Adaptionswerkzeuge unter Einbeziehung der SaaS-Reife vorgenommen. Die Generalisierbarkeit der Zusammenhänge zwischen dem architektonischen Systemkonzept und den bereitgestellten Adaptionswerkzeugen wird anhand der folgenden Hypothese untersucht:

H3: *Die SaaS-Reife eines cloudbasierten ERP-Systems hat Einfluss auf die Art der bereitgestellten Adaptionswerkzeuge*

Wenn es Zusammenhänge zwischen der SaaS-Reife eines cloudbasierten ERP-Systems und den einzelnen Werkzeug-Kategorien gibt, sollten diese im Rahmen einer statistischen Korrelationsanalyse nachgewiesen werden.

Im Hinblick auf die Effizienz der anwenderseitigen Customizing-Prozesse bietet die vorgenommene Literaturauswertung wenige Informationen. Die bereitgestellten Mechanismen der Anwenderunterstützung werden lediglich in wenigen Quellen explizit beschrieben.

In diesem Kontext deutet Link (2013) darauf hin, dass die Änderung der modularen und funktionalen Zusammensetzung eines cloudbasierten ERP-Systems im Vergleich zu On-Premise-Systemen einfacher durchzuführen ist (vgl. Link 2013, S. 268). Diese Meinung wird auch von Seethamraju (2015, S. 479) vertreten. Die beiden Autoren nennen jedoch keine Details bezüglich der Konzepte, die zur Erleichterung bzw. Vereinfachung der anwenderseitigen Adaptionsprozesse in den bereitgestellten Customizing-Werkzeugen verankert werden können.

Die Untersuchungen von Ortner und Krenn (2016) sowie Kurbel und Nowak (2013a-b) wiesen auf konkrete Mechanismen hin, welche zur effektiven Unterstützung der Anwendungsexperten bei der Durchführung der Adaptionsprozesse bereitgestellt werden. Die erste Studie legt ihren Fokus auf die eingebauten Mechanismen, welche den Anwendungsexperten bei der Systemimplementierung aktiv lenken bzw. die einzelnen Schritte der Systemkonfiguration automatisieren (vgl. Ortner und Krenn 2016). Die Untersuchungen von Kurbel und Nowak demonstrieren die eingebauten Werkzeuge, die mit wenig Aufwand eine Verankerung systemexterner Inhalte bzw. die Kopplung mit anderen Anwendungen und Webdiensten ermöglichen. Die Autoren der zitierten Studien weisen auf die positive Auswirkung der eingebauten Mechanismen auf die Effizienz der anwenderseitigen Adaptionsprozesse hin.

Obwohl die Studien jeweils nur ein cloudbasiertes ERP-System untersuchen und daher keine Generalisierbarkeit der Erkenntnisse sichergestellt ist, stellen sie aber eine erste Prämisse dar – nämlich dass die Systemhersteller der cloudbasierten ERP-Systeme großen Wert auf die Erleichterung der anwenderseitigen Adaptionsprozesse legen. Zur Verifikation der Zusammenhänge zwischen dem bereitgestellten Umfang der Anwenderunterstützung und der architektonischen Reife der cloudbasierten ERP-Systeme wird im Studienrahmen die folgende Hypothese aufgestellt:

H4: *Der Umfang einer werkzeugbasierten Anwenderunterstützung bei Durchführung der Systemadaptionen ist durch den Reifegrad des cloudbasierten ERP-Systems beeinflusst*

In Bezug auf die Konfigurierbarkeit im Rahmen des On-Demand-Selfservices (siehe Kapitel 3.3.2), die eines der grundlegenden SaaS-Merkmale darstellt, wäre eine eher positive Korrelation zwischen der SaaS-Reife eines cloudbasierten ERP-Systems und dem Umfang der bereitgestellten Anwenderunterstützung zu erwarten. Diese Behauptung ist auf die Erkenntnisse von Scavo et al. (2012) zurückzuführen; die Autoren weisen auf eine steigende Rolle des anwenderseitigen Customizing bei den ausgereiften cloudnativen ERP-Systemen hin (vgl. Scavo et al. 2012, S. 6).

# 5 Umsetzbarkeit der Systemadaptionen

Das zentrale Thema dieses Kapitels ist die vergleichende Analyse der Anpassungs- und Modifikationsmöglichkeiten der ausgewählten Cloud-ERP-Systeme. Die in den nachfolgenden Kapiteln vorgenommenen Machbarkeitsstudien tragen systematisch durch Validierung der Adaptierbarkeit einzelner Systemelemente zur Ermittlung des adaptierbaren Systemumfangs und damit zur Beantwortung der aufgestellten Forschungsfrage HF 1 bei. Darüber hinaus ermöglicht es die vorgenommene Analyse konkreter Werkzeuge und Anpassungstechniken, die Systematisierung und die Bewertung der bereitgestellten Anpassungskonzepte vorzunehmen.

## 5.1 Untersuchungsrahmen

Vor der explorativen Untersuchung wurde der Untersuchungsrahmen festgelegt, welcher die Determinierung des methodischen Vorgehens, die Auswahl der Untersuchungsgegenstände und die Konzipierung der Machbarkeitsszenarien umfasst. Die einzelnen Aspekte werden in den nachfolgenden Unterabschnitten genauer beschrieben.

### 5.1.1 Methodik und Verlauf der Untersuchung

Den Untersuchungen dieses Kapitels liegt der gestaltungsorientierte Forschungsansatz (vgl. Krcmar 1995; Wilde und Hess 2007, S. 281) zugrunde, welcher den Schwerpunkt auf die Entwicklung der Lösungskonzepte und auf die Untersuchung der Durchführbarkeit, bzw. die Validierung der technischen Machbarkeit der konzipierten Artefakte, legt. Struktur und Ablauf der Untersuchungen dieses Kapitels wurden in der Abbildung 5.1 veranschaulicht.

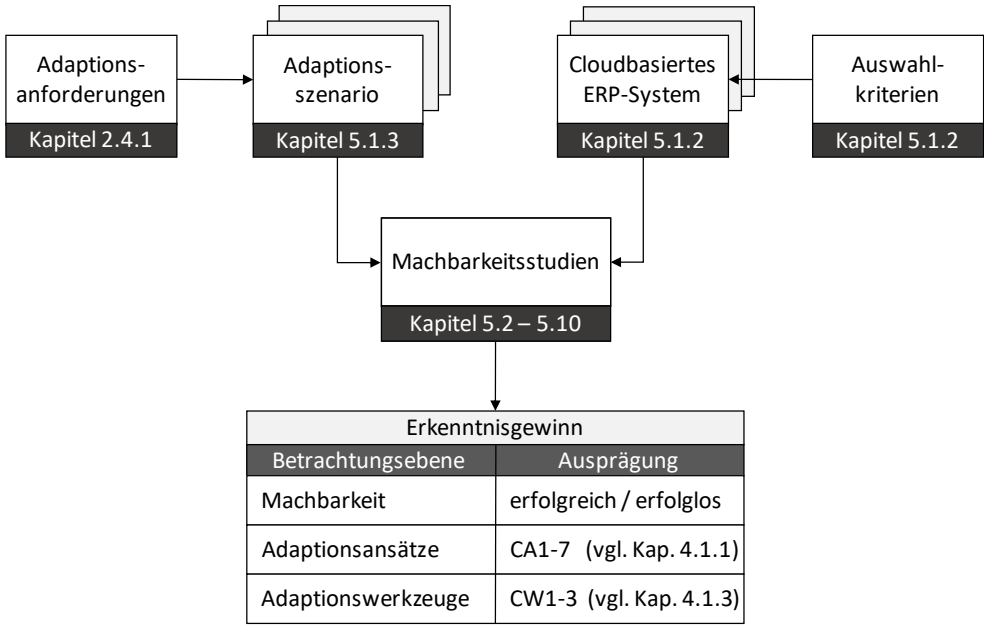



**Abbildung 5.1:** Struktur und Verlauf der explorativen Untersuchung

In diesem Kapitel werden anhand der im Kapitel 2.4.1 ermittelten Customizing-Anforderungen bezüglich einzelner Systemkomponenten beispielhafte Adaptionsszenarien konzipiert, mit deren Hilfe die Umsetzungsfähigkeit der einzelnen Adaptionen im Rahmen der repräsentativ ausgewählten cloudbasierten ERP-Systeme validiert wird.

In den Machbarkeitsstudien wird die technische Umsetzbarkeit der einzelnen Systemadaptionen unter Inanspruchnahme der intern und extern zur Verfügung stehenden Anpassungswerkzeuge untersucht. Diese werden für alle definierten Szenarien und ERP-Systeme durchgeführt. Jede erfolgreiche bzw. -lose Machbarkeitsstudie trägt zum Erkenntnisgewinn bei. In diesem Zusammenhang werden neben der allgemeinen Umsetzungsfähigkeit auch die für die Umsetzung bereitgestellten Werkzeuge bzw. Konzepte und die ihnen zugrunde gelegten Customizing-Ansätzen analysiert. Die Ergebnisse der durchgeführten Machbarkeitsstudien werden in den Kapiteln 5.2-5.10 beschrieben.

Jede vorgenommene Machbarkeitsstudie wird dabei als eine getrennte Fallstudie verstanden, die zur Beantwortung der Forschungsfragen bezüglich der durch die cloudbasierten ERP-Systeme bereitgestellten Anpassungstechniken, Werkzeuge und Konzepte beiträgt.

Anhand der vergleichenden Analyse der Machbarkeitsstudien, die bei mehreren cloudbasierten ERP-Systemen vorgenommen werden, können verallgemeinerbare Erkenntnisse hinsichtlich der bereitgestellten Anpassungs- und Modifikationsmöglichkeiten gewonnen werden. Durch die Zusammenstellung der gewonnenen Erkenntnisse bezüglich der vorgenommenen Adaptionen wird gezeigt, in welchem Umfang die Systemhersteller bzw. -anbieter erlauben, das erworbene System eigenständig zu adaptieren und somit, inwieweit dem Cloud-Computing-Prinzip des On-Demand-Selfservices nachgekommen wird. Daneben trägt jede Machbarkeitsstudie zur Bewertung der bereitgestellten Adaptionskonzepte hinsichtlich deren Effektivität bzw. des erforderlichen Implementierungsaufwands bei.

### 5.1.2 Untersuchungsgegenstände

Den definierten Untersuchungszielen wurden drei cloudbasierte ERP-Systeme als Untersuchungsgegenstände zugrunde gelegt. Die Systeme wurden gemäß der Methode einer kriterienorientierten Stichprobenziehung ausgewählt, bei der bekannte bzw. vorausgesetzte Eigenschaften der Untersuchungsgegenstände bei der Stichprobenauswahl berücksichtigt werden (vgl. Schreier 2011, S. 252). Dieses Vorgehen ist mit den Unterschieden im Funktionsumfang bzw. in der Komplexität der auf dem Markt verfügbaren Systeme begründet, welche die Vergleichbarkeit der Untersuchungsergebnisse beeinträchtigen könnten (vgl. Guo et al. 2011).

Um eine Vergleichbarkeit sicherzustellen, wurden einige Kriterien bei der Auswahl der Untersuchungsgegenstände zugrunde gelegt. In diesem Zusammenhang wurde vorausgesetzt, dass das System modular aufgebaut ist und seine Funktionalität mindestens unternehmerische Kernbereiche – Finanzbuchhaltung, Kunden- und Lieferantenmanagement, Vertrieb, Verkauf und Beschaffung – abdeckt. Darüber hinaus wurde erwartet, dass das ERP-System vollständig cloudbasiert ist und als solches über das Internet bereitgestellt wird. Bei den Systemen, die in unterschiedlichen Lösungspaketen bereitgestellt werden, wurde darauf geachtet, dass mit dem zum Test ausgewählten Lösungsumfang alle von dem jeweiligen Systemhersteller zur Systemadaption vorgesehenen Werkzeuge (z. B. API, IDE) bereitgestellt sind. Darüber hinaus soll entsprechende Flexibilität bei der Bearbeitung der Machbarkeitsstudien durch einen administrativen Systemzugang sichergestellt werden, der für einen anwenderseitigen Anwendungsexperten eingerichtet wird.

Basierend auf den genannten Voraussetzungen wurden drei cloudbasierte ERP-Systeme ausgewählt, welche als Untersuchungsgegenstände für die Machbarkeitsstudien benutzt wurden. Die

grundlegenden Charakteristika der ausgewählten ERP-Systeme sind in der Tabelle 5.1 gegenübergestellt.

**Tabelle 5.1:** Grundlegende Charakteristika der ausgewählten Untersuchungsgegenstände

| | **System I** | **System II** | **System III** |
|---|---|---|---|
| **Systemname** | SAP Business ByDesign | Scopevisio | Odoo |
| **Version** | 1602 | 201602291753 | 9.saas~9.1 |
| **Systemhersteller** | SAP SE | Scopevisio AG | Odoo S. A. |
| **Cloud-Modell** | Public Cloud | Public Cloud | Private Cloud |
| **Service-Modell** | PaaS | SaaS | SaaS |
| **Abgeleiteter SaaS-Reifegrad** | IV | IV | II |

*Untersuchungsgegenstand I – SAP Business ByDesign*

SAP Business ByDesign stellt ein cloudbasiertes ERP-System dar, das von SAP SE – dem weltweiten Marktführer im Bereich ERP-Systeme – entwickelt wurde. Das System wurde im Jahr 2007 als das On-Demand-ERP-System für kleine und mittelständische Unternehmen veröffentlicht und seither weiterentwickelt.

Das cloudbasierte ERP-System von SAP besteht aus mehreren Funktionen und Modulen, welche unter anderem die Kernbereiche Rechnungswesen, Geschäftspartnermanagement, Fertigung, Materialmanagement, Verkauf und Vertrieb, Personalwesen und Projektmanagement abdecken (vgl. SAP SE 2017). Die einzelnen Systembestandteile können im Rahmen des On-Demand-Selfservices von dem Anwenderunternehmen eigenständig ein- bzw. abgeschaltet werden. Die monatliche Abrechnung erfolgt anhand der Benutzerzahlen.

Business ByDesign wird ausschließlich im Rechenzentrum von SAP betrieben und dem Kundenunternehmen in Form einer Dienstleistung über das Internet zur Verfügung gestellt. Dabei werden die konventionellen Internetbrowser als Clients verwendet, wodurch die Plattformunabhängigkeit sichergestellt wird. Als ein völlig mandantenfähiges und hoch skalierbares System erfüllt Business ByDesign die im Kapitel 4.1.4 abgeleiteten Anforderungen der vierten Stufe des SaaS-Reifegradmodells. Darüber hinaus ermöglicht es der Systemhersteller, den cloudbasierten Systembetrieb mit der Funktionalität der Plattform-as-a-Service (PaaS) zu erweitern. Dadurch können die Kunden den standardisierten Lösungsumfang um weitere eigenentwickelte bzw. fremdbeschaffte Systemkomponenten ergänzen.

Neben dem öffentlichen Cloud-Bereitstellungsmodell, kann Business ByDesign in einer gehosteten, privaten Cloud bereitgestellt werden. Dieses Bereitstellungsmodell richtet sich an den Kundenunternehmen aus, welche sich für die Mindestanzahl von zehn Mandanten entscheiden (vgl. SAP SE 2017). Im Rahmen der Machbarkeitsstudien wurde das ERP-System (SAP Business ByDesign Version 1602) in einem öffentlichen Bereitstellungsmodell verwendet, welches die PaaS-Funktionalität sicherstellt. Dies ermöglichte für die Bearbeitung der Beispielszenarien alle durch das System bereitgestellte Adaptions- und Modifikationsmöglichkeiten anzuwenden, die von einem Kundenunternehmen im Rahmen eines eigenständigen Customizing verwendet werden können.

*Untersuchungsgegenstand II – Scopevisio*

Scopevisio stellt ein On-Demand-ERP-System dar, das von dem Systemhersteller Scopevisio AG mit Sitz in Bonn vertrieben und weiterentwickelt wird. Scopevisio wurde im Jahr 2008 als Buchhaltungssoftware veröffentlicht und seither von dem Systemhersteller systematisch mit weiteren Modulen ergänzt (vgl. Marwan 2011). Das System richtet sich an die Dienstleistungsunternehmen im deutschsprachigen Raum und wird heutzutage bei mehr als tausend Anwenderunternehmen eingesetzt (vgl. Scopevisio AG 2016b).

Die Kernbereiche von Scopevisio umfassen die Finanzbuchhaltung, das Kunden- und Lieferantenmanagement, den Vertrieb, das Projektmanagement, Marketing und Dokumentenmanagement. Die Systemfunktionen können von dem Anwenderunternehmen einzeln zusammengestellt, bzw. in einem der drei von dem Systemhersteller bereitgestellten Paketen gemietet werden (vgl. Scopevisio AG 2016c). Die Kundenabrechnung erfolgt dabei anhand der aktiven Benutzerzahlen der jeweils gemieteten Systemmodule (vgl. Scopevisio AG 2016d).

Scopevisio stellt das cloudbasierte ERP-System dar, welches ausschließlich in einem öffentlichen Cloud-Bereitstellungsmodell angeboten wird. Der Benutzerzugriff auf die ERPaaS erfolgt über einen lokal installierbaren Thin-Client, der personalisiert zum Herunterladen bereitgestellt wird. Als eine cloudnative SaaS-Lösung ist die Software durch Mandantenfähigkeit und Skalierbarkeit gekennzeichnet und stellt die Konfigurierbarkeit im Rahmen des On-Demand-Selfservice sicher. Vergleicht man die Systemcharakteristika mit den aufgenommenen Kriterien (vgl. Tabelle 4.5), so ist Scopevisio durch den vierten SaaS-Reifegrad gekennzeichnet.

Im Rahmen der Machbarkeitsstudien wurde das mittlere Paket – *Scopevisio BusinessLine* – verwendet. Diese Option wurde ausgewählt, da sie alle Systemmodule beinhaltet und zugleich die OpenScope-API bereitstellt, welche im Rahmen des kleineren Pakets nicht enthalten ist (vgl. Scopevisio AG 2016e). Der Unterschied zwischen dem ausgewählten Lösungsumfang und dem größten *EnterpriseLine-Paket* besteht hauptsächlich in der verfügbaren Speicherkapazität, welche ohne Einfluss auf die vorgenommene Untersuchung bleibt.

*Untersuchungsgegenstand III – Odoo*

Odoo stellt ein quelloffenes ERP-System dar, welches ab der ersten Veröffentlichung im Jahr 2005 von seiner Entwicklergemeinschaft kontinuierlich weiterentwickelt wird. Das ursprünglich für einen On-Premise-Betrieb konzipierte System wurde im Jahr 2011 mit einem Web-Client ausgestattet und systematisch an den Cloud-Betrieb angepasst. Heutzutage bietet der Software-Editor – Odoo S. A.– eine cloudfähige Version des Systems an (vgl. Odoo S. A. 2016g), die über einen konventionellen Internetbrowser zur Verfügung gestellt werden kann. Im Gegensatz zu dem On-Premise-System wird die cloudbasierte Systemversion entgeltlich angeboten.

Odoo ist ein stark modularisiertes System mit ca. 30 Kernmodulen, die mit weiteren 400 von der Entwicklergemeinschaft entwickelten Erweiterungen ergänzt werden können. Die Kernmodule des On-Demand-ERP-Systems umfassen unter anderem die Waren- und Finanzwirtschaft, Produktionsplanung, das Projektmanagement, Marketing und Personalmanagement. Die einzelnen Systembestandteile können von dem Anwenderunternehmen beliebig zusammengestellt werden. Die monatliche Abrechnung erfolgt anhand der gemieteten Anwendungen und der Anzahl der aktiven Accounts (vgl. Odoo S. A. 2016f).

Odoo stellt ein cloudfähiges ERP-System, welches sowohl in einem On-Premise- als auch in einem On-Demand-Modell zur Verfügung gestellt werden kann. Das cloudfähige ERP-System kann in einer privaten virtuellen Cloud zur Verfügung gestellt werden, der eine mandantendedizierte Datenbank zugrunde liegt. Als eine cloudfähige, modularisierte und konfigurierbare Lösung ist Odoo durch den zweiten SaaS-Reifegrad gekennzeichnet.

Für die bevorstehenden Machbarkeitsstudien wurde eine individuelle Zusammenstellung der sechs oben genannten Kernmodule ausgewählt, welche in einer privaten Cloud bereitgestellt wurden.

### 5.1.3 Motivation der ausgewählten Szenarien

Im Hinblick auf die bevorstehende Systematisierung und Bewertung der Adaptionsmöglichkeiten der cloudbasierten ERP-Systeme wurde, basierend auf der Analyse der häufig auftretenden Customizing-Anforderungen (siehe Kapitel 2.4.1), die konzeptuelle Entwicklung der Beispielszenarien vorgenommen. Die Tabelle 5.2 stellt eine zusammenfasende Übersicht der konzipierten Adaptierbarkeitsszenarien dar, welche die Grundlage der Machbarkeitsstudien bilden.

**Tabelle 5.2:** Übersicht der konzipierten Adaptierbarkeitsszenarien

| Szenario | Beschreibung | Systemelement | Kapitel |
|---|---|---|---|
| Änderung des modularen Systemumfangs | Diese Fallstudie setzt eine nachträgliche Implementierung eines der verfügbaren Systemmodule voraus, welches die bestehende Systemfunktionalität um weitere Funktionen ergänzt. | SE1 | 5.2 |
| Änderung der Parametereinstellungen | Dieses Szenario umfasst eine Änderung der funktionsbezogenen Einstellungen eines ausgewählten Systemmoduls durch die Vergabe der neuen Parameterwerte. | SE2 | 5.3 |
| Adaption der Systemoberfläche | Diese Fallstudie bezieht sich auf die Anpassung der Startseite eines ERP-Systems durch Änderung der farblichen Gestaltung und Komposition der Benutzeroberfläche sowie durch das Anbinden von unternehmensinternen und externen Informationen in Form von Berichten, Landkarten, Nachrichten etc. | SE3 | 5.4 |
| Erweiterung der Geschäftsobjekte und Dokumentenvorlagen | Dieses Szenario umfasst eine Erweiterung des bestehenden Geschäftsobjekts (Kundenauftrag) um ein benutzerdefiniertes Eingabefeld. Das neu definierte Feld soll nachträglich auf der Dokumentenvorlage referenziert werden. | SE4 | 5.5 |
| Erweiterung der Geschäftsanalytik | Dieses Szenario ist auf die Definition einer neuen Kennzahl ausgerichtet, der ein benutzerdefiniertes Eingabefeld zugrunde liegt. Folglich soll im Rahmen der Machbarkeitsstudie die zeitliche Entwicklung der Kennzahl anhand eines Liniendiagramms visualisiert werden. | SE5 | 5.6 |
| Adaption der Aufbauorganisation | Im Rahmen dieses Szenarios soll die bestehende Organisationsstruktur des jeweiligen Musterunternehmens um eine neue Niederlassung erweitert werden. Das Szenario setzt neben der allgemeinen Konfiguration der neu angelegten Organisationseinheit eine Zuweisung des Niederlassungsleiters voraus. | SE6 | 5.7 |
| Adaption des Prozessablaufs | Dieses Szenario umfasst eine Erweiterung des standardisierten Verkaufsprozesses, um einen zusätzlichen Genehmigungsschritt, welcher in Abhängigkeit von dem Auftragswert und dem Bestellkanal ausgeführt werden soll. | SE7 | 5.8 |
| Integration externer Webanwendung | Diese Aufgabestellung konzentriert sich auf die Integration des ERP-Systems und eines (webbasierten) Dienstes auf der Ebene des Datenaustausches und der Systemoberfläche. | SE8 | 5.9 |
| Entwicklung und Integration neuer Systemfunktionalität | Dieses Szenario umfasst eine Entwicklung der benutzerdefinierten Systemfunktionalität in Form eines Systemmoduls und dessen nachträgliche Implementierung in Systemrahmen. | SE9 | 5.10 |

Für die Bewertung der systemübergreifenden Adaptierbarkeit der cloudbasierten ERP-Systeme wurden neun verschiedene Adaptionsszenarien entwickelt, welche jeweils ein unterschiedliches Systemelement betreffen und zusammengestellt die relevantesten Anpassungsanforderungen (vgl. Kapitel 4.1.2) abdecken. Die Szenarien wurden so konzipiert, dass sie jeweils möglichst auf eine konkrete Systemadaption fokussieren, die getrennt von anderen Adaptionen durchgeführt werden kann. Dies wurde vorgenommen, um den möglichen Effekt auszuschließen, dass wegen direkter Abhängigkeit zwischen den Beispielszenarien die Umsetzung der Folgeaktionen durch erfolgslose Machbarkeit der vorigen Fallstudie beeinträchtigt wird.

Anhand der definierten Systemadaptionen wird überprüft, ob der jeweiligen Anforderung an eigenständiges Customizing im Rahmen der cloudbasierten ERP-Systeme entsprochen werden kann. Darüber hinaus soll im Rahmen des einzelnen Beispielszenarios überprüft werden, wie die vorgesehene Systemadaption durch den Anwendungsexperten eigenständig umgesetzt werden kann und welche Werkzeuge zur Unterstützung des Anpassungsprozesses bereitgestellt werden.

Diese Untersuchung bezieht sich hauptsächlich auf die Validierung der anwenderseitigen Customizing- und Modifikationsmöglichkeiten der untersuchten Cloud-ERP-Systeme. Sind von dem Systemhersteller alternative Adaptionsmöglichkeiten zugelassen, die z. B. als partnerseitige Supports etc. angeboten werden, wird auf diese Dienstleistungen in der Beschreibung der jeweiligen Machbarkeitsstudie explizit hingewiesen.

Die Reihenfolge der Machbarkeitsstudien entspricht der ansteigenden Komplexität der geplanten Systemadaption und ist durch die letzte Spalte der Tabelle 5.2 bestimmt. In jedem Unterabschnitt werden die Anpassungsmöglichkeiten der einzelnen ERP-Systeme hinsichtlich der geplanten Adaption analysiert und die Ergebnisse der vorgenommenen Machbarkeitsstudien beschrieben.

## 5.2 Änderung des modularen Systemumfangs

Die Änderung des initialen, modularen Umfangs eines ERP-Systems stellt eine häufig auftretende Anforderung dar, die unter anderem als Resultat des Organisationswachstums und Erweiterung der funktionalen Organisationsbereiche entstehen kann. Die Erweiterung des Systems durch Implementierung erforderlicher Systemmodule ist allerdings nur dann möglich, wenn sich die Module im Angebot des Systemherstellers bzw. der Drittanbieter befinden. Ist diese Voraussetzung nicht erfüllt, muss die benötigte Systemfunktionalität in der Regel durch eine Individualentwicklung bereitgestellt werden (vgl. Kapitel 5.10).

Das Beispielszenario setzt die Implementierung eines der verfügbaren Systemmodule – Finanzbuchhaltung, Vertrieb, Verkauf oder Beschaffung – in den bestehenden Lösungsumfang voraus. Der aufgestellten Voraussetzung des modularen Systemaufbaus und der Verfügbarkeit der Module, die der Auswahl der Untersuchungsgegenstände zugrunde gelegt wurden, folgend, wird im Rahmen dieses Szenarios überprüft, ob eine nachträgliche Erweiterung des jeweiligen Lösungsumfangs möglich ist. Dabei soll insbesondere die Frage beantwortet werden, ob die Änderung des Lösungsumfangs von dem Anwenderunternehmen eigenständig vorgenommen werden kann und welche Werkzeuge zur Unterstützung der Adaption initialer Systemzusammensetzung bereitgestellt werden.

### 5.2.1 System I: Business ByDesign

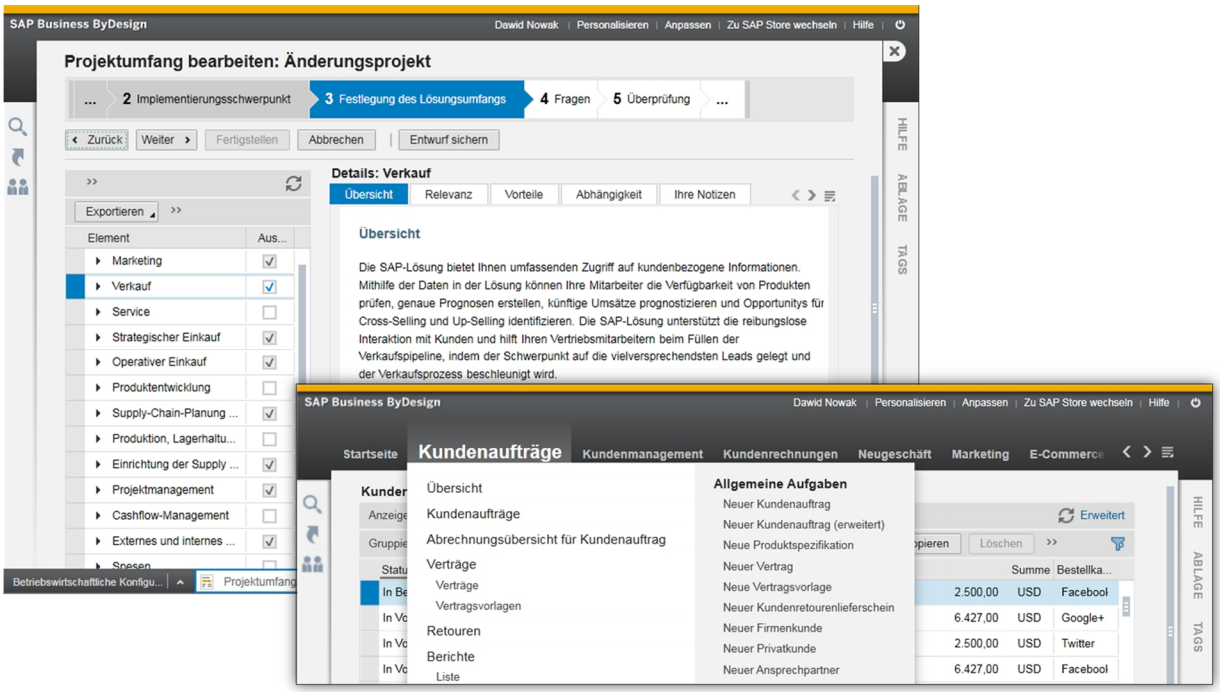
Die Festlegung der Systemzusammenstellung stellt in Business ByDesign den integralen Teil des Prozesses der betrieblichen Konfiguration dar, in dessen Rahmen der Systemumfang an die Anforderungen des Kundenunternehmens angepasst wird. Die initiale Zusammenstellung der Systemkomponenten wird in der Implementierungsphase vorgenommen und kann in der Post-

implementierungsphase gemäß der geänderten Anforderungen des Anwenderunternehmens angepasst werden. Zum Zweck der Festlegung des Systemumfangs stehen dem Anwendungsexperten die integrierten Werkzeuge zur Verfügung.

Die Grundlage für den Prozess der Festlegung des Lösungsumfangs bildet der sogenannte *Adaptionskatalog*, der alle Systemfunktionen umfasst und hierarchisch anordnet. Mithilfe des eingebauten Assistenten werden anhand des Adaptionskatalogs diejenigen Module und Funktionen ausgewählt, die für das Unternehmen relevant sind. Der Assistent stellt in diesem Zusammenhang eine Abfolge von Dialogen dar, die von dem Anwendungsexperten die benötigten Informationen erfragen und anhand derer die Parameterwerte zu den vordefinierten Parametern zuweisen (vgl. Weidenhaupt 2001, S. 31). Auf Basis der ausgewählten Systembestandteile wird das unternehmensindividuelle Konfigurationsprofil des Systems erstellt, das in der nächsten Phase der betrieblichen Konfiguration parametrisiert werden muss.

Werden zu einem späteren Zeitpunkt in der Betriebsphase weitere Komponenten benötigt, die sich in dem ursprünglich ausgewählten Lösungsumfang nicht befinden, muss eine Rekonfiguration des Systems vorgenommen werden. Die Rekonfiguration erfolgt im Rahmen des sogenannten *Änderungsprojektes*, das auf die Erstellung eines neuen Konfigurationsprofils abzielt. Das Änderungsprojekt ähnelt der ersten Implementierung und wird durch die gleichen integrierten Werkzeuge unterstützt.

Das Beispiel einer nachträglichen Erweiterung des Lösungsumfangs im Rahmen eines Änderungsprojektes wurde in der Abbildung 5.2 dargestellt.




© SAP SE

**Abbildung 5.2:** Änderung der modulbasierten Zusammenstellung von Business ByDesign

Die Abbildung im Hintergrund stellt eine Übersicht über den Prozessschritt der Festlegung des Lösungsumfangs im Rahmen eines Änderungsprojektes dar. Die Abbildung im Vordergrund veranschaulicht die neu aktivierten Systembereiche im Anwendermodus.

Im Rahmen des Beispielszenarios wurde das standardisierte ERP-System um die Komponente – *Verkauf* – erweitert. Diese Systemkomponente ist standardmäßig in dem Adaptionskatalog

enthalten und kann von dem Anwendungsexperten im Rahmen eines Änderungsprojektes in den Lösungsumfang übernommen werden (siehe Abbildung 5.2, links). Nach erfolgreichem Erweiterungsvorgang erweitert die neuimplementierte Funktionalität den bisherigen Lösungsumfang und ist über die Benutzeroberfläche zugänglich (siehe Abbildung 5.2, rechts). Im Rahmen des aktivierten Verkauf-Systemmoduls wurden die Systembereiche – *Kundenaufträge*, *Kundenmanagement*, *Kundenrechnungen* und *Neugeschäft* – aktiviert, die jeweils ihre Funktionalität in der Systemoberfläche zur Verfügung stellen.

Der im Hintergrund der Abbildung 5.2 (links) angezeigte Adaptionskatalog kann auch um weitere nicht standardisierte Komponenten ergänzt werden, die im Rahmen der Erstimplementierung bzw. eines Änderungsprojektes in den Lösungsumfang übernommen werden können. Dies kann entweder durch Einkauf der benötigten Systemerweiterungen über die kommerzielle Plattform – *SAP Store* – bzw. durch die Entwicklung der Systemkomponenten mithilfe der von dem Systemhersteller bereitgestellten Entwicklungsumgebung – *SAP Cloud Applications Studio* – vorgenommen werden (vgl. Konstantinidis et al. 2012, S. 622 ff.).

SAP Cloud Applications Studio stellt eine integrierte Entwicklungsumgebung (IDE) dar, die die Adaption bzw. Erweiterung von Business ByDesign ermöglicht. Die IDE stellt eine lokal installierbare Software dar, die an das System-Backend des in der Cloud betriebenen ERP-Systems des Kundenunternehmens gekoppelt werden kann. Mithilfe der bereitgestellten Entwicklungsumgebung haben die Entwickler die Möglichkeit, die Systemerweiterungen zu programmieren und direkt im System des Kundenunternehmens zu aktivieren. Die entwickelten Systemerweiterungen können auch über die von SAP bereitgestellte Vertriebsplattform – SAP Store – an andere Anwenderunternehmen verkauft werden. Nach einer Zertifizierung der mithilfe der IDE erstellten Systemerweiterung kann diese in SAP Store veröffentlicht werden. Die Kunden können im Store nach Systemerweiterungen suchen, die Lösungen testen und kaufen. Die über SAP Store erworbene Funktionalität erweitert den Adaptionskatalog des Kundensystems und kann von einem Anwendungsexperten im Rahmen eines Änderungsprojektes aktiviert werden (vgl. Schneider 2012, S. 454).

Bei schwerwiegenden Änderungen des Lösungsumfangs, bei denen die Auswirkung der zu implementierenden Komponenten auf die bestehenden Systembestandteile aufgrund deren Komplexität schwierig abzuschätzen ist, kann im Rahmen eines On-Demand-Selfservices ein Testsystem angefordert werden, das eine Kopie des laufenden ERP-Systems darstellt (vgl. Konstantinidis et al. 2012, S. 209 ff.). Damit kann die Auswirkung der geplanten Systemadaptionen auf die bestehenden Lösungskomponenten vor der endgültigen Übernahme in die laufende Lösung überprüft werden.

### 5.2.2 System II: Scopevisio

Das ERP-System von Scopevisio besteht aus fünf Kernmodulen, die je nach Bedarf des Anwenderunternehmens bei der Erstimplementierung ausgewählt und integriert werden können. Der bei der Erstimplementierung erstellte Lösungsumfang kann nachträglich während der Betriebsphase geändert werden. Das nachträgliche Hinzufügen neuer Module und Funktionen erfolgt in Form eines On-Demand-Selfservices mithilfe der im System eingebauten *Administration-Werkzeuge* und kann von dem Anwendungsexperten vorgenommen werden.

Der Anwendungsexperte ist berechtigt, die predefinierten Module und Funktionen über die proprietäre Vertriebsplattform – *Scopevisio Shop* – einzukaufen. Der Scopevisio-Shop ist über die Benutzeroberfläche des ERP-Systems zugänglich und bietet zusätzliche Komponenten an, die in Form neuer Module bzw. mobiler Anwendungen vermarktet werden. Im Rahmen der webbasierten Vertriebsplattform werden die Anwendungen angeboten, die von dem Systemhersteller entwickelt und bereitgestellt werden.

Die Abbildung 5.3 veranschaulicht den Prozess der Erweiterung des initialen Lösungsumfangs um die neuen Systembereiche (*Abrechnung* und *Projekte)*.

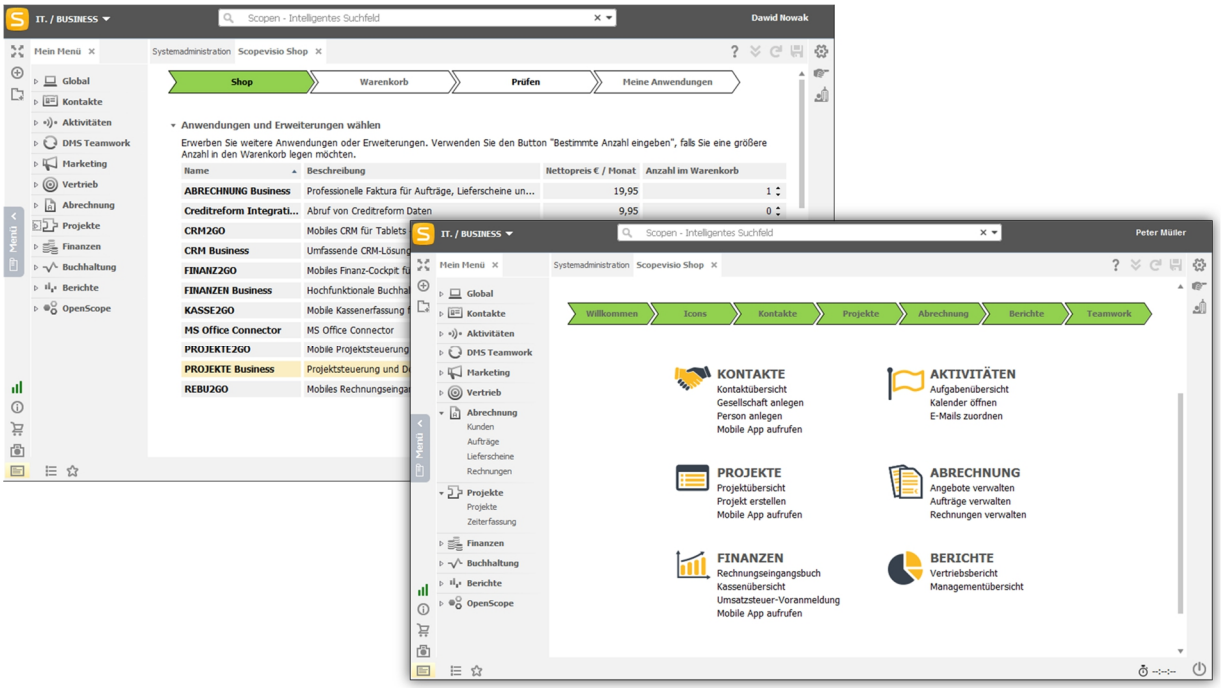





**Abbildung 5.3:** Änderung der modulbasierten Zusammenstellung von Scopevisio

Die Abbildung im Hintergrund stellt den integrierten *Scopevisio Shop-Assistenten* dar, der den Anwendungsexperten durch die einzelnen Schritte der Systemerweiterung führt. Der Anwendungsexperte legt die benötigte Anzahl an Accounts zu jedem der ausgewählten Module fest und bestätigt nach dem erfolgreichen Prüfungsvorgang die Kauftransaktion. Nachdem die neuen Funktionen bezahlt worden sind, werden sie im System des Anwenderunternehmens aktiviert und können den einzelnen Systembenutzern zugewiesen werden. Der um die Abrechnung- und Projekte-Funktionalität erweiterte Lösungsumfang wird im Vordergrund angezeigt. Die zwei neu implementierten Module erscheinen auch als Einträge in dem *Mein Menü-Bereich* (linker Bereich des Fensters).

### 5.2.3 System III: Odoo

Der initiale Lösungsumfang des cloudbasierten Odoo-ERP-Systems wird aus den zur Verfügung stehenden Kernmodulen bei der Implementierungsphase zusammengestellt. Die initial festgelegte Systemzusammenstellung kann nachträglich während der Betriebsphase im Rahmen eines On-Demand-Selfservices angepasst werden.

In diesem Zusammenhang haben die unternehmensinternen Administratoren einen Zugriff auf die integrierten *Administrationswerkzeuge*, die ihnen ermöglichen, den bestehenden Lösungsumfang entsprechend der Anforderungen des Anwenderunternehmens anzupassen. Das zentrale Werkzeug für die Adaption der modularen Systemzusammenstellung stellt das *Apps-Werkzeug* dar, das eine Liste der gerade installierten als auch der noch installierbaren Systemkomponente beinhaltet. Das Werkzeug ist über die Systemoberfläche zugänglich und ermöglicht, unternehmensinternen Administratoren die einzelnen Systembestandteile ein- bzw. abzuschalten.

Odoo stellt im Standard ca. 400 Systemkomponenten, die mittels des integrierten Apps-Werkzeuges in den Lösungsumfang eigenständig übernommen werden können. Die Liste der Systemerweiterungen enthält die Module bzw. Funktionen, die sowohl von dem Systemanbieter selbst als auch von den unabhängigen Entwicklern bereitgestellt werden.

Bei der Erweiterung des Lösungsumfangs überprüft das System automatisch, ob alle für die Integration des ausgewählten Moduls erforderlichen Bestandteile im Lösungsumfang vorhanden sind und ob eine Integration erfolgen kann. Die Installation des neu ausgewählten Moduls verläuft in Form eines automatisierten Prozesses, welcher durch das System im Hintergrund ausgeführt wird.

Die Abbildung 5.4 stellt das Apps-Werkzeug und den erweiterten Lösungsumfang von Odoo dar.

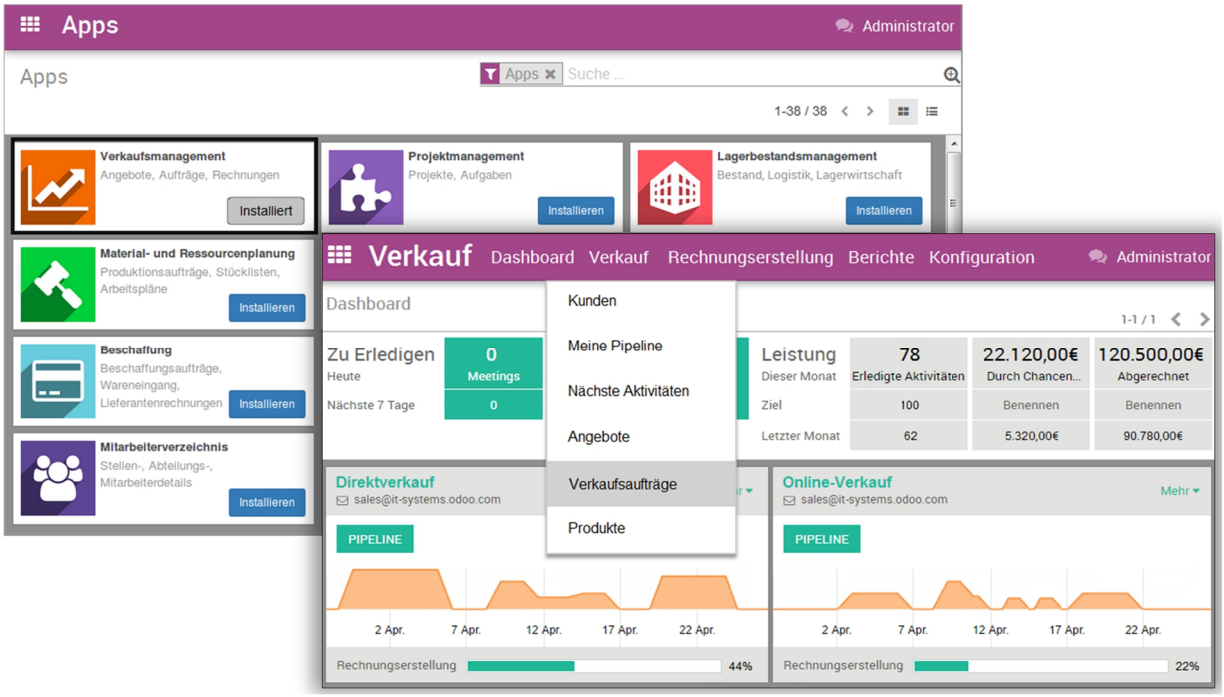





**Abbildung 5.4:** Änderung der modulbasierten Zusammenstellung von Odoo

Der Screenshot im Hintergrund veranschaulicht den Prozess der Modulauswahl. Das Apps-Werkzeug stellt die Benutzeroberfläche dar, in der die verfügbaren funktionalen Odoo-Module in Form der Rechtecke aufgelistet sind. Anhand der Schaltflächen, die unter jedem Modul angezeigt werden, kann entnommen werden, welche Module sich gerade im Lösungsumfang befinden und welche dagegen noch installierbar sind. Die einzelnen Systembestandteile werden in Form der Icons aufgelistet, die per Klick installiert werden können.

Im Rahmen des Beispielszenarios wurde das *Verkaufsmanagement-Modul* ausgewählt und installiert. Der Installationsprozess verläuft im Hintergrund und endet mit dem Hinzufügen des Modul-Icons zum Hauptmenü auf der Systemstartseite. Die installierte Komponente erweitert den initialen Lösungsumfang und kann sofort über die Benutzeroberfläche benutzt werden. Die Abbildung im Vordergrund präsentiert die Übersichtsseite (Dashboard) des neu installierten Systemmoduls im Anwendermodus und verschafft einen Überblick über die neu aktivierten Modulbereiche, die in der Drop-Down-Liste aufgelistet sind (siehe Abbildung 5.4, rechts).

## 5.3 Änderung der Parametereinstellungen

Die Anpassung der prozess- und funktionsbezogenen Parametereinstellungen stellt eine der wichtigen und relativ häufig auftretenden Aufgaben der Postimplementierungsphase dar. Das Anwenderunternehmen soll auch im Rahmen eines cloudbasierten ERP-Systems die Möglichkeit haben, auf die geänderten unternehmensinternen und -externen Gegebenheiten entsprechend reagieren zu können. Dies kann unter anderem durch eine nachträgliche Änderung der initial festgelegten Parameterwerte vorgenommen werden, soweit die erforderlichen Funktionen in dem Lösungsumfang vorhanden sind und durch die Parametrisierung aktiviert werden können.

Aufgrund vorhersehbarer, funktionaler Vielfältigkeit der untersuchten Systeme und der potenziellen Unterschiede in dem parametrisierbaren Systemumfang wird der Fallstudie keine bestimmte Voraussetzung bezüglich der zu ändernden Parameter zugrunde gelegt. Vielmehr konzentriert sich die Machbarkeitsstudie auf eine demonstrative Änderung ausgewählter Parametereinstellungen des ursprünglich installierten Systemmoduls (siehe Abschnitt 5.2) und die Veranschaulichung der daraus resultierenden Systemanpassung.

Die Machbarkeitsstudien dieses Abschnitts tragen exemplarisch zur Beantwortung der Frage bei, ob eine nachträgliche Anpassung der Einstellungen einzelner Systemfunktionen von dem Anwendungsexperten im Rahmen eines On-Demand-Selfservices vorgenommen werden kann und werfen zugleich mehr Licht auf die Form der Abwicklung der Parametrisierungsprozesse bei den cloudbasierten ERP-Systemen.

### 5.3.1 System I: Business ByDesign

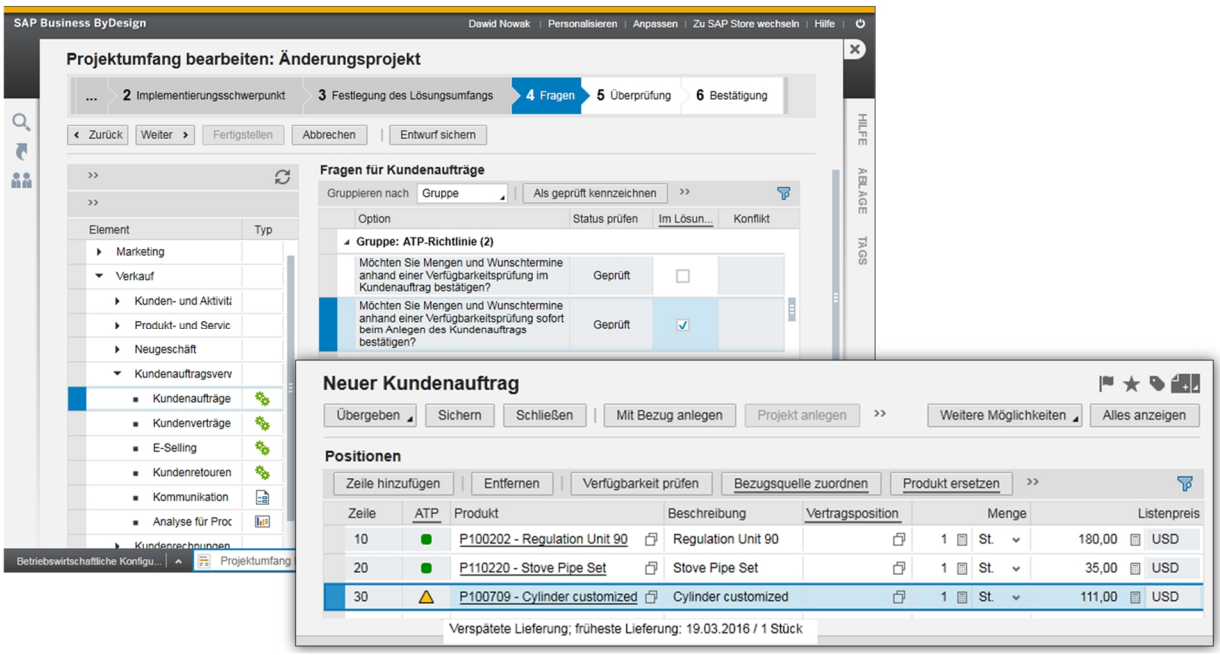
Die Parametrisierung baut auf dem Lösungsumfang auf, der bei der Phase der modulbasierten Systemzusammenstellung angefordert wurde. Zwecks der Vereinfachung der Parametrisierungsphase stellt Business ByDesign einen automatisierten Parametrisierungsprozess mit integrierter Kundenentscheidungsunterstützung bereit.

Sowohl im Fall der initialen Parametrisierung in der Implementierungsphase als auch bei den Reparametrisierungen der Betriebsphase wird der Anwendungsexperte von einem eingebauten Assistenten durch alle Schritte des Parametrisierungsprozesses begleitet. Im Rahmen dieses Prozesses werden durch das System an den Anwendungsexperten die unternehmensbezogenen Fragen gestellt, die den kompletten Lösungsumfang abdecken. Zu jeder gestellten Frage bietet der Assistent die möglichen Auswahloptionen sowie den Vorschlagswert an, der anhand der bewährten Geschäftspraktiken (Best Practice; siehe Kapitel 2.3.1) bestimmt worden ist. Dabei überprüft das System automatisch die Abhängigkeiten zwischen den zu parametrisierenden Komponenten des ausgewählten Lösungsumfangs und weist auf die eventuell entstandenen Inkonsistenzen hin, die von den Anwendungsexperten beseitigt werden müssen.

Anhand der Antworten, die der Anwendungsexperte auf die gestellten Fragen erteilt, werden im Hintergrund die Parameterwerte zu den einzelnen Parametern des unternehmensindividuellen Konfigurationsprofils des Systems zugewiesen. Die festgelegten Parameterwerte können jederzeit wieder geändert werden. Der eventuelle Prozess einer Änderung der aktuell festgelegten Systemeinstellungen wird durch die gleichen Parametrisierungswerkzeuge unterstützt.

Das Beispiel einer nachträglichen Änderung der Systemparameter während der Betriebsphase wurde in der Abbildung 5.5 dargestellt. Im Rahmen der Fallstudie wurde das Änderungsprojekt angelegt, in dessen Rahmen der Zeitpunkt der mengen- und zeitbezogenen Verfügbarkeitsprüfung (der sogenannten Available-to-Promise-Prüfung) der im Kundenauftrag aufgelisteten Positionen geändert werden soll. Anstatt die ATP-Prüfung erst im nachfolgenden Schritt des Auftragsabwicklungsprozesses durchzuführen, soll die Prüfung bereits beim Anlegen des

Kundenauftrags vorgenommen werden. Aus diesem Grund wurde im linken Bereich des im Hintergrund angezeigten Parametrisierung-Fensters das Element – *Kundenaufträge* – ausgewählt und im rechten Bereich die obere durch die untere Auswahloption ersetzt. Nach erfolgreicher Überprüfung und Sicherung der vorgenommenen Einstellung wird die neue ATP-Richtlinie sofort bei dem nächsten Kundenauftrag wirksam.




© SAP SE

**Abbildung 5.5:** Parametrisierung der ATP-Prüfung in Business ByDesign

Beim Hinzufügen der neuen Kundenauftragspositionen wird die ATP-Prüfung umgehend vom System durchgeführt. Dies ist auf der Abbildung (rechts) erkennbar. Die ATP-Spalte der Positionen-Tabelle gibt das Ergebnis der Materialverfügbarkeitsprüfung wieder. Die Rechtecke entsprechen dem positiven Ergebnis der Prüfung, während die Dreiecke auf die fehlende Verfügbarkeit des bestellten Produktes hindeuten. In dem präsentierten Beispiel ist das negative Ergebnis der ATP-Prüfung bei der dritten Auftragsposition durch eine Warnmeldung signalisiert. Das System informiert den Anwender dabei über eine verspätete Lieferung und ermittelt den frühestmöglichen Lieferzeitpunkt.

Ähnlich können mit den bereitgestellten Parametrisierungswerkzeugen weitere Änderungen der initial festgelegten Parameterwerte vorgenommen werden. Business ByDesign stellt im Gesamtumfang ca. 1.200 Parameter bereit, die in dem gesamten Systemlebenszyklus mithilfe der integrierten Werkzeuge eingestellt werden können.

### 5.3.2 System II: Scopevisio

Im Rahmen des cloudbasierten Systems von Scopevisio werden dem Anwenderunternehmen die integrierten *Administration-Werkzeuge* zur Verfügung gestellt. Mithilfe der eingebauten Werkzeuge hat der Anwendungsexperte die Möglichkeit, die unternehmenskonformen Änderungen der systemübergreifenden und modulbezogenen Einstellungen in dem gesamten Systemlebenszyklus vorzunehmen.

Die in Scopevisio bereitgestellte Funktionalität ist allerdings hoch standardisiert und lediglich innerhalb bestimmter Systembereiche parametrisierbar. Der Systemhersteller ermöglicht dem Anwenderunternehmen neben den allgemeinen Systemeinstellungen, den land- und branchen-

bezogenen Parametern vor allem, die Anpassungen der Finanzbuchhaltung (des Kontenplans, der Nummernkreise und der Zahlungsbedingungen) selbständig vorzunehmen.

Zum Zweck der Parametrisierung der Stammdaten wird ein Stammdatenassistent zur Verfügung gestellt, der den Anwendungsexperten durch die einzelnen Schritte der Einrichtung der Finanzbuchhaltung führt. Die Parametrisierung erfolgt durch Auswahl der predefinierten Auswahloptionen, auf deren Basis das unternehmensbezogene Konfigurationsprofil im Hintergrund erzeugt wird.

Die Abbildung 5.6 veranschaulicht die Parametrisierung des Prozessschritts *Rechnungseingang* (im Hintergrund) und zeigt die Auswirkung der vorgenommenen Änderungen (im Vordergrund).

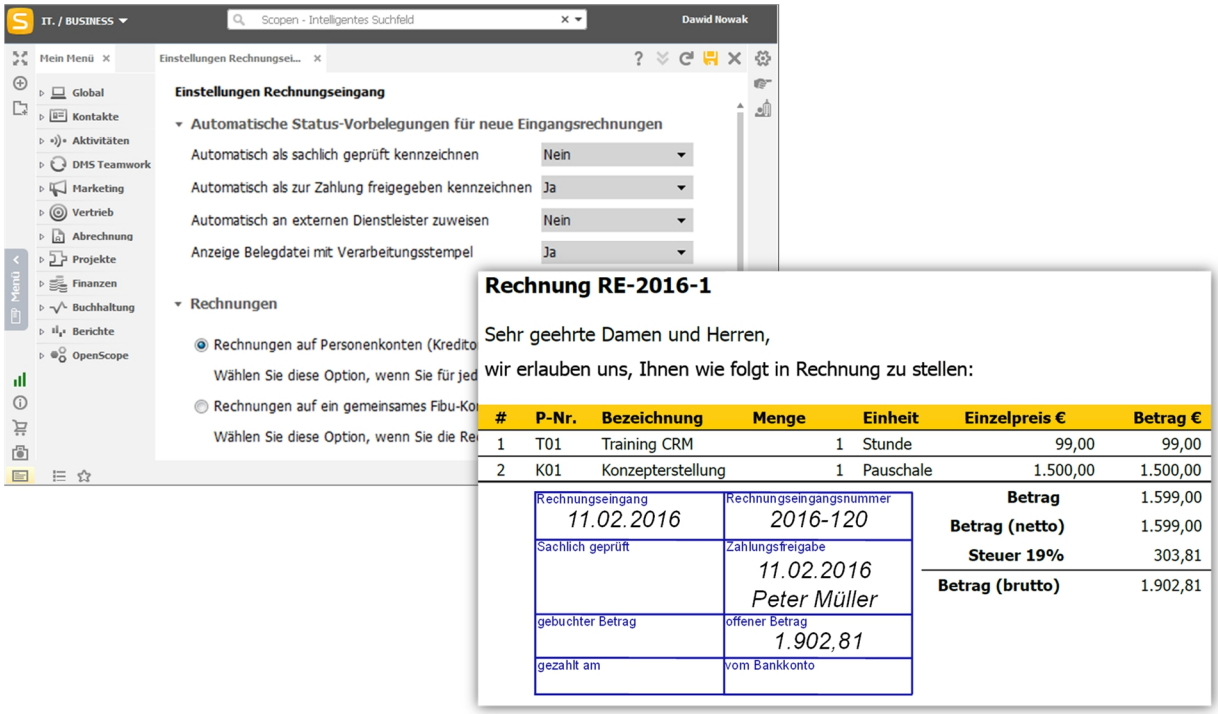





**Abbildung 5.6:** Parametrisierung des Rechnungseingangs in Scopevisio

Im Rahmen des Beispielszenarios wurden die standardisierten Einstellungen des Rechungseingangs geändert, indem zwei der automatischen Status-Vorbelegungen für neue Eingangsrechnungen aktiviert wurden. Infolge der vorgenommenen Einstellungen werden die neu erfassten Lieferantenrechnungen (rechts) durch das System automatisch als *„zur Zahlung freigegeben“* gekennzeichnet und mit dem Verarbeitungsstempel und -datum versehen. Die vorgenommenen Einstellungen sind bei der nächsten Eingangsrechnung wirksam und können nachträglich mithilfe des gleichen Parametrisierungswerkzeuges geändert werden.

Die integrierten Einstellungen-Seiten einzelner Systemmodule ermöglichen dem Anwendungsexperten die initial festgelegten Moduleinstellungen an die Anforderungen des Anwenderunternehmens eigenständig anzupassen. Das cloudbasierte ERP-System von Scopevisio stellt im Gesamtumfang ca. 250 Parameter mit systemübergreifender und modulbezogener Wirkung zur Verfügung, die im Rahmen einer anwenderseitigen Parametrisierung unternehmenskonform eingestellt werden können.

### 5.3.3 System III: Odoo

Odoo stellt unterschiedliche Werkzeuge bereit, mit deren Hilfe differenzierte Systembereiche im Rahmen des On-Demand-Selfservices parametrisiert werden können. Die Parametrisierungswerkzeuge sind in die Benutzeroberfläche integriert und stehen in dem gesamten Lebenszyklus zur Verfügung.

Die allgemeinen unternehmensbezogenen bzw. systemübergreifenden Einstellungen – Land, Branche, Sprachen etc. – werden mithilfe der Administrationswerkzeuge verwaltet, die standardmäßig im Lösungsumfang (im Fenster *Allgemeine Einstellungen*) enthalten sind. Die funktionalen Module von Odoo werden dagegen getrennt parametrisiert.

Jedes der installierten Systemmodule stellt in der Regel den Systemadministratoren eine zusätzliche Konfigurationsseite bereit, auf der die verfügbaren Moduleinstellungen kontextbezogen zusammengestellt werden. Die Parametrisierung erfolgt durch die Aktivierung der einzelnen Optionsfelder, die den gewünschten Moduleinstellungen entsprechen. Anhand der ausgewählten Optionen erfolgt die Zuweisung der Parameterwerte zu den modulbezogenen Parametern im Hintergrund.

Im Rahmen des Beispielszenarios wurde die Parametrisierung der stammdatenbezogenen Einstellungen des Verkaufsmoduls vorgenommen, die in der Abbildung 5.7 visualisiert wird.

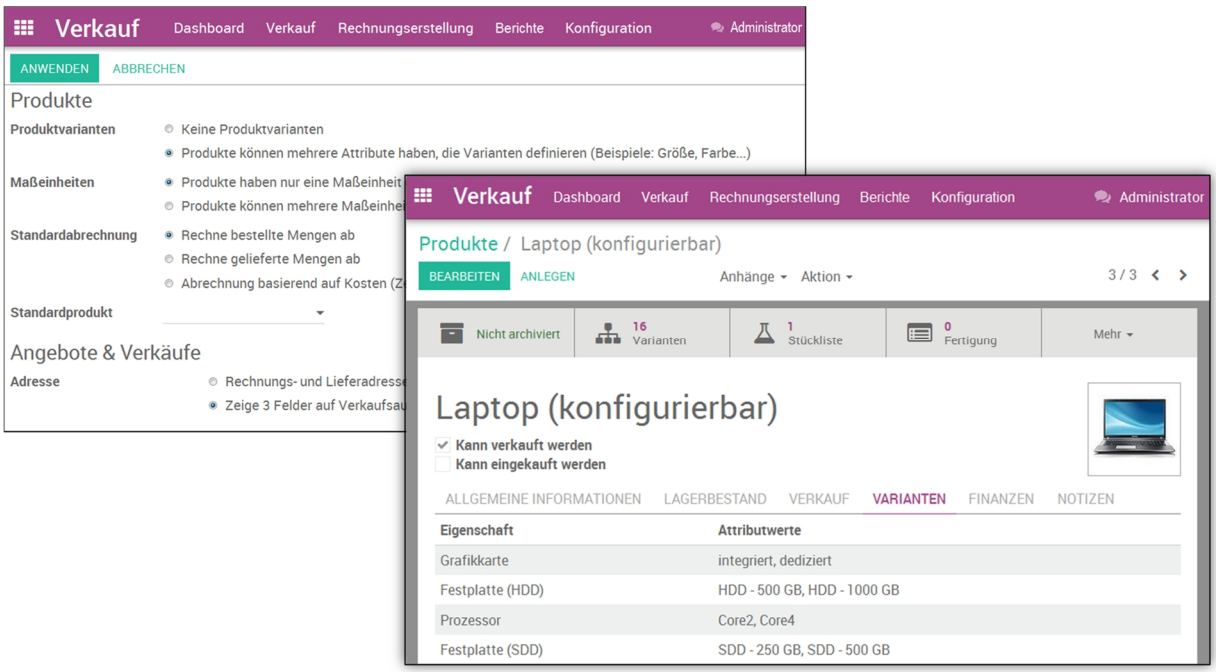





**Abbildung 5.7:** Parametrisierung der Produktstammdaten in Odoo

Bei der Reparametrisierung des Verkaufsmoduls (links) wurde die Produktvarianten-Funktionalität aktiviert, welche die Berücksichtigung unterschiedlicher *Produktvarianten* im Rahmen eines global definierten Produktes ermöglicht. Die *Produktvarianten* stellen ähnliche Produkte dar, die zum großen Teil aus identischen Komponenten bestehen (vgl. Norm DIN 199-1).

Infolge der vorgenommenen Einstellung wird das standardisierte Layout der Produktstammdatenseite mit einem *Varianten-Reiter* erweitert (rechts), der eine Definition einzelner konfigurierbarer Produkteigenschaften und den ihnen zugehörigen Attributwerten ermöglicht. Die aktivierte Funktionalität ermöglicht den Verkauf von konfigurierbaren Produkten, bei denen

der Kunde über einzelne Produkteigenschaften im Rahmen des Verkaufsprozesses selbst entscheiden kann. In dem dargestellten Beispielszenario wurde ein Laptop als konfigurierbares Produkt angelegt, der in Abhängigkeit von den vier vorgegebenen Eigenschaften und den jeweils zwei zur Auswahl stehenden Attributwerten in 16 Varianten angeboten wird.

Die in Odoo eingebauten Parametrisierungswerkzeuge ermöglichen es, die unternehmenskonforme Änderung der initial festgelegten Parametereinstellungen im gesamten Systemlebenszyklus vorzunehmen. Die cloudbasierte Version von Odoo stellt im Standard ca. 800 Parameter mit modul- und systemübergreifender Wirkung bereit.

## 5.4 Adaption der Systemoberfläche

Die Anpassung der Benutzeroberfläche eines ERP-Systems an die unternehmerischen Anforderungen stellt eine Customizing-Aufgabe dar, die einen Bestandteil nahezu jeder Implementierung darstellt. Dabei wird jedoch nicht nur die Anpassung der farblichen Gestaltung an das Corporate Design des Anwenderunternehmens verstanden, sondern auch die Änderung der Komposition der Benutzeroberfläche durch das Hinzufügen weiterer systeminternen bzw. -externen Oberflächenkomponenten.

Im Rahmen dieser Fallstudie soll überprüft werden in welchem Umfang die Systemoberfläche an die Anforderungen des Anwenderunternehmens angepasst werden kann. Dabei wird insbesondere untersucht, ob das Anwenderunternehmen eigenständig die Adaption der einzelnen Oberflächenkomponenten vornehmen kann und welche Anpassungswerkzeuge zwecks Umsetzung visueller Systemanpassungen von dem Systemhersteller bereitgestellt werden.

### 5.4.1 System I: Business ByDesign

Business ByDesign stellt die eingebauten *Adaption- und Personalisierungswerkzeuge* bereit, die über die Benutzeroberfläche zugänglich sind. Diese Werkzeuge ermöglichen dem Anwendungsexperten einige unternehmenskonformen Adaptionen der Benutzeroberfläche vorzunehmen, wie z. B.:

- Einspielen eines neuen an das Corporate-Design angepassten Hintergrundbildes,
- Anpassung der Komposition der einzelnen Systemseiten durch Ein- und Ausblenden bzw. Repositionierung der Oberflächenkomponenten,
- Anpassung der Beschriftungen einzelner Seitenbereiche an die Terminologie des Anwenderunternehmens,
- Definition unternehmensindividueller Felder,
- Anpassung der Zahlen- sowie der Zeit- und Datumsformate,
- Anpassung der Darstellung von Diagrammen und Tabellen (vgl. SAP AG 2016a).

Die Personalisierungswerkzeuge stehen im gesamten Systemlebenszyklus zur Verfügung und können zur Anpassung jeder Systemseite angewendet werden.

Die Abbildung 5.8 stellt ein Beispiel für eine individuell angepasste Systemseite dar. Das im Hintergrund präsentierte Systemfenster zeigt die standardisierte Ansicht des Startbildschirms von Business ByDesign an.

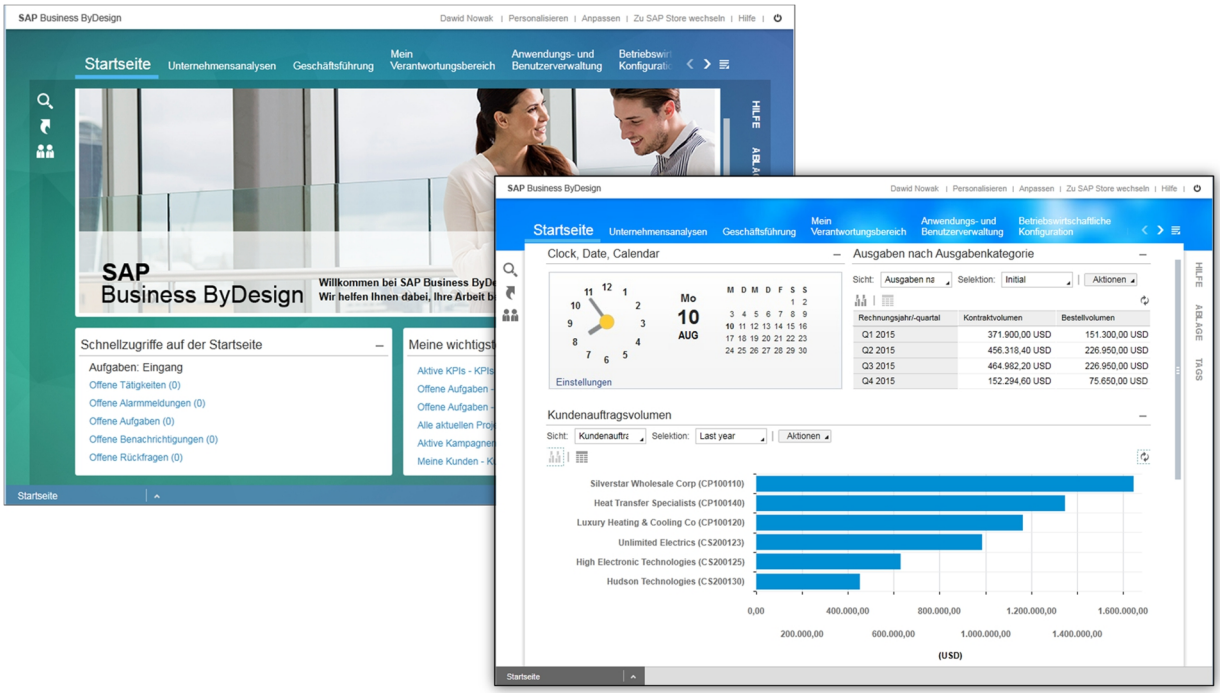





**Abbildung 5.8:** Angepasste Startseite von SAP Business ByDesign

Mithilfe der bereitgestellten Anpassungswerkzeuge wurden die standardisierten visuellen Komponenten der Startseite (oben) ersetzt und die neue farbliche Gestaltung mit einem Hintergrundbild eingespielt. Die neue Komposition der Startseite (unten) besteht aus einer Uhr mit dem Kalender und zwei Berichten. Die beiden Berichte sind mit den Datenquellen verlinkt und werden in regulärer Zeitabständen aktualisiert. Die Uhr mit integrierter Kalenderfunktionalität stellt eine externe Webanwendung dar, die mithilfe der Personalisierungswerkzeuge in die Systemseite eingebettet wurde. Die genauere Beschreibung der verwendeten Integrationsmechanismen erfolgt im Abschnitt 5.9.1. Die neuen Einstellungen der Benutzeroberfläche wurden unternehmensübergreifend vorgenommen und sind für alle angemeldeten Systemanwender wirksam.

Neben den Bildschirmkomponenten, die in Business ByDesign standardmäßig enthalten sind und vom Anwendungsexperten ausgewählt und aktiviert werden können, ermöglicht es der Systemhersteller auch, die unternehmensindividuellen Komponenten der Benutzeroberfläche eigenständig zu entwickeln. Dafür ist die Entwicklungsumgebung – SAP Cloud Applications Studio – vorgesehen, mit deren Hilfe die unternehmensdedizierten Oberflächenkomponenten, Systemseiten und funktionale Systembereiche (sogenannte Work-Center; SAP AG 2016b) entwickelt werden können. Die eigenentwickelten Komponenten der Benutzeroberfläche können nach erfolgreicher Integration in die Basisanwendung mithilfe der Personalisierungswerkzeuge in der ausgewählten Systemseite aktiviert werden.

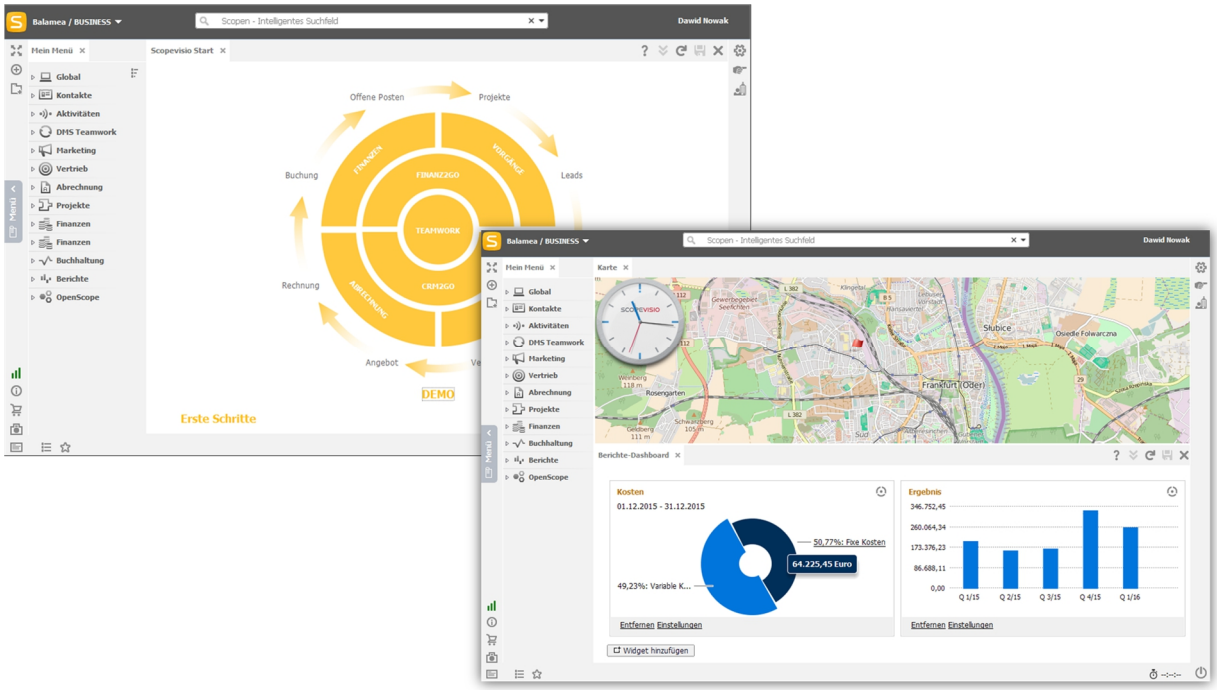
### 5.4.2 System II: Scopevisio

Scopevisio bietet einen integrierten Layout-Editor, der den Anwendungsexperten in die Lage versetzt, die einzelnen Komponenten der Systemoberfläche zu adaptieren. Der in die Systemoberfläche integrierte Editor ermöglicht eine Anpassung der Komposition einzelner Systemseiten durch Ein- und Ausblenden bzw. Repositionierung der standardisierten Oberflächenkomponenten und die Anpassung der Darstellung von predefinierten Diagrammen.

In Scopevisio kann nur die Komposition einiger Systemseiten an die unternehmerischen Anforderungen angepasst werden bzw. können nicht alle Oberflächenelemente den einzelnen Systembereichen zugewiesen werden. In diesem Zusammenhang können zum Beispiel die Diagramme nur in der Seite *Berichte-Dashbord* angelegt werden. Daneben existieren auch Komponenten (z. B. die Uhr), die systemübergreifend in der Oberfläche verankert werden können.

Im Rahmen des Beispielszenarios wurde die standardisierte Startseite des Scopevisio-Systems durch eine neue Systemseite ersetzt, die nachfolgend an die der Fallstudie zugrunde gelegten Anforderungen angepasst wurde.

Die Abbildung 5.9 zeigt die standardisierte Systemseite im Hintergrund und das Ergebnis einer Anpassung, die mithilfe des bereitgestellten Editors vorgenommen wurde.




© Scopevisio AG

**Abbildung 5.9:** Standardisierte und angepasste Systemseite in Scopevisio

Das Endergebnis der vorgenommenen Adaption der Benutzeroberfläche ist im Vordergrund (rechts) dargestellt. Mithilfe des Layout-Editors wurde die standardisierte Systemseite mit einer Landkarte mit der Kundenübersicht, zwei Berichten und einer Uhr angereichert. Alle dargestellten Komponenten stellen standardisierte Bestandteile dar, die in Form eines Widget in der Systemoberfläche verankert werden können. Die Widgets sind standardmäßig im ERP-System enthalten bzw. können vom Systemhersteller hinzugefügt werden. Scopevisio bietet dem Anwendungsexperten keine Möglichkeit an, selbsterstellte Widgets in die Systemoberfläche selbstständig zu integrieren.

### 5.4.3 System III: Odoo

Das cloudbasierte ERP-System von Odoo S. A. stellt den Anwendungsexperten differenzierte Werkzeuge zur Verfügung, mit deren Hilfe die Benutzeroberfläche des cloudbasierten ERP-Systems in unterschiedlichem Umfang an die Anforderungen des Anwenderunternehmens angepasst werden kann.

Im Rahmen einer anwenderseitigen Adaption der Benutzeroberfläche ist der Anwendungsexperte berechtigt, unternehmensindividuelle *Pinnwände* zu erstellen. Eine Pinnwand in Odoo stellt eine frei definierbare Systemseite dar, die die ausgewählten in der Regel unabhängigen Oberflächenkomponenten (z. B. Berichte, Kalender, Landkarten, etc.) wiederverwendet und in Form eines Dashboard zusammenstellt. Die einzelnen Pinnwandkomponenten können im gesamten Bereich der Pinnwand per Drag-and-Drop beliebig angeordnet und ein- bzw. ausgeblendet werden.

Neben den frei definierbaren Pinnwänden können von den Anwendungsexperten auch die Layouts der standardisierten Systemseiten adaptiert bzw. neue Oberflächenkomponenten erstellt werden. Zum Zweck derartiger tiefgreifender Adaption der Benutzeroberfläche stellt der Systemhersteller die *Technischen Werkzeuge* bereit, die im Entwicklermodus anwendbar sind. Diese Werkzeuge ermöglichen es, eine weitreichende Modifikation der Systemoberfläche vorzunehmen, die direkt im HTML/XML-Quellcode erfolgen muss. Mithilfe der integrierten Werkzeuge können zum Beispiel neue Systemseiten, Oberflächenkomponenten und Erweiterungsfelder entwickelt werden, aber auch externe Webinhalte in den standardisierten Systemseiten verankert werden.

Die Abbildung 5.10 stellt die benutzerdefinierte Systemseite in Odoo dar, die im Rahmen dieser Machbarkeitsstudie angelegt wurde.

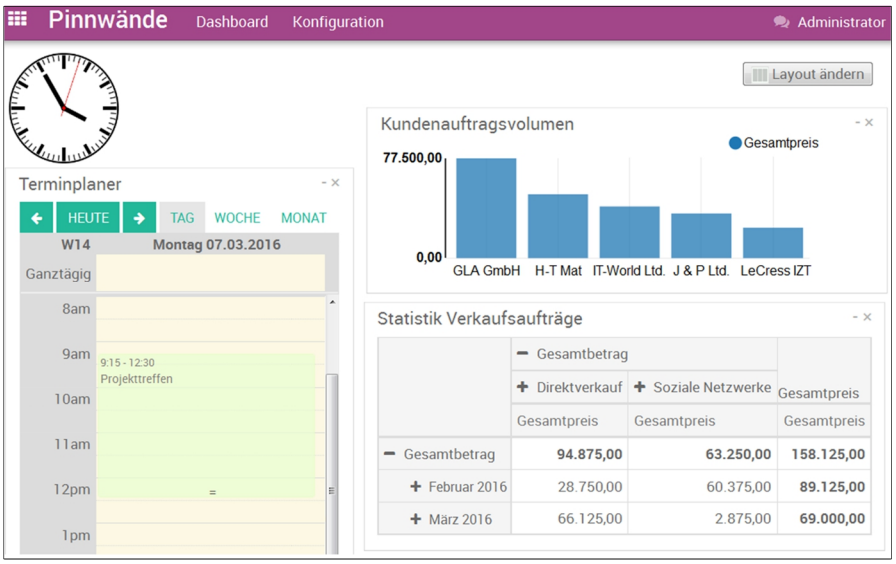





**Abbildung 5.10:** Angepasste Übersichtsseite in Odoo

Mithilfe der integrierten Werkzeuge wurde eine neue Übersichtsseite (Dashboard) erstellt, die mit zwei Berichten, einem Terminpläner und einer Uhr vervollständigt wurde. Die Berichte und Terminplaner stellen systeminterne Komponenten dar, die im Standard enthalten sind und standardmäßig zu einem Dashboard hinzugefügt werden können. Die weiteren Elemente stellen externe Webinhalte dar, die mithilfe des integrierten View-Editors in der Übersichtsseite als *iFrames* verankert wurden. Ein *iFrame* ist eine HTML-Komponente, die auf einen externen Webinhalt verweist und erlaubt, ihn als eine Komponente anzubinden (vgl. W3C 2017a).

Die integrierten Werkzeuge stehen dem Anwendungsexperten im ganzen Systemlebenszyklus zur Verfügung und ermöglichen es, die visuellen Aspekte von Odoo an die unternehmerischen Anforderungen eigenständig anzupassen.

## 5.5 Erweiterung der Geschäftsobjekte und Dokumentenvorlagen

Die Analyse der häufigen Anpassungsanforderungen der ERP-Standardanwendungssoftware weist auf die Bedeutung der Adaption der Geschäftsobjekte bzw. der Datenmodelle sowie auf die Wichtigkeit der Sicherstellung der erforderlichen Flexibilität der unternehmerischen Dokumentenvorlagen hin. Die Validierung der Adaptierbarkeit dieser Systemelemente im Rahmen der untersuchten Cloud-ERP-Systeme stellt den zentralen Beitrag dieses Abschnitts dar.

Aufbauend auf dem steigenden Interesse des Vertriebs von Produkten und Dienstleistungen über soziale Netzwerke setzt diese Fallstudie die Verwendung der Sozialkanäle als alternative Bestellwege voraus. Dies bedeutet, dass die Kundenaufträge neben den klassischen Bestellkanälen (z. B. E-Mail, Fax, Telefonat etc.) auch über die durch die sozialen Plattformen (z. B. Facebook, Twitter, Google+) bereitgestellten Nachrichtendienste eingereicht werden können.

Die Information bzgl. des Bestellkanals soll im ERP-System entsprechend gesichert werden, um weitere Analysen in Bezug auf das Kundenverhalten zu ermöglichen. In diesem Zusammenhang muss das Layout des Kundenauftragsformulars um das benutzerdefinierte Feld – *Bestellkanal* – erweitert werden. Das Feld soll dem Systemanwender die Möglichkeit geben, einen der drei vordefinierten Kanäle (Facebook, Twitter, Google+) aus einer Dropdownliste auszuwählen. Der Kanal der Kundenauftragserteilung soll mit anderen Auftragsdaten erfasst und gesichert werden, um zu einem späteren Zeitpunkt eine gezielte Datenauswertung zu ermöglichen (siehe Abschnitt 5.6). Darüber hinaus soll aus Gründen der besseren Übersichtlichkeit die Information über den Bestellkanal auf dem Kundenauftragsbeleg angezeigt werden.

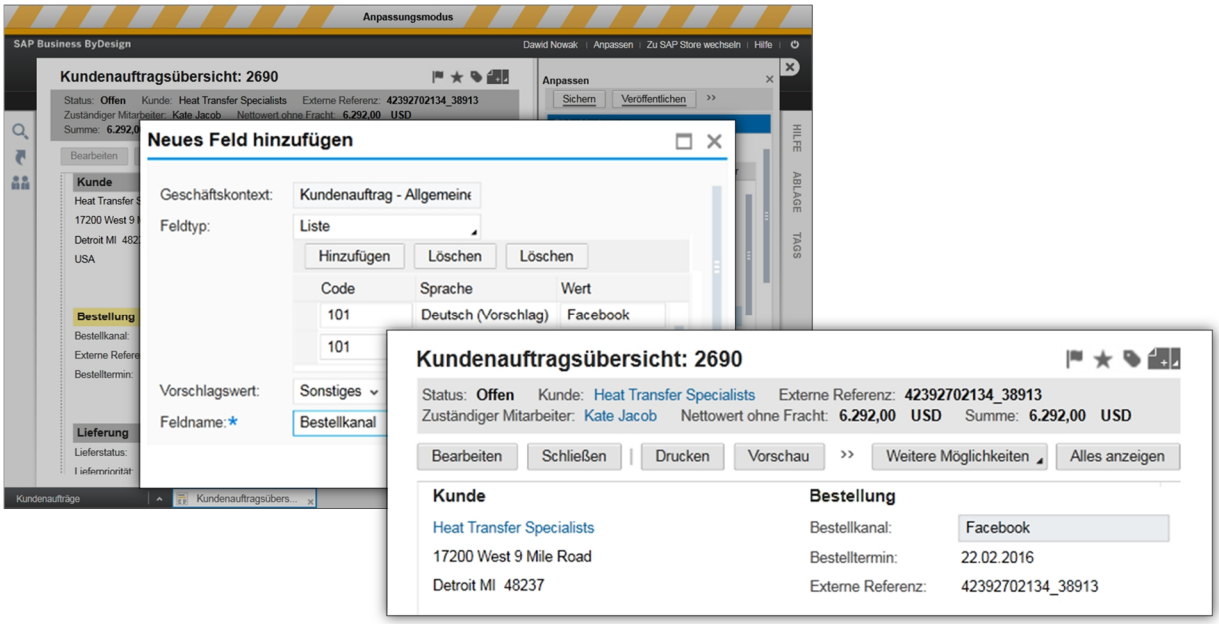
Im Rahmen dieses Szenarios wird geprüft, ob das Anwenderunternehmen eine Möglichkeit hat, im Systemrahmen die standardisierten Geschäftsobjekte mit neuen Attributen eigenständig zu erweitern. Neben der Adaptierbarkeit der Datenebene wird in diesem Kontext auch die Adaptierbarkeit der Systemoberfläche und Dokumentenvorlagen evaluiert, die das erweiterte Geschäftsobjekt referenzieren.

### 5.5.1 System I: Business ByDesign

Das cloudbasierte ERP-System von SAP stellt dem Anwendungsexperten die integrierten Adaptionswerkzeuge zur Verfügung, die ihn in die Lage versetzen, mehrere unternehmensdedizierte Eingabefelder systemübergreifend zu definieren und diese unterschiedlichen Geschäftsszenarien zuzuordnen. Die neu definierten Felder können mittels des eingebauten Layout-Editors in dem ausgewählten Bereich der Systemseite verankert werden bzw. mithilfe des integrierten Vorlageassistenten die unternehmerischen Dokumentenvorlagen erweitern.

Die Erweiterung der Systemseite mithilfe des in Business ByDesign integrierten Layout-Editors wurde in der Abbildung 5.11 veranschaulicht.

Das Hinzufügen eines neuen benutzerdefinierten Feldes erfordert unter anderem die Auswahl eines Feldtypes (z. B. Zahl, Text, Datum, Liste, Berechnungsfeld, etc.) und die Vergabe eines Feldnamens. Bei den Dropdownlisten müssen zunächst die Listeneinträge gepflegt und der Vorschlagswert definiert werden, der bei fehlenden Benutzerangaben standardmäßig ausgewählt wird. Technisch gesehen erweitert das angelegte Feld neben dem Metamodell der Systemseite auch die dem Geschäftsobjekt zugrunde liegende Tabelle um eine neue Attribut-Spalte. Die benutzerdefinierten Felder werden in Business ByDesign bezogen auf den konkreten Geschäftskontext angelegt und können auf allen Systemseiten bzw. Dokumentenvorlagen etc. referenziert werden, die im gleichen Kontext verwendet werden. In Bezug auf die Fallstudie bedeutet das, dass die im Kundenauftrag-Geschäftskontext definierten Felder auf allen Systemseiten etc. referenziert werden können, welche die Kundenauftrag-Stammdaten wiederverwenden.




© SAP SE

**Abbildung 5.11:** Konfiguration und Verwendung benutzerdefinierter Felder in Business ByDesign

In dem dargestellten Szenario wurde die Kundenauftragsübersichtsseite mithilfe des Layout-Editors um das Listen-Feld *Bestellkanal* erweitert. Als zulässige Eingabewerte wurden die Namen der drei sozialen Kanäle – *Facebook*, *Google+* und *Twitter* – vordefiniert, die während der Erfassung der Bestelldaten als Auswahlmöglichkeiten gelten. Daneben wurden im *Bestellung-Bereich* der angezeigten Auftragsübersicht zwei weitere Eingabefelder – *Externe Referenz* und *Bestelltermin* – hinzugefügt. Das Feld *Externe Referenz* beinhaltet die kanalbezogene Gesprächsnummer, die auf das konkrete Kundengespräch verweist. Das *Bestelltermin-Feld* enthält das Datum der Bestellung. Die Inhalte der benutzerdefinierten Felder werden mit allen anderen auftragsbezogenen Daten gesichert und können zu einem späteren Zeitpunkt auf der Kundenauftragsübersicht wiedergefunden werden.

Die benutzerdefinierten Felder erweitern das Geschäftsobjekt und können neben den Systemseiten auch auf den Dokumentenvorlagen referenziert werden. Die Erweiterung der unternehmerischen Dokumentenvorlagen um die benutzerdefinierten Datenfelder kann in Business ByDesign mittels des integrierten Vorlage-Editors vorgenommen werden. Der eingebaute Vorlage-Editor ermöglicht es, die benutzerdefinierten Felder in dem ausgewählten Vorlagebereich zu verankern und zu aktivieren. Die Neugestaltung der unternehmerischen Dokumentenvorlagen ist allerdings auch in Business ByDesign möglich. Zu diesem Zweck stellt der Systemhersteller eine Anwendung – Adobe LiveCycle Designer (Adobe Systems 2017) – bereit, die heruntergeladen und lokal installiert werden muss (vgl. Hufgard und Krüger 2012, S. 206 f.). Sowohl im Fall des integrierten als auch des lokal installierbaren Vorlage-Editors werden die benutzerdefinierten Vorlagen in Form der Metamodelle gespeichert, die im ganzen Systemlebenszyklus gemäß den Anforderungen des Anwenderunternehmens adaptiert werden können.

### 5.5.2 System II: Scopevisio

Das cloudbasierte ERP-System von Scopevisio stellt dem Anwenderunternehmen das integrierte Werkzeug zur Verfügung, das es ermöglicht, bis zu zwanzig benutzerdefinierte Felder im Rahmen eines On-Demand-Selfservices anzulegen. Die benutzerdefinierten Felder erweitern das zugrunde liegende Geschäftsobjekt und können von dem Anwendungsexperten kon-

textabhängig auf dem jeweiligen Formular bzw. der Dokumentenvorlage aktiviert werden, welche dieses Geschäftsobjekt referenzieren.

Scopevisio stellt allerdings keinen im System eingebauten Vorlage-Editor bereit. Es ermöglicht stattdessen aber, die Dokumentenvorlagen mit nahezu jedem externen Textverarbeitungsprogramm (z. B. Microsoft Office, OpenOffice) zu editieren bzw. zu erstellen. Eine lokal erstellte Vorlage kann nach dem erfolgreichen Hochladen in das ERP-System zur Erzeugung der elektronischen Dokumente verwendet werden.

Die Abbildung 5.12 stellt die Konfiguration des benutzerdefinierten Feldes (links) und die um das benutzerdefinierte Feld erweiterte Kundenauftrag-Vorlage in der Entwurfsansicht (rechts) dar.

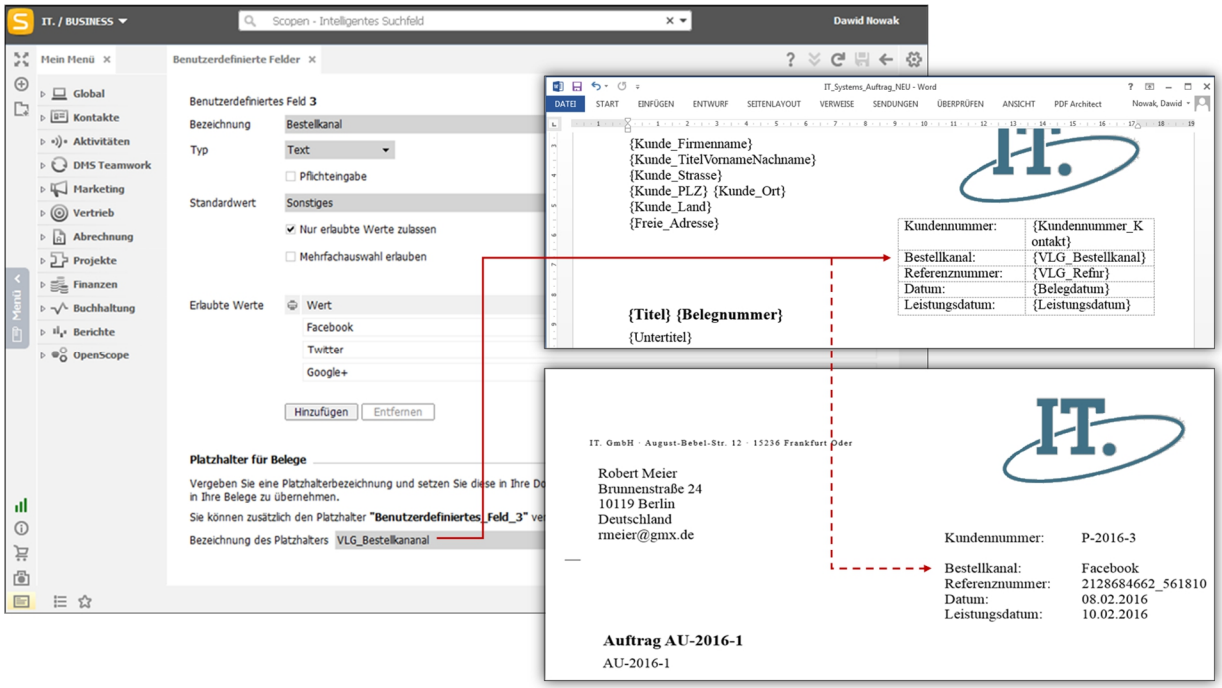





**Abbildung 5.12:** Benutzerdefiniertes Listenfeld in Scopevisio im Feld- und Vorlage-Editor

Beim Anlegen eines neuen benutzerdefinierten Feldes werden seine Bezeichnung und sein Datentyp eindeutig definiert. Wird das Kontrollfeld – *Nur erlaubte Werte zulassen* – selektiert, kann eine Liste zulässiger Auswahloptionen vorgegeben werden. Die als *Standardwert* festgelegte Option wird bei fehlenden Benutzerangaben standardmäßig durch das System zugewiesen. Die vorgenommene Erweiterung des Geschäftsobjektes wird als eine neue Spalte in der dem Geschäftsobjekt zugrunde liegenden Tabelle angelegt.

Im Rahmen des Beispielszenarios wurde das benutzerdefinierte Eingabefeld – *Bestellkanal* – als eine Liste mit drei alternativen Auswahloptionen – *Facebook*, *Twitter* und *Google+* – angelegt (links). Die in unterem Bereich des Konfigurationsformulars definierte Platzhalterbezeichnung (*VLG_Bestellkanal*; siehe Abbildung 5.12, links) erlaubt es, das neue Feld in den Dokumentenvorlagen zu referenzieren.

Mithilfe von Microsoft Word wurde eine benutzerdefinierte Vorlagedatei für die Kundenauftragsbestätigung erstellt (rechts), in deren rechten Bereich eine Referenz zum definierten *Bestellkanal-Feld* erstellt wurde. Die beispielhafte Ausgabedatei, die mithilfe der neuen Vor-

lage erzeugt wurde, ist im Vordergrund der Abbildung dargestellt. Der Pfeil illustriert, wie bei Druckausgabe des Dokumentes anhand der angelegten Vorlage die referenzierte Information bezüglich des Bestellkanals abgefragt und anstelle des Platzhalters ausgewiesen wird.

Scopevisio stellt eine angemessene Erweiterbarkeit der standardisierten Geschäftsobjekte in dem gesamten Systemlebenszyklus sicher. Im Fall umfangreicher und komplexer Adaptionen des Datenmodells, die über die festgesetzten Rahmen einer kundenseitigen Erweiterung hinausgehen, bietet der Systemhersteller im Rahmen eines Supports die Möglichkeit einer dedizierten Adaption der Geschäftsobjekte an die Anforderungen des Kundenunternehmens an.

### 5.5.3 System III: Odoo

Odoo stellt eine weitreichende Flexibilität der Datenebene durch die getrennten und mandantendedizierten Datenbankschemata sicher. Das cloudbasierte ERP-System stellt dem Anwendungsexperten die integrierten *Technischen Adaptionswerkzeuge* zur Verfügung, die unternehmensindividuelle Anpassbarkeit bzw. Erweiterbarkeit der bestehenden Datenstruktur sicherstellen. Neben den benutzerdefinierten Erweiterungsfeldern können von dem Anwendungsexperten mithilfe der bereitgestellten Werkzeuge individuelle Datentabellen angelegt und Tabellenbeziehungen definiert werden.

Die unternehmensdidizierten Erweiterungsfelder in Odoo werden in Bezug auf ein konkretes Geschäftsobjekt angelegt und können in Abhängigkeit von dem Geschäftskontext auf den ausgewählten Systemseiten hinzugefügt und aktiviert werden. Die Verankerung der benutzerdefinierten Eingabefelder in der Benutzeroberfläche kann mithilfe des eingebauten View-Editors vorgenommen werden, der dem Anwendungsexperten im Entwicklermodus zur Verfügung steht.

Im Rahmen des Beispielszenarios wurden zwei benutzerdefinierte Felder – *Bestellkanal* und *Externe Referenz* – angelegt, die auf dem Kundenauftragsformular und der Vorlage der Kundenauftragsbestätigung referenziert wurden.

Die Abbildung 5.13 stellt die Definition des benutzerdefinierten *Bestellkanal-Feldes* (links) und die erweiterte Verkaufsauftragsübersichtsseite (rechts) in der Benutzeransicht dar.

Der Definitionsprozess erfolgt im Feld-Editor und erfordert die Vergabe eines eindeutigen Feldnamens, die Auswahl des Modells (der Geschäftsobjekt-Tabelle) sowie die Definition der Feldbezeichnung und des Feldtypes. Beim Anlegen eines Selektionsfeldes werden im nächsten Schritt im Bereich *Grundeinstellungen* die Auswahloptionen definiert. Die Definition der zulässigen Auswahloptionen erfolgt durch Erstellung einer Werteliste in Python-Sprache, die entsprechend die Positionsnummer und Bezeichnung enthält. Nachdem das neue Feld gesichert wurde, wird die dem Geschäftsobjekt zugrunde liegende Tabelle um eine neue Attribut-Spalte erweitert.

Das neu definierte Feld kann danach in der Verkaufsauftragsübersichtsseite verankert werden. Dies erfolgt durch die Anpassung des HTML/XML-basierten Seiten-Quellcodes mithilfe des integrierten View-Editors (vgl. Odoo S. A. 2016c). Sobald das neue Feld im Quellcode der Systemseite eingebettet wird, ist es auch in der Benutzeransicht verfügbar.

Das Ergebnis der durchgeführten Erweiterung des Eingabeformulars des Verkaufsauftrags ist im Vordergrund der Abbildung 5.13 zu erkennen. Die erweiterte Ansicht ermöglicht es bei der Erfassung der Auftragsdaten, den Bestellkanal und die eindeutige Gesprächsnummer, welche den konkreten Nachrichtenaustausch im Rahmen eines Sozialkanals referenziert, im Feld *Externe Referenz* abzusichern.

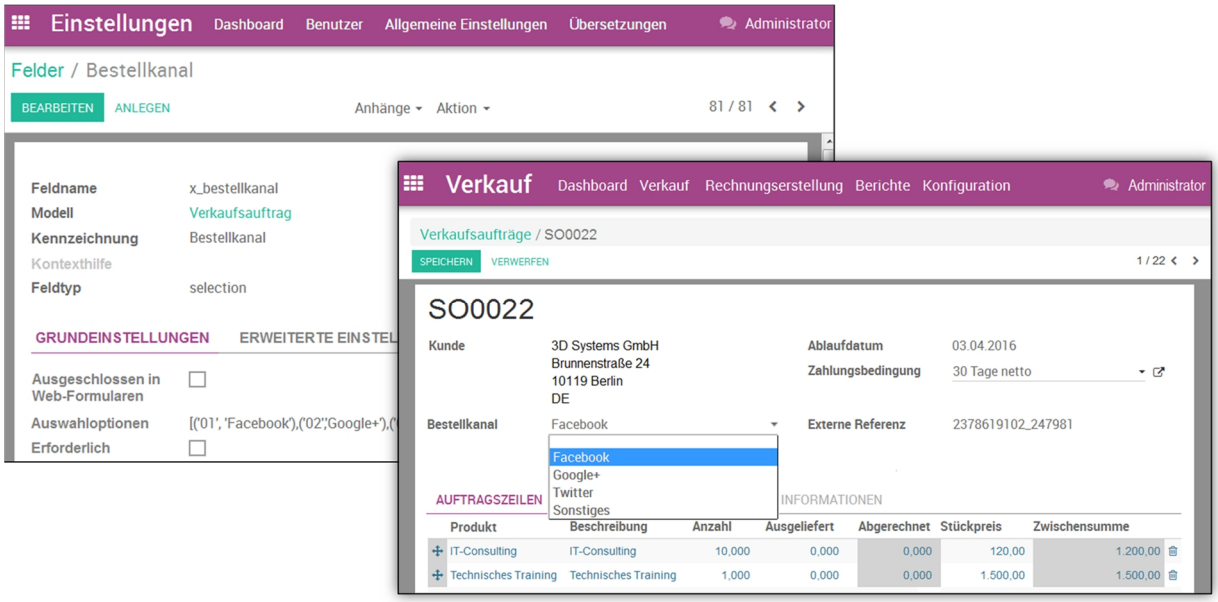





**Abbildung 5.13:** Konfiguration und Verwendung eines benutzerdefinierten Listenfelds in Odoo

Die kundenauftragsbezogenen Informationen können auch auf den Dokumentenvorlagen referenziert werden. Der Anwendungsexperte kann dank dem in Odoo integrierten Vorlage-Editor die Dokumentenvorlagen erweitern bzw. modifizieren. Die Adaption der Vorlagen erfolgte in der proprietären Auszeichnungssprache, die die standardisierte XML-Semantik um die Kontrollanweisungen (u. a. Schleifen, bedingte Anweisungen etc.) erweitert (vgl. Odoo S. A. 2016d). Beim Ausdrucken der Kundenauftragsübersicht werden die einzelnen benutzerdefinierten Felder – *Bestellkanal* und *Externe Referenz* – abgefragt und ihr Inhalt auf dem Ausdruck ausgewiesen.

Die vorgenommene Machbarkeitsstudie weist auf die Flexibilität des standardisierten Datenmodells bzw. der Geschäftsobjekte von Odoo hin und bestätigt die anwenderseitige Adaptierbarkeit der Systemseiten und der Dokumentenvorlagen, die durch die integrierten Werkzeuge sichergestellt wird.

## 5.6 Erweiterung der Geschäftsanalytik

Im Rahmen dieses Abschnitts soll die Flexibilität der cloudbasierten ERP-Systeme im Bereich der Geschäftsanalytik untersucht werden. Mithilfe der konzipierten Fallstudie soll die Frage beantwortet werden, ob die unternehmensinternen Systemadministratoren in die Lage versetzt werden, die analytischen Komponenten – insbesondere Kennzahlen und Berichte – zwecks einer gezielten Analyse der Betriebsdaten selbstständig zu definieren.

Das Leitmotiv des vorigen Szenarios fortsetzend, soll überprüft werden, ob die benutzerdefinierbaren Erweiterungsfelder im analytischen Kontext wiederverwendet werden können. Insbesondere wird im Rahmen der Machbarkeitsstudie geprüft, ob die unternehmensindividuellen Erweiterungen der standardisierten Geschäftsobjekte in den Datenquellen, Berichten und Kennzahlen referenziert werden können. Diese Fallstudie setzt die Entwicklung einer neuen Kennzahl und eines auf ihr basierenden Berichts voraus, die eine vertiefende Analyse der Verkaufsaktivitäten ermöglichen.

Aufbauend auf dem um das Bestellkanal-Feld erweiterten Geschäftsobjekt soll in jedem der untersuchten ERP-Systeme eine benutzerdefinierte Kennzahl erstellt werden, die eine genauere Analyse der Bedeutung sozialer Bestellwege für den Produktabsatz ermöglicht. Die Kennzahl – *Verkaufsrate pro Bestellkanal* – wird als prozentualer Anteil des Nettowertes aller ver-

kauften Produkte, die über den jeweiligen Bestellkanal bestellt wurden, an dem Gesamtwert aller Verkäufe berechnet. Der benutzerdefinierte Bericht soll die zeitbezogene Entwicklung der Verkaufsraten visualisieren.

### 5.6.1 System I: Business ByDesign

Im Rahmen des cloudbasierten ERP-Systems von SAP werden dem Anwenderunternehmen die integrierten Werkzeuge bereitgestellt, die eine eigenständige Adaption der bestehenden bzw. die Definition der neuen Analyseinstrumente ermöglichen. Die Erstellung der unternehmensdedizierten Kennzahlen und Berichte erfolgt in Form eines iterativen Prozesses. Der Anwendungsexperte wird von einem integrierten Assistenten durch die einzelnen Prozessschritte geführt.

Der Ablauf des Erstellungsprozesses kann in drei Hauptschritte unterteilt werden, in denen sukzessiv die Datenquelle, die Kennzahl und der Bericht definiert werden. Der Prozess wird durch die Auswahl einer Datenquelle initiiert, welche die dem Bericht zugrunde gelegten Daten beinhaltet. Bei diesem Schritt können auch nach Bedarf mehrere Datenquellen miteinander verknüpft werden. Anschließend werden die Kennzahlen ausgewählt bzw. neue Kennzahlen angelegt, welche die Faktenspalten der ausgewählten Datenquelle referenzieren. Im letzten Schritt erfolgt der Berichtsentwurf durch die Zuweisung der Kennzahlen und Merkmale zu den Spalten und Zeilen des Diagramms.

Die Schritte der Kennzahl-Definition und des Berichtsentwurfs mithilfe der in Business ByDesign integrierten Editoren wurden in der Abbildung 5.14 veranschaulicht.

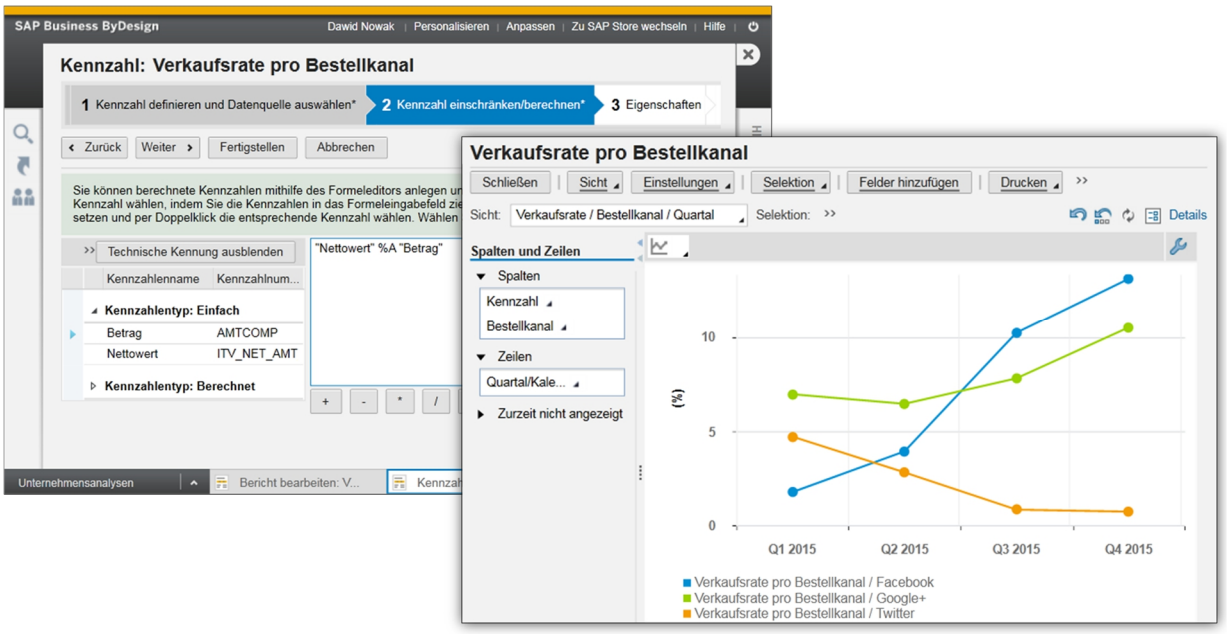





**Abbildung 5.14:** Benutzerdefinierte Analyseinstrumente in Business ByDesign

SAP Business ByDesign bietet dem Anwendungsexperten die Möglichkeit, die benutzerdefinierten Erweiterungsfelder auch im analytischen Kontext wiederzuverwenden. Aufbauend auf der vorigen Fallstudie wurde im Rahmen dieses Beispielszenarios die um die Bestellkanal-Spalte erweiterte Kundenauftrag-Tabelle als Datenquelle für die Definition der neuen Kennzahl verwendet.

Die Kennzahl wurde mithilfe des integrierten Kennzahleneditors (links) als eine sogenannte *berechnete Kennzahl* angelegt, indem die zwei Faktenspalten der referenzierten Datenquelle – *Nettowert* und *Betrag* – ausgewählt und die Kennzahl-Berechnungsformel im Editorbereich definiert wurden. Anhand der angelegten Formel berechnet das System den prozentualen Anteil (*%A*) der bestellkanalbezogenen Nettoverkäufe (*Nettowert*) am Gesamtwert aller Verkäufe (*Betrag*). Die benutzerdefinierte Kennzahl kann bei Definition der Berichte wiederverwendet werden, die die ihr zugrunde gelegte Datenquelle referenzieren.

Darauf aufbauend wurde mithilfe des integrierten Berichtseditors (rechts) ein benutzerdefinierter Bericht in Form eines Liniendiagramms erstellt, das die quartalsbezogene Entwicklung der Verkaufsraten veranschaulicht. Die Definition eines Berichts erfolgt im dargestellten Berichtseditor durch manuelle Zuweisung der einzelnen Kennzahlen und Merkmale, die in der Datenquelle vorhanden sind, entsprechend zu den Spalten und Zeilen des Diagramms. Zum Zweck der Visualisierung der quartalsbezogenen Entwicklung der Verkaufsraten, wird die *Kennzahl* der vertikalen (*Spalte*) und das *Quartal-Merkmal* der horizontalen (*Zeile*) Achse zugeordnet. Zum Zweck der bestellkanalbezogenen Differenzierung der Entwicklung der Verkaufsraten erweitert das Merkmal *Bestellkanal* die Spaltendimension.

Das Ergebnis der durchgeführten Machbarkeitsstudie ist im Vordergrund der Abbildung 5.14 veranschaulicht. Das benutzerdefinierte Liniendiagramm stellt eine quartalsbezogene Entwicklung der bestellkanalbezogenen Verkaufsraten wieder, die anhand der Verkaufsdaten des Jahres 2015 berechnet wurden. Die Darstellung der Daten kann auch nachträglich von dem Systembenutzer durch Auswahl des anderen Diagrammtyps bzw. einer tabellarischen Ansicht in der Benutzeransicht des Berichts geändert werden.

Die Adaptionen der Geschäftsanalytik und des Berichtswesens können auch mithilfe der Entwicklungsumgebung vorgenommen werden (vgl. Schneider 2012, S. 336). Die IDE gewinnt an Bedeutung, wenn kontextintensive und komplexe Geschäftslogik bei der Berechnung der zu implementierenden Kennzahlen benötigt wird, die im Rahmen des im System vorhandenen Editors nicht umsetzbar ist.

### 5.6.2 System II: Scopevisio

Scopevisio ermöglicht die im System vorkonfigurierten Analyseinstrumente anzuwenden, stellt jedoch keine Werkzeuge zur anwenderseitigen Definition unternehmensindividuellen Kennzahlen und Berichte zur Verfügung.

Die predefinierten Berichte können in Form eines konfigurierbaren Widgets in der Berichte-Systemseite verankert werden. Der in Scopevisio bereitgestellte Berichtskonfigurator ermöglicht es dem Anwendungsexperten nach der Auswahl des Diagrammtyps (u. a. Kreis-, Linien-, Balkendiagramm), ca. achtzehn vordefinierte und kontextbezogen gruppierte Berichte auszuwählen und zu aktivieren. Die benutzerdefinierte Anpassung des Berichts ist auf die Definition seiner Bezeichnung und die Auswahl des Farbschemas begrenzt. Darauf aufbauend kann in Abhängigkeit von dem ausgewählten Berichtstyp in der Regel noch die Kennzahl (sogenannte Berichtsposition) ausgewählt sowie der Zeitraum ihrer Entwicklung festgelegt werden.

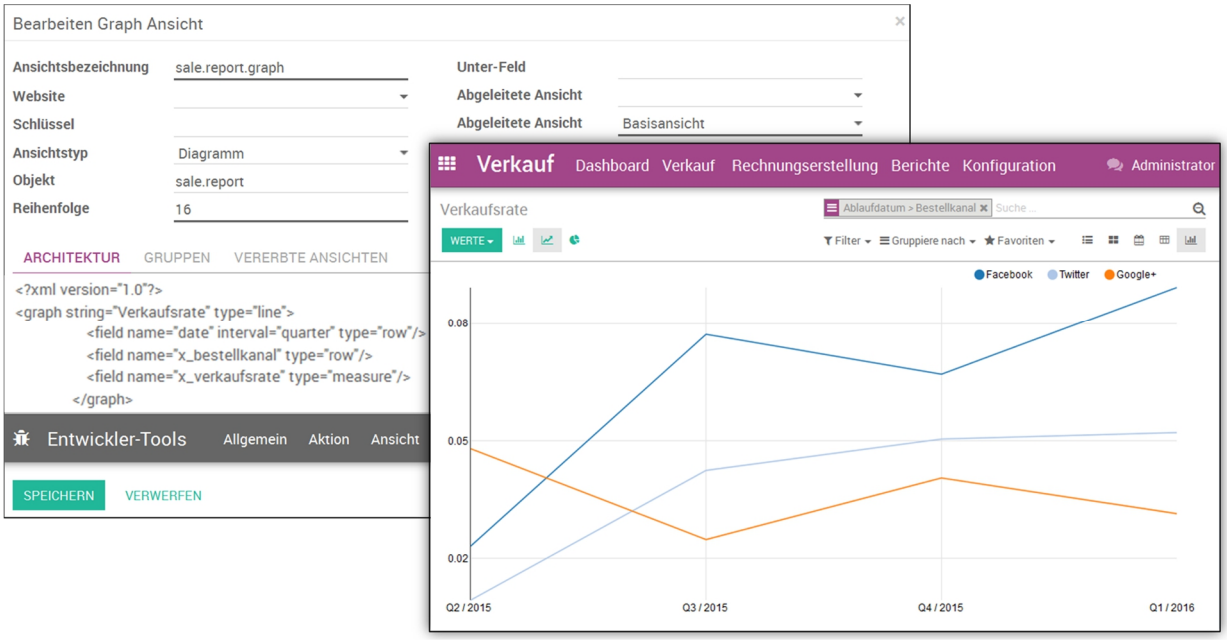
Die anwenderseitige Adaption der bestehenden bzw. die Erstellung von neuen benutzerdefinierten Berichten und Kennzahlen ist im Systemrahmen nicht vorgesehen. Aufgrund dessen, dass die benutzerdefinierten Erweiterungsfelder im analytischen Kontext nicht wiederverwendet werden können, war die Umsetzung des geplanten Szenarios im Systemrahmen nicht möglich.

### 5.6.3 System III: Odoo

Odoo stellt dem Anwenderunternehmen mehrere standardisierte Berichte zur Verfügung, deren Gestaltung mithilfe des im Anwendermodus verfügbaren Berichtskonfigurators individuell angepasst werden kann. Darüber hinaus ermöglicht Odoo einen Entwurf der unternehmensindividuellen Berichte, die die benutzerdefinierten Felder als Merkmale bzw. Kennzahlen wiederverwenden. Die Erweiterung der standardisierten bzw. Erstellung der neuen Analyseinstrumente erfolgt im Entwicklermodus mithilfe des integrierten Editors, die dem Anwendungsexperten als die sogenannten technischen Werkzeuge zur Verfügung stehen.

Jedem Bericht liegt ein kontextbezogenes *analytisches Modell* zugrunde, das in der Regel modulbezogen angelegt ist. Das analytische Modell stellt eine Datenquelle dar, die durch mehrere Berichte referenziert werden kann. Die Erstellung einer neuen Kennzahl bzw. Charakteristik ist mit der Erweiterung des analytischen Modells verbunden. Die Definition einer Kennzahl erfolgt mithilfe des integrierten Feld-Editors und ähnelt der Erweiterung eines Geschäftsobjektes mit dem Unterschied, dass zusätzlich zu dem neuen Feld seine Berechnungsprozedur in Python-Sprache hinterlegt werden muss.

Im Rahmen der Machbarkeitsstudie wurde das standardisierte analytische Verkaufsstatistik-Modell (*sale.report*) um die benutzerdefinierte, berechnete Kennzahl – *Verkaufsrate* – erweitert, die als prozentualer Anteil der bestellkanalbezogenen Nettoverkäufe am Gesamtwert aller Verkäufe berechnet wird. Aufbauend auf der erweiterten Datenquelle wurde ein neuer Bericht erstellt, welcher die zeitliche Entwicklung der Verkaufsraten in Form eines Liniendiagramms darstellt.




© Odoo S. A.

**Abbildung 5.15:** Anlegen eines benutzerdefinierten Berichts in Odoo

Der benutzerdefinierte XML-Quellcode definiert die Diagrammansicht in Form eines Liniendiagramms (*type="line"*) und weist entsprechend der Zeilen-Dimension die Merkmale Bestellkanal (*x_bestellkanal*) und Datum (*date*), und der Fakten-Dimension die Verkaufsrate-Kennzahl (*x_verkaufsrate*) zu. Anhand des im Berichtseditor hinterlegten XML-Quellcodes wird in der Benutzeransicht das benutzerdefinierte Diagramm erzeugt. Der im Rahmen der Machbarkeitsstudie erzeugte Bericht ist im Vordergrund der Abbildung 5.15 dargestellt.

Die Definition des Berichts erfolgt mithilfe des integrierten Berichtseditors, der im Hintergrund der Abbildung 5.15 dargestellt ist. Der Definitionsvorgang erfordert unter anderem die Vergabe einer eindeutigen Berichtsbezeichnung, die Auswahl des Ansichtstypes und der zu referenzierenden Datenquelle. Anschließend muss die Diagrammansicht im XML-Quellcode definiert werden, indem der Diagrammtyp ausgewählt und die einzelnen Merkmale und Kennzahlen der Zeilen- und Fakten-Dimension zugewiesen werden.

Das präsentierte Liniendiagramm gibt die quartalsbezogene Entwicklung der bestellkanalbezogenen Verkaufsraten wieder, die anhand der Verkaufsdaten der letzten zwölf Monate berechnet wurden. Die Berichtsansicht kann vom Endbenutzer im Anwendermodus individuell durch die Auswahl eines anderen Diagrammtyps (z. B. Säule, Kreis) bzw. das Wechseln der Ansichtsform (Tabelle, Diagramm) angepasst werden.

Odoo ermöglicht es, die Adaption bzw. Definition der unternehmensdedizierten analytischen Komponenten im Rahmen des On-Demand-Selfservices vorzunehmen. Die eigenständige Entwicklung von neuen Kennzahlen und Diagrammen erfordert jedoch vom Anwendungsexperten entsprechende Programmierkenntnisse und detailliertes Systemwissen. Alternativ stellt Odoo S. A. zusätzliche Entwicklungspakete bereit, in deren Rahmen die einzelnen Adaptionen der Analyseinstrumente von einem Entwicklerteam übernommen werden.

## 5.7 Adaption der Aufbauorganisation

Im Hinblick auf eine korrekte Regelung der Zuständigkeiten für die arbeitsteilige Erfüllung der Unternehmensaufgaben soll die organisatorische Struktur des Anwenderunternehmens im ERP-System stets akkurat abgebildet werden. Die Aktualisierung der unternehmerischen Organisationsstruktur im Systemrahmen stellt eine der wichtigen Adaptionsanforderungen der Postimplementierungsphase dar (vgl. Kapitel 2.4.1). Durch die direkten Interdependenzen zwischen der Aufbau- und Ablauforganisation müssen deren Änderungen meist zusammen erfolgen (vgl. Abbildung 2.3).

Im Rahmen dieser Machbarkeitsstudie soll die im ERP-System abgebildete Organisationsstruktur des jeweiligen Musterunternehmens um eine neue Niederlassung in Frankfurt (Oder) erweitert werden. Die Fallstudie setzt neben der Übernahme der neuen Organisationseinheit in die bestehende Organisationsstruktur eine Zuweisung eines Filialleiters voraus, der die Leitung der neuen Organisationseinheit übernimmt.

In diesem Szenario soll die systembezogene Flexibilität in Bezug auf die Änderungen der Aufbauorganisation des Musterunternehmens überprüft werden. Insbesondere soll untersucht werden, ob das Anwenderunternehmen eigenständig die Adaption der bestehenden Unternehmensstruktur vornehmen kann und welche Werkzeuge zwecks Umsetzung organisatorischer Änderungen von dem Systemhersteller bereitgestellt werden.

### 5.7.1 System I: Business ByDesign

Im Rahmen der Systemimplementierungsphase von Business ByDesign muss die initiale Aufbauorganisation angelegt werden, die mit den ausgewählten Kerngeschäftsbereichen des zu implementierenden Lösungsumfangs übereinstimmt. Die initiale Organisationsstruktur kann in der Betriebsphase geändert bzw. durch eine neu definierte Struktur ganz ersetzt werden.

Zwecks Unterstützung der organisatorischen Adaptionen sind im Systemrahmen Werkzeuge vorgesehen, die das gezielte Organisationsmanagement erleichtern und zugleich die Konsistenz der Aufbauorganisation mit dem bestehenden Lösungsumfang sicherstellen. Diese Werkzeuge werden den unternehmensinternen Administratoren zur Verfügung gestellt, die die Verantwortung für die Verwaltung der Aufbauorganisation übernehmen und über erforderliche

Systemrechte verfügen. Mithilfe der in Business ByDesign integrierten Werkzeuge können sowohl die ad hoc Änderungen der bestehenden Organisationsstruktur durchgeführt, als auch die Entwürfe der künftig geplanten Organisationsstrukturen angelegt werden.

Die Adaptionen der Organisationsstruktur erfolgt mithilfe des in die Systemoberfläche integrierten Editors, der in der Abbildung 5.16 dargestellt wurde.

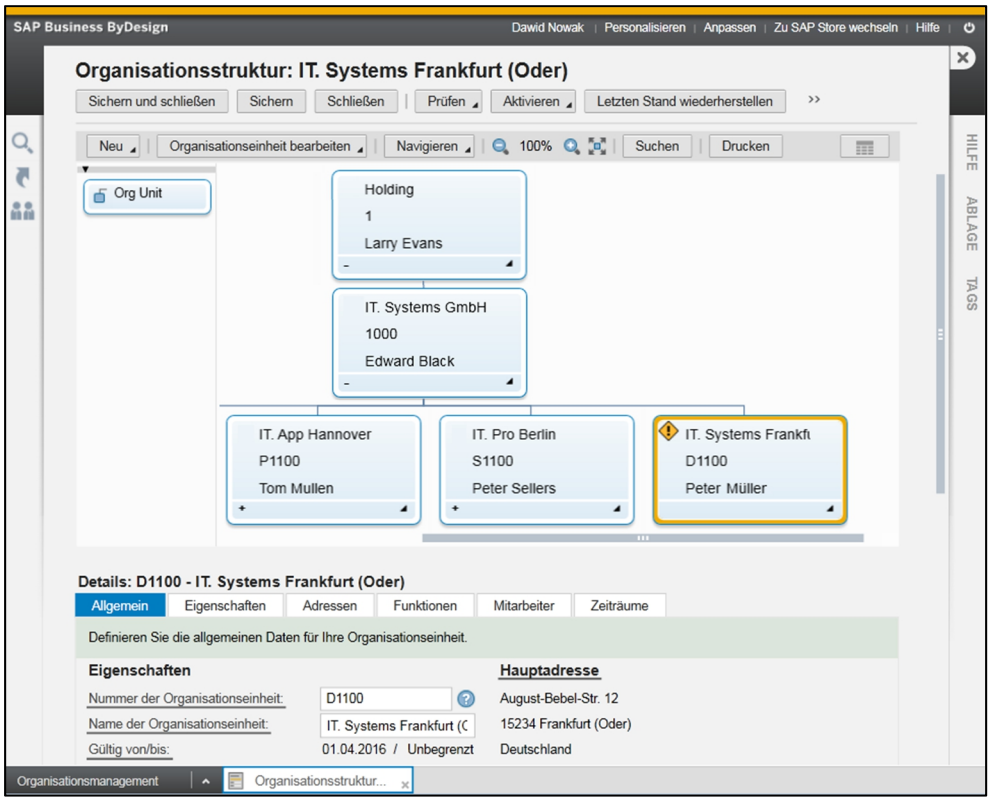





**Abbildung 5.16:** Erweiterte Organisationsstruktur des Musterunternehmens in Business ByDesign

Der Editor besteht aus zwei Bereichen – dem Modellierungsbereich in der Mitte des Fensters und dem sich darunter befindenden *Details-Bereich*. In dem Modellierungsbereich kann die Organisationsstruktur bearbeitet werden, indem die Organisationseinheiten per Drag-and-Drop hinzugefügt bzw. auf die entsprechende Ebene des dargestellten Organigramms verschoben werden. Im Anschluss daran müssen im Details-Bereich des Editors für jede neu angelegte Organisationseinheit die kontextsensitiven betrieblichen Daten vervollständigt werden. Diese Daten umfassen sowohl die allgemeinen Informationen über die neuen Organisationseinheiten – den Namen, die individuelle Nummer der Organisationseinheit, ihren Gültigkeitszeitraum und die Adresse – als auch die rechtlichen, finanzrechtlichen und funktionalen Eigenschaften, die in den nachfolgenden Registerkarten gepflegt werden müssen. Die modifizierte bzw. neu definierte Organisationsstruktur kann nach Überprüfung ihrer Konsistenz sofort aktiviert bzw. als geplante Struktur gespeichert werden. Während die aktivierte Struktur sofort gültig ist, wird die geplante Struktur automatisch am ersten Tag ihrer Gültigkeit durch das System aktiviert.

Technisch betrachtet wird infolge des dargestellten Modellierungs- und Parametrisierungsvorgangs ein Konfigurationsprofil der Aufbauorganisation in den Systemtabellen abgelegt, das durch die anderen Systemmodule referenziert wird.

In dem in der Abbildung 5.16 dargestellten Beispielszenario wurde mithilfe des Editors die bestehende Organisationsstruktur des Musterunternehmens (*IT. Systems GmbH*) um eine neue Niederlassung in Frankfurt (Oder) erweitert. Im Details-Bereich wurden die geschäftsbezogenen Daten vervollständigt und der Filialleiter (*Peter Müller*) zu der neuen Organisationseinheit zugewiesen. Die neue Niederlassung wird ab dem 01.04.2016 in die bestehende Organisationsstruktur übernommen.

Die in Business ByDesign eingebauten Werkzeuge ermöglichen es, die Änderungen der Aufbauorganisation im gesamten Systemlebenszyklus eigenständig durchzuführen. Die vorgenommene Machbarkeitsstudie stellt einen Beweis für die Flexibilität des cloudbasierten ERP-Systems im Kontext der Adaption der unternehmerischen Organisationsstruktur dar und bestätigt die Umsetzbarkeit der organisatorischen Änderungen im Rahmen eines On-Demand-Selfservices.

### 5.7.2 System II: Scopevisio

Die Umsetzbarkeit jeglicher Adaptionen der Aufbauorganisation in Scopevisio ist durch die integrierten Werkzeuge sichergestellt, die für die unternehmensinternen Administratoren über die Benutzeroberfläche des cloudbasierten Systems zugänglich sind.

Das zentrale Werkzeug stellt der in der Abbildung 5.17 präsentierte *Organigramm-Editor* dar, mit dessen Hilfe die Aufbauorganisation des Unternehmens im Systemrahmen abgebildet werden kann. Das Werkzeug steht im gesamten Systemlebenszyklus zur Verfügung und wird sowohl zur Abbildung der initialen Organisationsstruktur als auch zur Umsetzung ihrer Anpassungen während der Postimplementierungsphase verwendet.

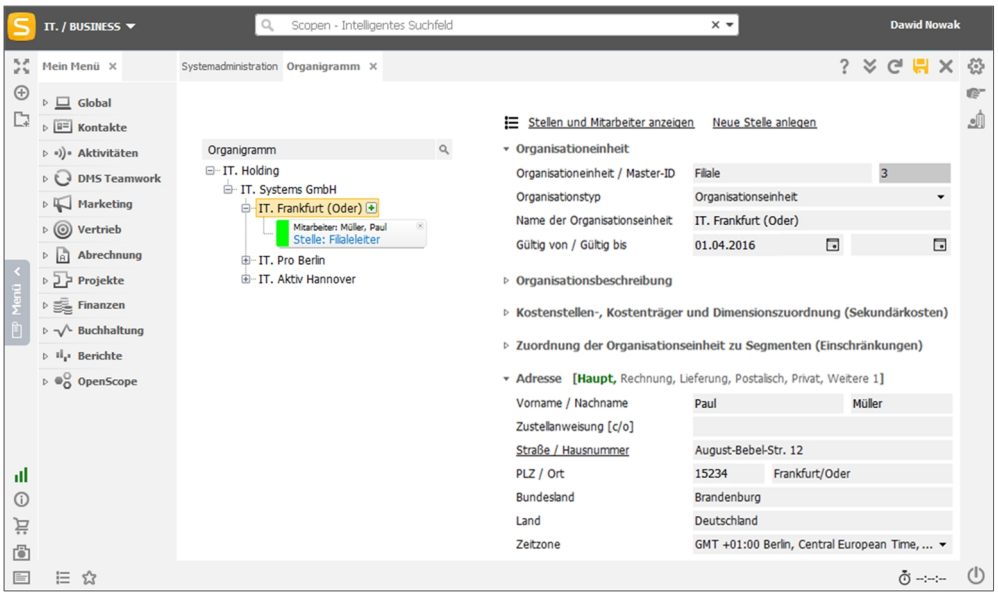





**Abbildung 5.17:** Benutzerdefinierte Organisationsstruktur in Scopevisio

Die Editor-Oberfläche besteht aus zwei Bereichen, die entsprechend zur Modellierung der Organisationsstruktur und Erfassung der kontextintensiven, organisationseinheitsbezogenen Informationen vorgesehen sind. Das im zentralen Bereich der Systemseite verankerte *Organigramm* stellt ein interaktives Modellierungswerkzeug dar, mit dessen Hilfe die Hierarchie der Aufbauorganisation in Form einer Baumstruktur abgebildet werden kann.

Die neuen Organisationseinheiten können durch Anklicken der [+]-Schaltfläche auf jeder beliebigen Organisationsebene hinzugefügt werden. Für jede angelegte Organisationeinheit müssen

in dem rechts angezeigten Formular neben den allgemeinen Informationen (Name, Adresse, etc.) auch die kontextintensiven Betriebsdaten vervollständigt werden, die eine Integration der Aufbauorganisation mit anderen Systemmodulen sicherstellen. Anhand der benutzerdefinierten Gültigkeitsdaten wird eine automatische Aktivierung bzw. Deaktivierung der Organisationseinheit durch das System ausgeführt.

Im Anschluss können der neu angelegten Organisationseinheit die Mitarbeiter zugewiesen werden. In diesem Kontext wird das dedizierte Mitarbeiterformular (rechts) verwendet, das durch Anklicken der Organisationseinheit aktiviert wird. Die Zuweisung des Mitarbeiters zu der Organisationseinheit erfolgt durch Vervollständigung der *Abteilung*- und *Position-Felder* des angezeigten Formulars.

Im Rahmen des dargestellten Beispielszenarios wurde mithilfe des Organigramm-Editors die bestehende Organisationsstruktur des Musterunternehmens (*IT. Systems GmbH*) mit einer neuen Niederlassung in Frankfurt (Oder) erweitert. Im rechten Bereich wurden die geschäftsbezogenen Daten vervollständigt und ein Filialleiter (*Paul Müller*) der neuen Organisationseinheit zugewiesen. Die neu definierte Organisationseinheit erweitert die bestehende Organisationsstruktur ab dem 01.04.2016.

Ähnlich wie im Fall des ursprünglich beschriebenen Systems wird in Scopevisio anhand des erstellten Organigramms (links) und der vergebenen Parameterwerte (rechts) ein Referenzmodell erstellt, welches durch andere Systemmodule referenziert wird.

Die durchgeführte Machbarkeitsstudie weist auf die anwenderseitige Adaptierbarkeit der im Systemrahmen abgebildeten Aufbauorganisation hin und bestätigt die Flexibilität der Organisationsstruktur im Rahmen der Postimplementierungsphase.

### 5.7.3 System III: Odoo

Die Verwaltung der Aufbauorganisation in Odoo ist durch die Werkzeuge sichergestellt, die im *Mitarbeiter-Modul* enthalten sind. Die bereitgestellten Werkzeuge stehen den unternehmensinternen Administratoren im gesamten Systemlebenszyklus zur Verfügung und ermöglichen es, die erforderlichen Adaptionen der Aufbauorganisation und des Organisationsmanagements eigenständig umzusetzen.

Sowohl die initiale Definition der Organisationsstruktur als auch ihre nachfolgenden Adaptionen werden mithilfe des in die Benutzeroberfläche integrierten Werkzeuges vorgenommen, das im Hintergrund der Abbildung 5.18 dargestellt ist.

Die neuen Organisationseinheiten werden nacheinander über das oben angezeigte Eingabeformular zur Organisationsstruktur des Unternehmens hinzugefügt. In diesem Zusammenhang müssen drei Eingabefelder ausgefüllt werden, die entsprechend für die Definition des Namens der Abteilung, die Zuweisung des Abteilungsleiters und zur Bestimmung der organisatorischen Zugehörigkeit bzw. Hierarchie vorgesehen sind.

Die im ERP-System vorhandenen Organisationseinheiten werden auf der Übersichtsseite – *Abteilungen* – in Form von Rechtecken dargestellt. Der Anwendungsexperte kann die Hintergrundfarbe der einzelnen Rechtecke ändern bzw. die Organisationseinheiten nach ausgewählten Kriterien (z. B. Hauptabteilung, Leiter etc.) gruppieren. Durch Anklicken der innerhalb des einzelnen Rechtecks angezeigten *Mitarbeiter-Schaltfläche* können der ausgewählten Organisationseinheit die Mitarbeiter zugewiesen werden. Die Erfassung der Mitarbeiterdaten erfolgt über das dedizierte Mitarbeiterformular (rechts), das neben den allgemeinen Personaldaten (Adresse, Telefonnummer, etc.) die Information über die organisatorische Zuordnung und die besetzte Arbeitsstelle enthalten soll.

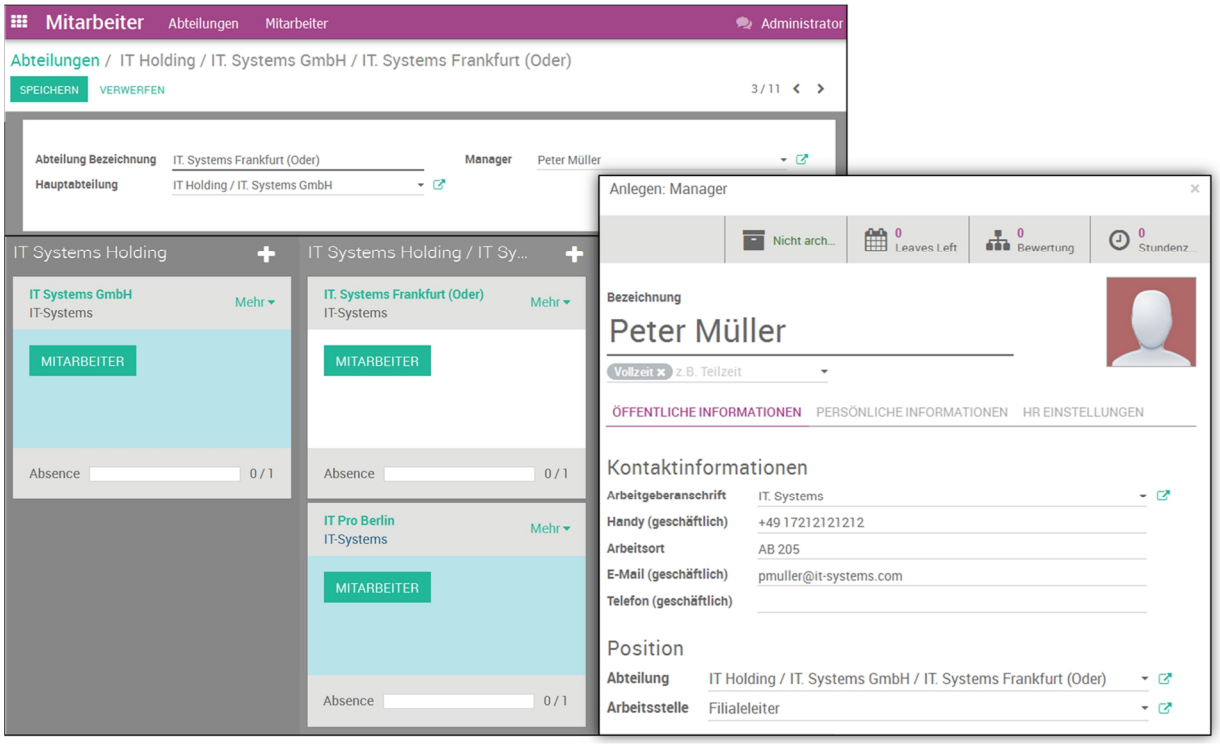





**Abbildung 5.18:** Benutzerdefinierte Organisationsstruktur in Odoo

Der initiale Schritt der Erstellung einer neuen Organisationseinheit wurde in der Abbildung (links) veranschaulicht. Im Rahmen des Beispielszenarios wurde die Aufbauorganisation der Mustergesellschaft (*IT. Systems GmbH*) um die neue Organisationseinheit – *IT. Systems Frankfurt (Oder)* – erweitert, der *Peter Müller* als Leiter zugewiesen wurde.

Nach dem Speichern der Eingaben in der *Abteilungen-Konfigurationsseite* erscheint die neue Abteilung auf der Übersichtsseite, der die aktuelle Struktur des Musterunternehmens entnommen werden kann. Darauf aufbauend wurde der neuen Organisationseinheit ein Mitarbeiter mithilfe des dedizierten Formulars zugewiesen. In diesem Kontext wurden die positionsbezogenen Daten des im Vordergrund angezeigten Mitarbeiterformulars mit dem Abteilungsnamen und der Bezeichnung der zu besetzenden Arbeitsstelle vervollständigt.

Die im Rahmen der Machbarkeitsstudie verwendeten Werkzeuge stehen dem Anwendungsexperten im gesamten Systemlebenszyklus zur Verfügung und ermöglichen es, jegliche Änderung der unternehmerischen Aufbauorganisation im Systemrahmen nachzubilden. Die Ergebnisse der vorgenommenen Studie bestätigen somit die Adaptierbarkeit der unternehmerischen Strukturen im Rahmen eines On-Demand-Selfservices und stellen einen Beweis für die Flexibilität des Odoo-Systems im Kontext der organisatorischen Änderungen dar.

## 5.8 Adaption des Prozessablaufs

Die Adaption der betrieblichen Prozesse und Prozeduren stellt eine der wichtigen Customizing-Aufgaben der Implementierungsphase dar. Die Änderung bzw. die Erweiterung der bestehenden Geschäftsabläufe kann jedoch auch während der Postimplementierungsphase notwendig sein, wenn z. B. infolge der Geschäftsprozessreorganisation Inkonsistenzen zwischen den realen und den im System abgebildeten Prozessen entstehen. Die Untersuchung der bereitgestellten Systemflexibilität im Kontext der Geschäftsprozesse und Prozeduren stellt den Kernpunkt dieses Abschnitts dar.

Die Machbarkeitsstudie setzt die Erweiterung des bestehenden Bestellabwicklungsprozesses um ein internes Genehmigungsverfahren voraus, dessen Schritte in der Abbildung 5.19 veranschaulicht wurden.

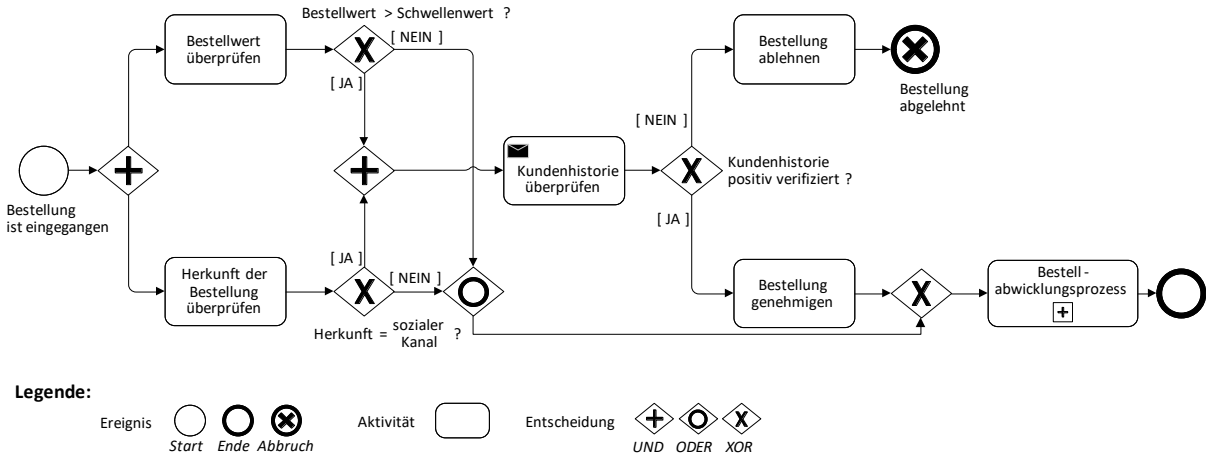



**Abbildung 5.19:** BPMN-Diagramm zum Ablauf des Genehmigungsverfahrens

Der Anpassungsanforderung liegt die unternehmerische Regelung zugrunde, die eine zusätzliche Kontrolle der Kundenhistorie bei den über die sozialen Kanäle eingereichten Bestellungen erfordert. Voraussetzung dafür ist, dass bei Bestellungen mit einem Nettowert, der größer als 2.000 EUR ist, eine gesonderte Genehmigung des Kundenauftrags seitens eines Verantwortlichen erfolgen muss.

Zur Beschleunigung des Prozessablaufs sollen bei der Erfassung der Bestelldaten der Bestellwert und die Herkunft der Auftragserteilung automatisch durch das ERP-System überprüft werden. Bei den Bestellungen, die über soziale Nachrichtendienste eingereicht wurden und deren Bestellwert den festgelegten Schwellenwert übersteigt, muss der Bestellabwicklung ein Genehmigungsschritt vorausgegangen sein. In diesem Fall generiert das System eine Genehmigungsaufgabe in der Aufgabenliste des Mitarbeiters, der für die Validierung der bisherigen Kundenhistorie und der darauffolgenden Entscheidung über die Genehmigung bzw. Stornierung der Bestellung zuständig ist. Ist das Ergebnis der Kundenvalidierung positiv, wird die Bestellung genehmigt und die Bestellabwicklung in die Wege geleitet. Bei einem negativen Prüfergebnis wird die Bestellung abgelehnt und der Kundenauftrag kommt nicht zustande. In Fällen, bei denen die Bestellung über traditionelle Bestellwege (z. B. E-Mail, Fax, Telefonat) erfolgt bzw. der Bestellwert kleiner als der festgelegte Schwellenwert ist, wird der Bestellabwicklungsprozess sofort ohne zusätzliche Genehmigung initiiert.

Im Rahmen dieses Beispielszenarios soll überprüft werden, ob die standardisierten Geschäftsprozesse gemäß den Anforderungen des Anwenderunternehmens während der Postimplementierungsphase adaptiert bzw. modifiziert werden können. Besonderer Fokus wird dabei auf die Werkzeuge gelegt, mit deren Hilfe die geplante Prozesserweiterung im Rahmen eines On-Demand-Selfservices umgesetzt werden kann.

### 5.8.1 System I: Business ByDesign

Im Rahmen des cloudbasierten ERP-Systems von SAP sind zum Zweck der Definition eines benutzerdefinierten Genehmigungsverfahrens die integrierten Konfigurationswerkzeuge vorgesehen, mit deren Hilfe die Einkaufs- und Verkaufsprozesse um zusätzliche benutzerdefinierte Genehmigungsregeln erweitert werden können. Die konfigurierbaren Genehmigungsregeln erlau-

ben es, in Abhängigkeit von den eingestellten Bedingungen in den laufenden Prozess einzugreifen und den Prozessablauf zu steuern.

Die Definition einer Genehmigungsregel in Business ByDesign kann von einem Anwendungsexperten vorgenommen werden, der die einzelnen Ablaufschritte der Genehmigungsregel konfigurieren muss. Für jeden Schritt des Genehmigungsverfahrens werden die Kriterien im System angelegt, anhand derer die automatische Ausführung des konkreten Genehmigungsschrittes erfolgen wird. Daneben muss für jeden einzelnen Schritt, die für die Genehmigung verantwortliche Person zugewiesen werden.

Im Rahmen des Beispielszenarios wurde das geplante Genehmigungsverfahren mithilfe der integrierten Konfigurationswerkzeuge implementiert. Die Konfiguration der Genehmigungsregel wurde im Hintergrund der Abbildung 5.20 veranschaulicht.

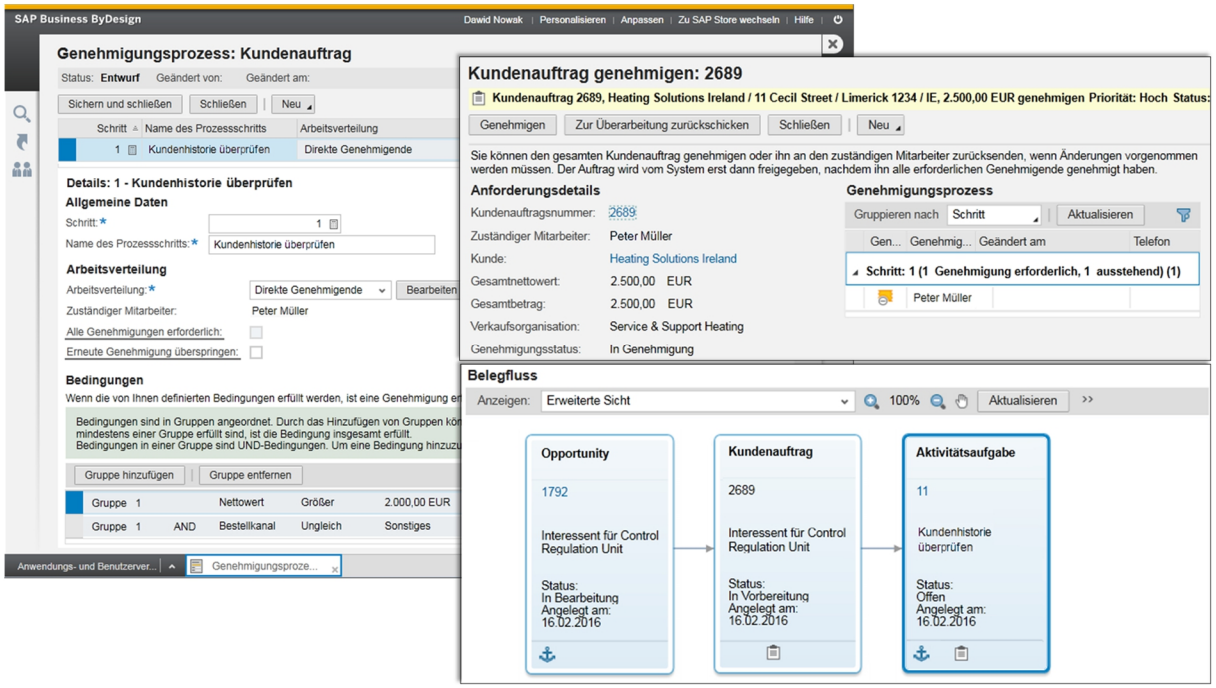





**Abbildung 5.20:** Der erweiterte Auftragsabwicklungsprozess in Business ByDesign

Die definierte Regel besteht aus einem Genehmigungsschritt – *Kundenhistorie überprüfen* –, dessen Einzelheiten im Details-Bereich der Konfigurationsseite definiert wurden. Die automatische Ausführung der definierten Genehmigung erfolgt anhand der im Bedingungen-Bereich festgelegten Kriterien. Werden die Kriterien für beide Attribute der neu angelegten Bestellung erfüllt, d. h. übersteigt der Nettowert 2.000 EUR und ist die Bestellung über einen sozialen Bestellkanal eingegangen, wird der Bestellabwicklungsprozess mit dem Prozessschritt *Kundenhistorie überprüfen* fortgesetzt. Bei der Ausführung dieses benutzerdefinierten Prozessschrittes wird eine Aktivitätsaufgabe erstellt und diese wird dem direkt Genehmigenden (*Peter Müller*) zugewiesen. Die Aktivitätsaufgabe stellt eine Anforderung zur Ausführung einer Aktion, die in der Aufgabenliste des Genehmigenden erscheint.

Der Ablauf eines benutzerdefinierten Genehmigungsverfahrens wurde im Vordergrund der Abbildung 5.20 anhand des Belegflusses eines Kundenauftrags veranschaulicht.

Beim Anlegen eines neuen Kundenauftrags (Nr. 2689) überprüft das System seinen Nettowert und die Herkunft. Da die Bestellung über einen sozialen Bestellkanal eingereicht wurde und

der Bestellwert 2.000 EUR übersteigt, erzeugt das System automatisch eine Aktivitätsaufgabe (*Kundenhistorie überprüfen*), die an den zuständigen Mitarbeiter (*Peter Müller*) weitergeleitet wird. Dabei wird vorausgesetzt, dass der für die Auftragsgenehmigung zuständige Mitarbeiter eine Verifikation der Kundenhistorie durchführt und die Entscheidung über eine Annahme bzw. Ablehnung des erhaltenen Auftrags trifft (rechts). Ist die Verifikation positiv, kann der Kundenauftrag durch Anklicken der *Genehmigen-Schaltfläche* in der Aufgabenansicht genehmigt und die weiteren Schritte des Bestellabwicklungsprozesses können ausgeführt werden. Wird die Genehmigung verweigert, wird der Kundenauftrag abgesagt und der Bestellabwicklungsprozess kommt nicht zustande. Die Aktivitätsaufgabe ist im Belegfluss der angelegten Bestellung ausgewiesen. Erst nach der Genehmigung der Bestellung wird sie in einen Kundenauftrag umgewandelt und die nachfolgenden Schritte des Bestellabwicklungsprozesses werden ausgeführt.

Business ByDesign ermöglicht es, die zusätzlichen Aktivitätsaufgaben auch manuell bei einzelnen Prozessschritten anzulegen und dadurch die Abwicklung der Geschäftsprozesse zu beeinflussen. Obwohl die frei definierbaren Genehmigungsregeln und Aktivitätsaufgaben einige zusätzliche Flexibilität des Ablaufs sicherstellen, erlauben sie keine weitreichende Adaption der bestehenden Geschäftsprozesse – insbesondere der Prozesslogik – vorzunehmen. Die Adaption des ausgewählten Lösungsumfangs kann im Systemrahmen lediglich mithilfe der Parametrisierung vorgenommen werden, soweit die gewünschte Adaption als eine Variante im Adaptionskatalog vorgesehen wurde und aktivierbar ist (vgl. Abschnitt 5.3.1). Allerdings ermöglichen die im System eingebauten Werkzeuge keine Modifikation bzw. Erweiterung der bestehenden Geschäftsabläufe vorzunehmen.

Mehr Freiheit in diesem Kontext bietet die bereitgestellte Entwicklungsumgebung an, die eine Anpassung bzw. Erweiterung der bestehenden Geschäftslogik gestattet. Die eigenentwickelten Systemadaptionen können im Adaptionskatalog verankert werden und im Rahmen einer Reparametrisierung den bestehenden Lösungsumfang erweitern. Die eigenständige Erweiterung des Lösungsumfangs von Business ByDesign mittels der bereitgestellten Entwicklungsumgebung wird in der Machbarkeitsstudie im Abschnitt 5.10.1 genauer analysiert.

### 5.8.2 System II: Scopevisio

Die in Scopevisio enthaltenen Prozesse stellen den Best-Practices abgeleiteten Standards dar, die im anwenderseitigen Customizing lediglich mithilfe der vordefinierten Parameter innerhalb des vom Systemhersteller vorgesehenen Rahmens an die Anforderungen des Anwenderunternehmens angepasst werden können.

Neben der Adaption der Prozesslogik, die mittels der Einstellung einzelner Parameterwerte vorgenommen werden kann, werden vom Systemhersteller keine Werkzeuge zur Verfügung gestellt, die eine unternehmenskonforme Adaption der bestehenden bzw. eine Festlegung der alternativen Prozessabläufe ermöglichen. Wegen der mangelnden Flexibilität des Systems in Bezug auf die Adaptierbarkeit der Geschäftsabläufe konnte das der Machbarkeitsstudie zugrunde gelegte Genehmigungsverfahren im Systemrahmen nicht umgesetzt werden.

### 5.8.3 System III: Odoo

Der direkte Zugang zum Quellcode der Basisprozesse ist in der bereitgestellten cloudbasierten Version des ERP-Systems nicht sichergestellt, obwohl Odoo grundsätzlich als ein quelloffenes System vermarktet wird. Wenn einerseits eine direkte Modifikation der standardisierten Geschäftsprozesse nicht vorgenommen werden kann, werden andererseits im Systemrahmen Werkzeuge vorgesehen, mit deren Hilfe eine zusätzliche benutzerdefinierte Geschäftslogik angelegt werden kann. Im Kontext der Erweiterung der Geschäftslogik stehen dem Anwen-

dungsexperten die sogenannten *automatischen* bzw. *serverseitigen Aktionen* zur Verfügung, die im Entwicklermodus mithilfe der integrierten technischen Werkzeuge erstellt werden können.

Eine automatische Aktion in Odoo stellt eine benutzerdefinierbare Regel dar, die in Abhängigkeit von den definierten Kriterien automatisch durch das System ausgeführt wird. Eine automatische Aktion besteht in der Regel aus mindestens einer *Serveraktion*, die eine aufrufbare serverseitige Python-Prozedur darstellt (vgl. Odoo S. A. 2016b). In Odoo sind neben der standardisierten vorkonfigurierten Serveraktionen (z. B. *E-Mail senden*, *Datensatz ändern*, etc.) auch frei definierbare Python-Prozeduren als Bestandteile einer automatischen Aktion zulässig, die eigenentwickelt und im Systemrahmen gespeichert werden können.

In der Machbarkeitsstudie wurde die geplante Genehmigungsregel mithilfe der integrierten technischen Werkzeuge in Form einer automatischen Aktion angelegt. Die Konfiguration der automatischen Aktion wurde im Hintergrund der Abbildung 5.21 veranschaulicht.

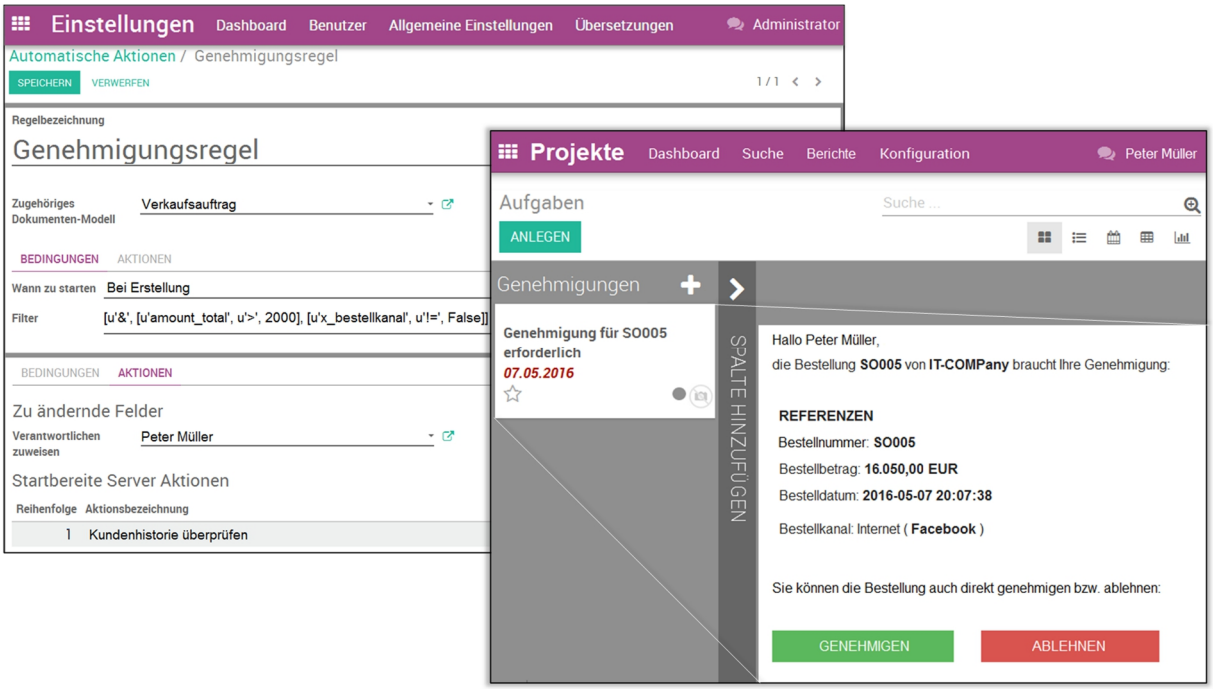





**Abbildung 5.21:** Die benutzerdefinierte Genehmigungsregel in Odoo

Die Systemseite im Hintergrund stellt die automatische Aktion in der Entwurfsansicht dar. Die angelegte Aktion wird *bei Erstellung* eines neuen Verkaufsauftrags durch das System automatisch ausgeführt. Die Aktionsausführung erfolgt anhand der definierten *Filter-Formel*. Das System überprüft dabei die Konditionen, die für zwei Attribute der neu angelegten Instanz des Verkaufsauftrags – den Bestellbetrag und den Bestellkanal – vordefiniert wurden. Liegt der Bestellbetrag über 2.000 EUR (*amount_total > 2000*) und wurde die Bestellung über einen der sozialen Bestellkanäle (*x_bestellkanal != False*) eingereicht, führt das System die benutzerdefinierte Serveraktion – *Kundenhistorie überprüfen* – durch. Die benutzerdefinierte Serveraktion stellt eine Python-Prozedur dar, die die einzelnen Auftragsdaten (u. a. Bestellnummer, Bestellbetrag, Bestelldatum etc.) abfragt und dynamisch eine HTML-basierte Vorlage der Aktivitätsanforderung erzeugt (siehe Anhang B.1).

Infolge der ausgeführten Serveraktion erscheint die mit der zu genehmigenden Bestellung (*SO005*) korrespondierende Aktivitätsanforderung unter Genehmigungsaufgaben des Verantwortlichen (*Peter Müller*), wie im Vordergrund der Abbildung 5.21 dargestellt ist.

Nach dem geplanten Prozedurverlauf wird erwartet, dass der verantwortliche Mitarbeiter eine Verifikation der Kundenhistorie durchführt und darauf aufbauend die Entscheidung über die Annahme bzw. die Ablehnung der Bestellung trifft. In diesem Zusammenhang stehen dem Genehmigenden in der Benutzeransicht der zugewiesenen Aufgabe zwei Schaltflächen – *Genehmigen* und *Ablehnen* – zur Verfügung. Durch Anklicken der gewählten Schaltfläche wird die entsprechende Aktion getriggert. Wird die *Ablehnen-Schaltfläche* angeklickt, bricht das System an dem aktuellen Prozessschritt ab und storniert die Bestellung. Genehmigt der Verkaufsmanager die Bestellung, wird der nachfolgende Schritt des Bestellabwicklungsprozesses ausgeführt.

Mithilfe der im ERP-System integrierten technischen Werkzeuge wurde das geplante Genehmigungsverfahren ohne eine direkte Änderung des standardisierten Bestellabwicklungsprozesses umgesetzt. Trotzt der bewiesenen Anwendbarkeit stellen die automatischen Aktionen objektiv gesehen keine Alternative zu einer direkten Adaption der standardisierten Geschäftsprozesse dar.

Die Limitation der automatisierten Aktionen im Kontext der Anpassung des Prozessablaufs ist insbesondere bei komplexen kontextübergreifenden Geschäftsszenarien ersichtlich, die eine Modifikation der standardisierten Geschäftslogik voraussetzen. Aus diesem Grund bieten die Systemanbieter eine entgeltliche Customizing-Dienstleistung an, in der die standardisierten Odoo-Workflows von anbieterseitigen Entwicklern an die Anforderungen des Anwenderunternehmens adaptiert werden können.

## 5.9 Integration externer Webanwendung

Die Kopplung des ERP-Systems mit anderen Anwendungen des Geschäftsumfelds und gebrauchsfertigen Webdiensten stellt eine wichtige Aufgabe dar, die häufig im Rahmen der Adaptionsprozesse umgesetzt werden muss. Viele Studien zeigen Beispiele für eine Erweiterung der Systemfunktionalität durch Einbindung der Webdienste bzw. eine wirkungsvolle Unterstützung der Geschäftsprozesse durch Integration von externen Anwendungen (vgl. z. B. Nowak und Kurbel 2014; Grabot et al. 2013; Shankararaman und Lum 2013; Wang et al. 2011).

In diesem Beispielszenario soll die Möglichkeit einer anwenderseitigen Integration externer Inhalte, insbesondere auf der Anwendungs- und Präsentationsebene validiert werden. Die Integration auf der Präsentationsebene besteht in der Einbindung der Schnittstellen der zu integrierenden Systeme in eine Benutzerschnittstelle, mit deren Hilfe ein automatisierter Datenaustausch zwischen den Anwendungen realisiert werden kann. Dadurch wird der Benutzer im Hinblick auf eine effiziente Bearbeitung der Geschäftsprozesse unterstützt. Die Integration auf der Anwendungsebene erfolgt durch die Verwendung der Funktionen einer Anwendung in einer anderen Anwendung.

Das konzipierte Beispielszenario setzt die Integration des webbasierten Geolokalisationsdienstes – Google Maps (Google 2017b) – und der ihm zugehörigen Visualisierungstools in die Systemumgebung zur Veranschaulichung der Effektivität von internetbasierten Verkaufsaktionen im geographischen Kontext voraus. Die Realisierung des Szenarios erfordert die Sicherstellung eines effektiven Informationsaustausches zwischen einem Webdienst und dem ERP-System mittels Inanspruchnahme der bereitgestellten Programmierschnittstellen.

Im Rahmen dieses Beispielszenarios wird untersucht, welche Möglichkeiten das jeweilige cloudbasierte ERP-System hinsichtlich der Integration mit webbasierten Anwendungen zur Verfügung stellt und wie sie von dem Anwendungsexperten im Rahmen eines On-Demand-Selfservices in Anspruch genommen werden können.

### 5.9.1 System I: Business ByDesign

Das cloudbasierte ERP-System von SAP ermöglicht eine Integration sowohl mit webbasierten Anwendungen als auch lokal installierbaren Systemen im Rahmen eines On-Demend-Self-Service zu erstellen. SAP Business ByDesign stellt eine umfangreiche gut dokumentierte Schnittstellen-Bibliothek zur Verfügung, die insgesamt über 250 unterschiedliche Webdienste und APIs enthält (vgl. SAP SE 2016c). Diese Webdienste ermöglichen einen Datenaustausch zwischen dem cloudbasierten ERP-System von SAP und den weiteren Bestandteilen des unternehmerischen Umfelds zu erstellen. Mithilfe der API-Befehle kann eine externe Anwendung sowohl die systeminterne Datenbestände abfragen als auch einige Datenänderungen durchführen.

Neben der Integration, die auf der Anwendungsebene stattfinden kann, erlaubt Business ByDesign auch, die Integration der systemexternen Inhalte und Anwendungen direkt in der Benutzeroberfläche vorzunehmen.

In diesem Kontext erhält der Anwendungsexperte einen Zugriff auf den integrierten *Mashup-Editor*, mit dessen Hilfe individuelle Komponenten der Systemoberfläche in Form der *Mashups* entwickelt werden können. Mashups stellen Anwendungen dar, die die externen Webkomponenten bzw. webbasierten Datenquellen mit den systemintern verfügbaren Informationen zusammenführen und in Form einer Oberflächenkomponente in der Benutzeroberfläche verankert werden können (vgl. Nowak und Kurbel 2014).

In der Machbarkeitsstudie wurde mithilfe des bereitgestellten Mashup-Editors ein HTML-basierter Mashup erstellt und als eine Oberflächenkomponente in der *Kundenaufträge-Übersichtsseite* des ERP-Systems verankert. Technisch gesehen stellt der angelegte Mashup eine Schnittstelle zwischen den systemexternen Webanwendungen und der Benutzeroberfläche von Business ByDesign dar. Die Grundlage des Mashups bildete der im Mashup-Editor angelegte JavaScript-Programmcode, der im Hintergrund der Abbildung 5.23 zu sehen ist. Der angelegte Mashup erstellt eine Verbindung zu den systeminternen und -externen Schnittstellen. Durch die Inanspruchnahme der durch die APIs bereitgestellten Funktionen verarbeitet er die systeminternen Daten und erzeugt eine virtuelle Landkarte in der Benutzeroberfläche.

Die einzelnen Schritte des Informationsaustausches zwischen den Schnittstellen als auch des Ablaufs der Datenverarbeitung wurden in Form eines Sequenzdiagramms in der Abbildung 5.22 veranschaulicht.

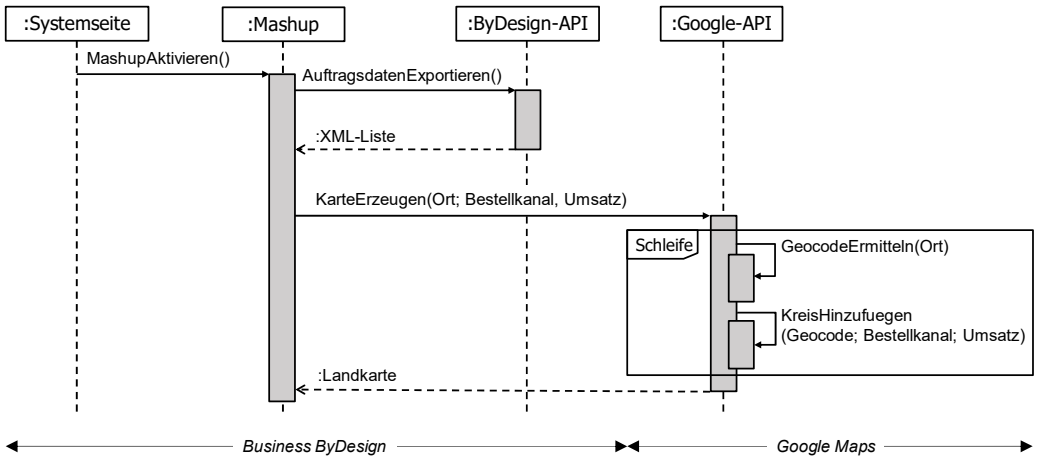
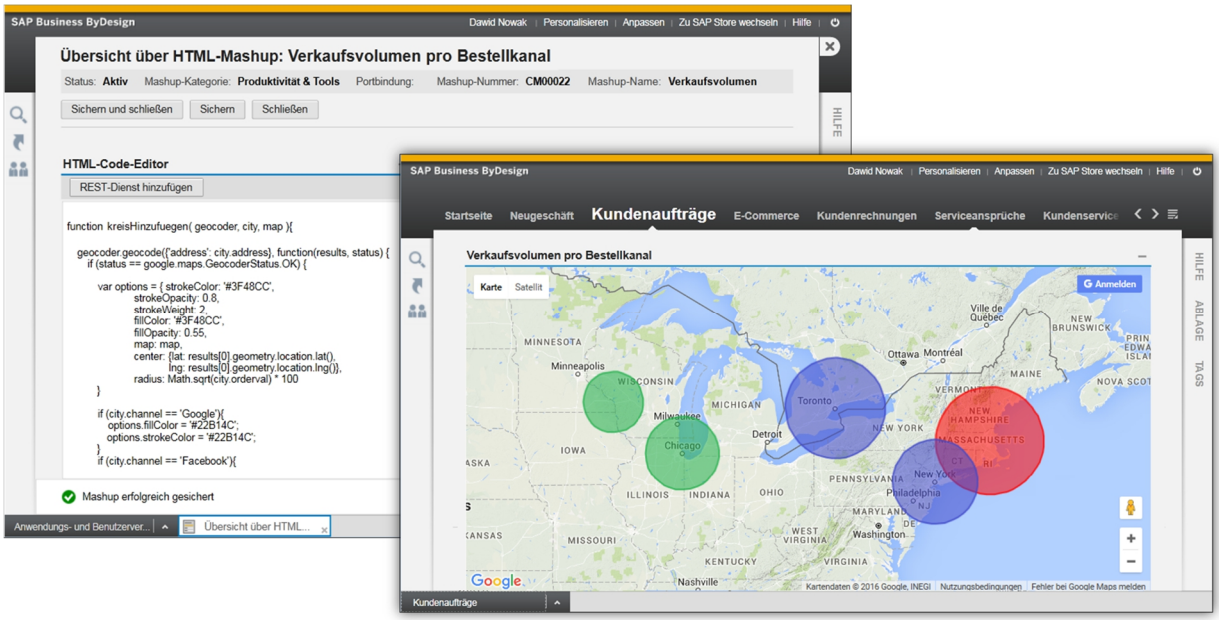



**Abbildung 5.22:** Sequenzdiagramm – Integration von Google-Maps in Business ByDesign

Beim Öffnen der Systemseite wird der Mashup aktiviert, der eine Anfrage an die systeminterne API sendet. Die API generiert eine XML-basierte Austauschdatei als Antwort, welche die Liste der Verkaufsorte und der korrespondierenden voraggregierten Auftragswerte je Bestellkanal enthält. Diese Werteliste wird an die Google Geocoder-API (Google 2017a) gesendet, die anhand der Inputdatei die Geodaten für jeden Verkaufsort ermittelt. Nachfolgend generiert Google Maps-API eine Landkarte mit den Kreisdiagrammen, welche die in der jeweiligen Stadt realisierten Kundenauftragsvolumen widerspiegeln. Die erzeugte Landkarte wird im Rahmen des in der Benutzeroberfläche verankerten Mashup angezeigt, wie die Abbildung 5.23 veranschaulicht.




© SAP SE

**Abbildung 5.23:** Google Maps-Mashup in Business ByDesign

Die Abbildung im Vordergrund zeigt die Mashup-Oberfläche, die das Ergebnis der Datenverarbeitung darstellt. Unter Inanspruchnahme der Visualisierungsinstrumente von Google-Maps wurden die aggregierten Umsatzgrößen ortsbezogen in Form der Kreise unterschiedlicher Größe und Farbe auf einer virtuellen Landkarte dargestellt. Anhand des erzeugten Diagramms kann die Bedeutung der einzelnen Verkaufsorte in Nordamerika allgemein bzw. bezogen auf den ausgewählten Bestellkanal analysiert werden.

Analog wie im dargestellten Beispiel können andere webbasierte Anwendungen bzw. Datenquellen mit Business ByDesign auf der Präsentationsebene gekoppelt werden. In Abhängigkeit von der Architektur des verwendeten Webdienstes werden vom Systemhersteller folgende Mashup-Kategorien unterschieden:

- *Daten-Mashups* führen externe und interne Datenquellen zusammen und stellen das Ergebnis in einer Mashup-Oberfläche dar,
- *URL-Mashups* senden die Serviceanfrage an die URL-adressierbaren Webdienste und präsentieren das erhaltene Ergebnis in der Mashup-Oberfläche,
- *HTML-Mashups* ermöglichen eine direkte Integration der HTML- bzw. JavaScript-basierten Webseiten bzw. Webanwendungen in die Systemoberfläche vorzunehmen (vgl. Schneider 2012, S. 310).

Neben den drei genannten Mashup-Kategorien, die mithilfe des integrierten Mashup-Editors definiert werden können, existieren die sogenannten *benutzerdefinierten Mashups*, die als eine Systemerweiterung eigenständig mithilfe des SAP Cloud Applications Studios entwickelt bzw. über den SAP-Store beschafft werden können (vgl. Konstantinidis et al. 2012, S. 593).

### 5.9.2 System II: Scopevisio

Das ERP-System von Scopevisio stellt eine umfangreiche Webschnittstelle bereit, die die unternehmensinternen Administratoren bzw. Entwickler in die Lage versetzt, eine Integration mit anderen Anwendungen des Geschäftsumfelds vorzunehmen. Die bereitgestellte API-Bibliothek beinhaltet ca. 70 Webdienste, die eine gezielte Kommunikation mit Scopevisio ermöglichen (vgl. Scopevisio AG 2016a). Dabei ist es möglich, einen lesenden und schreibenden Zugriff auf systeminterne Daten von anderen Anwendungen des Geschäftsumfelds zu erstellen.

Im Gegensatz zu dem ursprünglich beschriebenen ERP-System bietet Scopevisio keine Integrationsmöglichkeit auf der Präsentationsebene an. Wegen der Inflexibilität der Benutzeroberfläche können die systemexternen Inhalte nicht direkt in der Systemseite verankert werden. Dies stellte eine zusätzliche Herausforderung an die konzipierte Untersuchung dar und war ein Ansatzpunkt für die Entwicklung einer alternativen Lösung, in der das Ergebnis der vorgenommenen Anwendungsintegration visualisiert werden könnte.

Zur Bewältigung der entstandenen Herausforderung wurde eine konzeptionelle Lösung vorgeschlagen, deren Grundlage eine eigenentwickelte PHP/JavaScript-basierte Webanwendung bildete (siehe Anhang B.2-3). Die Webanwendung wurde auf einem externen Server gehostet und diente als eine Schnittstelle zwischen der Scopevisio-API und den von Google Maps (Google 2017b) bereitgestellten Webdiensten. Im Rahmen der Verarbeitung der systeminternen Daten nimmt die Webanwendung die durch den Geolokalisationsdienst bereitgestellte Funktionalität in Anspruch und erzeugt eine Oberfläche in Form einer HTML-Webseite. Der dem vorgenommenen Integrationsszenario zugrunde liegende Mechanismus wurde in der Abbildung 5.24 mithilfe des Sequenzdiagramms veranschaulicht.

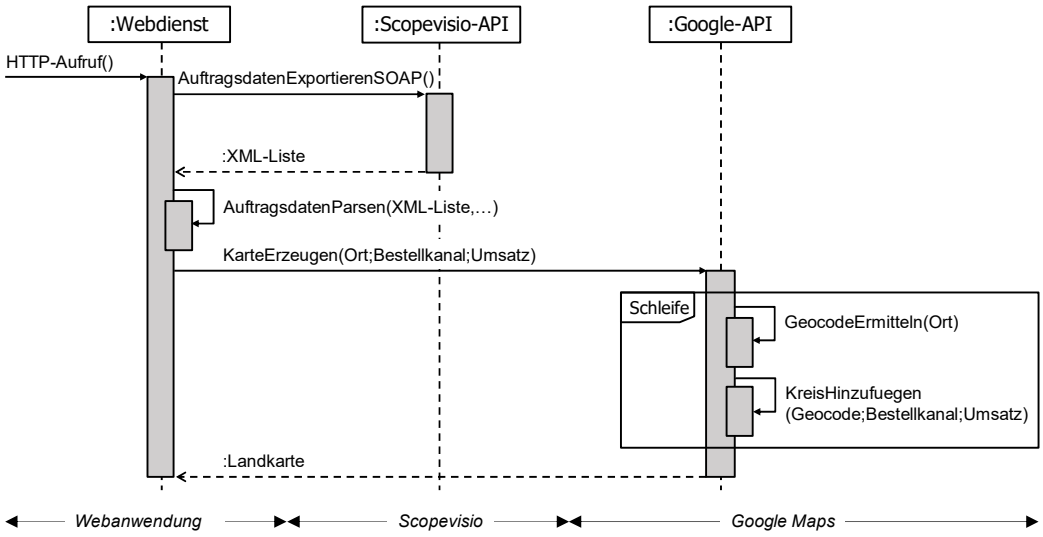


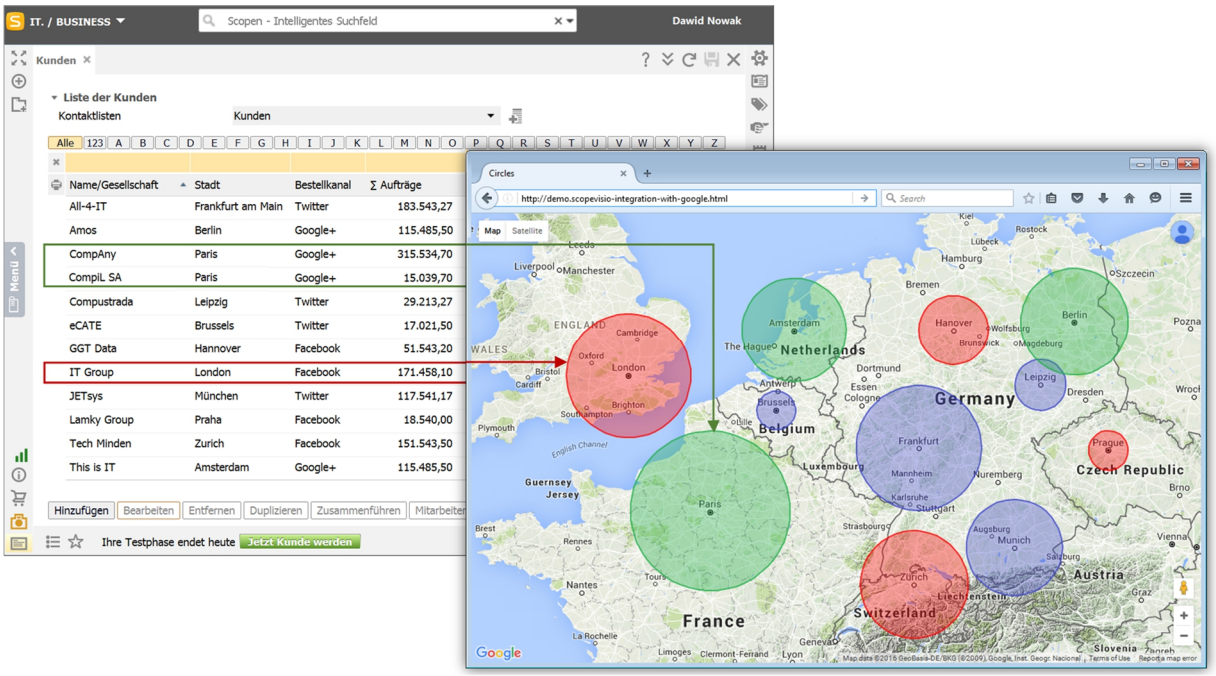

**Abbildung 5.24:** Sequenzdiagramm – Integration von Google-Maps in Scopevisio

Die angelegte Webanwendung wird durch die Ausführung ihrer URL-Adresse im Internetbrowser aktiviert. Infolge des HTTP-basierten Aufrufs (HTTP-Request) richtet die Webanwendung eine SOAP-Nachricht an die webbasierte Scopevisio-API. Die systeminterne API stellt nach

Bearbeitung der eingehenden Anfrage eine XML-basierte Austauschdatei bereit, die die kundenauftragsbezogenen Daten beinhaltet. Die erhaltene XML-Datei wird durch den Webdienst geparst – die Kundenauftragsvolumina werden pro Verkaufsort und Bestellkanal aggregiert.

Die voraggregierten Daten stellen einen Input für die Geocodierung-API (Google 2017a) dar, die die geographische Länge und Breite jedes Verkaufsortes ermittelt. Anschließend werden die einzelnen Verkaufsorte durch den Google Map-Wedienst anhand ihrer Geokoordinaten unter Berücksichtigung der Bestellkanäle und Umsatzwerte auf einer virtuellen Karte markiert. Die durch den Google Map-Webdienst erzeugte Landkarte stellt das Endergebnis des umgesetzten Integrationsszenarios dar, das in der Oberfläche der entwickelten Webanwendung angezeigt wird.

Die Abbildung 5.25 zeigt das Ergebnis des vorgenommenen Integrationsszenarios und verdeutlicht den Zusammenhang zwischen den systeminternen Auftragsdaten und der erstellten Landkarte.




© Scopevisio AG

**Abbildung 5.25:** Anwendungsintegration am Beispiel von Scopevisio

Die Abbildung im Hintergrund gibt einen internen Bericht in Scopevisio wieder, in dem die Kundenauftragsdaten tabellarisch geordnet nach dem Kundennamen, Verkaufsort und Bestellkanal zusammengestellt wurden. Die systeminternen Daten wurden mithilfe der angelegten Webanwendung im geographischen Kontext dargestellt.

Die Abbildung im Vordergrund stellt die Oberfläche der entwickelten Webanwendung dar, in der das Ergebnis der systemübergreifenden Integration verankert wurde. Der angezeigte Inhalt ist eine virtuelle Karte, mit deren Hilfe die systeminternen Verkaufsdaten durch Kreise unterschiedlicher Größe und Farbe visualisiert werden. Die Mitte des Kreises ist durch die geographischen Koordinaten des jeweiligen Verkaufsortes bestimmt, die Kreisgröße entspricht dem aggregierten Kundenauftragswert, der im Rahmen des Ortes realisiert wurde. Zum Zweck der Unterscheidung der bestellkanalbezogenen Verkaufsgrößen wurde eine farbliche Differenzierung vorgenommen. Die Pfeile verdeutlichen den Zusammenhang zwischen den zwei Berich-

ten und verbinden die Kundenauftragsdaten mit dem korrespondierenden Kreis in der Landkarte.

Die durchgeführte Machbarkeitsstudie bestätigt die Möglichkeit einer Kopplung des cloudbasierten ERP-Systems von Scopevisio mit den externen Anwendungen und Webdiensten. Dank der bereitgestellten Programmierschnittstellen erlaubt es Scopevisio, die Integration anderer Anwendungen des unternehmerischen Umfelds im Rahmen eines On-Demand-Selfservices vorzunehmen. Dabei handelt sich um lesende und schreibende API-Befehle, mit deren Hilfe ein effektiver Datenaustausch realisiert werden kann. Im Gegensatz zu den anderen Untersuchungsgegenständen erlaubt es Scopevisio nicht, eine Integration der externen Webinhalte in die Benutzeroberfläche vorzunehmen, so dass die Integration ausschließlich auf der Anwendungsebene realisiert werden kann. Infolge der fehlenden Flexibilität der Benutzeroberfläche konnte das vorgesehene Szenario lediglich zum Teil realisiert werden.

### 5.9.3 System III: Odoo

Odoo stellt den unternehmensinternen Administratoren die Werkzeuge zur Verfügung, mit deren Hilfe eine Integration anderer externen Anwendungen im Rahmen eines On-Demand-Selfservices vorgenommen werden kann. Der Anwendungsexperte kann mit den eingebauten Werkzeugen die externen, webbasierten Inhalte in die Oberfläche des cloudbasierten ERP-Systems einbetten. Dies kann mithilfe des integrierten Views-Editors vorgenommen werden, der eine Entwicklung bzw. Modifikation der Elemente der Benutzeroberfläche ermöglicht. Neben der Integration, die auf der Präsentationsebene stattfinden kann, erlaubt es Odoo, auch eine Integration auf der Anwendungsebene vorzunehmen.

Die bereitgestellte webbasierte Programmierschnittstelle erlaubt es, einen lesenden bzw. schreibenden Zugriff auf die Odoo-Datenmodelle von externen Anwendungen des Geschäftsumfelds herzustellen (vgl. Odoo S. A. 2016e). Mit Unterstützung des webbasierten XML-RPC-Zugriffs können neben der typischen Manipulation an den Datenbeständen auch einige Erweiterungen der bestehenden Datenmodelle realisiert werden. Darüber hinaus stellt die API die dedizierten Webdienste zur Verfügung, die das Auslösen von Workflows und die Ausgabe der in Odoo definierten Berichte von systemexternen Anwendungen ermöglichen.

Im Rahmen der Machbarkeitsstudie wurde mithilfe des bereitgestellten View-Editors ein *iFrame-Element* in der *Pinnwand-Übersichtsseite* verankert. Das iFrame-Element stellt eine HTML-Komponente dar, die es ermöglicht, externe Webinhalte im definierten Bereich der Systemseite zu verankern. Die Grundlage der implementierten Lösung stellt der JavaScript-Quellcode dar (siehe Anhang B.2), anhand dessen der Inhalt des iFrame dynamisch erzeugt wird.

Die entwickelte Webanwendung stellt eine clientseitige JavaScript-basierte Anwendung dar, die die Funktionalität der Odoo-API und Google-Webdienste ergebnisorientiert wiederverwendet. Der Ablauf der einzelnen Datenaustausch- und Datenbearbeitungsprozessschritte wurde in der Abbildung 5.26 durch ein Sequenzdiagramms visualisiert.

Der im iFrame-Element enthaltene Programmcode wird beim Öffnen bzw. Aktualisieren der Systemseite ausgeführt. Im ersten Schritt erstellt die angelegte JavaScript-Funktion eine Verbindung mit der Odoo-API und richtet eine Anfrage an sie. Nach Bearbeitung der Anfrage stellt die systeminterne API eine XML-basierte Datei bereit, die alle Kundenaufträge in Form einer Liste der Verkaufsorte mit den korrespondierenden Auftragswerten und Bestellkanälen beinhaltet.

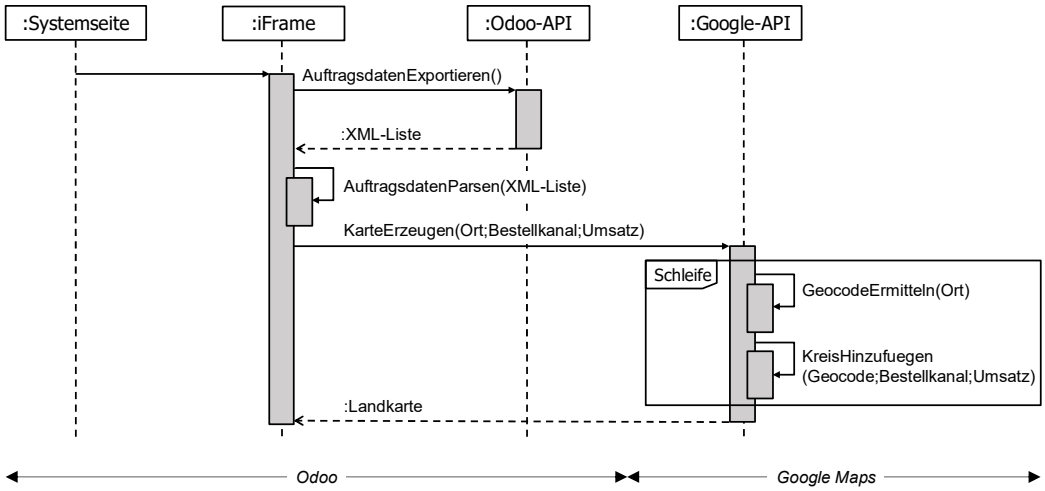
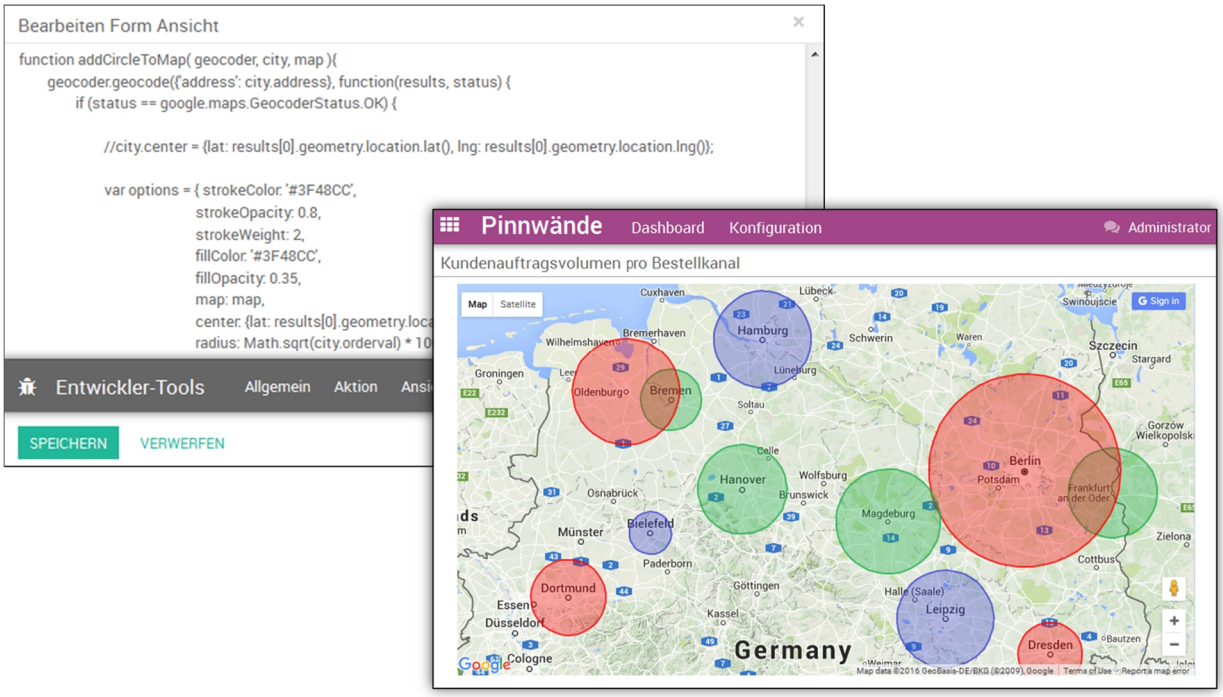



**Abbildung 5.26:** Sequenzdiagramm – Integration von Google-Maps in Odoo

Die durch die API generierte Austauschdatei stellt einen Input für die Parsing-Prozedur dar, die die Auftragswerte bestellkanalbezogen für jeden Ort aggregiert. Das Ergebnis der vorgenommenen Aggregation wird als Parameter einer Anfrage an die Google-API (Google 2017b) gesendet. Der Geolokalisationsdienst ermittelt anschließend die geographische Lage jedes Verkaufsortes und erzeugt aufbauend darauf eine virtuelle Landkarte, in der die gelisteten Verkaufsorte durch Kreise markiert werden. Die Größe und Farbe der Markierung spiegeln entsprechend den aggregierten Kundenauftragswert und den Bestellkanal wider (siehe Abbildung 5.27, rechts). Die generierte virtuelle Karte wird innerhalb des iFrame-Elements dargestellt, welches in die Systemseite eingebettet ist.

Das Ergebnis der vorgenommenen Integration ist im Vordergrund der Abbildung 5.27 veranschaulicht.




© Odoo S. A.

**Abbildung 5.27:** Odoo-Oberfläche mit integriertem Google Maps-iFrame

Der Screenshot im Vordergrund präsentiert die Systemoberfläche, in der die systeminternen kundenauftragsbezogenen Daten mithilfe eines externen Webdienstes visualisiert werden. Im Rahmen des in der Benutzeroberfläche verankerten iFrame-Elementes wird eine virtuelle Karte erzeugt, die eine Übersicht der Verkaufsdaten im geographischen Kontext darstellt. Jeder der angezeigten Kreise spiegelt den aggregierten Auftragswert wider, der als Summe kanalbezogener Auftragswerte innerhalb eines Verkaufsortes interpretiert werden soll. Die Farbe des Kreises differenziert die Bestellkanäle, die für die Einreichung der Kundenaufträge verwendet wurden.

Die in der Machbarkeitsstudie umgesetzten Implementierungen stellen einen Beweis für die anwenderseitige Integration des Odoo-ERP-Systems dar, die auf der Anwendungsebene mithilfe der bereitgestellten Programmierschnittstellen bzw. auf der Ebene der Benutzeroberfläche dank des flexiblen Oberflächemodells realisiert werden kann. Die von Odoo bereitgestellten Integrationsmechanismen erfordern eine quellcodebezogene Adaption bzw. Neuentwicklung der Systembestandteile und setzen entsprechende Programmierkenntnisse der Anwendungsexperten bzw. -entwickler voraus.

## 5.10 Entwicklung und Integration neuer Systemfunktionalität

Die Analyse der häufig auftretenden Anpassungsanforderungen weist auf die Adaptionen hin, die nur durch Entwicklung und nachträgliche Integration der benötigten Systemkomponenten realisiert werden können. Im Rahmen dieses Beispielszenarios wird untersucht, ob die von dem Systemhersteller bzw. -anbieter bereitgestellten Werkzeuge eine eigenständige Entwicklung unternehmensindividueller Systemerweiterungen ermöglichen.

Der Machbarkeitsstudie dieses Abschnitts wurde die Entwicklung einer Systemerweiterung zugrunde gelegt, mit deren Hilfe die Funktionalität eines Bonusverwaltungssystems im Systemrahmen bereitgestellt werden kann. Ableitend dem von Rose (2015) dargestellten Konzept eines Bonussystems, soll nach dem Erreichen eines bestimmten Planwertes der Nettoverkäufe eine Prämie auf das Gehalt des Verkäufers angerechnet werden. In diesem Kontext soll die Anwendung die entsprechende Flexibilität sicherstellen, individuelle Bonuspläne für unterschiedliche Mitarbeiter definieren zu können.

Das Beispielszenario integriert zwecks einer komplexen Betrachtung der systembezogenen Erweiterbarkeit die Entwicklungen, die auf der Ebene der Daten, der Anwendungslogik und der Benutzeroberfläche vorgenommen werden müssen. Die wichtigsten Komponenten, die in der Machbarkeitsstudie in jedem untersuchten ERP-System angelegt werden sollen, wurden in der Tabelle 5.3 zusammengestellt.

Die zu entwickelnden Bestandteile müssen mit den anderen Systemelementen entsprechend gekoppelt werden, um einen reibungslosen Datenaustausch sicherstellen zu können. Die Anwendung soll mindestens an zwei Stellen mit anderen Modulen gekoppelt werden – einerseits soll die Anwendung eine Auswahl der Mitarbeiter ermöglichen (Leseberechtigung für aktuelle Mitarbeiterdaten) und andererseits eine Aktualisierung der Gehaltsdaten zur Folge haben (Schreibberechtigung für die Gehaltsdaten).

Die Machbarkeitsstudien dieses Kapitels sollen die Fragen bezüglich der Erweiterbarkeit der cloudbasierten ERP-Systeme beantworten und die vom Systemhersteller bereitgestellten Konzepte bzw. Werkzeuge demonstrieren, mit deren Hilfe die Erweiterungen auf den einzelnen architektonischen Schichten im Rahmen des On-Demand-Selfservices von der Anwenderseite realisiert werden können.

**Tabelle 5.3:** Grundlegende Komponenten des Bonusverwaltung-Erweiterungsszenarios

| Komponente | Beschreibung |
|---|---|
| Bonusplan-Datenmodell | Datentabelle mit den typisierten Datenfeldern [Datentyp] zur Absicherung der erfassten Bonusplandaten:<br>Bonusplannummer [Zahl], Mitarbeiter [Text], Gültigkeit (ab / bis) [Datum], Planwert [Zahl], Bonus (%) [Zahl], Gesamtbonus [Zahl] |
| SQL-Abfragen | Select-, Insert-, Update-, Delete-Operationen am Bonusplan |
| Übersichtsseite | Tabellarische Darstellung der aktiven Bonuspläne |
| Details-Seite | Eingabeformular zur Erfassung der Bonusplandaten |
| Prozedur für die Bonus-Berechnung | Prozedur zur Ermittlung des aktuellen Bonuswertes anhand des Umsatzwertes des Mitarbeiters und der Bonusplandaten; generiert der Inhalt des Gesamtbonus-Feldes anhand der Formel:<br>`Bonuswert = Maximum{([∑ Umsatz] – [Planwert]) * [Bonus (%)]; 0}` |
| Prozedur für die Gehalt-Aktualisierung | Trigger-Prozedur zur Aktualisierung der Gehälter anhand der berechneten Bonuswerte, getriggert am Ende der Gültigkeitsperiode einzelner Bonuspläne |

### 5.10.1 System I: Business ByDesign

Die Erweiterungen für SAP Business ByDesign können mittels der Inanspruchnahme der dedizierten Entwicklungsumgebung (SAP Cloud Applications Studio IDE; SAP SE 2016b) vorgenommen werden. Die IDE stellt eine lokal installierbare Entwicklungsumgebung dar, die mithilfe des eingebauten Clients an die cloudbasierte Entwicklungsplattform (PaaS) gekoppelt ist. Dank dieser Kopplung können die lokal entwickelten Anwendungskomponenten sofort auf dem Anwendungsserver in der Cloud aktiviert und in das ERP-System integriert werden.

Mit der IDE hat der Entwickler die Möglichkeit, das bestehende System zu erweitern bzw. die bestehenden Systemkomponenten in einem bestimmten Umfang zu modifizieren. Es ist für ihn möglich, ausgehend von den Komponenten der Benutzeroberfläche, über die Geschäftsobjekte und Webdienste bis hin zu den vollständigen Geschäftsprozessen, diese selbständig zu entwickeln und danach im Rahmen des ERP-Systems zu aktivieren.

Die Erweiterungen des cloudbasierten ERP-Systems werden unter Wiederverwendung der bereitgestellten Bibliotheken und Entwurfsmuster in der systemnativen Skriptsprache entwickelt (vgl. SAP SE 2016b, S. 168). Die Wiederverwendbarkeit einzelner Oberflächenkomponenten stellt die systemadequate Ansicht und Funktionalität der eigenentwickelten Systembestandteile sicher und verspricht kürzere Entwicklungszeiten im Vergleich zur herkömmlichen Anwendungsentwicklung (vgl. Kurbel 2016, S. 320).

Im Rahmen dieser Machbarkeitsstudie wurde die IDE benutzt, um die bestehende Lösung um die Funktionalität eines Bonusverwaltungssystems zu erweitern. In diesem Zusammenhang wurde mithilfe der Entwicklungsumgebung ein neues Geschäftsobjekt – *Bonusplan* – entwickelt und im Systemrahmen aktiviert. Das Geschäftsobjekt definiert u. a. die Datenstruktur des Bonusplans bzw. der Tabelle, in der die Bonusplandaten gesichert werden. Ergänzend zu dem Geschäftsobjekt wurden die Oberflächenkomponenten – das Work-Center und die Systemseiten – angelegt. Folgend dem Design-Konzept von Business ByDesign stellt das Work-Center die Funktionen eines Unternehmensbereiches zusammen und wird als Eintrag im Systemmenü sichtbar (vgl. SAP SE 2016b, S. 63). Die Benutzeroberfläche der neu entwickelten Lösung bilden zwei Systemseiten, die in der Abbildung 5.28 dargestellt sind. Die Systemseiten wurden mithilfe der IDE unter Wiederverwendung der predefinierten Elemente der Benutzeroberfläche erstellt. Darauf aufbauend wurde in den einzelnen Steuerelementen die

Geschäftslogik in der Skriptsprache hinterlegt, welche die Funktionalität des neuen Moduls sicherstellt.

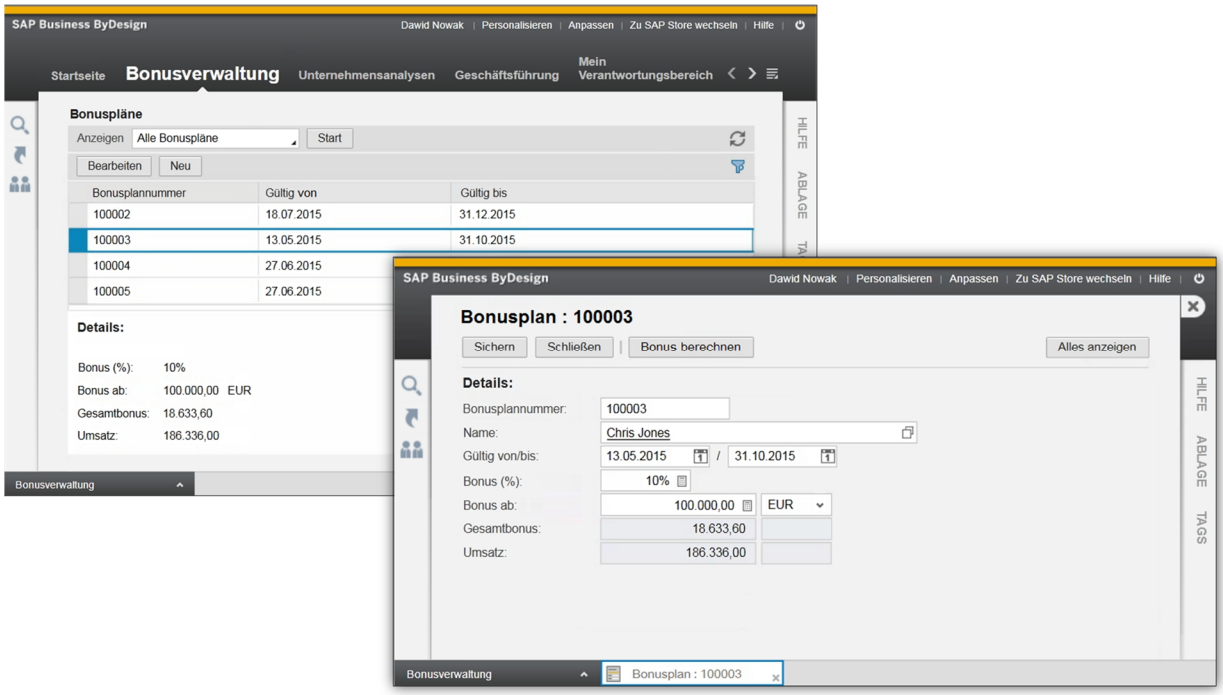





**Abbildung 5.28:** Eigenentwickelte Systemerweiterung integriert in Business ByDesign

Die Systemseite im Hintergrund beinhaltet eine Tabelle, die einen Überblick über die im System existierenden Bonuspläne bietet. Die tabellarische Übersicht wurde mit zwei Schaltflächen (*Bearbeiten* und *Neu*) erweitert, die entsprechend eine Bearbeitung der bestehenden bzw. Definition der neuen Bonuspläne erlauben. Wird eines der beiden Steuerelemente angeklickt, wird die untere Systemseite geöffnet, in der die einzelnen Attribute des Bonusplans erfasst bzw. bearbeitet werden können.

Die Definition eines neuen Bonusplanes beginnt in der Regel mit der Auswahl des Mitarbeiters, für den der Bonusplan vorgesehen ist. Zu diesem Zweck wurde in der Benutzeroberfläche eine Dropdownliste verankert, die die Mitarbeitertabelle referenziert. Anschließend werden die einzelnen Eingabefelder ausgefüllt, die entsprechend zur Erfassung der Gültigkeitsdaten, des prozentualen Bonusanteils am Umsatz und des Umsatzwertes, ab dem der Bonus angerechnet wird, vorgesehen sind. Die Felder *Bonusplannummer*, *Umsatz* und *Gesamtbonus* werden durch das System beim Sichern der Benutzereingaben automatisch ausgefüllt. Das *Umsatz-Feld* stellt den Gesamtwert aller Verkaufstransaktionen dar, die der ausgewählte Mitarbeiter in dem genannten Zeitraum abgeschlossen hat. Das *Gesamtbonus-Feld* beinhaltet die anhand der hinterlegten Formel berechnete Prämie (vgl. Tabelle 5.3), die aktuell an den Verkäufer auszuzahlen wäre.

Die Abbildung in Vordergrund stellt einen Bonusplan dar, der für einen Mitarbeiter (*Chris Jones*) angelegt wurde. Angesichts der erfassten Bonusplandaten erhält der Verkäufer eine Prämie in der Höhe von 10 Prozent des generierten Umsatzüberschusses, soweit seine Nettoverkäufe in dem Zeitraum 13.05-31.10.2015 den Umsatzplanwert von 100.000 EUR übersteigen. Der Verkaufsmanager kann zu jeder Zeit eine Simulation des aktuellen Bonuswertes durchführen. Nachdem die Schaltfläche *Bonus berechnen* angeklickt wird, werden die Umsätze des ausgewählten Verkäufers für den genannten Zeitraum aufsummiert (*Umsatz*) und der Bonus-

wert (*Gesamtbonus*) ermittelt. Die am Bonusplan vorgenommenen Änderungen können abgespeichert (*Sichern*) bzw. verworfen (*Schließen*) werden. Die abgesicherten Bonuspläne werden in der Tabelle der oberen Systemseite angezeigt und stellen eine Grundlage für die Prozedur der Prämienberechnung dar.

Jede Prämie wird automatisch durch das System anhand der Vorgabewerten und der individuellen Verkaufsdaten des jeweiligen Mitarbeiters kalkuliert. Die Berechnung des Bonuswertes findet erst am Ende des Gültigkeitszeitraums statt und wird durch die hinterlegte Trigger-Prozedur ausgeführt. Der berechnete Betrag bildet einen Aufschlag auf den Gehaltswert des Mitarbeiters, der in der Gehaltstabelle abgespeichert ist.

### 5.10.2 System II: Scopevisio

Das geplante Beispielszenario konnte wegen der mangelnden Erweiterbarkeit des cloudbasierten ERP-Systems von Scopevisio nicht umgesetzt werden.

Scopevisio erlaubt es nicht, eine Erweiterungsprogrammierung bzw. Modifikation der standardisierten Systembestandteile im Rahmen des On-Demand-Selfservices vorzunehmen. Gemäß der vom Systemhersteller gewählten Bereitstellungsstrategie ist er allein für Weiterentwicklung und Erweiterung der Systemfunktionalität zuständig. Dieser Strategie folgend stellt die Scopevisio AG seinen Kundenunternehmen keine an das ERP-System gekoppelte Entwicklungsumgebung bzw. -werkzeuge zur Verfügung, die eine anwenderseitige systemnative Entwicklung und nachträgliche Integration der benutzerdefinierten Systemkomponenten ermöglichen.

Die individuellen Erweiterungen können ausschließlich in Form der systemexternen Anwendungen entwickelt und gegebenenfalls mittels der von Scopevisio bereitgestellten API an das cloudbasierte ERP-System gekoppelt werden (vgl. Scopevisio AG 2016a). Die Integration ist allerdings ausschließlich auf der Anwendungsebene sichergestellt und genügt somit nicht den der Machbarkeitsstudie zugrunde gelegten Anforderungen.

### 5.10.3 System III: Odoo

Odoo stellt im Rahmen der eingebauten *Technischen Werkzeuge* die Editoren zur Verfügung, mit deren Hilfe die bestehenden Systemmodule erweitert bzw. individuelle Systemkomponenten eigenentwickelt werden können. Die unternehmensinternen Entwickler können unternehmensindividuelle Datenmodelle, die Geschäftslogik und die Elemente der Benutzeroberfläche eigenständig programmieren. Die Entwicklung der Logik-Komponenten findet in Python-Sprache statt, die Gestaltung der Systemseiten erfolgt dagegen mit HTML bzw. Java-Script.

Im Rahmen des Beispielszenarios wurde mit den bereitgestellten *Technischen Werkzeugen* das bestehende System um die Funktionalität eines Bonusverwaltungssystems erweitert. Das dem Szenario zugrunde liegende Konzept wurde unter Inanspruchnahme der im Entwicklermodus verfügbaren Werkzeuge umgesetzt, die zur Erstellung des Datenmodells, der Systemseiten und der erforderlichen Elementen der Geschäftslogik verwendet wurden.

Die Definition des durch die Fallstudie vorausgesetzten Datenmodells erfolgte unter Einsatz des integrierten Modell-Editors, mit dessen Hilfe die einzelnen *Bonusplan-Datenfelder* definiert und typisiert wurden. Aufgrund des direkten Bezuges zu der Mitarbeiterverwaltung und Gehaltsabrechnung wurde entschieden, die neu entwickelten Komponenten in das standardisierte *Mitarbeiter-Modul* zu integrieren. In diesem Zusammenhang wurde die vorhandene Menüleiste der Übersichtsseite um einen neuen Menüeintrag – *Bonusverwaltung* – erweitert. Darüber hinaus wurden unter Inanspruchnahme des integrierten View-Editors zwei Systemseiten angelegt, die entsprechend zur Darstellung einer tabellarischen Übersicht aller Bonus-

pläne und der detaillierten Ansicht eines Bonusplanes verwendet werden. Die angelegten Systemseiten sind in der Abbildung 5.29 dargestellt.

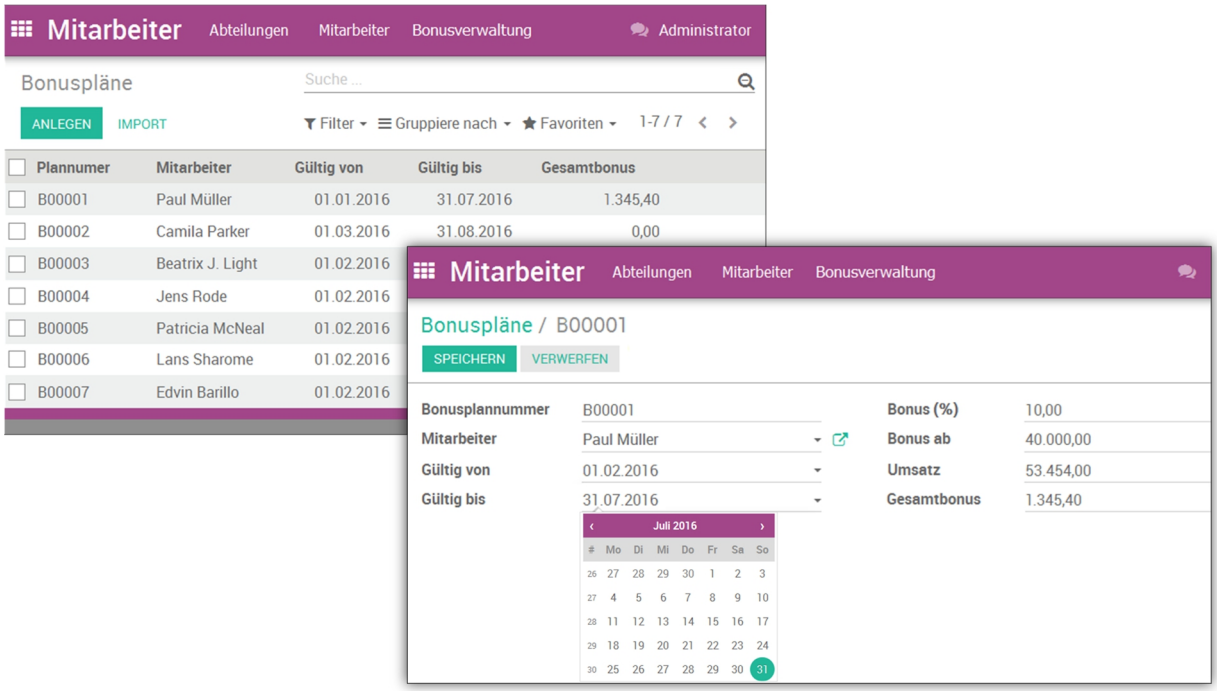





**Abbildung 5.29:** Eigenentwickelte Systemerweiterung integriert in Odoo

Die von den beiden Systemseiten wiederverwendeten predefinierten UI-Komponenten von Odoo stellen die systemkonforme Ansicht und Funktionalität sicher. Die Systemseite im Hintergrund stellt eine tabellarische Übersicht aller definierten Bonuspläne dar, der die allgemeinen Informationen über die einzelnen Bonusplanpositionen entnommen werden können. Sie enthält eine eindeutige Bonusplannummer, den Mitarbeiternamen und die Gültigkeitsdaten. Darüber hinaus zeigt die letzte Spalte den aktuellen Bonuswert des jeweiligen Mitarbeiters, der anhand seiner Verkaufsdaten durch das System berechnet wurde. Die angelegte Seite ermöglicht es, die angezeigten Bonuspläne anhand der Attribute zu filtern bzw. zu gruppieren sowie die Ansicht zu den Favoriten hinzuzufügen. Mithilfe der zwei Schaltflächen (*Anlegen* und *Import*) können die neuen Bonuspläne entsprechend manuell angelegt bzw. aus einer CSV-Datei eingelesen werden. Beim Anklicken der *Anlegen-Schaltfläche* bzw. eines der tabellarisch angezeigten Bonuspläne wird die *Bonusplan-Details-Seite* geöffnet (rechts), die eine Erfassung bzw. Bearbeitung der Bonusplandaten ermöglicht.

Die Abbildung im Vordergrund veranschaulicht den Erstellungsprozess eines neuen Bonusplans für den Mitarbeiter (*Paul Müller*). Im ersten Schritt wählt der Verkaufsmanager einen Mitarbeiter aus der Auswahlliste aus, für den der Bonusplan vorgesehen ist. Danach sollen die Gültigkeitsfelder ausgefüllt und der prozentuale Bonusanteil sowie die *Bonus-ab-Grenze* festgelegt werden. Die Felder *Bonusplannummer*, *Gesamtbonus* und *Umsatz* werden automatisch durch das System ausgefüllt und können vom Benutzer nicht editiert werden.

Erreicht der Mitarbeiter (*Paul Müller*) in dem festgelegten Zeitraum (*01.02-31.07.2016*) die Bonus-ab-Grenze (*40.000 EUR*), erhält er am Ende der Gültigkeitsperiode eine Prämie in Höhe von 10 Prozent des generierten Umsatzüberschusses. Anhand der Eingaben des Verkaufsmanagers und der aktuellen Verkaufsdaten führt das System eine Simulation des Bonuswertes durch, der im Feld Gesamtbonus ausgegeben wird. Der angezeigte Bonusplan kann per An-

klicken der im Kopf des Formulars verankerten Schaltflächen gesichert bzw. verworfen werden. Die gesicherten Bonuspläne werden in der Übersichtsseite tabellarisch angezeigt und stellen die Inputdaten für die Prozedur der Gehaltsabrechnung bereit, die automatisch durch das System ausgeführt wird.

## 5.11 Zusammenfassung der Erkenntnisse

Die durchgeführten Machbarkeitsstudien bestätigen die Bedeutung einer anwenderseitigen Adaptierbarkeit der cloudbasierten ERP-Systeme im Rahmen eines On-Demand-Selfservices.

Die drei untersuchten ERP-Systeme stellen unterschiedliche Flexibilität hinsichtlich der analysierten Systemelemente bereit. Die Adaptionen einzelner Systemelemente können in der Regel in unterschiedlichem Umfang und unter Inanspruchnahme differenzierter Customizing-Ansätze und Werkzeuge realisiert werden.

*Adaptierbarkeit der Systemelemente*

Die Validierung der systembezogenen Adaptierbarkeit erfolgte anhand der 27 Machbarkeitsstudien. Davon konnten zwanzig der getesteten Adaptionen (74,1 %) vollständig, vier (14,8 %) unvollständig und weitere drei (11,1 %) gar nicht umgesetzt werden. Die Ergebnisse der durchgeführten Machbarkeitsstudien hinsichtlich technischer Umsetzbarkeit der geplanten Systemadaptionen wurden in der Tabelle 5.4 zusammengestellt.

**Tabelle 5.4:** Zusammenstellung der Ergebnisse der Machbarkeitsstudien

| Szenario | Betroffene Systemelemente | Technische Machbarkeit | | |
|---|---|---|---|---|
| | | System I | System II | System III |
| Änderung der modularen Systemumfangs | SE1 | ✓ | ✓ | ✓ |
| Änderung der Parametereinstellungen | SE2 | ✓ | ✓ | ✓ |
| Adaption der Systemoberfläche | SE3 | ✓ | ✓ / – | ✓ |
| Erweiterung der Geschäftsobjekte und Dokumentenvorlagen | SE4 | ✓ | ✓ | ✓ |
| Erweiterung der Geschäftsanalytik | SE5 | ✓ | – | ✓ |
| Adaption der Aufbauorganisation | SE6 | ✓ | ✓ | ✓ |
| Adaption des Prozessablaufs | SE7 | ✓ / – | – | ✓ / – |
| Integration externer Webanwendung | SE8 | ✓ | ✓ / – | ✓ |
| Entwicklung und Integration neuer Systemfunktionalität | SE9 | ✓ | – | ✓ |

**Legende:** ✓ vollständig umgesetzt, ✓/– teilweise umgesetzt, – nicht umgesetzt

Die ‚✓'-Zeichen weisen auf eine vollständige Umsetzungsfähigkeit der geplanten Systemadaption im Rahmen eines On-Demand-Selfservices hin. Das Symbol ‚–' kennzeichnet die Fallstudien, welche im Systemrahmen nicht umgesetzt werden konnten. Die Kombination der

beiden Zeichen wird genutzt, um auf die Machbarkeitsstudien aufmerksam zu machen, die von einem Anwendungsexperten nicht in vollem Umfang realisiert werden können.

Die vorgenommenen Machbarkeitsstudien weisen auf eine weitreichende anwenderseitige Adaptierbarkeit und Erweiterbarkeit mehrerer Systemelemente hin. Im anwenderseitigen Customizing können differenzierte Adaptionen auf unterschiedlichen architektonischen Ebenen der cloudbasierten ERP-Systeme umgesetzt werden.

Die durchgeführten Machbarkeitsstudien bestätigen die Möglichkeit einer unternehmensindividuellen Systemgestaltung durch die bedarfsgerechte Auswahl und Zusammenstellung der modularen Systemkomponenten im Rahmen des anwenderseitigen Customizing. Die unternehmensindividuelle Gestaltung der Systemfunktionalität bzw. der Geschäftsabläufe kann eigenständig – durch das Setzen vordefinierten Parameter – aus einer Vielzahl im System vorhandener Funktionalitätsvarianten bzw. Ablaufoptionen festgelegt werden. Die Adaptionen der Aufbauorganisation kann mithilfe der integrierten Werkzeuge vorgenommen werden, welche in der Regel neben der Abbildung bzw. Aktualisierung der Organisationsstruktur auch die Rollen- und Rechteverwaltung umfassen.

Darüber hinaus wird im Rahmen eines On-Demand-Selfservices die Individualisierung der standardisierten Geschäftsobjekte sichergestellt, indem diese mit benutzerdefinierbaren Datenfeldern bzw. unternehmensspezifischen Skripten mit eigener Geschäftslogik erweitert werden können. In diesem Kontext wird in Abhängigkeit von der bereitgestellten Flexibilität der Datenebene neben der Definition eigener Datenfelder auch die Deklaration unternehmensspezifischer Datenstrukturen sowie individueller Datenmodelle gewährleistet. Die bereitgestellten Funktionen ermöglichen die eigenständige Individualisierung der Formulare und Belegvorlagen, welche die individuellen Datenobjekte referenzieren.

Die Gegenüberstellung der Ergebnisse der durchgeführten Machbarkeitsstudien verdeutlicht die relativ begrenzte Flexibilität der cloudbasierten ERP-Systeme hinsichtlich der Adaption der standardisierten Geschäftsabläufe. In zwei der drei Machbarkeitsstudien war die Umsetzung der geforderten Prozesserweiterungen nur bedingt mithilfe der bereitgestellten Customizing-Werkzeuge realisierbar. In den genannten Fällen erfolgte die Adaption auf der Ebene eines Geschäftsobjektes durch das Hinzufügen der erforderlichen Geschäftslogik in Form der konfigurierbaren Genehmigungsregeln bzw. der automatisierten Aktionen ohne direkte Adaption des standardisierten Prozessmodells.

Neben vielen Gemeinsamkeiten hinsichtlich der Flexibilität einzelner Systemelemente verdeutlichen die Ergebnisse der durchgeführten Machbarkeitsstudien auch einige systemabhängige Differenzen. Während bei der Adaption der Benutzeroberfläche, bzw. bei der Integration externer Webinhalte, ein gewisser Umfang der Umsetzbarkeit vom Systemhersteller sichergestellt wird, können aus den durchgeführten Fallstudien keine verallgemeinerbaren Erkenntnisse bezüglich der Adaptierbarkeit der Geschäftsanalytik und Erweiterbarkeit der Systemfunktionalität abgeleitet werden. Die zwei Untersuchungsgegenstände (Systeme I und III) stellen die weitreichende Adaptierbarkeit bzw. Erweiterbarkeit der analytischen Modelle sicher, welche die Definition der benutzerdefinierten Datenquellen, Kennzahlen und Berichte umfassen. Im Rahmen von System II können lediglich die vordefinierten Berichte benutzerkonform parametrisiert werden.

*Adaptionsansätze*

Neben der allgemeinen technischen Umsetzbarkeit einzelner Adaptionsanforderungen weisen die vorgenommenen Machbarkeitsstudien auf einige ansatzbezogene Gemeinsamkeiten und Differenzen zwischen den Untersuchungsgegenständen hin. Zum Zweck einer systemübergreifenden Betrachtung der anwendbaren Adaptionsansätze wurde eine dem Konzept von Sun

et al. (2008) angelehnte Gegenüberstellung der systembezogenen Customizing-Strategien vorgenommen, die in Abbildung 5.30 dargestellt ist.

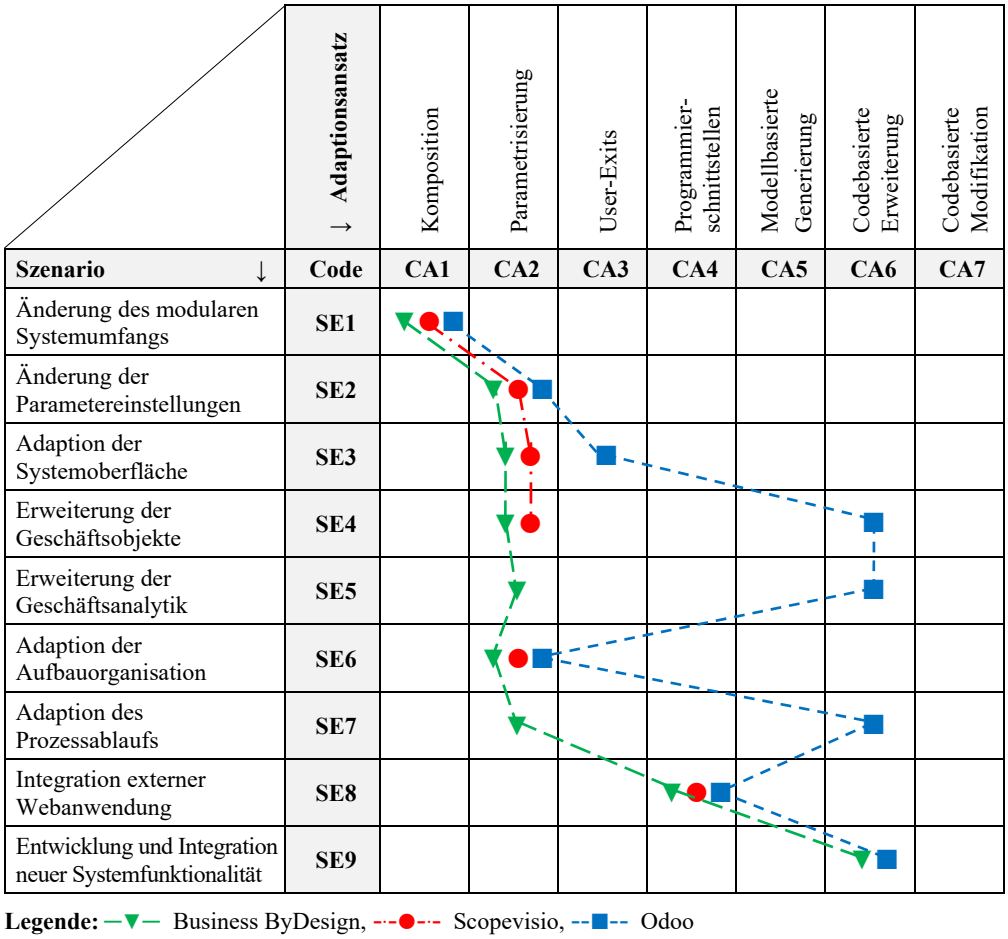



**Abbildung 5.30:** Systemindividuelle Adaptionsprofile gemäß den Ergebnissen der Machbarkeitsstudien

Die dargestellte Matrix listet die Machbarkeitsszenarien bzw. Adaptionsanforderungen mit den betroffenen Systemelementen vertikal und die Adaptionsansätze horizontal auf. Die einzelnen Punkte in der Matrix weisen auf die Adaptionsansätze hin, welche bei der Umsetzung der jeweiligen Machbarkeitsstudie die entscheidenden Rollen spielen. Die gestrichelten Linien verbinden die Punkte, die das jeweilige ERP-System beschreiben. Sie erzeugen zugleich das systemindividuelle Adaptionsprofil. Die dargestellten Profillinien verdeutlichen die Bedeutung der einzelnen Adaptionsansätze im Rahmen der implementierten Adaptionsstrategie des Systemherstellers und ermöglichen es, die systemübergreifenden Tendenzen bzw. Inkonsistenzen im Hinblick auf die Adaptierbarkeit einzelner Systemelemente zu identifizieren.

Ableitend von der Abbildung 5.30 stellt die Parametrisierung erwartungsgemäß den wichtigsten bzw. meist angewandten Adaptionsansatz dar. Gerade zwölf der geplanten Systemadaptionen konnten vollständig mittels Einstellung der system- bzw. funktionsbezogenen Parameter umgesetzt werden. Die Anwendbarkeit dieses Adaptionsansatzes erstreckt sich in Abhängigkeit von dem ERP-System über differenzierte Adaptionen der Daten-, Anwendungs- und Präsentationsebene.

Eine ebenso wichtige Rolle bei der Sicherstellung der Systemflexibilität im Cloud-Umfeld spielt der komponentenbasierte Ansatz, der aufbauend auf dem Prinzip der Wiederverwend-

barkeit, eine bedarfsgerechte Auswahl und Zusammenstellung der vordefinierten Komponenten voraussetzt. Dieser Customizing-Ansatz wird gemäß der Machbarkeitsstudien insbesondere zur Adaption der modularen Systemzusammenstellung, bzw. zur Sicherstellung der Flexibilität bei der Gestaltung unternehmensindividueller Oberflächenkompositionen eingesetzt.

Die Programmierschnittstellen stellen gemäß den Untersuchungsergebnissen eine weitere systemübergreifend bevorzugte Adaptionstechnik dar, die zur Sicherstellung der erforderlichen Integrationsfähigkeit mit anderen Anwendungen des Geschäftsumfelds angewendet wird. Die durch die analysierten ERP-Systeme bereitgestellten Schnittstellen stellen in der Regel mehrere Webdienste zur Verfügung, die neben den standardisierten Abfragen der systeminternen Datenbestände auch die Datenmanipulation gewährleisten. Auf Grundlage der lesenden und schreibenden API-Befehle wird ein bidirektionaler Datenaustauch sichergestellt, welcher einen Kernpunkt der Anwendungsintegration darstellt.

Ergänzend zu denjenigen Ansätzen, welche auf die Wiederverwendung der standardmäßig zur Verfügung gestellten Komponenten ausgerichtet sind, ermöglichen einige der analysierten Cloud-ERP-Systeme eine Erweiterung der Standardlösung durch eigenentwickelte Artefakte. Die codebasierte Systemerweiterung stellte einen häufig im Rahmen der Fallstudienbearbeitung verwendeten Customizing-Ansatz dar, mit welchem die Erweiterung differenzierter Systemaspekte auf unterschiedlichen Systemebenen vorgenommen wurde. In Abhängigkeit vom Untersuchungsgegenstand und dem entwickelten Objekt erfolgte die Systemindividualisierung in einer Skript- bzw. Programmiersprache.

Die vordefinierten User-Exits bzw. Erweiterungspunkte stellen einen anderen Customizing-Ansatz dar, welcher zur Individualisierung der cloudbasierten ERP-Systeme im Rahmen eines On-Demand-Selfservices angewendet wird. Über die bereitgestellten User-Exits können die eigenentwickelten Skripte angebunden werden, welche die vordefinierten Systemelemente mit unternehmensspezifischer Funktionalität anreichern. Die durchgeführten Machbarkeitsstudien bestätigen den Einsatz des auf User-Exits aufbauenden Ansatzes, insbesondere zur Definition individueller Felder mit eigener Logik, dedizierter Kennzahlen und der Verankerung externer Inhalte in der Benutzeroberfläche.

Ableitend von den Ergebnissen der Machbarkeitsstudien finden die Ansätze, die eine Modifikation des Systemquellcodes bzw. des dem Quellcode zugrunde gelegten Modells erfordern, keinen Einsatz beim Customizing der cloudbasierten ERP-Systeme. Diese Erkenntnisse bestätigen somit die Tendenz der Systemhersteller zur Sicherstellung einer einheitlichen Codebasis.

Im Hinblick auf das Customizing einzelner Systemelemente verdeutlicht die Gegenüberstellung neben vielen ansatzbezogenen Ähnlichkeiten auch einige Unterschiede zwischen den Untersuchungsgegenständen. Während die Adaption der Benutzeroberfläche in Scopevisio mittels der Parametrisierung erfolgt, erfordert die Durchführung derselben Adaption in Odoo in der Regel eine Inanspruchnahme der User-Exits und eine codebasierte Erweiterung der Benutzeroberfläche. Ähnliche Unterschiede hinsichtlich der anwendbaren Adaptionstechniken wurden bei der Erweiterung der Geschäftsanalytik sowie der Änderung des Prozessablaufs identifiziert, welche entsprechend in Business ByDesign mittels Parametrisierung der vorhandenen Funktionalität und in Odoo mithilfe der User-Exits bzw. Eigenentwicklung implementiert werden konnten.

Eine genauere Analyse der Profillinien ermöglicht es, auch die systemübergreifenden Tendenzen in Adaptionsansätzen aufzuzeigen. Es ist anzumerken, dass die Profillinien von Scopevisio und Business ByDesign im großen Umfang konvergent sind. Aufgrund dessen, dass die beiden ERP-Systeme im Gegensatz zu Odoo cloudnative ERP-Systeme (IV-Reifegrad) sind, stellt dies eine Prämisse für die Existenz reifebezogener Unterschiede im Einsatz einzelner Customizing-

Ansätze (siehe Hypothese H1) dar. Die statistische Validierung dieser Hypothese wird im Rahmen des nachfolgenden Kapitels vorgenommen.

*Adaptionswerkzeuge*

Die meisten Adaptionen in den durchgeführten Machbarkeitsstudien konnten mithilfe der systeminternen Adaptionswerkzeuge vorgenommen werden. Diese Werkzeuge stellen einen integralen Bestandteil der cloudbasierten ERP-Systeme dar und sind für die anwenderseitigen Administratoren über die Systemoberfläche zugänglich. In Abhängigkeit von dem anzupassenden Systemelement werden in der Regel differenzierte Werkzeuge bereitgestellt, mit deren Hilfe Adaptionen der Benutzeroberfläche, Erweiterungen des Datenmodells und der Anwendungslogik umgesetzt werden können. Die eingebauten Werkzeuge folgen in der Regel dem Konzept eines metadatengetriebenen Customizing. Die implementierten Systemänderungen werden als mandantenspezifische Metadaten gesichert und in der Systemlaufzeit mittels eines Interpreters automatisch eingelesen (vgl. Kang et al. 2011).

Die durchgeführten Machbarkeitsstudien weisen auf die weitgehende Tendenz der Systemanbieter zu einer Vereinfachung der anwenderseitigen Customizing-Prozesse durch die Integration unterstützender Werkzeugkonzepte hin. Zur Unterstützung der Anwender bei der Durchführung verschiedener Adaptionsprozesse wurden in den durchgeführten Fallstudien vor allem vier Mechanismen ersichtlich:

- *Assistenten*: stellen eine geführte Abfolge von Dialogen dar, mittels derer die benötigten Informationen von dem Anwendungsexperten erfragt werden. Die einzelnen Dialogfenster werden in der Regel mit Hinweisboxen und Hilfstexten versehen, die das Verständnis während des Konfigurationsprozesses erleichtern. Anhand der erteilten Informationen wird die Parametrisierung automatisch durch das Werkzeug im Hintergrund vorgenommen.
- *Visuelle Editoren*: stellen eine Benutzeroberfläche dar, in welcher die Adaption der ausgewählten Systemelemente intuitiv mittels Drag-and-Drop-Funktionalitäten und Auswahl der predefinierten Einstellungen erfolgt. Die analysierten Editoren folgen einem WYSIWYG-Konzept (engl. What You See Is What You Get), das sicherstellt, dass der Anwendungsexperte in der Entwurfsansicht das Endergebnis des angepassten Elements einschätzen kann.
- *Konfigurationsformulare*: stellen die dem zu bearbeitenden Systemelement zugehörigen Einstellungsoptionen zusammen. Die Parametrisierung erfolgt durch Auswahl der vordefinierten Optionen, bzw. durch Eingabe der benutzerdefinierten Werte in die Pflichtfelder. Die einzelnen Einstellungsoptionen werden mit den Hilfstexten bzw. einer Kontexthilfe versehen, welche die Einstellung erleichtern.
- *Code-Editoren*: stellen einen webbasierten integrierten Text-Editor zum Editieren des Quellcodes eines Systemelementes dar. Die Eingabe erfolgt in Abhängigkeit von dem System in einer herkömmlichen oder proprietären Programmiersprache. Die analysierten Editoren werden mit unterstützenden Mechanismen ausgestattet, welche die Benutzereingaben beim Speichern validieren und die Fehleingaben zurückweisen.

Bei einem systemübergreifenden Vergleich der integrierten Adaptionswerkzeuge bleibt bemerkenswert, dass bei den ausgereiften ERPaaS die Adaptionen überwiegend mithilfe der Assistenten, Editoren und Konfigurationsformulare durchgeführt werden können, so dass die codebasierten Adaptionen in der Regel nicht notwendig sind. Im Vergleich dazu werden in Odoo grundsätzlich codebasierte Adaptionen mittels Code-Editoren vorgenommen.

Neben den meist eingesetzten systeminternen Customizing-Tools wurden in der Fallstudienbearbeitung auch systemexterne, lokal installierbare Customizing-Maßnahmen verwendet. Die

systemexternen Werkzeuge werden von den Systemanbietern bereitgestellt, um die Limitierungen einer rein Browser-basierten Adaption zu umgehen.

Die im Fall von Business ByDesign bereitgestellte Entwicklungsumgebung stellt ein Beispiel für eine gezielte Erweiterung des anwenderseitigen Customizing-Rahmens dar. Mittels der bereitgestellten Entwicklungsumgebung können die Entwickler nicht nur die bestehenden Systemelementen (z. B. Geschäftsobjekte, Funktionalitäten etc.) anpassen, sondern auch neue Komponenten selbstständig entwickeln und das System entsprechend den individuellen Anforderungen des Anwenderunternehmens erweitern.

Die Strategie der Systemanbieter hinsichtlich der Bereitstellung von systemexternen Softwareentwicklungstools für seine Kunden ist jedoch nicht einheitlich. Während im Fall des PaaS-basierten ERP-Systems von SAP eine Entwicklungsumgebung durch Kunden erworben werden kann, stellen zwei andere Systemhersteller grundsätzlich keine Entwicklungstools bereit.

Neben den zwei genannten Werkzeugkonzepten werden von den Systemanbietern auch die Systemadaptionen und -erweiterungen in Form einer Dienstleistung angeboten. Die Strategie bezüglich der Bereitstellung von Systemadaptionen kann jedoch von System zu System variieren. Während im Fall von Scopevisio die anbieterseitigen Adaptionen bzw. Erweiterungen ausschließlich seitens des Systemherstellers vorgenommen werden, können in zwei anderen ERP-Systemen die Systemerweiterungen auch von Drittanbietern bereitgestellt werden.

Eine innovative Methode für die Bereitstellung der Systemerweiterungen im Bereich der cloudbasierten ERP-Standardanwendungssoftware stellen die internetbasierten Vertriebsplattformen (oft als Anwendungs-Stores bezeichnet) dar. Die Stores folgen dem Geschäftskonzept, welches aus der Vermarktung der Anwendungen für Mobilgeräte bereits bekannt ist. Die anwenderseitigen Administratoren können auf dem digitalen Marktplatz nach benötigten Systemerweiterungen suchen, die Kompatibilitätstests mit dem Kernsystem durchführen und Erweiterungslösungen kaufen. Die erworbene Erweiterung stellt nach einem Integrationsvorgang mit dem Kernsystem die zusätzliche Funktionalität sicher. Die digitalen Marktplätze stellen somit neben einer Entwicklungsumgebung ein weiteres Konzept für die anwenderseitige Systemerweiterung dar, die im Rahmen eines On-Demand-Selfservices, ohne Involvierung des Systemanbieters, erfolgen kann.

Die vorgenommene Gegenüberstellung der differenzierten Customizing-Konzepte weist einerseits auf eine weitgehende Tendenz zur Vereinfachung des anwenderseitigen Customizing hin, andererseits bestätigt sie auch eine wesentliche Rolle des PaaS auf dem Weg zur Erweiterung des Customizing-Rahmens im Cloud-Umfeld. Trotz zahlreicher Differenzen in der Bereitstellung einzelner Werkzeugkonzepte zwischen den analysierten cloudbasierten ERP-Systemen sind die Customizing-Strategien der cloudnativen ERP-Systeme durch ein größeres Spektrum der bereitgestellten Adaptionskonzepte gekennzeichnet.

Im nachfolgenden Kapitel wird eine empirische Untersuchung anhand einer größeren Anzahl cloudbasierter ERP-Systeme vorgenommen, welche die bisherigen Erkenntnisse hinterfragt und systematisiert.

# 6 Empirische Studie und Datenauswertung

Dem angenommenen Untersuchungskonzept (siehe Kapitel 1.3) folgend, widmet sich dieser Abschnitt einer empirischen Untersuchung, die zur Verifikation der aufgestellten Hypothesen (siehe Kapitel 4.3) und zur Ermittlung potenzieller Zusammenhänge zwischen der Systemarchitektur und den Aspekten der Systemadaptierbarkeit vorgenommen wurde.

Im Hinblick auf die zugrunde gelegten Architekturkonzepte der Untersuchungsgegenstände, welche sich in der Realisierung einzelner Eigenschaften einer dem Cloud-Bereitstellungsmodell ausgesetzten Anwendung widerspiegeln (siehe Kapitel 3.3), liefern die nachstehenden Unterabschnitte eine vergleichende Analyse der Customizing-Strategien mehrerer cloudbasierter ERP-Systeme.

Durch die Auswertung und Systematisierung der gewonnenen Erkenntnisse schafft das vorliegende Kapitel einen Bezugsrahmen für den architekturbezogenen Vergleich der Adaptierbarkeit der cloudbasierten ERP-Systeme.

## 6.1 Empirische Untersuchung

Zur Beantwortung der aufgestellten Forschungsfragen wurde im Rahmen des vorliegenden Dissertationsprojektes eine quantitative empirische Forschung durchgeführt, die auf einer strukturierten Umfrage als Datenerhebungsmethode aufbaute. Die Auswahl des quantitativen Forschungsansatzes ist dadurch begründet, dass die Untersuchung die Zusammenhänge zwischen der Customizing-Strategie und dem architektonischen Konzept der cloudbasierten ERP-Systeme untersucht und das Vorhandensein eines allgemeingültigen Relationsmusters verifizieren soll. Die statistische Auswertung der quantitativen Umfragedaten ermöglicht es zusätzlich, eine Validierung bzw. Verallgemeinerung der Erkenntnisse der qualitativen Untersuchung des vorigen Kapitels vorzunehmen, wodurch die Aussagekraft des Gesamtergebnisses zusätzlich erhöht werden kann.

Die nachfolgenden Unterabschnitte widmen sich der Charakteristika der vorgenommenen empirischen Untersuchung.

### 6.1.1 Umfragestruktur und Umfragewerkzeuge

Die Umfrageergebnisse sollen eine Grundlage für die Signifikanztests der aufgestellten Hypothesen schaffen. Um das gewährleisten zu können, wurden die einzelnen Fragen anhand der Ergebnisse der Literaturauswertung (siehe Abschnitte 4.3.1 – 4.3.3) erstellt und während der Testphase auf Vollständigkeit und Eindeutigkeit überprüft bzw. überarbeitet. Die Tabelle 6.1 gibt einen Überblick über die endgültige Struktur des Fragebogens, welcher in der finalen Umfrage eingesetzt wurde. Die einzelnen Spalten beinhalten entsprechend die Fragegruppe, die Fragenummer, die Fragestellung und den Frage-Code, der im Rahmen der Beschreibung der Untersuchungsergebnisse als Referenz verwendet wird. Für die vollständige Darstellung des Fragebogens wird hier auf Anhang C.1 verwiesen.

**Tabelle 6.1:** Umfragestruktur

| Fragegruppe | Nr. | Fragestellung | Frage-Code |
|---|---|---|---|
| Systemarchitektur | 1 | In welchen Betreibermodellen wird das ERP-System angeboten? | FSA01 |
| | 2 | In welchem Cloud-Bereitstellungsmodell wird das ERP-System primär angeboten? | FSA02 |
| | 3 | In welchen anderen Cloud-Bereitstellungsmodellen wird das cloudbasierte ERP-System angeboten? | FSA03 |
| | 4 | Stellt das genannte ERP-System eine vollständig cloudnative Anwendung dar? | FSA04 |
| | 5 | Welche Merkmale einer cloudnativen Anwendung weist die Architektur des genannten ERP-Systems auf? | FSA05 |
| | 6 | Wenn das ERP-System mandantenfähig ist; welches Konzept zur Sicherstellung der Mandantenfähigkeit wurde auf der Ebene des Anwendungsservers angewendet? | FSA06 |
| | 7 | Wenn das ERP-System mandantenfähig ist; welches Konzept zur Sicherstellung der Mandantenfähigkeit wurde auf der Ebene des Datenbankservers angewendet? | FSA07 |
| Customizing-Ansätze | 8 | Welche der aufgelisteten Customizing-Ansätze stellen einen integralen Bestandteil der Customizing-Strategie des Systemanbieters dar? | FCA01 |
| | 9 | Welche Rolle spielen die folgenden Customizing-Ansätze im Rahmen der gesamten Customizing-Strategie des Systemanbieters? | FCA02 |
| Adaptierbare Systemelemente | 10 | In welchem Umfang können die einzelnen Systemschichten adaptiert werden? | FSE01 |
| | 11 | In welchem Umfang können die folgenden Systemelemente von einem Anwendungs-experten angepasst werden? | FSE02 |
| Customizing-Werkzeuge | 12 | Welche der folgenden Werkzeug-Kategorien werden von dem Systemanbieter im Rahmen der Customizing-Strategie bereitgestellt? | FCW01 |
| | 13 | Welche Rolle für die anwenderseitigen Customizing-Prozesse spielen die folgenden Werkzeug-Kategorien? | FCW02 |
| | 14 | Welche Mechanismen und Konzepte zur Unterstützung der anwenderseitigen Adaptionen der folgenden Systemelemente werden in den bereitgestellten Werkzeugen verankert? | FCW03 |

Der Hauptteil des Fragebogens bestand aus vierzehn Fragen. Die Fragen wurden thematisch in vier Gruppen gegliedert. Diese sind: Systemarchitektur, Customizing-Ansätze, Adaptierbare Systemelemente und Customizing-Werkzeuge.

Die Gruppe aus Fragen zur Systemarchitektur beinhaltete allgemeine, architekturbezogene Fragen, die u. a. zur Herleitung eines SaaS-Reifegrads der analysierten ERP-Systeme benutzt wurden. Dabei konnten von den Befragten die relevanten, qualitativen bzw. nominalen Systemmerkmale ausgewählt werden. Wenn keine der vordefinierten Auswahloptionen zutreffend war, konnten eigene Merkmale eingegeben werden.

Bei den Fragen 9-11 und 13 wurden anhand einer fünfwertigen Likert-Skala (vgl. Boslaugh 2012, S. 145 f.) die Einstellung des Befragten bezüglich der vom Systemhersteller eingesetzten Customizing-Konzepte bzw. -Ansätze ermittelt. Eine Antwort konnte einen Wert zwischen 1 (gar nicht) und 5 (vollständig) annehmen. Die Beschränkung der Anzahl von offenen Fragen hatte, wie die Pilotstudie zeigte, eine positive Auswirkung auf die Rücklaufzeit des Fragebogens und ermöglichte eine einfachere Auswertung der Umfrageergebnisse. Ähnliche positive Auswirkungen wurden von anderen Autoren dokumentiert (vgl. Fan und Yan 2010, S. 133; Galešic und Bosnjak 2009).

Die oben erwähnte Studie dauerte vom 16. September 2015 bis zum 15. November 2015. Sie wurde durchgeführt, um die Verständlichkeit der Fragen zu überprüfen und erste Ergebnisse

für die Analysen zu gewinnen. Die Einladungen für die Testumfrage wurden an die ausgewählten Fachleute gesendet, welche sich mit dem SaaS-Thema beschäftigen. Die Umfragestruktur, die Art der Fragen, das Frageverhalten sowie die vorgegebenen Antwortmöglichkeiten wurden mit potenziellen Respondenten (u. a. ERP-Systemherstellern, Systemanbietern, Systemberatern und -entwicklern) während einer der ERP-Fachkonferenzen in Österreich besprochen. Die gewonnenen Anregungen und Hinweise führten zur Reduzierung der Anzahl der offenen Fragen, zur Paraphrasierung von einigen Fragestellungen und zur kompakteren Gestaltung der gesamten Umfrage mittels einer erneuten Aufteilung in Fragengruppen sowie einer Definition der Umfragelogik. Die Umfragelogik sorgte für eine automatische und zielgerichtete Anpassung des Fragebogens, in dem die nachfolgenden Fragen in Abhängigkeit von den gerade erfassten Antworten des Befragten ein- bzw. ausgeblendet werden. Bei der Umfrage sind z. B. die Fragen bezüglich der Mandantenfähigkeit auf der Applikations- bzw. Datenebene (Fragen 6 und 7) nur dann sichtbar, wenn bei der Frage nach den Systemeigenschaften (Frage 5) das System als mandantenfähig eingestuft wurde.

Die finale Befragung wurde ausschließlich online durchgeführt. Die webbasierte Umfrage wurde mithilfe der freien Online-Umfrage-Software LimeSurvey (LimeSurvey 2015) erstellt, welche auf einem Webserver installiert wurde. Im Vergleich zu anderer, meist kommerzieller Software bietet LimeSurvey weitere Features an, wie: multilinguale Umfragen, eine frei definierbare Umfrage-Logik und einen direkten Export nach SPSS bzw. Excel (vgl. LimeSurvey 2017), welche als entscheidende Auswahlkriterien galten. Die Datenerfassung startete am 15. Januar 2016 und dauerte 60 Tage.

Die Datenanalyse und statistischen Auswertungen wurden mithilfe der SPSS-Software von IBM (IBM 2017) durchgeführt. SPSS ist ein professionelles Software-Paket, welches alle für die Analyse der Umfragedaten erforderlichen statistischen Werkzeuge bereitstellen konnte. Die Ergebnisse der durchgeführten Pilotstudie wurden auf einer ERP-Fachkonferenz präsentiert (vgl. Nowak und Kurbel 2017).

### 6.1.2 Charakteristika der Respondentengruppe

Um möglichst relevante und objektive Umfrageergebnisse zu erzielen, wurde von den Befragten erwartet, ein Verständnis für das Systemarchitekturkonzept zu haben sowie über einen detaillierten Überblick über alle in bzw. mit dem System angebotenen Anpassungswerkzeuge und Customizing-Möglichkeiten zu verfügen. Zur Erfüllung dieser Voraussetzung richtete sich die Umfrage ausschließlich an die Systemanbieterseite.

Der erste Schritt zur Durchführung der Umfrage bestand in der Ermittlung von relevanten Systemanbietern, die ihre ERP-Systeme als cloudbasierte Dienste anbieten. Dies wurde anhand einer Analyse der auf die ERP-Branche spezialisierten Fachzeitschriften (z. B. Scavo et al. 2012; Eggert et al. 2013, S. 25), Marktübersichten und Berichten (z. B. Keckeis und Weiss 2014; Hestermann et al. 2014; Hamerman et al. 2015; Business-Software 2015) sowie Ratings und Statistiken (z. B. Software Advice 2015; Graphiq 2015) vorgenommen. Basierend auf der durchgeführten Marktanalyse wurde eine Grundgesamtheit von 42 potenziellen Systemanbietern ermittelt, die insgesamt 47 cloudbasierte ERP-Systeme im Angebot haben.

Aufgrund des Untersuchungscharakters der Studie wurden als potenzielle Respondenten die Personen ausgewählt, die entsprechende Erfahrung in der Anpassung der einzelnen Systeme besitzen. Die Umfrage richtete sich in erster Linie an Berater, welche für die Prozesse der Systemimplementierung bzw. -instandhaltung zuständig sind, sowie an Systementwickler bzw. Erweiterungsprogrammierer und Produktmanager. Die Experten auf diesen Gebieten wurden bevorzugt, da sie über ein fundiertes und komplexes System-Know-how verfügen und sich mit der Anpassung bzw. Erweiterung des ERP-Systems im Alltag beschäftigen (vgl. Pfadenhauer 2009, S. 452).

Um das für die Beantwortung der Umfrage erforderliche systembezogene Fachwissen der Befragten sicherstellen zu können, wurde eine mindestens fünfjährige Erfahrung in dem jeweiligen Gebiet vorausgesetzt.

Um mit den einzelnen Personen Kontakt aufzunehmen, wurden das soziale Netzwerk für Berufstätige – XING (XING 2017) – verwendet sowie die auf der Webseite des Systemherstellers bzw. seiner Partner erhältlichen E-Mail-Adressen und Kontaktformulare. Die XING-Plattform stellt dem registrierten Premium-Benutzer eine erweiterte Suchfunktionalität zur Verfügung, die eine zielgerichtete Auswahl der Mitgliederprofile u. a. nach Arbeitgeber, Arbeitsstelle und Erfahrung ermöglicht.

Auf Basis der XING-bezogenen Suche und der direkten E-Mail-Anfragen wurden 127 potenzielle Respondenten ermittelt, die sich mit der Implementierung bzw. mit der Adaption der ausgewählten, cloudbasierten ERP-Systeme beschäftigen. Die Einladungen zur Teilnahme an der Befragung wurden einzeln per E-Mail bzw. mithilfe des in die soziale Plattform integrierten Nachrichtdienstes versendet.

Die Umfrage wurde anonym durchgeführt, worüber die Befragten in der Einladungsnachricht bzw. auf der Umfrageseite informiert wurden. Die Anonymität der Umfragedaten wurde sichergestellt, um bei den Befragten eine höhere Bereitschaft zu einer offenen und zuverlässigen Evaluierung der bereitgestellten Customizing-Strategien und -Konzepte zu ermöglichen.

### 6.1.3 Charakteristika der Untersuchungsstichprobe

Die Phase der online durchgeführten Datenerhebung dauerte zwei Monate. In dieser Zeit gab es insgesamt 43 Respondenten. Dabei wurde der Fragebogen in 31 Fällen vollständig und in den anderen zwölf Fällen teilweise ausgefüllt. Da für die weitere Analyse eine Ermittlung des systembezogenen SaaS-Reifegrades erforderlich war, mussten aufgrund der fehlenden Angaben die unvollständigen Datensätze aus der weiteren Analyse ausgeschlossen werden. Die Rücklaufquote betrug somit 24,4%. Die Datenerhebung endete mit insgesamt 31 Datensätzen über 31 unterschiedliche cloudbasierte ERP-Systeme und bildete eine Grundlage für die statistische Auswertung.

*Service-Betreibermodelle*

Hinsichtlich der unterstützten Betreibermodelle werden alle in der Studie genannten ERP-Systeme (n=31) als cloudbasierte Anwendungen eingestuft, die im Rahmen eines SaaS-Modells bereitgestellt werden können. Die detaillierte Charakteristik der analysierten Stichprobe im Hinblick auf die unterstützten Service-Betreibermodelle ist in Abbildung 6.1 (a) dargestellt.

81,6% (n=25) der in der Umfrage genannten Cloud-ERP-Systeme werden ausschließlich im SaaS-Betreibermodell bereitgestellt. Weitere 19,4% (n=6) werden als PaaS-gekoppelte Systeme angeboten.

Von den 31 cloudbasierten ERP-Systemen stellen 29% (n=9) cloudnative Systeme dar, die von Anfang an gemäß der Voraussetzungen des SaaS-Modells entwickelt worden sind. Die weiteren 71% (n=22) der analysierten ERP-Systeme stellen cloudfähige Lösungen dar, die neben dem SaaS-Modell auch als konventionelle, auf der Anwenderseite installierbare On-Premise-Software betrieben werden.

Diese Dualität der Bereitstellungskonzepte resultiert daraus, dass viele Systemhersteller ihre ursprünglich für das On-Premise-Betreibermodell konzipierten ERP-Systeme für den cloudbasierten Betrieb zwischenzeitlich angepasst haben. Über die Tendenz zur Adaption der On-Premise-Systeme zum Cloud-Modell wurde auch von Bezemer und Zaidman (2010b, S. 88), Scavo et al. (2012, S. 3) und Johansson et al. (2014, S. 2) berichtet.

*Cloud-Bereitstellungsmodelle*

Die Umfrageteilnehmer hatten auch die durch ihre ERP-Systeme unterstützten Cloud-Bereitstellungsmodelle angegeben. In Abbildung 6.1 (b) ist die Charakteristika der analysierten Stichprobe hinsichtlich der unterstützten Cloud-Bereitstellungsmodelle dargestellt.

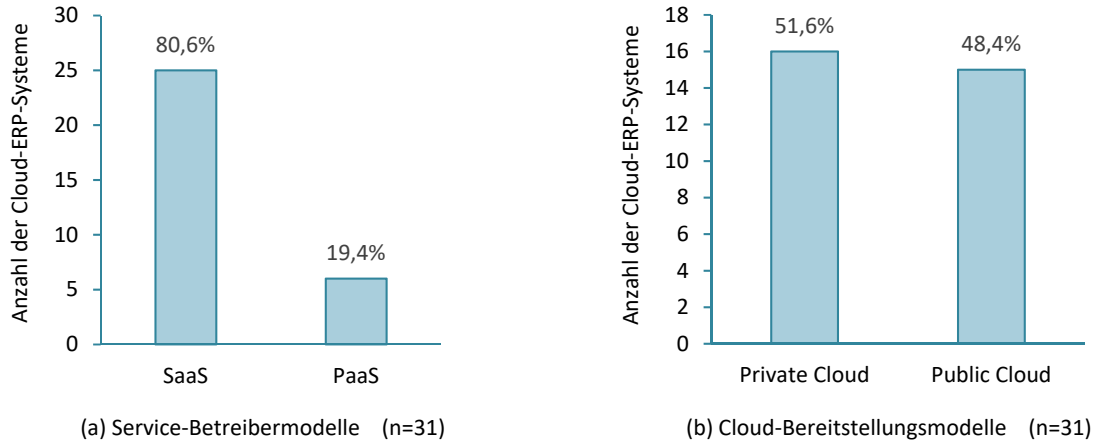



**Abbildung 6.1:** Charakteristika der untersuchten Stichprobe – die unterstützten Service-Betreibermodelle (a) und Cloud-Bereitstellungsmodelle (b)

Insgesamt 48,4% (n=15) der Systeme wurden von den Respondenten als Public-Cloudfähig eingestuft. In diesem Cloud-Modell wird eine von dem Drittanbieter gehostete Instanz des ERP-Systems von mehreren Unternehmen gleichzeitig verwendet, welche die Infrastrukturressourcen gemeinsam nutzen. Dabei werden 19,4% (n=6) der Systeme von ihren Anbietern ausschließlich in einer Public-Cloud betrieben. Weitere 29% (n=9) bieten die Public-Cloud als eine der möglichen Betreibermodelle an.

Von den Respondenten erklärten 51,6% (n=16), dass ihr System ausschließlich Private-Cloudfähig ist. In diesem Cloud-Modell wird das ERP-System als dedizierter Service einem Anwenderunternehmen zur Verfügung gestellt.

*SaaS-Reifegrad der ERP-Systeme*

Einen wichtigen Aspekt für die bevorstehende Analyse stellt der Reifegrad des cloudbasierten ERP-Systems dar (siehe Kapitel 3.3). Basierend auf den in der Phase der Datenerhebung gesammelten Daten, wurde der SaaS-Reifegrad für alle in der Studie ermittelten cloudbasierten ERP-Systeme hergeleitet. In der Datenanalyse wurde das Ausmaß berücksichtigt, in welchem das On-Demand-ERP-System die Voraussetzungen des SaaS-Reifegradmodells realisiert (siehe Kapitel 4.1.4).

Die Bestimmung des systembezogenen Reifegrads wurde anhand der Antworten der Respondenten hinsichtlich der Fragen (FSA01-7) der ersten Fragengruppe vorgenommen. Die Ergebnisse der vorgenommenen Analyse wurden in der Tabelle 6.2 zusammengefasst. Die Tabelle stellt die Häufigkeiten sowie die prozentualen Anteile der jeweiligen SaaS-Reife in der Untersuchungsstichprobe dar.

Bei 22,6% (n=7) der analysierten Systeme wurde der höchste, also der vierte Reifegrad ermittelt. Lediglich 12,9% (n=4) der Systeme wurden der Gruppe zugeordnet, welche durch den dritten SaaS-Reifegrad gekennzeichnet ist. Die Mehrheit der Systeme (51,6%, n=16) wurde auf der zweiten SaaS-Reife positioniert, während die erste SaaS-Reife bei 12,9% (n=4) der Systeme festgestellt wurde.

**Tabelle 6.2:** Die analysierten ERPaaS gegliedert nach dem SaaS-Reifegrad (n=31)

| SaaS-Reife | Anzahl | Anteil (%) | Anteil kumulativ (%) |
|---|---|---|---|
| **I** | 4 | 12,9 | 12,9 |
| **II** | 16 | 51,6 | 64,5 |
| **III** | 4 | 12,9 | 77,4 |
| **IV** | 7 | 22,6 | 100,0 |

## 6.2 Bereitgestellte Customizing-Ansätze

Den statistischen Auswertungen ging ein Test auf die Normalverteilung voraus. Aufgrund der ordinalen Natur der Likert-Skala sind statistisch signifikante Abweichungen von der Normalverteilung in der analysierten Datenmenge zu erwarten (vgl. Boslaugh 2012, S. 146). Unter Berücksichtigung der kleinen Datenmenge wurde der Shapiro-Wilk-Test auf die Normalverteilung angewendet, der im Vergleich zum Kolmogorov-Smirnov-Test bei kleinen Stichproben genauere Ergebnisse liefert (vgl. Tufféry 2011, S. 56). Die Testergebnisse bestätigten eine statistisch signifikante Abweichung der Umfragedaten von der Normalverteilung auf dem angenommenen Signifikanzniveau ($\alpha=0{,}05$) und machten den Einsatz von parameterfreien statistischen Tests in den weiteren Analysen erforderlich.

### 6.2.1 Bevorzugte Customizing-Ansätze

Der Typologie der Adaptionsansätze des Kapitels 4.1.1 folgend, wurden die Customizing-Strategien der einzelnen Anbieter der cloudbasierten ERP-Systeme analysiert. In Bezug auf die erste Frage dieser Fragegruppe (FCA01) haben die Befragten die Adaptionsansätze genannt, welche einen Bestandteil der bestehenden Customizing-Strategie des Systemanbieters darstellen. Die Aufteilung der einzelnen Customizing-Konzepte in der analysierten Stichprobe wurde in Tabelle 6.3 zusammengefasst.

**Tabelle 6.3:** Adaptionsansätze in der analysierten Stichprobe (n=31)

| Adaptionsansatz | Anzahl | Anteil (%) |
|---|---|---|
| Komposition | 28 | 90% |
| Parametrisierung | 31 | 100% |
| Modellbasierte Generierung | 5 | 16% |
| User-Exits | 17 | 55% |
| Programmierschnittstellen | 26 | 84% |
| Codebasierte Erweiterung | 26 | 84% |
| Codebasierte Modifikation | 4 | 13% |

Die überwiegende Mehrheit der Respondenten bestätigt, dass das von ihnen angebotene ERP-System mithilfe der modulbasierter Komposition (90%) und der Parametrisierung (100%) angepasst werden kann. Bei der Mehrheit der analysierten, cloudbasierten ERP-Systeme (84%) werden Systemerweiterungen als eine der Anpassungstechniken eingesetzt. Daneben sind von 84% der analysierten Systemanbieter die Systemadaptionen mittels Programmierschnittstellen vorgesehen. 55% der analysierten ERPaaS bieten User-Exits an. Lediglich bei 16% der betrachteten On-Demand-ERP-Systeme können die Anpassungen mittels modellbasierter

Generierung des Komponenten-Quellcodes und bei weiteren 13 % mithilfe codebasierter Modifikation vorgenommen werden.

Die durchgeführte Analyse bestätigt eindeutig, dass die cloudbasierten ERP-Systeme unterschiedliche Customizing-Ansätze bereitstellen, die über den durch die parameterbasierte Konfiguration gekennzeichneten Rahmen hinausgehen. Bei jedem der analysierten ERP-Systeme wurden mindestens drei unterschiedliche Customizing-Ansätze genannt, die einen Bestandteil der Customizing-Strategie des Systemanbieters darstellen und im Rahmen der anwenderseitigen Adaptionsprozesse verwendet werden können.

Überraschenderweise wurde im Fall der vier analysierten ERP-Systeme die codebasierte Modifikation als eine der bereitgestellten Customizing-Ansätze genannt. Diese Antworten bedürfen einer weiteren Klarstellung. Die befragten Experten haben erklärt, dass eine Modifikation des Systemquellcodes weiterhin möglich ist, wenn das ERP-System in einer privaten Cloud bereitgestellt wird. Von jedem der vier Systemanbieter wird die codebasierte Modifikation als eine zusätzliche Dienstleistung angeboten, welche im Rahmen der standardisierten Abonnementlizenz nicht enthalten ist. Keiner der befragten Systemanbieter erlaubt es, die Codeänderungen eigenständig von der Anwenderseite durchzuführen noch stellt er Werkzeuge für die codebasierte Modifikation zur Verfügung.

Nachfolgend wurden die Befragten gebeten, die Rolle der einzelnen Ansätze im Rahmen der gesamten Customizing-Strategie des Systemanbieters zu bewerten (siehe Frage FCA02). Die Ergebnisse sind in Tabelle 6.4 dargestellt.

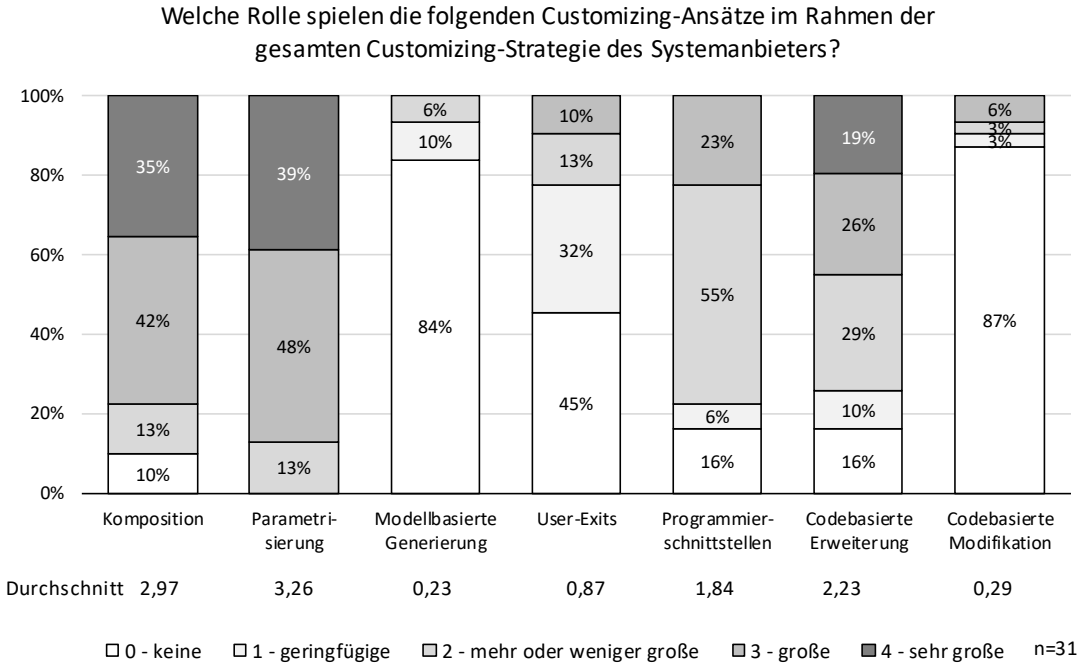



**Abbildung 6.2:** Wichtigkeit der Customizing-Ansätze

Bezogen auf die Wichtigkeit für das anwenderseitige Customizing waren die am höchsten bewerteten Adaptionsansätze die Parametrisierung (im Durchschnitt 3,26) und die Komposition (2,97), gefolgt von den codebasierten Erweiterungen (2,23) und den Programmierschnittstellen (1,84). Die User-Exits (0,87), die codebasierte Modifikation (0,29) und modellbasierte Generierung (0,23) wurden deutlich niedriger bewertet.

In Bezug auf die erste Forschungsfrage (FF 1.1) lässt sich zusammenfassen, dass die analysierten On-Demand-ERP-Systeme verschiedene Customizing-Ansätze verfolgen, welche in der Entwurfsphase des Systems festgelegt wurden. Viele Systeme bieten eine kohärente Zusammenstellung unterschiedlicher Anpassungstechniken an, um möglichst genau an die Bedürfnisse des Anwenderunternehmens angepasst werden zu können. Auch der Einsatzbereich der jeweiligen Anpassungstechnik ist bei verschiedenen ERP-Systemen unterschiedlich. Daher erfordern Anpassungen, die in einem System mittels der Parametrisierung durchgeführt werden, bei anderen Systemen den Einsatz von User-Exits bzw. codebasierten Erweiterungen.

### 6.2.2 Adaptionsansätze in Abhängigkeit von der Systemarchitektur

Zur Untersuchung von Zusammenhängen zwischen der SaaS-Reife eines ERP-Systems und dem Ausmaß der Systemanpassungen, die mittels der bereitgestellten Customizing-Ansätze vorgenommen werden können (siehe Hypothese H1), wurden die Spearman-Rangkorrelationskoeffizienten (Spearman Rho, r) berechnet und auf Signifikanz überprüft. Ein Spearman-Koeffizient bewegt sich im Intervall von −1 bis +1, wobei der Betrag die Stärke des Zusammenhanges beider Merkmale angibt und das Vorzeichen die Richtung der Korrelation definiert.

Tabelle 6.4 stellt die ermittelten Rangkorrelationskoeffizienten und die zugehörigen Signifikanzwerte (p-Werte) dar. Statistisch signifikante Ergebnisse sind fett gedruckt.

**Tabelle 6.4:** Ergebnisse der Korrelationsanalyse zwischen der SaaS-Reife und der Bewertung der Wichtigkeit einzelner Adaptionsansätze

| Adaptionsansatz | SaaS-Reife | |
|---|---|---|
| | Rangkorrelation (r) | Signifikanzwert (p-Wert) |
| Komposition | **0,524 **** | **0,002** |
| Parametrisierung | **0,663 **** | **0,001** |
| Modellbasierte Generierung | 0,214 | 0,248 |
| User-Exits | -0,342 | 0,060 |
| Programmierschnittstellen | 0,046 | 0,807 |
| Codebasierte Erweiterung | 0,009 | 0,962 |
| Codebasierte Modifikation | **-0,400 *** | **0,026** |

**Legende:** * Korrelation ist bei Niveau 0,05 signifikant (zweiseitig). ** Korrelation ist bei Niveau 0,01 signifikant (zweiseitig).

Die Analyse der Rangkorrelationen hat gezeigt, dass sich mit ansteigender SaaS-Reife des ERP-Systems die Bedeutung der modulbasierten Komposition und Parametrisierung erhöht, und sich der Einsatz von codebasierten Modifikationen verringert. Diese Zusammenhänge sind auf dem angenommenen Signifikanzniveau ($\alpha = 0{,}05$) statistisch signifikant.

Die Testergebnisse unterstützen somit die Hypothese H1, welche besagt, dass die ausgereiften Cloud-ERP-Systeme weniger codebasierte Customizing-Ansätze zur Verfügung stellen als die nicht ausgereiften ERPaaS.

In der untersuchten Stichprobe ist mit ansteigender SaaS-Reife ein Anstieg der Bedeutung der weiteren Customizing-Ansätze – wie modellbasierte Generierung und Programmierschnittstellen – sichtbar, der durch einen Rückgang der Bedeutung von User-Exits begleitet wird. Diese Korrelationen gelten jedoch auf dem angenommenen Signifikanzniveau als nicht statistisch signifikant ($p > 0{,}05$). Daher wird die Hypothese H1 nur im Falle der drei zuerst genannten

Customizing-Ansätze, d. h. der modulbasierten Komposition, Parametrisierung und codebasierten Modifikation, unterstützt.

Zum Zweck einer detaillierteren Analyse der Intensität des Einsatzes unterschiedlicher Customizing-Ansätze an den einzelnen Reifegraden des SaaS-Modells wurde der parameterfreie Mann-Whitney-U-Test (MWU-Test) durchgeführt. Mithilfe der paarweisen Vergleiche zwischen den einzelnen Reifegrad-Gruppen wurden die Inhomogenitäten beim Einsatz der analysierten Customizing-Ansätze untersucht.

Die Analyse der durch den Test ermittelten mittleren Ränge erlaubt es, die Inhomogenitäten im Einsatz der Customizing-Ansätze zwischen den einzelnen Reifegrad-Gruppen näher zu betrachten (siehe Anhang C.3). Erwartungsgemäß werden die deutlich sichtbaren Inhomogenitäten zwischen den stark ausgereiften Systemen (Reifegrad IV) und den nicht ausgereiften Systemen (Reifegrade I und II) Systemen im Fall der codefreien (Komposition und Parametrisierung) und der codebasierten Customizing-Ansätze (codebasierte Modifikation) nachgewiesen.

Im Fall der ERP-Systeme des zweiten Reifegrades wird noch die codebasierte Modifikation als einer der Adaptionsansätze angeboten. Dies ist aufgrund des Fehlens von Mandantenfähigkeitsmechanismen dieser ERP-Systeme möglich. Bei dieser Reifegruppe ist jedoch schon eine höhere Bedeutung der codefreien Customizing-Ansätze, insbesondere der Komposition (Mann-Whitney-U = 12, p = 0,042) und der Parametrisierung (U = 8, p = 0,01), erkennbar.

Die ERP-Systeme der höheren SaaS-Reifegrade präsentieren einen starken Rückgang im Einsatz der codebasierten Customizing-Ansätze. Die Analyse der mittleren Ränge weist auf eine systematisch fallende Bedeutung der codebasierten Modifikation und der User-Exits beim Übergang auf eine höhere SaaS-Reife eines cloudbasierten ERP-Systems hin. Obwohl in beiden Ansätzen diese Tendenz beobachtet wird, bestätigt der MWU-Test ($p < 0,05$) nur im Falle der codebasierten Modifikation die statistische Signifikanz der Ergebnisse.

Zwischen den ERP-Systemen, die durch die dritte und vierte SaaS-Reife gekennzeichnet sind, werden keine signifikanten Unterschiede bezüglich der Customizing-Konzepte erkannt. In beiden Reifegrad-Gruppen stellen die Komposition und Parametrisierung die wichtigsten Customizing-Ansätze dar. Die Analyse der mittleren Ränge der beiden SaaS-Reifen signalisiert neben der steigenden Bedeutung der Komposition und Parametrisierung auch einen leichten Anstieg der Rolle von Systemerweiterungen und Programmierschnittstellen bei den ausgereifteren Systemen. Diese Erkenntnisse erweisen sich jedoch im Rahmen der analysierten Stichprobe als nicht statistisch signifikant.

Eine allgemeine Beobachtung im Hinblick auf die Cloud-Bereitstellungsmodelle ist, dass die codebasierten Adaptionsansätze im Fall der privaten Cloud höher bewertet wurden als im Fall der öffentlichen Cloud, während für die codefreie Anpassung gerade das Gegenteil gilt. Statistisch signifikante Differenzen (MWU-Test, $p < 0,05$) zwischen den beiden Cloud-Bereitstellungsmodellen wurden jedoch ausschließlich im Fall der Parametrisierung und der codebasierten Modifikation bewiesen.

Diese Erkenntnisse stimmen mit den Ergebnissen anderer Studien überein (vgl. z. B. Uppström et al. 2015, S. 4226; Seethamraju 2015, S. 479), die auf eine grundsätzliche Präferenz der Systemanbieter für den Einsatz der Parametrisierung und eine Begrenzung der Anwendung von codebasierten Ansätzen für die Adaption der ERP-Systeme im öffentlichen Cloud-Modell hinweisen. Die cloudbasierten ERP-Systeme, die im öffentlichen Cloud-Modell bereitgestellt werden, werden für die Bedienung mehrerer Anwenderunternehmen mit der gleichen Anwendungsinstanz konzipiert. Dieses Bereitstellungskonzept erfordert somit die Sicherstellung einer einheitlichen Codebasis (vgl. Hofmann und Woods 2010).

## 6.3 Adaptierbarkeit der Systemelemente

Der Adaptierbarkeitsgrad eines ERP-Systems ist durch die Anpassungsmöglichkeiten jeder seiner architektonischen Ebenen (Schichten) definiert (vgl. Mijač et al. 2013, S. 137). Im Falle des ERP-Systems, dessen architektonisches Konzept dem Dreischichtenmodell entspricht, wird seine Adaptierbarkeit entsprechend durch die Anpassungsmöglichkeiten der Benutzeroberfläche, der Geschäftsprozesslogik und des Datenmodells definiert (vgl. Jiang et al. 2010, S. 279 f.). Ein zusätzlicher Aspekt, welcher für die meisten Anwenderunternehmen eine wichtige Rolle spielt, ist die Möglichkeit der Integration von anderen Elementen des Businesssystemumfelds in das ERP-System (vgl. Peng und Gala 2014, S. 28).

### 6.3.1 Anwenderseitige Adaptierbarkeit

Aufbauend auf dem Katalog der häufigsten Adaptionsanforderungen (siehe Kapitel 2.4.1) wurden die cloudbasierten ERP-Systeme im Rahmen der Frage FSE02 hinsichtlich der Flexibilität einzelner Systemelemente untersucht. Die Befragten bewerteten das Ausmaß, zu dem die einzelnen Systemelemente von den unternehmensinternen Benutzern mit administrativen Rechten (sogenannte Anwendungsexperten) an die Anforderungen des Anwenderunternehmens angepasst werden können.

Die Auswertung der gewonnenen Umfragedaten bezüglich dieser Frage wurde in Form eines Säulendiagrams in Abbildung 6.3 dargestellt.

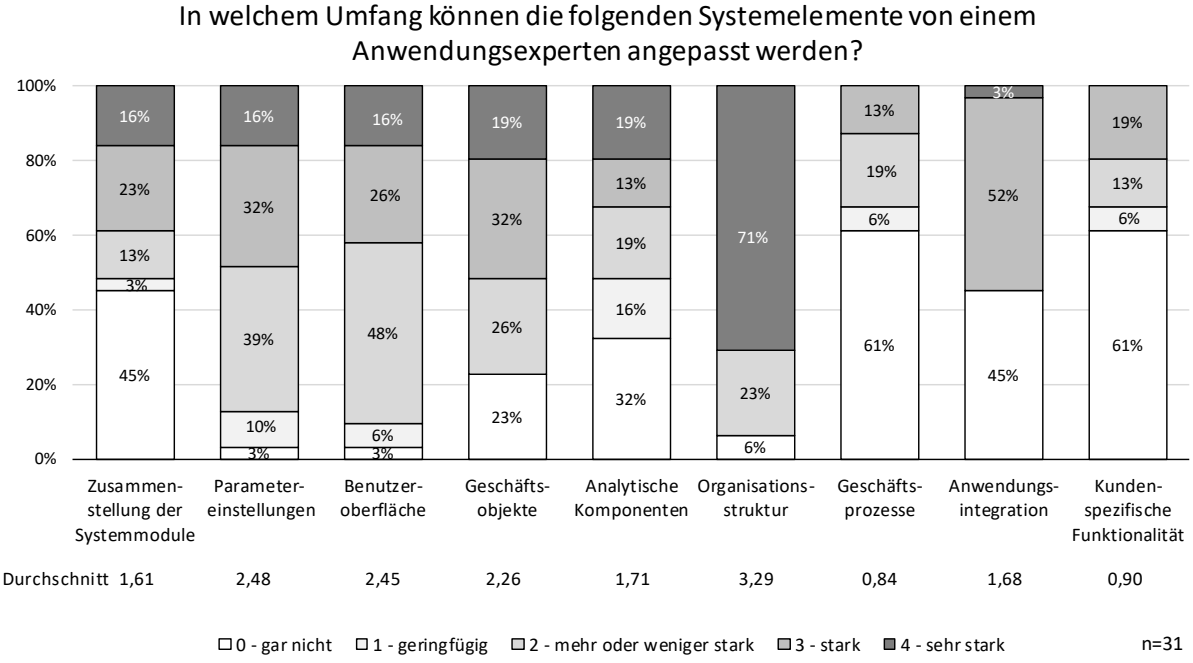



**Abbildung 6.3:** Bewertung der anwenderseitigen Adaptierbarkeit

Die Kategorie „gar nicht“ wird als Mangel an Möglichkeiten zur Durchführung von Systemadaptionen seitens der Anwendungsexperten bzw. als eine Notwendigkeit der Inanspruchnahme eines externen Supports bei der Systemanpassung interpretiert, welche nur vom Systemanbieter bzw. seinen Partnern vorgenommen werden kann.

Der Großteil (55 %) der befragten Systemanbieter bestätigt, dass die Anpassungen des funktionalen Systemumfangs durch eine Änderung der Modulzusammenstellung seitens der Anwen-

dungsexperten vorgenommen werden können. Eine Änderung der parameterbasierten System-Einstellungen kann bei nahezu jedem der betrachteten ERP-Systeme (97%) von der Anwenderseite erfolgen. Die Mehrheit (97%) der Systemanbieter erlaubt es den anwenderseitigen Administratoren, die Benutzeroberfläche eigenständig anzupassen. Ähnlich bewerteten die Befragten die Adaptierbarkeit der Organisationsstruktur, die bei 94% der analysierten Stichprobe selbstständig angepasst werden kann.

Bei 77% der betrachteten ERP-Systeme kann eine direkte Anpassung bzw. Erweiterung der Geschäftsobjekte seitens des Anwenderunternehmens vorgenommen werden. Mehr als die Hälfte der Respondenten (68%) bestätigt weiterhin, dass ihre cloudbasierten ERP-Systeme eine Individualisierung der analytischen Komponenten im Rahmen einer anwenderseitigen Anpassung ermöglichen. Ebenso wies die Mehrheit (55%) der Befragten auf die Flexibilität ihrer Systeme hinsichtlich der Integration externer Anwendungen hin. Im Gegensatz zu den weiteren Systemelementen ist die Adaptierbarkeit der funktionalen Systemaspekte relativ begrenzt. Lediglich im Fall von 39% der analysierten ERP-Systeme können kundenspezifische Erweiterungen der Systemfunktionalität durchgeführt werden und bei weiteren 39% Adaptionen der Geschäftsprozesse ohne die Inanspruchnahme des Supports seitens der Systemanbieter vorgenommen werden.

In Bezug auf den bereitgestellten Flexibilitätsumfang wurden die Systemelemente Organisationsstruktur (im Durchschnitt 3,29) und Systemparameter (2,48) am höchsten bewertet, gefolgt von der Benutzeroberfläche (2,45) und den Geschäftsobjekten (2,26). Die Geschäftsprozesse und funktionalen Systemelemente werden durch eine relativ geringe Flexibilität gekennzeichnet und wurden daher entsprechend niedriger (0,84 und 0,90) bewertet.

Im Hinblick auf die Flexibilität der einzelnen Systemschichten (siehe Tabelle 6.1, Frage FSE01) wird durch die Experten die Flexibilität der Präsentationsschicht (2,58) höher bewertet als die der Datenebene (2,29) und der Ebene der Anwendungslogik (1,94). Die bereitgestellte Adaptierbarkeit der Integrationsmechanismen wurde von den Befragten am geringsten bewertet (1,81).

Bei der Gegenüberstellung der Ergebnisse der Kapitel 6.2.1 und 6.3.1 kann festgestellt werden, dass die Systemanbieter den Umfang eines anwenderseitigen Customizing als einen wichtigen Faktor zur Sicherung der Systemakzeptanz bzw. zur Erweiterung der Wettbewerbsfähigkeit ihrer Systeme erkennen (vgl. z. B. Hohmann et al. 2013; Lechesa et al. 2012). Dieses Ergebnis wird durch die Tatsache untermauert, dass diejenigen Systemanbieter, welche die komponenten- und parameterbasierten Customizing-Ansätze höher bewerteten, einen höheren Umfang der anwenderseitigen Adaptierbarkeit bezüglich der modularen Systemzusammenstellung sowie der Parametereinstellungen sicherstellen (siehe Abbildung 6.3).

Die Testergebnisse zeigen eine sehr starke, signifikante Rangkorrelation ($r = 0{,}752$, $p = 000$) zwischen der Bedeutung des Komposition-Ansatzes im Rahmen der Customizing-Strategie des Systemanbieters und dem Umfang, in dem die Zusammenstellung der Systemmodule von einem Anwendungsexperten angepasst werden kann. Im Fall der Parametrisierung wurde eine schwache aber signifikante Rangkorrelation ($r = 0{,}387$, $p = 0{,}032$) zwischen der Rolle des Customizing-Ansatzes und dem Umfang der anwenderseitigen Adaptierbarkeit mittels Parametereinstellung bewiesen.

### 6.3.2 Flexibilität der Systemelemente in Abhängigkeit von der Systemarchitektur

Im Rückblick auf die vorangegangenen Analysen (siehe Kapitel 6.2.2), bei denen einige Zusammenhänge zwischen dem SaaS-Reifegrad und der Bedeutung einzelner Customizing-Ansätze nachgewiesen wurden, wurde die Analyse der Adaptierbarkeit einzelner Systemelemente unter Berücksichtigung der SaaS-Reife vorgenommen.

Zur Überprüfung der statistischen Signifikanz der Hypothese H2, die eine SaaS-Reifeabhängigkeit der anwenderseitigen Anpassbarkeit einzelner Systemkomponenten vorweist, wurden die Spearmanschen Rangkorrelationskoeffizienten der beiden Merkmale berechnet. Die Ergebnisse der Korrelationsanalyse mit den zugehörigen Signifikanzwerten wurden in der Tabelle 6.5 zusammengestellt. Die statistisch signifikanten Testergebnisse sind fett markiert.

Die Ergebnisse der Korrelationsanalyse zeigen, dass je ausgereifter das von dem Respondenten beschriebene ERP-System war, desto umfangreicher wurden die anwenderseitigen Änderungen der Modulzusammenstellung ($r=0{,}561$, $p=0{,}001$), der Systemparameter ($r=0{,}502$, $p=0{,}002$) sowie der Integration externer Anwendungen ($r=0{,}526$, $p=0{,}002$) bewertet. Diese Testergebnisse sind auf dem angenommenen Signifikanzniveau ($\alpha=0{,}05$) statistisch signifikant und unterstützen somit die Hypothese H2. Im Falle der anderen Systemelemente wurden keine statistisch signifikanten Zusammenhänge zwischen der bereitgestellten Flexibilität und dem SaaS-Reifegrad bewiesen.

**Tabelle 6.5:** Ergebnisse der Korrelationsanalyse zwischen der SaaS-Reife und der Bewertung der anwenderseitigen Adaptierbarkeit

| Systemelement | SaaS-Reife | |
|---|---|---|
| | **Rangkorrelation (r)** | **Signifikanzwert (p-Wert)** |
| Zusammenstellung der Systemmodule | **0,561 **** | **0,001** |
| Parametereinstellungen | **0,502 **** | **0,002** |
| Benutzeroberfläche | 0,268 | 0,144 |
| Geschäftsobjekte | 0,096 | 0,608 |
| Analytische Komponenten | 0,266 | 0,148 |
| Organisationsstruktur | 0,269 | 0,143 |
| Geschäftsprozesse | 0,303 | 0,098 |
| Anwendungsintegration | **0,526 **** | **0,002** |
| Kundenspezifische Funktionalität | 0,308 | 0,091 |

**Legende:** * Korrelation ist bei Niveau 0,05 signifikant (zweiseitig). ** Korrelation ist bei Niveau 0,01 signifikant (zweiseitig).

Die Ergebnisse bestätigen, dass die Anbieter der ausgereiften cloudbasierten ERP-Systeme im Vergleich zu den Herstellern der weniger ausgereiften Gegenstücke einen größeren Wert auf die Ermöglichung der eigenständigen Zusammenstellung von Systemkomponenten und der Änderung von Parametereinstellungen sowie der Integration anderer Anwendungen des Geschäftsumfelds durch das Anwenderunternehmen legen.

Diese Erkenntnis wird durch frühere Untersuchungen unterstützt, die zeigen, dass einige ausgereifte cloudnative ERP-Systeme, die mit integrierten Adaptionswerkzeugen und -funktionen ausgestattet sind, eine unternehmensindividuelle Zusammenstellung und Parametrisierung des Systemumfangs (vgl. Ortner und Krenn 2016) sowie eine Integration externer Webdienste und Anwendungen ermöglichen (vgl. Nowak und Kurbel 2014).

Wie die Ergebnisse der durchgeführten Machbarkeitsstudien bewiesen, werden die cloudbasierten ERP-Systeme in der Regel mit dedizierten Konfigurationsmechanismen und -werkzeugen ausgestattet, welche die Prozesse der komponentenbasierten Systemzusammenstellung und der anschließenden Parametrisierung weitgehend automatisieren. Die Ergebnisse bestätigen weiterhin, dass die Möglichkeit des Anwenderunternehmens zur selbständigen Durchführung

von Integrationsprozessen mit der Business-Software sowie den internetbasierten Diensten und Datenquellen sich mit dem Anstieg der SaaS-Reife des ERP-Systems erhöht.

Die Testergebnisse mit den Ergebnissen bezüglich der Hypothese H1 (siehe Kapitel 6.2.2) vergleichend, kann geschlussfolgert werden, dass der SaaS-Reife-abhängige Anstieg der Wichtigkeit von Komposition und Parametrisierung sich direkt im Ausmaß der von den Anwendungsexperten durchführbaren Anpassungen der modularen Systemzusammenstellung und der Systemeinstellungen widerspiegelt.

Ein weiteres erwähnenswertes Ergebnis ist die wichtige Rolle von Plattform-as-a-Service im Rahmen des anwenderseitigen Customizing. Gemäß den Studienergebnissen stellen die PaaS-basierten ERP-Systeme den anwenderseitigen Administratoren umfangreichere Customizing-Möglichkeiten bei allen betrachteten Systemelementen zur Verfügung.

Die Abbildung 6.4 visualisiert den positiven Effekt der Bereitstellung einer mit dem cloud-basierten ERP-System gekoppelten Entwicklungsplattform auf den Umfang der Systemadaptionen, die von einem Anwendungsexperten eigenständig vorgenommen werden können.

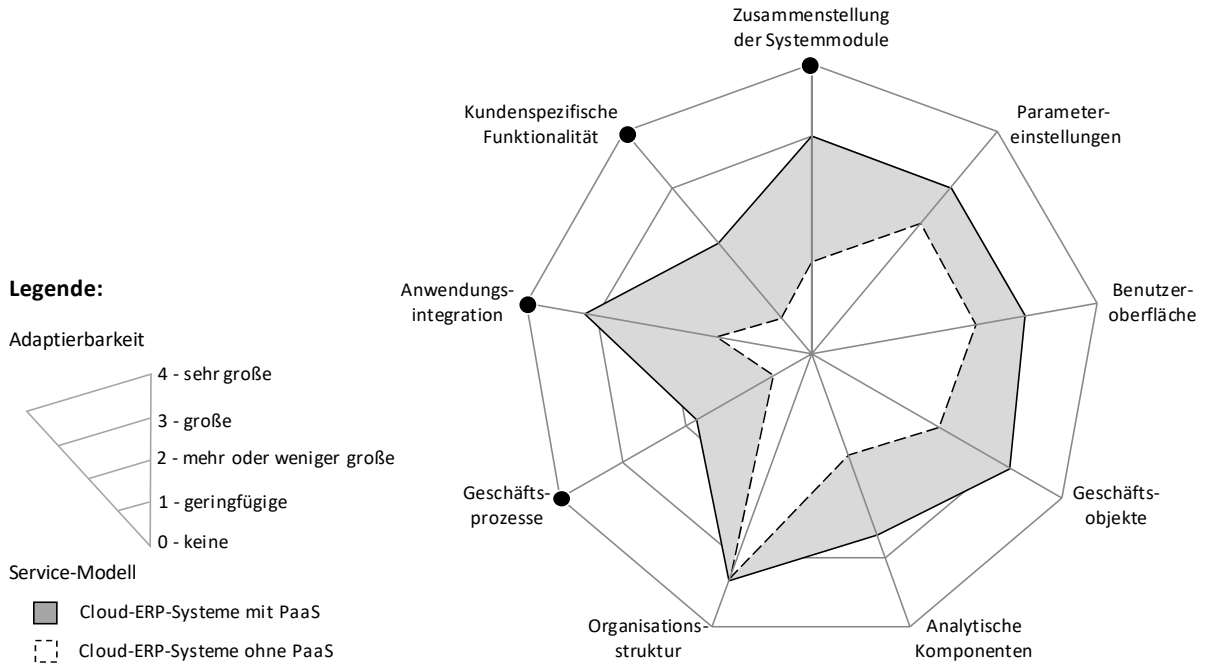



**Abbildung 6.4:** Bewertung der anwenderseitigen Adaptierbarkeit in Abhängigkeit von dem Service-Modell

Das Radardiagramm stellt die Mittelwerte gegenüber, die für zwei Systemgruppen – die cloud-basierten ERP-Systeme mit und ohne PaaS – und für jedes Systemelement berechnet wurden. Die statistisch signifikanten Unterschiede ($p<0{,}05$) in dem bereitgestellten Flexibilitätsumfang zwischen den SaaS-basierten und den PaaS-gekoppelten ERP-Systemen wurden durch den MWU-Test bestätigt. Die Testergebnisse zeigen, dass PaaS-gekoppelte ERP-Systeme insbesondere umfangreichere Adaptionen der funktionalen Systemelemente ermöglichen sowie weitergehende Änderungen der Geschäftsprozesse und Adaptionen der Integrationslogik zulassen. Die Systemelemente, welche für die PaaS-basierten ERP-Systeme signifikant höher eingestuft wurden, sind im Diagramm durch einen Punkt ‚●' gekennzeichnet.

## 6.4 Bereitgestellte Adaptionswerkzeuge

Die Umsetzung unterschiedlicher Customizing-Strategien macht die Bereitstellung von verschiedenen Adaptionswerkzeugen erforderlich, mit deren Hilfe das cloudbasierte ERP-System an die Anforderungen des Anwenderunternehmens angepasst werden kann.

Basierend auf der Typologie des Kapitels 4.1.3 wurde die Systematisierung der Adaptionswerkzeuge in Abhängigkeit vom Integrationsgrad mit der Basisanwendung vorgenommen.

### 6.4.1 Adaptionswerkzeuge des anwenderseitigen Customizing

Im Hinblick auf die im Kapitel 4.1.3 vorgestellte Kategorisierung wurden von den Befragten die durch den Systemhersteller bereitgestellten Customizing-Werkzeuge, systematisiert und deren Anwendungsumfang bestimmt (siehe Tabelle 6.1, Fragen FCW01-2). Die Übersicht über die im Studienrahmen identifizierten Anpassungswerkzeuge, bezogen auf den Integrationsgrad mit dem Kernsystem, wurde in der Tabelle 6.6 zusammengestellt. Neben jeder Werkzeug-Kategorie wurden die prozentualen Anteile aufgeführt, welche als Bruchteil der mit dem Werkzeug ausgestatteten ERPaaS an der Grundgesamtheit der untersuchten ERP-Systeme berechnet wurden.

**Tabelle 6.6:** Aufteilung der Customizing-Werkzeuge in der analysierten Stichprobe (n = 31)

| Werkzeug-Kategorie | Anzahl | Anteil (%) |
|---|---|---|
| Integrierte Adaptionswerkzeuge | 31 | 100 % |
| Externe lokal installierbare Adaptionswerkzeuge | 12 | 39 % |
| Externe cloudbasierte Adaptionswerkzeuge | 14 | 45 % |

Die Ergebnisse der Umfrage bestätigen, dass alle analysierten, cloudbasierten ERP-Systeme (100%) von ihren Systemherstellern mit integrierten Customizing-Werkzeugen ausgestattet sind, welche die eigenständige Durchführung einiger Systemanpassungen im Rahmen des On-Demand-Selfservices ermöglichen. Jedes ERP-System besitzt seine individuelle Zusammenstellung von internen Customizing-Werkzeugen, die abhängig von der verfolgten Customizing-Strategie des Systemanbieters im System verankert wurden.

Viele der betrachteten Systemanbieter (39 %) bieten bereits neben den systeminternen auch die externen Customizing-Werkzeuge an, die keinen integralen Bestandteil des ERP-Systems darstellen und in der Regel eine Installation auf einem lokalen Rechner des Anwendungsexperten bzw. Entwicklers erfordern. Im Rückblick auf die Erkenntnisse der Machbarkeitsstudien ermöglichen es die externen Werkzeuge in der Regel, umfangreichere Adaptionen und Erweiterungen am ERP-System vorzunehmen, als dies bei den integrierten Customizing-Werkzeuge möglich ist. Mittels der systemexternen Werkzeuge können die Benutzer nicht nur die bestehenden Systemelemente (z. B. Geschäftsdatenobjekte, Funktionalitäten etc. individualisieren; siehe. z. B. Kapitel 5.3.2, 5.4.3), sondern auch unternehmensindividuelle Komponenten eigenständig entwickeln und das System hinsichtlich der Anforderungen des Kundenunternehmens erweitern (siehe z. B. Kapitel 5.10.1).

Weiterhin stellen 45 % der analysierten Systemanbieter dedizierte cloudbasierte Dienste und Lösungen bereit, die eine gezielte Erweiterung des standardisierten Funktionalitätsumfangs des cloudbasierten ERP-Systems ermöglichen.

Zusammenfassend bestätigt die Analyse der Umfragedaten, dass die Strategie der Systemanbieter hinsichtlich der Bereitstellung von systemexternen Adaptionswerkzeugen für ihre Kunden nicht einheitlich ist. Im Vergleich dazu werden in jedem cloudbasierten ERP-System interne Werkzeuge verankert, die eine gezielte Adaption mehrerer Systemaspekte sicherstellen.

Im Anschluss bewerteten die Systemanbieter die Bedeutung der einzelnen Werkzeug-Kategorien für das anwenderseitige Customizing (Frage FCW02). Die erhobenen Daten sind in der Abbildung 6.5 veranschaulicht.

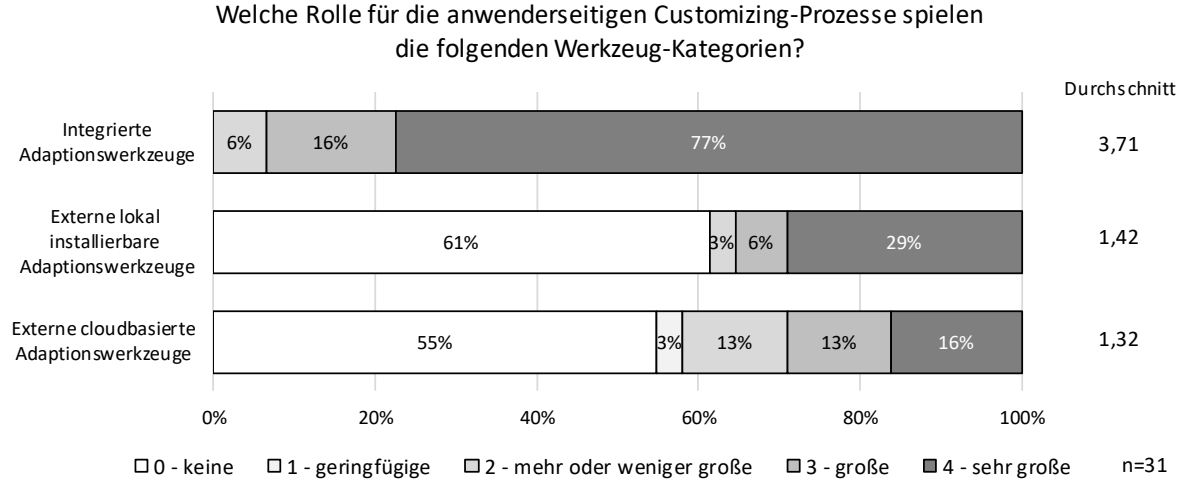



**Abbildung 6.5:** Bedeutung der Werkzeug-Kategorien für das anwenderseitige Customizing

In Bezug auf die Bedeutung der einzelnen Werkzeug-Kategorien für die anwenderseitigen Adaptionen stellen die systeminternen Werkzeuge (mit dem Durchschnitt 3,71) die höchstbewertete Werkzeug-Gruppe dar. Die Rolle der systemexternen, lokal installierbaren Customizing-Maßnahmen (1,42) wurde von den Befragten im Durchschnitt niedriger bewertet. Die systemexternen cloudbasierten Lösungen und Services (1,32) werden eher als Hilfskonzepte angesehen. Dabei sei anzumerken, dass die letzte Kategorie sich ausschließlich auf diejenigen Werkzeuge bezieht, welche im Rahmen des On-Demand-Selfservices seitens des Anwenderunternehmens verwendet werden können. Es wird z. B. die Inanspruchnahme der Dienstleistung einer direkten Programmierung seitens des Systemanbieters nicht berücksichtigt.

Die Gegenüberstellung der Bedeutung einzelner Werkzeug-Gruppen verdeutlicht die entscheidende Rolle der integrierten Werkzeuge für das anwenderseitige Customizing. Im Vergleich zu den anderen Werkzeugen sind die internen Tools in jedem System vorhanden. Die Analyse der Umfragedaten bestätigt, dass die Strategie der Systemanbieter hinsichtlich der Bereitstellung von systemexternen Softwareentwicklungs-Tools für ihre Kunden nicht einheitlich ist. Die nachstehenden Unterkapitel präsentieren eine Analyse der Werkzeug-Verfügbarkeit in Abhängigkeit von der SaaS-Reife des ERP-Systems und eine Bewertung der werkzeugbasierten Unterstützung der anwenderseitigen Adaptionsprozesse.

### 6.4.2 Werkzeugverfügbarkeit bei den einzelnen Stufen des Reifegradmodells

Die Verfügbarkeit bestimmter Kategorien der Customizing-Werkzeuge kann in Abhängigkeit von der angenommenen Systemarchitektur, dem Cloud-Bereitstellungsmodell (vgl. Uppström et al. 2015, S. 4224) sowie dem ERPaaS-Reifegrad variieren (vgl. Kang et al. 2010, S. 341 ff.; Sun et al. 2008, S. 20). Wie die im Kapitel 6.2.2 vorgenommene Analyse bestätigt hat, werden die

einzelnen Customizing-Ansätze mit unterschiedlicher Intensität bei den einzelnen SaaS-Reifegraden repräsentiert. Die genauere Analyse der untersuchten Systeme in Bezug auf die zur Verfügung stehenden Customizing-Werkzeuge wird in diesem Abschnitt durchgeführt.

Basierend auf den Umfragedaten bezüglich der Frage FCW01 wurde eine Analyse der Zusammenhänge zwischen der SaaS-Reife eines ERP-Systems und dem Vorhandensein der einzelnen Werkzeug-Kategorien vorgenommen. Für die Überprüfung der statistischen Signifikanz der Abhängigkeit beider Merkmale wurden die Spearmanschen Rangkorrelationskoeffizienten berechnet. Die Tabelle 6.7 stellt die ermittelten Korrelationskoeffizienten sowie die zugehörigen Signifikanzwerte dar.

Die durchgeführte Analyse weist auf die statistisch signifikanten Zusammenhänge zwischen der SaaS-Reife eines cloudbasierten ERP-Systems und der Verfügbarkeit der einzelnen Kategorien der Customizing-Werkzeuge hin.

Die Ergebnisse der Korrelationsanalyse zeigen, dass je ausgereifter das von dem Respondenten beschriebene ERP-System war, umso häufiger wurden die externen lokal installierbaren Adaptionswerkzeuge als Maßnahmen des anwenderseitigen Customizing genannt ($r = 0{,}436$, $p = 0{,}014$). Ebenso wurde bei den ausgereiften ERP-Systemen öfter angegeben, dass die cloudbasierten Customizing-Lösungen für die Anwenderunternehmen zugänglich sind ($r = 0{,}392$, $p = 0{,}029$).

**Tabelle 6.7:** Ergebnisse der Korrelationsanalyse zwischen der SaaS-Reife und der Verfügbarkeit einzelner Werkzeug-Kategorien

| **Werkzeug-Kategorie** | **SaaS-Reife** | |
|---|---|---|
| | **Rangkorrelation (r)** | **Signifikanzwert (p-Wert)** |
| Integrierte Adaptionswerkzeuge | 0,114 | 0,541 |
| Externe lokal installierbare Adaptionswerkzeuge | **0,436 *** | **0,014** |
| Externe cloudbasierte Adaptionswerkzeuge | **0,392 *** | **0,029** |

**Legende:** * Korrelation ist bei Niveau 0,05 signifikant (zweiseitig). ** Korrelation ist bei Niveau 0,01 signifikant (zweiseitig).

Die durchgeführte Analyse deutet keine statistisch signifikanten Inhomogenitäten bei der Bereitstellung der systeminternen Customizing-Werkzeuge zwischen einzelnen SaaS-Reifegraden an. Daraus schlussfolgernd werden die ERPaaS unabhängig von der SaaS-Reife mit den eingebauten Anpassungswerkzeugen ausgestattet. Diese Erkenntnis weist jedoch nicht zwangsläufig darauf hin, dass die gleiche Flexibilität der einzelnen Systemelemente auf unterschiedlichen Reifeebenen gewährleistet wird. Die vorgenommenen Machbarkeitsstudien des vorigen Kapitels weisen in diesem Kontext auf systembezogene Differenzen in dem bereitgestellten Flexibilitätsumfang hin.

Im Hinblick auf die Testergebnisse kann festgestellt werden, dass bei den wenig ausgereiften ERPaaS die anwenderseitige Systemanpassung hauptsächlich mithilfe der systeminternen Werkzeugen und Funktionen realisiert wird. Mit ansteigender Systemreife werden die Anwenderunternehmen in der Regel mit zusätzlichen, systemexternen Customizing-Werkzeugen ausgestattet, welche auch eine umfangreichere Individualisierung des cloudbasierten ERP-Systems ermöglichen und zugleich eine weitergehende Unabhängigkeit von dem Systemanbieter bei der Durchführung von Systemanpassungen zulassen (siehe Kapitel 6.3.2).

Das PaaS-Bereitstellungsmodell öffnet neue Möglichkeiten bei dem anwenderseitigen Customizing und begünstigt die Bereitstellung der von den externen Entwicklern dedizierten Ent-

wicklungsumgebungen, welche an die Entwicklungsplattform gekoppelt sind. Der mittelstarke positive Zusammenhang in der Bereitstellung der externen cloudbasierten Adaptionswerkzeuge und der lokal installierbaren Customizing-Tools für die Anwenderunternehmen wurde durch den Korrelationstest bestätigt ($r = 0{,}521$, $p = 0{,}003$). Dieses Ergebnis weist darauf hin, dass die Systemanbieter, die systemexterne Customizing-Werkzeuge für ihre Kunden zur Verfügung stellen, auch zusätzliche Dienstleistungen für die Integration der Lösungen von Drittanbietern anbieten. Dieser Zusammenhang ist insbesondere im Fall der PaaS-basierten ERP-Systeme ersichtlich, welche in der Regel mit einer internetbasierten Vertriebsplattform für Vermarktung der Systemerweiterungen ausgestattet werden und gegebenenfalls auch eine proprietäre Entwicklungsumgebung bereitstellen.

In diesem Kontext ist es jedoch merkwürdig, dass – obwohl für die ausgereiften, cloudbasierten ERP-Systeme die mit der cloudbasierten Plattform gekoppelten Entwicklungsumgebungen bereitgestellt werden – eine negative Korrelation zwischen der SaaS-Reife und der codebasierten Modifikation nachgewiesen wurde (siehe Kapitel 6.2.2). Dieses Phänomen kann jedoch dadurch erklärt werden, dass die anwenderdedizierten Entwicklungsumgebungen in der Regel zur Entwicklung der anwenderkonformen Systemerweiterungen und nicht zur direkten Modifikation des Systemquellcodes vorgesehen sind (vgl. z. B. Schneider 2012).

Im Hinblick auf die aufgestellte Forschungsfrage FF 1.3 kann zusammenfassend festgestellt werden, dass unterschiedliche Werkzeug-Kategorien im Rahmen des anwenderseitigen Customizing zur Anwendung kommen können. Trotz der allgemeingültigen Tendenz zur weitgehenden Verankerung der Customizing-Werkzeuge im Systemrahmen werden auch systemexterne Customizing-Maßnahmen bereitgestellt, die als lokal installierbare Anwendungen oder cloudbasierte Dienste angeboten werden. Im Fall der zwei zuletzt genannten Kategorien wird eine positive Korrelation mit der SaaS-Reife bestätigt, welche die Hypothese H3 unterstützt. Dies bedeutet, dass das anwenderseitige Customizing im Fall der ausgereiften ERP-Systeme mittels eines umfangreicheren Spektrums der Customizing-Werkzeuge erfolgen kann. Wie die Analyse der Systemflexibilität gezeigt hat (siehe Kapitel 6.3.2), spiegelt sich die Verfügbarkeit externer Werkzeuge im erweiterten Umfang des anwenderseitigen Customizing wider.

### 6.4.3 Werkzeugbasierte Unterstützung der anwenderseitigen Adaptionsprozesse

Im Anschluss an die Analyse der bereitgestellten Customizing-Werkzeuge, die auf einige reifebedingte Inhomogenitäten bei der Bereitstellung einzelner Werkzeug-Kategorien hingewiesen hat, wurde der Fokus auf die Konzepte gelegt, welche das anwenderseitige Customizing unterstützen. Das Ziel der Analyse bestand in der Überprüfung dessen, welche Konzepte zur Unterstützung bzw. Erleichterung der anwenderseitigen Customizing-Prozesse in den bereitgestellten Werkzeugen verankert werden, bzw. welche Anforderungen gegenüber den anwenderseitigen Administratoren gestellt werden.

Im Rahmen der Frage FCW03 wurde die Verfügbarkeit der einzelnen Mechanismen ermittelt, welche zwecks der Unterstützung des anwenderseitigen Customizing in den bereitgestellten Werkzeugkonzepten verankert sind. Die Respondenten haben für jedes der adaptierbaren Systemelemente (siehe Anhang C.1) die Mechanismen genannt, welche die anwenderseitigen Administratoren bei der Durchführung der Adaption unterstützen.

Die Auswertung der gewonnenen Umfragedaten bezüglich der werkzeugbezogenen Unterstützung der Adaptionsprozesse wurde in Form eines Säulendiagrams in der Abbildung 6.6 dargestellt.

Die Auswertung weist auf die dominierende Rolle der aktiven Prozessanleitung hin. Dieses Konzept der Anwenderunterstützung wird hauptsächlich in solche Werkzeuge verankert, welche zur Änderung der Parametereinstellungen (94 %), Anpassung der Organisationsstruktur

(68 %), der Benutzeroberfläche (48 %), der Adaption der Geschäftsobjekte (32 %) sowie zur Änderung der Zusammenstellung von Systemmodule (32 %) eingesetzt werden. Die Mechanismen der Prozesslenkung werden insbesondere in die Werkzeuge integriert, mit deren Hilfe die Individualisierung der Organisationstrukturen (26 %), der Benutzeroberfläche (26 %) und der analytischen Komponenten (19 %) vorgenommen wird. Die passive Beratung liegt in der Regel den Adaptionswerkzeugen zugrunde, die im Allgemeinen zur Adaption der funktionalen Systemelemente bereitgestellt werden. Die Anwenderunterstützung mittels der passiven Beratung findet daher insbesondere bei der Anwendungsintegration (45 %), der Adaption der Geschäftsprozesse (35 %) sowie der Implementierung der kundenspezifischen Systemfunktionalität (35 %) statt. Die Mechanismen der Prozessautomation werden insbesondere bei der Änderung der komponentenbasierten Zusammenstellung (23 %) und der Adaption der analytischen Komponenten (10 %) verwendet.

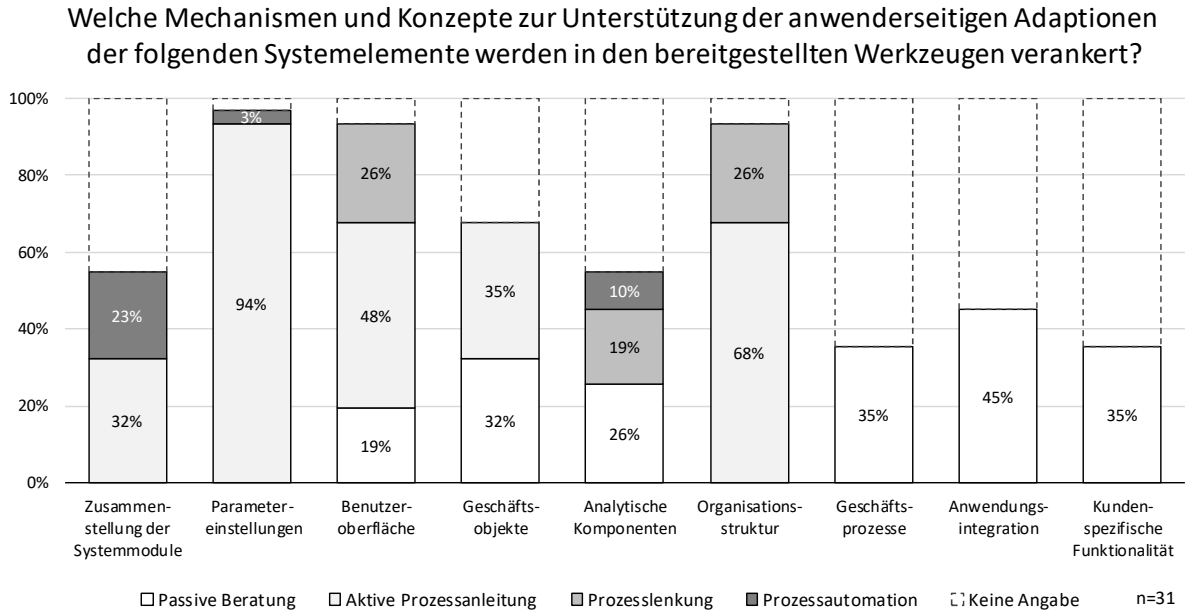



**Abbildung 6.6:** Konzepte der Anwenderunterstützung im Vergleich

Die Analyseergebnisse untermauern die Erkenntnisse der vorgenommenen Machbarkeitsstudien und bestätigen eine weitgehende Tendenz der Systemanbieter zur Erleichterung bzw. Automatisierung der anwenderseitigen Customizing-Prozesse durch die Integration unterstützender Werkzeugkonzepte.

Im Rahmen der Machbarkeitsstudien wurden jedoch auch einige Differenzen im Anwendungsumfang der einzelnen Mechanismen der Anwenderunterstützung identifiziert, welche insbesondere zwischen den ERP-Systemen des höchsten SaaS-Reifegrades und dem weniger ausgereiften ERP-System erkennbar sind. Diese Erkenntnisse stellten eine Prämisse für eine Abhängigkeit zwischen der SaaS-Reife und den bereitgestellten Mechanismen der Anwenderunterstützung dar.

Die Überprüfung des Zusammenhanges beider Merkmale erfolgte durch die Berechnung der Spearmanschen Rangkorrelationskoeffizienten und der zugehörigen Signifikanzwerte, welche in der Tabelle 6.8 zusammengestellt sind. Die statistisch signifikanten Zusammenhänge sind fett markiert.

**Tabelle 6.8:** Ergebnisse der Korrelationsanalyse zwischen der SaaS-Reife und den Mechanismen der Anwenderunterstützung

| Mechanismen der Anwenderunterstützung | SaaS-Reife | |
|---|---|---|
| | Rangkorrelation (r) | Signifikanzwert (p-Wert) |
| Passive Beratung | 0,231 | 0,211 |
| Aktive Prozessanleitung | 0,139 | 0,456 |
| Prozesslenkung | **0,399 *** | **0,026** |
| Prozessautomation | **0,642 **** | **0,000** |

**Legende:** * Korrelation ist bei Niveau 0,05 signifikant (zweiseitig). ** Korrelation ist bei Niveau 0,01 signifikant (zweiseitig).

Die in der Tabelle 6.8 enthaltenen Ergebnisse beweisen, dass der Umfang der Anwenderunterstützung, welcher durch die die Mechanismen der Prozesslenkung ($r=0,399$, $p=0,026$) und Prozessautomation ($r=0,642$, $p=0,000$) sichergestellt wird, umso größer ist, je ausgereifter das von dem Respondenten beschriebene ERP-System ist. Diese Testergebnisse sind auf dem angenommenen Signifikanzniveau ($\alpha=0,05$) statistisch signifikant und unterstützen daher die Hypothese H4. Im Fall der passiven Beratung und der aktiven Prozessanleitung wurde dagegen statistisch keine Korrelation mit dem SaaS-Reifegrad nachgewiesen.

## 6.5 Systematisierung der Erkenntnisse

Die vorgenommenen statistischen Auswertungen der Umfragedaten lieferten Antworten auf die erste Hauptforschungsfrage und ermöglichten die Verifizierung der aufgestellten Hypothesen, die systematisch zur Beantwortung der zweiten Hauptforschungsfrage beitrugen. Im Folgenden werden die Erkenntnisse des Dissertationsprojektes, bezogen auf die einzelnen Betrachtungsebenen des Systematisierungsmodells, zusammengestellt.

*Adaptionsansätze*

In Bezug auf die bereitgestellten Adaptionsansätze bietet die Untersuchung die folgenden Erkenntnisse an, die zugleich eine Antwort auf die erste der aufgestellten Untersuchungsfragen (FF 1.1) geben:

- Die Customizing-Strategien der cloudbasierten ERP-Systeme stellen eine systemindividuelle Zusammenstellung unterschiedlicher Adaptionsnsätze dar, deren Einsatzbereich von System zu System variieren kann. Trotz der systemübergreifend dominierenden Rolle der Parametrisierung und der Komposition gehen die Customizing-Strategien oft über die Grenzen einer rein codefreien Adaption hinaus. Sowohl die vorgenommenen Machbarkeitsstudien als auch die empirische Untersuchung weisen auf Möglichkeiten der Systemadaption mittels codebasierter Ansätze, unter anderem der User-Exits und der codebasierten Systemerweiterungen, hin.
- Die Rolle der einzelnen Adaptionsansätze im Rahmen der Customizing-Strategie weist einen Zusammenhang mit der SaaS-Reife des ERP-Systems auf. Die Stärke und Richtung der identifizierten Zusammenhänge mit dem SaaS-Reifegrad ist für verschiedene Adaptionsansätze unterschiedlich.
- Die Analyse der Rangkorrelationen bestätigt einen starken positiven Zusammenhang zwischen der SaaS-Reife eines cloudbasierten ERP-Systems und dem Ausmaß der Systemanpassungen, die mittels Komposition und Parametrisierung durchgeführt werden können. Mit anstei-

gender SaaS-Reife der analysierten ERP-Systeme wird der Einsatz der beiden Ansätze im Rahmen der Customizing-Strategien deutlicher ausgeprägt.

- Die Anbieter der weniger ausgereiften ERP-Systeme (I und II Reifegrad), die nicht durch Mandantenfähigkeit gekennzeichnet sind, lassen die codebasierten Modifikationen zu. Allerdings wird der Einsatz von codebasierten Modifikationen mit ansteigender SaaS-Reife fortlaufend reduziert. Bei den ausgereiften ERP-Systemen werden in der Regel keine signifikanten Unterschiede in den bereitgestellten Customizing-Konzepten ausgewiesen. Auf den höchsten Reifestufen (III und IV) stellen die modulbasierte Komposition und die Parametrisierung die wichtigsten Ansätze des anwenderseitigen Customizing dar.
- Die vorgenommene Analyse weist auf Inhomogenitäten beim Einsatz von codebasierten und codefreien Adaptionsansätzen zwischen den einzelnen SaaS-Reifen hin. Bei den ausgereiften ERPaaS werden weniger codebasierte Customizing-Ansätze zur Verfügung gestellt als bei den nicht ausgereiften ERPaaS. Die Hypothese H1, welche reifebedingte Unterschiede in der Bereitstellung einzelner Customizing-Ansätze vermutete, ist somit bestätigt.

*Adaptierbare Systemelemente*

Im Hinblick auf die bereitgestellte Flexibilität einzelner Systemelemente können von den vorgenommenen Untersuchungen die folgenden Erkenntnisse abgeleitet werden, welche zugleich die Antwort auf die aufgestellte Forschungsfrage FF 1.2 darstellen:

- Die Adaptierbarkeit einzelner Systemelemente wird durch die systemindividuelle Customizing-Strategie des Systemanbieters eindeutig vorgegeben. Die durchgeführten Untersuchungen weisen auf weitgehende Differenzen im Hinblick auf die bereitgestellte Flexibilität zwischen den unterschiedlichen cloudbasierten ERP-Systemen hin.
- Bei einer systemübergreifenden Betrachtung einzelner Systemelemente stellen die Organisationsstrukturen, die Systemparameter und die Benutzeroberfläche die größte Flexibilität bereit. Diese Systemelemente sind im Rahmen des anwenderseitigen Customizing weitgehend adaptierbar. Im Gegensatz dazu werden die Geschäftsprozesse und funktionale Systemelemente in der Regel durch die geringste Flexibilität gekennzeichnet. Eine differenzierte Analyse der einzelnen Systemschichten weist auf die relativ umfangreiche Adaptierbarkeit der Präsentationsschicht hin. Demgegenüber werden die Ebenen der Anwendungs- und Integrationslogik durch die vergleichbar geringste Flexibilität gekennzeichnet.
- Die Analyse der Rangkorrelationen hat im Falle der drei Systemelemente einen mittelstarken positiven Zusammenhang zwischen dem bereitgestellten Flexibilitätsumfang und der SaaS-Reife bewiesen. Die Hypothese H2 ist somit bestätigt. Die Ergebnisse bestätigen, dass die Anbieter der ausgereiften, cloudbasierten ERP-Systeme in der Regel einen größeren Wert auf die Ermöglichung der eigenständigen Zusammenstellung der Systemkomponenten, der Änderung der Parametereinstellungen und der Integration anderer Anwendungen des Geschäftsumfelds im Rahmen eines On-Demand-Selfservices legen.
- Die statistische Datenauswertung weist auf signifikante Inhomogenitäten bezüglich des bereitgestellten Flexibilitatsumfangs der cloudbasierten ERP-Systeme hin, die mit und ohne PaaS bereitgestellt werden. Gemäß den Studienergebnissen stellen die PaaS-gekoppelten ERP-Systeme den anwenderseitigen Administratoren umfangreichere Customizing-Möglichkeiten bei allen betrachteten Systemelementen zur Verfügung. Die größten Inhomogenitäten wurden im Fall der funktionalen Systemelemente statistisch nachgewiesen. Dies weist auf die bedeutende Rolle von Plattform-as-a-Service im Rahmen des anwenderseitigen Customizing hin.

- Die Ergebnisse der qualitativen und quantitativen Untersuchungen bestätigen, dass der bereitgestellte Flexibilitätsumfang von den Systemherstellern als wesentlicher Faktor wahrgenommen wird, der die Systemakzeptanz seitens der Anwenderunternehmen beeinflusst (vgl. Lechesa et al. 2012). Daher legen die Hersteller der cloudbasierten ERP-Systeme großen Wert auf die Bereitstellung effektiver, systeminterner Adaptionswerkzeuge, mit deren Hilfe erfolgskritische Systemanpassungen eigenständig und ohne Inanspruchnahme des Supports seitens der Systemanbieter umgesetzt werden können.

*Adaptionswerkzeuge*

Im Rahmen der durchgeführten Untersuchungen wurden zahlreiche Erkenntnisse auf der Ebene der bereitgestellten Adaptionswerkzeuge gesammelt, welche gleichzeitig zur Beantwortung der Forschungsfrage FF 1.3 beitragen:

- Die Hersteller von cloudbasierten ERP-Systemen stellen eine kohärente Zusammenstellung von Adaptionswerkzeugen bereit, welche aus den systeminternen bzw. -externen sowie cloudbasierten Lösungen und Tools bestehen kann. Systemübergreifend spielen die systeminternen Werkzeuge für die anwenderseitigen Adaptionsprozesse die wichtigste Rolle; diese sind in jedem ERP-System verankert und stellen eine gezielte Adaption mehrerer Systemelemente sicher.
- Die Strategien der Systemanbieter hinsichtlich der Bereitstellung einzelner Adaptionswerkzeuge für ihre Kunden sind nicht einheitlich. Bei den wenig ausgereiften ERPaaS stehen ausschließlich die eingebauten Adaptionswerkzeuge für die anwenderseitige Systemanpassung zur Verfügung. Mit ansteigender SaaS-Reife werden häufiger die externen Customizing-Werkzeuge (z. B. IDE) sowie die cloudbasierten Customizing-Lösungen für das Anwenderunternehmen bereitgestellt. Die Hypothese H3 wurde im Fall der zwei zuletzt genannten Werkzeug-Kategorien positiv validiert. Die Inhomogenitäten in deren Bereitstellung sind auf die Reife der ERPaaS zurückzuführen.
- Die Ergebnisse beider Untersuchungsphasen bestätigen eine weitgehende Tendenz der Systemanbieter zur Erleichterung bzw. Automatisierung der anwenderseitigen Customizing-Prozesse durch die Integration von unterstützenden Werkzeugkonzepten. Die durchgeführte Analyse zeigt, dass die Mehrheit der analysierten Adaptionen ohne Programmierung umgesetzt werden kann.
- Der Umfang an Mechanismen, welche den Anwendungsexperten bei Durchführung einzelner Systemadaptionen aktiv lenken, bzw. die anwenderseitigen Adaptionsprozesse automatisieren, ist positiv mit der SaaS-Reife korreliert. Die aufgestellte Hypothese H4 wurde bestätigt. Die Analyse der Rangkorrelationskoeffizienten weist auf einen reifebedingten Anstieg der Rolle von Mechanismen der Prozesslenkung und -automation für das anwenderseitige Customizing hin. Bei den ausgereiften ERP-Systemen wird ein größeres Spektrum an Systemadaptionen durch die beiden Mechanismen unterstützt.

Die durchgeführte Untersuchung bietet Antworten auf die aufgestellten Untersuchungsfragen und ermöglicht die statistische Verifizierung der einzelnen Hypothesen. Die Untersuchungsergebnisse weisen auf einige architekturbedingte Customizing-Beschränkungen der einzelnen SaaS-Reifegrade bzw. SaaS-Reifegrad-abhängigen Tendenzen in der Entwicklung der cloudbasierten ERP-Systeme, bzw. in der Adaption der ERP-Systeme zu einem Cloud-Betrieb, hin.

Die ermittelten Zusammenhänge zwischen den untersuchten Indikatoren und die Systematisierung der durch die ERPaaS bereitgestellten Customizing-Konzepte wurden in der Tabelle 6.9 zusammengestellt.

**Tabelle 6.9:** Zusammenfassende Charakteristika der Customizing-Strategien in Abhängigkeit von der SaaS-Reife

<table>
<tr><th rowspan="2">Customizing-Strategie</th><th colspan="4">SaaS-Reife</th></tr>
<tr><th>R1</th><th>R2</th><th>R3</th><th>R4</th></tr>
<tr><td rowspan="4">Adaptionsansätze</td><td colspan="4">Mit dem Reifegrad ansteigende Bedeutung der Parametrisierung [CA2] und Komposition [CA1] und fallende Bedeutung der codebasierten Modifikation [CA9] des ERP-Systems</td></tr>
<tr><td colspan="2">Relativ große Bedeutung der codebasierten Ansätze; die Modifikation des Systemquellcodes [CA9] gegebenenfalls möglich, da keine Mechanismen der Mandantenfähigkeit auf der Anwendungsebene vorhanden</td><td colspan="2">Unterschiedliche Adaptionsansätze mit überwiegend codefreien Ansätzen; sehr große Rolle der Parametrisierung [CA2] und Komposition [CA1]; keine Anwendung der codebasierten Modifikation des Systemquellcodes [CA9]</td></tr>
<tr><td>Bedeutung einzelner Ansätze ähnlich wie bei konventionellen On-Premise-Systemen</td><td>Anstieg der Rolle von codefreien Customizing-Ansätzen (erweiterte Parametrisierbarkeit [CA2] und Komposition [CA1])</td><td>Rückang der Bedeutung von codebasierten Adaptionsansätzen [CA7-9] im Vergleich zu der II-Reife [R2]</td><td>Leichter Anstieg der Bedeutung von APIs [CA8] und externen Erweiterungen [CA7] im Vergleich zu der III-Reife [R3]</td></tr>
<tr><td rowspan="4">Adaptierbare Systemelemente</td><td colspan="4">Mit dem Reifegrad ansteigende Flexibilität im Bereich der Parametereinstellungen [SE2], Zusammenstellung der Systemmodule [SE1] und der Integration anderer Anwendungen des Geschäftsumfelds [SE8]</td></tr>
<tr><td colspan="4">Adaptierbarkeit der weiteren Systemelemente [SE3-7 und 9] oft sichergestellt, jedoch keine reifeabhängige Tendenz statistisch nachgewiesen; Umfang der Systemflexibilität stark von dem Service-Modell bzw. der PaaS-Kopplung abhängig</td></tr>
<tr><td colspan="2" rowspan="2">Anwenderseitige Adaption durch die Fähigkeiten der internen Tools begrenzt;<br>Die Individualisierung der Systemfunktionalität gegebenenfalls durch Inanspruchnahme dedizierter Dienstleistungen seitens des Systemanbieters möglich;<br>Bei den cloudfähigen ERP-Systemen i.d.R. begrenzte Adaptionsmöglichkeiten im Vergleich zu ihren konventionellen Gegenstücken im On-Premise-Modell</td><td>Häufigere Bereitstellung eines PaaS-gekoppelten ERP-Systems mit dedizierten Plattform-Diensten;<br>Bei den PaaS-gekoppelten Systemen erweiterte Adaptierbarkeit der funktionalen Komponenten – insbesondere [SE7-9] – im Vergleich zu den ERP-Systemen ohne PaaS (i.d.R. [R1-2])</td></tr>
<tr><td>Bei den cloudbasierten ERP-Systemen ohne PaaS allgemeine Tendenz in [SE1-2] und [SE8] sichtbar</td></tr>
<tr><td rowspan="4">Customizing-Werkzeuge</td><td colspan="4">Mit dem Reifegrad ansteigende Bedeutung von systemexternen Customizing-Tools [CW2] und cloudbasierten Diensten [CW3] im Rahmen der Customizing-Strategie</td></tr>
<tr><td colspan="2">Anwenderseitige Adaption mittels der systeminternen Customizing-Werkzeuge und der eingebauten Funktionen möglich [CW01]; geringe Verfügbarkeit der externen Adaptionswerkzeuge [CW2-3]</td><td colspan="2">Größere Verfügbarkeit der systemexternen Tools, insbesondere der dedizierten Entwicklungsumgebungen und PaaS-basierten Dienste; gegebenenfalls Bereitstellung internetbasierten Vertriebsplattformen für Systemerweiterungen (sogenannten Anwendungs-Stores)</td></tr>
<tr><td colspan="4">Mit dem Reifegrad zunehmende Unterstützung der anwenderseitigen Customizing-Prozesse durch Integration bzw. Bereitstellung dedizierter Mechanismen und Tools [AU3-4]</td></tr>
<tr><td colspan="2">Überwiegend Code-Editoren und Konfigurationsformulare; relativ große Anforderungen gegenüber den Anwendungsexperten; meistens Programmierkenntnisse und tiefgreifendes Verständnis der Systemstrukturen vorausgesetzt; Anwenderunterstützung hauptsächlich mittels der Mechanismen der passiven Beratung [AU1] und der aktiven Prozessanleitung [AU2]</td><td colspan="2">Weitgehende Erleichterung der anwenderseitigen Adaptierbarkeit durch Bereitstellung interaktiver Anwenderunterstützung [AU3], insbesondere der Assistenten und visueller Editoren (WYSIWYG); starke Reduktion der Anforderungen gegenüber den anwenderseitigen Administratoren – positive Wirkung auf die Beschleunigung der Adaptionsprozesse</td></tr>
</table>

# 7 Schlussbetrachtung

Im Rahmen dieses Abschnitts werden zuerst in Kapitel 7.1 die Forschungsbeiträge der vorliegenden Dissertationsarbeit für die Theorie und Praxis vorgestellt. Darauf aufbauend werden in Abschnitt 7.2 die Vorschläge für die weitere Forschung gemacht und ein Ausblick gegeben.

## 7.1 Forschungsbeitrag

Der Ausgangspunkt des vorliegenden Dissertationsprojektes war die Feststellung, dass die Flexibilität der cloudbasierten ERP-Systeme trotz hoher Relevanz bisher sehr unzureichend untersucht war. In diesem Kontext wurde insbesondere festgestellt, dass eine klare Bewertung der Customizing-Möglichkeiten der cloudbasierten ERP-Systeme durch eine Vielfalt der gängigen Bereitstellungskonzepte und aufgrund der anhaltenden Entwicklung des Cloud-Marktes zusätzlich erschwert ist, so dass ein systemübergreifender Vergleich der bereitgestellten Flexibilität nicht sichergestellt ist.

Daher verfolgte die vorliegende Arbeit das Ziel, einen Forschungsbeitrag zur Systematisierung der Customizing-Möglichkeiten der cloudbasierten ERP-Systeme zu leisten. Um eine Grundlage für die Entwicklung des Systematisierungsrahmens zu schaffen, wurde nach einer intensiven Literaturanalyse die Typologie der Customizing-Ansätze abgeleitet und eine Zusammenstellung der häufig auftretenden Customizing-Anforderungen ermittelt. Im Anschluss daran wurden die Reifegradmodelle für SaaS analysiert und die Zusammenhänge zwischen dem architektonischen Konzept und der Flexibilität der cloudbasierten Systeme diskutiert. Darauf aufbauend wurde ein Systematisierungskonzept entwickelt, welches das Fundament der empirischen Untersuchung darstellte.

Entlang der aufgestellten Forschungsfragen und der Hypothesen trug die Arbeit systematisch dazu bei, die in der Einleitung aufgezeigten Forschungsdefizite zu reduzieren und Grundlage unterschiedlicher Bewertung der Adaptierbarkeit zwischen Akademikern und Praktikern aufzuklären. Im Hinblick auf die verfolgten Forschungsziele widmete sich die Arbeit einer umfassenden Analyse der Customizing-Strategien mehrerer Systemanbieter; diese Analyse wurde anhand der hierfür konzipierten Machbarkeitsstudien und der diese komplementierenden umfragebasierten empirischen Untersuchung vorgenommen.

Auf Basis der Forschungserkenntnisse der explorativen Untersuchung, welche durch die Ergebnisse der vorgenommenen Expertenumfrage erweitert bzw. generalisiert wurden, wurden die aufgestellten Forschungsfragen beantwortet und die Thematik wurde in einen ganzheitlichen Systematisierungsrahmen eingeordnet.

### 7.1.1 Zusammenfassende Beantwortung der Forschungsfragen

Den zentralen Forschungsbeitrag des vorliegenden Dissertationsprojektes stellt die Systematisierung der Customizing- und Modifikationsmöglichkeiten der cloudbasierten ERP-Systeme anhand der Theorie und Praxis dar. Auf diese Weise stellt die Untersuchung eine Grundlage für die korrekte Interpretation der bereitgestellten Systemflexibilität in einem nicht ausgereiften Markt für cloudbasierte ERP-Systeme sicher. In diesem Abschnitt werden die wichtigsten

Erkenntnisse der Arbeit, bezogen auf die in der Einleitung aufgestellten Hauptforschungsfragen, zusammenfassend dargestellt.

HF 1: *Wie können die cloudbasierten ERP-Systeme adaptiert werden?*

Die analysierten cloudbasierten ERP-Systeme verfolgen verschiedene Customizing-Strategien, welche in der Entwurfsphase des Systems festgelegt wurden. Viele Systeme bieten eine kohärente Zusammenstellung unterschiedlicher Adaptionskonzepte an, um möglichst genau an die Anforderungen der Anwenderunternehmen angepasst werden zu können. Dabei ist der Einsatzbereich einzelner Customizing-Ansätze bei verschiedenen ERP-Systemen unterschiedlich. Daher erfordern Adaptionen, die in einem System mittels der Parametrisierung durchgeführt werden, bei anderen Systemen einen Einsatz von User-Exits bzw. codebasierten Erweiterungen. Im Allgemeinen spielen jedoch die codefreien Customizing-Ansätze – insbesondere die Parametrisierung und die komponentenbasierte Zusammenstellung – die wichtigste Rolle im Rahmen der anwenderseitigen Adaptionsprozesse.

Die Effektivität der einzelnen Adaptionskonzepte wurde im Rahmen der vorgenommenen Machbarkeitsstudien umfangreich demonstriert. Die Ergebnisse der Untersuchung beweisen, dass der bereitgestellte Flexibilitätsumfang von den Systemherstellern als wesentlicher Faktor wahrgenommen wird, welcher die Systemakzeptanz seitens der Anwenderunternehmen beeinflusst (vgl. Hohmann et al. 2013). Daher legen die Hersteller der cloudbasierten ERP-Systeme großen Wert auf die Bereitstellung effektiver, systeminterner Adaptionswerkzeuge, mit deren Hilfe erfolgskritische Systemanpassungen eigenständig – ohne Inanspruchnahme des Supports seitens der Systemanbieter – umgesetzt werden können.

Während viele der unternehmerischen Adaptionsanforderungen gerade mithilfe der im System vorhandenen eingebauten Adaptionswerkzeuge eigenständig umgesetzt werden können, werden die Möglichkeiten des anwenderseitigen Customizing häufig von dem Systemhersteller zusätzlich durch die Bereitstellung einer systemexternen Entwicklungsplattform erweitert. Des Weiteren stellen einige Systemhersteller internetbasierte Vertriebsplattformen für Systemerweiterungen bereit, welche eine bedarfsgerechte Erweiterung des standardisierten Funktionalitätsumfangs der cloudbasierten ERP-Systeme durch Einkauf und Installation systemdedizierter Erweiterungslösungen ermöglichen.

In diesem Kontext ist insbesondere eine Tendenz zur weitgehenden Erleichterung der anwenderseitigen Adaptionsprozesse ersichtlich. Es wird in der Regel auf das manuelle tabellenbasierte Customizing verzichtet, welches von konventionellen ERP-Systemen bekannt ist. Stattdessen werden die cloudbasierten ERP-Systeme von ihren Herstellern zunehmend mit dedizierten Mechanismen ausgestattet, welche die Adaptionsprozesse automatisieren oder den Anwendungsexperten bei der Durchführung der Systemadaptionen begleiten sowie bei der Entscheidungsfindung unterstützen. Diese Verschiebung des Fokus im anwenderseitigen Customizing im Vergleich zu den On-Premise-ERP-Systemen führt zur Reduktion des mit Customizing verbundenen Aufwands und in der Konsequenz zur Senkung der Adaptionskosten.

HF 2: *Wie wirkt sich das architektonische Konzept eines cloudbasierten ERP-Systems auf die bereitgestellte Systemflexibilität aus?*

Die Ergebnisanalyse hat einerseits gezeigt, dass bei den meisten cloudbasierten ERP-Systemen unterschiedliche Customizing-Möglichkeiten sichergestellt werden, welche über den Rahmen einer parameterbasierten Konfiguration hinausgehen. Andererseits werden auch signifikante Differenzen zwischen den Customizing-Strategien der ausgereiften und der nicht ausgereiften cloudbasierten ERP-Systeme identifiziert. Insbesondere wurden Differenzen in Bezug auf die Verwendung der codebasierten und codefreien Customizing-Ansätze aufgezeigt.

Gemäß den Forschungsergebnissen stellen Anbieter der stark ausgereiften ERP-Systeme in der Regel umfangreichere Parametrisierungs- und Integrationsmöglichkeiten als Anbieter der cloudfähigen Systeme sicher. Dabei ist auch bei den ausgereiften ERP-Systemen eine weitergehende Erleichterung der anwenderseitigen Customizing-Prozesse durch die Verankerung der dedizierten unterstützenden Werkzeugkonzepte (z. B. Assistenten) ersichtlich. Unter diesem Aspekt ist bemerkenswert, dass mit dem Anstieg der SaaS-Reife ein größerer Fokus seitens der Systemhersteller auf die Sicherstellung weitreichender Automatisierung bzw. werkzeugbasierter Unterstützung der anwenderseitigen Adaptionsprozesse gelegt wird. Im Vergleich zu den weniger ausgereiften Cloud-ERP-Systemen können die einzelnen Adaptionsanforderungen im Rahmen der ERP-Systeme der höheren SaaS-Reifegrade in der Regel effektiver und effizienter umgesetzt werden.

Die Ergebnisanalyse bestätigt weiterhin die architekturabhängigen Unterschiede im Spektrum der bereitgestellten Customizing-Werkzeuge. In diesem Zusammenhang werden die cloudnativen ERP-Systeme der höheren SaaS-Reifegrade überwiegend mit zusätzlichen systemexternen Werkzeugen ausgestattet, welche die Limitierungen der eingebauten Werkzeuge kompensieren. Demgegenüber stellen die nicht ausgereiften cloudfähigen ERP-Systeme in der Regel ausschließlich systeminterne Customizing-Werkzeuge zur Verfügung.

Im Allgemeinen unterscheiden sich jedoch die cloudbasierten ERP-Systeme, bezogen auf den Umfang der bereitgestellten Adaptierbarkeit, stark voneinander. Die Analyse der gewonnenen Forschungsergebnisse zeigt, dass die Tatsache, ob ein ERP-System in einem SaaS-Modell bereitgestellt wird oder an eine PaaS-Umgebung gekoppelt ist, einen bedeutenden Einfluss auf die bereitgestellte Systemflexibilität bzw. auf den Umfang der anwenderseitigen Anpassbarkeit hat.

Folglich sollen die cloudbasierten ERP-Systeme nicht als ein einheitliches Architekturkonzept verstanden werden. Zwecks Sicherstellung einer entsprechenden Vergleichbarkeit unterschiedlicher cloudbasierter ERP-Systeme sollten diese Systeme nach ihrer SaaS-Reife bzw. der zugrunde liegenden Cloud-Architektur differenziert werden.

### 7.1.2 Methodischer Beitrag

Die Neuartigkeit der Untersuchung besteht in der komplexen Betrachtung der Customizing-Möglichkeiten cloudbasierter ERP-Systeme hinsichtlich der Customizing-Ansätze, der adaptierbaren Systemelemente und der Customizing-Werkzeuge. Durch diese vielseitige Betrachtung leistet die Forschung einen bedeutenden Beitrag zum Verständnis der Customizing-Strategien differenzierter cloudbasierter ERP-Systeme und des bereitgestellten Flexibilitätsumfangs.

Während die bisherigen Studien in der Regel zwischen den codebasierten und den codefreien Systemadaptionen differenzieren, baut die vorgenommene Untersuchung auf einer detaillierteren Aufteilung der Customizing-Ansätze auf, die eine umfassende Analyse der einzelnen Customizing-Strategien ermöglicht. Der Grundstein hierfür wird durch eine intensive Literaturrecherche gelegt, welche die verfügbaren Typologien der Customizing-Ansätze identifiziert und hinsichtlich ihres Detaillierungsgrades sowie der Abgrenzung und Aktualität einzelner Ansätze auswertet.

Die vorgeschlagene Aufteilung der Customizing-Ansätze stellt einerseits eine erforderliche Vergleichsbasis für differenzierte cloudbasierte ERP-Systeme sicher, andererseits ermöglicht sie auch die Verschiebung der Bedeutung einzelner Ansätze zwischen den Cloud-ERP und konventionellen On-Premise-Systemen zu erkennen. Dadurch liefert die Studie einen ersten Baustein zur Entwicklung eines interdisziplinären, systemübergreifenden Vergleichsmodells zur

Bewertung der Systemflexibilität, welches unter anderem durch die Studie von Uppström et al. (2015, S. 4227) gefordert wurde.

Ein weiterer bedeutender Aspekt, welcher die vorliegende Untersuchung von bisherigen Studien unterscheidet, ist die detailliertere Analyse der bereitgestellten Systemflexibilität, bezogen auf die konkreten Systemelemente und die erfolgskritischen Systemadaptionen. Der besondere Mehrwert entsteht durch die Systemelement-bezogene Aufstellung von Adaptionsanforderungen, wobei diese als Ergebnis einer ausführlichen Literaturanalyse der wissenschaftlich dokumentierten Implementierungen von ERP-Systemen entstanden ist. Der erarbeitete Anforderungskatalog bietet eine Grundlage für die Konstruktion der Fallstudien für die empirische Untersuchungsphase und stellt einen Bewertungsrahmen bezüglich der Flexibilität einzelner Systemelemente dar. Durch den hohen Detaillierungsgrad und Praxisbezug des Adaptionskatalogs werden die Defizite vieler anderer Untersuchungen beseitigt, welche die bereitgestellte Systemflexibilität zumeist nur konzentriert auf die einzelnen Architekturschichten betrachten.

Eine weitere Neuartigkeit des Systematisierungskonzeptes besteht in der Einbeziehung der SaaS-Reife, welche die architekturbezogene Konformität mit den Anforderungen des SaaS-Modells beschreibt, in das Analysemodell sowie in einer statistischen Auswertung der Zusammenhänge zwischen dem architektonischen Konzept und der Customizing-Strategie. Der architekturübergreifende Vergleich der Customizing-Strategien unterschiedlicher ERP-Systeme, die in differenzierten Cloud-Bereitstellungs- und Service-Modellen bereitgestellt werden, ermöglicht es die systemübergreifenden Tendenzen hinsichtlich der bereitgestellten Adaptierbarkeit zu identifizieren und die architekturbedingten Differenzen in dem bereitgestellten Flexibilitätsumfang zu erkennen. Die Verwendung unterschiedlicher Betrachtungsebenen in Verbindung mit dem Reifekonzept trägt wesentlich zur Verbesserung der Analysetransparenz bei und ermöglicht eine ganzheitliche Betrachtung der Problemstellung. Dadurch hebt sich das vorgeschlagene Untersuchungskonzept von den bisherigen Studien ab, die entweder auf einem ausgewählten ERP-System (z. B. Elragal und Kommos 2012; Seethamraju 2015) bzw. einem konkreten Cloud-Bereitstellungsmodell beruhen (z. B. Hao et al. 2012, S. 426).

Einen bedeutenden Beitrag zur Schließung der bestehenden Forschungslücke im Bereich des Customizing der cloudbasierten ERP-Systeme leisten die umfangreichen Machbarkeitsstudien. Durch die Validierung der systembezogenen Adaptierbarkeit und Erweiterbarkeit anhand konkreter, praxisnaher Beispielszenarien wird insbesondere ein, den bisherigen wissenschaftlichen Studien häufig fehlender Praxisbezug hergestellt. Aus praktischer Perspektive können die einzelnen Machbarkeitsstudien als Referenz für die Einschätzung der Anforderungen gegenüber den Anwendungsexperten genutzt werden, bzw. zur Abschätzung des mit der Umsetzung konkreter Adaptionen verbundenen Aufwands dienen. Weiterhin weisen die Ergebnisse das Potenzial auf, in Handlungsempfehlungen zur Gestaltung effektiver und anwenderkonformer Customizing-Werkzeuge und -Konzepte übertragen zu werden. Vor allem weisen die Erkenntnisse der durchgeführten Machbarkeitsstudien auf die Notwendigkeit einer differenzierten Betrachtung der Customizing- und Modifikationsmöglichkeiten von cloudbasierten ERP-Systemen hin, welche durch unterschiedliche SaaS-Reife gekennzeichnet sind.

Anhand der vorgenommenen Machbarkeitsstudien und der empirischen Untersuchung, die systematisch zur Validierung der Adaptierbarkeit einzelner Systemelemente beigetragen haben, wurde der adaptierbare Systemumfang ermittelt, welcher im Rahmen eines anwenderseitigen Customizing angepasst werden kann. Die Gültigkeit der gewonnenen Erkenntnisse und ihre Übertragbarkeit auf andere cloudbasierte ERP-Systeme innerhalb einzelner architektonischer Reifegrade wurden durch die Ergebnisse der statistischen Auswertung untermauert. Dadurch schafft die vorgenommene Untersuchung eine häufig geforderte Vergleichsbasis für die Einschätzung der bereitgestellten Flexibilität differenzierter Cloud-ERP (vgl. Mijač et al.

2013, S. 139). Neben der rein theoretischen Relevanz stellt die Systematisierung einen Bezugsrahmen dar, der als praktische Hilfestellung bei der Bewertung der Systemflexibilität differenzierter Cloud-ERP verwendet werden kann.

Durch die Analyse der systembezogenen Adaptierbarkeit unter Einbeziehung der architektonischen Konzepte, welche sich in der SaaS-Reife widerspiegeln, zeigt die vorgenommene Untersuchung eine mögliche Erklärungsgrundlage für Diskrepanzen zwischen den Akademikern und Praktikern hinsichtlich der Bewertung der Adaptierbarkeit der cloudbasierten ERP-Systeme auf. Des Weiteren stellen die Ergebnisse der Untersuchung eine Grundlage für die korrekte Interpretation bzw. Bewertung der bereitgestellten Systemflexibilität in einem nicht ausgereiften Markt für cloudbasierte ERP-Systeme sicher.

## 7.2 Anschlussforschung und Ausblick

Die vorliegende Dissertation thematisiert die Adaptierbarkeit der cloudbasierten ERP-Systeme. Im Rahmen der Forschung, die aus einer empirischen Untersuchung und nachfolgender Expertenumfrage bestand, wurden Customizing-Strategien von vierunddreißig cloudbasierten ERP-Systemen analysiert. Einerseits trägt die Arbeit dazu bei, die gegenwärtig bestehenden Forschungslücken zu schließen, andererseits weist sie auf den weiteren Forschungsbedarf und auf die Forschungsfragen hin, welchen im Rahmen einer Anschlussforschung nachgegangen werden sollte.

Mit ihrem neuartigen Konzept einer nach SaaS-Reife differenzierten Betrachtung ist die Untersuchung als erster Versuch einer architekturbezogenen Systematisierung und Bewertung der Customizing-Möglichkeiten anzusehen. Im Rückblick auf die Forschungserkenntnisse ist anzumerken, dass der Markt für cloudbasierte ERP-Systeme nicht ausgereift ist und einer langsamen Weiterentwicklung unterliegt. Trotz des steigenden Interesses an der SaaS seitens der ERP-Systemhersteller ist die Anzahl der cloudbasierten ERP-Systeme im Vergleich zu den konventionellen On-Premise-Lösungen immer noch relativ gering. Insbesondere fehlen ausgereifte, cloudnative ERP-Systeme, die konform mit dem SaaS-Modell entwickelt werden. Die Mehrheit der heutzutage in der Cloud bereitgestellten ERP-Systeme stellen altbewährte Lösungen dar, die in unterschiedlichem Umfang cloudifiziert wurden. Aufgrund der Marktcharakteristika erscheinen weitere Studien notwendig, welche die Customizing- und Modifikationsmöglichkeiten der cloudbasierten ERP-Systeme über einen längeren Zeitraum hinweg untersuchen.

Ausgehend von dem abgeleiteten Systematisierungsmodell und den konstruierten Machbarkeitsstudien, die im Rahmen der vorliegenden Arbeit beschrieben wurden, kann in der anschließenden Forschung die Flexibilität der cloudbasierten ERP-Systeme an einer größeren Stichprobe untersucht werden. Dies eröffnet die Möglichkeit für die Exploration weiterer Customizing-Strategien, was in der Konsequenz zum Ausbau und zur Fundierung bestehender Forschungserkenntnisse beitragen kann.

Die vorgenommene Systematisierung soll primär als ein Referenzrahmen für die anknüpfende Forschung dienen. Das Ziel der vorliegenden Arbeit bestand daher in einer wissenschaftlich objektiven, systemübergreifenden Bewertung der Adaptionsmöglichkeiten der cloudbasierten Systeme und nicht in der Herleitung der Richtlinien für eine unternehmensgerechte Systembewertung bzw. -auswahl. Die Einbeziehung konkreter ERP-Systeme und praxisnaher Adaptionsanforderungen in die Analysen stellte dabei eine Methode zur Identifizierung der systemübergreifenden Tendenzen in der bereitgestellten Systemflexibilität und zur Ermittlung der Zusammenhänge mit dem architektonischen Systemkonzept dar. Allerdings könnten die einzelnen Betrachtungsebenen des konstruierten Systematisierungsmodells im Rahmen der anschließenden Forschung mit dedizierten Gewichtungsfaktoren und Kennzahlenkonzepten

erweitert werden, mit deren Hilfe eine zielgerichtete Systembewertung im Hinblick auf die unternehmensindividuellen Adaptionsanforderungen möglich wäre.

Das bestehende Systematisierungsmodell kann im Rahmen der anschließenden Forschung an neue Entwicklungen des IT-Marktes weiter angepasst werden. Die steigende Bedeutung der system- und prozessübergreifenden Integration im Kontext der 4. industriellen Revolution (Industrie 4.0), deren Grundlage eine Vernetzung von ERP-Systemen mit mobilen Endgeräten, Webanwendungen und Maschinen bildet, liefert die ersten Ansatzpunkte für den möglichen Erweiterungsbedarf des bestehenden Kataloges der Adaptionsanforderungen in der Zukunft (vgl. Klink et al. 2016). Darüber hinaus kann das bestehende Modell im Rahmen der Anschlussforschung durch Konstruktion dedizierter Szenarien und anhand der Durchführung zusätzlicher Machbarkeitsstudien, welche auf weitere Systemelemente ausgerichtet sind, systematisch ausgebaut werden.

Die durchgeführten Machbarkeitsstudien bestätigen die Bedeutung einer anwenderseitigen Adaptierbarkeit der cloudbasierten ERP-Systeme im Rahmen eines On-Demand-Selfservices. Die bestehende Untersuchung könnte durch eine vertiefende Analyse der Adaptionsmöglichkeiten erweitert werden, die im Rahmen der bestehenden Customizing-Strategie ausschließlich seitens des Systemherstellers bzw. seiner Partner vorgenommen werden könnte.

Schließlich kann aufbauend auf den Erkenntnissen dieser Untersuchung, welche ausschließlich auf vollständig cloudbasierte ERP-Systeme ausgerichtet war, ein Systematisierungsrahmen erarbeitet werden, der eine vergleichende Bewertung der Adaptierbarkeit von ERP-Systemen in differenzierten Bereitstellungsmodellen (z. B. On-Premise, On-Demand, Hosting) ermöglicht. Im Hinblick auf eine domänenübergreifende Betrachtung der Customizing-Strategien wäre es auch angebracht, das bestehende Forschungsspektrum um ERP-Systeme in hybriden Bereitstellungsmodellen zu erweitern, deren Architekturschichten in differenzierten Bereitstellungsmodellen angeboten werden.

Die bereitgestellte Flexibilität eines ERP-Systems wird oft in der Literatur als wesentlicher Erfolgsfaktor sowohl aus Sicht des Anwender- als auch des Anbieterunternehmens angesehen, welcher entsprechend für die Erzielung der Wettbewerbsvorteile (vgl. Zach und Munkvold 2012) und für die Sicherstellung entsprechender Systemakzeptanz (vgl. Lechesa et al. 2012) verantwortlich sein kann. In diesem Zusammenhang bemerkenswert bleibt, dass die Anforderungen hinsichtlich der Systemflexibilität seitens der Anwenderunternehmen in letzter Zeit angestiegen sind (vgl. Uppström et al. 2015, S. 4222). Diese Erkenntnisse stellen eine mögliche Erklärungsgrundlage für die heute erkennbare intensive Forschung im Bereich der Adaptierbarkeit im mandantenfähigen Cloud-Umfeld dar (vgl. Walraven et al. 2016; Makki et al. 2016; Kabbedijk 2014).

Angesichts der anhaltenden Weiterentwicklung der Cloud-Technologien und des langsam reifenden Marktes für cloudbasierte ERP-Systeme ist die vorgenommene Systematisierung nicht als abschließend feststehendes, unveränderbares Konstrukt zu sehen. Vielmehr liefern die Ergebnisse der vorgenommenen Untersuchung einen Bezugsrahmen und weisen auf interessante Anhaltspunkte für weiterführende Forschung im Bereich der Adaptierbarkeit von cloudbasierten ERP-Systemen hin.

# Abkürzungsverzeichnis

ACM ..... Association for Computing Machinery
AISeL ..... Association for Information Systems eLibrary
API..... Application Programming Interface
ASP..... Application Service Provider
ATP..... Available-to-Promise
B2B..... Business-to-Business
BITKOM ..... Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V.
CSV ..... Comma-separated values, ein Dateiformat
ERP..... Enterprise Resource Planning
ERPaaS ..... ERP-as-a-Service
FF..... Forschungsfrage
HF ..... Hauptforschungsfrage
HTML ..... HyperText Markup Language
HTTP ..... HyperText Transfer Protocol
IaaS ..... Infrastructure-as-a-Service
IDE..... Integrated Development Environment
IEEE ..... Institute of Electrical and Electronics Engineers
IT ..... Information Technology
KPI..... Key Performance Indicator
MWU-Test..... Mann-Whitney-U-Test
NIST ..... National Institute of Standards and Technology
PaaS ..... Platform-as-a-Service
PHP..... Hypertext Preprocessor
SaaS ..... Software-as-a-Service
SOA ..... Serviceorientierte Architektur
SOAP..... Simple Object Access Protocol
SPSS ..... Statistical Product and Service Solution
SQL..... Structured Query Language
UI ..... User Interface
URL ..... Uniform Resource Locator
WYSIWYG ..... What You See Is What You Get
XaaS ..... Everything-as-a-Service
XML ..... Extensible Markup Language
XML-RPC ..... XML Remote Procedure Call

# Literaturverzeichnis


Ackerman und Turowski 2006

Ackermann, J.; Turowski, K.: Zur Rolle von Parametrisierung bei der fachlichen Anpassung betrieblicher Softwarekomponenten. In: Schelp, J.; Winter, R.; Frank, U.; Rieger, B.; Turowski, K. (Hrsg.): Integration, Informationslogisitik und Architektur. Proceedings DW2006, 21. - 22. Sept. 2006, Friedrichshafen, Lecture Notes in Informatics (LNI) P-90. Bonn: Gesellschaft für Informatik, 2006, S. 341–359.

Adobe Systems 2017

Adobe Systems, Inc: Adobe LiveCycle Designer ES4 [Internet]. 2017 [Abgerufen am 12.03.2017]. URL: http://www.adobe.com/de/products/livecycle/tools/designer.html

Al-Shardan und Ziani 2015

Al-Shardan, M. M.; Ziani, D.: Configuration as a Service in Multi-Tenant Enterprise Resource Planning System. In: Lecture Notes on Software Engineering 3 (2015), Nr. 2, S. 95–100.

Andrikopoulos et al. 2013

Andrikopoulos, V.; Binz, T.; Leymann, F.; Strauch, S.: How to Adapt Applications for the Cloud Environment. In: Computing 95 (2013), Nr. 6, S. 493–535.

Arbnor und Bjerke 2008

Arbnor, I.; Bjerke, B.: Methodology for Creating Business Knowledge. 3rd Edition. London: SAGE, 2008.

Arif et al. 2010

Arif, F.; Kanchymalay, K.; Suryana, N.; Krishnan, R.; Hashim, U. R.; Ismail, N. H.: Measuring the Effect of Customization in Influencing the Success of ERP Implementation. In: The International Conference on Industrial Engineering and Business Management (ICIEBM) 2010. Yogyakarta, Indonesia, 2010, S. 371–376.

Arya et al. 2010

Arya, P. K.; Venkatesakumar, V.; Palaniswami, S.: Configurability in SaaS for an Electronic Contract Management Application. In: Bulucea, C. A.; Kalamani, N.; Mastorakis, N.; Mladenov, V.: Recent Advances in Networking, VLSI & Signal Processing, Proceedings of the 12th International Conference on Networking, VLSI and Signal Processing (ICNVS'10), University of Cambridge, UK, February 20 - 22, 2010. Cambridge: WSEAS Press, 2010, S. 210–216.

Beatty und Williams 2006

Beatty, R. C.; Williams, C. D.: ERP II: Best Practices for Successfully Implementing an ERP Upgrade. In: Communications of the ACM 49 (2006), Nr. 3, S. 105–109.

Becker et al. 2009

Becker, J.; Knackstedt, R.; Pöppelbuß, J.: Entwicklung von Reifegradmodellen für das IT-Management – Vorgehensmodell und praktische Anwendung. In: Wirtschaftsinformatik 51 (2009), Nr. 3, S. 249–260.

Becker et al. 2003

Becker, J.; Holten, R.; Knackstedt, R.; Niehaves, B.: Forschungsmethodische Positionierung in der Wirtschaftsinformatik: Epistemologische, ontologische und linguistische Leitfragen. Arbeitsberichte des Instituts für Wirtschaftsinformatik, 2003, Nr. 93. Münster: Westfälische Wilhelms-Universität Münster, 2003.

Becker et al. 2015
Becker, M.; Berger, S.; Tsilimandou, M.: Ausgewählte Cloud-ERP-Lösungen im Vergleich: Für welches Produkt soll ich mich entscheiden?. Norderstedt: BoD–Books on Demand, 2015.

Bedner 2013
Bedner, M.: Cloud Computing: Technik, Sicherheit und rechtliche Gestaltung. Forum Wirtschaftsrecht, Band 14. Institut für Wirtschaftsrecht. Kassel, Universität Kassel, Institut für Wirtschaftsrecht, Dissertation, 2013.

Beheshti 2006
Beheshti, H.M.: What Managers Should Know about ERP/ERP II. In: Management Research News (2006), Nr. 29, S. 184–193.

Beijsterveld und Groenendaal 2015
Beijsterveld, J.A.A. van; Groenendaal, W.J.H. van: Solving Misfits in ERP Implementations by SMEs. In: Information Systems Journal 4 (2015), Nr. 26, S. 369–393.

Bento et al. 2015
Bento, A.; Bento, R.; Bento, A.: How Fast Are Enterprise Resource Planning (ERP) Systems Moving to the Cloud. In: Journal of Information Technology Management 26 (2015), Nr. 4, S. 35–44.

Bezemer und Zaidman 2010a
Bezemer, C.-P.; Zaidman, A.: Challenges of Reengineering into Multi-Tenant SaaS Applications. Delft University of Technology, Software Engineering Research Group, Technical Report Series, Report TUD-SERG-2010-012. Delft, Netherlands: Delft University of Technology, 2010, S. 1–12.

Bezemer und Zaidman 2010b
Bezemer, C.-P.; Zaidman, A.: Multi-Tenant SaaS Applications: Maintenance Dream or Nightmare? In: Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE). Antwerp, Belgium: ACM, 2010, S. 88–92.

Bhamangol et al. 2011
Bhamangol, B.N.; Nandavadekar, V.D.; Khilari, S.H.: Enterprise Resource Planning (ERP) System in Higher Education: A Literature Review. In: International Journal of Management Research and Development 1 (2011), Nr. 1, S. 1–7.

Borovskiy und Zeier 2009
Borovskiy, V.; Zeier, A.: Enabling Enterprise Composite Applications on Top of ERP Systems. In: 2009 IEEE Asia-Pacific Services Computing Conference (APSCC), December 7 - 11, 2009, Biopolis, Singapore. Piscataway: IEEE, 2009, S. 492–497.

Bortz und Döring 2006
Bortz, J.; Döring, N.: Forschungsmethoden und Evaluation für Human- und Sozialwissenschaftler. 4. Auflage. Berlin Heidelberg: Springer, 2006.

Boslaugh 2012
Boslaugh, S.: Statistics in a Nutshell. 2nd Edition. Sebastopol: O'Reilly Media, 2012.

Brehm et al. 2001
Brehm, L.; Heinzl, A.; Markus, M.L.: Tailoring ERP Systems: A Spectrum of Choices and their Implications. In: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS), Maui, Hawaii, January 3 - 6, 2001. Piscataway: IEEE, 2001, S. 8017–8026.

Brereton et al. 2007
Brereton, P.; Kitchenham, B.A.; Budgen, D.; Turner, M.; Khalil, M.: Lessons from Applying the Systematic Literature Review Process within the Software Engineering Domain. In: Journal of Systems and Software 80 (2007), Nr. 4, S. 571–583.

Brown und Nyarko 2012
Brown, C.W.; Nyarko, K.: Software as a Service (SaaS). In: Bento, A.M.; Aggarwal, A.K. (Hrsg.): Cloud Computing Service and Deployment Models: Layers and Management. Hershey: IGI Global, 2012, S. 50–69.

Business-Software 2015
Business-Software.com: Top 20 Enterprise Resource Planning Software Report – Comparison of the Leading ERP Software Vendors (Edition 2015) [Internet]. 2015 [Abgerufen am 10.02.2017]. URL: http://www.business-software.com/offer/top-20-erp-software/.

Buyya et al. 2009
Buyya, R.; Yeo, C. S.; Venugopal, S.; Broberg, J.; Brandic, I.: Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. In: Future Generation Computer Systems 25 (2009), Nr. 6, S. 599–616.

Callejas und Terzi 2013
Callejas, J. F.; Terzi, C.: Review of Enterprise Resource Planning (ERP) Systems in United Nations Organizations. Report JIU/REP/2013/1. Geneva: United Nations Joint Inspection Unit, 2013.

Cao 2010
Cao, L.: The Misalignment between Packaged Enterprise Systems and Chinese Context: A Context Study of Packaged ES Adoption in China. In: Proceedings of the Pacific Asia Conference on Information Systems, PACIS 2010, Taipei, Taiwan, July 9 - 12, 2010. Taipei: AISeL, 2010, S. 1596–1603.

Chen et al. 2009
Chen, C. C.; Law, C.; Yang, S. C.: Managing ERP Implementation Failure: A Project Management Perspective. In: IEEE Transactions on Engineering Management 56 (2009), Nr. 1, S. 157–170.

Chen et al. 2011
Chen, X.; Srivastava, A.; Sorenson, P.: Toward a QoS-Focused SaaS Evaluation Model. In: Ahson, S. A.; Ilyas, M. (Hrsg.): Cloud Computing and Software Services: Theory and Techniques. Boca Raton: CRC Press, 2011, S. 389–407.

Chen et al. 2015
Chen, C. S.; Liang, W. Y.; Hsu, H. Y.: A Cloud Computing Platform for ERP Applications. In: Applied Soft Computing 27 (2015), S. 127–136.

Chong et al. 2008
Chong, F.; Carraro, G.; Wolter, R.: Multi-Tenant Data Architecture [Internet]. 2008 [Abgerufen am 11.03.2017]. URL: http://msdn.microsoft.com/en-us/en%20-%20us/library/aa479086.aspx.

Chong und Carraro 2006
Chong, F.; Carraro, G.: Architecture Strategies for Catching the Long Tail [Internet].2006 [Abgerufen am 11.03.2017]. URL: http://msdn.microsoft.com/en-us/library/aa479069.aspx.

Davenport 1998
Davenport, T. H.: Putting the Enterprise into the Enterprise System. In: Harvard Business Review 76 (1998), Nr. 4, S. 121–131.

Davis 2005
Davis, A.: ERP Customization Impacts on Strategic Alignment and System Agility. In: Proceedings of the 2005 Southern Association of Information Systems Conference, 2005, S. 249–255.

Dittrich und Vaucouleur 2009
Dittrich, Y.; Vaucouleur, S.; Giff, S.: ERP Customization as Software Engineering: Knowledge Sharing and Cooperation. In: IEEE Software 26 (2009), Nr. 6, S. 41–47.

Dixit und Prakash 2011
Dixit, A. K.; Prakash, O.: A Study of Issues Affection ERP Implementation in SMEs. In: International Refereed Research Journal 2 (2011), Nr. 2, S. 77–85.

Doedt und Steffen 2011
Doedt, M.; Steffen, B.: Requirement-Driven Evaluation of Remote ERP-System Solutions: A Service-oriented Perspective. In: Proceedings of the 2011 34th IEEE Software Engineering Workshop (SEW). Limerick: IEEE, 2011, S. 57–66.

Duan et al. 2013
Duan, J.; Faker, P.; Fesak, A.; Stuart, T.: Benefits and Drawbacks of Cloud-based versus Traditional ERP Systems. In: Groenendaal, W.J.H. van (Hrsg.): Proceedings of the 2012-13 Course on Advanced Resource Planning. Tilburg: Tilburg University, 2013, S. 1–17.

Eggert et al. 2013
Eggert, S.; Schröder, E.; Stritzel, M.: Besonderheiten von Cloud-ERP-Systemen. In: ERP Management 4/2013: Betriebsformen moderner Systeme 9 (2013), Nr. 4, S. 24–26.

Eichler und Dostál 2012
Eichler, Z.; Dostál, M.: Adaptive User Interface Personalization in ERP Systems. In: Abramowicz, W.; Dominique, J.; Węcel, K. (Hrsg.): Business Information Systems Workshops, BIS 2012 International Workshops and Future Internet Symposium, Vilnius, Lithuania, May 21 - 23, 2012 Revised Papers. Berlin, Heidelberg: Springer, 2012, S. 49–60.

Elragal und Kommos 2012
Elragal, A.; Kommos, M.E.: In-House versus In-Cloud ERP Systems: A Comparative Study. In: Journal of Enterprise Resource Planning Studies 2012 (2012), S. 1–17.

Esteves und Pastor 1999
Esteves, J.; Pastor, J.: An ERP Life-cycle-based Research Agenda. In: Eder, J.; Maiden, N.; Missikoff, M. (Hrsg.): Proceedings of the First International Workshop on Enterprise Management Resource and Planning Systems. Atlanta: CAIS, 1999, S. 359–371.

Fan und Yan 2010
Fan, W.; Yan, Z.: Factors Affecting Response Rates of the Web Survey: A Systematic Review. In: Computers in Human Behavior 26 (2010), Nr. 2, S. 132–139.

Fiaidhi et al. 2012
Fiaidhi, J.; Bojanova, I.; Zhang, J.; Zhang, L.J.: Enforcing Multitenancy for Cloud Computing Environments. In: IT Professional 14 (2012), Nr. 1, S. 16–18.

Frick 2008
Frick, N.: Künftige Anforderungen an ERP-Systeme: Deutsche Anbieter im Fokus. In: Arbeitsberichte aus dem Fachbereich Informatik Nr. 11/2008. Institut für Wirtschafts- und Verwaltungsinformatik, Universität Koblenz-Landau (2008). Koblenz: Universität Koblenz-Landau, 2008.

Galešic und Bosnjak 2009
Galešic, M.; Bosnjak, M.: Effects of Questionnaire Length on Participation and Indicators of Response Quality in a Web Survey. In: Public Opinion Quarterly 73 (2009), Nr. 2, S. 349–360.

Gargeya und Brady 2005
Gargeya, V.B.; Brady, C.: Success and Failure Factors of Adopting SAP in ERP System Implementation. In: Business Process Management Journal 11 (2005), Nr. 5, S. 501–516.

Gattiker und Goodhue 2002
Gattiker T.F.; Goodhue D.L.: Software-driven Changes to Business Processes: An Empirical Study of Impacts of Enterprise Resource Planning (ERP) Systems at the Local Level. In: International Journal of Production Research 40 (2002), Nr. 18, S. 4799–4814.

Gerhardter und Ortner 2013
Gerhardter, A.; Ortner, W.: Flexibility and Improved Resource Utilization Through Cloud Based ERP Systems: Critical Success Factors of SaaS Solutions in SME. In: Piazolo, F.; Felderer, M. (Hrsg.): Innovation and Future of Enterprise Information Systems: ERP Future 2012 Conference, Salzburg, Austria, November 2012, Revised Papers, Lecture Notes in Information Systems and Organisation 4. Berlin Heidelberg: Springer, 2013, S. 171–182.

Google 2017a
Google, Inc: Google Developers - Google Maps JavaScript API [Internet]. 2017a [Abgerufen am 11.03. 2017]. URL: https://developers.google.com/maps/documentation/javascript/examples/geocoding-simple.

Google 2017b
Google, Inc: Google Maps [Internet]. 2017b [Abgerufen am 12.01.2017]. URL: https://maps.google.com/.

Grabot et al. 2013
Grabot, B.; Houé, R.; Lauroua, F.; Mayère, A.: Introducing "2.0" Functionalities in an ERP. In: Emmanouilidis, C.; Taisch, M.; Kiritsis D. (Hrsg.): Advances in Production Management Systems. Competitive Manufacturing for Innovative Products and Services, IFIP WG 5.7 International Conference, APMS 2012, Rhodes, Greece, September 24 - 26, 2012, Revised Selected Papers, Part II. Berlin Heidelberg: Springer, 2013, S. 104–111.

Graphiq 2015
Graphiq, Inc: SoftwareInsider: Compare Enterprise Resource Planning Software [Internet]. 2015 [Abgerufen am 12.09.2015]. URL: http://erp.softwareinsider.com/.

Grobman 2008
Grobman, J.: ERP-Systeme On Demand: Chancen, Risiken, Anforderungen, Trends. Hamburg: Diplomica Verlag, 2008.

Grubisic 2014
Grubisic, I.: ERP in clouds or still below. In: Journal of Systems and Information Technology 16 (2014), Nr. 1, S. 62–76.

Guo et al. 2007
Guo, C.J.; Sun, W.; Huang, Y.; Wang, Z.H.; Gao, B.: A Framework for Native Multi-Tenancy Application Development and Management. In: The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC-EEE 2007), Tokyo, 2007. Tokyo: IEEE, 2007, S. 551–558.

Guo et al. 2011
Guo, C.J.; Sun, W.; Jiang, Z.B.; Huang, Y.; Gao, B.; Wang, Z.H.: Study of Software as a Service Support Platform for Small and Medium Businesses. In: Agrawal, D.; Candan, K.S.; Li, W.S. (Hrsg.): New Frontiers in Information and Software as Services, Service and Application Design Challanges in the Cloud, Lecture Notes in Business Information Processing 74. Berlin Heidelberg: Springer, 2011, S. 1–30.

Hahn 2006
Hahn, H.: Die Etablierung von Märkten für Fachkomponenten: Eine vergleichende Analyse zur Beschaffung betrieblicher Anwendungssoftware und alternativer Güterkategorien. Augsburg, Universität Augsburg, Dissertation, 2006.

Haines 2009
Haines, M.N.: Understanding Enterprise System Customization: An Exploration of Implementation Realities and the Key Influence Factors. In: Information Systems Management 26 (2009), Nr. 2, S. 182–198.

Hamerman et al. 2015
Hamerman, P.D.; Andrews, C.; Miller, J.; Glazer, L.: The SaaS ERP Applications Landscape. Growing Demand Creates Vendor Melee In The Cloud As Traditional Vendors And SaaS Pure Plays Face Off. Forrester Report. Cambridge: Forrester Research Inc., 2015.

Hamerman et al. 2017
Hamerman, P.D.; Andrews, C.; Miller, J.; Seguin, B.; Reese, A.: Vendor Landscape: SaaS ERP Applications, 2017. The Shift To SaaS ERP Accelerates As Businesses Adapt To Digital Transformation. Forrester Report. Cambridge: Forrester Research Inc., 2017.

Hao et al. 2012
Hao, Y.; Juell-Skielse, G.; Uppström, E.: Cloud ERP Development Process Model from the Perspective of User Organizations. In: Møller, C.; Chaudhry, S.S. (Hrsg.): Advances in Enterprise Information Systems II. Proceedings of the Fifth International Conference on Research and Practical

Issues of Enterprise Information Systems (CONFENIS 2011), Aalborg, Denmark, October 16 - 18, 2011. London: CRC Press, 2012, S. 407–428.

Hausen 2017
Hausen, H.: silicon.de: Generationenwechsel im ERP-Markt [Internet]. 2016 [Abgerufen am 12. 03.2017]. URL: http://www.silicon.de/41620930/generationswechsel-im-erp-markt/.

Hecht et al. 2011
Hecht, S.; Wittges, H.; Krcmar, H.: IT Capabilities in ERP Maintenance - A Review of the ERP Post-Implementation Literature. In: Proceedings of the 19th European Conference on Information Systems, Helsinki, Finland. AISeL, 2011.

Hedman und Xiao 2016
Hedman, J.; Xiao, X.: Transition to the Cloud: A Vendor Perspective. In: 2016 49th Hawaii International Conference on System Sciences (HICSS), Koloa, Kauai, January 5 - 8, 2016. Washington: IEEE, 2016, S. 3989–3999.

Heng et al. 2012
Heng, L.; Dan, Y.; Xiaohong, Z.: Survey on Multi-Tenant Data Architecture for SaaS. In: International Journal of Computer Science Issues 9 (2012), Nr. 6, S. 198–204.

Henkes 2016
Henkes, H.: SaaS treibt den Markt für Public-Cloud-Services. Computerwoche [Internet]. 2016 [Abgerufen am 12.03.2017]. URL: http://www.computerwoche.de/a/saas-treibt-den-markt-fuer-public-cloud-services,3226679.

Hestermann et al. 2014
Hestermann, C.; Montgomery, N.; Pang, C.; Guay, M.: Gartner: Magic Quadrant for Single-Instance ERP for Product-Centric Midmarket Companies [Internet]. 2014 [Abgerufen am 12.03.2017]. URL: http://www.gartner.com/technology/reprints.do?id=1-25FCRSP&ct=141202&st=sg.

Hevner et al. 2004
Hevner, A. R.; March, S. T.; Park, J.; Ram, S.: Design Science in Information Systems Research. In: MIS Quarterly 28 (2004), Nr. 1, S. 75–105.

Hofmann und Woods 2010
Hofmann, P.; Woods, D.: Cloud Computing: The Limits of Public Clouds for Business Applications. In: IEEE Internet Computing 14 (2010), Nr. 6, S. 90–93.

Hohmann et al. 2013
Hohmann, P.; Kunz, N.; Özdil, E.: Herausforderungen an betriebswirtschaftliche Applikationen – insbesondere ERP in der Cloud. In: Barton, T.; Erdlenbruch, B.; Herrmann, F.; Müller, C.; Guckert, M.; Ritz, H. (Hrsg.): Herausforderungen an die Wirtschaftsinformatik: Integration und Konnexion; Tagungsband zur 26. AKWI-Jahrestagungvom 15. bis 18.09.2013 an der Technischen Hochschule Mittelhessen. Berlin: Verlag News & Media, 2013, S. 253–264.

Hufgard und Krüger 2012
Hufgard, A.; Krüger, S.: SAP Business ByDesign Geschäftsprozesse, Technologie und Implementierung anschaulich erklärt. Bonn: SAP Press, 2012.

Hustad et al. 2016
Hustad, E.; Haddara, M.; Kalvenes, B.: ERP and Organizational Misfits: An ERP Customization Journey. In: Procedia Computer Science 100 (2016), S. 429–439.

IBM 2017
IBM, Inc: IBM: SPSS Software – Lösungen und Software für Predictive Analytics [Internet]. 2017 [Abgerufen am 12.03.2017]. URL: http://www-01.ibm.com/software/de/analytics/spss/.

Iqbal et al. 2012
Iqbal, U.; Uppström, E.; Juell-Skielse, G.: Cloud ERP Implementation Challenges: A Study based on ERP Life Cycle Model. In: Møller, C.; Chaudhry, S. S. (Hrsg.): Advances in Enterprise Information Systems II. Proceedings of the Fifth International Conference on Research and Practical Issues of

Enterprise Information Systems (CONFENIS 2011), Aalborg, Denmark, October 16 - 18, 2011. London: CRC Press, 2012, S. 389–405.

Jacobs und Aulbach 2007
Jacobs, D.; Aulbach, S.: Ruminations on Multi-Tenant Databases. In: Datenbanksysteme in Business, Technologie und Web (BTW 2007), 12. Fachtagung des GI-Fachbereiches "Datenbanken und Informationssysteme" (DBIS), 5. - 9. März 2007, Aachen, Germany. Aachen: Verlag-Haus Mainz, 2007, S. 514–521.

Jadhav und Sonar 2009
Jadhav, A. S.; Sonar, R. M.: Evaluating and Selecting Software Packages: A review. In: Information and Software Technology 51 (2009), Nr. 3, S. 555–563.

Jamshidi et al. 2013
Jamshidi, P.; Ahmad, A.; Pahl, C.: Cloud Migration Research: A Systematic Review. In: IEEE Transactions Cloud Computing 1 (2013), Nr. 2, S. 142–157.

Jiang et al. 2010
Jiang, X.; Zhang, Y.; Liu, S.: A Well-designed SaaS Application Platform Based on Model-driven Approach. In: 2010 9th International Conference on Grid and Cloud Computing (GCC 2010), Nanjing, Jiangsu, China, 1 - 5 November 2010. Nanjing: IEEE, 2010, S. 276–281.

Johansson et al. 2014
Johansson, B.; Alajbegovic, A.; Alexopoulos, V.; Desalermos, A.: Cloud ERP Adoption Opportunities and Concerns: A Comparison between SMES and Large Companies [Internet]. In: Pre-ECIS 2014 Workshop "IT Operations Management" (ITOM 2014), Tel Aviv, 8 June 2014. Tel Aviv: o.V., 2014, S. 1–13. [Abgerufen am 12.03.2017]. URL: https://lup.lub.lu.se/search/publication/4770066.

Johansson und Ruivo 2013
Johansson, B.; Ruivo, P.: Exploring Factors for Adopting ERP as SaaS. In: Procedia Technology 9 (2013), S. 94–99.

Johansson und Sudzina 2008
Johansson, B.; Sudzina, F.: ERP Systems and Open Source: An Initial Review and Some Implications for SMEs. In: Journal of Enterprise Information Management 21 (2008), Nr. 6, S. 649–658.

Johnson et al. 2007
Johnson, R. B.; Onwuegbuzie, A. J.; Turner, L. A.: Toward a Definition of Mixed Methods Research. In: Journal of Mixed Methods Research 1 (2007), Nr. 2, S. 112–133.

Jouanne-Diedrich et al. 2012
Jouanne-Diedrich, H. K. von; Frey, F.; Schmidt, R.: Geschäftsprozesse aus der Cloud. In: Bundesverband Informationswirtschaft, Telekommunikation und neue Medien (BITKOM) e. V. (Hrsg.): Serviceorientierte Architekturen in der Cloud – Leitfaden und Nachschlagewerk. 1. Auflage [Internet]. Berlin: BITKOM e. V., 2012, S. 52–63. [Abgerufen am 12.03.2017]. URL: https://www.bitkom.org/noindex/Publikationen/2012/Leitfaden/Serviceorientierte-Architekturen-in-der-Cloud/bitkom-leitfaden-soa-in-der-cloud-2012v4.pdf.

Kabbedijk 2014
Kabbedijk, J.: Variability in Multi-Tenant Enterprise Software. Utrecht, Utrecht University, Department of Information and Computing Sciences, Dissertation, 2014.

Kabbedijk et al. 2014
Kabbedijk, J.; Pors, M.; Jansen, S.; Brinkkemper, S.: Multi-tenant Architecture Comparison. In: Avgeriou, P.; Zdun, U. (Hrsg.): Software Architecture: 8th European Conference, ECSA 2014, Vienna, Austria, August 25 - 29, 2014, Proceedings, Lecture Notes in Computer Science 8627. Cham: Springer, 2014, S. 202–209.

Kabbedijk et al. 2015
Kabbedijk, J.; Bezemer, C.-P.; Jansen, S.; Zaidman, A.: Defining Multi-Tenancy: A Systematic Mapping Study on the Academic and the Industrial Perspective. In: Journal of Systems and Software 100 (2015), S. 139–148.

Kabbedijk und Jansen 2011
Kabbedijk, J.; Jansen, S.: Variability in Multi-Tenant Enterprise Software: Architectural Design Patterns from Industry. In: De Troyer O.; Bauzer Medeiros C.; Billen R.; Hallot P.; Simitsis A.; Mingroot H. van (Hrsg.): Advances in Conceptual Modeling. Recent Developments and New Directions, ER 2011 Workshops FP-UML, MoRE-BI, Onto-CoM, SeCoGIS, Variability@ER, WISM, Brussels, Belgium, October 31 - November 3, 2011. Proceedings, Lecture Notes in Computer Science 6999. Berlin Heidelberg: Springer, 2011, S. 151–160.

Kanchymalay et al. 2013
Kanchymalay, K.; Krishnan, R.; Arif, F.; Amiruddin, S.; Salam, S.; Hashim, U. R.: The Extent of ERP Customization towards User Satisfaction in Daily Operation for Manufacturing Companies. In: Journal of Computers 8 (2013), Nr. 7, S. 1788–1792.

Kang et al. 2010
Kang, S.; Myung, J.; Yeon, J.; Ha, S. W.; Cho, T.; Chung, J. M.; Lee, S. G.: A General Maturity Model and Reference Architecture for SaaS Service. In: Kitagawa H., Ishikawa Y., Li Q., Watanabe C. (Hrsg.): Database Systems for Advanced Applications, 15th International Conference, DASFAA 2010, Tsukuba, Japan, April 1 - 4, 2010, Proceedings, Part II, Lecture Notes in Computer Science 5982. Berlin Heidelberg: Springer, 2010, S. 337–346.

Kang et al. 2011
Kang, S.; Kang, S.; Hur, S.: A Design of the Conceptual Architecture for a Multitenant SaaS Application Platform. In: Byun, Y.-C.; Akingbehin, K.; Hnetynka, P.; Lee, R. (Hrsg.): First ACIS-JNU International Conference on Computers, Networks, Systems and Industrial Engineering (CNSI), Jeju Island, Korea 23 - 25 May 2011. Tokyo: IEEE, 2011, S. 462–467.

Kassem und Schult 2008
Kassem, G.; Schult, R.: ERP Self-Adaptive Customizing. In: 2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications (ICTTA), 7-11 April 2008, Damascus, Syria. Piscataway: IEEE, 2008, S. 1–5.

Keckeis et al. 2016
Keckeis, J.; Dolezel, M.; Felderer, M.: Towards a Concept for Enterprise Systems Landscape Testing. In: Piazolo, F.; Felderer, M. (Hrsg.): Multidimensional Views on Enterprise Information Systems, Lecture Notes in Information Systems and Organisation 12, Proceedings of ERP Future 2014, Part I. Cham: Springer, 2016, S. 133–146.

Keckeis und Weiss 2014
Keckeis, J.; Weiss, C.: ERP und Enterprise Systems Marktübersicht. In: SIS Consulting GmbH (Hrsg.): ERP Booklet 2015. Wien: ADV Handelsgesellschaft mbH, 2014, S. 1–95.

Kimberling 2016
Kimberling, E.: 2016 Report on ERP Systems and Enterprise Software. A Panorama Consulting Solutions Research Report [Internet]. Denver: Panorama Consulting Solutions, 2016. [Abgerufen am 12.03.2017]. URL: http://go.panorama-consulting.com/rs/panoramaconsulting/images/2016-ERP-Report.pdf.

Kitchenham 2007
Kitchenham, B.: Guidelines for Performing Systematic Literature Reviews in Software Engineering, Version 2.3, EBSE Technical Report EBSE-2007-01. Keele: Keele University and University of Durham, 2007.

Klammer 2005
Klammer, B.: Empirische Sozialforschung: Eine Einführung für Kommunikationswissenschaftler und Journalisten. Konstanz: UVK Verlagsgesellschaft, 2005.

Klink et al. 2016
Klink, P.; Mertens, C.; Kompalka, K.: ERP-Marktstudie 2016: Auswirkungen von Industrie 4.0 auf die Anforderungen an ERP-Systeme. Dortmund: Fraunhofer Institut für Materialfluss und Logistik (IML), 2016.

Konstantinidis et al. 2012
Konstantinidis, C.; Kienegger, H.; Flormann, L.; Wittges, H.; Krcmar, H.: SAP Business ByDesign: Anpassung und Integration. Bonn: Galileo Press, 2012.

Koziolek 2011
Koziolek, H.: The SPOSAD Architectural Style for Multi-tenant Software Applications. In: 2011 9th Working IEEE/IFIP Conference on Software Architecture (WICSA 2011), Boulder, Colorado, USA, 20 - 24 June 2011. Washington: IEEE, 2011, S. 320–327.

KPMG und BITKOM 2016
KPMG und BITKOM: Cloud-Monitor 2016. Cloud-Computing in Deutschland – Status quo und Perspektiven [Internet]. Berlin: KPMG AG in Zusammenarbeit mit Bitkom Research GmbH, 2016. [Abgerufen am 12.03.2017]. URL: http://www.bitkom-research.de/CM2016.

Krajewski et al. 2016
Krajewski, L. J.; Ritzman, L. P.; Malhotra, M. K.: Operations Management: Processes and Supply Chains. 11th edition. Boston: Pearson, 2016.

Krcmar 1995
Krcmar, H.: Current Research and Practice in Information Systems in Germany (Panel). In: Doukidis, G.; Galliers, R.; Jelassi, T.; Krcmar, H.; Land, F. (Hrsg.): Proceedings of the Third European Conference on Information Systems, ECIS '95, Athens, Greece, June 1 - 3 1995, vol. I. Athens: Print Xpress, 1995, S. 1295–1297.

Krcmar 2015
Krcmar, H.: Informationsmanagement. 6., überarb. Auflage. Berlin Heidelberg: Springer, 2015.

Krebs et al. 2012
Krebs, R.; Momm, C.; Kounev, S.: Architectural Concerns in Multi-Tenant SaaS Applications. In: Proceedings of the 2nd International Conference on Cloud Computing and Service Science (CLOSER'12), Porto, Portugal, April 18 - 21, 2012. Porto: SciTePress, 2012, S. 426–431.

Kulkarni et al. 2012
Kulkarni, G.; Gambhir, J.; Palwe, R.: Cloud Computing-Software as Service. In: International Journal of Cloud Computing and Services Science 1 (2012), Nr. 1, S. 11–16.

Kumar et al. 2010
Kumar, M. N. V.; Suresh, A. V.; Subramanaya, K. N.: Application of an Analytical Hierarchy Process to Prioritize the Factors Affecting ERP Implementation. In: International Journal of Computer Applications 2 (2010), Nr. 2, S. 1–6.

Kurbel 2008
Kurbel, K.: The Making of Information Systems – Software Engineering and Management in a Globalized World. Berlin, Heidelberg: Springer, 2008.

Kurbel 2013
Kurbel, K.: Enterprise Resource Planning and Supply Chain Management – Functions, Business Processes and Software for Manufacturing Companies. Berlin, Heidelberg: Springer, 2013.

Kurbel 2016
Kurbel, K.: Enterprise Resource Planning und Supply Chain Management in der Industrie: Von MRP bis Industrie 4.0. 8., vollst. überarb. und erw. Auflage. Berlin: Walter de Gruyter GmbH & Co KG, 2016.

Kurbel et al. 2011
Kurbel, K.; Krybus, I.; Stankov, I.: Reviewing Criteria of Information Systems, Computer Science and Engineering Journals. In: Journal of Computer Information Systems 51 (2011), Nr. 4, S. 74–80.

Kurbel und Nowak 2013a
Kurbel, K.; Nowak, D.: Customization of On-demand ERP Software Using SAP Business ByDesign as an Example. In: Piazolo, F.; Felderer, M. (Hrsg.): Innovation and Future of Enterprise Information Systems: ERP Future 2012 Conference, Salzburg, Austria, November 2012, Revised Papers, Lecture Notes in Information Systems and Organisation 4. Berlin Heilderberg: Springer, 2013, S. 289–297.

Kurbel und Nowak 2013b
Kurbel, K.; Nowak, D.: Exploring Mobility and Social Media Integration in an On-Demand ERP System using SAP Business ByDesign as an Example. In: Issues in Information Systems 14 (2013), Nr. 1, S. 421–430.

Kwok und Mohindra 2008
Kwok, T.; Mohindra, A.: Resource Calculations with Constraints, and Placement of Tenants and Instances for Multi-tenant SaaS Applications. In: Bouguettaya, A.; Krueger, I.; Margaria, T. (Hrsg.): Service-Oriented Computing – ICSOC 2008, 6th International Conference, Sydney, Australia, December 2008, Proceedings, Lecture Notes in Computer Science 5364. Berlin Heidelberg: Springer, 2008, S. 633–648.

Lechesa et al. 2012
Lechesa, M.; Seymour, L.; Schuler, J.: ERP Software as Service (SaaS): Factors Affecting Adoption in South Africa. In: Møller, C.; Chaudhry, S. (Hrsg.): Re-conceptualizing Enterprise Information Systems, 5th IFIP WG 8.9 Working Conference, CONFENIS 2011, Aalborg, Denmark, October 2011, Revised Selected Papers, Lecture Notes in Business Information Processing 105. Berlin Heidelberg: Springer, 2012, S. 152–167.

Lenart 2011
Lenart, A.: ERP in the Cloud – Benefits and Challenges. In: Wrycza, S. (Hrsg.): Research in Systems Analysis and Design: Models and Methods, 4th SIGSAND/PLAIS EuroSymposium 2011, Gdańsk, Poland, September 29, 2011, Revised Selected Papers, Lecture Notes in Business Information Processing 93. Berlin Heidelberg: Springer, 2011, S. 39–50.

Leyh 2011
Leyh, C.: Tailoring of ERP Systems. In: Léger, P.M. ;Pellerin, R.; Babin, G. (Hrsg.): Readings on Enterprise Resource Planning (Preliminary Version). Montreal: ERsim Lab, HEC Montreal, 2011, S. 74–86.

Li et al. 2011
Li, J.; Xie, T.; Du, S.: Requirements Analysis on Flexibility of ERP System of Medium and Small Publishers. In: Procedia Engineering 15 (2011), S. 5493–5497.

Light 2001
Light, B.: The Maintenance Implications of the Customisation of ERP Software. In: Journal of Software Maintenance and Evolution: Research and Practice 13 (2001), Nr. 6, S. 415–429.

Light 2005
Light, B.: Going beyond ‘Misfit’ as a Reason for ERP Package Customisation. In: Computers in Industry 56 (2005), Nr. 6, S. 606–619.

LimeSurvey 2015
LimeSurvey: LimeSurvey [Internet]. 2015 [Abgerufen am 25.02.2016].
URL: https://www.limesurvey.org.

LimeSurvey 2017
LimeSurvey: LimeSurvey Bedienungsanleitung [Internet]. 2017 [Abgerufen am 05.03.2017].
URL: https://manual.limesurvey.org/LimeSurvey_Manual.

Link 2013
Link, B.: Considering the Company's Characteristics in Choosing between SaaS vs. On-Premise-ERPs. In: Alt, R.; Franczyk, B. (Hrsg.): 11. Internationale Tagung Wirtschaftsinformatik (WI-2013), February 27 - March 1, 2013. Leipzig, Germany. Leipzig: Universität Leipzig, 2013, S. 261–277.

Link und Back 2013
Link, B.; Back, A.: Systemic Differences between SaaS- and On-Premise-ERP: An Overview of a Qualitative Option Calculation Scheme. In: AIS Transactions on Enterprise Systems 4 (2013), Nr. 1, S. 11–23.

Lucas und Babaian 2009
Lucas, W.; Babaian, T.: Reasoning for Intelligent System-User Interactions with Enterprise Resource Planning Systems. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2009), Workshop on Intelligence and Interaction, Pasadena, CA, USA, 2009. Pasadena: o. V., 2009, S. 1–6.

Luo und Strong 2004
Luo, W.; Strong, D. M.: A Framework for Evaluating ERP Implementation Choices. In: IEEE Transactions on Engineering Management 51 (2004), Nr. 3, S. 322–333.

Magnusson und Enquist 2012
Magnusson, J.; Enquist, H.; Juell-Skielse, G.; Uppström, E.: Incumbents and Challengers: Conflicting Institutional Logics in SaaS ERP Business Models. In: Journal of Service Science and Management 5 (2012), Nr. 1, S. 69–76.

Magnusson und Nilsson 2013
Magnusson, J.; Nilsson, A.: Introducing App Stores into a Packaged Software Ecosystem: A Negotiated Order Perspective. In: International Journal of Business Information Systems 14 (2013), Nr. 2, S. 223–237.

Makki et al. 2016
Makki, M.; Landuyt, D. van; Walraven, S.; Joosen, W.: Scalable and Manageable Customization of Workflows in Multi-Tenant SaaS Offerings. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing, SAC 2016, April 4 - 8, 2016, Pisa, Italy. New York: ACM, 2016, S. 432–439.

Markus und Tanis 2000
Markus, M. L.; Tanis, C.: The Enterprise Systems Experience – From Adoption to Success. In: Zmud, R. W.; Price, F. M. (Hrsg.): Framing the Domains of IT Management: Projecting the Future Through the Past. Cincinnati: Pinnaflex Educational Resources, Inc., 2000, S. 173–207.

Marwan 2011
Marwan, P.: ZDNet: Scopevisio erweitert Cloud-Angebot um CRM, Faktura, Anlagenbuchhaltung und BI [Internet]. 2011 [Abgerufen am 12.03.2017]. URL: http://www.zdnet.de/41545419/scopevisio-erweitert-cloud-angebot-um-crm-faktura-anlagenbuchhaltung-und-bi/.

Mayring und Brunner 2009
Mayring, P.; Brunner, E.: Qualitative Inhaltsanalyse. In: Buber, R.; Holzmüller, H. (Hrsg.): Qualitative Marktforschung. Konzepte – Methoden – Analysen. Wiesbaden: Gabler, 2009, S. 669–680.

Meinel et al. 2011
Meinel, C.; Willems, C.; Roschke, S.; Schnjakin, M.: Virtualisierung und Cloud Computing: Konzepte, Technologiestudie, Marktübersicht. Technische Berichte Nr. 44 des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam. Potsdam: Universitätsverlag Potsdam, 2011.

Mell und Grance 2011
Mell, P.; Grance, T.: The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology, Special Publication 800-145. Washington: National Institute of Standards and Technology (NIST), 2011.

Mertens et al. 2010
Mertens, P.; Bodendorf, F.; König, W.; Picot, A.; Schumann, M.; Hess, T.: Grundzüge der Wirtschaftsinformatik. 10. Auflage. Berlin, Heidelberg, New York: Springer, 2010.

Mietzner et al. 2011
Mietzner, R.; Leymann, F.; Unger, T.: Horizontal and Vertical Combination of Multi-Tenancy Patterns in Service-Oriented Applications. In: Enterprise Information Systems 5 (2011), Nr. 1, S. 59–77.

Mijač et al. 2013
Mijač, M.; Picek, R.; Stapić, Z.: Cloud ERP System Customization Challenges. In: Central European Conference on Information and Intelligent Systems, September 18 - 20, 2013, Varaždin, Croatia. Varaždin: Faculty of Organization and Informatics, University of Zagreb, 2013, S. 132–140.

Momm und Krebs 2011
Momm, C.; Krebs, R.: A Qualitative Discussion of Different Approaches for Implementing Multi-Tenant SaaS Offerings. In: Software Engineering 11 (2011), S. 139–150.

Müller et al. 2009
Müller, J.; Krüger, J.; Enderlein, S.; Helmich, M.; Zeier, A.: Customizing Enterprise Software as a Service Applications: Back-End Extension in a Multi-tenancy Environment. In: Filipe, J.; Cordeiro, J. (Hrsg.): Enterprise Information Systems, 11th International Conference, ICEIS 2009, Milan, Italy, May 6 - 10, 2009, Proceedings, Lecture Notes in Business Information Processing 24. Berlin Heidelberg: Springer, 2009, S. 66–77.

Münzl et al. 2009
Münzl, G.; Przywara B.; Reti, M.; Schäfer, J.; Sondermann, K.; Weber, M.; Wilker, A.: Cloud Computing-Evolution in der Technik, Revolution im Business, BITKOM-Leitfaden [Internet]. Berlin: Bundesverband Informationswirtschaft, Telekommunikation und neue Medien (BITKOM) e. V., 2009. [Abgerufen am 12.03.2017]. URL: https://www.bitkom.org/noindex/Publikationen/2009/Leitfaden/Leitfaden-Cloud-Computing/090921-BITKOM-Leitfaden-CloudComputing-Web.pdf.

Nah et al. 2001
Nah, F. F.-H.; Faja, S.; Cata, T.: Characteristics of ERP Software Maintenance: A Multiple Case Study. In: Journal of Software Maintenance and Evolution: Research and Practice 13 (2001), Nr. 6, S. 399–414.

Nah und Delgado 2006
Nah, F. F.-H.; Delgado, S.: Critical Success Factors for Enterprise Resource Planning Implementation and Upgrade. In: Journal of Computer Information Systems 46 (2006), Nr. 5, S. 99–113.

Ng et al. 2002
Ng, C. S. P.; Gable, G. G.; Chan, T.: An ERP-client Benefit-oriented Maintenance Taxonomy. In: Journal of Systems and Software 64 (2002), Nr. 2, S. 87–109.

Ng et al. 2003
Ng, C. S. P.; Gable, G. G.; Chan, T.: An ERP Maintenance Model. In: Proceedings of the 36th Annual Hawaii International Conference on System Sciences, January 6 - 9, 2003, Big Island, Hawaii. Washington: IEEE, 2003, S. 1–10.

Nitu 2009
Nitu: Configurability in SaaS (Software as a Service) Applications. In Proceedings of the 2nd Annual India Software Engineering Conference, ISEC 2009, Pune, India, February 23 - 26, 2009. Pune: ACM, 2009, S. 19–26.

Norm DIN 199-1 2002
Norm DIN 199-1: Technische Produktdokumentation – CAD-Modelle, Zeichnungen und Stücklisten – Teil 1: Begriffe. Berlin: Beuth, 2002.

Nowak und Kurbel 2014
Nowak, D.; Kurbel, K.: Integration von Web 2.0-Technologien in ERP-as-a-Service am Beispiel von SAP Business ByDesign. In: Kundisch, D.; Suhl, L.; Beckmann, L. (Hrsg.): MKWI 2014 – Multikonferenz Wirtschaftsinformatik: 26. - 28. Februar 2014 in Paderborn; Tagungsband der Multi-Konferenz Wirtschaftsinformatik (MKWI 2014), Paderborn, 2014. Paderborn: Universität Paderborn, 2014, S. 429–442.

Nowak und Kurbel 2017
Nowak, D.; Kurbel, K.: Understanding the Flexibility of Cloud ERP Software. In: Piazolo, F.; Geist, V.; Brehm, L.; Schmidt, R. (Hrsg.): Innovations in Enterprise Information Systems Management and Engineering. 5th International Conference, ERP Future 2016 – Research, Hagenberg, Austria, November 14, 2016, Revised Papers. Lecture Notes in Business Information Processing 285. Cham: Springer, 2017, S. 135–146.

Odoo S. A. 2016a
Odoo S. A.: Odoo [Internet]. 2016a [Abgerufen am 11.03.2017]. URL: https://www.odoo.com/.

Odoo S. A. 2016b
Odoo S. A.: Odoo Documentation. Actions [Internet]. 2016b [Abgerufen am 11.03.2017]. URL: https://www.odoo.com/documentation/9.0/reference/actions.html.

Odoo S. A. 2016c
Odoo S. A.: Odoo Documentation. Forms [Internet]. 2016c [Abgerufen am 11.03.2017]. URL: https://www.odoo.com/documentation/9.0/reference/views.html#forms.

Odoo S. A. 2016d
Odoo S. A.: Odoo Documentation. QWeb Reports [Internet]. 2016d [Abgerufen am 11.03.2017]. URL: https://www.odoo.com/documentation/9.0/reference/reports.html.

Odoo S. A. 2016e
Odoo S. A.: Odoo Documentation. Web Service API [Internet]. 2016e [Abgerufen am 11.03.2017]. URL: https://www.odoo.com/documentation/9.0/api_integration.html.

Odoo S. A. 2016f
Odoo S. A.: Odoo Online Pricing [Internet]. 2016f [Abgerufen am 11.03.2017]. URL: https://www.odoo.com/pricing-online.

Odoo S. A. 2016g
Odoo S. A.: Odoo: Pricing [Internet]. 2016g [Abgerufen am 11.03.2017]. URL: https://www.odoo.com/editions.

Olson und Zhao 2007
Olson, D. L.; Zhao, F.: CIOs' Perspectives of Critical Success Factors in ERP Upgrade Projects. In: Enterprise Information Systems 1 (2007), Nr. 1, S. 129–138.

Ortner und Krenn 2016
Ortner, W.; Krenn, G.: Are New Configuration Methors 'the Key' to Shorter ERP Implementations?. In: Piazolo, F.; Felderer, M. (Hrsg.): Multidimensional Views on Enterprise Information Systems, Lecture Notes in Information Systems and Organisation 12, Proceedings of ERP Future 2014, Part I. Cham: Springer, 2016, S. 23–40.

Osipov et al. 2009
Osipov, C.; Goldszmidt, G.; Taylor, M.; Poddar, I.: Develop and Deploy Multi-Tenant Web-delivered Solutions using IBM middleware, Part 2: Approaches for enabling multi-tenancy [Internet]. 2009 [Abgerufen am 11.03.2017]. URL: http://www.ibm.com/developerworks/webservices/library/ws-multitenantpart2/index.html.

Pajk und Kovačič 2013
Pajk, D.; Kovačič, A.: Fit Gap Analysis – The Role of Business Process Reference Models. In: Economic and Business Review 15 (2013), Nr. 4, S. 319–338.

Parthasarathy 2013
Parthasarathy, S.: Potential Concerns and Common Benefits of Cloud-Based Enterprise Resource Planning (ERP). In: Mahmood, Z. (Hrsg.): Cloud Computing, Methods and Practical Approaches, Part III, Computer Communications and Networks. London: Springer, 2013, S. 177–195.

Parthasarathy und Daneva 2016
Parthasarathy, S.; Daneva, M.: An Approach to Estimation of Degree of Customization for ERP Projects Using Prioritized Requirements. In: The Journal of Systems and Software 117 (2016), S. 471–487.

Parthasarathy und Sharma 2016
Parthasarathy, S.; Sharma, S.: Efficiency Analysis of ERP Packages – A Customization Perspective. In: Computers in Industry 82 (2016), S. 19–27.

Peng und Gala 2014
Peng, G. C.; Gala, C. J.: Cloud ERP: A New Dilemma to Modern Organisations?. In: Journal of Computer Information Systems 54 (2014), Nr. 4, S. 22–30.

Pfadenhauer 2009
Pfadenhauer, M.: Das Experteninterview – Ein Gespräch auf gleicher Augenhöhe. In: Buber, R.; Holzmüller, H. (Hrsg.): Qualitative Marktforschung. Konzepte - Methoden - Analysen. Wiesbaden: Gabler, 2009, S. 449–461.

Pries-Heje 2006
Pries-Heje, L.: ERP Misfits: What is It and How do They Come About?. In: Spencer, S; Jenkins, A. (Hrsg.): Proceedings of the 17th Australasian Conference on Information Systems, Adelaide, 6-8 December 2006: Thought Leadership in IS. Paper 48. Sydney: Australasian Conference on Information Systems, 2006, S. 1–10.

Rabay'a et al. 2013
Rabay'a, A.; Dweib, M.; Abuzir, Y.: Implementing Cloud Computing in ERP. In: Journal of Emerging Trends in Computing and Information Sciences 4 (2013), Nr. 10, S. 770–777.

Ragusa und Puliafito 2011
Ragusa, C.; Puliafito, A.: Running Business Applications in the Cloud: A Use Case Perspective. In: Guarracino, M. R.; Vivien, F.; Träff, J. L.; Cannatoro, M.; Danelutto, M.; Hast, A.; Perla, F.; Knüpfer, A.; Di Martino, B.; Alexander, M. (Hrsg.): Euro-Par 2010 Parallel Processing Workshops, HeteroPar, HPCC, HiBB, CoreGrid, UCHPC, HPCF, PROPER, CCPI, VHPC, Ischia, Italy, August 31 - September 3, 2010, Revised Selected Papers, Lecture Notes in Computer Science 6586. Berlin Heidelberg: Springer, 2011, S. 595–602.

Repschläger 2013
Repschläger, J.: Entscheidungsfindung im Cloud Computing: Konzeption und Analyse eines Modells zur Anbieterauswahl. Berlin, Technische Universität Berlin, Dissertation, 2013.

Ried et al. 2008
Ried, S.; Rymer, J. R.; Iqbal, R.: Forrester's SaaS Maturity Model, Transforming Vendor Strategy While Managing Customer Expectations [Internet]. Forrester Research, Inc., 2008. [Abgerufen am 11.06.2016]. URL: https://www.forrester.com/report/Forresters+SaaS+Maturity+Model/-/E-RES46817.

Riedl und Roithmayr 2006
Riedl, R.; Roithmayr, F.: Zur Fallstudienforschung in der Disziplin Information Systems: Eine quantitative Inhaltsanalyse. In: Jung, R.; Myrach, T. (Hrsg.): Quo vadis Wirtschaftsinformatik? Wiesbaden: Gabler, 2006, S. 125–145.

Rose 2015
Rose, N.: openSAP: Application Development for SAP Business ByDesign. [Internet]. 2015 [Abgerufen am 11.03.2017]. URL: https://open.sap.com/courses/byd1#172.

Ross und Vitale 2000
Ross, J. W.; Vitale, M. R.: The ERP Revolution: Surviving vs. Thriving. In: Information Systems Frontiers 2 (2000), Nr. 2, S. 233–241.

Rothenberger und Srite 2009
Rothenberger, M. A.; Srite, M.: An Investigation of Customization in ERP System Implementations. In: IEEE Transactions on Engineering Management 56 (2009), Nr. 4, S. 663–676.

Sääksjärvi et al. 2005
Sääksjärvi, M.; Lassila, A.; Nordström, H.: Evaluating the Software as a Service Business Model: From CPU Time-Sharing to Online Innovation Sharing. In: Isaias, P.; McPherson, M.; Kommers, P. (Hrsg.): Proceedings of the IADIS International Conference on e-Society 2005, Qawra, Malta 27 - 30 June 2005. Qawra: IADIS Press, International Association for Development of the Information Society, 2005, S. 177–186.

Saeed et al. 2012
Saeed, I.; Juell-Skielse, G.; Uppström, E.: Cloud Enterprise Resource Planning Adoption: Motives & Barriers. In: Møller, C.; Chaudhry, S. S. (Hrsg.): Advances in Enterprise Information Systems II. Proceedings of the Fifth International Conference on Research and Practical Issues of Enterprise Information Systems (CONFENIS 2011), Aalborg, Denmark, October 16-18, 2011. London: CRC Press, 2012, S. 429–434.

SAP AG 2006
SAP AG: SAP Help - User-Exit [Internet]. 2006 [Abgerufen am 11.03.2017]. URL: http://help.sap.com/saphelp_46c/helpdata/de/35/26b334afab52b9e10000009b38f974/content.htm.

SAP SE 2016a
SAP SE: SAP Business ByDesign – Geschäftsflexibilität [Internet]. 2016 [Abgerufen am 11.03.2017]. URL: http://help-legacy.sap.com/saphelp_byd1608/de/KTP/Software-Components/0120061532010 0003379/SAP_BBD/Print_Files/PDF_FILES/BusinessFlexibility.pdf.

SAP SE 2016b
SAP SE: SAP Cloud Applications Studio 1602 (February 2016) [Internet]. 2016 [Abgerufen am 11.03.2017]. URL: https://help.sap.com/doc/saphelpiis_studio_1602_studio_od_1602_pdf/2016.02/en-US/studio_od_1602.pdf.

SAP SE 2016c
SAP SE: Web Service APIs in SAP Business ByDesign 1611 – November 2016 [Internet]. 2016 [Abgerufen am 11.03.2017]. URL: http://help.sap.com/saphelp_byd1611/en/PUBLISHING/Integration Services.html.

SAP SE 2016d
SAP SE: SAP Business ByDesign: Cloud-based ERP for Mid-market Companies and Subsidiaries [Internet]. 2016 [Abgerufen am 11.03.2017]. URL: http://go.sap.com/product/enterprise-management/business-bydesign.html.

Sarker et al. 2012
Sarker, S.; Sarker, S.; Sahaym, A.; Bjørn-Andersen, N.: Exploring Value Cocreation in Relationships Between an ERP Vendor and its Partners: A Revelatory Case Study. In: MIS Quarterly 36 (2012), Nr. 1, S. 317–338.

Scavo et al. 2012
Scavo, F.; Newton, B.; Longwell, M.: Choosing Between Cloud and Hosted ERP, and Why It Matters. In: Computer Economics Report 34 (2012), Nr. 8, S. 1–12.

Scherrer-Rathje und Boyle 2008
Scherrer-Rathje, M.; Boyle, T. A.: End-User Perspective on ERP Flexibility. In: Managing the Responsible Enterprise 29 (2008), Nr. 7, S. 83–98.

Schiller et al. 2011
Schiller, O.; Schiller, B.; Brodt, A.; Mitschang, B.: Native Support of Multi-tenancy in RDBMS for Software as a Service. In: Ailamaki, A.; Amer-Yahia, S.; Patel, J. M.; Risch, T.; Senellart, P.; Stoyanovich, J. (Hrsg.): EDBT 2011, 14th International Conference on Extending Database Technology, Uppsala, Sweden, March 21 - 24, 2011, Proceedings. New York: ACM, 2011, S. 117–128.

Schlichter und Kraemmergaard 2010
Schlichter, B. R.; Kraemmergaard, P.: A Comprehensive Literature Review of the ERP Research Field over a Decade. In: Journal of Enterprise Information Management 23 (2010), Nr. 4, S. 486–520.

Schmid und Rummler 2012
Schmid, K.; Rummler, A.: Cloud-based Software Product Lines. In: Proceedings of the 2nd International Workshop on Services, Clouds, and Alternative Design Strategies for Variant-Rich Software Systems (SCArVeS 2012) at the 16th International Software Product Line Conference (SPLC'12), Salvador, Brazil, September 2 - 7, 2012, Vol. 2. New York: ACM, 2012, S. 164–170.

Schneider 2012
Schneider, T.: SAP Business ByDesign Studio: Application Development. Boston: Galileo Press, 2012.

Scholze-Stubenrecht 2011
Scholze-Stubenrecht, W.: Duden - Deutsches Universalwörterbuch. 7., überarb. und erw. Auflage Mannheim: Dudenverlag, 2011.

Schreier 2011
Schreier, M.: Qualitative Stichprobenkonzepte. In: Naderer, G.; Balzer, E. (Hrsg.): Qualitative Marktforschung in Theorie und Praxis, Grundlagen, Methoden und Anwendungen. Wiesbaden: Gabler, 2011, S. 243–257.

Schubert und Adisa 2011
Schubert, P.; Adisa, F.: Cloud Computing for Standard ERP Systems: Reference Framework and Research Agenda. Arbeitsberichte aus dem Fachbereich Informatik Nr. 16/2011. Koblenz: Institut für Wirtschafts- und Verwaltungsinformatik, Universität Koblenz-Landau, 2011, S. 1–29.

Scopevisio AG 2016a
Scopevisio AG: Openscope: Openscope API [Internet]. 2016 [Abgerufen am 10.04.2016]. URL: https://openscope.de/api.html.

Scopevisio AG 2016b
Scopevisio AG: Scopevisio: Branchenlösungen [Internet]. 2016 [Abgerufen am 30.12.2016]. URL: https://www.scopevisio.com/branchen.

Scopevisio AG 2016c
Scopevisio AG: Scopevisio: Konfigurator [Internet]. 2016 [Abgerufen am 30.12.2016]. URL: https://www.scopevisio.com/anwendungen/konfigurator.

Scopevisio AG 2016d
Scopevisio AG: Scopevisio: Lizenzmodell der Scopevisio [Internet]. 2016 [Abgerufen am 30.12.2016]. URL: https://www.scopevisio.com/help/de/Allgemein/Lizenzmodell.

Scopevisio AG 2016e
Scopevisio AG: Scopevisio: Scopevisio Preis- und Funktionsübersicht [Internet]. 2016 [Abgerufen am 10.03.2016]. URL: https://www.scopevisio.com/preise.

Seethamraju 2015
Seethamraju, R.: Adoption of Software as a Service (SaaS) Enterprise Resource Planning (ERP) Systems in Small and Medium Sized Enterprises (SMEs). In: Information Systems Frontiers 17 (2015), Nr. 3, S. 475–492.

Seethamraju und Sundar 2013
Seethamraju, R.; Sundar, D. K.: Influence of ERP Systems on Business Process Agility. In: IIMB Management Review 25 (2013), Nr. 3, S. 137–149.

Sekatzek und Krcmar 2009
Sekatzek, E. P.; Krcmar, H.: Measurement of the Standard Proximity of Adapted Standard Business Software. In: Business & Information Systems Engineering 1 (2009), Nr. 3, S. 234–244.

Shankararaman und Lum 2013
Shankararaman, V.; Lum, E. K.: Integration of Social Media Technologies with ERP: A Prototype Implementation. In: Proceedings of the 19th Americas Conference on Information Systems, Chicago, Illinois, August 15 - 17, 2013. Paper 11. Chicago: o. V., 2013, S. 1–11.

Shao 2011
Shao, Q.: Towards Effective and Intelligent Multi-tenancy SaaS. Arizona: Arizona State University, Dissertation, 2011.

Software Advice 2015
Software Advice, Inc: Software Advice: Compare Cloud-based ERP Software [Internet]. 2015 [Abgerufen am 12.09.2015]. URL: http://www.softwareadvice.com/erp/web-based-erp-software-comparison/.

Soh et al. 2000
Soh, C.; Kien, S. S.; Tay-Yap, J.: Enterprise Resource Planning: Cultural Fits and Misfits: Is ERP a Universal Solution?. In: Communications of the ACM 43 (2000), Nr. 4, S. 47–51.

Soh et al. 2003
Soh, C.; Kien, S. S.; Fong Boh, W.; Tang, M.: Misalignments in ERP Implementation: A Dialectic Perspective. In: International Journal of Human-Computer Interaction 16 (2003), Nr. 1, S. 81–100.

Stahl et al. 2007
Stahl, T.; Völter, M.; Efftinge, S.; Haase, A.: Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management. 2., aktualisierte und erw. Auflage. Heidelberg: dpunkt, 2007.

Stahlknecht und Hasenkamp 2006
Stahlknecht, P.; Hasenkamp, U.: Arbeitsbuch Wirtschaftsinformatik. 4. Auflage. Berlin Heidelberg: Springer, 2006.

Subramoniam et al. 2009
Subramoniam, S.; Tounsi, M.; Krishnankutty, K. V.: The Role of BPR in the Implementation of ERP Systems. In: Business Process Management Journal 15 (2009), Nr. 5, S. 653–668.

Sun et al. 2008
Sun, W.; Zhang, X.; Guo, C. J.; Sun, P.; Su, H.: Software as a Service: Configuration and Customization Perspectives. In: 2008 IEEE Congress on Services, SERVICES 2008 Part 2, 23 - 26 September, 2008, Beijing, China. Tokyo: IEEE, 2008, S. 18–25.

Themistocleous et al. 2001
Themistocleous, M.; Irani, Z.; O'Keefe, R. M.; Paul, R.: ERP Problems and Application Integration Issues: An Empirical Survey. In: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS), Maui, Hawaii, January 3 - 6, 2001. Piscataway: IEEE, 2001, S. 1–10.

Tufféry 2011
Tufféry, S.: Data Mining and Statistics for Decision Making. Chichester, UK: John Wiley & Sons, 2011.

Turowski 2002
Turowski, K.: Fachkomponenten: Vereinheitlichte Spezifikation von Fachkomponenten – Memorandum des Arbeitskreises 5.10.3 Komponentenorientierte betriebliche Anwendungssysteme. Augsburg: Universität Augsburg, 2002.

Umble 2003
Umble, E. J.; Haft, R. R.; Umble, M. M.: Enterprise Resource Planning: Implementation Procedures and Critical Success Factors. In: European Journal of Operational Research 146 (2003), Nr. 2, S. 241–257.

Uppström et al. 2015
Uppström, E.; Lonn, C.-M.; Hoffsten, M.; Thorstrom, J.: New Implications for Customization of ERP Systems. In: Proceedings on the 2015 48th Hawaii International Conference on System Sciences (HICSS 2015), 5 - 8 January, 2015, Kauai, Hawaii. Washington: IEEE, 2015, S. 4220–4229.

Usman et al. 2014
Usman, A.; Coombs, C.; Neil, D.: Re-Conzeptualization of Tailoring Types and their Impacts: A Perspective for Contemporary ERP Systems. In: Mola, L.; Carugati, A.; Kokkinaki, A.; Pouloudi, N.

(Hrsg.): Proceedings of the 8th Mediterranean Conference on Information Systems, Verona, Italy, September 3 - 5, 2014. Paper 30. Verona: AISeL, 2014, S. 1–29.

Vaquero et al. 2009
Vaquero, L. M.; Rodero-Merino, L.; Caceres, J.; Lindner, M.: A Break in the Clouds: Towards a Cloud Definition. In: ACM SIGCOMM Computer Communication Review 39 (2009), Nr. 1, S. 50–55.

W3C 2017a
W3C: HTML <iframe> Tag [Internet]. 2017 [Abgerufen am 11.03.2017]. URL: http://www.w3schools.com/tags/tag_iframe.asp.

W3C 2017b
W3C: Web Services Glossary [Internet]. 2017 [Abgerufen am 11.03.2017]. URL: https://www.w3.org/TR/ws-gloss/#webservice.

Walraven et al. 2011
Walraven, S.; Truyen, E.; Joosen, W.: A Middleware Layer for Flexible and Cost-Efficient Multi-tenant Applications. In: Kon, F.; Kermarrec A.-M. (Hrsg.): Middleware 2011, ACM/IFIP/USENIX 12th International Middleware Conference, Lisbon, Portugal, December 12-16, 2011, Proceedings, Lecture Notes in Computer Science 7049. Berlin Heidelberg: Springer, 2011, S. 370–389.

Walraven et al. 2016
Walraven, S.; Lagaisse, B.; Joosen, W.: Efficient Customization of Multi-Tenant SaaS. Imec-DestriNet Research Group, KU Leuven. Leuven: University of Leuven, 2016, S. 1–6.

Wang et al. 2011
Wang, Y.; Greaseley, A.; Thanassoulis, E.: Combining ERP Systems with Enterprise 2.0. In: Cruz-Cunha, M. M.; Varajão, J.; Powell, P.; Martinho, R. (Hrsg.): ENTERprise Information Systems, International Conference, CENTERIS 2011, Vilamoura, Portugal, October 5 - 7, 2011, Proceedings, Part I, Communications in Computer and Information Science 219. Berlin Heidelberg: Springer, 2011, S. 198–207.

Webster und Watson 2002
Webster, J.; Watson, R. T.: Analyzing the Past to Prepare the Future: Writing a Literature Review. In: MIS Quarterly 26 (2002), Nr. 2, S. 13–23.

Weidenhaupt 2001
Weidenhaupt, K. L.: Anpassbarkeit von Software-Werkzeugen in prozessintegrierten Entwicklungsumgebungen. Aachen: Rheinisch-Westfälische Technische Hochschule Aachen, Dissertation, 2001.

Wilde und Hess 2007
Wilde, T.; Hess, T.: Forschungsmethoden der Wirtschaftsinformatik – Eine empirische Untersuchung. In: Wirtschaftsinformatik 49 (2007), Nr. 4, S. 280–287.

Willis und Willis-Brown 2002
Willis, T. H.; Willis-Brown, A. H.: Extending the Value of ERP. In: Industrial Management & Data Systems 102 (2002), Nr. 1, S. 35–38.

Wölfel 2015
Wölfel, K.: Automated ERP Category Configuration Support for Small Businesses. In: International Journal of Enterprise Information Systems 11 (2015), Nr. 2, S. 1–23.

Wong et al. 2005
Wong, A.; Chau, P. Y. K.; Scarbrough, H.; Davison, R.: Critical Failure Factors in ERP Implementation. In: Proceedings of the 9th Pacific Asia Conference on Information Systems (PACIS 2005), Bangkok, Thailand, Jul, 2005. Paper 40. Bangkok: Natl Sun Yat-Sen University, 2005, S. 492–505.

XING 2017
XING AG: XING [Internet]. 2017 [Abgerufen am 11.03.2017]. URL: https://www.xing.com/.

Yang und Tate 2012
Yang, H.; Tate, M.: A Descriptive Literature Review and Classification of Cloud Computing Research. In: Communications of the Association for Information Systems 31 (2012), Nr. 2, S. 35–60.

Yen et al. 2011
Yen, T.S.; Idrus, R.; Yusof, U.: A Framework for Classifying Misfits between Enterprise Resource Planning (ERP) Systems and Business Strategies. In: Asian Academy of Management Journal 16 (2011), Nr. 2, S. 53–75.

Zach und Munkvold 2012
Zach, O.; Munkvold, B.E.: Identifying Reasons for ERP System Customization in SMEs: A Multiple Case Study. In: Journal of Enterprise and Information Management 25 (2012), Nr. 5, S. 462–478.

Zhao und Kirche 2013
Zhao, F.; Kirche, E.: Continuing On-Premise or Adopt On-Demand? An Empirical Study of ERP Adoption in SMEs. In: Kurosu, M. (Hrsg.): Human-Computer Interaction, Users and Contexts of Use, 15th International Conference, HCI International 2013, Las Vegas, NV, USA, July 21 - 26, 2013, Proceedings, Part III, Lecture Notes in Computer Science 8006. Berlin Heidelberg: Springer, 2013, S. 492–500.

# Anhang

# A Details der Literaturanalyse

## A.1 Ergebnisse der quantitativen Literaturstudie – häufige Adaptionsanforderungen

| Adaptionsanforderung | Beijsterveld und Groenendaal (2015) | Brehm et al. (2001) | Callejas und Terzi (2013) | Hustad et al. (2016) | Kanchymalay et al. (2013) | Keckeis et al. (2016) | Leyh (2011) | Li et al. (2011) | Light (2005) | Parthasarathy und Sharma (2016) | Pries-Heje (2006) | Sherrer-Rathje und Boyle (2008) | Seethamraju und Sundar (2013) | Sekatzek und Krcmar (2009) | Soh et al. (2003) | Themistocleous et al. (2001) | Yen et al. (2011) | Zach und Munkvold (2012) | **Anzahl** | **Anteil** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Definition neuer Systemfunktionalität | X | X | | X | X | X | X | X | X | X | | X | | X | | X | X | X | 14 | 77,8 % |
| Definition neuer Berichte u. Dashboards | X | X | X | X | X | X | X | X | X | | X | X | X | | X | | X | | 14 | 77,8 % |
| Änderung der Parametereinstellungen | X | X | X | | | X | X | | X | X | | X | X | X | | X | | | 11 | 61,1 % |
| Adaption des Seiten-Layouts | X | X | | X | | X | | X | X | X | X | | X | X | X | | | | 11 | 61,1 % |
| Integration externer Anwendungen | X | X | | X | X | X | X | | | | X | X | | | X | X | | | 10 | 55,6 % |
| Ein-/Ausblenden der Oberflächenkomp. | X | | | | | | X | X | X | | | X | | | X | | X | | 7 | 38,9 % |
| Änderung der Systemzusammenstellung | | X | | | | X | | | X | | X | | | | | X | | X | 6 | 33,3 % |
| Individualisierung der Geschäftsprozesse | X | | X | | | X | | | | | | | X | X | | | | X | 6 | 33,3 % |
| Definition neuer Dokumentenvorlagen | X | | | X | | X | | X | X | | | | | | | | | | 5 | 27,8 % |
| Adaption der Prozedurabläufe | | | X | X | X | | | X | | | | | | | | | X | | 5 | 27,8 % |
| Integration externer Systemkomponenten | X | X | X | | | | | | | | | | | | X | | | X | 5 | 27,8 % |
| Adaption der standard. Datenformate | X | | | | X | | | | | | | | | X | | | X | | 5 | 27,8 % |
| Adaption der standard. Datenmodelle | | X | | | | X | | | X | | X | | | X | | | | | 5 | 27,8 % |
| Anpassung der standard. Nomenklatur | X | | | | | | | | X | | | | | X | | | X | | 4 | 22,2 % |
| Definition neuer Kennzahlen | X | | | | | | | | X | | | X | | | | | | | 3 | 16,7 % |
| Definition neuer Datenfelder | X | | | | | | | | | | | | | X | | | X | | 3 | 16,7 % |
| Anpassung der Organisationsstruktur | X | | | | | | | | | | | | | | | | X | X | 3 | 16,7 % |
| Adaption der Systemberechtigungen | | | | | | | | X | | | | | | | | | X | | 2 | 11,1 % |
| Definition neuer Datenquellen | X | | | | | | | X | | | | | | | | | | | 2 | 11,1 % |
| Adaption der Dokumentenflüsse | | | | | | | | | X | | X | | | | | | | | 2 | 11,1 % |
| Adaption der Formularansicht | | | | | | | X | | X | | | | | | | | | | 2 | 11,1 % |
| Adaption der Suchfunktionen | | | | | | | | | | | | | | X | | | | | 1 | 5,6 % |
| Adaption der Benutzer-Zuständigkeiten | | | | | | | | X | | | | | | | | | | | 1 | 5,6 % |
| Automatisierung der Prozessabläufe | | X | | | | | | | | | | | | | | | | | 1 | 5,6 % |
| Adaption des Datenimports | | | | | | | | | | | | | | | | | X | | 1 | 5,6 % |
| Adaption der Systemnavigation | | | X | | | | | | | | | | | | | | | | 1 | 5,6 % |
| Adaption der Authorisierungsdienste | | | | | | | | | | | | | | | | | X | | 1 | 5,6 % |

## A.2 Adaptionsanforderungen gruppiert nach Systemelementen

| Systemelement / Adaptionsanforderung | Beijsterveld und Groenendaal (2015) | Brehm et al. (2001) | Callejas und Terzi (2013) | Hustad et al. (2016) | Kanchymalay et al. (2013) | Keckeis et al. (2016) | Leyh (2011) | Li et al. (2011) | Light (2005) | Parthasarathy und Sharma (2016) | Pries-Heje (2006) | Sherrer-Rathje und Boyle (2008) | Seethamraju und Sundar (2013) | Sekatzek und Krcmar (2009) | Soh et al. (2003) | Themistocleous et al. (2001) | Yen et al. (2011) | Zach und Munkvold (2012) | Code |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Zusammenstellung der Systemmodule** | | | | | | | | | | | | | | | | | | | **SE 1** |
| Änderung der Systemzusammenstellung | | X | | | | X | | | X | | X | | | | | X | | X | |
| Integration externer Systemkomponenten | X | X | X | | | | | | | | | | | | X | | | X | |
| **Parametereinstellungen** | | | | | | | | | | | | | | | | | | | **SE 2** |
| Änderung der Parametereinstellungen | X | X | X | | | X | X | | X | X | | X | X | X | | X | | | |
| **Benutzeroberfläche** | | | | | | | | | | | | | | | | | | | **SE 3** |
| Adaption des Seiten-Layouts | X | X | | X | | X | | X | X | X | X | | X | X | X | | | | |
| Ein-/Ausblenden der Oberflächenkomp. | X | | | | | | X | X | X | | | X | | | X | | X | | |
| Anpassung der standard. Nomenklatur | X | | | | | | | | X | | | | | X | | | X | | |
| Definition neuer Dokumentenvorlagen | X | | | | | | | X | X | | | | | | | | | | |
| Adaption der Formularansicht | | | | | | | X | | X | | | | | | | | | | |
| **Geschäftsobjekte** | | | | | | | | | | | | | | | | | | | **SE 4** |
| Adaption der standard. Datenmodelle | | X | | | | X | | | X | | X | | | X | | | | | |
| Adaption der standard. Datenformate | X | | | | X | | | | | | | | | X | | | X | | |
| Definition neuer Datenfelder | X | | | | | | | | | | | | | X | | | X | | |
| **Analytische Komponenten** | | | | | | | | | | | | | | | | | | | **SE 5** |
| Definition neuer Berichte u. Dashboards | X | X | X | X | X | X | X | X | X | | X | X | X | | X | | X | | |
| Definition neuer Kennzahlen | X | | | | | | | | X | | | X | | | | | | | |
| Definition neuer Datenquellen | X | | | | | | | X | | | | | | | | | | | |
| **Organisationsstruktur** | | | | | | | | | | | | | | | | | | | **SE 6** |
| Anpassung der Organisationsstruktur | X | | | | | | | | | | | | | | | | X | X | |
| Adaption der Systemberechtigungen | | | | | | | | X | | | | | | | | | X | | |
| Adaption der Benutzer-Zuständigkeiten | | | | | | | | X | | | | | | | | | | | |
| **Geschäftsprozesse** | | | | | | | | | | | | | | | | | | | **SE 7** |
| Individualisierung der Geschäftsprozesse | X | | X | | | X | | | | | | | X | X | | | | X | |
| Adaption der Prozedurabläufe | | | X | X | X | | | X | | | | | | | | | X | | |
| Adaption der Dokumentenflüsse | | | | | | | | | X | | X | | | | | | | | |
| Automatisierung der Prozessabläufe | | X | | | | | | | | | | | | | | | | | |
| **Anwendungsintegration** | | | | | | | | | | | | | | | | | | | **SE 8** |
| Integration externer Anwendungen | X | X | | X | X | X | X | | | | X | X | | | X | X | | | |
| Integration externer Systemkomponenten | X | X | X | | | | | | | | | | | | X | | | X | |
| **Kundenspezifische Funktionalität** | | | | | | | | | | | | | | | | | | | **SE 9** |
| Definition neuer Systemfunktionalität | X | X | | X | X | X | X | X | X | X | | X | | X | | X | X | X | |
| Adaption der Suchfunktionen | | | | | | | | | | | | | | X | | | | | |
| Adaption des Datenimports | | | | | | | | | | | | | | | | | X | | |
| Adaption der Systemnavigation | | | X | | | | | | | | | | | | | | | | |
| Adaption der Authorisierungsdienste | | | | | | | | | | | | | | | | | X | | |

# B Details der Implementierung

## B.1 Parameterwerte und Quellcode zum Anlegen einer Serveraktion in Odoo

| Parameter/-gruppe | Parameterwert |
|---|---|
| AUTOMATISCHE AKTIONEN | |
| Regelbezeichnung | Genehmigungsregel |
| Zugehöriges Dokumenten-Modell | Verkaufsauftrag |
| BEDINGUNGEN | |
| Wann zu Starten [Auswahl] | Bei Erstellung und Aktualisierung |
| Filter [Python] | [u'&', [u'amount_total', u'>', 2000], [u'x_bestellkanal', u'!=', False]] |
| AKTIONEN | |
| Verantwortlichen zuweisen | Administrator |
| SERVER AKTIONEN | |
| Aktionsbezeichnung | Kundenhistorie überprüfen |
| Basismodell | Verkaufsauftrag |
| Folgeaktion | Create or Copy a Record |
| Zustand | True |
| Reihenfolge | 4 |
| ANLEGEN/SCHREIBEN/KOPIEREN | |
| Erstellungsregel [Auswahl] | Erstellen eines neuen Datensatzes in einem weiteren Modell |
| Ziel-Modell [Auswahl] | Aufgabe |
| Den neuen Beleg verknüpfen | Deaktiviert |
| Aktiv [Wert] | True |
| Aufgabentitel [Python] | "Genehmigung für " + object.name + "erforderlich" |
| Stufe [Wert] | 4 |
| Frist [Python Ausdruck] | object.create_date |
| Beschreibung [Wert] | E-Mail-Vorlage-Quellcode* |
| E-MAIL | |
| Vorlage [Auswahl] | Internal Confirmation |
| Gilt für [Auswahl] | Verkaufsauftrag |
| INHALT | |
| Betreff [Python] | Genehmigung für ${object.name} erforderlich |
| Mitteilung [Python] | Python-Quellcode (unten) |

```
##  Der folgende Quellcode erzeugt eine Aktivitätsanforderung (Aufgabe),
    die in der Abbildung 5.21 (rechts) veranschaulicht ist.
1   <div style=\color: rgb(34, 34, 34);background-color: \#FFF\">
2    <p>Hallo
3      <a href=mailto:" + object.user_id.email + " or \?subject=
4          Auftrag " + object.name + ">" + object.user_id.name + "
5      </a>,
6    <\/p>
7    <p>die Bestellung <strong>" + object.name + "</strong> von <b>" +
8      object.partner_id.name + "</b> braucht Ihre Genehmigung:
9    <\/p>
10   <p>
11     <strong>REFERENZEN</strong>
12   <\/p>
13   <p>
14     Bestellnummer: <strong>" + object.name + "</strong>
15   </p>
16   <p>
17     Bestellbetrag: <strong>" + '%.2f' % object.amount_total + " " +
```

Fortsetzung auf der nächsten Seite

```
18   object.pricelist_id.currency_id.name + "</strong>
19   <\/p>
20   <p>
21     Bestelldatum: <b>" + object.date_order + "<br></b>
22   </p>
23   <p>
24     Bestellkanal: Internet ( <b>" + object.x_bestellkanal + "</b> )
25   <\/p>
26   <p>
27     Sie können die Bestellung auch direkt genehmigen bzw. ablehnen:
28   <\/p>
29   </div>
30   <p>
31     <a class=\"btn btn-success\" href=\"https://it-systems.odoo.com\
32         /web\?debug=\#id=$%7Bobject.id%7D&amp;view_type=form&amp;
33         model=sale.order&amp;action=413&amp;menu_id=63\">
34         GENEHMIGEN
35     </a>
36     <a class=\"btn btn-danger\" href=\"https://it-systems.odoo.com\
37         /web\?debug=\#id=$%7Bobject.id%7D& amp;view_type=form&
38         amp;model=sale.order&amp;action=413&amp;menu_id=63\">
39         ABLEHNEN
40     </a>
41   <\/p>
42 </div>
```

| REFERENZEN | |
|---|---|
| Bestellnummer | ${object.name} |
| Bestellbetrag | ${object.amount_total} ${object.pricelist_id.currency_id.name} |
| Bestelldatum | ${object.date_order} |
| Bestellkanal | Internet (${object.x_bestellkanal in ('01','02','03') and 'Facebook' or 'Twitter' or 'Google+'}) |

## B.2 Programmcode zum Erzeugen einer virtuellen Landkarte

| Nr. | Programmcode |
|---|---|
| ## | *Der folgende HTML/JavaScript-Code erzeugt eine Verbindung mit der GoogleMap-API und visualisiert die aktuellen Verkaufsdaten auf einer virtuellen Landkarte, wie z. B. in der Abbildung 5.23 (rechts) dargestellt ist.* |

```
1  <html>
2   <head>
3    <meta name="viewport" content="initial-scale=1.0,user-scalable=no">
4    <meta charset="utf-8">
5    <title>Verkaufsvolumen pro Bestellkanal</title>
6    <style>
7      html, body {
8        height: 100%;
9        margin: 0;
10       padding: 0;
11     }
12     #map {
13       height: 100%;
14     }
15   </style>
16  </head>
17  <body>
18  <div id="map"></div>
19  <script>
20
21  function kreisHinzufuegen( geocoder, city, map ){
22    geocoder.geocode({'address':city.address},function(results,status){
23
24      if (status == google.maps.GeocoderStatus.OK) {
25        var options = {
26          strokeColor: '#3F48CC',
27          strokeOpacity: 0.8,
28          strokeWeight: 2,
29          fillColor: '#3F48CC',
30          fillOpacity: 0.35,
31          map: map,
32          center: {lat: results[0].geometry.location.lat(),
33                   lng: results[0].geometry.location.lng()},
34          radius: Math.sqrt(city.orderval) * 100
35        }
36
37        if (city.channel == 'Google+'){
38         options.fillColor = '#22B14C';
39         options.strokeColor = '#22B14C';
40        }
41
42        if (city.channel == 'Facebook'){
43         options.fillColor = '#FF0000';
44         options.strokeColor = '#FF0000';
45        }
46
47        var cityCircle = new google.maps.Circle(options);
48      }
49    });
50  }
51
```

Fortsetzung auf der nächsten Seite

| Nr. | Programmcode |
|---|---|
| 52 | function landkarteErzeugen(citymap[city]) { |
| 53 | var glng = 0; |
| 54 | var glat = 0; |
| 55 | var map = new google.maps.Map(document.getElementById('map'), { |
| 56 | zoom: 7, |
| 57 | center: {lat: 51.19, lng: 9.30}, |
| 58 | mapTypeId: google.maps.MapTypeId.TERRAIN |
| 59 | }); |
| 60 | |
| 61 | var geocoder = new google.maps.Geocoder(); |
| 62 | |
| 63 | for (var city in citymap) { |
| 64 | kreisHinzufuegen(geocoder, citymap[city], map); |
| 65 | } |
| 66 | } |
| 67 | |
| 68 | <!-- An dieser Stelle erfolgt der Aufruf der systeminternen API(s)--> |
| 69 | function initMap() { |
| 70 | $.ajax({ |
| 71 | url: "/AuftragsdatenExportierenSOAP", |
| 72 | dataType: 'json', |
| 73 | success: function( data ) { |
| 74 | loadDataToMap(data); |
| 75 | } |
| 76 | }); |
| 77 | } |
| 78 | |
| 79 | </script> |
| 80 | <script async defer src="https://maps.googleapis.com/maps/api/ |
| 81 | js?signed_in=true&callback=initMap"></script> |
| 82 | </body> |
| 83 | </html> |

## B.3 Programmcode zum Import der systeminternen Daten (Scopevisio-API)

| Nr. | Programmcode |
|---|---|
| ## | *Der folgende PHP-Skript erzeugt eine Verbindung mit der Scopevisio-API und importiert die aktuellen Auftragsdaten.* |

```
1   <?php
2    $app->get('/AuftragsdatenExportierenSOAP', function () {
3        $amount_cities = array();
4     try{
5
6      $urlCustomer = "https://appload.scopevisio.com/api/soap/contact/
7             Contact.exportExtendedCSV";
8
9      $urlOrderval = "https://appload.scopevisio.com/api/soap/accounting/
10                    SusaCustomer.read";
11
12     $customer = "number";
13     $user = "user@domain";
14     $pass = "password";
15     $organisation = "IT.Systems";
16
17     $requestCustomer = new SoapClient(NULL, array('trace' => true,
18                             'location' => $urlCustomer,
19                             'uri' => "https://www.scopevisio.com/"));
20
21     $requestOrderval = new SoapClient(NULL, array('trace' => true,
22                             'location' => $urlOrderval,
23                             'uri' => "https://www.scopevisio.com/"));
24
25     $authn = '
26      <authn>
27        <customer>' . $customer . '</customer>
28        <user>' . $user . '</user>
29        <pass>' . $pass . '</pass>
30        <language>de_DE</language>
31        <organisation>' . $organisation . '</organisation>
32      </authn>';
33
34     $paramsCustomer = $authn . '
35      <args>
36        <createdSinceTimestamp>1357599600000</createdSinceTimestamp>
37       <columns> Stadt1, debitorNumber,Bestellkanal</columns>
38      </args>';
39
40     $paramsOrderval = $authn . '
41      <args>
42        <outputFormat>csv</outputFormat>
43        <startDate>01.01.2016</startDate>
44        <endDate>31.12.2016</endDate>
45      </args>';
46
47      $responseCustomer = $requestCustomer->req(new SoapVar($paramsCustomer,
48                              XSD_ANYXML));
49
50      $responseOrderval = $requestOrderval->req(new SoapVar($paramsOrderval
51                              XSD_ANYXML));
52
53      $data = explode("\n", $response['data']);
54
```

Fortsetzung auf der nächsten Seite

| Nr. | Programmcode |
|---|---|
| 55 | `    foreach ($data as $singleVal){` |
| 56 | `        $singleVal = explode(";", $singleVal);` |
| 57 | |
| 58 | `        if(is_numeric($singleVal[1])){` |
| 59 | `            $newRecord = array();` |
| 60 | `            $newRecord["orderval"] = AuftragsdatenParsen($response_` |
| 61 | `                                     Orderval-> susa, $singleVal[1]);` |
| 62 | `            $newRecord["address"] = $singleVal[0];` |
| 63 | `            $newRecord["channel"] = $singleVal[2];` |
| 64 | `            $amount_cities[$singleVal[1]] = $newRecord;` |
| 65 | `        }` |
| 66 | `    }` |
| 67 | |
| 68 | `  }catch(Exception $e){` |
| 69 | `    $error = $e->getMessage();` |
| 70 | `    return json_encode($error);` |
| 71 | `  }` |
| 72 | `  return json_encode($amount_cities);` |
| 73 | `});` |
| 74 | |
| 75 | `function AuftragsdatenParsen($responseOrderval, $customerNumber){` |
| 76 | `  $data = explode("\n", $responseOrderval);` |
| 77 | `  $result = 0;` |
| 78 | |
| 79 | `  foreach ($data as $singleVal){` |
| 80 | `    $singleVal = explode(";", $singleVal);` |
| 81 | |
| 82 | `    if($singleVal[0] == $customerNumber){` |
| 83 | `            $result += $singleVal[3];` |
| 84 | `    }` |
| 85 | `  }` |
| 86 | `  return $result;` |
| 87 | `}` |
| 88 | |
| 89 | `$app->get('/', function () use ($app) {` |
| 90 | `  return $app['twig']->render('home.twig', array());` |
| 91 | `});` |
| 92 | |
| 93 | `$app->run();` |
| 94 | `?>` |

# C Details der Umfrage und Datenauswertung

## C.1 Vordruck des elektronischen Fragebogens

**Herzlich Willkommen !**

Sehr geehrte Damen und Herren,

vielen Dank, dass Sie sich entschieden haben, an dieser Online-Befragung über die Customizing-Strategien der cloudbasierten ERP-Systeme teilzunehmen. Diese Umfrage stellt einen Teil des Doktoranden-Projekts

**"Systematisierung und Bewertung der Customizing- und Modifikationsmöglichkeiten von cloudbasierten ERP-Systemen"**

dar, welches am Lehrstuhl für Wirtschaftsinformatik an der Europa-Universität Viadrina in Frankfurt (Oder) durchgeführt wird.

**Hintergrund:** Aufbauend auf den Prinzipien des Cloud-Computing-Paradigma kann die ERP-Standardsoftware heutzutage dem Anwenderunternehmen auch in Form einer Dienstleistung über das Internet zur Verfügung gestellt werden. Die Anforderungen dieses Bereitstellungsmodells bleiben jedoch nicht ohne Auswirkung auf die bereitgestellte Systemflexibilität. Die zahlreichen Studien weisen auf die begrenzte Adaptierbarkeit der cloudbasierten ERP-Systeme hin. Zwischen den Autoren herrscht jedoch keine Übereinstimmung hinsichtlich des bereitgestellten Flexibilitätsumfangs bzw. der Strategien der Systemadaption.

**Ziel:** Das Hauptziel dieser Umfrage besteht in der Identifizierung der Gemeinsamkeiten und Differenzen in den bereitgestellten Customizing-Strategien differenzierter cloudbasierter ERP-Systeme. Aufbauend auf den Ergebnissen der Online-Befragung sollen verallgemeinerbare Erkenntnisse bezüglich des bereitgestellten Flexibilitätsumfangs abgeleitet werden sowie die Zusammenhänge zwischen den einzelnen Elementen der Customizing-Strategie und der Systemarchitektur verifiziert werden. Diese Erkenntnisse liefern einen Beitrag zur Konstruktion eines Bezugsrahmens, der als praktische Hilfestellung bei der Bewertung der Systemflexibilität differenzierter Cloud-ERP-Systeme verwendet werden kann.

**Zielgruppe:** Systemhersteller, Systemanbieter, Systemberater, Erweiterungsprogrammierer.

**Format:** Die Umfrage besteht aus **20 Fragen** und dauert etwa **10 Minuten**.

**Umfrage starten**

**Vertraulichkeit:** Ihre Antworten werden anonym erfasst und vertraulich behandelt. Es werden keine privaten Informationen über die Teilnehmer gespeichert. Die Ergebnisse der Befragung werden in einer aggregierten Form veröffentlicht.

**Kontakt:** Bei Rückfragen zum Fragebogen kontaktieren Sie bitte Dawid Nowak, Europa-Universität Viadrina, Frankfurt (Oder) unter: +49-335-5534-2302 oder per E-Mail: danowak@europa-uni.de.

Für Ihre Teilnahme an der Befragung möchte ich mich schon jetzt herzlich bedanken.

Mit freundlichen Grüßen
Dawid Nowak

## Allgemeine Angaben

FPA01

**1. Welches der folgenden Unternehmensprofile der ERP-Branche beschreibt Ihr Unternehmen am besten?**

Bitte wählen Sie **eine** der folgenden Antworten aus:

- ❑ ERP-Systemhersteller
- ❑ ERP-Systemanbieter
- ❑ Partner- / Beraterunternehmen
- ❑ Unabhängiger Entwickler
- ❑ Sonstiges, bitte näher spezifizieren …….....

FPA02

**2. In welcher Position sind Sie gerade tätig?**

Bitte schreiben Sie Ihre Antwort auf:

FPA03

**3. Wie viele Jahre Berufserfahrung haben Sie in Ihrer aktuellen Tätigkeit?**

Bitte wählen Sie **eine** der folgenden Antworten aus:

| < 1 Jahr | 1-3 Jahre | 3-5 Jahre | 5-7 Jahre | 7-10 Jahre | > 10 Jahre |
|---|---|---|---|---|---|
| ❑ | ❑ | ❑ | ❑ | ❑ | ❑ |

FPA04

**4. Wie viele Jahre Erfahrung haben Sie mit den cloudbasierten ERP-Systemen?**

Bitte wählen Sie **eine** der folgenden Antworten aus:

| < 1 Jahr | 1-2 Jahre | 2-3 Jahre | 3-4 Jahre | 4-5 Jahre | > 5 Jahre |
|---|---|---|---|---|---|
| ❑ | ❑ | ❑ | ❑ | ❑ | ❑ |

FPA05

**5. Welche cloudbasierte ERP-Systeme sind Ihnen bekannt?**

Bitte schreiben Sie Ihre Antwort auf:

FPA06

**6. Aus Sicht welches cloudbasierten ERP-Systems wird der Fragebogen ausgefüllt?**

Bitte schreiben Sie Ihre Antwort auf:

## TEIL I: Systemarchitektur

FSA01

**7. In welchen Betreibermodellen wird das ERP-System angeboten?**

Bitte wählen Sie **alle** zutreffenden Antworten aus:

- ❑ Software-as-a-Service
- ❑ Plattform-as-a-Service
- ❑ Infrastructure-as-a-Service
- ❑ On-Premise
- ❑ Sonstiges, bitte näher spezifizieren ……

FSA02

**8. In welchem Cloud-Bereitstellungsmodell wird das ERP-System primär angeboten?**

**Bitte beachten Sie**: Alle Fragen bezüglich der Systemadaption sollen aus der Perspektive des primär ausgewählten Cloud-Bereitstellungsmodells beantwortet werden.

Bitte wählen Sie **eine** der folgenden Antworten aus:

- ❑ Public Cloud
- ❑ Private Cloud
- ❑ Hybrid Cloud
- ❑ Sonstiges, bitte näher spezifizieren ……

FSA03

**9. In welchen anderen Cloud-Bereitstellungsmodellen wird das cloudbasierte ERP-System angeboten?**

Bitte wählen Sie **alle** zutreffenden Antworten aus:

- ❑ Public Cloud
- ❑ Private Cloud
- ❑ Hybrid Cloud
- ❑ On-Premise
- ❑ Sonstiges, bitte näher spezifizieren ……

**10. Stellt das genannte ERP-System eine vollständig cloudnative Anwendung dar?** FSA04

Bitte wählen Sie **eine** der folgenden Antworten aus:

❑ Ja, das ERP-System wurde von Anfang an als cloudnative Anwendung konzipiert und entwickelt.

❑ Nein, es stellt eine cloudfähige Version des gerade existierenden konventionellen ERP-Systems dar.

Bemerkung zu Ihrer Auswahl:

**11. Welche Merkmale einer cloudnativen Anwendung weist die Architektur des genannten ERP-Systems auf?** FSA05

Bitte wählen Sie **alle** zutreffenden Antworten aus:

❑ Cloudfähigkeit
❑ Mandantenfähigkeit
❑ Konfigurierbarkeit
❑ Skalierbarkeit

**12. Wenn das ERP-System mandantenfähig ist; welches Konzept zur Sicherstellung der Mandantenfähigkeit wurde auf der Ebene des Anwendungsservers angewendet?** FSA06 Abhängig von FSA05

Bitte wählen Sie **eine** der folgenden Antworten aus:

❑ Gemeinsame Anwendungsinstanz

❑ Gemeinsame Middleware

❑ Gemeinsamer Anwendungsserver

❑ Sonstiges, bitte näher spezifizieren ……

**13. Wenn das ERP-System mandantenfähig ist; welches Konzept zur Sicherstellung der Mandantenfähigkeit wurde auf der Ebene des Datenbankservers angewendet?** FSA07 Abhängig von FSA05

Bitte wählen Sie **eine** der folgenden Antworten aus:

❑ Gemeinsames Datenschema

❑ Gemeinsame Datenbank

❑ Gemeinsamer Datenbankserver

❑ Sonstiges, bitte näher spezifizieren ……

## TEIL II: Customizing-Ansätze

**14. Welche der aufgelisteten Customizing-Ansätze stellen einen integralen Bestandteil der Customizing-Strategie des Systemanbieters dar?** FCA01

Bitte wählen Sie die zutreffende Antwort für jeden Punkt aus:

| Customizing-Ansatz | verfügbar | nicht verfügbar |
|---|---|---|
| Komposition | ❑ | ❑ |
| Parametrisierung | ❑ | ❑ |
| Modellbasierte Generierung | ❑ | ❑ |
| User-Exits | ❑ | ❑ |
| Programmierschnittstellen (APIs) | ❑ | ❑ |
| Systemerweiterungen | ❑ | ❑ |
| Codebasierte Modifikation | ❑ | ❑ |

**15. Welche Rolle spielen die folgenden Customizing-Ansätze im Rahmen der gesamten Customizing-Strategie des Systemanbieters?** FCA02 Abhängig von FCA01

Bitte wählen Sie die zutreffende Antwort für jeden Punkt aus:

| Customizing-Ansatz | keine | gering-fügige | mehr oder weniger große | große | sehr große |
|---|---|---|---|---|---|
| Komposition | ❑ | ❑ | ❑ | ❑ | ❑ |
| Parametrisierung | ❑ | ❑ | ❑ | ❑ | ❑ |
| Modellbasierte Generierung | ❑ | ❑ | ❑ | ❑ | ❑ |
| User-Exits | ❑ | ❑ | ❑ | ❑ | ❑ |
| Programmierschnittstellen (APIs) | ❑ | ❑ | ❑ | ❑ | ❑ |
| Systemerweiterungen | ❑ | ❑ | ❑ | ❑ | ❑ |
| Codebasierte Modifikation | ❑ | ❑ | ❑ | ❑ | ❑ |

## TEIL III: Adaptierbare Systemelemente

FSE01

**16. In welchem Umfang können die einzelnen Systemschichten adaptiert werden?**

Bitte wählen Sie die zutreffende Antwort für jeden Punkt aus:

| Systemschicht | gar nicht | gering-fügig | mehr oder weniger stark | stark | sehr stark |
|---|---|---|---|---|---|
| Präsentationsschicht | ❑ | ❑ | ❑ | ❑ | ❑ |
| Integrationsschicht | ❑ | ❑ | ❑ | ❑ | ❑ |
| Geschäftslogikschicht | ❑ | ❑ | ❑ | ❑ | ❑ |
| Datenzugriffsschicht | ❑ | ❑ | ❑ | ❑ | ❑ |

FSE02

**17. In welchem Umfang können die folgenden Systemelemente von einem Anwendungsexperten angepasst werden?**

Bitte wählen Sie die zutreffende Antwort für jeden Punkt aus:

| Systemelemente | gar nicht | gering-fügig | mehr oder weniger stark | stark | sehr stark |
|---|---|---|---|---|---|
| Zusammenstellung der Systemmodule | ❑ | ❑ | ❑ | ❑ | ❑ |
| Parametereinstellungen | ❑ | ❑ | ❑ | ❑ | ❑ |
| Benutzeroberfläche | ❑ | ❑ | ❑ | ❑ | ❑ |
| Geschäftsobjekte | ❑ | ❑ | ❑ | ❑ | ❑ |
| Analytische Komponenten | ❑ | ❑ | ❑ | ❑ | ❑ |
| Organisationsstruktur | ❑ | ❑ | ❑ | ❑ | ❑ |
| Geschäftsprozesse | ❑ | ❑ | ❑ | ❑ | ❑ |
| Anwendungsintegration | ❑ | ❑ | ❑ | ❑ | ❑ |
| Kundenspezifische Funktionalität | ❑ | ❑ | ❑ | ❑ | ❑ |

## TEIL IV: Customizing-Werkzeuge

**18. Welche der folgenden Werkzeug-Kategorien werden von dem Systemanbieter im Rahmen der Customizing-Strategie bereitgestellt?** FCW01

Bitte wählen Sie die zutreffende Antwort für jeden Punkt aus:

| Werkzeug-Kategorie | verfügbar | nicht verfügbar |
|---|---|---|
| **Integrierte Adaptionswerkzeuge** *sind direkt im Systemrahmen vorhanden und über die Benutzeroberfläche zugänglich. Sie stellen einen integralen untrennbaren Systembestandteil dar.* | ❑ | ❑ |
| **Externe lokal installierbare Adaptionswerkzeuge** *werden oft optional als komplementäre Software bereitgestellt. Sie werden lokal auf einem Rechner des Anwenders bzw. Entwicklers installiert.* | ❑ | ❑ |
| **Externe cloudbasierte Adaptionswerkzeuge** *werden ähnlich wie das Basissystem in der Cloud bereitgestellt und oft mit dem System-Backend in der PaaS gekoppelt.* | ❑ | ❑ |

Bemerkung zu Ihrer Auswahl:

**19. Welche Rolle für die anwenderseitigen Customizing-Prozesse spielen die folgenden Werkzeug-Kategorien?** FCW02 (Abhängig von FCW01)

Bitte wählen Sie die zutreffende Antwort für jeden Punkt aus:

| Werkzeug-Kategorie | keine | gering-fügige | mehr oder weniger große | große | sehr große |
|---|---|---|---|---|---|
| Integrierte Adaptionswerkzeuge | ❑ | ❑ | ❑ | ❑ | ❑ |
| Externe lokal installierbare Adaptionswerkzeuge | ❑ | ❑ | ❑ | ❑ | ❑ |
| Externe cloudbasierte Adaptionswerkzeuge | ❑ | ❑ | ❑ | ❑ | ❑ |

**20. Welche Mechanismen und Konzepte zur Unterstützung der anwenderseitigen Adaptionen der folgenden Systemelemente werden in den bereitgestellten Werkzeugen verankert?**

FCW03

Abhängig von FSE02

Bitte wählen Sie die zutreffende Antwort für jeden Punkt aus:

| Systemelemente | passive Beratung | aktive Anleitung | Prozess-lenkung | Prozess-automation | keine |
|---|---|---|---|---|---|
| Zusammenstellung der Systemmodule | ❑ | ❑ | ❑ | ❑ | ❑ |
| Parametereinstellungen | ❑ | ❑ | ❑ | ❑ | ❑ |
| Benutzeroberfläche | ❑ | ❑ | ❑ | ❑ | ❑ |
| Geschäftsobjekte | ❑ | ❑ | ❑ | ❑ | ❑ |
| Analytische Komponenten | ❑ | ❑ | ❑ | ❑ | ❑ |
| Organisationsstruktur | ❑ | ❑ | ❑ | ❑ | ❑ |
| Geschäftsprozesse | ❑ | ❑ | ❑ | ❑ | ❑ |
| Anwendungsintegration | ❑ | ❑ | ❑ | ❑ | ❑ |
| Kundenspezifische Funktionalität | ❑ | ❑ | ❑ | ❑ | ❑ |

## Ihre Bemerkungen

FBEM

Falls Sie mir noch Kommentare zum Fragebogen zukommen lassen möchten, können Sie dies hier tun.

Folgende Rückmeldung zum Fragebogen möchte ich Ihnen noch geben:

## C.2 Ergebnisse der Spearman‘schen Rangkorrelation zwischen untersuchten Variablen

| Merkmal | Spearman-Statistik | Customizing-Ansatz | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | CA1 | CA2 | CA3 | CA4 | CA5 | CA6 | CA7 |
| **REIFE** | Korrelationskoeffizient | 0,524** | 0,663** | 0,214 | -0,342 | 0,046 | 0,009 | -0,400* |
| | Sig. (2-seitig) | 0,002 | 0,001 | 0,248 | 0,060 | 0,807 | 0,962 | 0,026 |
| | N | 31 | 31 | 31 | 31 | 31 | 31 | 31 |

| Merkmal | Spearman-Statistik | Systemelement | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | SE1 | SE2 | SE3 | SE4 | SE5 | SE6 | SE7 | SE8 | SE9 |
| **REIFE** | Korrelationskoeffizient | 0,561** | 0,502** | 0,268 | 0,096 | 0,266 | 0,269 | 0,303 | 0,526** | 0,308 |
| | Sig. (2-seitig) | 0,001 | 0,002 | 0,144 | 0,608 | 0,148 | 0,143 | 0,098 | 0,002 | 0,091 |
| | N | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 |

| Merkmal | Spearman-Statistik | Werkzeug-Kategorie | | |
|---|---|---|---|---|
| | | CW1 | CW2 | CW3 |
| **REIFE** | Korrelationskoeffizient | 0,114 | 0,436* | 0,392* |
| | Sig. (2-seitig) | 0,541 | 0,014 | 0,029 |
| | N | 31 | 31 | 31 |

| Merkmal | Spearman-Statistik | Konzepte der Anwenderunterstützung | | | |
|---|---|---|---|---|---|
| | | AU1 | AU2 | AU3 | AU4 |
| **REIFE** | Korrelationskoeffizient | 0,231 | 0,139 | 0,399* | 0,642** |
| | Sig. (2-seitig) | 0,211 | 0,456 | 0,026 | 0,000 |
| | N | 31 | 31 | 31 | 31 |

| Merkmal | | Spearman-Statistik | Systemelement | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | SE1 | SE2 | SE3 | SE4 | SE5 | SE6 | SE7 | SE8 | SE9 |
| **Customizing-Ansatz** | **CA1** | Korrelationskoeffizient | 0,752** | 0,474** | 0,358* | 0,176 | 0,181 | -0,044 | 0,544** | 0,346 | 0,352 |
| | | Sig. (2-seitig) | 0,000 | 0,007 | 0,048 | 0,343 | 0,330 | 0,814 | 0,002 | 0,056 | 0,052 |
| | | N | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 |
| | **CA2** | Korrelationskoeffizient | 0,434* | 0,387* | 0,149 | 0,078 | 0,219 | 0,170 | 0,431* | 0,548** | 0,222 |
| | | Sig. (2-seitig) | 0,015 | 0,032 | 0,425 | 0,679 | 0,236 | 0,361 | 0,015 | 0,001 | 0,230 |
| | | N | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 |
| | **CA3** | Korrelationskoeffizient | 0,313 | 0,065 | 0,160 | 0,035 | 0,139 | -0,057 | 0,146 | 0,282 | 0,279 |
| | | Sig. (2-seitig) | 0,087 | 0,727 | 0,391 | 0,851 | 0,454 | 0,761 | 0,433 | 0,124 | 0,129 |
| | | N | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 |
| | **CA4** | Korrelationskoeffizient | -0,018 | -0,215 | 0,201 | 0,146 | 0,262 | -0,077 | 0,093 | 0,052 | 0,164 |
| | | Sig. (2-seitig) | 0,925 | 0,245 | 0,278 | 0,433 | 0,155 | 0,682 | 0,619 | 0,783 | 0,379 |
| | | N | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 |
| | **CA5** | Korrelationskoeffizient | 0,165 | 0,186 | 0,018 | 0,050 | 0,257 | 0,049 | 0,088 | 0,266 | 0,088 |
| | | Sig. (2-seitig) | 0,375 | 0,315 | 0,924 | 0,788 | 0,162 | 0,793 | 0,640 | 0,148 | 0,638 |
| | | N | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 |
| | **CA6** | Korrelationskoeffizient | 0,363* | -0,071 | 0,456* | 0,245 | 0,173 | 0,193 | 0,352 | 0,059 | 0,603** |
| | | Sig. (2-seitig) | 0,045 | 0,702 | 0,010 | 0,184 | 0,351 | 0,297 | 0,052 | 0,754 | 0,000 |
| | | N | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 |
| | **CA7** | Korrelationskoeffizient | -0,386* | -0,303 | -0,089 | 0,001 | -0,118 | -0,030 | -0,134 | -0,055 | -0,295 |
| | | Sig. (2-seitig) | 0,032 | 0,098 | 0,632 | 0,995 | 0,526 | 0,872 | 0,472 | 0,770 | 0,107 |
| | | N | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 |

**Legende:** * Korrelation ist bei Niveau 0,05 signifikant (zweiseitig). ** Korrelation ist bei Niveau 0,01 signifikant (zweiseitig).

C.3 Ergebnisse des Mann-Whitney-U-Tests zu paarweisen Vergleichen auf Inhomogenitäten beim Einsatz der Customizing-Ansätze zwischen den einzelnen Reifegrad-Gruppen

| Customizing-Ansatz | Reife | Reife | N | Mittlerer Rang | Rangsumme | Mann-Whitney-U | Signifikanzwert |
|---|---|---|---|---|---|---|---|
| CA1 | R1 | R2 | 4 | 5,50 | 22,00 | 12,000* | 0,042 |
| | | R3 | 4 | 3,50 | 14,00 | 4,000 | 0,237 |
| | | R4 | 4 | 2,75 | 11,00 | 1,000** | 0,009 |
| | R2 | R1 | 16 | 11,75 | 188,00 | 12,000* | 0,042 |
| | | R3 | 16 | 10,25 | 164,00 | 28,000 | 0,682 |
| | | R4 | 16 | 10,19 | 163,00 | 27,000* | 0,034 |
| | R3 | R1 | 4 | 5,50 | 22,00 | 4,000 | 0,237 |
| | | R2 | 4 | 11,50 | 46,00 | 28,000 | 0,682 |
| | | R4 | 4 | 5,00 | 20,00 | 10,000 | 0,375 |
| | R4 | R1 | 7 | 7,86 | 55,00 | 1,000** | 0,009 |
| | | R2 | 7 | 16,14 | 113,00 | 27,000* | 0,034 |
| | | R3 | 7 | 6,57 | 46,00 | 10,000 | 0,375 |
| CA2 | R1 | R2 | 4 | 4,50 | 18,00 | 8,000* | 0,010 |
| | | R3 | 4 | 2,75 | 11,00 | 1,000* | 0,032 |
| | | R4 | 4 | 2,63 | 10,50 | 0,500** | 0,005 |
| | R2 | R1 | 16 | 12,00 | 192,00 | 8,000* | 0,010 |
| | | R3 | 16 | 9,94 | 159,00 | 23,000 | 0,309 |
| | | R4 | 16 | 9,84 | 157,50 | 21,500** | 0,009 |
| | R3 | R1 | 4 | 6,25 | 25,00 | 1,000* | 0,032 |
| | | R2 | 4 | 12,75 | 51,00 | 23,000 | 0,309 |
| | | R4 | 4 | 4,75 | 19,00 | 9,000 | 0,223 |
| | R4 | R1 | 7 | 7,93 | 55,50 | 0,500** | 0,005 |
| | | R2 | 7 | 16,93 | 118,50 | 21,500** | 0,009 |
| | | R3 | 7 | 6,71 | 47,00 | 9,000 | 0,223 |
| CA3 | R1 | R2 | 4 | 8,50 | 34,00 | 26,000 | 0,278 |
| | | R3 | 4 | 4,50 | 18,00 | 8,000 | 1,000 |
| | | R4 | 4 | 4,50 | 18,00 | 10,000 | 0,148 |
| | R2 | R1 | 16 | 11,00 | 176,00 | 26,000 | 0,278 |
| | | R3 | 16 | 11,00 | 176,00 | 26,000 | 0,278 |
| | | R4 | 16 | 11,31 | 181,00 | 50,000 | 0,363 |
| | R3 | R1 | 4 | 4,50 | 18,00 | 8,000 | 1,000 |
| | | R2 | 4 | 8,50 | 34,00 | 26,000 | 0,278 |
| | | R4 | 4 | 4,50 | 18,00 | 10,000 | 0,148 |
| | R4 | R1 | 7 | 6,86 | 48,00 | 10,000 | 0,148 |
| | | R2 | 7 | 13,57 | 95,00 | 50,000 | 0,363 |
| | | R3 | 7 | 6,86 | 48,00 | 10,000 | 0,148 |
| CA4 | R1 | R2 | 4 | 13,63 | 54,50 | 19,500 | 0,216 |
| | | R3 | 4 | 5,75 | 23,00 | 3,000 | 0,134 |
| | | R4 | 4 | 8,25 | 33,00 | 5,000 | 0,071 |
| | R2 | R1 | 16 | 9,72 | 155,50 | 19,500 | 0,216 |
| | | R3 | 16 | 10,88 | 174,00 | 26,000 | 0,542 |
| | | R4 | 16 | 12,84 | 205,50 | 42,500 | 0,328 |
| | R3 | R1 | 4 | 3,25 | 13,00 | 3,000 | 0,134 |
| | | R2 | 4 | 9,00 | 36,00 | 26,000 | 0,542 |
| | | R4 | 4 | 6,25 | 25,00 | 13,000 | 0,827 |
| | R4 | R1 | 7 | 4,71 | 33,00 | 5,000 | 0,071 |
| | | R2 | 7 | 10,07 | 70,50 | 42,500 | 0,328 |
| | | R3 | 7 | 5,86 | 41,00 | 13,000 | 0,827 |
| CA5 | R1 | R2 | 4 | 10,50 | 42,00 | 32,000 | 1,000 |
| | | R3 | 4 | 5,00 | 20,00 | 6,000 | 0,317 |
| | | R4 | 4 | 5,50 | 22,00 | 12,000 | 0,629 |
| | R2 | R1 | 16 | 10,50 | 168,00 | 32,000 | 1,000 |
| | | R3 | 16 | 10,88 | 174,00 | 26,000 | 0,546 |
| | | R4 | 16 | 11,53 | 184,50 | 48,500 | 0,594 |

Fortsetzung auf der nächsten Seite

| Customizing-Ansatz | Reife | Reife | N | Mittlerer Rang | Rangsumme | Mann-Whitney-U | Signifikanzwert |
|---|---|---|---|---|---|---|---|
| CA5 | R3 | R1 | 4 | 4,00 | 16,00 | 6,000 | 0,317 |
| | | R2 | 4 | 9,00 | 36,00 | 26,000 | 0,546 |
| | | R4 | 4 | 4,75 | 19,00 | 9,000 | 0,272 |
| | R4 | R1 | 7 | 6,29 | 44,00 | 12,000 | 0,629 |
| | | R2 | 7 | 13,07 | 91,50 | 48,500 | 0,594 |
| | | R3 | 7 | 6,71 | 47,00 | 9,000 | 0,272 |
| CA6 | R1 | R2 | 4 | 8,00 | 32,00 | 22,000 | 0,326 |
| | | R3 | 4 | 4,50 | 18,00 | 6,000 | 1,000 |
| | | R4 | 4 | 6,00 | 24,00 | 14,000 | 1,000 |
| | R2 | R1 | 16 | 11,13 | 178,00 | 22,000 | 0,326 |
| | | R3 | 16 | 10,97 | 175,50 | 29,000 | 0,464 |
| | | R4 | 16 | 12,03 | 192,50 | 55,500 | 0,973 |
| | R3 | R1 | 4 | 4,50 | 18,00 | 6,000 | 1,000 |
| | | R2 | 4 | 8,63 | 34,50 | 29,000 | 0,464 |
| | | R4 | 4 | 5,88 | 23,50 | 14,000 | 0,923 |
| | R4 | R1 | 7 | 6,00 | 42,00 | 14,000 | 1,000 |
| | | R2 | 7 | 11,93 | 83,50 | 55,500 | 0,973 |
| | | R3 | 7 | 6,07 | 42,50 | 14,000 | 0,923 |
| CA7 | R1 | R2 | 4 | 13,63 | 54,50 | 19,500 | 0,091 |
| | | R3 | 4 | 5,50 | 22,00 | 4,000 | 0,131 |
| | | R4 | 4 | 7,75 | 31,00 | 7,000* | 0,049 |
| | R2 | R1 | 16 | 9,72 | 155,50 | 19,500 | 0,091 |
| | | R3 | 16 | 10,75 | 172,00 | 28,000 | 0,468 |
| | | R4 | 16 | 12,44 | 199,00 | 49,000 | 0,339 |
| | R3 | R1 | 4 | 3,50 | 14,00 | 4,000 | 0,131 |
| | | R2 | 4 | 9,50 | 38,00 | 28,000 | 0,468 |
| | | R4 | 4 | 6,00 | 24,00 | 14,000 | 1,000 |
| | R4 | R1 | 7 | 5,00 | 35,00 | 7,000* | 0,049 |
| | | R2 | 7 | 11,00 | 77,00 | 49,000 | 0,339 |
| | | R3 | 7 | 6,00 | 42,00 | 14,000 | 1,000 |

**Legende:** * MWU ist bei Niveau 0,05 signifikant (zweiseitig). ** MWU ist bei Niveau 0,01 signifikant (zweiseitig).

## C.4 Ergebnisse des Mann-Whitney-U-Tests zu paarweisen Vergleichen auf Inhomogenitäten beim Flexibilitätsumfang der Systemelemente zwischen den Service-Modellen

| Systemelement | Service-Modell | N | Mittlerer Rang | Rangsumme | Mann-Whitney-U | Signifikanzwert |
|---|---|---|---|---|---|---|
| SE1 | SaaS | 25 | 14,28 | 357,00 | 32,000* | 0,023 |
| | PaaS | 6 | 23,17 | 139,00 | | |
| SE2 | SaaS | 25 | 15,10 | 377,50 | 52,500 | 0,237 |
| | PaaS | 6 | 19,75 | 118,50 | | |
| SE3 | SaaS | 25 | 14,64 | 366,00 | 41,000 | 0,068 |
| | PaaS | 6 | 21,67 | 130,00 | | |
| SE4 | SaaS | 25 | 14,60 | 365,00 | 40,000 | 0,070 |
| | PaaS | 6 | 21,83 | 131,00 | | |
| SE5 | SaaS | 25 | 14,74 | 368,50 | 43,500 | 0,105 |
| | PaaS | 6 | 21,25 | 127,50 | | |
| SE6 | SaaS | 25 | 16,08 | 402,00 | 73,000 | 0,900 |
| | PaaS | 6 | 15,67 | 94,00 | | |
| SE7 | SaaS | 25 | 14,30 | 357,50 | 32,500* | 0,015 |
| | PaaS | 6 | 23,08 | 138,50 | | |
| SE8 | SaaS | 25 | 14,10 | 352,50 | 27,500** | 0,007 |
| | PaaS | 6 | 23,92 | 143,50 | | |
| SE9 | SaaS | 25 | 14,34 | 358,50 | 33,500* | 0,017 |
| | PaaS | 6 | 22,92 | 137,50 | | |

**Legende:** * MWU ist bei Niveau 0,05 signifikant (zweiseitig). ** MWU ist bei Niveau 0,01 signifikant (zweiseitig).

# Ehrenwörtliche Erklärung zu meiner Dissertation mit dem Titel

**Systematisierung und Bewertung der Customizing- und Modifikationsmöglichkeiten von cloudbasierten ERP-Systemen**

Sehr geehrte Damen und Herren,

hiermit erkläre ich, dass ich die beigefügte Dissertation selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel genutzt habe. Alle wörtlich oder inhaltlich übernommenen Stellen habe ich als solche gekennzeichnet.

Ich versichere außerdem, dass ich die beigefügte Dissertation nur in diesem und keinem anderen Promotionsverfahren eingereicht habe und, dass diesem Promotionsverfahren keine endgültig gescheiterten Promotionsverfahren vorausgegangen sind. Ich erkläre ehrenwörtlich, dass ich bisher an keiner Doktorprüfung teilgenommen habe.

Frankfurt (Oder), 05.05.2017

| | |
|---|---|
| Ort, Datum | Unterschrift |

# Résumé

## Dawid Nowak


| | |
|---|---|
| Geburtsdatum: | 06.05.1984, in Choszczno, Polen |
| Anschrift: | Główna 64, 73-231 Chłopowo, Polen |
| E-Mail: | dawid.nowak@yahoo.de |


### Akademischer Werdegang

| | |
|---|---|
| 01/2014 – 09/2017 | **Europa-Universität Viadrina, Frankfurt (Oder)**<br>Doktorand am Lehrstuhl für Allgemeine Betriebswirtschaftslehre, insb. Wirtschaftsinformatik, Promotion zum Dr. rer. pol.<br>Titel der Doktorarbeit: „Systematisierung und Bewertung der Customizing- und Modifikationsmöglichkeiten von cloudbasierten ERP-Systemen" |
| 10/2006 – 06/2009 | **Adam Mickiewicz University, Poznań, Polen**<br>Studium der Angewandten Informatik mit den Schwerpunkten Optimierungsmethoden, Algorithmen und Datenstrukturen, Speicherprogrammierbare Steuerung, Abschluss: Bachelor of Science |
| 09/2003 – 07/2008 | **Europa-Universität Viadrina, Frankfurt (Oder)**<br>Studium der Betriebswirtschaftslehre mit den Schwerpunkten Controlling, Logistik, Finanzierung, Abschluss: Diplom-Kaufmann<br>Titel der Diplomarbeit: „Bewertung von optimalen Optionsportfolien" |

### Beruflicher Werdegang

| | |
|---|---|
| 10/2016 – heute | **Freiberuflicher Software-Entwickler und IT-Berater**<br>IT-Beratung, -Projektmanagement und Software-Entwicklung mit besonderem Schwerpunkt auf Cloud Technologien |
| 10/2011 – 03/2016 | **Europa-Universität Viadrina in Frankfurt (Oder)**<br>Akademischer Mitarbeiter am Lehrstuhl für Allgemeine Betriebswirtschafts-lehre, insb. Wirtschaftsinformatik |
| 03/2010 – 09/2011 | **Towarzystwo Zarządzające SKOK Sp. z o. o. S. K. A., Gdynia, Polen**<br>Financial Controller, Business Analyst |
| 10/2006 – 09/2009 | **Europa-Universität Viadrina in Frankfurt (Oder)**<br>Studentische Hilfskraft am Lehrstuhl für Allgemeine Betriebswirtschafts-lehre, insb. Wirtschaftsinformatik |