Ein Modell zum zentralen Betrieb von hoch flexiblen SOA-Lösungen auf Basis definierter Standards

DISSERTATION

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von Dipl. Math. Liane Will

geb. am 04.10.1963 in Berlin

Gutachterinnen/Gutachter

Prof. Dr. Gunter Saake
Prof. Dr. Klaus Turowski
Prof. Dr. Robert U. Franz

Magdeburg, den 04.09.2014

Zusammenfassung

Design und Implementierung einer neuen Applikation gemäß den Anforderungen der potentiellen Anwender wird viel Zeit und Aufwand eingeräumt. Es geht darum, schneller und aufwandsarm, aber flexibel eine Applikation zu realisieren. Aber wird in diesen Phasen des Lebenszyklus bereits der spätere, permanente und unterbrechungsfreie, langfristige Betrieb dieser Applikation mit möglichst geringem Aufwand betrachtet und vorbereitet? Applikationen auf Basis einer service-orientierteren Architektur unterstützen die Flexibilität, Services bei Bedarf auszutauschen, um Fehler zu beheben oder sich neuen Anforderungen anzupassen. Daraus ergeben sich jedoch neue Anforderungen an den täglichen Betrieb im Rahmen der IT Service und Support Prozesse. In dieser Arbeit werden diese Anforderungen und die sich daraus ergebenen notwendigen Anpassungen untersucht. Als Grundlage der Untersuchung dient die IT Infrastructure Library, die in genereller Form die IT Service und Supportprozesse und den Betrieb von IT Applikationen beschreibt. Es wird gezeigt, welche Anforderungen sich im Betrieb durch die Nutzung der Flexibilität SOA-basierter Applikationen ergeben, soll die Flexibilität tatsächlich genutzt werden, was in der Praxis heute kaum der Fall ist. Durch den höheren Grad der Verteilung und der Granularität auf Services besteht auf der anderen Seite der Bedarf der Zentralisierung des Betriebs und der Vereinheitlichung unabhängig von der genutzten Technologie und konkreten Implementierung eines Services. Es wird gezeigt, wie die in SOA entstehenden bzw. entstandenen neuen Herausforderungen an den Betrieb mittels eines zentralen Betriebscockpits (COCo) gelöst werden können. Grundlage von COCo ist die Vereinheitlichung typischer Aufgaben und die Ausführung der Tätigkeiten als auch des Monitoring, d.h. der Überwachung der Applikationsprozesse und der daran beteiligten Services. Gleichzeitig erfordert dies die Formulierung von Nicht-funktionalen Anforderungen an die Services zur standardisierten Unterstützung des Betriebs des einzelnen Services und der darauf basierenden Applikationen. Bereits zur Design-Phase von Services müssen diese Anforderungen betrachtet und im Weiteren entsprechend implementiert werden. Auf diese Weise wird der Betrieb mit COCo weitestgehend unabhängig von der zur Laufzeit konkreten Service-Zusammensetzung einer Lösung. Der Austausch von Services beispielsweise erfordert damit keine wesentliche Anpassung der Betriebsaufgaben oder gar den Wechsel der Betriebswerkzeuge. Dies ist eine Grundvoraussetzung für die Nutzung der Flexibilität von SOA-basierten Lösungen. Diese Arbeit liefert eine Spezifikation des COCo und der notwendigen nicht-funktionalen Eigenschaften von Services sowie Service Repository und Service Bus. Es wird nachgewiesen, dass COCo ein ebenso fester Bestandteil von SOA sein muss wie das Service Repository oder der Service Bus.

Danksagung

Es hat länger gedauert als geplant und verlief auch anders als gedacht. Manchmal ist es besser, vorher nicht alle Hürden zu kennen, die einem bevorstehen. In lustiger Runde mit den UCC Mitarbeitern der Universität Magdeburg André Faustmann, André Siegling und René Zimmermann sowie meinem Kollegen Thomas Habersack ließ ich mich zu dem Vorhaben motivieren. Es gab Momente, da habe ich es bedauert. Aber heute sage ich danke!

Leider kann Claus Rautenstrauch den Abschluss dieses Projekts nicht mehr miterleben. Er war es, der den Grundstein dafür legte. Ich möchte Gunter Saake danken, dass er mir sein Vertrauen entgegen gebracht hat und mich als Doktorvater begleitet hat. Dank der Fürsorge von Veit Köppen und seiner Unterstützung habe ich das Projekt nun auch erfolgreich zum Abschluss gebracht. Ohne ihn hätte ich das definitiv nicht geschafft.

Ich möchte meinem Arbeitgeber, der SAP AG, und meinen Kollegen danken, dass ich die nötigen Freiräume bekam, diese Dissertation zu verwirklichen. Es war auch für sie ein eher ungewöhnliches Vorhaben einer gestandenen Mitarbeiterin.

Meine Familie und insbesondere meinem Mann danke ich für die Unterstützung auch in mentaler Hinsicht. Ich würde ja versprechen, nie wieder so ein gewaltiges Projekt in Angriff zu nehmen, aber über die Jahre bin ich doch immer wieder schwach geworden, wenn das Projekt nur interessant genug war. Die gute Nachricht ist, im Moment ist nichts geplant!

Nicht zuletzt danke ich all denen, die an mich geglaubt haben, auch wenn es mal wieder etwas länger gedauert hat.

Inhaltsverzeichnis

Verzeichnis der Abbildungen

Verzeichnis der Tabellen

Glossar

Abkürzung	Bedeutung
ARIS	Architektur integrierter Informationssysteme
CFIA	Component Failure Impact Analysis
CI	Konfigurationselement
CMDB	Konfigurationsmanagement Datenbank
CMS	Konfigurationsmanagement System
COCo	Zentrales Betriebscockpit
CTS	Change and Transport System der SAP AG
EPK	Ereignis-gesteuerte Prozesskette
ERP	Enterprise Resource Planning
GUI	Grafisches Benutzerinterface
ITIL	IT Infrastructure Library
ITSM	IT Service Management
KPI	Key Performance Indicator
OASIS	Organization for the Advancement of Structured Information Standards
PPS	Packets per Second
QTP	HP QuickText Professional
RfC	Request for Change
SB	Service Bus
SFIA	Service Failure Impact Analysis
SD	Sales and Distribution
SLA	Service Level Agreement
SLM	Service Level Management
SOA	Service-orientierte Architektur
SPOC	Single Point of Contact
SR	Service Repository
SRM	Software Release Manager
TPM	Transactions per Minute
TPS	Transactions per Second
UDDI	Universal Description Discovery and Integration

1 Motivation

Im Rahmen der Veröffentlichungen von Forschungsergebnissen der Gartner Inc. wurden 1996 erstmalig serviceorientierte Architekturen von Schulte und Natis beschrieben. Seit dem wird die Idee der stärkeren Modularisierung von Software-Lösungen zunehmend stärker verfolgt. Nur langsam entwickeln sich die bis dahin präferierten Client-Server-basierten Architekturen zu serviceorientierten Architekturen (SOA). Erst als sich die Vorteile in der Wiederverwendbarkeit von Services und der damit einhergehenden Vereinfachung des Designs und die Erhöhung der Flexibilität stärker herauskristallisieren, wie z.B. bei Krafzig et al. (2007) beschrieben, setzt ein wahrer Hype ein. Softwarehersteller richten ihre Produkte dahingehend aus, wie von Sprott (2005) oder der Software AG (2014) beschrieben und betonen die Vorteile ihres Ansatzes. Allerdings interpretiert auch jeder marktübliche Software-Hersteller SOA auf seine individuelle Weise, wie bei Liebhart (2007) dargestellt. Erst nach und nach entwickelten und entwickeln sich Prinzipien der SOA.

1.1 Problemstellung

Obwohl SOA bereits 1996 beschrieben wurde, existiert bis heute keine eindeutige Definition des Begriffs. Die bisher verwendeten Definitionen von SOA basieren entweder auf deren Bestandteilen oder beschreiben SOA über die Eigenschaften. Eine wesentliche Rolle dabei spielt die Organization for the Advancement of Structured Information Standards (Oasis), die an allgemeingültigen Definitionen und Standards arbeitet. In dieser Arbeit werden die verschiedenen Ansätze der Definition von SOA aufgegriffen. Naturgemäß ist die erste Herausforderung das Design und die Einführung von SOA-basierten Anwendungen. Dementsprechend gibt es dafür zahlreiche Untersuchungen und Darstellungen, wie bei Krafzig et al. (2007), Erl (2009) oder Trkman et al. (2011), die auch die Überlegenheit von SOA gegenüber den Client-Server-basierten Architekturen untermauern. Wichtige Vor- und Nachteile zeigt Tabelle 1.1. Dabei stehen die Vor- und Nachteile im Mittelpunkt, die für den Benutzer und Betreiber einer Lösung von Bedeutung sind.

Eigenschaft	Client-Server-Architektur	SOA
Modularität	Systemorientiert	Serviceorientiert
Erweiterbarkeit	Beschränkt durch relative Abgeschlossenheit der Systeme	beliebig
Anpassungs-fähigkeit	Beschränkt, da Systeme in sich geschlossen	Durch Serviceaustausch beliebig
Vergleichbarkeit	Auf Systemebene	Auf Serviceebene
Plattformunabhängigkeit	Systeme sind Technologie gebunden	Modularität auf Serviceebene reduziert Plattformabhängigkeit auf ein Minimum
Herstellerunabhängigkeit	Durch Herstellerabhängigkeit der Systeme sind Lösungen auch herstellerabhängig	Durch definierten Serviceumfang und Schnittstellen sind Services verschiedener Hersteller mit gleichen Parametern austauschbar
Austauschbarkeit technischer Komponenten	Kaum möglich	Da die Services autark sind, können sie untereinander mit gleichem funktionalen Umfang und Schnittstellen getauscht werden
Stabilität im Betrieb	Durch Stabilität des Lösungsaufbaus kaum Änderungen im Betrieb und dadurch stabiler Betrieb	Austausch von Services führt zu sich ändernden Betriebsbedingungen
Werkzeuge für den Betrieb	Herstellerabhängige und für das Produkt und die Architektur optimierte Werkzeuge vorhanden	Bisher keine Werkzeuge, die auf die Flexibilität einer SOA-Lösung automatisch reagieren können

Tabelle 1.1: Schwächen und Stärken der Architekturen

Durch den modularen Aufbau SOA-basierter Lösungen auf Basis von Services können diese Lösungen durch den gezielten Austausch gleichartiger Services angepasst oder erweitert werden, wie in den nachfolgenden Kapiteln dargestellt wird. Während die Lösung aus Geschäftsprozesssicht

dadurch kontrolliert angepasst werden kann, hat dies jedoch erheblichen Einfluss auf die Anforderungen im Betrieb der Lösung. Die durch die Austauschbarkeit von Services entstehende Flexibilität SOA-basierter Lösungen wird durch die bisher unzureichende Betreibbarkeit eingeschränkt. Die sich aus der Architektur ergebenen Konsequenzen für den langfristigen, täglichen Betrieb solcher SOA-basierten Lösungen stehen bisher nicht im Blickfeld der Forschung. Tatsächlich wirkt sich SOA aber auf den Betrieb aus, wie am nachfolgenden Beispiel gezeigt wird.

Einer der Vorteile von SOA aus Sicht des Benutzers derartiger Lösungen ist die Tatsache, dass der technische Aufbau einer Lösung vor dem Benutzer verborgen wird und für den Benutzer irrelevant ist. Benutzer greifen über ein einheitliches Benutzer-Interface auf die Lösung zu. Welche Services dabei in Anspruch genommen werden, ist für den Benutzer unerheblich. Aus Betriebssicht erschwert dies jedoch die Fehleranalyse. Um dies zu untermauern, betrachte man folgendes kleines Beispiel auf Basis von SAP Software. Abbildung 1.1 zeigt den Abbruch einer Aktion eines Benutzers, die im SAP Portal ausgeführt wurde. Der Benutzer wollte Urlaub beantragen. In diesem Fall wird dazu der Internet Explorer genutzt, in dem über das SAP Portal auf eine Anwendung in einem SAP Personalverwaltungssystem zugegriffen wird. Diese technischen Details bleiben dem Benutzer jedoch verborgen und sind auch nicht von Bedeutung für seine Tätigkeit. Als Folge des Abbruchs seiner Aktion erhält der Benutzer die in Abbildung 1.1 dargestellte Abbruchmeldung. Üblicherweise öffnet der Benutzer nun eine Störfallmeldung auf Basis der ihm zur Verfügung stehenden Informationen. Der Benutzer kann lediglich seine Aktionen beschreiben. Die ihm bekannten technischen Informationen sind eher gering. Daher muss die Störfallanalyse im ersten Schritt den eigentlich verursachenden, technischen Service identifizieren, der den Abbruch auslöste. In der Praxis wird beobachtet, das dafür auf die von Client-Server-Architekturen bekannten Methoden und Werkzeuge zurückgegriffen wird. Daraus resultiert, dass die an der Benutzeraktivität beteiligten technischen Komponenten schrittweise nach einem Problem untersucht werden, das den Abbruch, wie in Abbildung 1.1 dargestellt, verursacht haben könnte. Es wird also nicht zielgerichtet nach der Ursache geforscht, sondern nach dem Ausschlussverfahren vorgegangen. In der Störfallbearbeitung schlägt sich dies in auffällig häufigen Bearbeiterwechseln nieder: Der SAP Portal-Experte analysiert pauschal das SAP Portal und findet keinen Störfall. Er übergibt an den SAP-Personalverwaltungssystem-Experten. Dieser untersucht pauschal das System, ob es Ereignisse gab, die im Zusammenhang mit dem Abbruch aus Abbildung 1.1 stehen könnten. Falls er nichts findet, wird er an den nächsten Experten einer beteiligten technischen Komponente übergeben oder zurück an den SAP Portal-Experten, der noch einmal genauer analysieren soll. Findet er einen Störfall

muss der Experte prüfen, der in Abbildung 1.1 dargestellte Störfall mit dem im System entdeckten Störfall in Zusammenhang steht. Diese Arbeitsweise wird in SOA umso langwieriger je mehr Services auf Basis unterschiedlicher Technologien verwendet werden und Benutzer von technischem Wissen befreit werden.

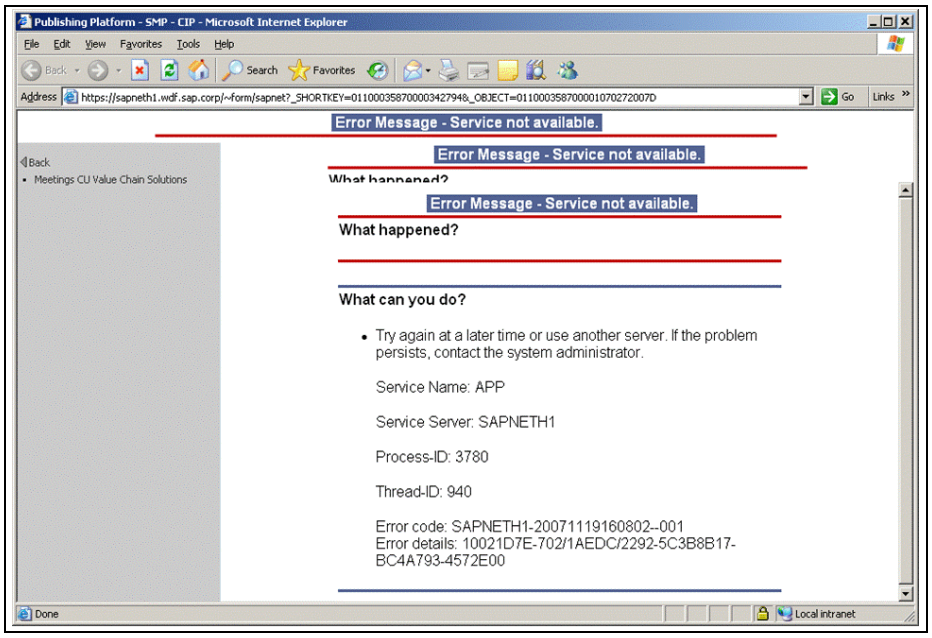

Abbildung 1.1: Abbruch einer Aktion im SAP Portal

Diese beschriebene Analyse ist bei Client-Server-basierten Lösungen wesentlich einfacher, da diese systemorientiert aufgebaut sind und der Benutzer z.B. das verursachende technische System bereits benennen kann. Da sich die technische Service-Zusammensetzung in SOA-basierten Lösungen zudem leichter ändern lässt, kommt dem beschriebenen Analyseschritt zur Ermittlung des verursachenden Services eine erhöhte Bedeutung zu.

Ausgehend von der Untersuchung der Auswirkungen bei der Nutzung der Flexibilität SOA-basierter Lösungen wird in dieser Arbeit untersucht, wie sich die Nutzung der potentiellen Flexibilität auf den Betrieb derartiger SOA-basierten Lösungen auswirkt und welche Anforderungen sich ergeben. In dieser Arbeit wird gezeigt, welche Eigenschaften ein Werkzeug zum Betrieb aufweisen muss, um diesen Anforderungen gerecht zu werden und der potentiellen Flexibilität SOA-basierter Lösungen nicht entgegen zu stehen, sondern sie optimal zu unterstützen. Dies wird durch die Evaluierung und den vergleich typischer Störfälle während des Betriebs einer Client-Server-basierten und SOA-basierten Lösung nachgewiesen.

1.2 Beitrag der Dissertation

Die vorliegende Untersuchung fokussiert sich auf den Betrieb einer SOA-basierten Lösung. Da es für den Begriff *Betrieb einer Lösung* keine allgemeingültige Definition gibt, wird in dieser Arbeit zunächst bestimmt, was im Einzelnen zu den täglichen Aufgaben im Betrieb gehört. Dazu wird auf den Rahmen der IT Infrastructure Library (ITIL) des Office of Governance Commerce (2007) zurückgegriffen. ITIL beschreibt unabhängig von Herstellern und Architektur in der Praxis bewährte Prozesse und Funktionen für das IT Service und Support Management entlang des Lebenszyklus von Dienstleistungen in Bezug auf den Applikations Management Lebenszyklus. Die ersten ITIL-Bücher wurden bereits 1989 vom Office of Governance Commerce veröffentlicht, so dass die Wurzeln auf der Basis des damals verfügbaren Wissens über IT Service und Support Management für Client-Server-Architekturen gelegt wurden. Durch die konsequente Reduzierung der Prozesse auf den herstellerunabhängigen Kern und die fortlaufende Überarbeitung sind die im ITIL beschriebenen und bewährten Vorgehensweisen weitestgehend allgemeingültig und architekturunabhängig, doch müssen eben diese Beschreibungen bei der praktischen Umsetzung architekturspezifisch und lösungsabhängig erweitert werden. Dies Arbeit fasst die Unterschiede in Client-Server- und SOA-basierten Lösungen in Bezug auf den Betrieb zusammen und leitet daraus die Notwendigkeit für die Anpassung des Betriebs und eines dementsprechenden Werkzeugs ab.

Für die Untersuchungen in dieser Arbeit wird ein einfacher Lebenszyklus bestehend aus der Planung einer Lösung, dem Aufbau oder der Entwicklung einer Lösung, dem Betrieb der Lösung und der Optimierung herangezogen. Dabei existieren auch für den Begriff des Lebenszyklus unterschiedliche Definitionen hinsichtlich des betrachteten Objekts wie Software, Service oder Applikation. Ebenso werden die Phasen des Lebenszyklus unterschiedlich benannt und auch unterteilt, siehe beispielsweise [Köhler, 2005] oder [Zarnekow et al., 2004]. Für die Untersuchungen in dieser Arbeit genügt es, den in Abbildung 1.2 dargestellten Lebenszyklus als Grundlage zu nutzen. Aus Sicht des Applikationsmanagement Lebenszyklus fokussieren sich die Untersuchungen in dieser Arbeit auf die Phase des technischen Betriebs von IT-Lösungen. Es wird erstmalig zusammenhängend dargestellt, welche Anforderungen sich aus den IT-Service und Support Prozessen ergeben, wenn die für SOA angenommenen Eigenschaften und Möglichkeiten konsequent genutzt werden sollen.

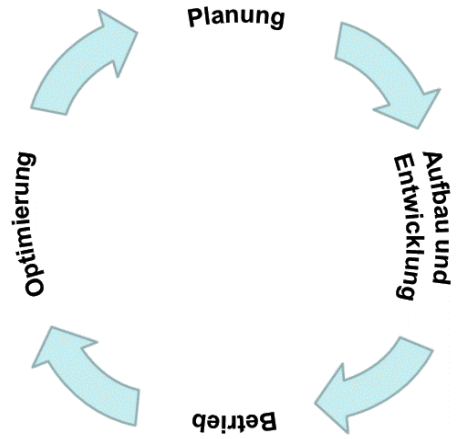

Abbildung 1.2: Vereinfachter Applikationsmanagement Lebenszyklus

Es wird untersucht, welche Konsequenzen es für den Betrieb hat, wenn tatsächlich während der Laufzeit Services ausgetauscht oder erweitert werden. Da es auch für die von einem Service verwendete Technologie keine Einschränkungen gibt, kann z.B. der Austausch eines Services auch zu technologischen Konsequenzen führen. Bis heute wird die von SOA unterstützte und beschriebene Flexibilität auf Basis der Service-Unabhängigkeit und –Austauschbarkeit nur sehr eingeschränkt genutzt. Die vorliegenden Untersuchungen liefern die Ursachen dafür.

Der Betrieb einer SOA-Lösung unter Ausnutzung des Flexibilitätspotentials erfordert die Erweiterung der bisher gängigen Definitionen von SOA. In dieser Arbeit wird gezeigt, dass die Vorteile von SOA erst dann ausgeschöpft werden können, wenn die in Client-Server-Architekturen systemorientierte Administration weiter entwickelt wird. Die bisherige technische und auf einzelne Systeme fokussierte Administration genügt nicht den Anforderungen, die sich durch die konsequente Nutzung der Vorteile von SOA-basierten Lösungen ergeben. Es können Services ausgetauscht oder eine Lösung an veränderte Anforderungen angepasst werden. Eine rein systemorientierte Administration kann auf derartige Änderungen nicht schnell genug reagieren. Es besteht die Notwendigkeit, Administration und Betrieb so zu entwickeln, dass die Prozesse in ihrer Gesamtheit erfasst und darauf ausgerichtet werden. Bisher im Einsatz befindliche Werkzeuge und ebenso das von Client-Server-Architekturen erworbene Verständnis müssen sich grundlegend ändern. In der Folge wird das bisherige Verständnis der zwangsweise erforderlichen Bausteine einer SOA in dieser Arbeit erweitert. Es wird nachgewiesen, dass ein zentraler Betriebsservice ein ebenso fester Bestandteil einer SOA sein muss, wie Service Bus oder Repository. In dieser Arbeit wird ein entsprechendes Werkzeug eingeführt, seine Leistungen und

Qualitätskriterien beschrieben. Ein zentraler Betriebsservice, hier Central Operations Cockpit (COCo) genannt, ist in der Lage die Aufgaben des täglichen Betriebs zu zentralisieren und mittels Standardisierung weitestgehend unabhängig vom technischen Aufbau und Zusammensetzung der Services in einer Lösung zu gestalten. Dazu wird an dieser Stelle beschrieben, welche Tätigkeiten standardisiert werden und wie Services von sich aus COCo automatisch unterstützen müssen ohne manuell vorzunehmende Vorbereitungen und Einstellungen durch den Betreiber, ähnlich wie es bereits jetzt für Service Bus oder Service Repository erfolgt. Das Vorgehen dazu und der Aufbau dieser Arbeit werden im nächsten Abschnitt in dieser Arbeit dargestellt.

1.3 Übersicht über die Arbeit

Zur Einführung in die nachfolgenden Untersuchungen werden die verwendeten Voraussetzungen, Begriffe und Grundlagen beschrieben. Dazu gehören die Darstellung des aktuellen Entwicklungstands der Definition von SOA sowie die Klärung des Begriffs Flexibilität. Diese Arbeit stellt die verschiedenen Ansätze zur Definition von SOA vor und weist die bisherigen Schwächen bzw. die Unvollständigkeit der Definitionen nach. Im nächsten Schritt wird untersucht, was die Flexibilität einer SOA-basierten Lösung bedeutet und welche Anforderungen sich daraus für deren Betrieb ergeben. Dies unterstreicht und zeigt das Beispiel eines vereinfachten Kundenauftragsprozesses, das durchgängig in dieser Arbeit zur Darstellung und Verifizierung der Ergebnisse verwendet wird. Anhand der Gegenüberstellung des Client-Server-Ansatzes und SOA an diesem Beispiel werden im Kapitel 2 die neuen Herausforderungen, sowie die Unzulänglichkeiten der Ansätze für den Betrieb herausgearbeitet.

In Kapitel 3 spezifiziert den *Betrieb einer Lösung* im Detail. Für den Betrieb von Lösungen oder auch Software gibt es keine allgemeingültige Definition. Daher wird in dieser Arbeit die IT Infrastructure Library (ITIL) der [Office of Governance Commerce, 2007] genutzt, um den Inhalt des Begriffs *Betrieb einer Lösung* einzugrenzen und erstmalig hinsichtlich der Client-Server-Architektur und SOA zu spezifizieren. In der Arbeit wird ITIL strikt in der deutschen Übersetzung verwendet. ITIL kann auf Grund der Abstraktionsebene auch für das IT Service und Support Management von SOA-basierten Lösungen eingesetzt werden. ITIL beschreibt jedoch nur die Prozesse, Rollen und Qualitätskriterien der verschiedenen IT Service und Support Prozesse im Lebenszyklus einer Lösung. Wie dies im Detail zu realisieren ist, beschreibt ITIL dagegen nicht, so dass man für die praktische Umsetzung vergeblich nach bewährten Richtlinien in ITIL sucht. Daher untersucht diese Arbeit, wie die von ITIL konzeptionell dargestellten

Prozesse und Tätigkeiten in einer SOA-Umgebung unter klarer Fokussierung auf den Betrieb umzusetzen sind.

Kapitel 4 weist anhand des eingeführten Beispiels des Kundenauftragsprozesses und verschiedener Fallstudien nach, dass die Übertragung bekannter Betriebskonzepte von Client-Server-Architekturen auf den Betrieb von SOA zu Instabilität und erhöhten Aufwänden im Betrieb einer Lösung führen und daher nicht praktikabel ist. Dazu werden in dieser Arbeit die neuen Herausforderungen und Unzulänglichkeiten anhand der spezifizierten Prozesse und Funktionen im Betrieb dargestellt. Es wird erstmalig analysiert, inwiefern existierende Administrationswerkzeuge den Anforderungen gerecht werden bzw. welche Schwächen sie in Bezug auf die zuvor beschriebenen Betriebsanforderungen von SOA haben. In der vorliegenden Untersuchung wird zur Lösung dieser Probleme ein sogenannter zentraler Betriebsservice (COCo) vorgeschlagen und beschrieben. Die Arbeit zeigt, wie der Betrieb von SOA mit Hilfe von COCo die Nutzung des Flexibilitätspotentials von SOA unterstützt. In der Schlussfolgerung wird COCo für den Betrieb SOA-basierter Lösungen als ein weiterer, zentraler und fester Baustein innerhalb der Architektur gefordert, der ebenso zwingend erforderlich ist in SOA, wie die Bestandteile Service Bus oder Service Repository.

Davon ausgehend liefert Kapitel 5 eine detaillierte Leistungsbeschreibung und Qualitätskriterien, welche COCo zum Betrieb einer SOA-basierenden Lösung erfüllen muss. In der Arbeit wird die Arbeitsweise des COCo dargestellt, um die erforderlichen Qualitätskriterien zu erfüllen und die Methoden für den Betrieb der Services hergeleitet, die eine Lösung bilden. Neben den funktionalen Anforderungen muss jeder Service weiteren, nicht-funktionalen Standards folgen, die die für die Lösungsadministration erforderlichen Informationen bereitstellen. Dem Trend der immer größeren Verteilung einer Lösung mittels Services steht der Bedarf der zentralen Integration und Administration einer solchen Lösung gegenüber. Ähnlich einem Puzzle fügen sich die administrativen Informationen an zentraler Stelle zu einem zentralen Betriebsservice für die spezifischen Belange des Lösungsbetriebs, wie auch IT Service und Supportprozesse [Office of Governance Commerce, 2007] zusammen, was durch die Untersuchungen in dieser Arbeit beschrieben wird. Der hier dargestellte zentrale Betriebsservice muss Adapter zur Verfügung stellen, die es gestatten, die von den Services einer Lösung gelieferten administrativen Informationen und Funktionen einzuklinken. Kapitel 5 liefert diese Beschreibungen.

Anhand des durchgängig verwendeten Beispiels eines Kundenauftragsprozesses wird in Kapitel 6 demonstriert und evaluiert, dass COCo die zuvor beschriebenen Probleme löst und einen erheblichen Beitrag zur Nutzbarkeit der Flexibilität von SOA leistet. Die daraus resultierenden

Schlussfolgerungen werden in Kapitel 7 zusammengefasst. Neben der technischen Machbarkeit von COCo spielen auch unternehmerische Aspekte eine wesentliche Rolle, die der Realisierung von COCo entgegen stehen können und an dieser Stelle diskutiert werden.

2 Grundbegriffe und Grundlagen

Die nachfolgenden Untersuchungen bauen auf einer Reihe von grundlegenden Begriffen auf. Im Folgenden werden die für diese Arbeit wichtigen Grundbegriffe und Grundlagen eingeführt, um ein gemeinsames Verständnis zu schaffen.

2.1 Der Begriff Lösung

Bei den nachfolgenden Untersuchungen wird der Begriff *Lösung* im Sinne verwendet von:

Definition 2.1 Lösung

Eine Lösung ist die Realisierung betriebswirtschaftlicher Anforderungen, Problemstellungen und Prozesse mit Hilfe von IT-Mitteln. Eine Lösung umfasst alle software- und hardwaretechnischen Bestandteile, die zur Umsetzung der Anforderungen mit IT-Mitteln notwendig sind.

Eine Lösung sei eine Menge von Benutzerszenarien. Diese Szenarien sind bestimmten Benutzeranforderungen zugeordnet. Aus technischer Sicht umfasst eine Lösung alle technischen Komponenten, wie Services oder Benutzeroberflächen, die zur Befriedigung der Benutzeranforderungen erforderlich sind. Dabei ist es unwesentlich, welche Hardware oder Software oder Architektur im Einzelnen zum Einsatz kommen. Wesentlich ist, dass die im Fokus stehenden Benutzeranforderungen durch eine Lösung abgedeckt werden.

Bei einer SOA-basierten Lösung folgen die zum Einsatz kommenden Komponenten demnach den Prinzipien von SOA. Client-Server-basierte Lösungen sind hinsichtlich der technischen Architektur als Client Server aufgebaut.

2.2 Serviceorientierte Architektur

Anders als beim Übergang zu den Client-Server-Architekturen ist die Entwicklung zu SOA evolutionär. Entsprechend entwickelt sich die Definition der SOA, wobei zwei wesentliche Ansätze verfolgt werden. In [Melzer und Eberhard, 2008] wird SOA über die Eigenschaften definiert.

Definition 2.2 SOA (auf Basis der Eigenschaften nach [Melzer und Eberhard, 2008])

Unter einer SOA versteht man eine Systemarchitektur, die vielfältige, verschiedene und eventuell inkompatible Methoden oder Applikationen als wieder verwendbare und offen zugreifbare Dienste repräsentiert und dadurch eine plattform- und sprachenunabhängige Nutzung und Wiederverwertung ermöglicht.

Diese Definition ist offen und konzentriert sich zur Bestimmung von SOA auf deren Eigenschaften als auch auf die Eigenschaften der Services. Dagegen wird z.B. in [Krafzig et al., 2007] SOA durch vier charakteristische Bestandteile beschrieben. „A Service-Oriented Architecture (SOA) is a software architecture that is based on the key concepts of an application frontend, service, service repository, and service bus. A service consists of a contract, one or more interfaces, and an implementation."

Definition 2.3 SOA (auf Basis der Bestandteile gemäß in [Krafzig et al., 2007])

Eine serviceorientierte Architektur (SOA) ist eine Software-Architektur, die auf den Schlüsselkonzepten eines Applikations-Zugang, Service, Service Repository und Service Bus besteht. Ein Service besteht aus einem Kontrakt, einem oder mehreren Schnittstellen und seiner Implementierung.

[Krafzig et al., 2007] beschreibt SOA durch deren Bestandteile, die erforderlich sind, um Eigenschaften leisten zu können, wie sie von [Melzer und Eberhard, 2008] charakterisiert werden. Auch [Goyal, 2009] oder [Erl, 2005] gehen von diesen Bestandteilen aus. [Melzer und Eberhard, 2008] verzichtet in der Definition auf die Spezifikation der Umsetzung. Es bleibt offen, wie man SOA technisch realisiert. [Krafzig et al., 2007] versucht dagegen aus technischer Sicht die Bausteine zu beschreiben, um SOA und die in [Melzer und Eberhard, 2008] bezeichneten Eigenschaften zu erzielen. Beide Ansätze zur Definitionen sind anwendbar und widersprechen sich nicht. In dieser Arbeit soll der Ansatz der Definition von SOA über deren Eigenschaften, also [Melzer und Eberhard, 2008] genutzt werden. Es soll gezeigt werden, dass bei konsequenter Verfolgung und Umsetzung der Eigenschaften von SOA das bisherige Verständnis zu den Bestandteilen von SOA und somit z.B. die Definition von [Krafzig et al., 2007] um einen weiteren Baustein zum Betrieb einer SOA-basierenden Lösung ergänzt werden muss. Zunächst werden die von [Krafzig et al., 2007] geforderten Bausteine kurz beschrieben.

Einig ist man sich in dem zentralen, grundlegenden Element: dem Service (oder Dienst). Die Organization for the Advancement of Structured Information Standards (OASIS) definiert den Begriff **Service**, wie folgt

[Oasis, 2011]: “A service is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description.”

Definition 2.4 Service [Oasis, 2011]

Ein Service ist ein Mechanismus, um den Zugang zu einer oder mehreren Funktionen zu erlangen, wobei diese mittels vorgeschriebener Schnittstellen ausgeübt werden und die Ausübung konsistent in den Bedingungen und Grundsätzen gemäß der Service-Beschreibung erfolgt.

Demnach weist ein Service Schnittstellen auf, die der Kommunikation und dem Datenaustausch zwischen anderen Services dienen. Die Service Beschreibung oder auch Service Kontrakt beinhalten die Metadaten eines Service, wie Namen, Version, Funktionsbeschreibung etc. die der Unterstützung des Service Repositorys dienen. Die eigentliche Implementierung des Services, also die Business Logik sowie das Datenmanagement bilden den dritten Bestandteil eines Service, siehe Abbildung 2.1.

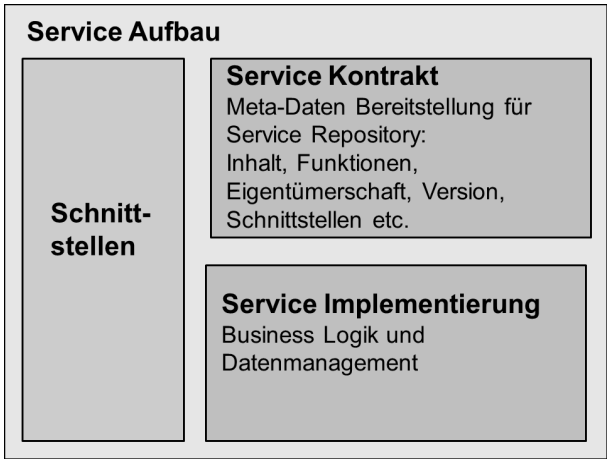

Abbildung 2.1: Bausteine eines Services

Der Umfang der Service Implementierung wird als **Granularität** [Melzer und Eberhard, 2008] bezeichnet. Es gibt keine zwingende Granularität von Services, aus denen ein Prozess zusammengesetzt wird. Er besteht aus einer Abfolge von verknüpften Aktivitäten, die in Abhängigkeit von Zwischenergebnissen auch verzweigen oder parallel erfolgen können. Wie groß der Anteil des einzelnen Service an dem gesamten Business Prozess ist, wird beim Entwurf der SOA-basierten Lösung festgelegt. Werden Services zu fein gewählt, erhält man eine hohe Anzahl von Services, was die

Verwaltung und Übersicht erschwert. Wird ein Service dagegen zu umfassend gefasst, erschwert dies die Austauschbarkeit. Die Granularität ist so zu wählen, dass der fachliche Zusammenhang zu den Teilaktivitäten eines Geschäftsprozesses erkennbar ist. Um Services austauschen zu können, müssen Informationen über Services verfügbar sein. Die Service-Komposition in einer SOA-basierten Lösung wird als **Orchestrierung** bezeichnet, z.B. [Erl, 2009].

Das *Service Repository* umfasst alle diese Informationen, die benötigt werden, um Services entsprechend den geänderten Anforderungen auszuwählen oder durch gleichwertige Services auszutauschen. Als quasi Standard hat sich die von OASIS spezifizierte *Universal Description Discovery and Integration* (UDDI) [Oasis, 2004] als Beschreibungssprache etabliert, da sie plattformunabhängig und erweiterbar ist. Tatsächlich stellt auch das Service Repository einen Service dar. Es verwaltet insbesondere, die von den Services bereitgestellten Metadaten.

Der *Service Bus*, der selbst wiederum als Service betrachtet werden kann, dient dem Datenaustausch zwischen Services. Er basiert auf den Interfaces, die die einzelnen Services anbieten. [Papazoglou und Heuvel, 2007] beschreiben den Service Bus als den integrierenden Bestandteil in der SOA-Landschft: „[…] pulls together application and discrete integration components to create assemblies of services to form composite business processes […].“ Gegenüber dem direkten Datenaustausch zwischen Services hat der Service Bus den Vorteil, dass er heterogene Technologien in Schnittstellen und Kommunikation integrieren kann. Daten können servicespezifisch und zentralisiert im Service Bus aufbereitet und verteilt werden. Dadurch kann eine Mehrfachbearbeitung gleicher Daten reduziert werden. Nicht zuletzt dient das vereinheitlichte *Graphical User Interface* (GUI) als Benutzerzugang zu der SOA-Lösung. Komplexität und Technik der Realisierung bleiben dem Benutzer verborgen. Auf technische oder inhaltliche Erfordernisse kann durch den Austausch von Services reagiert werden, ohne dass diese Änderungen für den Benutzer in der Oberfläche oder Handhabung sichtbar werden.

Je nach der Bedeutung und der im Service enthaltenden Business Logik werden sie den Schichten (Layer) einer SOA zugeordnet, siehe Abbildung 2.2. Die verschiedenen Schichten dienen der Gruppierung der Services der zu einer SOA gehörenden Lösung hinsichtlich ihres Aufgabengebiets. Der Unternehmensschicht werden alle Services zugeordnet, die dem Benutzer den Zugriff auf die Lösung dienen. Der Prozessschicht werden alle die Services zugeordnet, die der direkten Umsetzung der einzelnen Prozessschritte dienen. Services, die der Unterstützung bzw. Vermittlung dienen, wie Schnittstellenadapter oder Gateways, werden der Zwischenschicht zugeordnet. Die Basisschicht schließlich umfasst die

Services, die der grundlegenden technischen Umsetzung dienen, wie dem Datenmanagement.

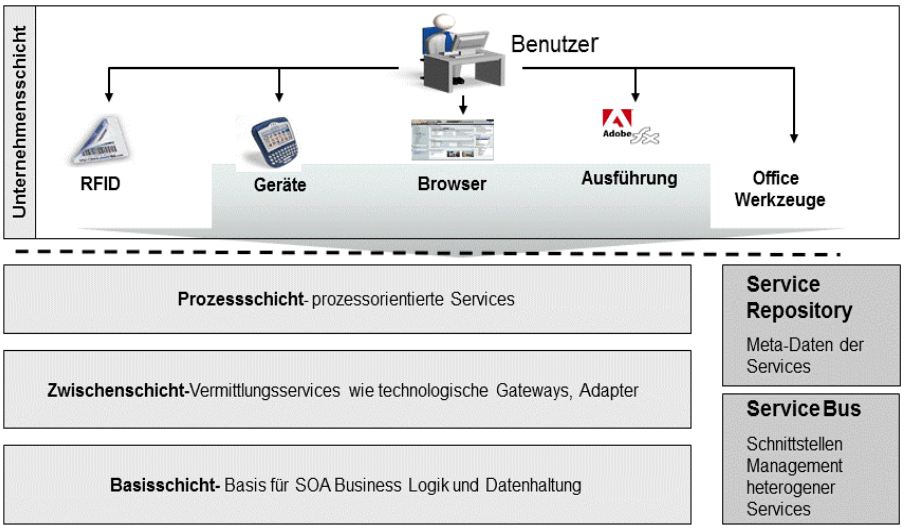

Abbildung 2.2: Aufbau SOA und Bausteine

Bei [Papazoglou, 2003] oder [Liebhart, 2007] finden sich zudem Ansätze, weitere Services zu unterscheiden. Es wurde zumindest erkannt, dass ein möglicher Dienstleister zum Betrieb einer solchen Lösung Nutzungsinformationen zwecks Verrechnung benötigt. Häufig wird zu diesem Zwecke zwischen dem Auftraggeber und –nehmer eine Service Leistungsvereinbarung (Service Level Agreement- SLA) abgeschlossen. [Liebhart, 2007] führt sogenannte gemanagte Services ein, die bei der Zertifizierung und Auswahl von einzusetzenden Services unterstützen sollen. „Jeder Dienst ist mit einem definierten SLA (Service Level Agreement) zu versehen, der einem Servicenehmer vertraglich eine garantierte Dienstverfügbarkeit zusichert. Diese Schicht umfasst also fast nur organisatorische Aufgaben.“ [Liebhart, 2007] S. 11. Sicherlich sind das für einen Betreiber wesentliche Aspekte, jedoch sind die Herausforderungen beim Betrieb einer Lösung weitaus komplexer und erfordern ein Umdenken im Aufbau einer SOA. Dies führt zu der Frage, was eigentlich der Betrieb von IT-Lösungen alles umfasst. Dies soll im nächsten Kapitel dargestellt werden.

2.3 Flexibilität

Durch die Möglichkeit, Services und die Zusammensetzung eines Prozesses aus Services zu verändern, ohne die Benutzer mit diesen Änderungen zu konfrontieren, kann die Flexibilität der Lösung maßgeblich erhöht werden. Für den Begriff *Flexibilität* gibt es jedoch keine einheitliche Definition. In [Eicker et al., 2007] werden verschiedene Begriffsdefinitionen von Flexibilität und ihren Dimensionen in der Literatur betrachtet und für die Flexibilität von Unternehmen zusammengeführt. [Gebauer und Schober, 2005] unterscheidet hinsichtlich Nutzungsflexibilität und Änderungsflexibilität eines Informationssystems. [Evans, 1991] hat die verschiedenen Begriffe und Verwendungen in Bezug auf Flexibilität in der englischsprachigen Literatur zusammengetragen, normiert und als Matrix zusammengestellt. Er unterscheidet drei Kriterien der Flexibilität:

- *Yielding to pressure*, als Fähigkeit auf externen Druck zu reagieren und sich anzupassen,
- *Capacity for new situation*, als die Menge oder den Freiraum von möglichen Anpassungen auf bevorstehende sich ändernde Bedingungen und Wünsche sowie
- *Suspectibility*, als Schwierigkeitsgrad, Modifikationen durchzuführen.

Die Flexibilität SOA-basierter Lösungen beruht auf deren Potential, Services während der Laufzeit auszutauschen oder anzupassen, um auf veränderte Anforderungen aus der Umwelt zu reagieren. Vorteile ergeben sich insbesondere für die Nutzer einer SOA durch schnellere Anpassungsfähigkeit auf Veränderungen und Wiederverwendbarkeit von Services.

Wenn man einen Service in einer komplexen SOA-basierten Lösung ändern oder ersetzen will, müssen die Eigenschaften der auszutauschenden Services betrachtet werden.

2.4 Eigenschaften von Services

Die Eigenschaften von Services sind in funktionale und nicht-funktionale zu unterteilen.

Definition 2.5: funktionale Eigenschaften eines Services

Funktionale Eigenschaften eines Services sind alle die Eigenschaften, die die gewünschten Funktionen einer Lösung bereitstellen.

Definition 2.6: Nicht-funktionale Eigenschaften eines Services

Nicht-funktionale Eigenschaften sind die Eigenschaften, die keinen direkten Beitrag zu den Funktionen einer Lösung leisten. Diese Eigenschaften kennzeichnen beispielsweise die Performance oder dienen dem Betrieb.

Der Austausch von Services erfolgt immer in Bezug auf die Lösung, in der die Services verwendet werden. Die Austauschbarkeit ist abhängig von den funktionalen Eigenschaften eines Services. Nicht-funktionale Eigenschaften werden dagegen außer Acht gelassen.

2.4.1 Gleichwertigkeit von Services hinsichtlich einer Lösung

Services, die gleiche funktionale Eigenschaften in Bezug auf eine Lösung aufweisen, können ausgetauscht werden, ohne die Funktion der Lösung zu verändern.

Definition 2.7 gleichwertige Services in Bezug auf eine Lösung

Services werden *gleichwertig in Bezug auf eine Lösung* genannt, wenn sie innerhalb einer Lösung ausgetauscht werden können, ohne die funktionalen Eigenschaften der Lösung zu verändern.

Die Gleichwertigkeit von Services bezieht sich immer auf eine konkrete Lösung. Services, die hinsichtlich einer Lösung gleichwertig sind, müssen hinsichtlich einer anderen Lösung nicht zwingend wiederum gleichwertig sein. In Bezug auf eine Lösung gleichwertige Services können sich dagegen hinsichtlich ihrer Technologie, Datenein- und -ausgabeformate oder anderen nicht-funktionalen Eigenschaften wie administrativen Eigenschaften unterscheiden. Ausschlaggebend ist lediglich der Erhalt des Prozessablaufs. Um den Ablauf zu gewährleisten, können Änderungen nur im Rahmen gleicher Eingangsdaten und einem gleichen Ausgangsdatenformat erfolgen. Die Geschäftsprozesslogik muss in jedem Fall gewahrt bleiben.

2.4.2 Ähnliche Services hinsichtlich einer Lösung

Denkbar ist dagegen auch, Services auszutauschen und damit bewusst eine Lösung im Rahmen bestimmter Vorgaben zu verändern. Dies kann einer geplanten Erweiterung oder Anpassung einer Lösung dienen. Dabei werden funktionale Eigenschaften durch den Austausch bewusst verändert. Derartige Services können sich wiederum hinsichtlich ihrer Technologie, Datenein- und -ausgabeformate oder anderen nicht-funktionalen Eigenschaften wie administrativen Eigenschaften unterscheiden.

Definition 2.8 ähnliche Services hinsichtlich einer Lösung

Services, die hinsichtlich einer Lösung austauschbar sind, um eine Lösung in ihren funktionalen Eigenschaften bewusst zu ändern, werden *ähnlich hinsichtlich einer Lösung* genannt.

Ein Service kann aber auch innerhalb weiterer, anderer Geschäftsprozesse bzw. Lösungen genutzt werden. Hier spielt die bereits erwähnte Granularität eines Services eine Rolle. Ist ein Service zu komplex bzw. leistet extrem unterschiedliche Aktivitäten, wird kaum ein gleichwertiger oder ähnlicher

Service für eine Lösung gefunden werden. Somit ist der Service faktisch nicht austauschbar. Projekterfahrungen zeigen, dass sich genau diese Situation heute häufig in der Praxis findet. Es stellt sich die Frage, inwieweit Softwarehersteller überhaupt interessiert sind, ihre Services austauschbar zu halten. Dadurch werden jedoch Rahmenbedingungen geschaffen, die die Flexibilität von SOA-basierten Lösungen ungerechtfertigt einschränken. Ebenso darf es z.B. für die verwendete Technologie eines Services keine Beschränkungen geben, d.h., ein Service in einer SOA-basierten Lösung kann durch einen anderen Service mit anderer Technologie ausgetauscht werden.

2.4.3 Einschränkung der zu betrachtenden Servicemenge

In dieser Arbeit wird angenommen, dass der Anwender einen Service selbst betreibt bzw. durch einen externen Dienstleister (Outsourcer) betreiben lässt. Im Zuge des Einzugs von Cloud-Technologien wird zunehmend diskutiert und auch genutzt, dass frei verfügbare Services aus dem Web in SOA-Lösungen integriert werden können, wie z.B. Suchmaschinen, Vergleichsplattformen oder Commerce Portale. Damit obliegt die Verantwortung für den Betrieb des einzelnen Services nicht mehr dem Anwender. Jedoch die Verantwortung für die Qualität des Betriebs der Geschäftsprozesse als Ganzes obliegt dem Anwender. In der Folge heißt das, dass auch Fremdservices in gewissem Maße in den kundeneigenen Betrieb eingebunden werden. Zwar entfallen Aufgaben, wie z.B. die Optimierung, Konfiguration oder Wartung des Fremdservice, aber andererseits muss z.B. die Gesamtperformance des eigenen Geschäftsprozesses übergreifend ermittelt und optimiert werden. Performancevereinbarungen sind Teil eines Service Leistungsvereinbarung (Service Level Agreement- SLA), deren Erfüllung regelmäßig nachgewiesen werden muss. Der Konflikt an dieser Stelle besteht in der möglicherweise sogar kostenfreien, aber auch unverbindlichen Verfügbarkeit und Leistungsfähigkeit eines solchen Services. Der Nutzer hat keine zugesicherten Garantien zur Verfügbarkeit. Derartige Fremdservices unterstützen bisher auch nicht die Integration in kundeneigene Betriebswerkzeuge. Vielmehr müssen Betriebswerkzeuge eigene, servicespezifische Schnittstellen bereitstellen, um die für solche Service notwendigen Betriebsaufgaben zu erfüllen, beispielsweise die Performancemessung für einen Geschäftsprozess unter Integration des Durchsatzes bei der Nutzung des Fremdservices. Bisher wird jedoch vernachlässigt, inwieweit sich Änderungen an Services, insbesondere wenn sie die verwendete Technologie oder auch Eigentümerschaft betreffen, auf typische Aufgaben des täglichen Betriebs auswirken. Wie können Änderungen an Services oder die Änderung der Service-Zusammensetzung innerhalb einer SOA-basierten Lösung wirklich umgesetzt werden? In dieser Arbeit wird der Fall der Nutzung von Services aus einer Cloud oder dem

Web außer Acht gelassen. Die Einbeziehung derartiger Services in den Betrieb würde die Analyse der Erfordernisse des Betriebs um einiges komplexer gestalten.

Die im Folgenden dargestellten Änderungen und Anforderungen für den Betrieb sind für den Betrieb von Lösungen mit Bestandteilen in Cloud oder Web analog durchzuführen. Es ist anzunehmen, dass die in dieser Arbeit gewonnen Erkenntnisse auf derartige Lösungen ebenso angewendet werden können, aber auch neue weiterführende und noch ungeklärte Aspekte des Betriebs nachzuweisen sind.

Dagegen werden Fragen vernachlässigt, wie

- wem gehört ein Service und wie ist die Kostenverteilung bei dessen Nutzung,
- wer entscheidet, ob ein verfügbarer Service genutzt auch von anderen Geschäftsabteilungen genutzt werden darf etc.,

wie sie von [Cummins, 2009] diskutiert werden. Es wird davon ausgegangen, dass verfügbare Services für alle nutzbar sind.

Im folgenden Abschnitt wird auf die fundamentalen Unterschiede zwischen Client-Server-Architekturen und SOA beim Betrieb einer darauf basierenden Anwendung eingegangen.

2.5 Einführung in das Beispiel Kundenauftragsprozess

Um die Schwierigkeiten und neuen Herausforderungen durch SOA zu illustrieren, wird ein fiktiver, vereinfachter Prozess zur Bearbeitung von Kundenaufträgen verwendet, wie in Abbildung 2.3 dargestellt. Für die grafische Darstellung des Prozesses werden Ereignis-gesteuerte Prozessketten (EPK) [Scheer, 2000] eingesetzt. Eine EPK besteht aus einer sequentiellen Folge von Ereignissen, Aktivitäten oder Funktionen und Prozessflüssen. Eine EPK ist ein gerichteter Graph, welcher als farbiges Petri-Netz [Chen und Scheer, 1994] zur Beschreibung von Geschäftsprozessen genutzt werden kann. Eine andere grafische Darstellungsmöglichkeit wären z.B. UML Aktivitätsdiagramme. Die verschiedenen Darstellungsformen sind aber durchaus untereinander transformierbar [Gernert et al., 2005], weswegen in diesem Beitrag ohne Beschränkung der Allgemeingültigkeit EPK zur Verdeutlichung der Problematik gewählt wurden. In einer EPK wird die Abfolge betrieblicher Aktivitäten Ereignis-orientiert gesteuert. Im Fall mehrerer möglicher Ergebnisse einer Aktivität wird ein Entscheidungsknoten (flow operator) verwendet, um die verschiedenen Prozessflüsse darzustellen. Zudem kann es auch parallele Prozessabläufe geben, das heißt, verschiedene Aktivitäten, die auch Funktion oder Geschäftsprozessschritt genannt werden, können parallel ausgeführt werden. Diese Form der Darstellung findet bei der Architektur

integrierter Informationssysteme (ARIS) [Scheer, 2002] Anwendung, welches die Geschäftsprozessmodellierung auf Basis von EPK unterstützt. Die Notation in Abbildung 2.3 entspricht der ARIS-Notation von EPKs.

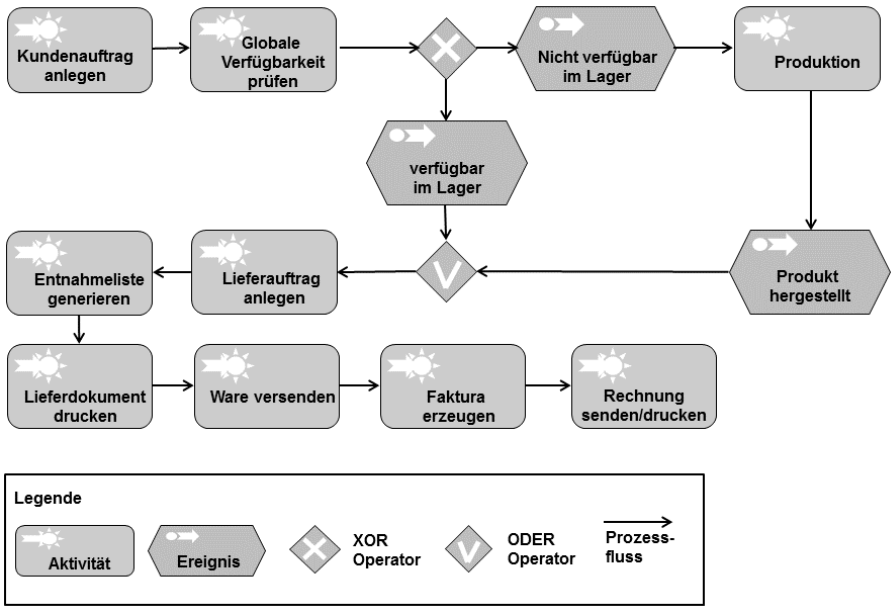

Abbildung 2.3: Beispiel: Kundenauftragsprozess

Die Bearbeitung von Kundenaufträgen beginnt mit dem Anlegen des Kundenauftrags. Die nächste Aktion ist die Überprüfung, ob die georderte Ware global verfügbar ist, z.B. in einem Lager. Falls die bestellten Produkte nicht verfügbar sind, sind sie zu produzieren. Der Einfachheit halber wurde die Produktion als eine einzige Aktivität zusammengefasst, was in der Realität kaum der Fall sein wird. Ist die bestellte Ware produziert bzw. ist bereits im Lager verfügbar, wird der Lieferauftrag angelegt. Der nächste Schritt ist die Generierung der Entnahme- oder Stückliste. Um die Lieferung zu komplettieren, sind die Lieferdokumente zu drucken. Nun kann die bestellte Ware an den Kunden versandt werden. Im nächsten Schritt wird die Faktura erstellt und schließlich die Rechnung gedruckt. In der Praxis wird der gesamte Kundenauftragsprozess weitaus komplexer sein. An dieser Stelle soll auf dieses einfache Beispiel beschränkt werden, weil bereits daran die neuen Herausforderungen gezeigt werden können.

In der Client-Server-Architektur-Welt sind die Aktivitäten innerhalb des Geschäftsprozesses mit den technischen Systemen verbunden. Man kann sie direkt den betreffenden Systemen zuordnen, wie in Abbildung 2.4 mit ARIS dargestellt.

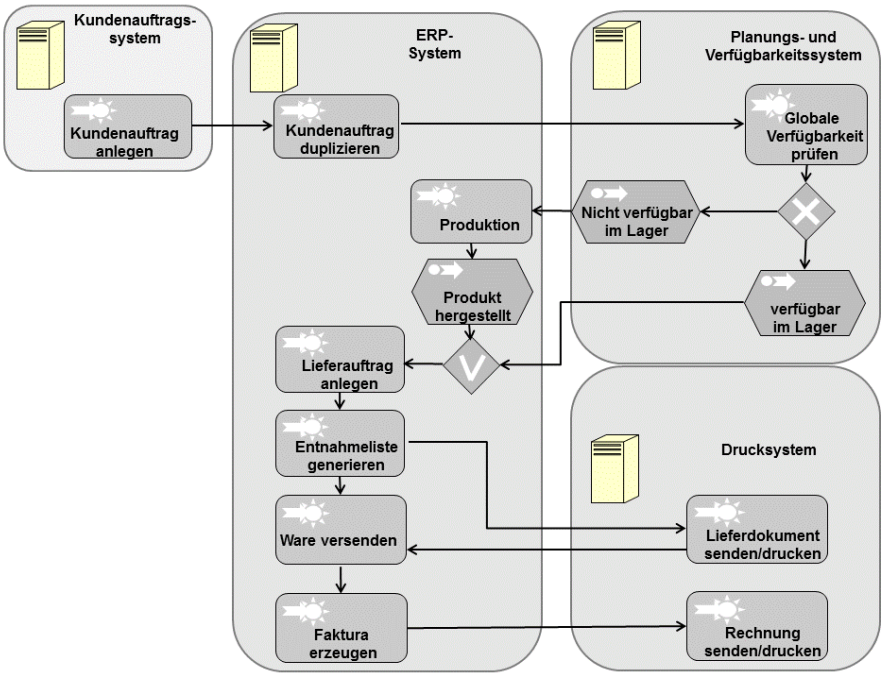

Abbildung 2.4: Zuordnung der Aktivitäten zu technischen Systemen in der Client-Server-Welt

Für den fiktiven Kundenauftragsprozess kann der Kundenauftrag zunächst in einem separaten Kundenauftragssystem angelegt werden. Die Daten werden dann in das zentrale Enterprise Ressource Planning (ERP)-System übertragen und bearbeitet. Die Verfügbarkeitsprüfung findet in einem speziell für Planung und Verfügbarkeitsmanagement vorgesehenem System statt. Die Produktion erfolgt unter Kontrolle des ERP-Systems. Speziell für das Drucken von Dokumenten gibt es ein Drucksystem. Mit der zunehmenden Service-Orientierung einer Lösung verschiebt sich die systemorientierte Sicht der Client-Server-Architektur zugunsten einer serviceorientierten Sicht. Abbildung 2.5 zeigt am Beispiel des Kundenauftragsprozesses in ARIS-Notation den Aufbau einer SOA-Lösung aus technischer Sicht. Den dargestellten Aktivitäten entspricht der Einfachheit halber jeweils ein Service.

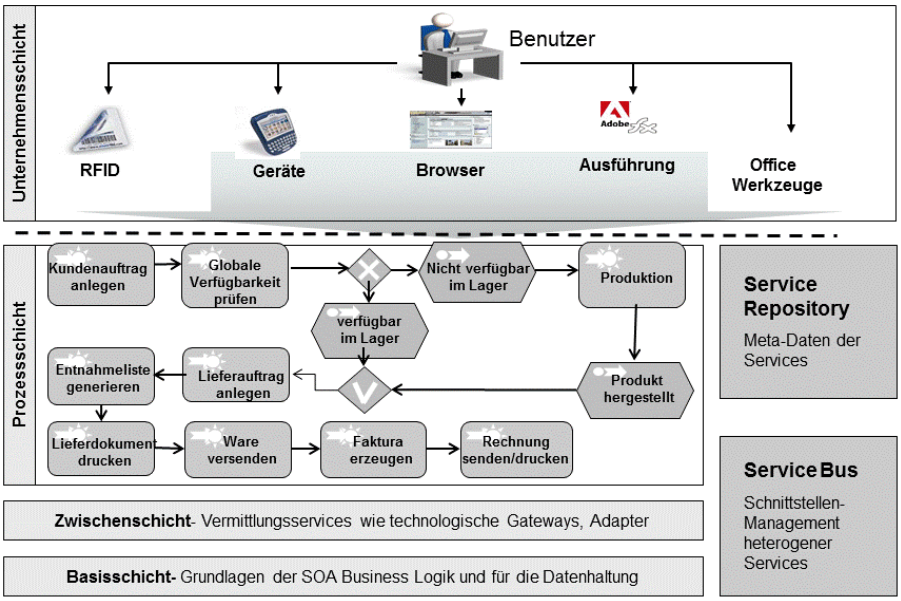

Abbildung 2.5: Technische Sicht auf die SOA-basierte Lösung eines Kundenauftragsprozesses

Im Allgemeinen haben die Benutzer ein vereinheitlichtes Benutzerinterface auf die IT-Lösungen, in diesem Fall also für die Nutzung des Kundenauftragsprozesses. Jeder Service innerhalb der SOA muss die Anforderungen von Service Repository und Service Bus bedienen, wie dies z.B. bei [Office of Governance Commerce, 2007] spezifiziert ist.

Ein wesentlicher Vorteil für die Endbenutzer solcher SOA-basierten Lösungen ist die Entkopplung von technischer Implementierung und Geschäftsfunktionalität. Es besteht keine Notwendigkeit für den Benutzer, die eingesetzten technischen Komponenten oder die Hardware zu kennen. Für den Benutzer ist es irrelevant, ob der Service *Kundenauftrag anlegen* beispielsweise in einer Java-Umgebung abläuft oder in einem proprietären System ausgeführt wird. Wird dieser Service ausgetauscht, erwartet der Benutzer, dass der Kundenauftragsprozess davon unbeeinträchtigt und unverändert verfügbar ist. Diese technischen Unterschiede sind allerdings für den Betrieb einer solchen Lösung von wesentlicher Bedeutung. Aus Betriebssicht ist es eine wesentliche Änderung, wenn die Benutzer von der Notwendigkeit der Kenntnis technischer Details der von ihnen verwendeten Lösung befreit werden. Bisher wurde angenommen, dass der Anwender einen Service selbst betreibt bzw. durch einen externen Dienstleister betreiben lässt. Ein weiterer wesentlicher Unterschied zum Aufbau von Client-Server-basierten Lösungen besteht darin, dass frei verfügbare Services aus dem Web in SOA-Lösungen integriert werden können, wie z.B. Suchmaschinen, Vergleichsplattformen oder Commerce Portale. Damit

obliegt die Verantwortung für den Betrieb des einzelnen Services nicht mehr dem Anwender. Jedoch die Verantwortung für die Qualität des Betriebs der Geschäftsprozesse als Ganzes obliegt dem Anwender. In der Folge heißt das, dass auch Fremdservices in gewissem Maße in den kundeneigenen Betrieb eingebunden werden. Zwar entfallen Aufgaben, wie z.B. die Optimierung, Konfiguration oder Wartung des Fremdservices aber andererseits muss z.B. die Gesamtperformance des eigenen Geschäftsprozesses übergreifend ermittelt und optimiert werden. Performancevereinbarungen sind Teil eines SLA, deren Erfüllung regelmäßig nachgewiesen werden muss. Der Konflikt an dieser Stelle besteht in der möglicherweise sogar kostenfreien, aber auch unverbindlichen Verfügbarkeit und Leistungsfähigkeit eines solchen Services. Der Nutzer hat keine zugesicherten Garantien zur Verfügbarkeit.

Entgegen der für den Betrieb von SOA-basierten Lösungen erforderlichen serviceorientierten Sicht sind die administrativen Aufgaben bei heutigen Lösungen herstellerspezifisch geprägt. Die aktuell beim Betrieb verwendeten Werkzeuge sind zudem noch häufig den Client-Server-basierten Lösungen verhaftet. Das heißt, der Betrieb basiert auf:

- Kunden- bzw. Dienstleister-Verantwortung für den Betrieb,
- System- und Technologie-spezifischen administrativen Werkzeugen,
- System- und Technologie-spezifischen Aktionen und Qualitätskennzahlen,
- System- und Technologie-spezifischem Wissen und Erfahrungen, um erstes und zweites einzusetzen und erfüllen zu können.

Mit dem Übergang zu SOA muss sich der systemorientierte Ansatz hin zu einem lösungsorientierten Ansatz verschieben. Bevor die Unterschiede im Detail analysiert werden, soll nachfolgenden Abschnitt dargestellt werden, welche Tätigkeiten und Prozesse üblicherweise dem nicht eindeutig definierten Begriff „Betrieb“ zuzurechnen sind.

3 Betrieb als Teil der IT-Service und Supportprozesse

Wie bereits angerissen, gibt es die Definition des Begriffs und des Inhalts des IT-Betriebs nicht. Umgangssprachlich wird darunter die regelmäßige, dauerhafte Bereitstellung von IT-Lösungen zur Nutzung verstanden. Damit erfolgt eine Abgrenzung zu Design und Entwicklung von IT-Lösungen. In dieser Arbeit wird auf die IT Infrastructure Library [Office of Governance Commerce, 2007] zurückgegriffen, um die für diese Arbeit relevanten Aufgaben und Prozesse des Betriebs zu beschreiben und zu analysieren.

3.1 Bedeutung der IT Infrastructure Library

Welche Anforderungen und Prozesse für den IT Service und Support bestehen und wie sie geeignet, aber herstellerunabhängig zu implementieren sind, beschreibt die IT Infrastructure Library (ITIL) [Office of Governance Commerce, 2007]. ITIL formuliert bewährte Techniken (*Good Practices*) bei der Etablierung und Durchführung von IT Service Management Prozessen entlang des Lebenszyklus einer Dienstleistung. Die Zielgruppe von ITIL sind daher in erster Linie IT-Dienstleister, die ihre Dienste innerhalb einer Firma oder einer externen Firma als externer Betreiber anbieten. Es ist darauf zu achten, dass Service im ITIL-Sinne als Dienstleistung zu verstehen ist und nicht gleichzusetzen ist mit Services als Teil der SOA. ITIL definiert eine Dienstleistung als einen Service, durch den ein Kunde einen Wert erlangt, in dem ihm Ergebnisse bereitgestellt werden, die der Kunde erreichen will, ohne selbst Eigentümer der spezifischen Kosten und Risiken der Erbringung zu sein. Im Sinne von ITIL sind sowohl die Bereitstellung von SOA-Services, als auch entsprechender Hardware und der Betrieb dessen als Dienstleistung zu betrachten. ITIL beschreibt Prozesse, Verantwortlichkeiten und Aktivitäten hersteller- und technik-unabhängig. Nichtsdestotrotz ist es auf der Basis der praktischen Erfahrungen der bis dahin aktuellen Techniken und Architekturen entstanden, die zumeist Client-Server-orientiert sind.

Abbildung 3.1 zeigt das Verständnis ITIL hinsichtlich der Lebensphasen von Dienstleistungen.

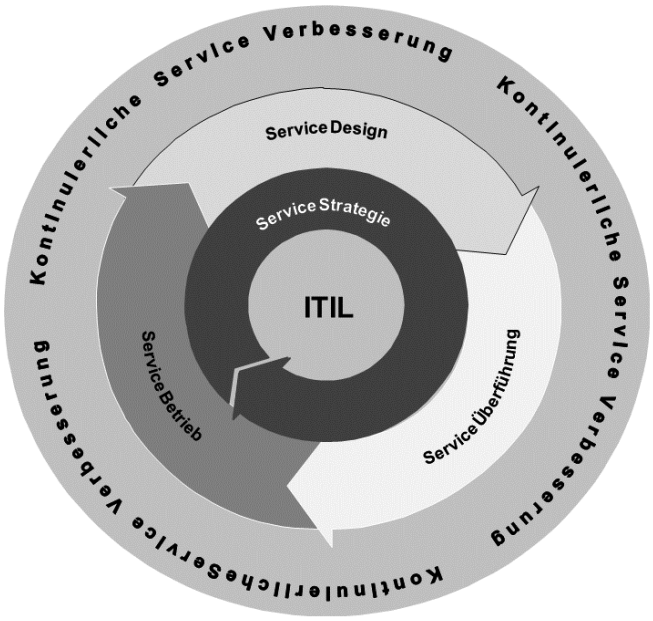

Abbildung 3.1: Service Lebenszyklus gemäß [Office of Governance Commerce, 2007]

Diese Phase Service Strategie (*Strategy*) umfasst alle Prozesse, die zur Planung eines Dienstleistungsangebots erforderlich sind. Damit ist sie stark management-orientiert und dient auch der Abstimmung zwischen dem Dienstleistungsanbieter und dem IT-Kunden. Im Sinne von ITIL ist auch die technische Bereitstellung von SOA-Services eine Dienstleistung. In dieser Phase würden Dienstleister und Kunde vereinbaren, welche Services zu der geplanten SOA-basierten Lösung integriert werden sollen und eventuell auch, welche Services als Stand-by bereitgestellt werden sollen. Diese Phase steht nicht direkt in Bezug zum Betrieb und wird daher im Folgenden nicht weiter betrachtet. In der Phase Service Design wird gemäß ITIL konzipiert, welche Dienstleistungen erforderlich sind. Es kommt darauf an, die Dienstleistungen mit dem Business der Firma in Einklang zu bringen. Dabei bezieht sich das einerseits auf die Erwartungen der Firma, die die Dienstleistungen in Anspruch nehmen wird, als auch auf das eigene IT-Business, für das sich die angebotenen Leistungen rentieren müssen. Die Phase Service Überführung (*Transition*) betrachtet die erforderlichen Abläufe und Vorbereitungen für den Produktivstart von Dienstleistungen. Alle Prozesse, die permanent und Tag für Tag auszuführen sind, betrachtet ITIL in der Phase Service Betrieb (*Operation*). Dazu gehören auch die möglichst effektive Lieferung der Dienstleistungen sowie die Benutzerbetreuung (*Support*). Zum Lebenszyklus gehört auch das permanente Streben nach Optimierung, so dass die Phasen von einem

kontinuierlichen Service Verbesserungsprozess (*Continual Service Improvement*) umspannt werden. Betrachtet man die Bereitstellung und Nutzung einer IT-Lösung in der Gesamtheit als eine mögliche Dienstleistung mit allen dazugehörigen Facetten, so werden die benannten Phasen beliebig oft durchlaufen. Gemäß ITIL beinhaltet die Phase Betrieb das tägliche Geschäft des Betriebs. Eine IT-Lösung, die genutzt wird, wird auch in regelmäßigen Abständen angepasst, d.h. geändert, womit z.B. auch die Ausführung des Änderungsprozesses teilweise dem Betrieb einer IT-Lösung zuzurechnen ist. Die eigentliche Implementierung des Änderungsmanagements ist jedoch eher der Phase der Service Überführung in die Produktion gemäß ITIL zuzuordnen.

3.2 Übersicht über ITIL

Im Nachfolgenden werden mittels ITIL die IT Service und Support Prozesse umrissen und bestimmt, welche Teile zum typischen Betriebsgeschäft gehören und somit in dieser Arbeit zu untersuchen sind. Abbildung 3.2 zeigt die Zuordnung der IT Service und Support Prozesse zu den verschiedenen Phasen.

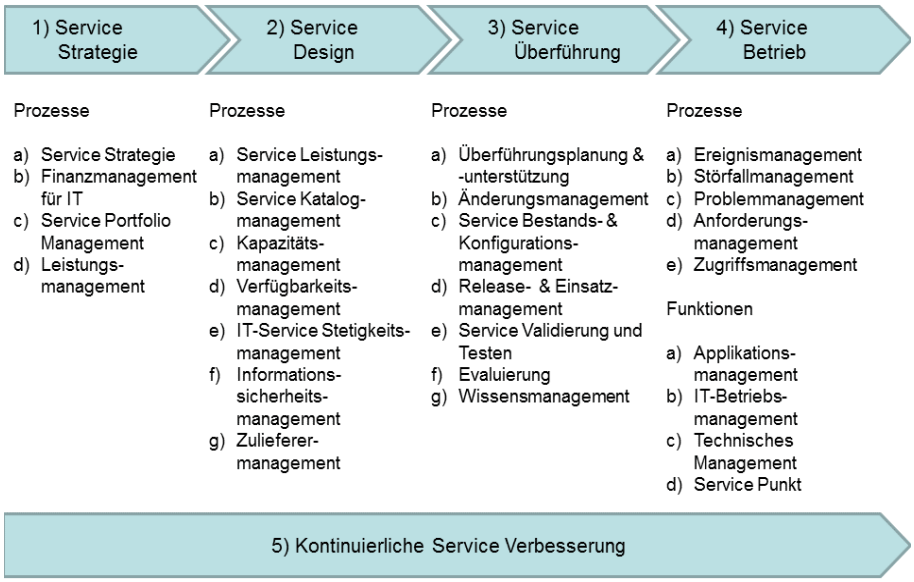

Abbildung 3.2: ITIL-Phasen und Prozesse in der Übersicht [Office of Governance Commerce, 2007]

Wie zu erkennen, können die ITIL-Prozesse und Funktionen sowie deren Qualitätskriterien auf den Betrieb beliebiger Lösungsarchitekturen bezogen werden, also auch auf SOA-basierte Lösungen. Im Folgenden wird die ITIL-

Struktur genutzt, um zu untersuchen inwiefern die Architektur einer Lösung den Betrieb beeinflusst. Die einzelnen Prozesse und Funktionen des ITIL werden umrissen und die für diese Arbeit technischen Betriebsaspekte dargestellt. Wird eine Aufgabe von einem einzelnen Expertenteam ausgeführt, wird dies in ITIL als *Funktion* bezeichnet. Werden dagegen die Aufgaben in einer Abfolge von Schritten von verschiedenen Teams ausgeführt, bezeichnet ITIL dies als *Prozess*.

3.3 Service Design Phase

In dieser Phase werden entsprechend der Erwartungshaltung des Kunden die Grundlagen für die späteren Dienstleistungen gelegt. Der Dienstleister selbst kann dabei der gleichen Firma angehören oder ein firmenexterner Anbieter sein. *Kunde* bezeichnet den Dienstleistungsnehmer. Schlussendlich handelt es sich dabei um die Benutzer von IT-Lösungen zur Ausführung definierter Geschäftsprozesse. Dementsprechend wird sich die Erwartungshaltung des Kunden in erster Linie in Forderungen hinsichtlich Qualität, Quantität, Verfügbarkeit und Performance seiner Geschäftsprozesse niederschlagen.

3.3.1 Service Leistungsmanagement

Das Service Leistungsmanagement (*Service Level Management*- SLM) ist der Prozess innerhalb der IT Service und Support Prozesse, der die Qualität aller Prozesse mit den vereinbarten Leistungskriterien vergleicht. Dazu werden ausgehend von der Erwartungshaltung des Kunden, dem Dienstleistungsnehmer, Qualitätskriterien definiert, im sogenannten Service Level Agreement hinterlegt und gemessen. ITIL beschreibt den Prozess des Service Level Management und die Verantwortlichkeiten. SLA sind kennzahlenbasierte Vereinbarungen eines Dienstleistungsanbieters mit seinem Kunden bezüglich der zu gewährleistenden Servicequalität [Bernhard, 2003]. Wie bei einem Vertragswerk üblich, bestimmen im SLA die Vertragsparteien Rollen, Leistungen und Verantwortlichkeiten. Die Ziele der Vertragspartner werden in Form von Kennzahlen und avisierten Messgrößen definiert. Des Weiteren werden in dem Vertrag Messsysteme und -methoden hinterlegt. Aus Sicht des Betriebs und dieser Arbeit sind in erster Linie die Qualitätskriterien und deren Messung von Bedeutung. Die Erwartungshaltung des Kunden hinsichtlich seiner Lösung ist unabhängig von der gewählten Architektur der Lösung. Die Schwierigkeit besteht vielmehr darin, die Erwartungshaltung in eine geeignete Matrix und Messkriterien abzubilden, so dass die Messwerte tatsächlich einen Rückschluss auf die Erfüllung der Erwartungshaltung zulassen. Es werden die Unterschiede zwischen Client-Server- und SOA-basierten Architekturen aufgezeigt.

3.3.2 Service Katalogmanagement

Dieser Prozess beschreibt das optimale Vorgehen, um einen Service Katalog (*Service Catalog*) zu erstellen und zu pflegen. Der Katalog beschreibt das Dienstleistungsangebot verbunden mit den entsprechenden Preisen, aus dem der Kunde wählen kann. Insbesondere für externe Dienstleister stellt der Katalog die Leistungsfähigkeit des Anbieters dar. Dabei umfasst dies natürlich auch die Leistungsfähigkeit in Bezug auf den Betrieb von Lösungen. Der Prozess ist jedoch nur indirekt von der Architektur einer Lösung betroffen. Eine interessante Fragestellung wäre z.B., wie sich die Kosten des Betriebs von SOA- basierten und Client-Server-basierten Lösungen unterscheiden, was jedoch in dieser Arbeit nicht thematisiert werden soll.

3.3.3 Kapazitätsmanagement

Die Aufgabe des Kapazitätsmanagements (*Capacity Management*) ist es sicherzustellen, dass jederzeit genügend Kapazität für die Erfüllung der Service Vereinbarungen verfügbar ist und zwar beginnend von der Infrastruktur bis zum nötigen Personal für das IT Service und Support Management. Aus Sicht dieser Arbeit wird auf das Kapazitätsmanagement der Hardware-Ressourcen fokussiert. Ein wichtiger Aspekt ist die Messung der verschiedenen Hardwareressourcen, wie CPU-Auslastung, Belegung des Speicherplatzes oder der Hauptspeichernutzung. Es wird gezeigt, welche neuen Anforderungen sich aus der Entwicklung zu SOA-basierten Lösungen ergeben.

3.3.4 Verfügbarkeitsmanagement

Die Mission des Verfügbarkeitsmanagements (*Availability Management*) bezieht sich aus ITIL-Sicht auf die Verfügbarkeit der mit dem Kunden vereinbarten Dienstleistungen. Dies umfasst auch die Verfügbarkeit von Lösungen, die ein Anbieter für einen Kunden betreibt. Innerhalb des SLA wird dabei die Verfügbarkeit der Geschäftsanwendungslösung jedoch häufig bereits mit technischer Verfügbarkeit gleichgesetzt und auf Verfügbarkeit von Systemen übertragen. Es wird gezeigt, dass diese Übertragung für SOA-basierte Lösungen nicht hinreichend ist. Gemäß ITIL bezieht sich Verfügbarkeit immer auf die mit dem Kunden getroffenen Vereinbarungen, also letztlich auf die Erwartungshaltung des Kunden. Verfügbarkeit setzt sich aus folgenden Bestandteilen zusammen:

- Verfügbarkeit (*Availability*) in dem Sinne wie lange ein Service oder eine Komponente überhaupt unterbrechungsfrei betreibbar sind, bevor z.B. eine Wartung erforderlich ist.
- Zuverlässigkeit (*Reliability*) der Lösung, sie im vereinbarten Zeitraum zu betreiben.

- Wartungsfähigkeit (*Maintainability*) umfasst die Möglichkeit eine Dienstleistung oder auch technische Komponenten im Fehlerfall wiederherzustellen, z.B. durch Recovery.
- Servicefähigkeit (*Serviceability*) bedeutet die Fähigkeit eine Lösung so zu warten, dass die Verfügbarkeit gewährleistet werden kann. Dies umfasst z.B. auch die Möglichkeit der Behebung von Störfällen im vereinbarten Zeitraum.
- Fehlertoleranz (*Resilience*) bezeichnet die Fähigkeit den normalen Betrieb und die Funktionstüchtigkeit auch im Fehlerfall zu erhalten. Eine gängige Methode Fehlertoleranz zu steigern ist Redundanz, wozu auch die Bereitstellung einer redundanten Lösung oder technischen Komponente (Stand by-Lösungen) gehört.
- Sicherheit (*Security*) ist nicht zuletzt ein wichtiger Aspekt, der die Gesamtverfügbarkeit einer Lösung beeinflusst. Sicherheit selbst besteht aus den Bereichen Datensicherheit, Integrität und Vertrauenswürdigkeit.

In dieser Arbeit werden die technischen Möglichkeiten und Voraussetzungen SOA-basierter Lösungen unter Ausnutzung des Flexibilitätspotentials untersucht. Auf Grund der Komplexität des Themas Sicherheit wird in dieser Arbeit auf die anderen Bestandteile der Verfügbarkeitsmanagement Prozesses fokussiert.

3.3.5 IT-Service Stetigkeitsmanagement

Unter dem Management der IT-Service Stetigkeit (*IT Service Continuity Management*) beschreibt ITIL das Management des Geschäfts im Fall einer Katastrophe. Unter einer Katastrophe versteht ITIL den mindestens teilweisen Ausfall der IT-Dienstleistungen und damit einhergehend auch den Ausfall der im Verfügbarkeitsmanagement vorbereiteten Maßnahmen und Prozesse zur Wiederherstellung des Normalbetriebs. Katastrophen sind definitiv Ausnahmesituationen die nicht zum alltäglichen Geschäft des Betriebs gehören. Daher wird das IT-Service Stetigkeitsmanagement lediglich im Zuge der Untersuchung des Einflusses der SOA auf das Verfügbarkeitsmanagement in der Arbeit allgemein umschrieben.

3.3.6 Informationssicherheitsmanagement

Der Fokus des Prozesses Informationssicherheitsmanagement (*Information Security Management*) liegt auf dem Management der IT-Sicherheit. Dabei gilt es die Balance zu finden zwischen Zugriffs- und Bearbeitungsbedarf einerseits und Sicherheit und gesetzlichen Regelungen andererseits. Der Prozess der Verwaltung, Überwachung und Bearbeitung unterscheidet sich

nicht zwischen Client-Server- und SOA-basierten Lösungen. Der wesentliche Unterschied liegt wiederum in der eigentlichen Umsetzung. Zwar kann auch das Informationssicherheitsmanagement als im weitesten Sinne für den Betrieb einer Lösung erforderlich angesehen werden, doch soll in dieser Arbeit nicht vertiefend darauf eingegangen werden. Neben technischen Aspekten kommen in diesem Prozess gesetzliche Regelungen ins Spiel, die den Prozess weit über technische Betrachtungen hinaus komplizieren. Sicherlich ist es dieses Thema wert, an anderer Stelle eingehend untersucht zu werden.

3.3.7 Zulieferermanagement

Der Prozess des Zulieferermanagements (*Supplier Management*) wird in dieser Arbeit nicht als Element des reinen Betriebs betrachtet. Denkt man an die Möglichkeit der Einbindung von Services externer Anbieter in eine existierende SOA-basierte Lösung, werden sich für diesen Prozess jedoch wesentliche neue Aspekte gegenüber der in Client-Server-basierten Verteilung von Dienstleistungen ergeben. Man denke beispielsweise an die Abrechnung von Nutzungsgebühren für in einer SOA-Lösung genutzte Services. Diese Themen werden jedoch nicht in dieser Arbeit betrachtet, da sie nicht der direkten Sicherstellung des Betriebs einer Lösung diesen.

3.3.8 Zusammenfassung der Phase Design

In der Design Phase der IT-Service und Unterstützungsprozesse werden auch die Grundlagen für den späteren Betrieb einer Lösung geschaffen. Es werden wesentliche Prozesse definiert und implementiert, die während der Phase des Betriebs einer Lösung täglich genutzt werden müssen. ITIL beschreibt dabei lediglich die Grundzüge, Rollen und Verantwortlichkeiten innerhalb der Prozesse. Damit bleibt ITIL unabhängig von der Architektur einer Lösung. Es wurde jedoch gezeigt, inwieweit sich die Architektur einer Lösung bereits in der Phase Design auf die Prozesse auswirkt.

3.4 Service Überführungsphase

Nach der Design Phase sind eine Lösung, ein Release einer Lösung oder auch eine größere Änderung an einer Lösung in die Produktion zu überführen. In der Phase der Überführung in die Produktion (*Service Transition*) werden genau die dafür nötigen Schritte inklusive einer besonderen Bereitschaftsphase für eine begrenzte Zeit nach dem Produktivstart geplant.

3.4.1 Überführungsplanung und -betreuung

Für die Entwicklung des Plans für den Produktivstart und einer besonderen Bereitschaft zur Unterstützung danach, zeichnet der Prozess

Überführungsplanung und -betreuung (*Transition Planning & Support*) verantwortlich. Dabei greift der Prozess auf die Möglichkeiten anderer ITIL-Prozesse zurück, insbesondere dem Störfall- und Problemmanagement. Das eigentliche Prozessmanagement ist wiederum unabhängig vom technischen Aufbau einer Lösung, vielmehr sind es die Werkzeuge und Methoden, die auch die Grundlage anderer Prozesse bilden. Dieser Prozess wird daher nicht separat auf seine Veränderungen hinsichtlich der Architektur einer Lösung untersucht, sondern im Zusammenhang mit denen zu ihm in direktem Bezug stehenden Prozesse.

3.4.2 Änderungsmanagement

Aus Sicht von ITIL bezieht sich eine Änderung (Change) auf jegliche Arten von Änderungen an der Dienstleistung eines Providers, was auch technische Änderungen an Programmen, technischen Komponenten oder auch Systemkonfigurationen beinhaltet. Die in ITIL beschriebenen bewährten Methoden beschreiben vorrangig das Management dieses Prozesses. Das Change Management wird in ITIL in der Überführungsphase gesehen, da es beschreibt, wie man Änderungen verwaltet, protokolliert und in die Produktion überführt. Dabei wird jedoch nicht explizit darauf eingegangen, wie eine Änderung mit welchen Werkzeugen in eine genutzte Lösung integriert wird. Während der Änderungsmanagement Prozess an sich inklusive seiner Qualitätskriterien erhalten bleibt, zeigen sich für die Übernahme einer Änderung in die aktiv genutzte Lösung wesentliche Unterschiede zwischen Client-Server- und SOA-basierten Lösungen.

3.4.3 Service Bestands- und Konfigurationsmanagement

Der Prozess bezieht sich auf das Management der technischen und softwarebezogenen Bestandteile. Als Konfigurationselement (*Configuration Item- CI*) werden die hardware-technischen als auch software-technischen Bestandteile einer Lösung verstanden, die eine Versionsverwaltung erfordern. Dazu gehören beispielsweise Computer, aber auch Drucker oder Router sowie Softwarebausteine wie Werkzeuge, Überwachungswerkzeuge, technische Komponenten und im Fall von SOA auch Services. Die Definition, was genau ein Konfigurationselement ist, bleibt in ITIL dem Nutzer überlassen. Es wird gezeigt, dass die Wahl geeigneter Konfigurationselemente sehr wohl von der Architektur beeinflusst wird. Das Service Bestands- und Konfigurationsmanagement (*Service Asset and Configuration Management*) im ITIL beschreibt eine Konfigurationsmanagement Datenbank (*Configuration Management Database- CMDB*) zur Verwaltung der Konfigurationselemente und ihrer Relationen. CMDB liefert die grundlegenden Informationen für das Änderungsmanagement. Umgekehrt beeinflusst das Änderungsmanagement

das Konfigurationsmanagement. Beide Prozesse sind eng miteinander verknüpft.

3.4.4 Release- und Einsatzmanagement

Analog dem Änderungsmanagement ist ein Release als eine größere Einheit einer Anzahl von Änderungen zu betrachten. Der Prozess des Release Management und Einsatzmanagement (*Release and Deployment Management*) beschreibt wann es sinnvoll ist, Änderungen zu einem Release zusammenzufassen. Wesentliches Kriterium sind dabei die Auswirkungen und das Risiko von Änderungen für den normalen Betrieb. Änderungen die den tagtäglichen Betrieb und die gewöhnliche Nutzung einer Lösung gefährden, werden bevorzugt in einem Release gebündelt und außerhalb des normalen Betriebs während so genannter Wartungsfenster in die produktive Lösungslandschaft freigegeben (D*eployment*). Aus Betriebssicht ist es somit erforderlich einzuschätzen, welche Änderungen den normalen Betrieb beeinflussen und daher außerhalb des normalen Betriebs in einem gesondert zu bestimmenden Wartungsfenster in den produktiven Betrieb übernommen werden. Des Weiteren gilt es zu planen, wie lang und aufwändig die Realisierung eines Releases ist und dementsprechend die Größe des Wartungsfensters zu definieren.

3.4.5 Service Validierung und Testen

Die Inhalte dieses Prozesses Service Validierung und Testen (*Service Validation and Testing*) wurden erst zum ITIL Release III als eigenständiger Prozess eingeführt. Jede Änderung und jedes neue Release müssen vor der produktiven Inbetriebnahme getestet werden. Dabei gibt es zwei wesentliche Dimensionen der Validierung und des Testens:

- Funktionstüchtigkeit- bezeichnet die erforderlichen Tests zur Überprüfung, ob die Änderung oder das Release den gestellten funktionalen Forderungen genügen,
- Verwendungsfähigkeit- testet hinsichtlich der Benutzbarkeit der Änderungen.

Diese Ziele des Testens sind unabhängig von jedweder Architektur einer Lösung und werden in dieser Arbeit nicht noch einmal vertieft. Die Architektur wirkt sich dagegen wesentlich auf die erforderliche Testumgebung aus. Während im Client-Server-Bereich eine Lösung zumeist in nur einem oder sehr wenigen Systemen abläuft, setzen sich SOA-basierte Lösungen aus Services zusammen. Daher muss für die Untersuchungen unterschieden werden zwischen Änderungen, die nur einen Service betreffen und solchen, die serviceübergreifender Natur sind. Eine wesentliche Herausforderung in SOA ergibt sich bei der Bereitstellung einer geeigneten Testumgebung, was in Abschnitt 4.8 untersucht wird.

3.4.6 Evaluierung

Der Evaluierungsprozess (*Evaluation*) untersucht Kosten und Nutzen von IT Dienstleistungen und deckt mögliches Optimierungspotential auf. Folglich resultieren aus diesem Prozess Änderungen an einer Lösung. Auch die Ausführung der Bewertung wird durch die Architektur einer Lösung beeinflusst, jedoch eher durch die Erweiterung der zu evaluierenden Möglichkeiten und deren Kosten. Der Prozess soll daher nicht untersucht werden. Kosten und Nutzen werden im Zusammenhang mit dem jeweiligen Prozess aus technischer Sicht beleuchtet.

3.4.7 Wissensmanagement

Die Hauptaufgabe des Wissensmanagement (*Knowledge Management*) ist die Erstellung und Bereitstellung des erforderlichen Wissens zur Nutzung und dem Betrieb einer Lösung. Indirekt wird das Wissensmanagement von der zugrunde liegenden Architektur einer Lösung beeinflusst. Während Client-Server-basierte Lösungen stark von dem System geprägt sind, unterstützen SOA-basierte Lösungen einen einheitlichen Benutzerzugang. Das reduziert das erforderliche Wissen zur Nutzung erheblich insbesondere in Hinsicht auf das erforderliche technische Wissen. Es wird untersucht, welche Konsequenzen die Nutzung der Flexibilität von SOA-basierten Lösungen hat.

3.4.8 Zusammenfassung der Phase Überführung

Die Phase der Überführung in die Produktion bildet den Übergang zum täglichen Betrieb einer Lösung. In dieser Phase muss der tägliche Betrieb in seiner Gesamtheit und abschließend vorbereitet werden. Durch das Abstraktionslevel von ITIL sind die Definitionen der zu dieser Phase gehörenden Prozesse wiederum unabhängig von der Architektur einer Lösung: betrachtet man jedoch die einzelnen Prozesse konkreter insbesondere hinsichtlich der praktischen Realisierung finden sich wesentliche Unterschiede zwischen Client-Server- und SOA-basierten Architekturen.

3.5 Service Betriebsphase

Die Phase des Betriebs (*Service Operation*) der Services beinhaltet alle Prozesse und Aufgaben, die beim regelmäßigen Betrieb einer Lösung anfallen. Um die Kosten möglichst gering zu halten, müssen die Abläufe möglichst effektiv und effizient gestaltet werden. Für den Fall einer Unterbrechung der normalen Geschäftsprozesse müssen schnelle, effektive Reaktionsmethoden zur Verfügung stehen.

3.5.1 Ereignismanagement

Als Ereignis wird von ITIL jede Statusänderung eines Konfigurationselements betrachtet. Darüber hinaus sind Ereignisse auch Statusänderungen von Kennzahlen, die eine Reaktion des verantwortlichen Betreibers erfordern. Ereignisse können in direkter Verbindung zu einem Störfall oder potentiellen Störfall stehen. ITIL beschreibt, wie die erforderlichen Abläufe im Ereignismanagement zu implementieren sind, definiert jedoch nicht, welche Kennzahlen oder Statusänderungen als Ereignis einzustufen sind oder gar wie zu reagieren ist. Genau dies ist jedoch stark von der Architektur einer Lösung geprägt. Wesentliches Element des Ereignismanagement ist ein Überwachungswerkzeug (*Monitoring*) für die bestimmten Ereignisse. In dieser Arbeit wird auf die Anforderungen an das Monitoring durch den Übergang von Client-Server- zu SOA-basierten Lösungen im Detail eingegangen, da das Ereignismanagement einer der besonders wichtigen Prozesse im täglichen Betrieb einer Lösung ist. Das Monitoring bildet den Kern der Überwachung der Funktionstüchtigkeit einer Lösung und auch die Quelle der Informationen zur kontinuierlichen Verbesserung einer Lösung und ihres Betriebs.

3.5.2 Störfallmanagement

Als Störfall (*Incident*) wird in ITIL jegliche Störung des Normalbetriebs einer Lösung bezeichnet, wodurch die vereinbarten Service Level verletzt werden könnten. Die Aufgabe des Störfallmanagements ist es, den Normalbetrieb im Falle eines Störfalls möglichst schnell wieder herzustellen. Das kann auch heißen, dass als temporäre Lösung eine Umgehung des Störfalls erstellt wird. Wichtiges Arbeitsmittel des Störfallmanagement ist eine Wissensdatenbank, in der bereits bekannte Störfälle und deren Umgehungen oder Lösungen dokumentiert sind. Stellt sich heraus, dass für einen Störfall noch keine Lösung verfügbar ist, wird der Störfall in ein Problem überführt und an das Problem Management zur Ursachenanalyse und Behebung übergeben.

3.5.3 Problemmanagement

Das Problemmanagement steht in engem Zusammenhang mit dem Störfallmanagement. Die Hauptaufgabe besteht in der Ursachenanalyse und der Beseitigung des Problems. Daher ist es auch der Prozess, der eng mit dem Änderungsmanagement in Verbindung steht. Die Behebung eines Problems erfordert häufig eine Änderung an der Lösung. Zusätzlich gilt es, die Wiederholung gleichartiger Probleme zu vermeiden. Daher pflegt das Problemmanagement die Wissensdatenbank mit der Beschreibung und Lösung des Problems.

Störfall- und Problemmanagement sind zentrale, besonders wichtige Prozesse innerhalb des ITSM. Die Qualität dieser Prozesse beeinflusst direkt

die Zufriedenheit des Kunden. Daher werden üblicherweise für diese beiden Prozesse besonders sorgfältig Qualitätskriterien definiert. Die Auswertung der Messung und Überwachung dieser Qualitätskriterien erfordert eigene Verwaltungswerkzeuge von Störfällen und Problemen, die idealerweise nahtlos mit dem Änderungsmanagement zusammenspielen. In dieser Arbeit wird ein solches Werkzeug vorausgesetzt. Nicht das Management steht im Vordergrund, sondern die eigentliche Behebung von Störfällen und Problemen. ITIL beschreibt zwar das Management beider Prozesse, jedoch geht es nicht detailliert darauf ein, wie ein Problem mit welchen Mitteln zu analysieren und zu lösen ist. Dabei wird gerade die Ursachenanalyse wesentlich durch die Architektur einer Lösung beeinflusst. Am Beispiel eines fiktiven Kundenauftragsprozesses und der Ursachenanalyse im Fehlerfall werden die Unterschiede zwischen Client-Server- und SOA-basierten Lösungen gezeigt.

3.5.4 Anforderungsmanagement

Das Anforderungsmanagement (*Request Management*) organisiert die Anforderungen der Benutzer, die im weitesten Sinn als Kunden der Lösung zu betrachten sind. Das umfasst insbesondere zusätzlichen Informationsbedarf als auch Anforderungen an neue Funktionalitäten. Obwohl sich durch SOA weitaus größere Möglichkeiten zur Erweiterung und Anpassung der Lösung auf Grund der Austauschbarkeit von Services ergeben, bezieht sich dieser Prozess nicht direkt auf den Betrieb und wird daher nicht genauer betrachtet.

3.5.5 Zugriffsmanagement

Das Zugriffsmanagement (*Access Management*) betrachtet und steuert die Zugriffs- und Nutzungsmöglichkeiten in Hinsicht auf die definierten Sicherheitsrichtlinien im Unternehmen. Es ist daher eng mit dem Sicherheitsmanagement verbunden. Obwohl der Prozess für Client-Server- und SOA-basierte Lösungen weitestgehend analog erfolgt, ändern sich die Objekte zu denen Zugriffe zu regulieren sind. Die Systemorientierung in Client-Server-basierten Lösungen erfordert ebenso eine systemorientierte Zugriffskontrolle und -mechanismen. Der erste Schritt ist die Regulierung des Zugriffs von Benutzern auf Systeme. Innerhalb des Systems gibt es systemeigene Berechtigungspflege-Instrumente. In SOA-basierten Lösungen hat man dagegen einen zentralen Zugang zu der Lösung. Dieser ist unabhängig von der darunterliegenden technischen Service-Implementierung, der technischen Ausstattung oder sonstigen technischen Details. Folglich muss die Zugriffssteuerung auf Basis der Lösung und geschäftsobjektorientierter Konzepte erfolgen anstelle eines systemorientierten Ansatzes. Im speziellen darf der Austausch eines einzelnen Services nicht die Anpassung der Berechtigungen erfordern. Die

Zugriffs- und Nutzungsberechtigung muss unabhängig von der Service-Zusammensetzung zum Ausführungszeitpunkt der Lösung sein. Jedweder anderer Ansatz würde die Flexibilität von SOA ungerechtfertigt einschränken. Da dieser Prozess nicht in direktem Zusammenhang mit dem täglichen Betrieb gesehen werden muss, wird in dieser Arbeit nicht gesondert darauf eingegangen. Im Gegensatz zu den anderen Service Phasen beinhaltet die Betriebsphase neben Prozessen auch Funktionen. Im Gegensatz zu den Prozessen, bei denen verschiedene Teams, Rollen und Aufgaben in definierter Abfolge ausgeführt werden, wird eine Funktion von nur einem Expertenteam ausgefüllt.

3.5.6 Funktion: Applikationsmanagement

Im Fokus der Funktion Applikationsmanagement (*Application Management*) steht, das für die Benutzerbetreuung und den Applikationsbetrieb erforderliche Wissen zu erarbeiten und bereitzustellen. Aus Sicht dieser Funktion liegt der Vorteil einer SOA-basierten Lösung in der klaren Trennung zwischen technischem Aufbau und Details und der Nutzung und den einheitlich Benutzerzugang zu der Lösung. Dadurch können Services innerhalb der SOA-basierten Lösung flexibel ausgetauscht werden, ohne dass sich dies unmittelbar auf den Benutzer auswirkt. SOA reduziert das erforderliche technische Wissen für den Endbenutzer der Applikation erheblich. Während das Applikationsmanagement direkt von SOA profitiert, können andere Bereiche des Wissensmanagement nur unter bestimmten Voraussetzungen profitieren, was in dieser Arbeit gezeigt wird.

3.5.7 Funktion: IT-Betriebsmanagement

Die Funktion des IT-Betriebsmanagement (*IT Operations Management*) fokussiert auf den regelmäßigen Betrieb der technischen Infrastruktur und ihrer Pflege. Es geht hauptsächlich um die Wartung und Kontrolle der physischen Hardware als auch der Kontrolle des regelmäßigen Betriebs. Als typische Aufgabe innerhalb des IT-Betriebsmanagement wird das technische Monitoring angesehen, wobei Monitoring im Sinne von Überwachung durch Messung von Kennzahlen verstanden wird. Hier ist zu diskutieren, welche Kennzahlen zu welcher Normierung zu messen sind. Es sind durchaus wesentliche Unterschiede zwischen Client-Server- und SOA-basierten Lösungen zu erwarten.

3.5.8 Funktion: Technisches Management

Dieser Funktion obliegt es, das für den IT Betrieb erforderliche Wissen zu ermitteln und bereit zu stellen. In dieser Arbeit wird nicht auf die Mittel und Methoden der Erarbeitung und Verwaltung dieses Wissen und dessen Bereitstellung eingegangen. Vielmehr wird gezeigt, wie das erforderliche

Wissen zum Betrieb mit der gewünschten Nutzung der Flexibilität von SOA-basierten Lösungen vereinbar ist.

3.5.9 Funktion: Service Punkt

Der Service Punkt (*Service Desk*) ist der Einstiegspunkt (Single Point of Contact- SPOC)) für alle eingehenden Störfälle. Die Funktion wird unabhängig von jeglicher Lösungsarchitektur benötigt und ist eng mit dem Prozess des Störfallmanagement verknüpft.

3.5.10 Zusammenfassung der Phase Betrieb

Der Betriebsphase ordnet ITIL alle Prozesse des täglichen Betriebs zu, die auch in der Eigentümerschaft den Teams im IT-Betrieb zuzuordnen sind. Wie in den vorangegangenen Phasen beschrieben, gibt es jedoch weitere Aktivitäten als Teil von IT-Prozessen auszuführen, die in ihrer Eigentümerschaft anderen Phase zuzuordnen sind. Als Beispiel sei das Änderungsmanagement genannt, das initial in der Phase der Überführung implementiert wird. Die Ausführung von Änderungen gehört jedoch auch zum täglichen Geschäft des Betriebs. Die der Betriebsphase zugeordneten Prozesse und Funktionen bilden definitiv das Kerngeschäft des täglichen Betriebs.

3.6 Kontinuierliche Service Verbesserung

Der Prozess zur kontinuierlichen Verbesserung der IT-Dienstleistungen (*Continual Service Improvement*) dient der Abstimmung der vereinbarten Dienstleistungen mit den sich womöglich ändernden Geschäftsprozessbedürfnissen. Wesentlicher Bestandteil dieses Prozesses sind geeignete Messkriterien, um die Auswirkungen vorgenommener Anpassungen auch beurteilen zu können. Die Messung und damit das Monitoring sind daher ein wesentlicher Bestandteil dieses Prozesses. Sie bilden die Grundlage des Prozesses, um daraus geeignete Maßnahmen abzuleiten und zu verfolgen. Die kontinuierliche Verbesserung einer Lösung ist ein wesentlicher Initiator von Änderungen und damit eng verbunden mit dem Änderungsmanagement. In dieser Arbeit wird auf den von ITIL beschriebenen Prozess nicht gesondert eingegangen, da er allgemein nicht von der darunter liegenden Lösungsarchitektur beeinflusst ist. Vielmehr ist die Grundlage des Verbesserungsmanagements, das Monitoring und Messen von Status und Kennzahlen sowie die daraus basierenden Schlussfolgerungen zur Identifizierung von Verbesserungspotential von besonderem Interesse.

Im Folgenden wird tabellarisch zusammengefasst, welche Prozesse und Funktionen im Fokus der Untersuchungen zum Betrieb von SOA-basierten Lösungen stehen.

3.7 Zusammenfassung

In den nachfolgenden Tabellen 3.1-3.4 wird je Phase zusammenfassend dargestellt, welche Prozesse bzw. Prozessteile des ITIL hinsichtlich ihrer Änderung und neuen Anforderungen durch die Evolution zu SOA-basierten Lösungen untersucht werden. Die Phase Service Strategie hat planerischen Charakter und wirkt sich nur geringfügig auf den später gelagerten Betrieb einer Lösung aus, wie in Tabelle 3.1 dargestellt.

Prozesse	**Relevanz für regelmäßigen Betrieb**
Service Strategie	-
Finanzmanagement für IT	-
Service Portfolio Management	-

Tabelle 3.1: Übersicht über die zu untersuchenden Betriebsthemen in der Phase Strategie

In der Phase des Designs wirkt sich die Architektur einer Lösung bereits stärker auf den späteren Betrieb aus, wie in Tabelle 3.2 dargestellt. In der konkreten Ausführung der Prozesse ergeben sich neue Anforderungen, die in dieser Arbeit untersucht werden.

Prozesse	**Relevanz für regelmäßigen Betrieb**
Service Leistungsmanagement	Abbildung der Erwartungshaltung des Kunden auf technische, messbare Qualitätskriterien
Service Katalogmanagement	-
Kapazitätsmanagement	Hardware Kapazitätsmessung, -planung und -management
Verfügbarkeits-management	Technische Voraussetzungen von: Verfügbarkeit, Zuverlässigkeit, Wartungsfähigkeit, Servicefähigkeit, Fehlertoleranz und Sicherheit
IT-Service Stabilitätsmanagement	Im Zusammenhang mit Verfügbarkeit
Informationssicherheits-management	-
Zulieferermanagement	-

Tabelle 3.2: Übersicht über die zu untersuchenden Betriebsthemen in der Phase Design

Die Überführung einer Lösung und der zugehörigen Services in die Produktion erfordert die Implementierung weiterer Prozesse. Zwar liegt die Eigentümerschaft dieser Prozesse in der Überführungsphase, doch müssen wesentliche Prozessschritte Tag für Tag im späteren Betrieb der Lösung ausgeführt werden, wie in Tabelle 3.3 dargestellt. Die Ausführung der Prozesse wird von der Lösungsarchitektur beeinflusst und führt zu veränderten Bedingungen mit SOA.

Prozesse	**Relevanz für regelmäßigen Betrieb**
Überführungsplanung und -betreuung	Im Zusammenhang mit der Realisierung von Störfall-, Problem-, Änderungs- und Release- Management
Änderungsmanagement	Übernahme von Änderungen in den produktiven Betrieb
Service Bestands- & Konfigurationsmanagement	Konfigurationsmanagement bezogen auf Services
Release- & Einsatzmanagement Management	Definition von Release
Service Validierung und Testen	Wandlung der Testeinheiten Testumgebung Gestaltung geeigneter Testfälle
Evaluierung	-
Wissensmanagement	-

Tabelle 3.3: Übersicht über die zu untersuchenden Betriebsthemen in der Phase Überführung

Tabelle 3.4 zeigt in der Übersicht die Kernprozesse des täglichen Betriebs, die auch der Betriebsphase in ITIL zugeordnet sind. Zwar bleiben auch hier die Prozesse in ihrer Grundstruktur unabhängig von der Lösungsarchitektur, doch erfordert der Übergang zu einer SOA-basierten Architektur maßgebliche Änderungen in der Umsetzung der beschriebenen Prozessschritte und die dafür verwendeten Werkzeuge.

Prozesse/Funktionen	Relevanz für regelmäßigen Betrieb
Ereignismanagement	Monitoring, Definition geeigneter Kennzahlen und deren Messung und Messmethoden
Störfallmanagement	Analyse des Störfalls und Umgehungsstrategien
Problemmanagement	Ursachenanalyse und Behebung
Anforderungsmanagement	-
Zugriffsmanagement	-
Funktion: Applikationsmanagement	In Bezug zum Wissensmanagement insgesamt
Funktion: IT-Betriebsmanagement	Monitoring zum Zweck der Überwachung und des Messens von Qualität
Funktion: Technisches Management	In Bezug zum Wissensmanagement insgesamt
Funktion: Service Punkt	-

Tabelle 3.4: Übersicht über die zu untersuchenden Betriebsthemen in der Betriebsphase

Wie bereits erwähnt, basiert die kontinuierliche Service Verbesserung maßgeblich auf die im Rahmen des täglichen Betriebs gelieferten Messungen im Rahmen des Monitoring oder Auswertungen aufgetretener Störfälle. Die Untersuchungen in dieser Arbeit beschränken sich daher auf diese Schwerpunkte, da genau diese auch von der Lösungsarchitektur beeinflusst sind.

ITIL beschränkt sich in der Beschreibung der Prozesse, Prozessschritte und Verantwortlichkeiten auf die Beschreibung des Prozessflusses. Wie die Schritte im Detail ausgeführt oder geeignete Werkzeuge aussehen, wird nicht bestimmt. Durch diese Oberflächlichkeit bleibt ITIL insbesondere architekturunabhängig. Der Einfluss der Architektur wirkt sich insbesondere bei der praktischen Umsetzung der von ITIL gelieferten Prozessbeschreibungen aus, wie auch in [Will, 2011] für SOA dargestellt. In dieser Arbeit werden nachfolgend diese Details des Betriebs untersucht und Lösungsvorschläge für die beim Übergang zu SOA auftretenden Probleme unterbreitet.

4 Veränderungen durch SOA

Für die im vorangegangenen Kapitel als wesentlich für den Betrieb einer Lösung erkannten Prozesse und Funktionen soll nun der Einfluss der Lösungsarchitektur auf den Betrieb untersucht werden. Nicht alle Prozesse sind gleichermaßen von der Architektur einer Lösung beeinflusst. Die grundsätzlichen Ziele, Abläufe und Qualitätskriterien für einen IT-Service und Support Prozess bleiben zwar gleich, jedoch zeigen sich die Unterschiede in der Umsetzung und Ausführung der Prozesse, siehe auch [Will und Köppen, 2012] und [Will, 2012]. Je relevanten Prozess werden die Unterschiede beim Betrieb einer Client-Server- und SOA-basierten Lösung dargestellt.

4.1 Störfall- und Problemmanagement

Störfall- und Problemmanagement dienen der Wiederherstellung des normalen Betriebs einer Lösung. Wie bereits dargestellt, ist es dabei die Hauptaufgabe des Störfallmanagements, den Geschäftsbetrieb wiederherzustellen. Dies heißt nicht zwangsweise die Behebung der Ursache des Zwischenfalls. Eine geeignete Umgehung wird ebenso als Lösung im Sinn des Störfallmanagements aufgefasst. Abbildung 4.1 stellt das Störfallmanagement in vereinfachter Form dar. Organisatorische Prozessschritte, wie die Priorität oder Kategorisierung von Störfällen, sind als weitestgehend unabhängig von der technischen Architektur einer Lösung zu betrachten. In einer Client-Server Architektur bedeutet dies zumeist die Wiederherstellung einer technischen Komponente oder Systems. In SOA-basierten Lösungen bedeutet dies die Wiederherstellung der Lösungsfunktionstüchtigkeit, was nicht zwangsweise die Wiederherstellung der Nutzbarkeit eines Services oder technischen Umgebung heißt. Denkbar ist auch der Austausch eines fehlerhaften Services durch einen anderen, in Bezug auf die Lösung gleichwertigen. Voraussetzung ist jedoch, dass die Ursachenanalyse bereits erfolgreich den verursachenden Service ermittelt hat und gleichwertige Services in Bezug auf die Lösung verfügbar sind. Das Störfallmanagement arbeitet eng mit der parallel zum Problemmanagement gepflegten Datenbank für bekannte Fehler zusammen.

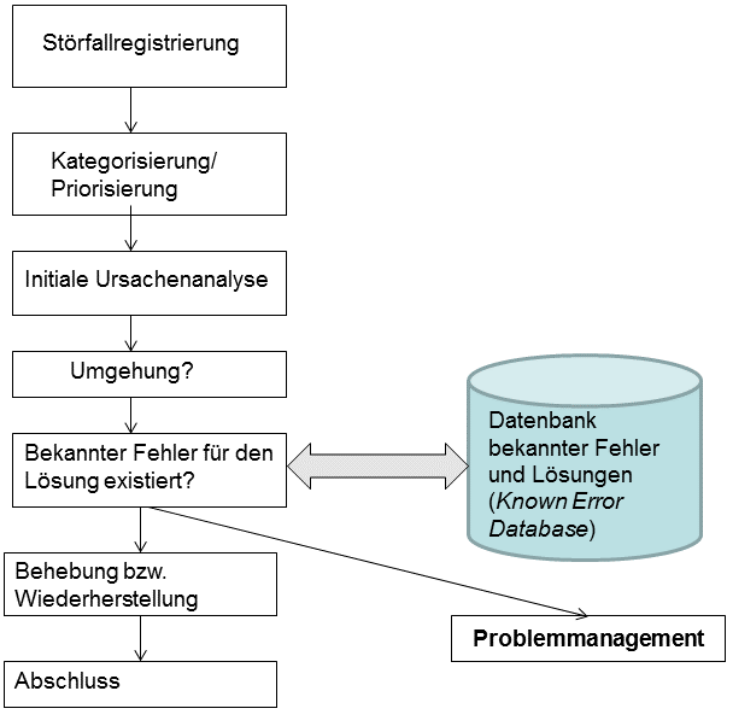

Abbildung 4.1: Vereinfachte Darstellung des Störfallmanagements

Kann das Störfallmanagement die Störung nicht beheben, erfolgt eine Übergabe an das Problemmanagement, dessen grober Ablauf in Abbildung 4.1 dargestellt wird. Neben den üblichen Verwaltungsaufgaben ist insbesondere die eingehende Ursachenanalyse von besonderer Bedeutung innerhalb des Prozesses. Erst wenn die eigentliche Ursache der Störung ermittelt ist, können Änderungen oder Anweisungen zur Vermeidung oder Lösung des Problems in Auftrag gegeben werden. Die Ursachenanalyse stellt somit einen Schwerpunkt in diesem Prozess dar. Im nachfolgenden wird dargestellt, wie sich das Vorgehen und die Anforderungen dabei mit dem Übergang zu SOA-basierten Lösungen ändern.

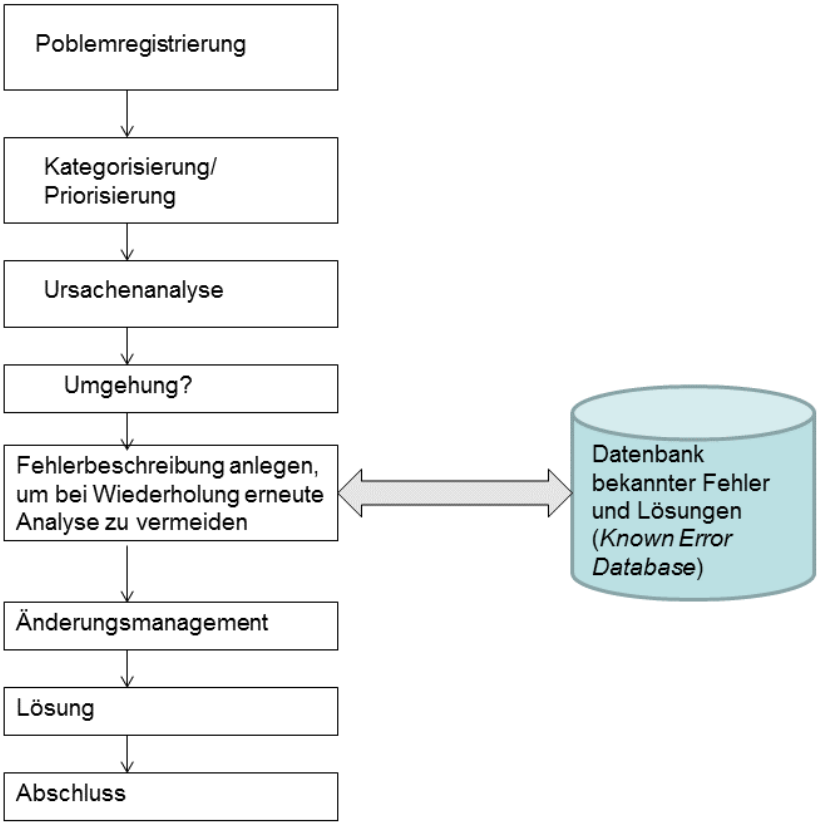

Abbildung 4.2: Vereinfachte Darstellung des Problemmanagements

4.2 Ursachenanalyse

Um dies zu verdeutlichen, wird der Fall betrachtet, dass der Endbenutzer eine Störung seines Geschäftsprozesses feststellt. Dazu betrachte man wiederum den bereits beschriebenen Kundenauftragsprozess. Der Benutzer berichtet über einen Ausfall des Prozesses oder über Probleme beim Anlegen des Kundenauftrags. Er wird nicht in der Lage sein, technische Informationen zur Unterstützung der Ursachenanalyse zu geben. In Client-Server-basierten Lösungen wird mit einer Systemanalyse begonnen, um die verursachende technische Komponente zu ermitteln. Das ist praktikabel, weil die Anzahl der technischen Komponenten in Client-Server-Architekturen eher niedrig ist. In vielen Fällen wird der Benutzer bereits das technische, fehlerhaft arbeitende System benennen. SOA-basierte Lösungen unterscheiden sich hierzu maßgeblich. Dem Benutzer sind der technische Aufbau und insbesondere technische Komponenten und Systeme verborgen. Er kann nicht die verursachende Komponente benennen. Versucht man die

aus Client-Server-basierten Lösungen bekannte systemorientierten Methoden zu übertragen, geht der lösungsorientierte Ansatz von SOA verloren. Würde man dies dennoch versuchen, müsste in einem ersten Schritt analysiert werden, welche technische Komponente die Störung verursacht, bevor überhaupt mit der Analyse der Störungsursache begonnen werden kann. Das führt in der Praxis dazu, dass nach dem Ausschlussverfahren gearbeitet wird. Es werden die Experten der jeweiligen technischen Komponenten herangezogen und jeder analysiert seine technische Komponente, um eine Störung auszuschließen. Je mehr Technologien in der SOA-basierten Lösung eingesetzt werden, desto höher ist die Anzahl benötigter Experten und der Aufwand steigt. Zudem wird man häufige Wechsel der Bearbeiter einer Störung wahrnehmen, da das Ausschlussverfahren häufig nicht zweifelsfrei funktioniert, und das Problem so zwischen den Bearbeitern hin- und hergeschoben wird. Die Störung wird langsam 'eingekreist'. Um dies zu vermeiden, muss die systemorientierte Analyse durch einen lösungs- und serviceorientierten Ansatz mit entsprechenden Werkzeugen zur Analyse abgelöst werden. Derartiges Vorgehen wird nahezu aussichtslos, nimmt man an, dass Services gemäß den Vorgaben auch möglichst dynamisch ausgetauscht werden können. Daher kann man bei einem weiteren Zwischenfall, der den gleichen Prozess betrifft nicht zwangsläufig davon ausgehen, dass die Service-Zusammensetzung des Prozesses sich inzwischen nicht auch geändert hat. Schlussendlich muss ein Prozess, in dem gewählten Beispiel die Bearbeitung des Kundenauftrags, in seiner Gesamtheit erfasst und analysiert werden. Davon ausgehend muss in einem ersten Schritt der für die Störung verantwortliche Service innerhalb des Prozesses ermittelt werden. Nicht das System oder die technische Komponente ist die zu analysierende Einheit, sondern der Service. Dafür werden neue Werkzeuge benötigt.

4.3 Service Level Management

Wie im Abschnitt 3.3 umrissen, besteht die Hauptaufgabe des SLM darin, ein SLA mit dem Kunden (Dienstleistungsnehmer) auszuhandeln, in dem sich die Anforderungen des Kunden an das IT Service Management (ITSM) widerspiegeln, und im Anschluss daran die Erfüllung des SLA auszuwerten. ITIL geht dabei nicht genauer darauf ein, welche Anforderungen ein Kunde haben kann bzw. was in einem SLA enthalten sein sollte. Andere Abhandlungen zum Anforderungsmanagement bleiben oberflächlich und unvollständig im Bereich des ITSM, wie im Volere Requirements Specification Template [Robertson und Robertson, 2006]. Die vorgeschlagenen Qualitätskriterien sind in der Praxis kaum messbar und erlauben daher wenig Rückschluss auf die wirkliche Qualität des Betriebs. Dementsprechend unterschiedlich sehen SLA in der Praxis aus. Unabhängig

von der technischen Architektur einer IT-Lösung, ist der Kunde auf seine Geschäftsprozesse fokussiert. Dementsprechend formuliert er Anforderungen an das IT Service Management, die geschäftsprozessorientiert sind. Bei der Formulierung des SLA-Vertrags werden diese geschäftsprozessorientierten Anforderungen zumeist bereits auf technische Kennzahlen abgebildet. Als Beispiel sei [Barnevik, 2012] genannt, der aus der Praxissicht die in einem SLA erforderlichen Kategorien von Kennzahlen beschreibt:

Verfügbarkeit

Darin werden alle Arten von Verfügbarkeit zusammengefasst von der Verfügbarkeit der Rechner bis zur Verfügbarkeit der Mitarbeiter des Dienstleisters, um Tätigkeiten auszuführen. Dabei wird davon ausgegangen, dass die Verfügbarkeit all dieser technischen und personellen Bausteine gleichzusetzen ist mit der Verfügbarkeit der IT-Lösung.

Antwortzeit

[Barnevik, 2012] formuliert dies als "Zeit zwischen einer Aktion des Anwenders und der Reaktion des Systems inklusive aller Bearbeitungs- und Reaktionszeiten", was durchaus als das bisher gängige Verständnis für die Kennzahl *Antwortzeit* aufgefasst werden kann. Bei dem Übergang zu SOA-basierten Lösungen und der Ausschöpfung der Flexibilität derartiger Architekturen muss auch diese Definition angepasst werden. Die Normierung mit Bezug auf das "System" ist für SOA-basierte Lösungen unzureichend. Vielmehr muss die Antwortzeit zur Basis Service erfasst und auch so gemessen werden.

Messmethode

Kennzahlen können nur dann bewertet werden, wenn auch geeignete Messwerkzeuge und Messmethoden verfügbar sind. Aus dieser Sicht ist hinzuzufügen, dass auch geeignete Metriken und das heißt auch vergleichbare Normierungen erforderlich sind. Andernfalls sind die Messwerte kaum vergleich- und einschätzbar.

Reaktionszeit

Unter der Reaktionszeit wird die Zeitspanne zwischen dem Eingang einer Störungsmeldung bis zu einer ersten Reaktion mit definiertem Umfang bei [Barnevik, 2012] verstanden, was durchaus marktüblich ist. Die eigentliche Schwierigkeit ist die Abbildung der Kundenerwartung auf eine messbare Kennzahl. Dem Kunden geht es natürlich darum, dass die Störung möglichst schnell behoben wird. Eine möglichst schnelle erste Reaktion lässt dagegen lediglich hoffen, dass bereits an der Störung gearbeitet wird und ist absolut nicht aussagekräftig, wann der Störfall behoben sein wird.

Des Weiteren gehören zu einem Vertragswerk wie dem SLA **Sanktionen** für den Fall dass die vereinbarten Leistungen nicht qualitätsgerecht erbracht

werden. **Umfeldbedingungen** beschreiben, welche Leistungen zu erbringen sind und wie überprüft werden können.

[Barnevik, 2012] sieht auch eine sogenannte Flexibilität vor, die sich jedoch im Wesentlichen auf die mögliche Anpassung des SLAs bezieht, wenn sich die Anzahl von Arbeitsplätzen oder die genutzten Rechner ändern.

Ein wichtiger Lieferant für das SLM ist das Monitoring. Die Aufgabe des Monitoring ist die möglichst vollständige Überwachung aller ITSM-Prozesse anhand der Messung festgelegter Kennzahlen und der Einschätzung dieser Messwerte. Obwohl SLA eigentlich die aus Endbenutzersicht erforderlichen Nutzungskriterien in technische Anforderungen und Kennzahlen umsetzen sollen, beinhalten SLA in der Praxis häufig technische Kennzahlen und fordern Messwerte, die sich nur indirekt zu den Benutzeranforderungen in Bezug bringen lassen. Darin spiegelt sich der Klimmzug zwischen technisch Messbarem und tatsächlich Erforderlichem wider. Daher sind übliche SLA häufig Technologie- und daher auch Service-abhängig. Will man jedoch Service- und Technologie-unabhängig sein, muss ein SLA zwangsweise auch Technologie- und Service-unabhängig formuliert werden.

4.4 Monitoring

Der Vorgang der technischen Überwachung und Messung von KPI wird als Monitoring bezeichnet. Das Monitoring leistet gleichzeitig einen wichtigen Beitrag zur Überwachung der Erfüllung des vereinbarten SLA, in dem es technische KPI misst und die Auswertung unterstützt. Im Folgenden werden typische KPI und deren Veränderungen beim Übergang zu SOA untersucht.

4.4.1 Analyse der aktuellen Situation

Das Monitoring ist eine besonders hervorzuhebende Aufgabe innerhalb des IT Service und Support Managements, da es die für die Einschätzung der Qualität der Prozesse notwendigen Messwerte liefert. ITIL ordnet diese Aufgabe dem IT Operations Management zu. Das Monitoring hat jedoch eine übergreifende Funktion innerhalb des IT Service und Support Management. Einerseits dient es der Messung der Qualität um nachzuweisen, dass die im SLA vereinbarten Qualitätskriterien erfüllt werden. Gleichzeitig kommt dem Monitoring eine wichtige Überwachungsfunktion zu. Mit Hilfe des Monitoring soll erkannt werden, wann eine Störung auftritt und umgehend die Analyse und Behebung eingeleitet werden. Idealerweise wird sogar bereits die Entwicklung eines potentiellen Störfalls erkannt und kann so vermieden werden (proaktiv).

Aber um dies zu erreichen, müssen geeignete und messbare Indikatoren definiert werden. Es wird zunächst betrachtet, was geeignete Indikatoren sind, um dann zu diskutieren, wie man sie messen kann.

Für das Bespiel des Kundenauftragsprozesses wäre eine Messung der Performance einzelner Systeme oder technischer Komponenten der Lösung ist jedoch nicht aussagekräftig. Vielmehr muss nun die Performance des gesamten Prozesses zur Basis der einzelnen Services gemessen werden mit der Option, dass ein einzelner Service jeder Zeit gegen einen anderen mit womöglich anderer Technologie ausgetauscht werden kann. Im Gegensatz dazu sind Client-Server-basierte IT-Lösungen in ihrem technischen Aufbau und ihrer Implementierung eher stabil und systemorientiert. Systeme sind nur unter größten Aufwänden austauschbar und nie während der Laufzeit einer Lösung. Dementsprechend sind auch der Betrieb und Werkzeuge, wie z.B. die Monitoring-Werkzeuge stabil und systemorientiert. Um jedoch die Flexibilität der SOA zu erhalten, dürfen Messwerkzeuge weder System- noch Service-abhängig sein. Wäre dies beim Betrieb einer SOA-basierten Lösung der Fall, würde der Austausch eines Services womöglich gleichzeitig den Wechsel, z.B. des Messwerkzeugs, bedeuten. Die Vergleichbarkeit der Messwerte wäre kaum zu gewährleisten. Aber auch die Bedienung des Messwerkzeugs kann stark abweichen. Genau diese Situation findet sich heute häufig in der Praxis, denn obwohl mittlerweile die meisten Software-Hersteller ihre Software als SOA-basiert anpreisen, wurde an den Betriebswerkzeugen nur wenig geändert. Zumeist wird System- anstatt Lösungs- bzw. Service-basiert gemessen und betrieben. Dadurch werden jedoch Rahmenbedingungen geschaffen, die die Flexibilität von SOA-basierten Lösungen ungerechtfertigt einschränken.

Im Betrieb von Client-Server-basierten Lösungen erfolgt das Monitoring in engem Bezug zu den verwendeten Systemen, den technischen Komponenten und der Hardware. Beispiele sind die durchschnittliche Antwortzeit eines Systems, Verfügbarkeit oder Last an einem System oder technischer Komponente. Dieses Vorgehen ist erfolgreich, da meistens nur wenige technische Systeme zum Einsatz kommen, also die überwiegende Anzahl von Geschäftsprozessschritten in einem technischen System ausgeführt werden. Somit ist beispielsweise die durchschnittliche Antwortzeit des Systems gleichzeitig ein Indikator für die Performance eines einzelnen Geschäftsprozesses. Zudem ist die technische Infrastruktur weitestgehend stabil. Client-Server-basierte Lösungen werden nicht konzipiert, um bei Bedarf Technische Komponenten oder gar Systeme auszutauschen. Anders dagegen SOA-basierte Lösungen. Die einen Geschäftsprozess bildenden Services sind lediglich lose gekoppelt. Bei Bedarf können sie ausgetauscht werden. Je mehr Services auf Basis verschiedener Technologien einen Geschäftsprozess bilden, desto weniger Wert hat ein Monitoring auf Basis von Systemen und technischen Komponenten. Die Kennzahlen sind nicht länger System-gebunden, sondern müssen in Relation zu den Services gemessen werden, wie in Abbildung 4.2 dargestellt. Die Erwartungshaltung des Kunden richtet sich auf den Geschäftsprozess aus. Es ist die Antwortzeit

des Kundenauftragsprozesses im Ganzen von Interesse. Sie setzt sich aus den Antwortzeiten der beteiligten Services zusammen.

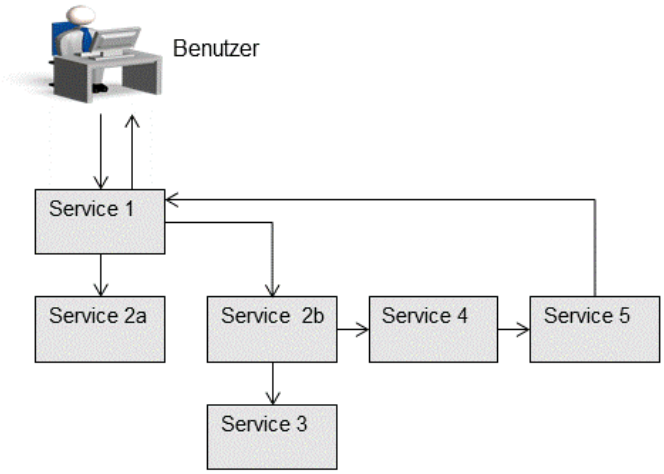

Abbildung 4.3: Verknüpfung der Services zur Laufzeit

Wichtige messbare Kategorien von KPI beim Monitoring von IT-Lösungen sind:

- Verfügbarkeit

  Jeder Kunde erwartet die Verfügbarkeit seiner IT-basierten Geschäftsprozesse während seiner Arbeitszeit. Dabei wird Verfügbarkeit mit Nutzbarkeit und auch Performance verbunden. In Client-Server-basierten Lösungen wird dies häufig auf die technische Verfügbarkeit einzelner Systeme projiziert. In SOA-basierten Lösungen muss dies jedoch auf die Verfügbarkeit der beteiligten Services abgebildet werden.

- Performance

  Die Kategorie Performance meint letztlich die vom Benutzer des Geschäftsprozesses empfundene Antwortzeit des Geschäftsprozesses. Daraus werden in der Client-Server-Welt zumeist KPI auf der Basis von Systemen und technischen Komponenten gemessen. Bei SOA-basierten Lösungen muss dies grundlegend auf der Performance von Services basieren. Die Messung muss dem Ablauf der aktuell verknüpften Services in einem Geschäftsprozess folgen.

- Durchsatz

  Unter Durchsatz werden die KPI zusammengefasst, die die prozessierte Menge widerspiegeln. In Client-Server-basierten Metriken erfolgt die Messung zur Basis technischer Objekte, wie zum Beispiel die Anzahl von Datensätzen innerhalb von Tabellen und letztendlich in Bytes. In

SOA geht es dagegen um den Durchsatz von Business Objekten je Service. Entscheidend ist z.B. die Anzahl von Kundenaufträgen, die prozessiert wird, die Menge von Verfügbarkeitsanfragen oder die Menge von Lieferscheinen, die gedruckt werden. Das heißt nicht, dass die Menge der prozessierten Bytes völlig irrelevant wird, doch muss die Einschätzung der Menge zur Basis Service oder Prozess erfolgen.

- Abbrüche (Exception)

  Eine auftretende Ausnahmesituation oder ein Abbruch innerhalb eines Services kann zu einer Verkettung von Aktionen und weiteren Abbrüchen führen. Um die initiale Ursache zu finden, ist die zeitliche und ursächliche Verknüpfung der Abbrüche zu überwachen. Dazu ist die Zuordnung der Abbrüche je Prozess und Prozessschritt in der Verknüpfung erforderlich. In Client-Server-basierten Systemen reicht es dagegen, Abbrüche innerhalb der Systeme zu überwachen, da sie so bereits den Prozesse zuzuordnen sind.

- Sicherheit

  Der Bereich Sicherheit ist ein sehr komplexes Thema, in dem zusätzlich auch Compliance Themen wie Sarbanes-Oxley-Act oder industriespezifische Verordnungen zum Tragen kommen. Die in der SOA-Landschaft bereitgestellten Services können bei Bedarf zum Einsatz kommen, jedoch muss überwacht werden, dass Services nur berechtigt verwendet werden. Unberechtigte Zugriffe müssen vermieden werden. Sicherheitsrelevante Aktionen müssen nachvollziehbar protokolliert werden. Auf Grund der Komplexität und in gewisser Weise auch Eigenständigkeit des Themas Sicherheit wird es in dieser Arbeit nicht gesondert betrachtet.

Tabelle 4.1 stellt die wichtigsten Kategorien technischer KPI dar und beschreibt, zu welcher Basis das KPI in SOA-basierten Lösungen gemessen werden muss.

Kategorie	KPI	Kriterium
Verfüg-barkeit	Herzschlag des Services, Prozessschritts, Prozess	Ausbleibender Herzschlag zeigt einen Störfall des Services an. Idealerweise ist ein vergleichbarer Service in der Landschaft verfügbar und kann eingetauscht werden.
Performance	Durchschnittliche Antwortzeit eines Services	Den Schwellenwert für Reaktionen stellt der vom Kunden formulierte Erwartungswert dar. Starke Abweichungen zeigen Handlungsbedarf an.
	Antwortzeit eines Geschäftsprozesses und -schritte in Bezug auf genutzte Services	Der Endbenutzer wird zumeist nur Erwartungen hinsichtlich des Geschäftsprozesses und einzelner -schritte haben. Daher sind KPI in Relation zu diesen Erwartungen erforderlich.
	Antwortzeiten der Services in den Schichten außerhalb der Prozessschicht relativ zu den benutzten Services des Prozessschicht	Schwellenwerte sollten die Werte sein, die für die nutzenden Services und den dazu in Bezug stehenden Prozess (-schritten) kritisch sind.
Durchsatz	Anzahl prozessierter Prozessobjekte	Sind auf Basis der Erwartung des Kunden und möglicherweise auf Basis historischer Erfahrungswerte formulierbar
	Hardware-Auslastung (CPU, I/O, Memory, Disks etc.) per Prozess(-schritt), und darin involvierte Services	Schwellenwerte für einen Alarm sind die für die Hardware kritischen Werte. Für die Ursachenanalyse sind diese Messwerte aber Service- und Prozess-bezogen erforderlich.
	Überlauf bzw. Wartesituationen bei der Prozessierung der Geschäftsobjekte	Der Datentransfer zwischen den Services erfolgt mittels des Service Bus. Warteschlangen zeigen Engpässe am Service Bus oder in der Leistungs-fähigkeit eines Services und müssen gelöst werden.

Tabelle 4.1: Darstellung von KPI für SOA-basierte Lösungen

Kategorie	KPI	Kriterium
Abbrüche (Exceptions)	Jeder Abbruch (Exception) in Bezug auf Service und Prozess(-schritt)	Es werden klare Zusammenhänge zwischen Exception und Services sowie Prozessen benötigt. Ein initialer Abbruch kann beliebig viele weitere Störfälle verursachen. Daher sind die Reihenfolge und die Zusammenhänge der Exceptions erforderlich.
Sicherheit	Unberechtigte und fehlgeschlagene Zugriffe auf Services oder Prozesse	Auslösung von Alarm bzw. Verhinderung des Zugriffs
	sicherheits-gefährdende Aktionen	Aufzeichnung in nachvollziehbaren Protokollen
	Benutzer mit sicherheits-kritischen Berechtigungen	Überwachung der Aktionen solcher Benutzer

Fortsetzung Tabelle 4.1: Darstellung von KPI für SOA-basierte Lösungen

Die in Tabelle 4.1 beschriebenen KPI beziehen sich auf die Services in der Prozessschicht und widerspiegeln am nächsten die Erwartungshaltung der Benutzer. Wie Abbildung 4.4 zeigt, nutzen Services in dieser Schicht jedoch Services in der darunter liegenden Schicht und stehen im Austausch zu Service Repository und Service Bus.

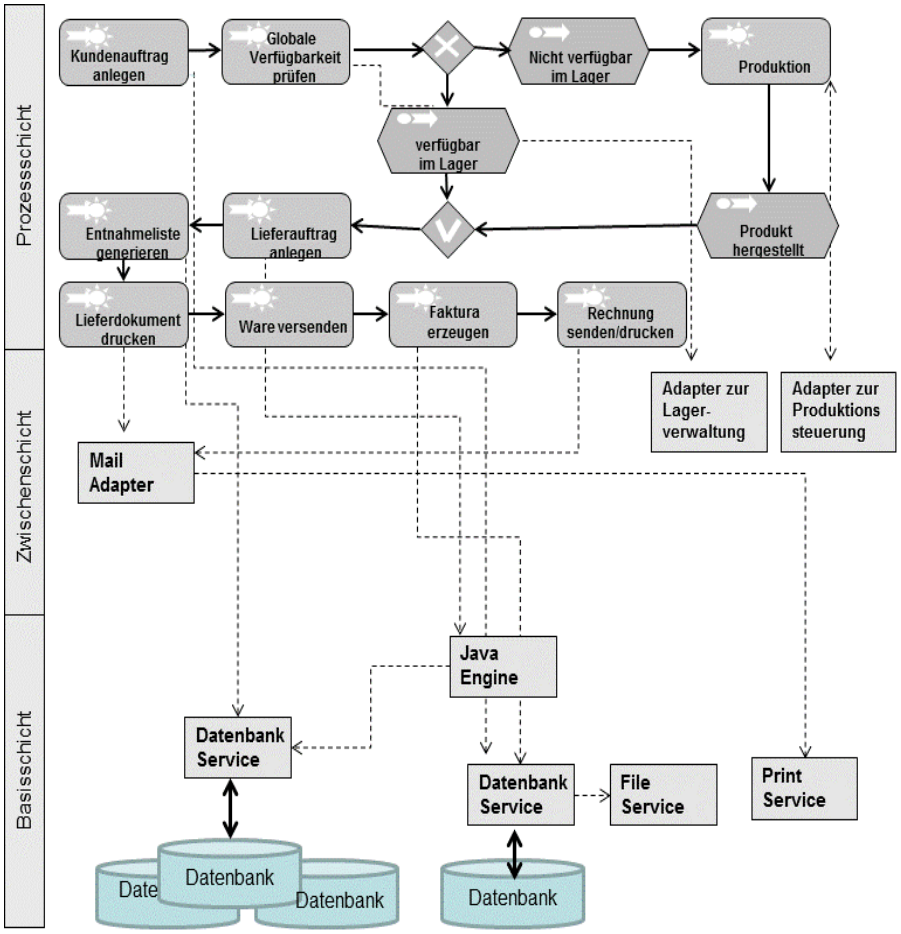

Abbildung 4.4: Zusammenspiel der Services in den verschiedenen Schichten

Somit setzen sich die KPI in Tabelle 4.1 wiederum aus den Messwerten der genutzten darunter liegenden Services zusammen. Als Beispiel sei Performance genannt. Für den Benutzer zählt lediglich die Performance der Geschäftsprozesse. Diese setzt sich jedoch aus der Performance der daran beteiligten Services und der für die Kommunikation und den Datenaustausch notwendigen Zeit zusammen. Damit ergibt sich für die Zusammensetzung der Performance eines Geschäftsprozesses ein Baum, in dessen Blättern die Services der Basisschicht stehen. Abbildung 4.4 zeigt dies für den Service *Lieferauftrag anlegen*. Die Antwortzeit dieses Services setzt sich im Beispiel aus den Antwortzeiten der genutzten Services *Java Engine* und *Datenbankservice A* sowie *Datenbankservice B* plus der Kommunikationszeit der Services zusammen.

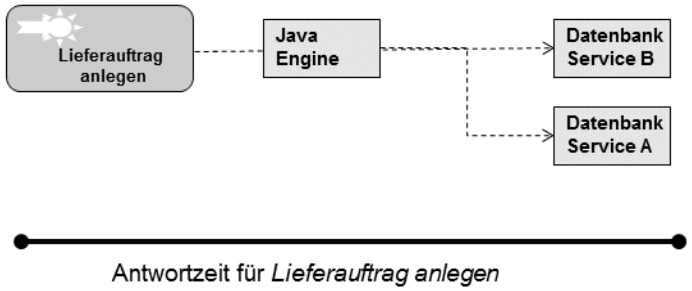

Abbildung 4.5: Zusammensetzung der Performance eines Services

Bei einer Performanceanalyse muss entlang diesem Baum die Ursache der unzureichenden Performance ermittelt werden. Die Performance ergibt sich als Summe der einzelnen Messwerte (Laufzeit der Services und Kommunikation). Abbildung 4.5 stellt zudem eine Vereinfachung dar. Es ist davon auszugehen, dass Services nicht direkt Daten untereinander austauschen, sondern vielmehr mit Hilfe des Service Bus, um die bereits beschriebene Unabhängigkeit der Datenaustauschformate zu zentralisieren. Somit ergibt sich die Laufzeit oder Performance als Summe aller Laufzeiten der Services, die direkt dem Prozessablauf dienen als auch solchen, die der Verwaltung von Metadaten oder dem Datenaustausch, also Service Repository oder Service Bus dienen. Nimmt man an, dass ein Service innerhalb eines Prozesses gegen einen anderen ausgetauscht werden soll, darf dies nicht die Performance-Messung als solche beeinträchtigen, da dies eine maßgebliche Einschränkung der Betreibbarkeit bedeuten würde.

Einen analogen Ansatz für eine Performance Matrix schlagen auch [Her et al., 2006] vor. Es wird im Detail untersucht, welche Kennzahlen gemessen werden müssten, um über die Performance und deren Zusammensetzung in SOA-basierten Prozessen darzustellen. Für andere Kategorien wäre dies ebenso aufzubauen. Mit welchen Methoden und unter Nutzung welcher Bausteine einer SOA man diese Messungen jedoch durchführen kann, bleibt offen. Während an dieser Stelle die Analyse aus Betriebssicht durchgeführt wurde, untersucht die Arbeit [Jeng, 2006] dagegen den direkten Zusammenhang und das Potential der Ermittlung von Antwortzeiten der Geschäftsprozesse und ihrer Schritte aus den IT-Abläufen. Ziel ist es, Schwachpunkte in den Abläufen der Geschäftsprozesse wie der Produktion zu ermitteln und zu beheben. In dieser Arbeit wird dagegen die Kategorie 'Antwortzeit' direkt auf die IT-Prozesse bezogen. Nichtsdestotrotz gelangt auch [Jeng, 2006] zu der Erkenntnis, dass Agenten insbesondere die Performance-Daten der Geschäftsprozessabläufe in direktem Zusammenspiel mit den IT- Services ermitteln müssen. [Franklin und Graesser, 1997] liefert eine zum damaligen Zeitpunkt umfassende Darstellung des Verständnisses von Agenten. Ohne auf die Feinheiten einzugehen, handelt es sich ín der

Quintessenz bei einem Agenten um ein autonomes Programm zur Analyse und Vermessung von Systemen. Autonom beschreibt dabei eine Unabhängigkeit von Zustandsänderungen, von denen der Agent unbeeinträchtigt seinen Aufgaben nachgeht. Eine wesentliche Änderung im Zuge des Übergangs zu SOA-basierten Lösungen ist also der Bezug von Agenten auf Services an Stelle von Systemen. Gleichzeitig müssen derartige Agenten Service-unabhängig arbeiten.

Auf Grund des anderen Fokus bleiben bei [Jeng, 2006] die Austauschbarkeit von Services und der Einfluss dessen auf die Agenten unberücksichtigt. Die Agententechnologie ist auch weiterhin eine gängige Technologie bei den aktuell im Markt verfügbaren Monitoring-Werkzeugen. Allerdings werden die Monitoring-Agenten üblicherweise für bekannte und verbreitete, das heißt in aller Regel für Services mit hoher Marktdurchdringung entwickelt. Insofern sind diese Agenten keineswegs autonom gegenüber verschiedenen Technologien, sondern vielmehr sogar Service-abhängig. Auch wenn dies an dieser Stelle nicht für alle KPI-Kategorien bis ins Detail beschrieben werden kann, verdeutlicht das Bespiel Performance, dass die Überwachung von SOA Service-übergreifend und gleichzeitig Service-unabhängig von Technologie und sonstigen funktionalen Eigenschaften erfolgen muss. Andernfalls würde dies die Integration der Messwerte behindern, sowie der gewünschten Flexibilität von SOA auf Basis des Austauschs von Services entgegen steht.

4.4.2 Marktsituation

An dieser Stelle sollen einige der wichtigsten und am weitesten verbreiteten Monitoring-Werkzeuge sowie deren Stärken und Schwächen dargestellt werden.

CA Wily Introscope

Dieses Werkzeug der Firma Computer Associates (Home Page http://www.ca.com) ist auf das Monitoring von Anwendungen auf der technologischen Basis von Java und Microsoft .NET ausgerichtet. Für jeden Service muss eine eigene Instrumentierung verfügbar sein. Unter Instrumentierung wird die spezifische Konfiguration verstanden, um eine vorwiegend technische Komponente zu überwachen und Beschreibungen von Ausnahmesituation zu erfassen sowie ein komponentenspezifisches Protokollierung zu ermöglichen. CA Wily Introscope unterstützt zudem die Performance Messung aus Benutzersicht mittels der Erstellung von Skripten, die die Simulation typischer Benutzeraktivitäten simulieren und deren Laufzeiten dabei gemessen werden.
Dieses Werkzeug ist definitiv Technologie-abhängig. Services können nur überwacht und vermessen werden, wenn eine entsprechende Instrumentierung zur Verfügung steht. Teilweise wird diese Instrumentierung von den Nutzern erstellt. Für Services oder insbesondere

technische Komponenten wird teilweise auch vom Hersteller des Services oder insbesondere technischer Komponenten eine entsprechende Instrumentierung mitgeliefert. Dabei heißt Instrumentierung nur, dass die gewünschten Daten gesammelt werden. Die Vergleichbarkeit der Messungen für unterschiedliche Services ist nicht gewährleistet, wodurch die Austauschbarkeit von Services behindert wird. Die Auswertung der gemessenen Daten erfordert zudem servicespezifisches Know How. Um dies zu ändern, empfiehlt sich die Service-übergreifende Normierung von Messwerten oder jeder Service liefert eine Matrix zur Bewertung der für ihn gemessenen Werte mit.

HP OpenView

Die Produktfamilie HP OpenView der Firma HP (Home Page http://welcome.hp.com) unterstützt mit speziellen Angeboten unter dem Namen Performance Agent und Manager die Überwachung verschiedener Systeme unterschiedlicher Technologien. Dabei bereitet die Firma HP entsprechende Schnittstellen für die am weitesten verbreiteten oder namhaftesten Systeme andere Hersteller vor, wie SAP, IBM, Microsoft SQL Server oder Oracle. Der Schwerpunkt der Überwachung liegt auf sehr Technologie-nahen Kennzahlen, wie genutzter Speicherplatz, I/O-Durchsatz oder CPU-Auslastung. Die Bewertung der Messungen obliegt dem Nutzer, wodurch vom Nutzer entsprechendes Know How zur Interpretation der Werte in Bezug auf die eingesetzte Technologie erfordert wird. Anwendung findet HP OpenView damit insbesondere bei Hardware-Betreibern.

Ähnliche Konzepte und ein vergleichbares Leistungsspektrum verfolgen IBM (Home Page http://www.ibm.com/software/tivoli) mit der Produktgruppe Tivoli oder BMC (Home Page http://www.bmc.com) mit Patrol. Allen gemein ist die System- und Technologie-orientierte Ausrichtung. Der Anwender ist in jedem Fall davon abhängig, inwiefern der Hersteller des Überwachungswerkzeugs eine entsprechende Schnittstelle ausliefert oder ob umgekehrt der Hersteller des Systems, Technologie oder Services eine Schnittstelle für den Anschluss an das Monitoring-Werkzeug bereitstellt. Betrachtet man diese Situation aus Sicht SOA-basierter Lösungen und dem Anspruch, Services unterschiedlicher Technologien austauschen zu können, ergeben sich die bereits beschriebenen Hindernisse.

Integrierte Administrationswerkzeuge in Produkten von SAP, IBM oder Oracle

Neben den beschriebenen Werkzeugen der Technologie-Hersteller bieten auch die Hersteller von Applikationen Werkzeuge an. Üblicherweise liefert jeder namhafte Hersteller von Applikationen mit der Software auch eigene, integrierte Monitoring- oder auch Administrationswerkzeuge aus. Diese Werkzeuge sind meist sehr effektiv, aber auch im höchsten Maße herstellerspezifisch. Ein Austausch eines Systems oder Services erfordert

damit das Erlernen komplett neuer, herstellerspezifischer Werkzeuge. Die Integration derartiger Werkzeuge zwischen verschiedenen Herstellern ist nicht gewährleistet. SOA-basierte Lösungen, die Services verschiedener Hersteller nutzen, sind somit auf Basis derartiger Werkzeuge nicht integriert zu überwachen oder administrieren.

SAP Solution Manager

Mit diesem Werkzeug verspricht die SAP die Realisierung des übergreifenden und integrierten Applikations Lebenszyklus Managements über verschiedene SAP Lösungen und Technologien hinweg und der Integration von Nicht-SAP-Technologien [Schäfer und Melich, 2012]. Dazu werden die in den SAP-Lösungen auf Basis des SAP Web Application Servers ABAP enthaltenden Überwachungs- und Administrationswerkzeuge eingebunden. Für Lösungen auf Basis der Java-Technologie wird der von SAP eingesetzte CA Wily Introscope genutzt und mit entsprechender Instrumentierung geliefert. Prinzipiell können so auch Services anderer Hersteller eingebunden werden, jedoch ist der Anwender darauf angewiesen, dass SAP entsprechende Schnittstellen entwickelt und ausliefert. Dies schränkt die Integrationsfähigkeit des SAP Solution Managers ein. Zudem ist auch hier die Normierung der Messwerte über verschiedene Technologien und auch Services hinweg nicht gewährleistet. Der beliebige Austausch von gleichwertigen Services hinsichtlich einer Lösung wird damit behindert, wenn nicht gar unmöglich. Der SAP Solution Manager ist nicht in der Lage, die zur Laufzeit einer Lösung genutzte Service-Zusammensetzung automatisch zu erkennen und die Überwachung darauf auszurichten. Vielmehr ist der SAP Solution Manager auch weiterhin auf Systeme und technische Komponenten ausgerichtet, die zur Vorbereitung der Nutzung des SAP Solution Managers einzeln angeschlossen werden müssen, siehe auch [Teuber et al., 2013]. Der Anschluss bedeutet dabei, dass:

- sogenannte Datensammler in Form von Agenten auf den technischen Komponenten, Systemen bzw. Servern installiert,
- Verbindungen auf Systemebene vom SAP Solution Manager zu den Systemen konfiguriert,
- Systemspezifische Konfigurationen im SAP Solution Manager vorgenommen werden müssen.

Neben dem rein technischen Monitoring bietet der SAP Solution Manager ein Business Process Monitoring, wie in Abbildung 4.6 dargestellt.

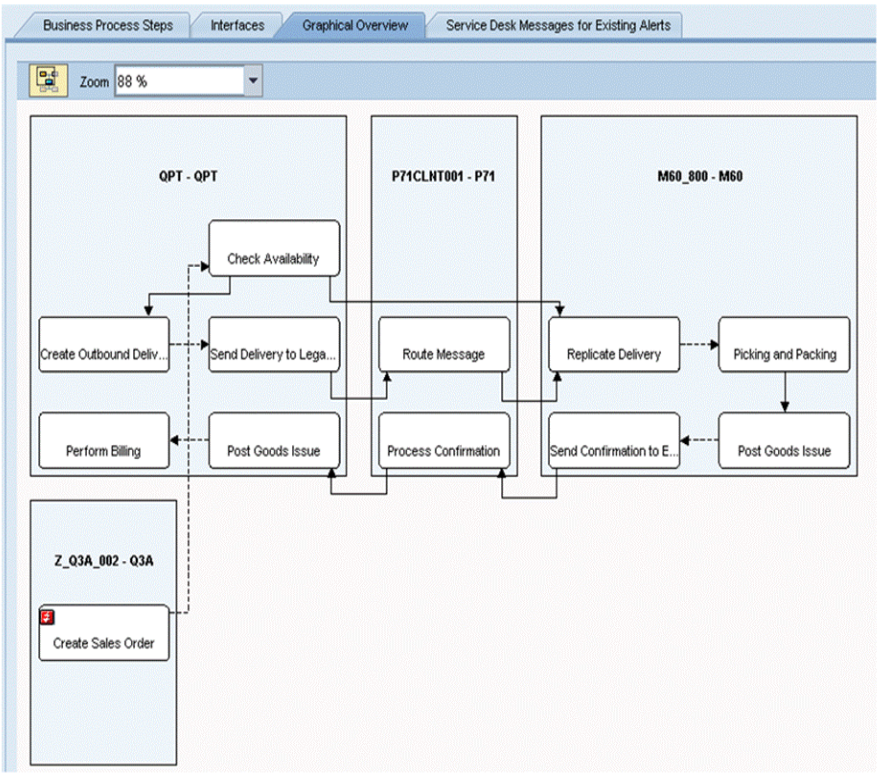

Abbildung 4.6: Bildschirmabzug des Business Process Monitoring im SAP Solution Manager für das Beispiel des Kundenauftragsprozesses

Die einzelnen Prozessschritte werden den technischen Systemen zugeordnet. In jedem Prozessschritt können die SAP-spezifischen Kennzahlen gemessen und kundenspezifisch bewertet werden. In Abbildung 4.6 wird für den Schritt *Create Sales Order* ein roter Alarm signalisiert. Durch entsprechende Doppelklicks auf dieses Element kann zu den Details und den Analysewerkzeugen verzweigt werden. Das Business Process Monitoring nutzt dabei die Messwerte des technischen Monitoring und ordnet sie den Prozessschritten zu. Die Prozessschritte selbst sind Systemen zugeordnet und werden nicht wie eigenständige Services behandelt, was sie in SAP-Systemen auch nicht sind. Der Austausch von Services in der Prozessschicht ist so nicht vorgesehen.

4.4.3 Zusammenfassung

Das Monitoring ist das wesentliche Instrument zur Überwachung der Funktionstüchtigkeit und Leistungsfähigkeit einer Lösung. Erst wenn eine Lösung in ihrer Gesamtheit überwacht wird, sind die Messwerte in Bezug auf die Lösung aussagekräftig. Es wurde gezeigt, dass das Monitoring in

SOA zur Basis der Services und der daraus gebildeten Geschäftsprozesse arbeiten muss. Um das Flexibilitätspotential von SOA-basierten Lösungen zu erhalten, muss ein solches Werkzeug unabhängig vom einzelnen Service arbeiten. Der Austausch von Services darf die Funktionsfähigkeit des Werkzeugs nicht beeinträchtigen. Daraus lassen sich folgende Schlussfolgerungen ziehen:

1. Ein solches Monitoring-Werkzeug muss zentral arbeiten.
2. Sowohl die Messung der Daten als auch die Bewertung der Messungen müssen standardisiert sein.

Nur wenn alle Messungen und Bewertungen in Bezug auf Services und die daraus zusammengesetzten Prozesse an zentraler Stelle zusammenfließen, kann eine einheitliche, konsolidierte Sicht auf die Qualität und Quantität der SOA-basierten Lösung und ihren Betrieb gewährleistet werden. Ein solches Werkzeug muss unabhängig von den Eigenschaften, insbesondere von der verwendeten Technologie eines Services sein. Andernfalls würde dies die Möglichkeit des Austauschs von Services im Bedarfsfall einschränken. Daraus folgt zwingend, dass ein solches Werkzeug nur auf Basis einer gewissen Standardisierung arbeiten kann.

4.5 Änderungsmanagement

Der Änderungsmanagementprozess wird in seiner Gesamtheit aus ITIL-Sicht zwar der Phase Transition zugeordnet, doch gehört das Änderungsmanagement zu den regelmäßigen, üblichen Aufgaben bei dem Betrieb einer Lösung. Für die Notwendigkeit einer Änderung einer Lösung gibt es zwei grundsätzliche Ursachen:

1. eine geplante Anpassung der Lösung auf Grund von Veränderungen des Geschäftsprozesses

2. eine ungeplante Änderung auf Grund eines aufgetretenen Problems mit der Lösung

4.5.1 Analyse der Anforderungen

Um die Unterschiede zwischen Client-Server- und SOA-basierten Lösungen darzustellen, wird wiederum auf das Beispiel Kundenauftragsprozess zurückgegriffen. Man nehme an, dass der Kundenauftragsprozess erweitert werden soll. Im Fall der Nichtverfügbarkeit eines bestellten Produkts soll nicht zwangsläufig die eigene Produktion angestoßen werden, sondern nur im Fall eines Produkts namens A. Ein Produkt B soll dagegen von einem Drittanbieter eingekauft werden. Abbildung 4.7 zeigt, wie der Kundenauftragsprozess erweitert werden soll. In dem bisherigen Kundenauftragsprozess sollte für ein nicht im Lager verfügbares Produkt die Produktion angestoßen werden. Nun soll dies für das Produkt B geändert werden. Produkt B soll von entsprechenden Zulieferern eingekauft werden

an Stelle der eigenen Produktion. Dabei handelt es sich um eine geplante Änderung. In einer Client-Server-basierten Umgebung würde die Erweiterung eines bestehenden Geschäftsprozesses das Design, die Implementierung etc. der neuen Funktionalität entweder in einem der bereits eingesetzten Systeme erfordern oder im ungünstigsten Fall die Einbindung eines neuen Systems, in dem die neue Funktionalität zu entwickeln oder zu nutzen wäre. Durch den SOA-Ansatz werde angenommen, ein entsprechender Service werde durch den Drittanbieter des Produkts selbst angeboten und kann in die bestehende Lösung integriert werden. Selbstverständlich ist die Welt nicht nur schwarz-weiß und der Übergang von der Client-Server-Welt zu SOA ist fließend.

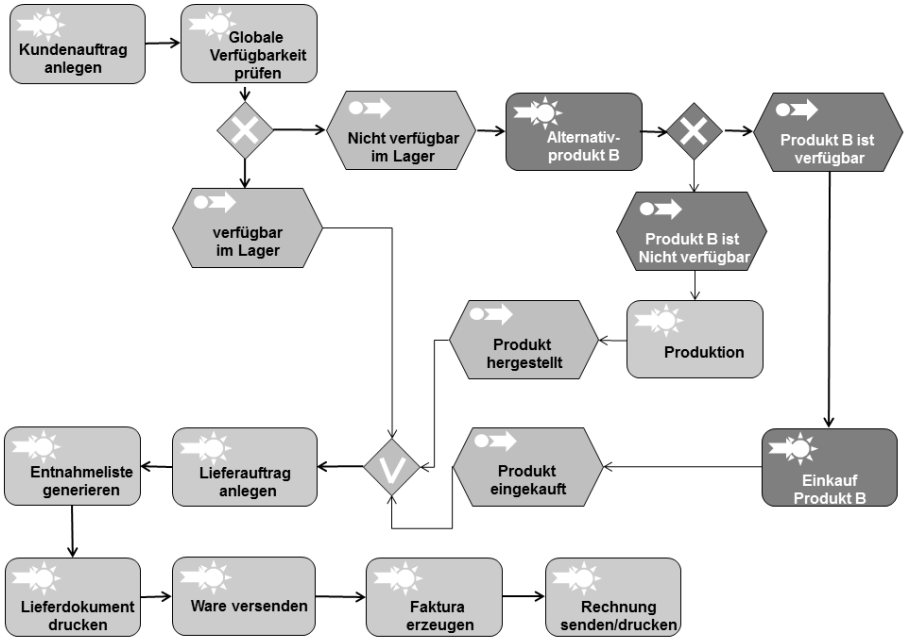

Abbildung 4.7: Geplante Erweiterung des Kundenauftragsprozesses

Daher ist auch eine Mischung von Client-Server-basiertem Ansatz und SOA-Elementen denkbar. Man fokussiere daher auf die Integration eines neuen Services in eine existierende Lösung. Man nehme des Weiteren an, dass dieser neue Service auf einer anderen Technologie basiert, die bisher nicht in der existierenden Lösung zum Einsatz kam. Bisher gibt es keine allgemeinen Regeln, wie die Freigabe (*Deployment*) eines neuen Services zu erfolgen hat. Daher hat jeder Service-Anbieter eigene Deployment Werkzeuge, die zudem abhängig sind von der verwendeten Technologie. In dem gewählten Beispiel kann der neue Service in die existierende Lösung eingebunden werden, doch vermutlich mit Hilfe des ihm eigenen, mitgelieferten Deployment-

Werkzeugs. Zum Zeitpunkt der erstmaligen Integration des neuen Services beschränkt sich im günstigsten Fall die Herausforderung auf die Bedienung des neuen Deployment Werkzeugs. Das Change Management steht jedoch in engen Zusammenhang mit dem täglichen Betrieb einer Lösung. Änderungen gehören zum Tagesgeschäft in einer produktiven Umgebung. Damit ergeben sich grundsätzliche Probleme, die neue Lösung in ihrer Gesamtheit regelmäßig zu betreiben:

Für das neue Deployment-Werkzeug ist neues Know How zur Nutzung erforderlich. Durch unterschiedliche, nicht integrierte Deployment-Werkzeuge können verknüpfte Änderungen nicht synchronisiert werden. Es existiert kein übergreifendes Änderungsinformationssystem. Unter den beschriebenen Umständen ergeben sich für den Betrieb einer SOA-basierten Lösung folgende grundsätzliche Fragen:

- Es kommt zu einer Störung. Wurde etwas an der Lösung geändert, was die Ursache der Störung sein könnte?
- Wie können abhängige Änderungen an Services mit unterschiedlichen Technologien und Deployment-Werkzeugen synchronisiert werden?
- Wie kann man eine zentrale Änderungshistorie in Bezug auf eine Lösung pflegen und bei Bedarf auswerten?
- Wie kann eine neue Service-Technologie bzw. Deployment Werkzeug in das tägliche Geschäft integriert werden, wobei das dafür erforderliche Wissen ein wesentlicher Faktor ist?

Unter der Voraussetzung, eine möglichst hohe Flexibilität einer Lösung auf Basis austausch- und veränderbarer Services zu bewahren und nutzen zu können, kann die Schlussfolgerung zur Beantwortung dieser Fragen nur lauten, dass für das Change Management in SOA-basierten Lösungen ein Deployment-Werkzeug mit neuen Eigenschaften benötigt wird. Es ist eine Synchronisation der einzelnen Änderung und eine serviceübergreifende Protokollierung erforderlich. Es ergeben sich drei wesentliche, neue Anforderungen im SOA-Umfeld:

- zentrale Ausführung, Kontrolle und Auswertung des Change Management Prozesses (lösungsorientierte Protokollierung),
- lösungsorientiertes Deployment von Änderungen (Synchronisation von Changes über Services hinweg) und
- Deployment von Änderungen unabhängig von der eigentlichen Technologie eines Services.

Bisher ist es noch immer üblich, dass technische Systeme und auch Services mit einem eigenen und häufig herstellerspezifischen Deployment Werkzeug einhergehen. Dadurch ist es kaum möglich, komplexe Änderungen zu

realisieren, die die Synchronisation der darin enthaltenden Änderungen an mehr als einem Service von verschiedenen Hersteller erfordern.

4.5.2 Marktsituation

In der Praxis wurde der Bedarf neuer Werkzeuge zum Teil erkannt. Einige Hersteller von Softwarebausteinen haben für SOA-basierte Lösungen eigene Konzepte entwickelt, wie z.B. mit den Plattformen HP OpenView und IBM Tivoli oder SAP Solution Manager 7.x. Allen gemein ist der Ansatz, verschiedene technische Komponenten oder Services zentral zu überwachen und zu administrieren. Allerdings wird dazu jeweils ein spezielles Interface entwickelt. Somit können eine andere Technologie oder Service nur eingebunden werden, wenn zuvor ein Interface entwickelt wurde. Jedes Werkzeug hat seine Stärken, insbesondere beim Management der herstellereigenen Produkte. Bisher wird jedoch ein eher reaktiver Ansatz verfolgt. Erst wenn eine Komponente oder ein Service eingesetzt werden oder eine hinreichend große Verbreitung zu erwarten ist, werden Schnittstellen für das Deployment entwickelt. Dieses Vorgehen ist dabei durchaus begründbar, denn üblicherweise folgt man aus Kostengründen dem Marktdruck.

Beispiel SAP Solution Manager

Anhand des SAP Solution Managers [Schäfer und Melich, 2012] wird gezeigt, wo Vorteile und Grenzen dieses Vorgehens liegen. SAP Solution Manager verfolgt einen lösungs- und lebenszyklusübergreifenden Ansatz. Für SAP gestaltet sich die Situation insofern weniger kompliziert, da die meisten Lösungen auf der ABAP- oder Java-Technologie beruhen und in diesen die gleiche Deployment Technik genutzt werden kann. Mit der Einführung der ABAP-Technologie wurde gleichzeitig eine systemorientierte Change Management Methodik mit entsprechenden Werkzeugen (Change and Transport Systems- CTS) aufgebaut. Im Lauf der Entwicklung wurde diese Technologie verbessert und weiter ausgebaut, so dass schließlich das Change Management für eine Landschaft aus mehreren SAP-Systemen auf Basis der ABAP Technologie zentral durchgeführt werden konnte. Die Einführung der Java-basierten Komponenten durchbrach diese Homogenität, da die neue Technik mit eigenen Deployment-Techniken und Entwicklungswerkzeugen für die Umsetzung von Änderungen einherging. Da es sich jedoch "nur" um eine Technologie handelte, verfolgte und verfolgt SAP die Methode der Integration in die bereits vorhandene ABAP-Change Management Methodik (Change and Transport System Plus (CTS+)). Inzwischen ist auch diese Technologie-Basis durchbrochen und erweitert worden durch z.B. den Zukauf neuer Produkte oder die Eigenentwicklung wie HANA. Mit jeder neuen Technologie muss SAP entsprechende Schnittstellen entwickeln und die Methodik anpassen. Dies

gilt sowohl für SAP-Lösungen als auch für Lösungen anderer Hersteller: Für jede zu integrierende Lösung müssen explizit Schnittstellen geschaffen werden. Damit fällt mit jedem neuen Produkt Entwicklungsaufwand an. Eine Vereinheitlichung der Handhabung im SAP Solution Manager ist ebenso kaum möglich, da trotz allem jedes Produkt Spezifika aufweist, was wiederum lösungs- bzw. herstellerspezifisches Know-how erfordert. Tabelle 4.2 stellt die Vorteile dieses Ansatzes und die noch bestehenden Herausforderungen beim Change Management SOA-basierter Lösungen mit dem SAP Solution Manager dar.

Vorteile	**offene Herausforderungen**
Systemübergreifend	SAP-zentrisch
unterstützt verschiedene SAP-Techniken des Deployments (ABAP, JAVA) durch Wiederverwendung des ABAP-gestützten Korrektur- und Transportwesens	keine direkte Integration mit Service Repository und Service Bus
Möglichkeit der Einbindung anderer Deployment-Techniken	eigene Instrumentierung je Service erforderlich, fehlende Anbindung an Service-spezifische, nicht-SAP Entwicklungswerkzeuge, keine automatische Versionsverwaltung für Nicht-SAP-Services
Synchronisation von systemübergreifender Changes	nur für integrierte Deployment-Techniken
zentrale Zugriffsmöglichkeit auf Änderungsprotokolle integrierter Deployment-Techniken	kein Standard für Protokolle, keine Normierung zur Bewertung der Inhalte

Tabelle 4.2: Vorteile und Unzulänglichkeiten des SAP Solution Managers

IBM Rational

Bei der Rational Software handelt es sich um ein umfängliches Produkt zur Unterstützung von Entwicklungen, was ursprünglich im Zuge der Übernahme der gleichnamigen Firma Rational Software im Jahr 2008 übernommen wurde. Rational Software unterstützt maßgeblich die organisatorischen und funktionalen Belange der Prozesse des Apllication Lifecycle Managements, die unabhängig von der Architektur einer zu entwickelnden Lösung sind [IBM Inc.]. Dazu gehören

- Organisation und Verwaltung der Entwicklungskapazitäten

- Unterstützung der Abstimmung und Kommunikation der beteiligten Teams
- Vereinheitlichung der Infrastruktur zur Entwicklung auf einer zentralen Plattform
- Unterstützung bei der Einhaltung von Vorschriften zur Rückverfolgbarkeit von Ressourcen und Freigabeworkflows
- Unterstützung agiler Tests und Umsetzung von Änderungen.

Innerhalb der Rational Software unterstützt Rational ClearCase Change Management Solution [IBM Inc., 2011] insbesondere das Änderungsmanagement durch eine Versionsverwaltung und –steuerung. Dabei unterstützt das Produkt unterschiedliche Entwicklungsumgebungen, wie Rational Application Developer des IBM WebSphere und Eclipse des Microsoft Development Studio. Teil des Rational ClearCase ist ein Repository zur Verwaltung und Überwachung der Entwicklungseinheiten und der Berechtigungsverwaltung der Entwickler.

Mit Hilfe von Rational ClearCase Change Management Solution können somit im weitesten Sinne Datei-orientierte Entwicklungen in verschiedenen Entwicklungsumgebungen verwaltet werden. Weit verbreitete Entwicklungsumgebungen werden unterstützt. Schwierigkeiten bereiten Service- oder gar System-spezifische Deployment-Techniken.

4.5.3 Zusammenfassung

Damit liefern der SAP Solution Manager und IBM Rational Software eine erhebliche Verbesserung der Möglichkeiten des Change Management und weiterer ITIL-Prozesse [SW06] im SAP-SOA-Umfeld. Um jedoch allgemein SOA-basierte Lösungen auf Basis von Services beliebiger Hersteller zentral administrieren zu können, wird ein herstellerunabhängiges, zentrales Werkzeug auf Basis von allgemein gültigen Normierungen benötigt. Tatsächlich ist ein einheitliches Interface erforderlich, das alle Services einer SOA entsprechend unterstützen. Nur so kann mit Hilfe eines zentralen Werkzeugs, einem Cockpit, der Change Management Prozess lösungsorientiert gesteuert und realisiert werden. Wird ein Service ausgetauscht, muss die Unterstützung dieses Cockpits gewährleistet sein, um den Change Management Prozess nicht zu stören.

4.6 Release Management

In engem Zusammenhang mit dem Änderungsmanagement steht das Release Management. Ein Release ist die Zusammenfassung einer größeren und komplexeren Anzahl von Änderungen. Die Entscheidung, wann eine Anzahl

von Änderungen zu einem Release gebündelt wird, ist letztlich individuell. Ausschlaggebend sind Kriterien, wie:

- Anzahl und Umfang von Änderungen,
- Kritikalität der Änderungen hinsichtlich Geschäftsprozessen,
- Auswirkungen auf die Bedien- und Betreibbarkeit,
- Abhängigkeiten zwischen den geplanten Änderungen zu anderen Objekten.

Auf Grund dieser Kriterien ist ein Release im Gegensatz zu einer einzelnen Änderung immer komplexerer Natur. Es birgt ein größeres Risiko für den Geschäftsbetrieb und erfordert daher umfassender Absicherungs- und Vorbereitungsmaßnahmen, wie die Schulung der potentiellen Nutzer oder eine sorgfältige Planung des Testens und dessen Ausführung.

In [van der Hoek et al., 1997] wird untersucht, wie sich die damalige Einführung vom Komponenten verschiedener Technologien und mit gewisser Unabhängigkeit auf die Entwicklung und Realisierung mittels Release Management auswirkt. Dabei fokussierte man sich jedoch in erster Linie auf die Möglichkeiten der Koordination und des Managements der Entwickler in Bezug auf die genutzten Komponenten. [van der Hoek et al., 1997] kommen zu dem Ergebnis, dass ein sogenannter Software Release Manager (SRM) dafür erforderlich und beschrieben einen Prototypen. Dieser Ansatz ist sicherlich richtig, ist aber in den sich zwischenzeitlich entwickelten serviceorientierten Lösungen keinesfalls mehr hinreichend. Services sind als unabhängig voneinander zu betrachten, wodurch Änderungen an Services ebenso unabhängig sind. Erst zur Laufzeit einer Lösung werden die Abhängigkeiten zwischen den Services hergestellt. Damit können Änderungen an Services zur Änderung einer Lösung im Gesamtkontext ebenso unabhängig erfolgen. Die Herausforderungen verschiebt sich zu Gunsten des Managements dieser einzelnen Änderungen in Form eines Releases. Dieses Release Management muss wiederum unabhängig von der Technologie der Services erfolgen, die von den Änderungen in einem Release betroffen sind. Da ein Release, wie beschrieben, aus einzelnen Änderungen besteht, und bereits im vorigen Abschnitt gezeigt wurde, dass für die Ausführung von Änderungen unzureichende Werkzeuge zur Verfügung stehen, impliziert dies ein vergleichbare Situation für die Ausführung eines Release-Wechsels.

Die Verwaltung von Releases sieht vor, dass Releases mit eineindeutigen Bezeichnungen versehen werden. Im Gegensatz zu Änderungen werden Inhalte von Releases stärker dokumentiert, um die zu erwartenden umfangreichen Änderungen herauszustellen. Darin enthalten ist auch die Beschreibung der zu erwartenden Auswirkungen auf den Betrieb. Ein

Release wird aus ITIL-Sicht als ein Konfigurationselement (engl. Configuration Item) betrachtet und dementsprechend verwaltet, was zu dem Prozess des Konfigurationsmanagements führt.

4.7 Konfigurationsmanagement

Für die Sammlung von Informationen zu Konfiguration und Änderungen sieht ITIL das Configuration Management System (CMS) vor. Dies sei eine Datenbank zur Speicherung und Verwaltung der sogenannten Konfigurationselemente (Configuration Items- CI) und deren Beziehungen. Unter CI versteht ITIL alle zu einer Lösung gehörenden Hardware- als auch Software-technischen Bausteine, wie Computer, Drucker oder Softwarepakete, Systeme und letztlich auch Services. Die Architektur einer Lösung beeinflusst beispielsweise die Wahl der Einheit des CI. Mit dem Übergang zu SOA tritt als CI der Service in den Mittelpunkt softwareorientierter CI. Im Folgenden werden die Auswirkungen und neuen Anforderungen an das Konfigurationsmanagement dargestellt.

4.7.1 Analyse der Anforderungen

Das Konfigurationsmanagement ist direkt verknüpft mit dem Change und Release Management, da sich Change und Release Management sich direkt auf die Eigenschaften eines CI auswirken. Das Konfigurationsmanagement bewahrt wesentliche Status der Gesamtheit aller Eigenschaften eines CI als sogenannte Version auf und gestattet die übergreifende Auswertung. Änderungen an CI müssen insbesondere in Problemfällen lückenlos nachvollziehbar sein. Während in Client-Server-Architekturen das Konfigurationsmanagement stark ausgerichtet ist auf die reinen technologischen Einheiten, wie das System, die Komponente, die Version oder den Korrekturstand eines Softwarepakets spielt in SOA-basierten Lösungen der Service als CI die tragende Rolle. Im Gegensatz zu Client-Server-basierten Lösungen stehen die CI in SOA-basierten Architekturen in direktem Zusammenhang mit der Lösung. Natürlich werden auch die reinen technologischen CI benötigt, doch hinsichtlich der zu verwaltenden Beziehungen auf Technologie-Ebene tritt die Beziehung in Bezug zur Lösung. Diese CI sind insbesondere dann von essentieller Bedeutung, wenn es gilt zu entscheiden, ob ein CI in Form eines Services austauschbar ist oder nicht.

In SOA muss für die Beantwortung der Frage, was in einem Zeitraum geändert wurde und somit eventuell Ursache eines Problems in einer Lösung ist, von der Hardware bis zum Service einer Lösung eine lückenlose Historie erstellbar sein.

4.7.2 Marktsituation

Die praktische Umsetzung des CMS erfolgt heute unabhängig vom Service Repository mit dem einzigen Ziel, die verwendeten Versionen von CI zu ermitteln und auszuwerten. Dieser Ansatz ist für die Realisierung und Nutzung der Flexibilität von SOA basierten Lösungen nicht hinreichend.

Beispiel: theGuard! CMDB Realtech

Zur Unterstützung des Konfigurations Management bietet z.B. die Realtech AG das Produkt theGuard! Configuration Management Database (CMDB) an. Dieses Produkt fokussiert auf die Umsetzung der Anforderungen von ITIL beschriebenen Konfigurations Management Prozesses und bietet mit der verfügbaren Datenbank komplexe Möglichkeiten zur Verwaltung der CI, auch IT-Assets genannt. Die folgende Abbildung 4.8 zeigt das Zusammenspiel zwischen dem Realtech Produkt und den verschiedenen ITIL-Prozessen. Um die IT-Assets automatisch zu ermitteln, werden dabei Standardprotokolle wie TCP/UDP, SNMP oder http genutzt. Daraus ergibt sich zwangsläufig die Ausrichtung auf die Ermittlung und Verwaltung von IT-Geräten und Systemen. Das bei SOA grundlegende Konzept der losen Kopplung und Austauschbarkeit von Services als grundlegender Einheit ist nicht vorgesehen, was daran liegen mag, dass eben diese Austauschbarkeit kaum praktiziert wird. the Guard! CMDB offeriert auch die Integration mit dem SAP Solution Manager, was die Nähe zu den Produkten der SAP AG zeigt.

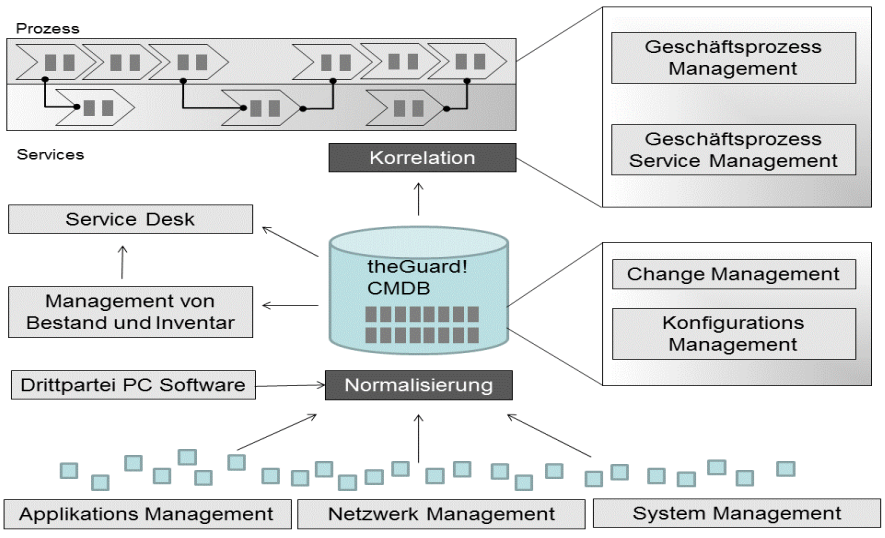

Abbildung 4.8: Integration des theGuard! CMDB in die IT-Prozesse aus [Realtech AG]

4.8 Validierung und Testen

ITIL beschreibt unabhängig von den technischen Gegebenheiten der IT-Landschaft und Implementierung, wie der Prozess der Dienstleistungsvalidierung und des Testens idealerweise ablaufen sollte. Im Folgenden wird untersucht, wie sich der Übergang von Client-Server-basierten zu SOA-basierten Lösungen auf diese Abläufe auswirkt.

4.8.1 Analyse der neuen Anforderungen

ITIL betrachtet Testen vom Standpunkt eines Betreibers. Das Testen dient der Qualitätskontrolle, um sicher zu stellen, dass Änderungen den Anforderungen und Erwartungen gemäß betrieben werden können. Den .Zusammenhang mit Releases. Welche Bedeutung dem Testen zukommt, lässt sich auch anhand des dafür verwendeten Budgets im Vergleich zum IT-Gesamtbudget erkennen. Gemäß der Umfrage von [Simon und Simon, 2012] wenden Firmen zwischen 11-30% des Budget für das Qualitätsmanagement auf. Trotz der verhältnismäßig hohen Aufwendungen ergab die Umfrage, dass etwa ein Drittel der Projekte mit schwerwiegenden Fehlern in die Produktion gehen und wenigstens 40% nicht fehlerfrei sind. Dies zeigt, dass die aktuelle Situation in der Praxis keineswegs zufriedenstellend ist.

Auf Grund der Bedeutung des Testens im IT-Umfeld wurde dafür ein expliziter Standard entwickelt [ISO/IEC, 2012]. Dieser Standard unterteilt sich in Definition und Vokabular, Beschreibung des Testprozesses, Test Dokumentation und schließlich die Techniken des Testens. Wesentlicher Bestandteil des vierten Teils ist die Definition und Beschreibung von Ziel und Inhalten der verschiedenen Testarten. Dazu gehören beispielsweise Accessibility, Backup/Recovery, Kompatibilitäts-, Konversions-, Funktionales, Wartbarkeits-, Stabilitäts-, Performance, Benutzbarkeits-, Interoperabilitäts- Tests und einige weitere. In welcher Art und Weise man derartige Tests technisch realisiert, wird jedoch nicht erläutert. Genau darin verbergen sich aber die neuen Herausforderungen mit dem Übergang der Client-Server-Architektur zu SOA.

Eine unter ca. 1000 DV-Nutzern durchgeführte Umfrage wurde in[Simon und Simon, 2012] ausgewertet. Demnach werden heute unabhängig von Branche und Technologie zwischen 11 und 30% des Gesamtbudgets für ein IT-Projekt für das Testen ausgegeben, siehe Abbildung 4.9. Diese statistische Befragung wurde zwar unabhängig von der technischen Architektur einer Lösung durchgeführt, unterstreicht aber die Bedeutung des Testens aus finanzieller Sicht. Da der Übergang zu SOA-basierten Lösungen als ein allgemeiner Trend angesehen werden kann, sind die aus der Statistik möglichen Schlussfolgerungen auf jeden Fall auch auf SOA übertragbar. Trotz dieses Aufwands sind die Ergebnisse häufig unzureichend, was heißt

zu teuer im Verhältnis zu der erlangten Sicherheit, zu langwierig oder zu unzuverlässig, d.h., Fehler wurden nicht gefunden.

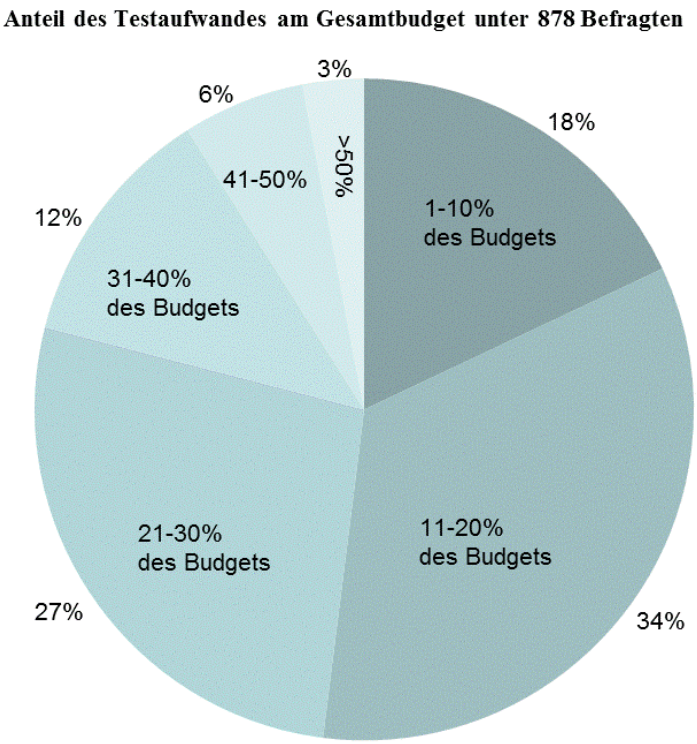

Abbildung 4.9: Prozentualer Anteil der Testkosten am Gesamtbudgets eines IT-Produkt gemäß [Simon und Simon, 2012]

Dieses ernüchternde Ergebnis reduziert zwangsläufig die Bereitwilligkeit, Änderungen und damit verbundene Tests vorzunehmen, wozu auch das Austauschen von Services gehört und Grundlage der Flexibilität von SOA ist. Die Testqualität und -effektivität ist somit von grundlegender Bedeutung, will man die Flexibilität von SOA nutzen.

In [Canfora und Di Penta, 2009] und [Canfora und Di Penta, 2006] wurden die neuen Herausforderungen des Testens im Umfeld von SOA untersucht und folgende benannt:

- Fehlende Analysemöglichkeiten des Service Codes

  Die Nutzer eines Services haben zumeist keinen Zugriff auf den eigentlichen Programm-Code eines Services. Somit bleibt nur das Testen der Schnittstellen und damit der Tests, ob geeignete Eingaben die erwarteten Ausgaben/Ergebnisse erzeugen.

- Hohe Dynamik und Anpassungsmöglichkeiten von SOA-basierten Lösungen

  Bei den traditionellen Client-Server-basierten Lösungen ist die Zusammensetzung aus den technischen Komponenten determiniert, während mit dem Übergang zu SOA der Prozessfluss erst zum Laufzeitpunkt bestimmt wird.

- Mangelnde Kontrollmöglichkeiten gegenüber den Service Anbietern

  Technische Komponente und Bestandteile sind in einem System fest integriert. Services dagegen können in unabhängiger Infrastruktur und unter der Kontrolle der Service-Anbieter ablaufen. Dadurch kann nicht mehr ohne weiteres entschieden werden, das Release eines Services zu wechseln.

- Kosten der Tests

  Die Autoren sehen hier in erster Linie die Kosten für die Nutzung von Services. Auch Tests erzeugen so Kosten. Dass dies tatsächlich verglichen zu Client-Server-basierten Lösungen zu Mehrkosten führt, bleibt zu bezweifeln. Schließlich müssen in Client-Server-Basierten Systemen entsprechende Testsysteme verfügbar sein, deren Nutzung höchstwahrscheinlich auch Lizenzkosten und damit im weitesten Sinne Nutzungskosten erzeugt.

Auch der Analyse insgesamt kann nur in Teilen zugestimmt werden. Tatsächlich erfordert der Wechsel zu SOA-basierten Lösungen ein Umdenken und die Anpassung von Tests bereits zu einem viel früheren Zeitpunkt des Testprozesses, so dass die benannten, vermeintlich neuen Probleme vermieden werden können. Um dies zu verdeutlichen, wird wiederum auf das Beispiel des Kundenauftragsprozesses zurückgegriffen.

Wie in Abbildung 2.4 beispielhaft für den Kundenauftragsprozess dargestellt, werden in der Client-Server-Architektur die Applikationen durch wenige technische Systeme und Komponenten prozessiert. Die Zusammensetzung der Infrastruktur ist als nahezu stabil zu betrachten. Damit lässt sich eine geeignete, ideale Testlandschaft immer als Kopie der produktiven Infrastruktur inklusive ihrer Daten herstellen. Änderungen und Releases betreffen funktionale Teile dieser Landschaft und können entsprechend getestet werden. An dieser Stelle stehen die auch in [Ribarov et al., 2007] untersuchten Unit Testing, Integration Tests sowie Funktionale und Nicht-Funktionale Systemtests im Fokus.

Unit Tests

Unter Unit Testing oder auch Modul oder Komponenten-Testing versteht man das Testen einzelner Einheiten innerhalb einer Lösung. Wie bereits in den vorangegangenen Kapiteln festgestellt, verschiebt sich auch beim Testen

die Einheit des Systems oder der technischen Komponente hin zum Service in SOA. Im Unterschied zu Client-Server-basierten Lösungen, wo die Anzahl von Systemen und technischen Komponenten eher gering ist, ist die Anzahl von Services weitaus größer. Zudem werden die Services erst zur Laufzeit gekoppelt und sollen austauschbar sein. Eine besondere Schwierigkeit bei der Ausführung eines einzelnen Service-Tests ergibt sich daraus, dass ein Service nicht zwangsweise ein Benutzerinterface aufweisen muss. Damit ergeben sich zwei grundsätzliche Ansätze für das Testen einzelner Services.

1. Der Test erfolgt in dem analogen Umfeld, wie der Service faktisch produktiv eingesetzt werden soll. Das impliziert, dass die produktive SOA-basierte Lösung in eine Testumgebung inklusive repräsentativer Testdaten kopiert wird.
2. Analog zur produktiven Umgebung wird zum Test des Services eine analoge Testumgebung simuliert.

Um zu entscheiden, welches die präferierte Methode des Aufbaus einer Testumgebung ist, sind die Arten von Änderungen zu klassifizieren. Von besonderer Bedeutung ist, was genau geändert wurde oder werden soll. Um den Ansprüchen der Austauschbarkeit von Services gerecht zu werden, dürfen Änderungen an Services nicht die Kommunikationsinterfaces betreffen. Änderungen der Interfaces würden automatisch Anpassungen in Vorgänger- und Nachfolgeservices bedingen. Sicherlich sind Ausnahmen möglich, doch erfordern diese grundsätzlich eine Erweiterung des Testumfangs. Man kann also davon ausgehen, dass Änderungen in aller Regel funktionale und Nicht-funktionale Eigenschaften eines Services betreffen. Demnach würde es genügen, Vorgänger- und Nachfolgeservice des zu testenden Services zu simulieren und dessen Ausführung und Ergebnisse zu evaluieren. Der Vorteil dieses Vorgehens besteht in der relativ kleinen, dementsprechend geringen Ressourcenanspruch einer solchen Testumgebung. Der Nachteil besteht in dem Aufwand, eine solche Testumgebung zu schaffen. Und es besteht ein höheres Risiko, dass die entwickelte Simulation nicht genügend mit der späteren produktiven Umgebung übereinstimmt, Testergebnisse somit nicht aussagekräftig sind.

In [Ribarov et al., 2007] wird dieser Ansatz präferiert. Als mögliche Lösungsidee werden sogenannte *Hooks* als Erweiterung des Service Bus vorgeschlagen: „[…] Test-enabled ESB is an enhancement of a regular ESB that provides hooks to be used by testing services.“ Der Service Bus dient ohnehin der Kommunikation zwischen Services. Somit liegt es nahe diese Kommunikation mit Hilfe des Service Bus nachzubilden. Betrachtet man Abbildung 2.5 so sieht man, dass ein Service aber nicht nur einen Vorgänger- und einen Nachfolgeservice haben muss, sondern vielmehr seinerseits Services aus anderen Schichten nutzen kann, um seine Aufgaben

zu erfüllen. Dadurch gestaltet sich der Aufbau einer geeigneten Testumgebung weitaus schwieriger.

Die originalgetreue Kopie der produktiven Lösung in eine Testumgebung analogen Aufbaus ist dagegen mit Überschnitt verbunden, der sich letztlich auch in Kosten niederschlägt. Gleichzeitig verschwindet damit der Aufwand zum Aufbau einer Testumgebung für einen einzelnen Service. Durch die Kopie erstellt man eine technische Testumgebung, in der alle enthaltenden Services analog der Produktion getestet werden können. Ein weiterer Vorteil ergibt sich durch die dadurch erlangte Flexibilität der Testumgebung: Jeder darin enthaltene bzw. zu integrierende Service kann ohne weiteren Aufwand getestet werden. Unter dem Ansatz die potentielle Flexibilität von SOA-basierten Lösungen auch wirklich nutzen zu wollen, darf davon ausgegangen werden, dass die Änderungshäufigkeit eher groß ist und somit verschiedene Services häufig zu testen sind. Daher kann der auch von [Ribarov et al., 2007] vorgeschlagene Ansatz 1 nur dann vorteilhaft sein, wenn die Erstellung der Testumgebung von einzelnen Services sehr geringe Aufwände erfordert. Unit Tests sollten jedoch auch im Zusammenhang mit den für die anderen Tests erforderlichen Testumgebungen betrachtet werden.

Integrationstest

Neben den Unit Tests sind Integrationstests erforderlich. Im Integrationstest werden die einzelnen Einheiten zu einer Lösung zusammengefügt und diese in ihrer Gesamtheit oder zumindest in Teilprozessen getestet. Ebenso wie bei Unit Tests verschiebt sich die zu testende Einheit vom System oder der Komponente in Client-Server-Architekturen hin zu Services in SOA. Nimmt man an, dass die Schnittstellen der Services in SOA von Änderungen nicht betroffen sind, also kompatibel bleiben, können Integrationstests auf ein Minimum reduziert werden. Das Minimum ist die Funktionstüchtigkeit der Prozesse, in denen ein von Änderungen betroffener Service integriert ist.

Funktionale und nicht-Funktionale Tests

Neben dem was und wie werden Tests auch in funktionale und nicht-funktionale Tests unterschieden. Funktionale Tests beziehen sich direkt auf die Funktionen der zu testenden Einheit, die in direktem Zusammenhang mit einem Geschäftsprozess bzw. einer Lösung stehen. Diese Eigenschaften der zu testenden Einheit dienen der Realisierung der Lösung. Als nicht-funktionale Tests werden üblicherweise Eigenschaften wie Performance, Recovery-Fähigkeit oder Wartbarkeit geprüft. Der Begriff Nicht-funktional wird dabei nur auf den Beitrag dieser Eigenschaften zur direkten Funktionsweise einer Lösung, wie dem Kundenauftragsprozess bezogen werden. In Hinsicht auf die IT Service und Support Prozesse sind diese Eigenschaften sehr wohl funktionaler Natur.

Wie bei den vorangegangen Tests auch muss wiederum die zu testendende Einheit System in die neuen Strukturen der SOA abgebildet werden. An Stelle des Systems treten als Einheit der Geschäftsprozess bzw. die Lösung, die in ihrer Gesamtheit zu testen sind. Die Ziele der ehemaligen Systemtests bleiben jedoch weitestgehend unverändert, jedoch müssen sie im Detail angepasst werden. In dieser Arbeit wird gerade gezeigt, welche zusätzlichen bzw. angepassten Funktionen ein Service in einer SOA-basierten Lösung aufweisen muss, um den neuen Anforderungen im IT-Betrieb gerecht werden zu können. Es wird gezeigt, dass sich gerade bei den Nicht-funktionalen Anforderungen ein neues Verständnis erforderlich ist. Daraus ergibt sich, dass die entsprechenden Nicht-funktionalen Tests auch angepasst werden müssen.

Regressionstest

Von Regressionstests spricht man, wenn zum Test nach Veränderungen an softwaretechnischen Bausteinen bereits bei vorherigen Versionen zum Einsatz gekommene Testfälle wiederholt ausgeführt und deren Ergebnisse verglichen werden [Seidl et al., 2012]. Regressionstest erlangen mit dem Übergang zu SOA-basierten Lösungen an Bedeutung. Man nehme an, es sollen gleichwertige Services in Bezug auf eine Lösung gegeneinander ausgetauscht werden. Um die Gleichwertigkeit wirklich zu testen, bieten sich Regressionstests an. Da die Services in Bezug auf die Lösung gleichwertig sind, darf der Regressionstest keine Unterschiede funktionaler Natur zeigen. Aus praktischer Sicht besteht jedoch wieder die Herausforderung, eine serviceorientierte Testumgebung zu schaffen. Um aber überhaupt testen zu können, sind geeignete Testdaten erforderlich. Dies führt zu einem weiteren Schwerpunkt.

4.8.2 Erzeugung von Testdaten

Die Verfügbarkeit von geeigneten Testdaten ist ein essentielles Element für das Testen. Geeignet sind Testdaten dann, wenn auf ihrer Basis Tests durchgeführt werden können, die den produktiven Prozessen möglichst ähnlich sind, die Ergebnisse also auch vergleichbar sind. Die einfachste Art eine Testumgebung inklusive Daten zu erzeugen, ist das Spiegeln (Kopieren) sowohl der Daten als auch aller Bestandteile der Software und Hardware. Gleichzeitig birgt diese Strategie jedoch auch hohe Sicherheitsrisiken, da z.B. Tester damit Zugriff auf Produktive Daten bekommen. Hinzukommt die häufig sehr große Datenmenge im produktiven Umgebungen, die entsprechend große Hardware erfordern. Damit wird die einfache Kopie der produktiven Umgebung und ihrer Daten auch aus Kostensicht unattraktiv.

Durch die Systemorientierung in Client-Server-basierten Lösungen ist zumeist das System die kleinste autarke Komponente, die in gleicher Weise in der Testumgebung aufzubauen ist. Daher fokussiert man sich auf die

Erzeugung möglichst geringer Mengen von Testdaten, um die erforderliche Hardware-Größe zu reduzieren. Durch die enge Verknüpfung der Daten mit dem System sind die Werkzeuge zur Testdatenerzeugung daher in der Praxis zwangsweise ebenso systemabhängig und -spezifisch, können also nur für die Systeme bestimmter Hersteller verwendet werden. Ein typisches Beispiel ist auch hier wieder SAP mit dem Test Data Management Server (TDMS) [SAP AG, 2014] oder Testdatengeneratoren für Oracle-Systeme [Datanamic Solutions EV] oder [EMS Database Management Solutions]. Daneben ist die möglichst flexible Erzeugung von Testdaten auch ein Bereich intensiver Forschungstätigkeit, wie zahlreiche Veröffentlichungen zeigen, [Rapps und Weyuker, 1985], [Pinto und Vergilio, 2012; Ali et al., 2011] oder [Varshney und Mehrotra, 2013]. Der wesentliche Unterschied bei der Erzeugung einer Testumgebung inklusive Daten zwischen einer Client-Server-basierten Lösung und einer strikt SOA-basierten Lösung ist die kleinste gekapselte Einheit, die im Falle von SOA der Service ist. Gleichzeitig erlangt das Testen einen noch höheren Stellenwert, will man die potentielle Flexibilität von SOA nutzen, womit noch häufigere Änderungen und damit erforderliche Tests verbunden sind.

Im Folgenden werden einige häufig verwendete Werkzeuge zur Unterstützung von Tests vorgestellt.

4.8.3 Marktsituation

An zwei renommierten Software-Produkten soll im Folgenden gezeigt werden, welche Strategien bei der Unterstützung des Testmanagements verfolgt werden

HP QuickTest Professional

HP QuickTest Professional (QTP) wurde ursprünglich von Mercury Interactive entwickelt und in 2006 von HP übernommen. Dieses Werkzeug unterstützt funktionale sowie Regressionstests durch Automatisierung. Auf Basis von VBScript können geeignete Testfälle entworfen und ausgeführt werden. HP QTP ist dabei auf Technologien bzw. Systeme und deren Technologien ausgelegt. Es unterstützt eine Vielzahl von Produkten wie SAP, Oracle, Delphi, Power Builder oder Windows Mobile.

HP QTP ist Teil des HP Quality Center [Hewlett Packard] welches den Prozess des Testmanagement innerhalb des Software Lebenszyklus unterstützt. Zum Gesamtpaket gehören auch Werkzeuge zur Erzeugung von Testdaten aus vorhandenen produktiven Systemen. Dazu werden von HP die entsprechenden, systemspezifischen Schnittstellen bereitgestellt. Die Systemorientierung des HP QTP ist damit offensichtlich. Zu testende Einheiten können Funktionen oder Geschäftsprozesse sein. Der Idee des Services und der Austauschbarkeit von Services wird keine Rechnung

getragen. Vielmehr gibt es eine klare Systemorientierung. Prozesse laufen fest eingebunden in Systemen ab. Testumgebungen werden durch einen analogen Systemaufbau mit bestenfalls speziell aus den produktiven Systemen erzeugten Testdaten geschaffen. Der Service wird nicht als eigenständiger Bestandteil betrachtet und somit auch nicht in Bezug auf seine funktionalen und Nicht-funktionalen Eigenschaften getestet. Es können maximal die Nicht-funktionalen Eigenschaften der Systeme und der Lösungen innerhalb der Systeme getestet werden.

IBM Rational Quality Manager

Einen ähnlichen Ansatz verfolgt IBM Rational Quality Manager [IBM Inc.]. Es unterstützt ebenfalls den Testmanagement Prozess. Es bietet Werkzeuge zur Verwaltung, dem Design und zur Ausführung und Auswertung von Teststrategien und Testfällen. Neben der Erstellung von Regressions- und Integrationstests liegt der Schwerpunkt auf der Auswertung der Testergebnisse und des -fortschritts. Auch hier wird auf das System als Testumgebung fokussiert. Die eigentliche Herstellung geeigneter Testumgebungen bleibt dem Nutzer überlassen. Aus Sicht der Anforderungen für das Testen von SOA-basierten Lösungen ergeben sich damit ähnliche Problem wie bei dem zuvor beschriebenen Produkt HP QTP. Zwar werden das eigentliche Management von Testfällen und die Auswertung von Testergebnissen optimal unterstützt, doch geht IBM Rational Quality Manager ebenso von einer systemorientierten zu testenden Lösung aus. Das Testen von Services z.B. hinsichtlich ihrer Gleichwertigkeit in Bezug auf eine Lösung ist nicht vorgesehen. Nicht-funktionale Eigenschaften werden auch hier in Bezug auf ein System betrachtet.

4.8.4 Zusammenfassung

Bereits in [Canfora und Di Penta, 2009] wurden die neuen Herausforderungen des Testens im SOA-Umfeld umrissen. Dabei wurde jedoch davon ausgegangen, dass die Nicht-funktionalen Anforderungen an ein System und einen Service vergleichbar sind. Wie aber gerade in dieser Arbeit nachgewiesen wird, ergeben sich aus dem Betrieb und insbesondere der Nutzung der Flexibilität von SOA auf Basis der Austauschbarkeit von Services neue Anforderungen an die Architektur von SOA und deren Services. Gleichzeitig eröffnen sich aber auch neue Möglichkeiten für das Testen durch die relative Unabhängigkeit von Services, wie die Idee der Hooks in [Ribarov et al., 2007] zeigt.

4.9 Kapazitätsmanagement

Das Kapazitätsmanagement bezieht sich auf den Prozess der Planung sämtlicher Ressourcen sowohl Hardware und Software als auch personeller Natur.

4.9.1 Analyse der neuen Anforderungen

ITIL folgend ist dieser Prozess der Phase Design zuzuordnen, da insbesondere auch die Ressourcen zur Einführung einer neuen oder geänderten Applikation geplant werden müssen. Gleichzeitig umfasst das Kapazitätsmanagement aber auch wichtige Aufgaben in der Phase des Betriebs einer Lösung. In dieser Arbeit stehen diese Aufgaben des Kapazitätsmanagements in Bezug auf die Hardware-Ressourcen im tagtäglichen Betrieb im Mittelpunkt. Auch wenn Limitierungen der Speicherkapazität immer weiter verschoben werden, so bedeutet ungebremster Ressourcenbedarf trotzdem Kosten und auch einen erhöhten Management-Aufwand. Daher wird die Nutzung der Ressourcen, wie Festplattenspeicher, CPU-Auslastung und Memory während des Betriebs gemessen unabhängig von der Architektur einer Lösung. Unterschiede zwischen Client-Server- und SOA-basierten Lösungen ergeben sich erst bei der Messeinheit. Durch die Fixierung Client-Server-basierter Lösungen auf einige, eher wenige Systeme und technische Komponenten, genügt es, die Ressourcennutzung auf diese Einheiten bezogen zu messen. Allenfalls die je Prozess genutzten Ressourcen sind noch von Interesse. Das Kapazitätsmanagement erfolgt auf Basis technischer Objekte, wie System, technische Komponente oder physischer Tabelle. In diesen Größen wird sowohl geplant auch als in der Phase des Betriebs überwacht und gemessen.

In SOA-basierten Lösungen existiert dagegen diese feste Kopplung zwischen Service und technischen Komponenten nicht. Selbst die Services werden erst zur Laufzeit gekoppelt. Erst die lose Kopplung gestattet auch den möglichen Austausch von gleichwertigen Services hinsichtlich einer Lösung. Aus diesem Grund liegt es nahe, dass die Planung und Messung des Kapazitätsbedarfs in SOA auf Basis des Services und der Lösung erfolgen muss. Um bereits eine Kapazitätsplanung in der Design Phase vorzunehmen, erfordert dies die Herstellung funktionaler Zusammenhänge zwischen Einsatz, Verwendung und Durchsatz eines Services zu den erforderlichen Ressourcen. Zu ähnlichen Ergebnissen gelangt auch [Alter et al., 2009], die sich mit der Planung von IT Governance-Prozessen auseinandersetzen. Aus Sicht eines IT-Providers kommen automatisch Kostenaspekte hinzu, die bisher mit dem Ressourcengebrauch von Systemen verknüpft sind. Wird jedoch die SOA-Architektur genutzt und auch wirklich gelebt, müssen Kosten auf den Ressourcengebrauch der genutzten Services abgebildet werden. Eng damit verknüpft ist die Frage der Berechnung von Lizenzkosten. Das bisher häufig genutzte Modell der Lizensierung auf Basis

der Anzahl Nutzer in einem System lässt sich nicht mehr aufrecht erhalten. An Stelle dessen müssen andere Modelle auf Basis der Service-Bereitstellung und Nutzung gefunden werden. In dieser Arbeit soll jedoch auf Grund der Komplexität auf den Kostenaspekt nicht weiter eingegangen werden. Vielmehr wird darauf fokussiert, den Betrieb von flexiblen SOA-Lösungen technisch zu realisieren. Neben der eher dynamischen Nutzung von Hardware-Ressourcen wie CPU und Memory werden Festplattenspeicher permanent für die Speicherung von Daten genutzt. Beim längerfristigen Betrieb einer Lösung wird sich zwangsweise nach und nach die Frage stellen, wie man veraltete, nicht mehr in der Landschaft benötigte Daten archiviert, auslagert bzw. aus dem laufenden Geschäftsbetrieb entfernt. Während in Client-Server-basierten Systemen das Datenmanagement auch hier auf Basis der technischen Objekte erfolgt, muss im SOA-basierten Umfeld ein neuer Ansatz entwickelt werden. Wie bereits dargestellt, wird der Benutzer zunehmend vom technischen Know How hinsichtlich der Umsetzung befreit. In der Konsequenz wird er daher die Lösung auch Prozessorientiert managen z.B. in der Kapazitätsplanung. Das heißt, nicht mehr das technische Objekt wird gemanagt, sondern der Benutzer fokussiert sich auf das Prozessobjekt. Tabelle 4.3 stellt die unterschiedlichen Ansätze beider Architekturen beim Kapazitätsmanagement gegenüber.

Client-Server- basierte Lösung	**SOA**
Messung der Datenmengengröße in technischen Objekten- Datenbank, Tabelle, Index	Definition von Service-übergreifenden Prozessobjekten Messung der Datenmengengröße in Bezug auf diese Prozessobjekte unabhängig von der konkreten technischen Realisierung des Datenbank Service
Export oder Import von Daten in technischen Objekten	Export oder Import aller Daten, die zu einem Prozessobjekt gehören, beispielsweise wenn ein Datenbank Service ausgetauscht werden soll oder für Recovery-Zwecke

Tabelle 4.3: Gegenüberstellung der Maßgrößen in Client-Server-basierten und SOA-basierten Lösungen

Client-Server- basierte Lösung	SOA
Archivierung und Löschung von Daten aus Datenbank-tabellenorientiert	Archivierung bzw. Löschung von Prozessobjekten (Datenbankübergreifend!) unter Verwendung von Datenbank Services
Überwachung von technischen Kennzahlen je Datenbankmanagementsystem, wie CPU, I/O, Memory, Festplattenspeicher, Betriebssystem	Zusätzlich: Überwachung der Hardwarelast je Service, wie CPU, I/O, Memory, Festplattenspeicher, Betriebssystem
Durchschnittliche Antwortzeiten Langlaufende (SQL)-Kommandos	Antwortzeiten je Service Langläufer je Service

Fortsetzung Tabelle 4.3: Gegenüberstellung der Maßgrößen in Client-Server-basierten und SOA-basierten Lösungen

Im Fall des Kundenauftragsprozesses sind die zu messenden und zu verwaltenden Datenmengen eben die Menge von Kundenaufträgen, Stücklisten, Lieferaufträgen oder Rechnungen. Um dies zu realisieren, ist ein Mapping zwischen technischen Speicherobjekten und Prozessobjekten erforderlich. Ist ein solches Mapping nicht verfügbar, kann sowohl im Betrieb als auch in der Planung der Kapazität nicht auf den technischen Ressourcengebrauch geschlossen werden. Ein Benutzer oder die Geschäftsabteilungen sind in aller Regel auskunftsfähig hinsichtlich der erwarteten Entwicklung ihres Bussiness. Existiert ein Mapping zwischen erwarteter Anzahl und Umfang von Kundenaufträgen, wie im gezeigten Beispiel, kann auf die zu erwartende erforderliche Kapazität geschlossen werden.

In [Brosig et al., 2010] wird es ebenso als Problem gesehen, dass die Nutzung der Flexibilität von SOA durch den Austausch von Services bei Bedarf zu unvorhergesehen Ressourcenengpässen und damit Performance-Problemen führen kann. Als Lösungsidee wird ein sogenanntes „Self-Aware Performance und Kapazitätsmanagement“ vorgeschlagen. Darin wird von einer gegebenen Hardwarekapazität ausgegangen und der Gedanke verfolgt, wie ein abgeschlossenes System sich selbst optimal konfigurieren kann. Prinzipiell erscheint dieser Gedanke lukrativ. Sollte es gelingen, diese Ideen umzusetzen, würde trotz allem die Kapazitätsplanung nicht entfallen. Der von [Brosig et al., 2010] beschriebene Ansatz ist strikt reaktiv. Die Ressourcen werden als gegeben betrachtet. Die Risiken können nicht

ausgeschlossen werden, dass die gegebenen Kapazitäten zu gering oder zu mächtig und daher unangemessen teuer sind. Um diese Risiken zu eliminieren, muss eine vorausschauende Planung in der Designphase einer Lösung erfolgen. Die Entwicklung der Hardwarekapazität in Abstimmung mit der Entwicklung des Geschäfts bleibt davon unberührt. Ausgangpunkt auch einer Selbstoptimierung eines Systems werden trotzdem geeignete Messpunkte bilden. Die Messungen können nur in Bezug zu der Einheit Service erfolgen.

Um dies zu realisieren, muss ein Service weitaus mehr Informationen über sich bereitstellen, als das bisher die Regel ist. Damit die Ressourcenabschätzung bereits in der Designphase eines Projektes vorbereitet werden kann, muss bereits der Hersteller eines Services entsprechende Daten bereitstellen. Faktisch ist dies möglich, da im Zuge von entsprechenden Tests vor der Auslieferung eines Services üblicherweise auch Performanceanalysen auf unterschiedlicher Hardware und unterschiedlichen Kapazitäten erfolgen. Tatsächlich kann der Service-Hersteller also durchaus Performance-Richtwerte und Regeln zur Berechnung der erforderlichen Hardware-Ressourcen bereitstellen.

Im Folgenden sollen einige reale Ansätze aus der Praxis untersucht werden.

4.9.2 Marktsituation

Dass es möglich ist, vor der eigentlichen Nutzung einer Software oder eines Services bereits die insbesondere erforderlichen Hardware-Ressourcen abzuschätzen, beweisen bereits existierende Werkzeuge zur Kapazitätsplanung einiger Software-Hersteller, von denen im Folgenden der SAP Quicksizer und das Sizing für Oracle in den Grundzügen dargestellt werden sollen.

Quicksizer von SAP

Bei dem sogenannten Quicksizer von SAP handelt es sich um ein komplexes Regelwerk, das auf Basis von während der SAP-Tests ermittelten funktionalen Zusammenhängen und vom Kunden bereit zu stellenden Größenangaben hinsichtlich geplanter Geschäftsprozesse und deren Nutzung und Benutzer erforderliche Hardware-Kapazitäten für ein geplantes System ermittelt. Ein Schlüsselelement des Quicksizers bildet dabei die Einheit SAPS [Schneider, 2013], was für SAP Performance Standard steht. Diese Einheit hat sich im Laufe der Zeit für die Prozessierungsgeschwindigkeit von Rechnern etabliert und beruht auf dem sogenannten Sales und Distribution (SD) Benchmark. Zum Zeitpunkt der Einführung dieser Anwendung in den damaligen SAP-Systemen stellte diese Anwendung besondere Herausforderungen an die Hardware-Kapazität. Um verschiedene Hardware miteinander vergleichen und Bedarfe abschätzen zu können,

wurde die Anzahl möglicher SD-Transaktionen in einer Stunde gemessen. 100 SAPS entsprechen der Kapazität, um 2000 Auftragspositionen je Stunde zu prozessieren, was 6000 Dialogschritte und 2000 Verbuchungen bedeutet. Der Vorteil der Einheit SAPS besteht in der Unabhängigkeit von Hersteller-spezifischen Größen. Die Leistungsfähigkeit verschiedener CPU kann in SAPS gemessen und so verglichen werden. Andere SAP-Applikationen werden zudem zu diesen SD-Transaktionen ins Verhältnis gesetzt. Damit lässt sich die erforderliche Prozessierungskapazität eines SAP-Systems mittels SAPS Hardware-unabhängig darstellen.

Mit Hilfe des Quicksizers werden die wichtigsten Kennzahlen der Kapazität eines Rechners für den Betrieb der Applikationsserver ermittelt:

- Rechenleistung
- Datenspeicher (Permanentspeicher)
- Hauptspeicher
- Ein- und Ausgabedurchsatz
- Netzwerkdurchsatz

Im Ergebnis des Quicksizers werden immer die erforderlichen Hardware-Kapazitäten für ein System (Server) ermittelt. Der Quicksizer folgt damit den Architekturbausteinen der Client/Server-Technologie. Der serviceorientierte Ansatz spielt eine keine Rolle. Bei der Kapazitätsermittlung wird für den Benutzer der Zusammenhang zwischen geschäftlichen Aktivitäten und benötigtem Datenspeicher in einem Datenbank-Management-System nicht dargestellt. Die Relation zwischen geschäftlichen Aktivitäten und z.B. physischen Tabellen bleibt unbekannt.

Dieses Beispiel zeigt, dass der Hersteller einer Software in der Lage ist, Regeln für die Ermittlung der erforderlichen Kapazität für den Betrieb einer Applikation zur Verfügung zu stellen. SAP nutzt dabei einen sehr SAP-spezifischen und damit schwer auf andere Hersteller übertragbaren Performance Standard. Für SOA-basierte Lösungen, wo Services verschiedener Hersteller in einer Lösung zusammen betrieben und Services im Bedarfsfall Hersteller-unabhängig ausgetauscht werden sollen, kann dieser Standard daher nicht verwendet werden. Der SAP Quicksizer ist zudem System- und nicht serviceorientiert.

Oracle Transaktionen pro Sekunde

Oracle verwendet als Einheit für die Rechenleistungsfähigkeit einer Hardware die Größe *Transactions Per Second* (TPS) [Burleson und Danchenkov, 2005] oder [Oracle USA, 2007]. Darunter versteht Oracle im Endeffekt die Anzahl von durchgeführten Commits bzw. Rollback-Anweisungen, die das Ende einer Transaktion darstellen. Der Inhalt einer Transaktion besteht aus einer Abfolge von SQL-Anweisungen. Die

Vergleichbarkeit einzelner Transaktionen ist nicht gewährleistet, da Anzahl und Komplexität von Transaktionen nicht definiert sind und daher von Applikation zu Applikation unterschiedlich sein können. Im Gegensatz zur Planung wird für die laufende Performanceoptimierung in einem produktiven System die Größe TPS noch einmal genauer spezifiziert. Es wird zusätzlich die Anzahl von Festplatten Transaktionen je Sekunde gemessen, was die Anzahl von Lese- und Schreibaktionen auf der Festplatte umfasst, die durch eine Transaktion veranlasst wird. Des Weiteren wird die Anzahl von Betriebssystem Transaktionen je Sekunde ermittelt, was die Inanspruchnahme von Betriebssystemprozessen je Transaktion beinhaltet. Mit Hilfe dieser Messungen kann im Betrieb einer Lösung entschieden werden, ob die verfügbare Hardwarekapazität hinreichend ist oder erweitert werden muss. Die Einheit TPS ist jedoch ausschließlich technischer Natur. Oracle definiert keine direkten Relationen zu Anwendungen. Für Oracle eigene Applikationen existieren dagegen teilweise weitere, spezifische Anleitungen für die Kapazitätsermittlung.

Für das Kapazitätsmanagement eines Datenbank-Management-Systems ist zu dem die zu erwartende Datenbank- und gegebenenfalls Tabellengrößen von Bedeutung. Für die Planung und Ermittlung dieser Größen bietet Oracle wiederum technische Berechnungsformeln, wie z.B. in [Ault, 2003] dargestellt. Um die Berechnungsregeln für die Planung des permanent benötigten Speicherplatzes für eine Lösung einsetzen zu können, muss der Anwender wissen, welche Datenstruktur eine Applikation benötigt, wie die die erzeugten Objekte darin abgelegt werden und welche Datenmengen für eine Applikation zu erwarten sind. Mit diesem Ansatz wird man erfolglos versuchen, auf SOA-basierte Lösungen unter dem Aspekt der avisierten Flexibilität zu übertragen, da die Kapazitätsplanung für das Datenbank-Management-System Oracle stark Oracle-spezifisch, systemorientiert und applikationsunabhängig aufgebaut ist. Der Service-Gedanke spielt keine Rolle.

Andere Einheiten der Kapazitätsplanung

Neben den vorgestellten Beispielen von Oracle und SAP für die Hardware-Kapazitätsplanung ihrer Applikationen sind zahlreiche weitere, entweder Hersteller-abhängige oder Technik-abhängige Kapazitätsgrößen gebräuchlich. Genannt seien:

- Transactions per minute (TPM) in verschiedenen Ausprägungen hervorgegangen aus den TPC-Benchmarks in den entsprechenden Ausprägungen[Transaction Processing Performance Council]
- Floating-Point Operations Per Second (FLOPS) im Bereich komplexer, wissenschaftlicher Berechnungen mit Gleitkommazahlen

- Packets per Second (PPS) für den Durchsatz von Datenpaketen in Netzen, z.B. bei [Cisco Systems, 2013].

4.9.3 Zusammenfassung

Allen dargestellten Einheiten für das Kapazitätsmanagement im Bereich der Rechenleistung ist gemein, dass Performance immer in Bezug auf Aufgaben je Zeiteinheit geplant und ermittelt wird. Dabei sind die Einheiten jedoch je Hersteller und Einsatzgebiet unterschiedlich und nur scher übertragbar. Tatsächlich gibt es zum jetzigen Zeitpunkt kein Werkzeug, das Service-übergreifend und damit insbesondere eine herstellerunabhängige Kapazitätsplanung erlaubt oder derartige Informationen in SOA-basierten Lösungen sammelt und bereitstellt.

4.10 Verfügbarkeitsmanagement

Wie bereits erläutert, besteht das Verfügbarkeitsmanagement gemäß ITIL aus den Komponenten Verfügbarkeit, Maintainability, Resilience, Reliability, Serviceability und Security. Zentrale Herausforderung ist die ununterbrochene Nutzung einer Lösung, was im Idealfall einen vollständig unterbrechungsfreien Betrieb der Lösung erfordert. Die Bestandteile des Verfügbarkeitsmanagement geben die verschiedenen Anforderungsbereiche dafür wieder. Insbesondere muss eine Lösung entsprechend den Vorgaben wartungsfähig, zuverlässig, wiederherstellbar und entsprechend fehlertolerant sein. Wie in angekündigt, soll auf das Security Management in seiner Gesamtheit auf Grund der Komplexität nicht weiter eingegangen werden.

4.10.1 Analyse der Anforderungen

Die eigentliche Verfügbarkeit einer Lösung wird immer in Bezug zu dem mit dem Kunden vereinbarten Zeitrahmen betrachtet. Ähnlich wie das bereits für andere Prozesse analysiert wurde, wird auch hier bei Client-Server-basierten Lösungen die Verfügbarkeit einer Lösung übertragen auf die Verfügbarkeit der technischen Systeme und Komponenten. Zwar kann bereits auch dies zu Inkonsistenzen führen, doch ist der Rückschluss von der technischen Verfügbarkeit der beteiligten Systeme auf die Verfügbarkeit einer Lösung relativ sicher, da die Anzahl der technischen Systeme eher gering ist. Trotzdem sind technische Verfügbarkeit aller Komponenten und Systeme nicht gleich zu setzen mit der Verfügbarkeit einer Lösung. Obwohl die technische Komponente verfügbar ist, kann beispielsweise extrem hohe Last zu einer Verschlechterung der Performance führen, so dass die Verfügbarkeit der Lösung faktisch nicht gegeben ist. Beschränkungen von Benutzerzahlen können ebenso zur Nichtverfügbarkeit einer Lösung führen.

Die technische Verfügbarkeit der Komponenten ist somit notwendig, aber nicht hinreichend um die Verfügbarkeit der Gesamtlösung zu garantieren.

4.10.2 Verfügbarkeit am Beispiel des Kundenauftragsprozesses

Man betrachte wiederum das Beispiel des Kundenauftragsprozesses aus Abbildung 2.3. Mit dem Übergang zu SOA-basierten Lösungen und der Service-Ausrichtung wird die auf technische Systeme und Komponenten bezogene Betrachtungsweise zunehmend unzulänglich. Um dies zu verdeutlichen, wird anhand eines wesentlichen Werkzeugs des Verfügbarkeitsmanagements, der Component Failure Impact Analysis (CFIA) demonstriert, wie sich Client-Server- und SOA-basierte Lösungen unterscheiden. Bei der CFIA handelt es sich um eine Matrix, die die technischen Bestandteile in Bezug auf die Verfügbarkeit der Geschäftsprozesse untersucht und beschreibt, was beim Ausfall einer der technischen Komponente zur Wiederherstellung der Verfügbarkeit zu tun ist. CFIA ist ebenso ein wichtiges Werkzeug im IT-Business Continuity Management. Im Gegensatz zum Verfügbarkeitsmanagement wird beim IT-Business Continuity Management der absolute Crash angenommen und jegliche Mittel zur wenigstens manuellen Fortführung des Business untersucht. Auch dafür müssen die Zusammenhänge zwischen Bestandteilen einer Lösung und Prozessen verfügbar sein. Am Beispiel des Kundenauftragsprozesses und der in Abbildung 2.4 angenommenen technischen Implementierung auf Basis einer Client-Server-Architektur wären die zu betrachtenden technischen Systeme bzw. Komponenten das Kundenauftragssystem, das ERP-System, das Planungs- und Verfügbarkeitssystem und das Drucksystem. Hinzu kommen technische Komponenten wie beispielsweise das Datenbankmanagementsystem oder Schnittstellen. Damit ergäbe sich der in Tabelle 4.4 dargestellte CFIA-Rahmen.

Technische Komponente\ Geschäftsprozess	**Kundenauftragsprozess**
Kundenauftragssystem	
ERP-System	
Planungs- und Verfügbarkeitssystem	
Drucksystem	
Datenbankmanagementsystem	
Schnittstelle zum ERP-System	
Schnittstelle zum Drucksystem	

Schnittstelle zum Planungs- und Verfügbarkeitssystem	

Tabelle 4.4: Vorlage für eine CFIA für das Beispiel des Kundenauftragsprozesses in einer Client-Server-basierten Landschaft

Üblicherweise wird ein technisches System oder Komponente nicht nur von einem Prozess genutzt, so dass ein Ausfall meist mehr als einen Prozess betrifft. Im Fall des Kundenauftragsprozesses wird man je nach erforderter Verfügbarkeit des Kundenauftragsprozesses Hochverfügbarkeitslösungen auf Basis der technischen Systeme einführen müssen. Es wird schwer möglich sein, für alle in dem jeweiligen System auszuführenden Geschäftsprozessschritte Alternativen zu finden oder gar ein Alternativsystem. In SOA-basierten Lösungen eröffnen sich dagegen neue Möglichkeiten. Nimmt man an, dass der Kundenauftragsprozess strikt serviceorientiert aufgebaut wurde, ist die für Client-Server-basierte Lösungen benutzte CFIA unzureichend und muss vielmehr durch eine ***Service Failure Impact Analysis (SFIA)*** ersetzt werden. Für den in Abbildung 2.3 dargestellten Kundenauftragsprozess zeigt Tabelle 4.5 wie eine solche SFIA aussehen kann.

Layer	**Service**	**Kundenauftragsprozess**
Prozessschicht	Kundenauftrag anlegen	
	Verfügbarkeitsprüfung	
	Produktion	
	Lieferauftrag anlegen	
	Entnahmeliste generieren	
	Lieferdokument senden/drucken	
	Ware versenden	
	Faktura erzeugen	
	Rechnung senden/drucken	

Tabelle 4.5: Vorlage für Service Failure Impact Analysis für den Kundenauftragsprozess in SOA

Es wird vorgeschlagen, die SFIA schrittweise, hierarchisch aufzubauen. In einem ersten Schritt sind die Auswirkungen und die Reaktionsmöglichkeiten beim Ausfall der Services in der Prozessschicht auf den Kundenauftragsprozess zu untersuchen. Des Weiteren müssen die Auswirkungen und Möglichkeiten beim Ausfall der genutzten Services aus der Zwischenschicht und dem Basisschicht in Bezug auf die Services im

Prozessschicht untersucht werden. Dazu muss bekannt sein, welche Services zum Ausführungszeitpunkt lose gekoppelt werden, d.h., die Service-Hierarchie muss bekannt sein. Ein vergleichbarer Ansatz wurde bei [Xie et al., 2008] mittels Workflows dargestellt. Ausgangspunkt ist ein Business Workflow. Davon ausgehend werden die Workflows der genutzten Prozesse aufgebaut und schlussendlich auf die erforderlichen IT-Ressourcen abgebildet und deren notwendige Verfügbarkeit abgeleitet. Offen bleibt jedoch, wie man diese möglichst mit technischer Unterstützung Workflows aufbaut. Die entsprechende Verfügbarkeit von IT-Ressourcen ist eine notwendige Voraussetzung, doch kann Verfügbarkeit von SOA-basierten Lösungen auch durch die redundante Bereithaltung gleichwertiger Services hinsichtlich einer Lösung gesteuert werden. Damit steht Verfügbarkeit von SOA-basierten Lösungen in engem Zusammenhang mit **Resilience**, also der redundanten Bereitstellung von gleichwertigen Services für eine Lösung. Resilience bezieht sich in Client-Server-basierten Lösungen immer auf die Redundanz von Systemen und Hardware. Für SOA-basierte Lösungen ergeben sich durch die Redundanz von Services völlig neue Optionen, Verfügbarkeit zu gewährleisten.

Die Unterstützung eines solchen Ansatzes durch das Service Repository ist jedoch bisher nicht eindeutig festgeschrieben. Das Service Repository wird als das zentrale Element einer SOA beschrieben, das alle Metadaten eines Services bereithält. Dazu gehören die Beschreibung der Funktionalität eines Services, Bedingungen für seine Nutzung, Schnittstellenbeschreibungen (benötigte Daten und Formate) und Eigentümerschaft. [Oasis, 2011] und [Oasis, 2004] definieren zwar die Bestandteile einer SOA und bieten einen Standard für eine Beschreibungssprache, doch bleibt Oasis auf theoretischem, wenig konkretem Niveau.

4.10.3 Zusammenfassung

Damit ergeben sich für das Verfügbarkeitsmanagement, wie hier beschrieben, zwei grundsätzliche, bisher nicht festgeschriebene Informationslücken:

1. In SOA gibt es keine standardisierte Methode und Informationsspeicher, der die Gleichwertigkeit von Services hinsichtlich der im Einsatz befindlichen Lösungen bereithält.
2. Es wird nicht standardisiert dokumentiert, wie Services der verschiedenen Schichten abfolgen, noch welche Services zum Ausführungszeitpunkt einen Geschäftsprozess abbilden.

Um diese Lücken zu schließen, ergeben sich weitere Anforderungen an die Bestandteile einer SOA, worauf nachfolgend eingegangen werden soll.

4.11 Applikationsmanagement

Die Funktion des Applikationsmanagement ist es, das für den Betrieb erforderliche Wissen zu erarbeiten und in die Prozesse einfließen zu lassen. Damit berührt diese Funktion einen besonders sensiblen Bereich des Betriebs, welches Wissen ist wo erforderlich, um den Betrieb gemäß den Qualitätskriterien zu gewährleisten. "Wissen" ist dabei zutiefst menschlich geprägt. In der Konsequenz kann der Mensch nur innerhalb seiner biologischen Grenzen Wissen speichern, ändern und verwenden. Flexibilität kann zu einem kritischen, begrenzenden Faktor werden, wenn sie die Änderung des menschlich verfügbaren Wissens in seinen Grenzen überschreitet. Diese menschlichen Grenzen können jedoch durch geeignete Werkzeuge positiv verschoben werden. In Bezug auf SOA-basierte Lösungen und der Nutzung der Flexibilität kann dies mit den heute verfügbaren Techniken und Werkzeugen nicht erreicht werden. Da es keine Standards auf Detailebene, sondern lediglich auf Prozessebene (ITIL) gibt, erfordert jede Änderung beim Einsatz von Services und insbesondere beim Austausch von Services auf Basis unterschiedlicher Technologien eine Anpassung des Know Hows zum Betrieb, sei es durch den Einsatz anderer Werkzeuge oder durch die wechselnde Interpretation von Kennzahlen und Messwerten und das, obwohl die Zielsetzung des Betriebs und der IT Prozesse insgesamt faktisch gleich bleibt: Erfüllung der Erwartungen der Endbenutzer bei der Verwendung der Geschäftsprozesse. In der Konsequenz kann dies nur heißen, dass der Betrieb und damit das dafür erforderliche Wissen über die unterschiedlichen Services und Technologien hinweg vereinheitlicht wird.

5 COCo als zentrales Betriebselement

In den vorangegangenen Abschnitten wurde erstmals gezeigt, dass sich für die IT Service und Supportprozesse von SOA-basierten Lösungen in bestimmten Bereichen maßgeblich neue Anforderungen ergeben und dass diese mit den herkömmlichen und bisher verfolgten Ansätzen nicht lösbar sind, wenn die Flexibilität von SOA auf Basis der Austauschbarkeit von Services genutzt werden soll. Es wurde der Nachweis erbracht, dass durch diese Anforderungen ein weiterer zentraler Baustein in SOA notwendig ist: der als zentrales Cockpit fungierende zentrale Betriebsservice COCo. Daraus ergibt sich die Notwendigkeit der Anpassung der Definition einer SOA auf Basis ihrer Bausteine. In diesem Kapitel werden die sich für den Betrieb einer SOA-basierten Lösung unter Nutzung der Flexibilität von SOA ergebenen Anforderungen zusammengeführt. Für die einzelnen Bestandteile einer SOA, Service, Service Repository, Service Bus und COCo werden die sich aus dem Betrieb ergebenen Anforderungen formuliert und erläutert, wie die Bausteine diesen Anforderungen gerecht werden können. Die Erfüllung der hier zusammengestellten Anforderungen bildet die Grundlage des stabilen und ökonomischen Betriebs einer SOA unter gleichzeitiger Nutzung der Flexibilität der Lösung auf der Grundlage des Austauschs bzw. der Anpassung von Services.

5.1 Defintion SOA

Die im weitesten Sinne wesentlichen Prozesse und Aktionen für den Lösungsbetrieb wurden untersucht. Es wurden die Unterschiede zwischen dem Betrieb Client-Server-basierter und SOA-basierter Lösungen gezeigt und die Unzulänglichkeiten der bisher im Einsatz befindlichen Werkzeuge für derartige Prozesse dargestellt.

Wenn SOA-basierte Lösungen analog zu Client-Server-basierten Lösungen betrieben werden, führt dies:

- zum Verlust der Flexibilität von SOA,
- zu steigenden Betriebskosten mit der Anzahl integrierter Services und Technologien sowie
- zur Gefährdung allgemeiner Qualitätskriterien wie Stabilität, Verfügbarkeit und Performance.

Die in Client-Server-basierten übliche systemorientierte Ausrichtung des Betriebs muss sich mit SOA zu einem service- und lösungsorientierten Betrieb entwickeln. Tabelle 5.1 stellt die Ansätze gegenüber.

Systemadministration in Client-Server basierten Lösungen	Lösungsadministration in SOA basierten Lösungen
Technologie- und systemorientierte Werkzeuge	Vereinheitlichte zentrale Werkzeuge mit Fokus auf die Lösung
systemorientierte Aktionen, Einheiten und Begriffe	service- und lösungsorientierte Aktivitäten, Einheiten und Begriffe
Einsatz der Werkzeuge und Durchführung der Aktivitäten erfordert system- und technologie-spezifisches Wissen	Vereinheitlichung der lösungs- und serviceorientierten Administration gestattet den Betrieb ohne tiefgreifende Kenntnis der Service Technologie und Spezifika

Tabelle 5.1: Zusammengefasste Gegenüberstellung der Administrationsanforderungen in Client-Server- und SOA-basierten Lösungen

Die Nutzung der Flexibilität von SOA-basierten Lösungen auf Basis des Austauschs von Services stellt nur einen Vorteil dar, wenn dieser Austausch keine bzw. nur geringe zusätzliche Aufwände im Betrieb der Lösung hervorruft. In der Konsequenz muss der Betrieb demnach unabhängig von den Eigenschaften der in der Lösung genutzten Services erfolgen.

Des Weiteren erzwingt die Nutzung der Flexibilität eine Service- und Lösungsorientierung des Betriebs und eines ebensolchen Werkzeugs. Eine Technologie- und Systemorientierung würde die flexible Servicenutzung ungerechtfertigt einschränken. Daher kann ein solches SOA-Betriebswerkzeug nur als zentraler Baustein in der SOA existieren. In der Konsequenz wird die Einführung eines zentralen Betriebscockpits (*Central Operations Cockpit-COCo*) gefordert. Die von [Melzer und Eberhard, 2008] gelieferte Definition von SOA auf Basis der Eigenschaften erfordert eine Erweiterung der bisher angenommenen Bausteine einer SOA, wie in Abbildung 5.1 verdeutlicht.

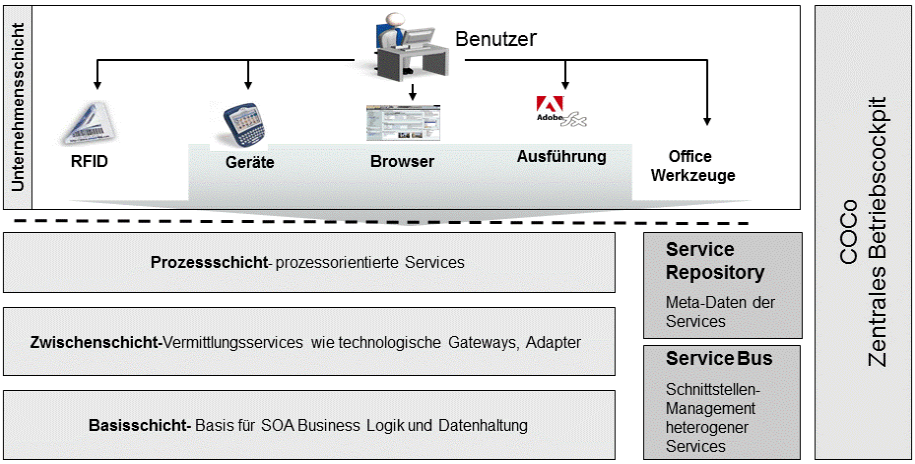

Abbildung 5.1: Erweiterung der SOA-Bestandteile um einen zentralen Betriebsservice

Damit ergibt sich als Definition für SOA auf Basis ihrer Bestandteile:

Definition 5.1: SOA

Eine serviceorientierte Architektur basiert auf den Schlüsselelementen

- Applikations-Frontend,
- Service,
- Service Repository,
- Service Bus und
- Zentrales Betriebscockpit (COCo).

Ein Service besteht aus einem Kontrakt, ein oder mehreren Interfaces, der Implementierung und unterstützt die nicht-funktionalen Anforderungen aus dem Betrieb der SOA.

Entgegen dem bisherigen Verständnis von Services müssen Services um einen Bestandteil zur Unterstützung der ITSM-Prozesse erweitert werden. Dieser Bestandteil dient der Unterstützung der allgemein als Standard anerkannten ITSM-Prozesse, wie in Abbildung 5.2 hervorgehoben, dargestellt (vergleiche Abbildung 2.1). ITIL beweist, dass es möglich ist, die Prozesse, Rollen, Tätigkeiten und Verantwortlichkeiten im IT Service und Supportmanagement allgemeingültig und standardisiert auf einem technologieunabhängigen Abstraktionsniveau zu beschreiben.

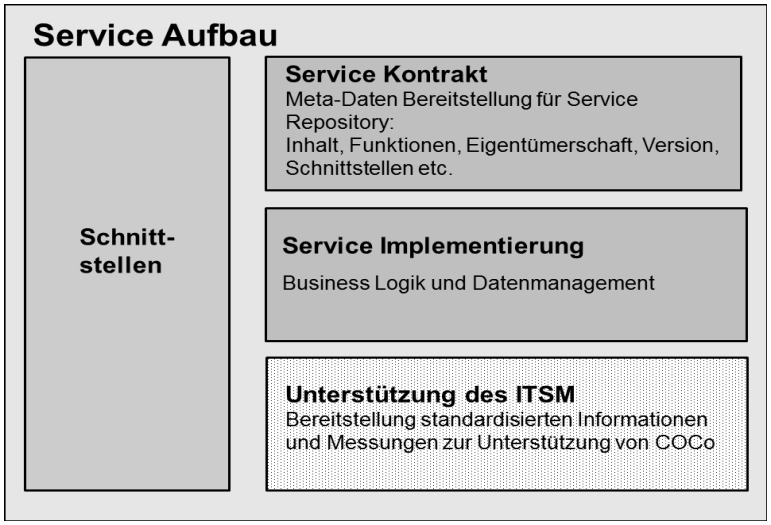

Abbildung 5.2: Erweiterung der Bestandteile von Services

Die Umsetzung der von ITIL beschriebenen optimalen Vorgehensweisen erfordert jedoch eine genauere Spezifizierung unter den konkreten technischen Aspekten, wie z.B. der in dieser Arbeit im Mittelpunkt stehenden SOA. Trotz der in Kapitel 4 beschriebenen veränderten Bedingungen im Betrieb einer SOA-basierten Lösung ist es möglich und erforderlich, ein Werkzeug zu erstellen, das den Betrieb unabhängig von der Service-Zusammensetzung in SOA erlaubt. Neben dem zentralen Ansatz muss ein solches Werkzeug die bereits als feste Bestandteile bekannten Bausteine einer SOA unterstützen, ausnutzen und mit ihnen kooperieren. Gleichzeitig müssen aber auch Service Repository und Service Bus den zentralen Betriebsservice geeignet unterstützen.

Der nächste Schritt ist die Standardisierung der technischen Umsetzung dieser Prozesse, wie in Abbildung 5.3 dargestellt.

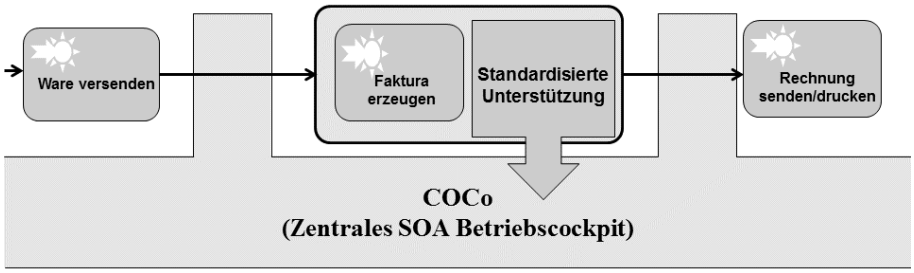

Abbildung 5.3: Zusammenspiel mit dem zentralen Betriebscockpit (COCo)

Im Abschnitt 4.4 wurde gezeigt und nachgewiesen, dass serviceunabhängige Qualitätskriterien in Form von Metriken definiert werden müssen. Im

Folgenden wird nun gezeigt, was dazu erforderlich ist und welche Anforderungen sich für die Schlüsselelemente der SOA ergeben.

5.2 Erweiterung des Service Repository

Das Service Repository dient der Verwaltung der Metadaten von Services. Es ist von entscheidender Bedeutung für die zentrale Bereitstellung von Informationen über die eingesetzten und darüber hinaus bereitgestellten Services. In der SOA-Landschaft. Zwar wurde von der Oasis als Beschreibungssprache für die im Service Repository zu verwaltenden Informationen UDDI [Oasis, 2004] definiert, doch welche Informationen im Detail ein Service Repository umfassen soll, wurde bisher nicht standardisiert. [Buchwald et al., 2010] untersucht die Anforderungen an ein SOA-Repository, um eine Austauschbarkeit von Services realisieren zu können. Jegliche Anforderungen, die sich aus dieser Austauschbarkeit für den Betrieb an ein Service Repository ergeben, wurden außer Acht gelassen. Das Service Repository ist der Grundstein für die Nutzung der Flexibilität von SOA. Die Herausforderungen bei der Nutzung der Flexibilität von SOA bestehen zunächst darin, überhaupt herauszufinden, welche Services in einer Landschaft verfügbar sind, welche eingesetzt werden und welche gleichwertig sind in Bezug auf eine Lösung. Genau diesen Aspekt untersucht [Buchwald et al., 2010]. In der Studie wird zwischen fachlichen und technischen Services unterschieden, was die Struktur einer Lösung vereinfacht. Tatsächlich müssen die Services hinsichtlich der Schichten unterschieden und ihre Beziehungen dargestellt werden (vergleiche auch Abbildung 2.2). Die Vereinfachung bei [Buchwald et al., 2010] genügt jedoch den Anforderungen der Untersuchung zu den Anforderungen an ein Service Repository unter dem Aspekt der Austauschbarkeit von Services. Sie liefert zunächst grundlegende Anforderungen, die aus Betriebssicht aber noch weiter ergänzt werden müssen. Im Ergebnis werden bei [Buchwald et al., 2010] folgende acht Anforderungen an das Service Repository (SR) formuliert, die hier wiedergegeben und entsprechend kommentiert und gegebenenfalls gemäß den Erkenntnissen in dieser Arbeit angepasst werden:

In der Anforderung 1 verlangt Buchwald, dass ein Service erst dann genutzt werden kann, wenn er im Service Repository publiziert und der Service Kontrakt gespeichert wurde, siehe [Buchwald et al, 2010] „Bevor ein Service genutzt werden kann, sollte er in einem SOA-Repository publiziert werden. Anschließend kann ein potentieller Nutzer im Repository nach Services suchen, z.B. unter Verwendung von Suchkriterien wie angebotene Funktionalität oder verwendete Datenobjekte.“ Für den Betreiber ist es jedoch ebenso relevant, zu jedem Zeitpunkt ermitteln zu können, ob ein Service genutzt wird, d.h., in eine Lösung eingebunden ist. Grundsätzlich ist

ebenfalls eine Nutzungsvereinbarung von Bedeutung, die die Rechte und Pflichten eines Nutzers des Services umfasst, die aber in dieser Arbeit nicht weiter verfolgt wird. Im Ergebnis dieser Arbeit kann diese von Buchwald formulierte Anforderung aus Betriebssicht nur bestätigt werden.

SR-Anforderung 1: Zentrale Verwaltung der Service Publikation und des Kontrakts

Jeglichen Aktivitäten zum Betrieb der Lösung muss die Information zur Verfügung stehen, welche Services wo genutzt werden.

Aus Sicht von [Buchwald et al., 2010] ist es in der Praxis wesentlich, innerhalb des Repositorys Domänen zu definieren, was zu seiner Anforderung 2 „*Domänen zur Strukturierung*" des Service Repositorys führt. Dies folgt dem Gedanken, dass innerhalb einer Firma organisatorisch weitestgehend unabhängige Organisationseinheiten existieren und dem folgend mittels Domänen innerhalb des Service Repositorys ähnlich unabhängige Einheiten definiert werden können. Aus Sicht der hier vorliegenden Untersuchungen basiert die Anforderung von Buchwald auf rein organisatorischen Gründen und ist keine Voraussetzung für die Austauschbarkeit von Services noch für die Betreibbarkeit. Im Gegenteil würde das Service Repository in unabhängige Domänen unterteilt, müssten die IT-Prozesse in der Folge der Domänenstruktur folgen. Die Nutzung eines Service in unterschiedlichen Domänen ist aus Betriebssicht kritisch, da z.B. administrative Aufgaben immer in Bezug auf den Service gültig sind und nicht dem Domänengedanken folgen können. Aus Betriebssicht muss daher mit unabhängigen Versionen oder Kopien der Services in den Domänen gearbeitet werden. [Buchwald et al., 2010] führt dies zu der Frage, wie viele Service Repositorys erforderlich sind, findet jedoch keine eindeutige Antwort. Wie hier dargestellt, würde die Strukturierung des Service Repository in unabhängige Domänen auf Grund von ebenso unabhängigen Organisationseinheiten und daher folglich auch unabhängigen Geschäftsprozessen zu getrennten Versionen gleicher Services und ihrer unabhängigen Administration führen. Daher wird im Gegensatz zu Buchwald für ein zentrales, integriertes Service Repository plädiert, auch wenn verschiedene relativ unabhängige Organisationseinheiten existieren, die aber gleiche Services nutzen.

Auf Grund der Ergebnisse in dieser Arbeit wird die Anforderung 2 von Buchwald daher angepasst, wie folgt.

SR-Anforderung 2: Zentralisierung des Service Repository

In der Anforderung 3 zur „*Informationserzeugung und –verwendung*" werden von Buchwald fachliche als auch technische Spezifikationen eines Services verstanden, wie auch Dokumentationen der Funktionalität, die im Service Repository für die Services bereitgestellt werden.

Welche Spezifikationen insbesondere für den Betrieb einer Lösung auf Basis der genutzten Services verfügbar sein sollten, wird in der Untersuchung von [Buchwald et al., 2010] wiederum nicht gesondert untersucht. Daher ist auch diese Anforderung zu erweitern:

SR-Anforderung 3: Informationserzeugung und -verwendung zur Unterstützung des Betriebs

Was das im Detail heißt, wird in den folgenden Anforderungen eingehender verdeutlicht. An dieser Stelle wird festgestellt, dass der mögliche Austausch von Services nicht nur zu Anforderungen an das Service Repository aus funktionaler Sicht der Lösung führt, sondern auch entsprechende Informationen für den Betrieb bereitgestellt werden müssen.

Aus Sicht von [Buchwald et al., 2010]. durchlaufen Services verschiedene Zustände innerhalb ihres Lebenszyklus. Buchwald plädiert in der Anforderung 4 „Service-Lifecycle-Management“ dafür, diese Zustände im Service Repository zu hinterlegen. Typische Zustände wären *in der Entwicklung* oder *Freigegeben*. Aus Sicht des Betriebs gewinnt diese Anforderung an Bedeutung, wie in Abschnitt 4.5 beschrieben.

SR-Anforderung 4: Zentrale Informationsverwaltung für das Service-Lebenszyklus-Management

Insbesondere in den Abschnitten 4.5 und 4.6 wurden die Änderungs-management und Release Management Prozesse als Teil des Lebenszyklus-Managements hinsichtlich der Betreibbarkeit unter dem Aspekt der Service-Austauschbarkeit untersucht. Es wurde gezeigt, dass Änderungen an Services einer zentralen Steuerung bedürfen, die nicht abhängig von der Service Technologie sein darf. Änderungs- und Release Management müssen über die verschiedenen Lebensphasen der Entwicklung eines Services hinweg gesteuert werden, was auch die Überführung von einem Zustand in den nächsten umfasst. Daraus ist zu schlussfolgern, dass die Zustände eines Services hinterlegt werden müssen. Aus Sicht des Betriebs ist es jedoch nur von Bedeutung, dass diese Information verfügbar ist. Sie muss nicht zwingend im Service Repository vorgehalten werden. Diese Frage wird in der Anforderung 10 noch einmal aufgegriffen und diskutiert.

Die Speicherung der Beziehungen zwischen Repository-Objekten fordert Buchwald in seiner Anforderung 5 „*Speicherung der Beziehungen zwischen Repository-Objekten“*. Um Services austauschen zu können, muss es eine Informationsquelle geben um festzustellen, welcher Service zur Laufzeit welche anderen Services nutzt bzw. von welchen genutzt wird. Es bietet sich an, derartige Informationen im Service Repository zu hinterlegen. Diese Anforderung wird durch die Ergebnisse dieser Arbeit auch aus Betriebssicht gestützt und wird übernommen.

SR-Anforderung 5: Speicherung der Beziehungen zwischen Repository-Objekten

Insbesondere durch das erforderliche Monitoring auf Basis der Prozesse und Services ergibt sich die Notwendigkeit, die Service-Verknüpfungen zur Laufzeit ermitteln zu können.

In der Anforderung 6 „*Service-Versionen und Plantermine*“ unterstreicht Buchwald die Wichtigkeit der Service-Versionen und Plantermine. Für die Services ist eine Versionsverwaltung erforderlich, die es gestattet, neue Versionen zu planen bzw. veraltete gegebenenfalls abzuschalten. Diese Anforderung steht in direktem Zusammenhang mit dem Release Management und der SR-Anforderung 4. Aus Betriebssicht muss diese Anforderung jedoch noch weiter gehen und wird daher, wie folgt, formuliert:

SR-Anforderung 6: Verwaltung und Auswertbarkeit der Service-Versionen und Plantermine

Insbesondere müssen auftretende Statusänderungen oder Störfälle in Zusammenhang mit Änderungen an Service-Versionen gesetzt werden können. Dazu ist der zeitliche Zusammenhang zwischen Versionswechseln einzelner Konfigurationselemente, wie Services, und aufgetretenen Störfällen zu bilden.

In der Anforderung 7 „*Langfristige Archivierung von Prozess- und Service-Modell*en“ formuliert Buchwald die Notwendigkeit der langfristigen Archivierung von Prozess- und Service-Modellen. Die vorliegenden Untersuchungen unterstreichen diese Anforderung. Zwecks Nachvollziehbarkeit ist eine Speicherung der fachlichen und technischen Abläufe und Zusammenhänge erforderlich. Diese Anforderung kann aus Sicht des Betriebs insbesondere aus Sicht des Änderungs- und Problem Management weiter spezifiziert werden. Die im Service Repository hinterlegten Informationen müssen es ermöglichen, durchgeführte Änderungen jeglicher Art auszuwerten und so die Ursachenanalyse im Problem Management unterstützen. Die hinterlegten Informationen müssen es mit möglichst wenig Aufwand gestatten, Zusammenhänge zwischen Änderungen und Störfällen zu erkennen bzw. auszuschließen.

SR-Anforderung 7: Langfristige Archivierung von Prozess- und Service-Modellen zur Herstellung von zeitlichen Zusammenhängen zwischen Störfällen und Änderungen

Gemäß Buchwalds Anforderung 8 „Beziehungen zwischen fachlichen und technischen Aktivitäten“ müssen die Beziehungen zwischen fachlichen und technischen Aktivitäten im Service Repository gespeichert werden. Es wird gefordert, dass im SOA-Repository einzelne Aktivitäten der fachlichen und technischen Prozessmodelle und deren Beziehungen verwaltet werden im

Sinn einer Dokumentation, um die Nachvollziehbarkeit zu erhöhen. Auch diese Anforderung kann bestätigt werden. Diese Informationen unterstützen maßgeblich, die Anpassungsfähigkeit einer Lösung. Dies ermöglicht die Informationsfindung im Falle, dass sich fachliche Anforderungen ändern und Services ausgetauscht werden müssen oder sollten. Nur mit Hilfe dieser Informationen kann entschieden werden, ob ein Service gegen einen anderen, verfügbaren austauschbar ist.

SR-Anforderung 8: Verwaltung auswertbarer Beziehungen zwischen fachlichen und technischen Aktivitäten

In der hier vorgestellten Untersuchung von Buchwald stand lediglich die Frage im Mittelpunkt, welche Informationen im Service Repository verfügbar sein müssen, um überhaupt feststellen zu können, inwiefern Services austauschbar sind und welche Auswirkungen dies auf eine Lösung hat. Die so von ihm formulierten Anforderungen sind valide und wurden durch die vorliegende Untersuchung bestätigt oder geringfügig erweitert. Aus dem Gesichtspunkt des Betriebs und der Nutzung der Flexibilität von SOA zeigt sich, dass diese Anforderungen notwendig, jedoch nicht hinreichend sind. Neben weiteren Anforderungen an das Service Repository im Wechselspiel mit COCo müssen auch die anderen Bestandteile einer SOA, wie Services und Service Bus Anforderungen erfüllen.

Alle hier gegebenen Anforderungen sind nur mit Unterstützung der Services zu erfüllen, d.h., die Informationen, die im Service Repository verwaltet werden sollen, müssen von den Services in der SOA entsprechend an das Service Repository geliefert werden.

Damit ergeben sich an die Services folgende Anforderungen, bezeichnet mit S-Anforderung:

S-Anforderung 1: Vollumfängliche Unterstützung des Service Repository zur Erfüllung der Anforderungen des Typs SR

Für das Service Repository wurden die Anforderungen 1-8 formuliert. Dies setzt allerdings voraus, dass die vom Service Repository zu verwaltenden Informationen von den Services bereitgestellt werden.

Des Weiteren lässt sich daraus die entsprechende Anforderung an COCo ableiten:

COCo-Anforderung 1: Vollumfängliche Unterstützung und Aufbereitung der im Service Repository verwalteten Informationen

Die im Service Repository verwalteten Daten lassen sich nur dann sinnvoll für den Lösungsbetrieb verwenden, wenn COCo die für den jeweiligen Betriebsprozess erforderlichen Informationen aufbereitet und in einer geeigneten Sicht darstellt. Da das Service Repository gemäß Anforderung 2

ein zentrales Element in der SOA darstellt, kann COCo durch den Zugriff auf das Service Repository auf alle gespeicherten Informationen zu den eingesetzten Services zugreifen. Für Service Repositorys existiert bereits UDDI als allgemein akzeptierte Beschreibungssprache, was die Auswertung der gespeicherten Daten mittels COCo vereinfacht. Die im Service Repository hinterlegten Informationen müssen dabei unabhängig von der Technologie eines Service sein. Ebenso unabhängig von der Technologie eines Service muss die Darstellung der Auswertungen in COCo erfolgen, so dass die Betriebsaufgaben und Entscheidungen unabhängig vom eigentlichen Service und insbesondere seiner Technologie erfolgen und darauf aufbauend ausgeführt werden können.

5.3 Änderungs- und Release Management

Betrachtet man den Änderungsmanagement Prozess in vereinfachter Form, so lässt er sich gemäß Abbildung 5.4 darstellen. Organisatorische Aspekte wurden weitestgehend vernachlässigt, da sie nicht im Fokus dieser Arbeit stehen. Den Ausgangspunkt bildet die Änderungsanforderung, gemäß ITIL *Request for Change* (RfC). Eine Änderungsanforderung kann durch den Problemmanagement Prozess oder durch die Evolution der Nutzung initiiert werden.

In der Phase der Evaluierung der gewünschten Änderungen ergeben sich im SOA-Umfeld neue Herausforderungen. ITIL geht davon aus, dass Änderungen immer an einem bestehenden technischen Objekt durchgeführt werden. Durch SOA ist eine neue Art von Änderung möglich: der Austausch eines Services gegen einen hinsichtlich der Lösung gleichwertigen Service oder die bewusste Anpassung einer Lösung, in dem ein Service gegen einen zur Lösung ähnlichen Service ausgetauscht wird. Um dies zu ermöglichen, müssen geeignete Informationen verfügbar sein.

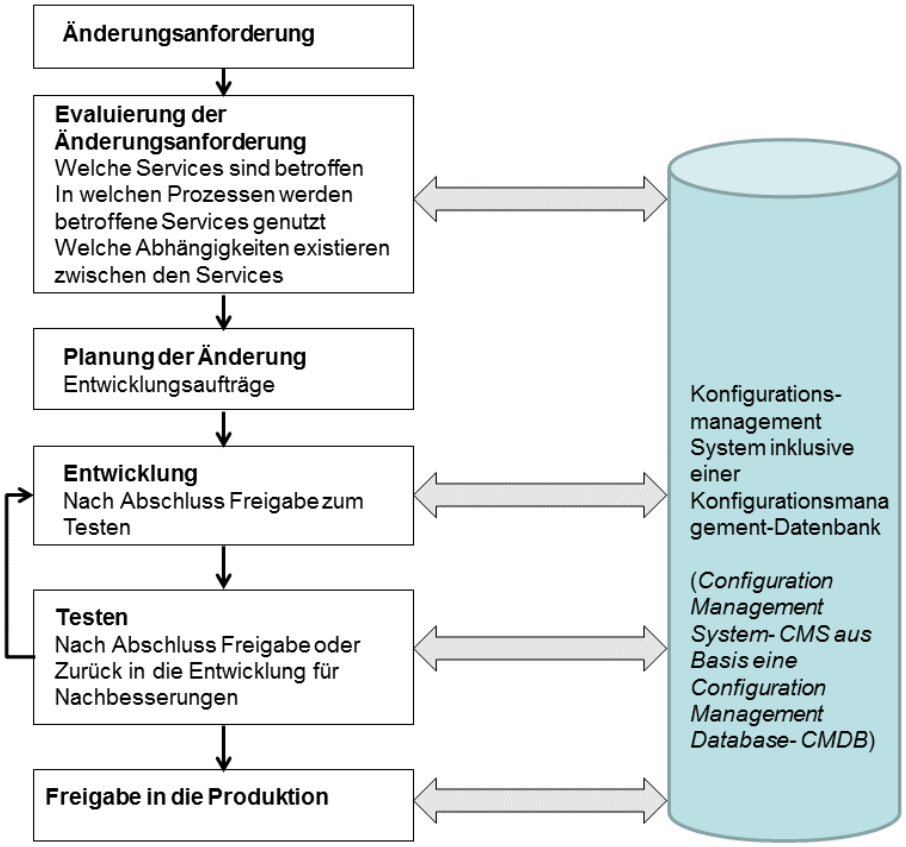

Abbildung 5.4: Vereinfachte Darstellung des Änderungsmanagement Prozesses

Im Abschnitt 4.5 wurden die sich aus der gewünschten Flexibilität und dem Betrieb ergebenen Änderungen und zu lösenden Probleme bzw. Herausforderungen dargestellt. Es kristallisierten sich folgende Schwerpunkte aus Betreiber-Sicht heraus:

- Zentrale Steuerung und Kontrolle von Änderungen und Releases;
- Zentrale Ausführung in dem Sinne der gesteuerten Übernahme aus der Entwicklungs- bzw. Testumgebung in die produktive Lösung und dortige Aktivierung (Deployment) unabhängig von der technologischen Basis eines Services;
- lösungsorientiertes Deployment, d.h. Synchronisation von abhängigen Änderungen in Bezug auf eine Lösung über mehrere Services hinweg;

- Service-übergreifendes, Lösungs-orientiertes Reporting über alle Änderungen, die in Bezug auf eine Lösung und ihrer Services vorgenommen wurden;
- Nachverfolgung und Nachvollziehbarkeit von Änderungen einer Lösung.

Im Release Management werden einzelne, womöglich voneinander abhängige Änderungen organisatorisch zu einem Release zusammengefasst und verwaltet. In SOA ist das Service Repository der zentrale Baustein für die Bereitstellung von Metadaten zu den Services. Für das Änderungs- und Release Management ist damit das Wechselspiel mit dem Service Repository von besonderer Bedeutung, da dieses alle notwendigen Informationen über einen Service bereitstellt. Im Abschnitt 5.2 wurden bereits Anforderungen an das Repository formuliert. Zusätzlich zu den dort bereits beschriebenen Anforderungen ergeben sich weitere Anforderungen.

SR-Anforderung 9: Integration von CMDB und Service Repository zu einem zentralen Baustein

Änderungs- und Release Management gemäß ITIL erfordern die Pflege einer Konfigurationsmanagement-Datenbank (CMDB) als Grundlage des Konfigurationsmanagement Prozesses. Gleichzeitig stellt das Service Repository den zentralen Baustein für Metadaten zu Services dar. Gemäß Anforderung 6 und Anforderung 4 an das Service Repository ergibt sich zur Vermeidung doppelter Datenhaltung die Notwendigkeit, CMDB und Service Repository miteinander zu verschmelzen.

Die Realisierung des Änderungsmanagement erfordert zudem Entwicklungsumgebungen und auch Testumgebungen, um die Entwicklung von Änderungen unabhängig und risikofrei für die produktive Lösung durchzuführen. Da die Überführung einer fertiggestellten Änderung in die Produktion eine Betriebsaufgabe ist, benötigt der Betrieb übergreifende Informationen über z.B. die Entwicklungsumgebung und die Testumgebung einer Lösung und der darin genutzten Services. Daraus ergibt sich die folgende Anforderung an das Service Repository.

SR-Anforderung 10: Zentrales Service Repository als integrierender Baustein zwischen den verschiedenen Zuständen des Lebenszyklus einer SOA-basierten Lösung

SR-Anforderung 10 ist eine Erweiterung der SR-Anforderung 4 und SR-Anforderung 5. Es ist nicht hinreichend, nur die Informationen über den Status von Services innerhalb des Lebenszyklus zu verwalten und die Relationen zwischen Services, sondern mögliche Entwicklungs- oder Testumgebungen müssen als virtuelle Einheit analog der produktiven Umgebung hinterlegt werden. Nur auf dieser Basis kann die Überführung eines Services von z.B. dem Zustand *in Entwicklung in Freigegeben* geändert und aus der Entwicklungs- oder Testumgebung in die produktive

Umgebung überführt werden. Das Service Repository muss also nicht nur zentral hinsichtlich der produktiven Umgebung einer SOA-basierten Umgebung sein, sondern zentral hinsichtlich der verschiedenen Umgebungen entlang des Lebenszyklus einer Lösung, wie in Abbildung 5.5 am Beispiel des Kundenauftragsprozesses und des darin enthaltenen Services *Globale Verfügbarkeitsprüfung* dargestellt. Die Umsetzung der Nutzungsfreigabe des Services, d.h., die Überführung aus der Testumgebung in die produktive Umgebung erfolgt mittels des zentralen Betriebsservice COCo im Zusammenspiel mit dem Service Repository. Das Service Repository ist der zentrale Informationsspeicher aller Metadaten der Services in allen Status des Lebenszyklus. In diesem Beispiel wurde lediglich von einer Entwicklungsstufe und einer Testumgebung ausgegangen. Analog kann so auch ein mehrstufiger Lebenszyklus realisiert werden.

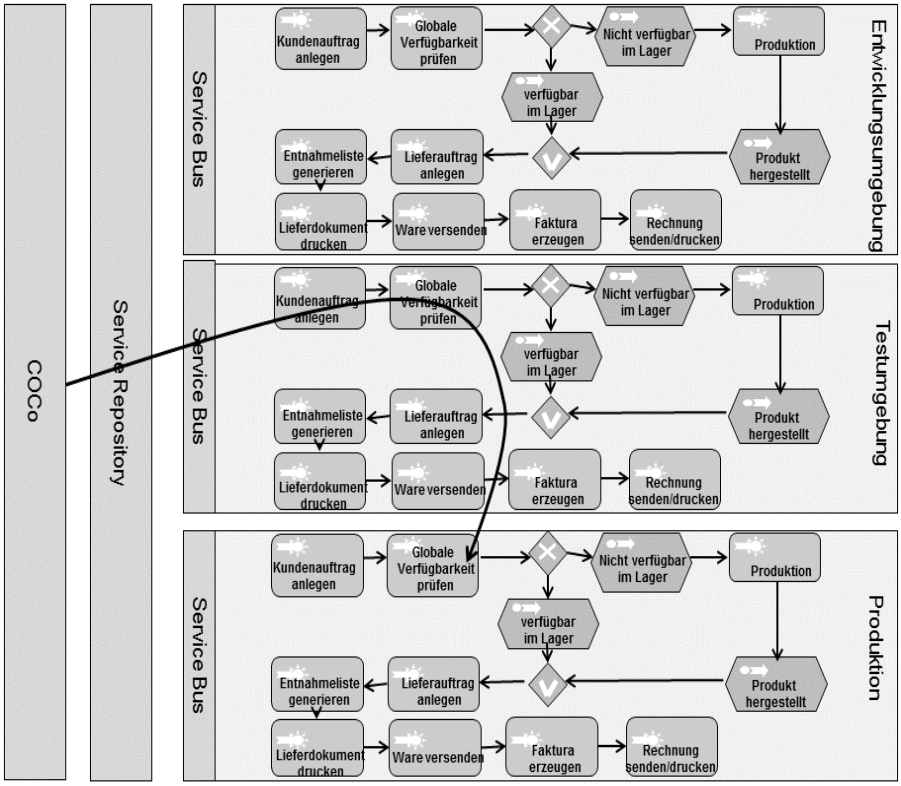

Abbildung 5.5: Kontrollierte Überführung eines getesteten Services 'Globale Verfügbarkeitsprüfung' aus der Testumgebung in die Produktion

Es wird ausdrücklich darauf hingewiesen, dass die Forderung eines zentralen Repositorys nicht ausschließt, dass es je Entwicklungsumgebung und -stufe weitere, lokale Repositorys gibt. Denkbar ist auch ein hierarchischer Aufbau

von Service Repositorys, die je Umgebung separiert sind, aber zu einem zentralen Repository in der Wurzel integriert sind. Für den produktiven Betrieb einer Lösung müssen alle Informationen über Services in allen Umgebungen zentral in einem Repository verfügbar sein.

Eben diese Notwendigkeit der zentral koordinierten Überführung eines Services von einem Lebenszyklus-Status in einen anderen und die damit verbundene physische Überführung von einer Lebenszyklus-Umgebung in eine andere führt auch zu Anforderungen an Services als auch dem zentralen Betriebsservice.

S-Anforderung 2: Standardisierte Schnittstelle für Wechsel der Lebenszyklus-Umgebung (Deployment)

Wie bereits dargestellt, kann die Einführung eines universellen, zentralen Betriebsservice COCo nur mit einer auch teilweisen Standardisierung der Services einhergehen. Würde jeder Service in Abhängigkeit von Hersteller, Technologie etc. eigene Werkzeuge und Informationen liefern, führt dies automatisch zu höheren Aufwänden beim Betrieb und Abhängigkeiten von einzelnen Services. Services sind nur dann flexibel austauschbar in einer SOA-basierten Lösung, wenn sie mit möglichst wenig Aufwand aus Betreibersicht austauschbar sind. Aus Sicht der Freigabe einer Serviceänderung, das heißt dem Statuswechsel in Bezug auf den Lebenszyklus, ist es erforderlich, dass die tatsächliche technische Realisierung gekapselt wird. Der Statuswechsel eines Services z.B. aus der Testumgebung in die Produktivumgebung wird durch COCo initiiert. Dafür muss jeder Service eine Schnittstelle bereitstellen. Um die Unabhängigkeit von COCo von den Spezifika eines Services zu gewährleisten, wird in dieser Untersuchung eine Kapselung mittels Nutzung des Service Bus (SB) vorgeschlagen, dessen Aufgabe ohnehin die Kontrolle, der Transfer und möglicherweise die Umwandlung der Nachrichten zwischen den Services ist [Papazoglu und Heuvel, 2007] „[…] is responsible fort he proper control, flow, and translation of all messages between services, using any number of possible mesagging.]

SB-Anforderung 1: Standardisierte Adapter für die Service-Deployment-Schnittstelle

Auf der anderen Seite erfordert ein standardisiertes Deployment von Services auch eine universelle, standardisierte Schnittstelle des zentralen Betriebsservices COCo mit Hilfe derer der Statuswechsel eines Services gesteuert und vollzogen werden kann.

COCo-Anforderung 2: Standardisierte Schnittstelle zur Umsetzung von Statuswechsel von Services (Deployment-Schnittstelle)

Sämtliche Informationen zu der Deployment-Schnittstelle als auch über alle anderen Schnittstellen eines Services sind im Service Repository vorzuhalten.

SR-Anforderung 11: Speicher für die Informationen zu den Schnittstellen der Services

Ein Service kann hinsichtlich einer Lösung nur dann als gleichwertig zu einem anderen Service hinsichtlich einer Lösung sein, wenn der Service bei gleichen Eingabewerten gleiche Ergebnisse liefert. Damit ein Service in dem für ihn passenden Eingabeformat mit Werten versorgt werden kann, müssen die genauen Beschreibungen der Schnittstellen in der SOA verfügbar sein. Da das Service Repository alle Metadaten zu den Services bereitstellt, ist es auch dafür prädestiniert, die Schnittstelleninformationen bereit zu stellen und zwar so, dass der Service Bus diese Informationen lesen und die zu übergebenen Daten in dem erforderlichen Eingabeformat aufbereiten kann bzw. nach der Ausführung des Services die ausgegebenen Daten abholen und für den Folgeservice aufbereiten kann. Damit ergibt sich:

S-Anforderung 3: Serviceverantwortlichkeit für die Service-Schnittstelleninformation im Service Repository

Jeder Service muss seine Schnittstelleninformationen in standardisierter Weise im Service Repository hinterlegen. Um eine solche Beschreibung zu hinterlegen, bietet sich wiederum UDDI als Beschreibungssprache an. Die Aufbereitung, Auswertung und Nutzung dieser Informationen obliegt dem Service Bus.

SB-Anforderung 2: Aufbereitung der Eingabe- und Ausgabewerte eines Services gemäß den im Service Repository hinterlegten Schnittstellenbeschreibungen

Jegliche Art von Änderungen, egal ob einzelner oder komplexer Natur wie bei einem Release, erfordert das Testen.

Bevor aber eine Änderung oder ein Release in die Produktion übernommen werden können, müssen die Änderungen getestet und bei Erfolg freigegeben werden.

5.4 Testen

Bisher wurde genaugenommen lediglich der produktive Betrieb bzw. die Nutzung einer SOA-basierten Lösung betrachtet. Neben dem täglichen Betrieb einer solchen Lösung gehört aber auch die Governance der Lösung zu den regelmäßigen Aufgaben. Die Umsetzung von Änderungen parallel

zur täglichen produktiven Nutzung einer Lösung gehört ebenso zu den täglichen Betriebsaufgaben, wenngleich die Planung, Ausführung und das Testen von Änderungen anderen Phasen des Lebenszyklus einer Lösung zuzurechnen sind. Das Testen ist unabdingbar mit Änderungen verknüpft. Flexibilität bedeutet ein hohes Maß an Änderungen und auch Änderungsmöglichkeiten, die mit möglichst wenig Aufwand aber mit umso größerer Sicherheit durchgeführt werden müssen.

Wie in Abschnitt 4.8 dargestellt, kristallisieren sich zwei Problemschwerpunkte heraus, soll die Flexibilität auf Basis des Service Austauschs nutzbar sein. Dies sind:

- das serviceorientierte Testen und
- die Schaffung einer geeigneten Testumgebung.

Um zu entscheiden, ob ein in der produktiven Landschaft verfügbarer Service gegen einen anderen gefahrlos ausgetauscht werden kann, sollten entsprechende Konstellationen getestet worden sein. Es wird als gegeben vorausgesetzt, dass die bisher formulierten Anforderungen insbesondere an das Service Repository uneingeschränkt erfüllt sind und die so in der SOA verfügbaren Informationen und die über den zentralen Betriebsservice vorgenommenen Auswertungen 100% korrekt und vollständig sind. Die darauf aufbauenden Schlussfolgerungen und Entscheidungen seien daher ebenso valide.

Die Qualität eines Tests ist eng damit verknüpft, wie übertragbar Testergebnisse in einer Testumgebung auf die produktive Umgebung sind. Dies führt zu der Frage, wie eine geeignete Testumgebung geschaffen werden kann.

Gemäß den bereits dargestellten Anforderungen an das Service Repository enthält das Service Repository auch die in der Landschaft verfügbaren Versionen von Services in ihren verschiedenen Lebenszyklen.

5.4.1 Klassifizierung von Änderungen

Wie dargestellt, ergeben sich aus Art und Umfang einer Änderung entsprechend unterschiedliche Tests, die in Bezug auf einen Service durchgeführt werden. Im Folgenden werden daher Änderungen hinsichtlich ihres Umfangs und des erforderlichen Testumfangs unterschieden. Welche Tests erforderlich sind, hängt maßgeblich von der Art der durchgeführten Änderungen in Bezug auf die oder den Service ab. Grundsätzlich lassen sich Änderungen wie folgt klassifizieren:

Definition 5.2: Innere Änderung

Unter einer *Inneren Änderung* wird eine Änderung verstanden, die einen Service lediglich hinsichtlich der Verarbeitung ändert. Die Schnittstellen des Services bleiben unverändert.

Es wird davon ausgegangen, dass gleiche Eingabewerte gleiche Ausgabewerte erzeugen. Eine solche Änderung hat somit keine Auswirkungen auf andere Services. Die geänderte Version eines Services A sei A´. Dann sind Service A und Service A` gleichwertig hinsichtlich einer Lösung. Eine solche Änderung erfordert lediglich die Anpassung der Versionsverwaltung innerhalb des Service Repositorys. In diesem Fall ist es notwendig und hinreichend ein Unit Testing durchzuführen. Die zu testende Einheit ist der Service.

Definition 5.3: Äußere Änderung

Eine Änderung wird *äußere* genannt, wenn sie einen Service nach außen hin sichtbar verändert, d.h. es ändern sich Eingabe- und/oder Ausgabewerte oder die Funktionalität eines Services.

Dies kann entsprechende Änderungen im Service Repository nach sich ziehen. Um derartige Änderungen zu testen, sind neben entsprechenden Unit Tests in jedem Fall auch Integration Tests erforderlich.

Definition 5.4: Komplexe Änderung

Eine *komplexe Änderung* ist eine Änderung, die sich aus beliebigen inneren und äußeren Änderungen an mehreren Services zusammensetzt, die untereinander abhängig sind.

Dadurch ist eine Änderung selbst als Einheit zu betrachten und muss in dieser Gesamtheit getestet und auch den Status wechseln, also zum Beispiel freigegeben werden. Eine komplexe Änderung erfordert umfangreichere Tests, wie Unit Tests der geänderten Services, Integrationstests und zwar funktionaler und nicht-funktionaler Art. Regressionstests in Bezug auf den Prozess und auf die Services können helfen, den Aufwand zu reduzieren.

In Abhängigkeit von der Art der Änderung stehen die erforderlichen Tests mit den dafür entsprechend aufzubauenden Testumgebungen.

5.4.2 Testumgebung

Aus der Art der Änderung ergeben sich drei prinzipielle, zu testende Einheiten:

- ein einzelner Service,
- der einzelne Service und seine Vorgänger- und Nachfolgeservices,
- alle von einer komplexen Änderung betroffenen Services und deren Zusammenspiel in den Lösungen.

Das Testen erfordert immer eine geeignete Testumgebung. Das heißt, dass die in der Testumgebung gewonnenen Ergebnisse Rückschlüsse auf die Qualität der getesteten Services in der produktiven Lösung und Umgebung erlauben. Der einfachste Ansatz eine solche Testumgebung aufzubauen, ist die Kopie der produktiven Umgebung inklusive gleicher Hardware und Daten aus der Produktion. Also faktisch der Aufbau der Testumgebung identisch zur Produktion. Kosten und Unzulänglichkeiten wurden bereits in 4.8 diskutiert. Der Übergang zu SOA eröffnet auch hier neue Möglichkeiten bzw. führt zu Herausforderungen. Betrachtet man den Aufbau einer SOA, so kommt dem Service Bus eine tragende Rolle bei der Inter-Service-Kommunikation zu. Der Service Bus stellt einem Service die erforderlichen Eingabedaten gemäß der im Service Repository vorliegenden Schnittstelleninformationen zur Verfügung und nimmt die Ausgaben des Services wiederum entgegen, um sie gegebenenfalls für den Nachfolgeservice aufzubereiten. Damit ist der Service Bus prädestiniert, das Testen zu unterstützen.

Im Abschnitt 4.8.1 wurde bereits auf den Vorschlag von Hooks [Ribarov et al., 2007] als Erweiterung des Service Bus eingegangen und auch die Unzulänglichkeiten aufgezeigt. Die Realisierung derartiger Hooks verlangt mindestens die Erfüllung der an dieser Stelle bereits formulierten Anforderungen an Services S1-S3, Service Repository SR1-SR11 und einen zentralen Betriebsservice COCo 1-2. Abbildung 5.6 zeigt, wie das Testen eines einzelnen Services, wie *Lieferauftrag anlegen*, mit Hilfe eines Hooks realisiert werden kann. Gemäß SB-Anforderung 1 und SR-Anforderung 11 verfügt der Service Bus über alle Informationen hinsichtlich der Schnittstellen eines Services. Zusammen mit der im Service Repository hinterlegten Information, welche Services zur Laufzeit einem Service folgen (*Entnahmeliste generieren*) bzw. vorangehen (*verfügbar im Lager* und *Produktioin*) und zuliefern, kann der Datenfluss eines produktiven Ablaufs simuliert werden. Damit ergibt sich eine Möglichkeit, einen einzelnen Service zu testen.

SB-Anforderung 3: Simulation der Datenübergabe von Vorgänger- und Nachfolgeservices gemäß der im Service Repository verfügbaren Schnittstelleninformationen

Eine unabhängig von der Lösungsarchitektur bestehende Herausforderung ist die Bereitstellung geeigneter Testdaten. Wie bereits festgestellt, ist das einfache Kopieren der Daten in den Datenbanken nur bedingt realisierbar. Neben der normalerweise großen Datenmenge spielen Fragen der Datensicherheit und Vertraulichkeit eine ausschlaggebende Rolle. Das Sicherheitsrisiko entsteht durch den beim manuellen Testen erforderlichen Zugriff auf die Kopien von produktiven und damit eventuell vertraulichen Daten. Dieses Problem wird daher umgangen, indem mit Hilfe von

speziellen Programmen aus ehemals produktiven Daten verfälschte Testdaten erzeugt werden, wie in Abschnitt 4.8.2 vorgestellt.

Diese Technik ist genauso in einer SOA-basierten Umgebung einsetzbar.

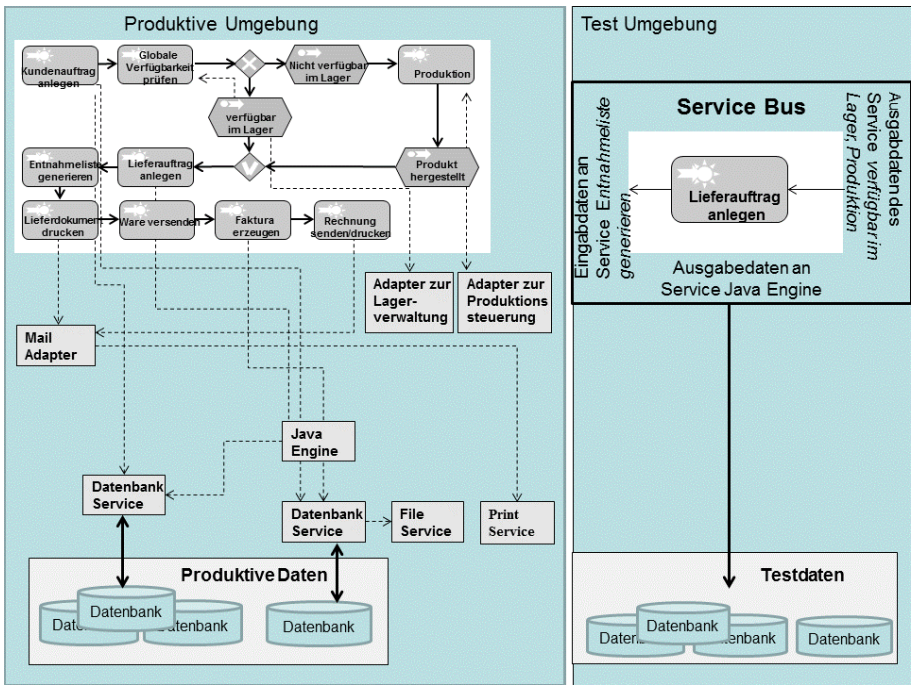

Abbildung 5.6: Unit Testing des Servicse Lieferauftrag anlegen mit Hilfe des Service Bus

Weitaus vorteilhafter ist jedoch die Automatisierung der Tests, für die durch eine Erweiterung der dargestellten Strategie der Hooks neue Möglichkeiten entstehen Zu dieser Schlussfolgerung gelangt auch [Ribarov et al, 2007] „[…] automation of the testing process, as one of the purposes of the SOA applications is fast development […]. Das Testen ist ein wesentlicher Zeitfaktor bei der Entwicklung und Freigabe neuer Services. In Abbildung 5.6 wird dargestellt, wie das Testen eines einzelnen Services mit Unterstützung des Service Bus ermöglicht werden kann. Mit Hilfe der gleichen Technik lässt sich eine Testumgebung für Teilprozesse einer produktiven Lösung aufbauen. Abbildung 5.7 zeigt, wie mit Hilfe der Kapselung der Teilprozesskette *Lieferauftrag anlegen- Entnahmeliste generieren* in einer vom Service Bus generierten Umgebung getestet werden kann.

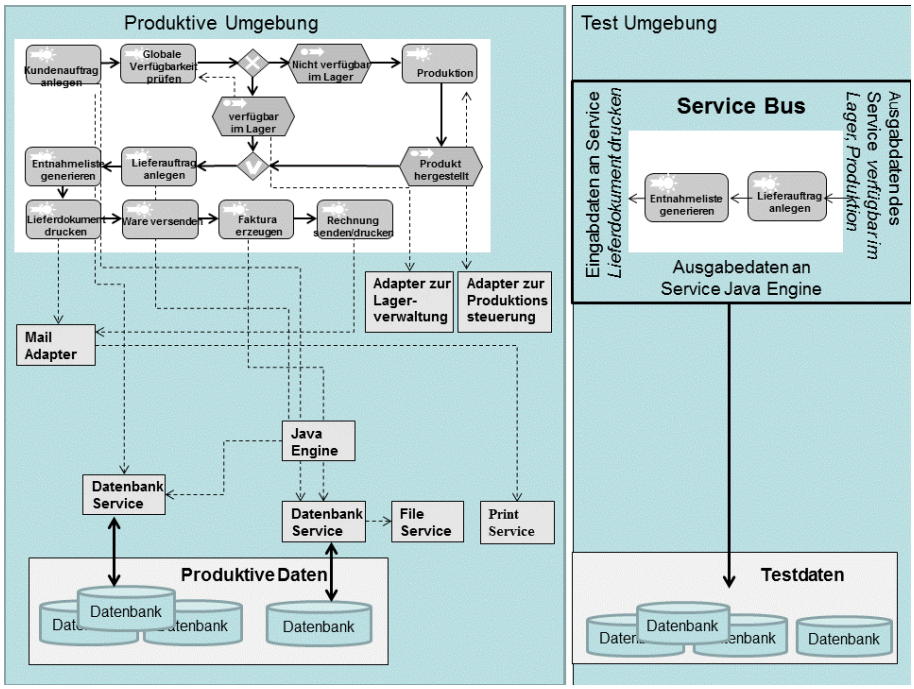

Abbildung 5.7: Gekapselte Testumgebung für eine Teilprozesskette

Somit können auch geeignete Testumgebungen für äußere und komplexe Änderungen simuliert werden. Durch die Automatisierung von Testfällen lässt sich zudem der manuelle Aufwand beim Testen erheblich reduzieren. Dies ist eine ausschlaggebende Voraussetzung will man von der Flexibilität von SOA profitieren. Der Austausch von Services stellt ebenfalls eine Änderung dar, die getestet werden muss. Automatisierung schafft Unabhängigkeit von der personellen Verfügbarkeit von Testern. Insbesondere wenn für das Testen die fachliche Unterstützung der Geschäftsabteilungen erforderlich ist, kann dies zu einem Engpass führen. Je höher der Grad der Automatisierung von Tests ist, desto flexibler können Tests durchgeführt werden. In der Praxis werden dem vollständig automatisierten Testen jedoch Grenzen gesetzt sein.

5.5 Störfall- und Problemmanagement

Wie im Abschnitt 4.2 am Beispiel des Kundenauftragsprozesses demonstriert, wirken sich SOA und die angestrebte Flexibilität insbesondere auf die Ursachenanalyse des Problemmanagements aus. Da die Service-Zusammensetzung einer Lösung nicht mehr statisch ist, sondern bei Bedarf geändert werden kann, erfordert dies ebenso flexible, aber andererseits in der Benutzung stabile Werkzeuge.

Die Stabilität in dem Sinne von Gleichartigkeit der Nutzung und Bedienung von Analysewerkzeugen je Service kann nur durch Standardisierung und Zentralisierung in COCo erreicht werden.

An Stelle des systemorientierten Ansatzes tritt ein prozess- und serviceorientierter Ansatz. Prozess-orientiert bedeutet insbesondere für die Ursachenanalyse, dass sie den Prozess als Ganzes und seine Zusammensetzung aus Services im Fokus hat. Betrachtet man die aus der Client-Server-Welt bekannten Problemstellungen und möglichen Analysewerkzeuge und überträgt dies auf SOA-basierte Lösungen, ergibt sich die Notwendigkeit für die folgenden, beschriebenen Werkzeuge.

5.5.1 Prozess-Protokoll

Um die Abläufe in einem Prozess nachvollziehen zu können, ist eine serviceübergreifende Protokollierung (*Trace*) der Ausführungsschritte inklusive Metadaten je Prozess erforderlich. Es muss gewährleistet sein, dass jeder Service bei Bedarf in standardisierter Form seine Aktionen protokolliert. Es ist davon auszugehen, dass ein solcher Prozess-Trace Hardware-Ressourcen bindet, wie CPU oder Speicherplatz. Daher soll der Prozess-Trace lediglich bei konkreten Anlässen aktiviert, wie zur zielgerichteten Analyse eines auftretenden Störfalls, also zum Beispiel in Bezug auf einen Benutzer oder einen konkreten Prozess.

Der Prozess-Protokoll soll umfassen:

- Aktion, wie z.B. ausgeführtes Modul inklusive Eingabe- und Ausgabewerte,
- Ausführungsstart- und endezeitpunkt,
- Genutzte Hardware, wie CPU-Last, Ein- und Ausgabemengen.

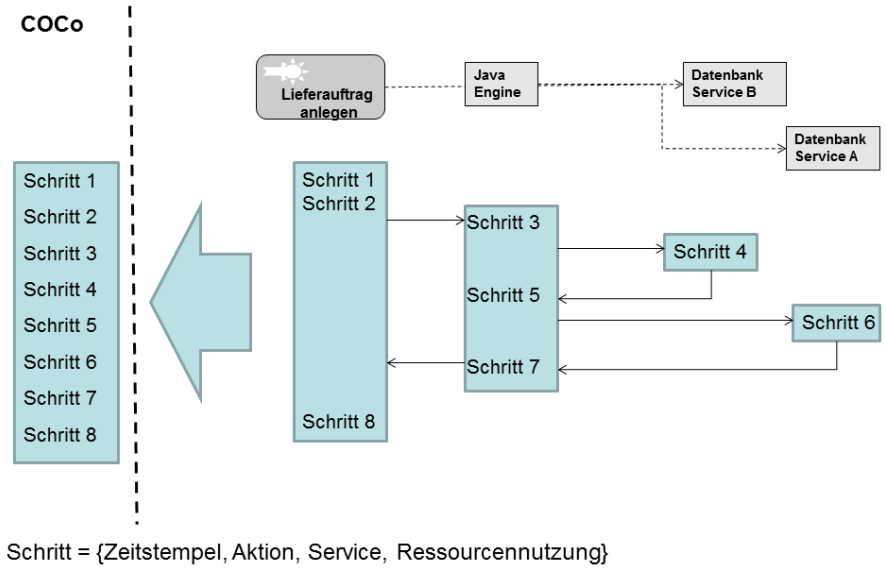

Abbildung 5.8: Bildung des Service-übergreifenden, integrierten Prozess-Traces

Abbildung 5.8 stellt dies reduziert auf den Service *Lieferauftrag anlegen* aus dem Beispiel des *Kundenauftragsprozesses* dar. Wie in Abbildung 5.8 dargestellt, müssen die Protokolleinträge an zentraler Stelle gemäß dem zeitlichen Ablauf in einem Prozess zu einem serviceübergreifenden Protokoll zusammengeführt werden. Dieser zentrale Service sei das beschriebene COCo. Damit ergeben sich folgende Anforderungen an Service und COCo.

S-Anforderung 4: Unterstützung eines zuschaltbaren Protokolls inklusive der im Service durchgeführten Aktionen und Metainformationen

Die Aktivierung eines solchen Protokolls muss ebenfalls standardisiert und zentral über COCo möglich sein.

COCo-Anforderung 2: Zielgerichtetes Ein- und Ausschalten eines Prozess-Protokolls

Die zentrale Aktivierung eines solchen Protokolls kann nur in COCo erfolgen. In COCo erfolgt das Einschalten des Protokolls für z.B: bestimmte Services, Benutzer, Prozesse oder Teilprozessketten. COCo bildet dies mit Hilfe der im Service Repository vorliegenden Informationen in entsprechenden Aktivierungen des Protokolls für die einzelnen betroffenen

Services um und führt die Protokolle zentral in der zeitlichen Abfolge zusammen, wie in Abbildung 5.8 dargestellt.

COCo-Anforderung 3: Bereitstellung einer standardisierten Schnittstelle zur Übergabe der von den Services erzeugten Protokolleinträge

Der zentrale Betriebsservice integriert die von den einzelnen Services entlang des Prozesses erzeugten Protokolleinträge zu einer zentralen Sicht. Dies erleichtert die Nachvollziehbarkeit der Abläufe. Im Fall von Störungen können so die Abläufe aufgezeichnet und analysiert werden. Dank der beigefügten Zeitstempel lassen sich auch Analysen der Laufzeit (Performance) unterstützen.

5.5.2 Integrierte Abbruchanalyse

Ein weiterer wichtiger Aspekt im täglichen Betrieb einer Lösung sind die Möglichkeiten der Analyse von Abbrüchen (Exceptions) des Prozesses. Wird im Ablauf eines Prozesses von einem Service eine Ausnahme erzeugt, kann dies weitere Ausnahmen nach sich ziehen, die letztlich zum Abbruch des gesamten Prozesses führen können. Als Teil der Problemanalyse ist es daher notwendig, den zeitlichen Ablauf der in dem Prozess aufgetretenen Abbrüche nachvollziehen zu können, um die initiale Ursache zu ermitteln, siehe auch Abschnitt 2.6. Unabhängig von der Technologie der in der Lösung genutzten Services als auch der Service-Zusammensetzung selbst muss die zeitliche Nachvollziehbarkeit und Verständlichkeit gewährleistet sein. Dies ist nur möglich, wenn ein auftretender Abbruch an COCo übergeben und den Prozessschritten zugeordnet wird. Dies verdeutlicht Abbildung 5.9. Die Funktionalität der Ausnahmebehandlung muss dabei permanent verfügbar sein, da eine Ausnahme jederzeit und nicht vorhersehbar eintreten kann. Damit ergeben sich weitere Anforderungen an Services und den zentralen Betriebsservice.

S-Anforderung 5: Weiterleitung von standardisierten Abbruchinformationen

Tritt innerhalb eines Service ein Abbruch auf, so muss diese Information inklusive Metadaten an den zentralen Betriebsservice weitergeleitet werden. Wesentliche Metadaten sind der auslösende Service, Zeitpunkt, Abbruchtext und Inhalt, wie in Abbildung 5.9 verdeutlicht. Die gelieferten Daten muss der zentrale Betriebsservice aufnehmen und speichern können.

COCo-Anforderung 4: Bereitstellung einer standardisierten Schnittstelle zur Aufnahme der von den Services übergebenen Abbruchinformationen an den zentralen Betriebsservice

Die Aufnahme der gelieferten Abbruchinformation ist jedoch nicht hinreichend. Vielmehr muss die Zuordnung der Abbrüche zu Services und Prozess ebenfalls aufgezeichnet werden.

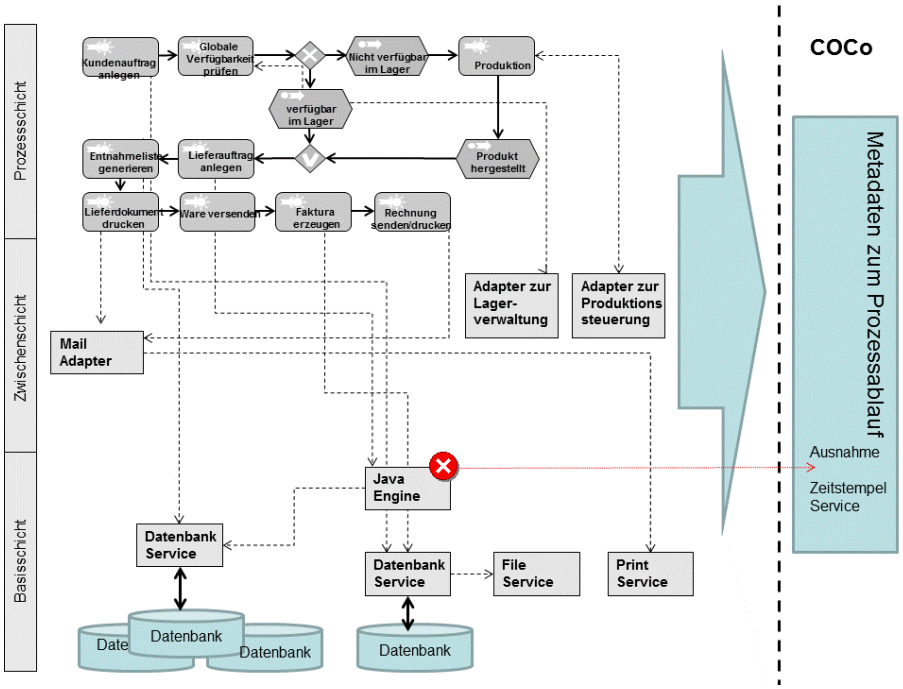

Abbildung 5.9: Übergabe einer Abbruchmeldung an den zentralen Betriebsservice

COCo-Anforderung 5: Aufzeichnung der Reihenfolge der Serviceabfolge innerhalb eines Prozesses und Zuordnung eingehender Ausnahmen

Bisher gibt es keinerlei Vorgaben für Service-Entwickler hinsichtlich Fehler- oder Beschreibungstext oder Fehlernummern bei der Entwicklung von Services. Daher ist der Inhalt eines Abbruchs maßgeblich von der zugrundeliegenden Technologie, den persönlichen Vorstellungen des Entwicklers oder bestenfalls den in einem Unternehmen vorgegebenen Regeln zur Pflege von Abbrüchen geprägt. Soll jedoch eine firmenübergreifende Integration und Nutzung von Services und sogar der problemlose Austausch von gleichartigen Services hinsichtlich einer Lösung möglich sein, führt dies zur Notwendigkeit der Vereinheitlichung und Standardisierung der Inhalte von Abbrüchen. Damit ergibt sich also eine weitere Anforderung an Services.

S-Anforderung 6: Vereinheitlichung bzw. Standardisierung der gelieferten Informationsinhalte im Fall eines Abbruchs weitestgehend unabhängig von der eingesetzten Technologie eines Services

Die von den Services gelieferten Informationen müssen auch ohne tiefgreifende Kenntnisse der Technologie eines Services auswertbar sein. Hinsichtlich der Auswertung dieser Informationen sind zumindest zwei Szenarien von COCo zwingend erforderlich zu unterstützen. Um mögliche

Probleme in einem Prozess zu analysieren, wird eine Auswertung der Abfolge von Abbrüchen inklusive der Metainformationen innerhalb des Prozessablaufs benötigt (Abbruchanalyse aus Prozesssicht). Ein anderes Einsatzgebiet ist die prophylaktische Analyse von aufgetretenen, unvorhergesehenen Abbrüchen bei Services, die sich z.B. nicht unmittelbar auf einen Prozess ausgewirkt haben, die aber doch Prozess-bezogen analysiert werden müssen (Abbruchanalyse mit Rückschlüssen auf Prozesse).

Während die Prozess-Protokollierung und die Ausnahmenbehandlung eher reaktiven Charakter haben, d.h., zum Einsatz kommen, wenn es bereits zu einer Störung kam, wird das Monitoring neben dem reaktiven Aspekt auch zur Vermeidung von Störfällen eingesetzt.

5.6 Monitoring und SLM

Eine wesentliche Aufgabe innerhalb des täglichen Betriebs von Lösungen ist die Überwachung der genutzten Services und Prozesse. Im Rahmen des Ereignismanagements werden dazu geeignete zu überwachende Kennzahlen definiert, deren Statuswechsel bzw. Anstieg oder Abfall die Gefahr eines Störfalls signalisieren. Überwacht werden dabei die Kategorien Verfügbarkeit, Performance und Durchsatz, Abbrüche und Sicherheit, wie in Abschnitt 4.4 und 4.3 beschrieben. Ziel ist es, genau die und nur die KPI zu messen, die in einer oder mehrerer der Kategorien bei Eintreten oder Über- oder Unterschreiten von Schwellenwerten einen Störfall signalisieren bzw. davor warnen. Im Vorangegangenen wurde gezeigt, dass im Monitoring und dem damit verbundenen Ereignismanagement und auch Service Level Reporting die größten Hindernisse bei der Nutzung des in SOA vorhandenen Flexibilitäts-Potentials folgende sind:

- Fehlende Ausrichtung auf Services,
- Fehlende Integration verschiedener Monitoring-Werkzeuge,
- Fehlende Normierung von Messungen spezifischer Monitoring-Werkzeuge.

Die bisherige Technologie möglicherweise relevante Messungen für KPI mittels eigener Datensammler des Monitoring-Werkzeugs vorzunehmen, ist unzureichend, wie auch in [Köppen und Will, 2012] dargestellt. Um die für die Inanspruchnahme der Flexibilität von SOA-basierten Lösungen erforderliche Unabhängigkeit von der Service-Zusammensetzung zu erreichen, muss das Verfahren der Messung genau umgekehrt werden. Nicht mehr das Monitoring-Werkzeug kann für die Beschaffung der Daten zuständig sein, sondern vielmehr müssen die erforderlichen Messungen selbstständig von den Services durchgeführt und an COCo geliefert werden.

In einem ersten Schritt ist zu standardisieren, welche Kategorien von Kennzahlen für den täglichen Betrieb erforderlich und damit auch verbindlich sind. Vorleistungen auf relativ hohem Niveau liefert wiederum ITIL. In Tabelle 4.1 wurde dies verfeinert. Bereits in der Design Phase muss konzipiert werden, welche KPI für einen Service relevant und welche Messbereiche als unbedenklich einzustufen sind. Das heißt, der Service selbst muss die Informationen mit sich führen, die dem in COCo bereitgestellten Monitoring die Entscheidung ermöglichen, ob ein Messwert positiv oder negativ zu bewerten ist. Nur dadurch kann das für den Betrieb erforderliche Wissen über einen Service auf ein Minimum reduziert werden. Der Service selbst muss als Teil seiner selbst beinhalten:

S-Anforderungen 7: Metrik von zu messenden KPI

S-Anforderung 8: Bereitstellung eines Regelsets zur Bewertung verknüpfter KPI (Zusammenhangsmetrik)

S-Anforderung 9: Bereitstellung von Messbereichen zur Bewertung der Messungen von KPI

S-Anforderung 10: Messung der KPI durch den Service bei Ausführung seiner selbst

S-Anforderung 11: Standardisierte Übermittlung der Messwerte an den zentralen Betriebsservice

Es werden zwei grundsätzliche Strategien gesehen, die von der Kategorie und dem KPI abhängig sind. Einige KPI können durch universelle Schnittstellen innerhalb des Monitoring-Werkzeugs ermittelt werden. Dazu zählt z.B. die Antwortzeit eines einzelnen Services. Da die Kommunikation der Services untereinander über den Service Bus arbeitet, ist am Service Bus bekannt, wann die Datenübergabe an einen Service erfolgt und an den nächsten übergeben wird. Gleichzeitig impliziert das, dass die Services ausschließlich über den Service Bus die zu verarbeitenden Daten erhalten. Dies ist in der Praxis nicht immer der Fall. Dadurch entfällt die Möglichkeit der Messung an zentraler Stelle durch den Service Bus.

Die zweite Strategie der Standardisierung erfordert die Anpassung jeglicher Art von Services, die in SOA betrieben werden sollen. Bereits während der Design Phase von Services müssen die notwendigen nicht-funktionalen Voraussetzungen zur Unterstützung des Monitoring geschaffen werden. Einen derartigen Ansatz für die Performance-Messung verfolgt [The Open Group, 2007] mit dem Standard für Application Response Measurement (ARM) 4.0 Dabei sind bereits im Quellcode eines Services entsprechende Anweisungen zu integrieren, die später genutzt werden können, um Performance-Messungen abzurufen. ARM beschränkt sich jedoch auf die Standardisierung des Application Programming Interface (API). Die

Vergleichbarkeit und Bewertbarkeit der gemessenen Werte wird außer Acht gelassen.

COCo muss jedoch die eintreffenden Daten verarbeiten können, was in erster Linie bewerten und anzeigen bedeutet. Die Bewertung kann sich bei komplexen Zusammenhängen zwischen einzelnen, gemessenen KPI durchaus schwierig gestalten. Daher muss ein Service in diesem Fall mit einem entsprechenden Regelset (S-Anforderung 8) ausgeliefert werden.

COCo-Anforderung 6: Bereitstellung einer standardisierten Schnittstelle zur Entgegennahme der Messwerte der Services, um sie in einem Monitoring geeignet darzustellen

Die Darstellung der von den Services gelieferten Messwerte erfolgt in Bezug auf den Service aber in standardisierter, serviceunabhängiger Form. Das Prozess- und serviceorientierte Monitoring wird benötigt, um Messungen zuzuordnen und bewerten zu können. Gleichzeitig wird durch die serviceunabhängige Darstellung der Messungen die Umgewöhnung bei einem möglichen Serviceaustausch so gering wie möglich gehalten.

COCo-Anforderung 7: Prozess- und serviceorientierte Darstellung des Monitoring

Entgegen dem bisher gängigen systemorientierten Monitoring muss das Monitoring SOA-basierter Lösungen prozess- und serviceorientiert erfolgen. Um die Übersichtlichkeit zu gewährleisten, empfiehlt sich das Monitoring gemäß den Schichten zu aggregieren.

Im Rahmen des Monitoring werden auch wesentliche Kennzahlen der Performance und des Durchsatzes überwacht. Diese Kennzahlen stehen in direktem Bezug zur Kapazität der Hardware inklusive Speicherplatz. Es ist die Aufgabe des Kapazitätsmanagements, diese Messgrößen und Tendenzen auszuwerten und zu agieren.

5.7 Kapazitätsmanagement

In Abschnitt 4.9 wurde dargestellt, welche grundlegenden Schwierigkeiten sich bei der Übertragung des aktuell verfügbaren Kapazitätsmanagement auf SOA-basierte Lösungen und deren Flexibilität ergeben. Maßgebliche Hindernisse sind die von jedem Hersteller eigens entwickelten Messgrößen und Werkzeuge und der fehlende Zusammenhang zwischen Lösung und geschäftlichen Aktivitäten sowie den technischen Messgrößen. Das Kapazitätsmanagement aus technischer Sicht muss insbesondere folgende Ressourcen planen und messen:

- Rechenleistung,
- Datenspeicher (Permanentspeicher),
- Ein- und Ausgabemengen und Geschwindigkeiten,
- Netzwerkdurchsatz.

Um die Flexibilität von SOA nutzen zu können, muss dies dabei service-unabhängig und in der Folge daraus auch technologie- und herstellerunabhängig erfolgen. Denn sollen zwei zu einer Lösung gleichwertige Services gegeneinander ausgetauscht werden, muss die Frage, ob die vorhandene Kapazität zum Betrieb genügt, zügig beantwortet werden. Daraus folgt, dass der Ressourcenbedarf zweier Services vergleichbar sein muss. Dies ist nur dann der Fall, wenn der Ressourcenbedarf von Services in vergleichbaren Einheiten ermittelbar ist. Daraus ergeben sich grundlegende Anforderungen für den Betrieb SOA-basierter Lösungen.

COCo-Anforderung 8: Standardisierung der Einheit zur Darstellung des Ressourcenbedarfs von Services

Um den Ressourcenbedarf einer SOA-basierten Lösung übergreifend ermitteln zu können, bedarf es einer normierten Einheit. Als Beispiel möge der SAP-spezifische Performance Standards SAPS angesehen werden. Alle Applikationen der SAP werden in Verhältnis zu der Anwendung Vertrieb (*Sales Distribution- SD*) gesetzt, unabhängig von physischen Hardwaregrößen. Ein ähnlicher Ansatz ist für beliebige Services denkbar, z.B. auf Basis existierender Hardware-Benchmarks wie TPC, wie in Abschnitt 4.9 gezeigt.

Performance oder Leistung werden immer in einer Einheit auf Basis von Aktivitäten je Zeiteinheit ermittelt. Die Aktivitäten von Services können sehr unterschiedlich sein. Wird die Aktivität zu servicespezifisch gewählt, wird die Standardisierung kaum gelingen. Zugleich darf die zu findende Normierung auch nicht den Bezug zu den Prozessobjekten verlieren. Andernfalls ließe sich die Relation zwischen Lösung und bearbeiteten Objekten nicht bilden. Bezogen auf das Beispiel des Kundenauftragsprozesses bedeutet dies, dass das Kapazitätsmanagement in Relation zu der Anzahl Kundenaufträgen stehen muss, die mit dem Kundenauftragsprozess zu bewältigen sind. Damit ergibt sich z.B. für die Ermittlung der erforderlichen Kapazität des Permanentspeichers die in Abbildung 5.10 dargestellte Situation. Das Datenvolumenmanagement folgt den Prozessobjekten je SOA-Schicht. Die Prozessobjekte stehen in direktem Zusammenhang mit dem jeweiligen Service. So kann von der Aktion 'Einen Kundenauftrag mit x-Positionen' entlang der Prozessobjekte bis zur Tabellen- und Indexgröße geschlossen werden. Am Ende der Schlusskette ergeben sich die erforderlichen Bytes.

∑(#Kundenaufträge* <durchschnittliche Auftragsgröße in kB>)

= ∑(#Kundenstammdaten*<durchschnittliche Größe eines Kundenstammsatzes in kB>

+#Bestelldaten* <durchschnittliche Größe in KB)>….)

= ∑ (#Tabelleinträge * < Größe eines Datensatzs in kB>

+#Indexeinträge* (durchschnittliche Größe in kB>…)

= < Speicherplatz in kB>

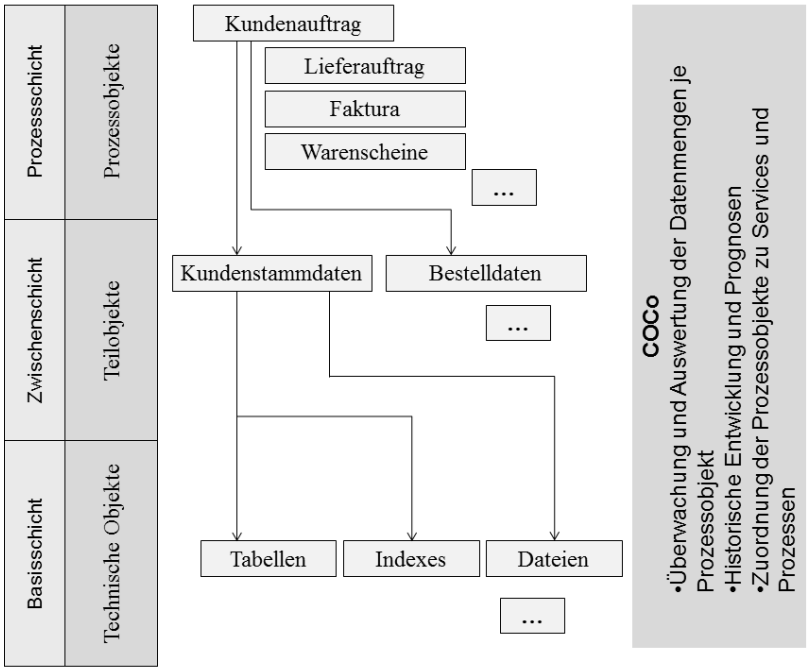

Abbildung 5.10: Datenvolumenmanagement je Serviceobjekt entlang den SOA-Schichten

Die für den Betrieb eines Service erforderlichen Kapazitäten können zudem nicht erst bei Verwendung eines Services vom Betreiber ermittelt werden, sondern sind wesentlicher Bestandteil der Entscheidung, ob ein Service genutzt werden kann und soll oder nicht. Dies führt zu der nächsten Anforderung an Services.

S-Anforderung 12: Bereitstellung des Ressourcenbedarfs in standardisierten Einheiten

Wie bereits in Abschnitt 4.9 an konkreten Beispielen gezeigt, unterstützen die Hersteller das Kapazitätsmanagement mit spezifischen Werkzeugen oder Beschreibungen zum Ressourcenmanagement. Um den Anforderungen des Betriebs von SOA-basierten Lösungen und der angestrebten Flexibilität

gerecht zu werden, müssen diese Informationen gemäß der zuvor formulierten Anforderung in standardisierter Form innerhalb der SOA bereitgestellt werden in dem Sinne von auswertbar und kalkulierbar. Da das Service Repository der zentrale Baustein zur Verwaltung der Metadaten der Services ist, bietet sich das Service Repository ebenso zur Aufbewahrung der von den Services mitgelieferten Daten zum Kapazitätsmanagement an.

SR-Anforderung 12: Speicherung der Informationen zum Ressourcenbedarf je Service

Das Service Repository wird um die Speicherung des Ressourcenbedarfs je eingesetzten Service erweitert. Die Auswertung der bereitgestellten Daten ist Aufgabe des Betriebs und somit ist dies ein Bestandteil des zentralen Betriebsservice.

COCo-Anforderung 9: Unterstützung der Auswertung und Bewertung der von den Services im Service Repository bereitgestellten Kennzahlen für das Kapazitätsmanagement

Nur wenn diese Anforderungen an Services, Service Repository und den zentralen Betriebsservice in dieser Form erfüllt sind, kann das Kapazitätsmanagement unabhängig vom einzelnen Service, seiner Technologie und seiner Herstellung erfolgen. Von ausschlaggebender Bedeutung ist die Normierung der Messeinheit der Leistungs- und Speichergrößen.

Häufig ist die Nutzung von Bereitstellung und Nutzung von Ressourcen mittels Konfigurationsparametern steuerbar. Im Folgenden werden die sich aus dem Konfigurationsmanagement ergebenen Anforderungen an den SOA-Betrieb vorgestellt.

5.8 Konfigurationsmanagement

Im Abschnitt 4.7 wurde gezeigt, dass das Konfigurationsmanagement bisher unabhängig vom Service Repository betrieben wird. Es ist zudem auf den IT-Bestand, d.h., Hardware Bausteine wie PC, Drucker und Server sowie Softwarepakete und Systeme ausgerichtet. ITIL schlägt die Nutzung eines *Configuration Management Systems* vor, dessen Kern eine Datenbank zur Verwaltung und Auswertung der Konfigurationselemente (CI) und deren Beziehungen ist.

SOA erfordert ein Service Repository, das die Metadaten eines jeden Services verwaltet und Auswertungen unterstützt. Da Konfigurationsdaten ebenso Metadaten zu einem Service sind, ergibt sich zwangsläufig die Notwendigkeit, CMDB und Service Repository für den Betrieb einer SOA-basierten Lösung zu integrieren. Insbesondere aus Sicht der Verwaltung von Hardware-technischen CI ist die Separierung der CMDB durchaus

begründbar. Doch mit dem Übergang zu SOA muss der bisherige system- und Software-Paket-orientierte Ansatz in einen serviceorientierten Ansatz zur Definition der CI übergehen. Die CMDB muss den "Service" als CI und dessen Relationen zu anderen Services und insbesondere zu Geschäftsprozessen verwalten. Damit ergibt sich für die Verwaltung der Konfigurations-Metadaten eines Services die bereits erwähnte Überlappung mit dem Service Repository. Abbildung 5.11 zeigt das erforderliche Zusammenspiel von CMDB und Service Repository, das durch COCo genutzt wird.

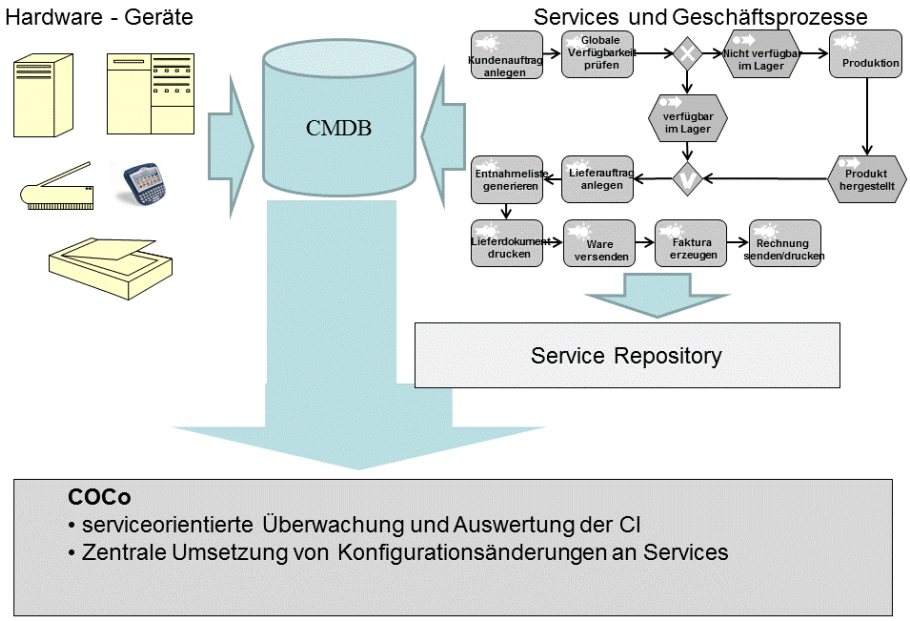

Abbildung 5.11: Zusammenspiel von CMDB, Service Repository und COCo

Neben der reinen Speicherung der Konfigurationseigenschaften eines Services und seiner Beziehung zu anderen CI gehört es zu den üblichen Aufgaben des Betriebs, diese Konfigurationen bei Bedarf anzupassen. Der Betrieb einer SOA kann nur dann flexibel hinsichtlich der Service-Zusammensetzung erfolgen, wenn die Konfiguration von Services standardisiert und zentral erfolgen kann. Servicespezifische Werkzeuge zur Änderung und Umsetzung serviceeigener Konfigurationen wirken der Flexibilität direkt entgegen. Daraus ergeben sich:

S-Anforderung 13: Standardisierte Schnittstelle, um CI wirksam zu verändern

COCo-Anforderung 10: Standardisierte Schnittstelle, um CI eines Service anzupassen

Neben geplanten Anpassungen von CI, sind jedoch häufig auch reaktive Änderungen erforderlich. Das heißt, zumeist durch das Monitoring oder das Problemmanagement wird ein Änderungsbedarf festgestellt.

Die Umsetzung erforderte bisher detailliertes Wissen in Bezug auf den Service. Um eine weitgehende Unabhängigkeit von servicespezifischem Wissen zu erreichen, also einen Service austauschbar zu betreiben, müssen die Services selbst Metriken zur Entscheidungsfindung bereitstellen. Dies führt zur S-Anforderung 14, die eng mit der S-Anforderung 7 zur Bereitstellung geeigneter Metriken für KPI verknüpft ist.

S-Anforderung 14: Bereitstellung von Metriken zur Einschätzung und Entscheidung von Änderungen an CI

Werden diese Anforderungen erfüllt, können zentral in COCo z.B. Parametereinstellungen von Services wirksam geändert werden. Statt servicespezifische Werkzeuge zu verwenden, die jeweils servicespezifisches Wissen in der Bedienung erfordern, ist nur eine Plattform COCo zu erlernen. Gleichzeitig führt dies zu einem durchgängigen Arbeitsfluss zwischen Überwachung und Reaktion: Im Monitoring innerhalb COCo können Fehlentwicklungen bzw. Störfälle erkannt und darauf gegebenenfalls durch Anpassungen der CI mit Hilfe wiederum von COCo reagiert werden.

5.9 Verfügbarkeitsmanagement

Wie bereits dargestellt, birgt die Flexibilität von SOA eine besonders wichtige Option: der Ausfall eines Services kann durch die Nutzung eines anderen, zur Lösung gleichwertigen Services abgefangen werden. Unter der Annahme, dass die bisher beschriebenen Anforderungen an die Bestandteile der SOA erfüllt sind, kann der Austausch ohne große Probleme oder Änderungen im Betrieb erfolgen. Die größte Herausforderung wäre es, einen in der Landschaft verfügbaren, zur Lösung gleichwertigen Service verfügbar zu haben und diesen aufzuspüren. Daraus ergibt sich eine weitere Anforderung an das Service Repository.

S-Anforderung 15: Bereitstellung von Metadaten zur eigenen Funktionalität, die die Vergleichbarkeit von Services gestattet

Damit die gelieferten Metadaten eines Services vergleichbar sind, müssen diese zwangsweise in standardisierter Form und Inhalt an das Service Repository geliefert werden.

SR-Anforderung 13: Verwaltung und Auswertbarkeit der von den Services bereitgestellten Metadaten, die den Vergleich und die Entscheidung über die Gleichwertigkeit von Services zu einer Lösung erlaubt

Der Ausfall bzw. die Einschränkung der Funktionstüchtigkeit eines Services ist in jedem Fall Aufgabe des Monitoring. Eine entsprechende Meldung muss dem zentralen Betriebsservice zugestellt werden. In Zusammenarbeit mit dem Service Repository muss es COCo möglich sein, zu einer Lösung gleichwertige Services zu erkennen und den Austausch zu veranlassen.

COCo-Anforderung 11: Analyse der Austauschbarkeit von Services und Ausführung des Austauschs

COCo fällt damit eine ausgesprochen wichtige Aufgabe zu, deren Realisierung von ausschlaggebender Bedeutung für die Nutzung der Flexibilität von SOA auf Basis der Austauschbarkeit von Services ist. COCo kann diese Aufgabe jedoch nur ausfüllen, wenn die Services die dafür erforderlichen Informationen in auswertbarer Form an das Service Repository liefern.

In diesem Kapitel wurden die Anforderungen an die Bestandteile einer SOA zur Unterstützung des IT Service und Support Managements zusammengetragen. Gleichzeitig wurden Realisierungsmöglichkeiten beschrieben. Als wesentlich neuer Bestandteil einer SOA wurde COCo eingeführt. Nur mit Hilfe eines zentralen Betriebscockpits ist die Unabhängigkeit von servicespezifischen Werkzeugen und deren Technologie bei gleichzeitiger prozess- und serviceorientiertem Betrieb zu erreichen. Nachfolgend werden die Anforderungen zusammengefasst.

5.10 Zusammenfassung

Es wurde gezeigt, dass das gegenwärtige Verständnis der Bausteine von SOA um COCo erweitert werden muss, will man die Flexibilität SOA-basierter Lösungen tatsächlich intensiv nutzen. Neben den akzeptierten Bausteinen Applikations-Frontend, Service Repository, Service Bus und Services kann nur mit COCo der zentrale und serviceunabhängige Betrieb die allgemein üblichen und z.B. in ITIL beschriebenen Prozesse und Funktionen unterstützen. Zusätzlich ergeben sich neue Anforderungen, die nachfolgend zusammengefasst dargestellt werden.

Tabelle 5.2 fasst die gefundenen Anforderungen an Services zusammen.

Nr	Anforderung
1	Vollumfängliche Unterstützung des Service Repository zur Erfüllung der Anforderungen des Typs SR
2	Standardisierte Schnittstelle für Wechsel der Lebenszyklus-Umgebung (Deployment)
3	Serviceverantwortlichkeit für die Service-Schnittstelleninformation im Service Repository
4	Unterstützung eines zuschaltbaren Protokolls der im Service durchgeführten Aktionen und Metainformationen
5	Weiterleitung von standardisierten Abbruchinformationen
6	Vereinheitlichung bzw. Standardisierung der gelieferten Informationsinhalte im Fall eines Abbruchs weitestgehend unabhängig von der eingesetzten Technologie eines Services
7	Metrik von zu messenden KPI
8	Bereitstellung eines Regelsets zur Bewertung verknüpfter KPI (Zusammenhangsmetrik)
9	Bereitstellung von Messbereichen zur Bewertung der Messungen von KPI
10	Messung der KPI durch den Service bei Ausführung seiner selbst
11	Standardisierte Übermittlung der Messwerte an den zentralen Betriebsservice
12	Bereitstellung des Ressourcenbedarfs in den standardisierten Einheiten
13	Standardisierte Schnittstelle, um CI wirksam zu verändern
14	Bereitstellung von Metadaten zur Einschätzung und Entscheidung von Änderungen an CI
15	Bereitstellung von Metadaten zur eigenen Funktionalität, die die Vergleichbarkeit von Services gestattet

Tabelle 5.2: Zusammenfassende Darstellung aller Anforderungen an Services

Die nachfolgende Tabelle 5.3 stellt die neuen Anforderungen an den Service Bus dar.

Nr	Anforderung
1	Standardisierte Adapter für die Service-Deployment-Schnittstelle
2	Aufbereitung der Eingabe- und Ausgabewerte eines Services gemäß den im Service Repository hinterlegten Schnittstellenbeschreibungen
3	Simulation der Datenübergabe von Vorgänger- und Nachfolgeservices gemäß der im Service Repository verfügbaren Schnittstelleninformationen

Tabelle 5.3: Anforderungen an den Service Bus

Für das Service Repository ergeben sich zahlreiche neue Anforderungen zur Unterstützung von COCo, die in Tabelle 5.4 aufgelistet sind.

Nr	Anforderung
1	Verwaltung der Service Publikation und des Kontrakts
2	Zentralisierung des Service Repository
3	Informationserzeugung und -verwendung zur Unterstützung des Betriebs
4	Zentrale Informationsverwaltung für das Service-Lebenszyklus-Management
5	Speicherung der Beziehungen zwischen Repository-Objekten
6	Verwaltung und Auswertbarkeit der Service-Versionen und Plantermine
7	Langfristige Archivierung von Prozess- und Service-Modellen zur Herstellung von zeitlichen Zusammenhängen zwischen Störfällen und Änderungen
8	Verwaltung auswertbarer Beziehungen zwischen fachlichen und technischen Aktivitäten
9	Integration von CMDB und Service Repository zu einem zentralen Baustein
10	Zentrales Service Repository als integrierender Baustein zwischen den verschiedenen Zuständen des Lebenszyklus einer SOA-basierten Lösung
11	Speicher für die Informationen zu den Schnittstellen der Services
12	Speicherung der Informationen zum Ressourcenbedarf je Service
13	Verwaltung und Auswertbarkeit der von den Services bereitgestellten Metadaten, die den Vergleich und die Entscheidung über die Gleichwertigkeit von Services zu einer Lösung erlaubt

Tabelle 5.4: Anforderungen an das Service Repository

Für den neuen zentralen Baustein in SOA COCo gibt es die in der Tabelle 5.5 zusammengefassten Anforderungen.

Nr.	Anforderung
1	Vollumfängliche Unterstützung und Aufbereitung der im Service Repository verwalteten Informationen
2	Standardisierte Schnittstelle zur Umsetzung von Statuswechsel von Services (Deployment-Schnittstelle)
3	Bereitstellung einer standardisierten Schnittstelle zur Übergabe der von den Services erzeugten Protokolleinträge
4	Bereitstellung einer standardisierten Schnittstelle zur Aufnahme der von den Services übergebenen Abbruch-Informationen an den zentralen Betriebsservice
5	Aufzeichnung der Reihenfolge der Serviceabfolge innerhalb eines Prozesses und Zuordnung eingehender Abbrüche
6	Bereitstellung einer standardisierten Schnittstelle zur Entgegennahme der Messwerte der Services, um sie in einem Monitoring geeignet darzustellen.
7	Prozess- und serviceorientierte Darstellung des Monitoring.
8	Standardisierung der Einheit zur Darstellung des Ressourcenbedarfs von Services
9	Unterstützung der Auswertung und Bewertung der von den Services im Service Repository bereitgestellten Kennzahlen für das Kapazitätsmanagement
10	Standardisierte Schnittstelle, um CI eines Services anzupassen
11	Analyse der Austauschbarkeit von Services und Ausführung des Austauschs.

Tabelle 5.5: Zusammenfassende Darstellung aller Anforderungen an COCo

Im Folgenden sollen beispielhaft die Prozesse des Störfall- und Problemmanagements evaluiert und gezeigt werden, wie sich der Betrieb einer SOA-basierten Lösung bei der Nutzung der potentiellen Flexibilität verbessert oder gar erst möglich wird, wenn die in diesem Kapitel zusammengefassten Anforderungen erfüllt sind.

6 Evaluation

Wie der Betrieb einer SOA-basierten Lösung bei Erfüllung der formulierten Anforderungen unter Einbeziehung des beschriebenen COCo ablaufen kann, soll an einem typischen IT-Prozessablauf evaluiert werden. Es wird betrachtet, wie Ereignis-, Störfall- und Problemmanagement ausgeführt werden, wenn neben den allgemein anerkannten Eigenschaften einer SOA (Abschnitt 5.1) und auch die in dieser Arbeit formulierten zusätzlichen Anforderungen an die Bausteine einer SOA erfüllt sind, siehe Kapitel 5 .

Dazu sollen drei Situationen untersucht werden:

1. Ein Nutzer meldet einen Störfall;
2. im Monitoring wird ein Störfall festgestellt;
3. mit Hilfe des Monitorings wird die potentielle Gefahr eines Störfalls prognostiziert.

Situation 1 und 2 sind dem Störfallmanagement zuzuordnen. Der Störfall ist bereits eingetreten. Es muss reagiert werden, um den Störfall zu umgehen und zu beheben. In der Situation 3 dagegen ist der Störfall noch nicht eingetreten. Gemäß ITIL obliegt es dem Ereignismanagement, die zur Vermeidung des potentiellen Störfalls nötigen Aktionen einzuleiten.

Zur Verdeutlichung der Situationen wird wiederum der in Abschnitt 2.3 vorgestellte Kundenauftragsprozess herangezogen.

6.1 Situation 1: Nutzer meldet Störfall

In der Situation 1 meldet der Benutzer die Störung des Ablaufs aus seiner Geschäftsprozesssicht. Aus Betriebssicht zeigt eine solche Situation die Unzulänglichkeit des Monitorings, denn idealerweise sollten Störfälle immer zuerst im Zuge des Ereignismanagements vom Monitoring festgestellt werden. Nichtsdestotrotz wird man diese ideale Qualität im Monitoring nie vollständig erreichen. Stellt also zuerst der Benutzer statt des Monitorings die Störung fest, muss der Betrieb zunächst aus technischer Sicht die Störung lokalisieren.

Schritt 1: Identifikation des gestörten Prozesses aus Servicesicht

Üblicherweise wird der Nutzer von einer Störung eines Prozessschritts in seinem Geschäftsprozess berichten. Er bezieht sich darauf, dass er z.B. innerhalb der Kundenauftragserfassung keinen Lieferauftrag anlegen konnte. Unter Umständen lautet die Meldung auch nur, dass die Kundenauftragserfassung nicht funktioniert. Wenn das Service Repository

die Anforderungen erfüllt und insbesondere auch Service Repository-8 „*Verwaltung auswertbarer Beziehungen zwischen fachlichen und technischen Aktivitäten“*, kann die Beziehung zwischen fachlichen und technischen Aktivitäten hergestellt werden. Das Service Repository ist auskunftsfähig über die zur Laufzeit genutzte Service-Zusammensetzung, so dass sich für den Betreiber aus der Prozesssicht in COCo die Abbruchstelle ergibt, wie in Abbildung 6.1 für den Kundenauftragsprozess dargestellt. Nun muss die eigentliche Ursache unter Einbeziehung möglicher Verkettung von Störfällen ermittelt werden.

Schritt 2: Ursachenanalyse

Durch die Erfüllung der Anforderungen Service-5 „*Weiterleitung von Abbruchinformationen“* und COCo-5 „*Aufzeichnung und Zuordnung eingehender Abbrüche zu Prozessen“* können zunächst alle für diesen Prozess aufgetretenen Abbrüche in COCo überprüft werden. Auf Grund der Anforderung 6 an Services zur „*Standardisierung der Abbruchinformationen“* ist kein servicespezifisches Knowhow zur Auswertung erforderlich. Findet sich hier kein Hinweis auf einen Störfall, kann mit Hilfe von COCo und den in den Anforderungen Service-4 „*Unterstützung zuschaltbarer Protokolle“* und COCo-3 „*Bereitstellung einer standardisierten Schnittstelle zur Übergabe der erzeugten Protokolle“* das Aktionsprotokoll innerhalb COCo für alle an dem Prozess Kundenauftragserfassung beteiligten Services und den Benutzer aktiviert werden. Die vom Benutzer beschriebenen Prozessschritte werden wiederholt, im Aktionsprotokoll aufgezeichnet und in COCo ausgewertet. Ergibt auch diese Analyse kein Ergebnis, kann mittels der Unterstützung des Service Bus eine Simulation der Abläufe in der Testumgebung vorgenommen werden. Dies wird durch die Anforderungen

- Service Bus-2 „*Aufbereitung von Ein- und Ausgabewerten eines Services“*,
- Service Bus-3 „*Simulation der Datenübergabe an einen Service gemäß Service Repository”* ermöglicht.

Zusätzlich stehen gemäß Service-4 „*Zuschaltbares Aktionsprotokoll“* weitere Protokollaufzeichnungen zur Verfügung, wie das aktivierbare, serviceübergreifende, integrierte Prozessprotokoll, das standardisiert und unabhängig von der Technologie des Services in COCo mittels COCo-3 „*Standardisierte Auswertung der Protokolle“* ausgewertet werden kann.

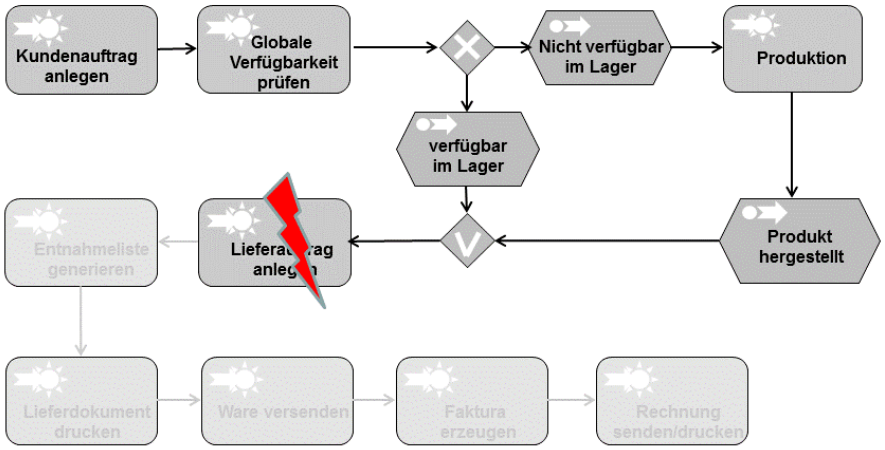

Abbildung 6.1: Störfall beim Anlegen eines Lieferauftrags im Prozess der Kundenauftragsverwaltung

Spätestens damit werden der für den Störfall verantwortliche Service und die Ursache identifiziert. Mittels des Monitorings COCo-7 „*Prozess- und serviceorientiertes Monitoring*“ wird nun geprüft, ob der Service eventuell auch von anderen Geschäftsprozessen genutzt wird und auch diese gestört sind. Falls ja, können die betroffenen Benutzer zeitnah informiert und mögliche Gegenmaßnahmen ergriffen werden. Wenn das Problem lokalisiert ist, ist die Problembehebung oder Problemvermeidung der nächste Schritt.

Schritt 3: Problembehebung

Die Problembehebung kann vielseitig sein. Meist wird die Problembehebung irgendeine Art von Änderungen erfordern. Damit steht das Problemmanagement in direkter Verbindung zum Änderungsmanagement, das an dieser Stelle nur generell, also nicht vollständig evaluiert wird. Es wird auf die drei typischen Optionen möglicher Änderungen fokussiert.

Option 1: Coding-Anpassung

Es kann beispielsweise eine Coding-Änderung erforderlich sein. Nicht immer wird die eigene Entwicklungsabteilung dafür verantwortlich sein. Gegebenenfalls muss der externer Hersteller des Services kontaktiert werden. Die Übergabe an das Änderungsmanagement erfolgt mittels einer Änderungsanforderung (Request for Change RFC), siehe Abschnitt 4.5.

Option 2: Konfigurationsanpassung

Die Problembehebung kann auch darin bestehen, die Konfiguration zu ändern. Diese Änderung kann unter den formulierten Anforderungen mittels der standardisierten Schnittstelle des Services zur Änderung von CI (Service-13 „*Standardisierte Schnittstelle zur Änderung von CI*“) über den

Zentralen Betriebsservice COCo-8 „*Standardisierte Darstellung des Ressourcenbedarfs“* vorgenommen werden. Neben den bekannten Methoden zur Problembehebung eröffnet sich aber dank SOA auch die Möglichkeit, den verursachenden Service gegebenenfalls auszutauschen.

Option 3: Service-Austausch

Ist in der SOA-Landschaft ein zum gestörten Prozess gleichwertiger Service verfügbar, können die Services gegeneinander getauscht werden.

Mit Hilfe des Service Repositorys kann nach einem für den Prozess gleichwertigen Service gesucht werden (Service Repository-13 „*Auswertbarkeit der Service Metadaten zur Feststellung von Gleichwertigkeit“*).

Ist ein zum gestörten Prozess gleichwertiger Service vorhanden, kann der defekte Service ausgetauscht werden. Ist dies der Fall, wird der Änderungsmanagementprozess angestoßen. Es werden die vorhandenen Kapazitäten zum Betrieb des neuen Services geprüft mit Hilfe von Service-12 „*standardisierte Bereitstellung des Kapazitätsbedarfs“* und der im Service Repository hinterlegten, standardisierten Daten je Service (Service Repository-12 „*Verwaltung der Informationen zum Ressourcenbedarf“*).

Unabhängig davon, wer die Änderung der vorgestellten Optionen 1, 2 oder 3 ausführt, schließt sich daran das Testmanagement an, das in Abschnitt 4.8 untersucht wurde. An dieser Stelle wird dies skizziert.

Schritt 4: Testen

In der hier vorgestellten Situation beschränkt sich die vorgenommene Änderung lediglich auf einen Service. Es handelt sich also um eine einfache Änderung (siehe Abschnitt 4.5.1). Dadurch genügt es für den Test des Services, die durch den Service Bus unterstützte, gekapselte Testumgebung für die Prozessintegration zu nutzen.

Mittels der Anforderungen an den Service Bus

- Service Bus-2 *„Aufbereitung von Ein- und Ausgabewerten eines Services“,*
- Service Bus-3 „*Simulation der Datenübergabe an einen Service gemäß Service Repository”*

wird das Zusammenspiel des neuen Services im Prozess getestet. Ist dies erfolgreich, kann der neue Service den gestörten Service zur Laufzeit ersetzen. Nur wenn der Service-Austausch möglichst einfach und ohne die Gefährdung weiterer Prozesse möglich ist, ist dieser Aspekt der Flexibilität von SOA wirklich nutzbar. Die in Kapitel 5 beschriebenen Anforderungen sind dafür unabdingbar. Liefert das Testen positive Ergebnisse, kann die

Änderung in die Produktion übernommen werden. Falls nicht, muss das Änderungsmanagement u.U. wiederholt durchlaufen werden.

Schritt 5: Freigabe des neuen Services in die Produktion

Zur Freigabe eines Services, d.h. der Übernahme eines Service in die produktiven Prozesse, muss der geänderte Service aus der Testumgebung in die Produktivumgebung transferiert werden. Durch die erfüllten Anforderungen:

- Service Repository-10 „*Zentrale Verwaltung der Lebenszyklusstatus der Services“*,
- Service Bus-1 „*Standardisierte Adapter für die Service-Freigabe“*,
- Service-2 „*Standardisierte Schnittstelle für Wechsel der Lebenszyklusumgebung“*

wird die Freigabe des geänderten Services oder des Ersatzservices durch COCo-2 „*Standardisierte Schnittstelle zur Umsetzung von Statuswechseln von Services“* oder COCo-11 „*Analyse und Ausführung eines Serviceaustauschs“* in die Produktivumgebung initiiert. Der gestörte Service wird gleichzeitig aus dem Betrieb genommen. Der Änderungsauftrag kann geschlossen werden.

Der neue Service überträgt automatisch seine Metadaten an das Service Repository und wird in COCo integriert, wie in den Serviceanforderungen Service-1 – Service-15 dargestellt, siehe Tabelle 5.2. Der Service Bus transferiert die Daten zum und von dem neuen Service gemäß der gelieferten Schnittstellenbeschreibung

- Service-3 „*Übermittlung der Schnittstelleinformationen an das Service Repository“*,
- Service Repository-11 „*Speicherung der Schnittstelleninformationen zum Service“* und
- Service Bus-2 „*Aufbereitung von Ein- und Ausgabewerten eines Services“*.

Durch die in Tabelle 5.2 dargestellten Eigenschaften eines Services wird der neue Service ohne weitere manuelle Aktionen in das serviceorientierte Monitoring in COCo integriert. Der Erfolg des Service-Austauschs wird umgehend sichtbar und kontrolliert. Fällt das Ergebnis positiv aus, kann das Problem geschlossen werden. Durch die Standardisierung und Zentralisierung der Serviceinformationen und des Servicemanagements in COCo kann der Betrieb ungeachtet des Serviceaustauschs oder -änderung in der gleichen Art und Weise wie zuvor fortgesetzt werden.

Der Vollständigkeit halber sei angemerkt, dass an dieser Stelle der kontinuierliche Verbesserungsprozess einsetzen sollte, um das Monitoring

möglichst so zu verbessern, dass ein derartiger Störfall, wie in Situation 1 beschrieben, vom Monitoring automatisch registriert wird, bevor der Endbenutzer den Störfall melden muss.

In der nachfolgenden Situation wird untersucht, wie die Prozesse ablaufen, wenn der Störfall vom Monitoring an Stelle des Nutzers gemeldet wird.

6.2 Situation 2: Monitoring meldet Störfall

Während in der Situation 1 der betroffene Service zunächst ermittelt werden muss, wird in der Situation 2 dem Betreiber der Störfall bei einem Service und den damit zusammenhängenden Services und Prozessen bereits an COCo mit entsprechenden Metadaten gemeldet, siehe Abbildung 6.2. Dort kommt es initial zu einer Störung in der Java Engine, die zur Störung des Services zum Anlegen eines Lieferauftrags führt.

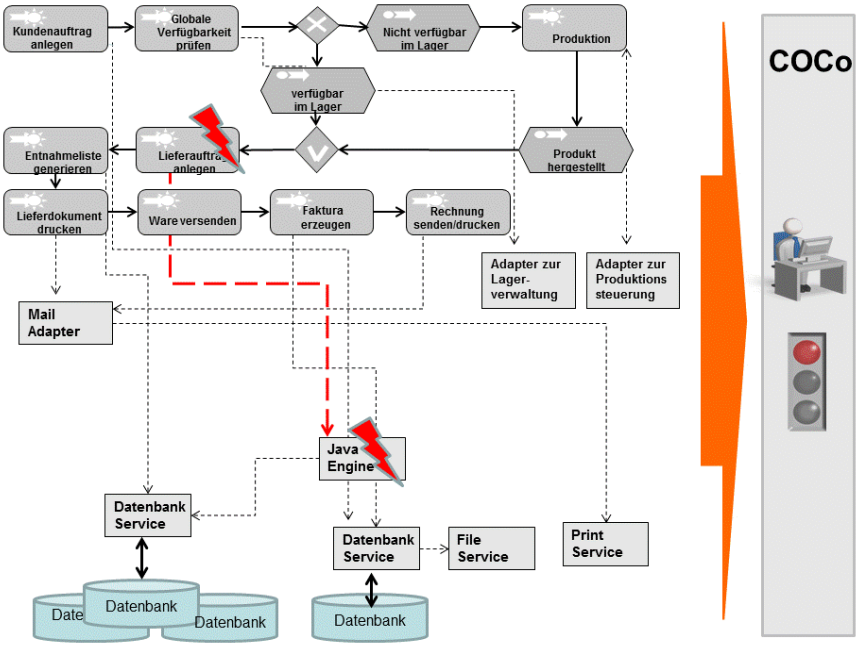

Abbildung 6.2: Prozessorientiertes Monitoring und Alarm mit COCo

Das Monitoring kann die initiale Störung bereits einem Service in einer Prozesskette zuordnen auf Grund der erfüllten Anforderungen

- Service-5 „*Weiterleitung von standardisierten Abbruchinformationen*“,
- COCo-5 „*Aufzeichnung der Serviceabfolge und Zuordnung von Abbrüchen*“,
- COCo-4 „*Standardisierte Schnittstelle zur Entgegennahme der Abbruchinformationen von Services*“.

Damit stehen dem Betreiber der Lösung bereits alle Informationen zu dem Service und dem Prozess bereit. Die Störfallanalyse kann sich umgehend auf den verursachenden Service fokussieren. Zusätzliche Befragungen des Nutzers zu Art und Hergang seiner Prozessarbeit können entfallen. Durch Service-8 „*Bereitstellung eines Regelsets zur Bewertung verknüpfter KPI“* und Service-7 „*Metrik von zu messenden KPI“* kann das Monitoring zudem kritische Situationen bewerten. Dadurch kann der Betreiber vorbeugend den Endbenutzer über den Störfall im Prozess der Kundenauftragsbearbeitung und auch anderer Prozesse informieren, die vielleicht denselben Service verwenden. Da der ursprüngliche, die Störung verursachende Service bereits bekannt ist, kann sich die Ursachenanalyse sehr direkt auf diesen Service fokussieren und die Behebung erarbeiten. Das weitere Vorgehen ist analog ab Schritt 3 aus der Situation 1.

Analog zu Situation 1 kann die Ursache des Störfalls oder Problems entweder behoben werden oder die Austauschbarkeit des betroffenen Services geprüft und der Austausch vorgenommen werden. Alle dafür notwendigen Tätigkeiten können zentral und standardisiert in COCo ausgeführt werden, wodurch servicespezifisches Wissen nicht erforderlich ist.

Die Situationen 1 und 2 sind aus Sicht des Betriebs als reaktive Abläufe anzusehen. Der Störfall trat ein und wurde entweder vom Benutzer oder vom Monitoring bemerkt. Es muss reagiert werden, um den Störfall zu beheben oder zumindest temporär zu umgehen. In der nachfolgenden Situation 3 wird dagegen das proaktive Vorgehen beschrieben. Durch das Monitoring wird die Entwicklung einer kritischen Situation signalisiert, aber der Störfall ist noch nicht eingetreten.

6.3 Situation 3: Monitoring prognostiziert Störfall

Durch die Anforderungen

- Service-8 „*Bereitstellung eines Regelsets zur Bewertung verknüpfter KPI“,*
- Service-7 „*Metrik von zu messenden KPI automatisch“*

und die permanente Messung bzw. Überwachung der Entwicklung der KPI kann COCo kritische Entwicklungen erkennen und bewerten. Dies kann z.B. die sich andeutende Ausschöpfung der vorhandenen Hardware-Kapazitäten (Service-12 „*Standardisierte Bereitstellung des Kapazitätsbedarfs“*) oder der fortlaufende Performanceabfall sein, der die mitgelieferten Erwartungswerte nicht mehr erfüllt (Service-8 „*Bereitstellung eines Regelsets zur Bewertung verknüpfter KPI“*). Ziel muss es sein, die sich abzeichnenden Beeinträchtigungen des produktiven Betriebs der Prozesse

durch rechtzeitige Änderungen zu verhindern. Abbildung 6.3 stellt die Situation dar. Durch die fortlaufende Messung des KPI kann die historische und auch zu erwartende Entwicklung dargestellt und bewertet werden. Wurde ein kritischer Schwellenwert hinterlegt, erlaubt dies eine Warnung vor dem bevorstehenden Störfall. In Abbildung 6.3 könnte dies beispielweise die permanent ansteigende Antwortzeit der Java Engine sein, die in absehbarer Zeit die Applikation beeinträchtigen wird.

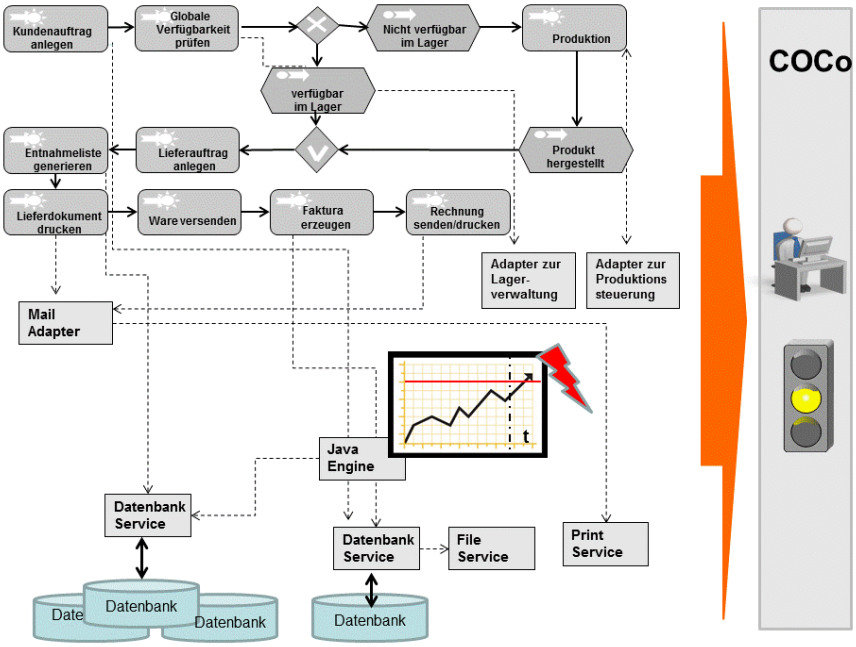

Abbildung 6.3: Störfallprognose auf Basis permanenter Messung eines KPI

Wird eine solche kritische Entwicklung mit Hilfe von COCo erkannt, kann die Ursache analog zu den Situationen 1 und 2 mit den in COCo vorhandenen Werkzeugen untersucht werden. Da ein Störfall nur prognostiziert und noch nicht eingetreten ist, werden auf Grund der Warnung geeignete Gegenmaßnahmen eingeleitet, was im Allgemeinen bedeutet, dass Änderungen vorgenommen werden. Es schließen sich demnach an die Ursachenanalyse und Identifizierung, die aus den Situationen 1 und 2 bereits bekannten Schritte ab 3 an.

Die Unterstützung dieses vorbeugenden Agierens zur Vermeidung von Störfällen ist von besonderem Interesse sowohl für die Benutzer als auch für den Betrieb. Jeder eingetretene Störfall impliziert Ausfälle in den Geschäftsabläufen. Daher muss die Behebung eines Störfalls möglichst zeitnah erfolgen. Dies führt zu einem erheblich höheren Druck auf die

Bearbeiter eines Störfalls und unterbricht zudem den normalen IT Support Ablauf. Daher ist es Ziel der Betriebsorganisation, den Anteil Störfall vermeidender Aktionen im Vergleich zu reaktiver Tätigkeit möglich hoch zu gestalten. Insbesondere gilt es, die Situation 1 zu vermeiden, bei der der Benutzer als erster den Störfall erfährt. Je früher kritische Situationen oder Entwicklungen prognostiziert werden, desto größer ist die Chance, negative Auswirkungen auf die Benutzer und die Ausführung der produktiven Prozesse einzuschränken oder zu vermeiden. COCo bietet dafür die geeigneten Grundlagen.

6.4 Zusammenfassung der situationsbedingten Störfallbearbeitung

Die dargestellten Situationen zeigen in abstrakter Form die typischen Arbeitsschritte zur Analyse und Behebung bzw. Vermeidung von Störfällen. Abbildung 6.4 zeigt in der Übersicht, welche Anforderungen bei welchem Arbeitsschritt von der Entdeckung bzw. Prognose eine Störfalls bis zur Anpassung der Applikation von wesentlicher Bedeutung sind und erfüllt sein müssen.

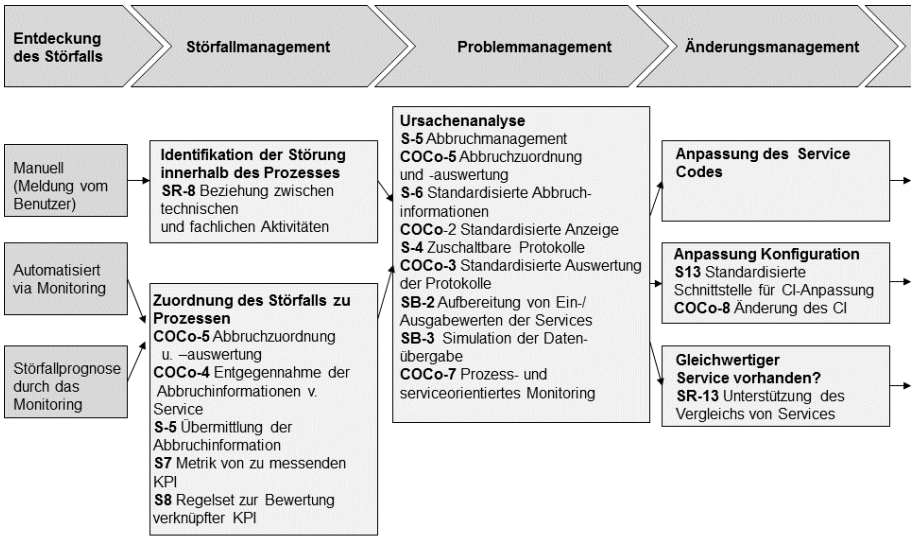

Abbildung 6.4: Übersicht über die bei der Störfallbearbeitung erforderlichen Voraussetzungen- Teil I

Die vorgestellten Situationen bei der Behebung bzw. Vermeidung eines Störfalls zeigen, dass die Ursachenanalyse und Identifikation des betroffenen Service sowie die Ermittlung der Auswirkungen des Störfalls auf andere Prozesse einen Schwerpunkt der Analyse bilden. Durch den Übergang zu

SOA und der losen Kopplung von Services zur Laufzeit sind andere Werkzeuge als in Client-Server-Architekturen notwendig. Die Anforderungen an COCo sowie die Anforderungen an Service Repository, Service Bus und Services sind zwingend, um diesen ersten Schritt in der Störfallbearbeitung erheblich zu vereinfachen. Ebenso vorteilhaft wirkt sich COCo bei der Ursachenanalyse aus, die durch den Übergang zu SOA mit den herkömmlichen Client-Server-Werkzeugen weitaus komplexer und langwieriger oder sogar kaum noch realisierbar ist. Abbildung 6.5 fasst die genutzten Anforderungen ausgehend vom Änderungsmanagement bis zur Freigabe in die Produktion zusammen.

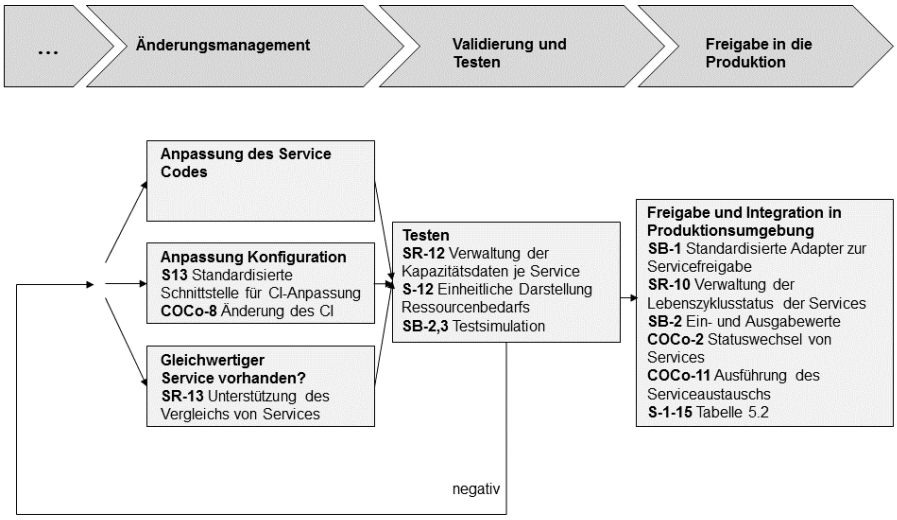

Abbildung 6.5: Übersicht über die bei der Störfallbearbeitung erforderlichen Voraussetzungen- Teil II

Für den Austausch von Services, die gleichwertig sind zu einer Lösung, sind COCo und die an Service Repository, Service Bus und Services gestellten Forderungen zwingend. In Tabelle 1.1. der Gegenüberstellung von Client-Server- und SOA-basierten Lösungen wurden insbesondere die fehlenden Werkzeuge als Hindernis zur Nutzung der Flexibilität von SOA dargestellt. Durch die Erfüllung der in den Tabellen 5.1-5.5. dargestellten Anforderungen und die Einführung eines zentralen Betriebscockpit COCo können diese Schwächen nicht nur behoben, sondern in Stärken gewandelt werden, wie in diesem Kapitel gezeigt. Die ausschlaggebende Voraussetzung für solch ein beschriebenes COCo ist jedoch, ob es gelingt, einen quasi Standard bzw. die Normierung von Informationen zu entwickeln und durchzusetzen, die ein Service zu erbringen hat. In dieser Arbeit wurden die Grundlagen dafür aufgezeigt.

7 Schlussfolgerungen und Ausblick

In der vorliegenden Arbeit wurde untersucht, wie sich die technologische Architektur auf die eher praktische Seite des Betriebs einer darauf basierenden Applikation auswirkt. Es wurde gezeigt, dass sich die Unterschiede zwischen Client-Server-Architektur und SOA massiv auf den Betrieb einer darauf basierenden Lösung auswirken, insbesondere wenn von der in SOA unterstützten Flexibilität in der Service-Orchestrierung Gebrauch gemacht wird. [Hevner et al, 2004] argumentiert, dass sich das Design und die Theorie einer IT-Lösung auf die Organisationen, die Menschen und die Arbeitsmethoden auswirken. Auch diese Implikationen müssen bei der wissenschaftlichen Forschung untersucht werden „[…] scientific research should be evaluated in light of its practical implications." Der Betrieb einer implementierten Applikation auf Basis einer Client-Server-Architektur oder SOA ist eine eher praktische Anforderung, aus der aber weitere fundamentale und nicht-funktionale Anforderungen resultieren, wie in dieser Arbeit gezeigt. Um das technische Potential von SOA wirklich nutzen zu können, sind neue Werkzeuge und Anpassungen der IT Service und Support Prozesse erforderlich.

In dieser Arbeit wurde nachgewiesen, dass bereits beim Design neuer Services und IT-Lösungen zur Erfüllung immer neuer funktionaler Anforderungen aus dem Business auch der spätere, unterbrechungsfreie Betrieb betrachtet werden muss. Tabelle 1.1. zeigt, dass SOA unter den bisherigen Gesichtspunkten Schwächen bei der Stabilität und den Betriebswerkzeugen aufweist, wenn die Flexibilität von SOA auf Basis der Serviceaustauschbarkeit genutzt wird oder gar so weit führt, dass auf die Nutzung der Serviceaustauschbarkeit verzichtet wird, um die Stabilität der SOA-basierten Lösung zu gewährleisten. Legt man das von [DeLone und McLean, 1992] definierte Erfolgsmodell für Informationssysteme zugrunde, wie in Abbildung 7.1 dargestellt, ist im Umkehrschluss dadurch der Erfolg von SOA und seiner Flexibilität gefährdet.

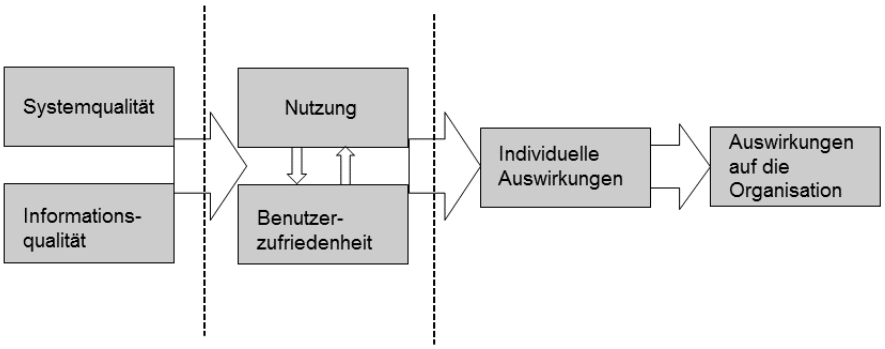

Abbildung 7.1: DeLong& McLean Informationssystem Erfolgsmodell gemäß [DeLone und McLean, 1992]

In der Aktualisierung [DeLone und McLean, 2011] wird der Erfolg eines Informationssystems (IS) als mehrdimensional und ineinandergreifend beschrieben „[…] IS success is a multidimensional and interdependent construct.“ Dabei wird die Systemqualität für die Messung des technischen Erfolgs und die Informationsqualität zur Messung des semantischen Erfolgs gesehen „ […] “systems quality” measures technical success; “information quality” measures semantic success;“, Anwendbarkeit und Benutzerzufriedenheit haben Auswirkungen auf das einzelne Individuum und damit letztlich auch auf die Organisation „[…] “use, user satisfaction, individual impacts,” and “organizational impacts” measure effectiveness success.” Die Qualität des möglichen Betriebs einer IT-Applikation ist demnach der Systemqualität zuzuordnen und ebenso ein Schlüssel für den Erfolg einer IT-Lösung, wie die Erfüllung funktionaler Anforderungen. Dies unterstreicht die Bedeutung der Untersuchungen in dieser Arbeit.

In den vorangegangenen Kapiteln wurde erstmals zusammenhängend dargestellt, wie sich durch die Evolution zu SOA-basierten Lösungen die Anforderungen an den Betrieb einer IT- Lösung verändern, will man wirklich die Flexibilität auf Basis des Austauschs von Services nutzen. Zwar bieten SOA-basierte Applikationen auf Grund ihrer Architektur das Potential flexibler Anpassung an die sich permanent ändernden Geschäftsbedingungen, doch wird diese Flexibilität tatsächlich verhalten genutzt. In Kapitel 4 wurde untersucht und gezeigt, dass sich die Nutzung der Flexibilität unter den aktuellen Umständen nachteilig auf den Betrieb SOA-basierter Applikationen auswirkt. Die häufig noch aus der Client-Server-Welt stammenden, zum Teil weiter entwickelten Werkzeuge genügen nicht den sich neu entwickelten Anforderungen der flexiblen SOA-basierten Lösungen. Auf Grund der bisher unzureichenden Betriebsmöglichkeiten ist der Austausch von Services immer mit einem erheblichen Aufwand und der Gefährdung der Stabilität einer Lösung verbunden. Solange diese Risiken

größer sind als der Nutzen der Flexibilität der SOA-basierten Lösungen, wird der Betreiber den dynamischen Austausch von Services vermeiden. Daraus folgt aber auch die Abhängigkeit der Nutzer von den eingesetzten Services und damit von den Herstellern dieser Services.

Es ist verständlich, dass die Service-Hersteller selbst nur ein bedingtes Interesse an dieser Flexibilität aufbringen, würde dies doch umgekehrt den Wechsel zu Services anderer Hersteller vereinfachen. In Kapitel 4 wurde gezeigt, wie mit Hilfe eines zentralen und den Betrieb unabhängig von der Service-Technologie vereinheitlichenden Betriebscockpit COCo die bisherigen Schwächen von SOA bei Nutzung der Flexibilität ausgeglichen werden können. Wie in der Arbeit gezeigt, ist die Vereinheitlichung des Betriebs, was sowohl die standardisierte Unterstützung durch die Services als auch eines standardisierten Betriebswerkzeugs COCo beinhaltet, die wesentliche Voraussetzung, um die Flexibilität von SOA auf Basis des Service-Austauschs zu nutzen. Die wichtigsten Interessenten einer solchen Entwicklung sind jedoch die Kunden der Service-Hersteller sowie die potentiellen Betreiber SOA-basierter Applikationen. Die Entwicklung von Standards für den Betrieb und zur Unterstützung eines Technologie- und damit auch Service-herstellerunabhängigen COCo steht bisher nicht im Fokus der Service-Hersteller. In der Schlussfolgerung sind es die Service-Nutzer und -Betreiber, die durch entsprechenden Druck auf die Hersteller die Entwicklung von Standards für den Lösungsbetrieb vorantreiben müssen. In dieser Arbeit wurden die grundsätzlichen Eigenschaften eines solchen zentralen Betriebsservice sowie die dazu erforderlichen Standardisierungen dargelegt. Prinzipiell wäre es auch denkbar, dass sich ein von den Markt-beherrschenden Herstellern von betriebswirtschaftlichen Lösungen unabhängiger Service-Hersteller der Betriebsproblematik annimmt. Mit der zwingend erforderlichen Unterstützung durch die Betreiber und einer einhergehenden Standardisierung der durch einen Service zu leistenden Unterstützung des Betriebs kann es möglich sein, ein solches, in dieser Arbeit beschriebenes Werkzeug zu entwickeln.

Maßgeblich für die weitere Entwicklung dürfte der finanzielle Aspekt des Betriebs sein. In dieser Arbeit wurde gezeigt, dass in der aktuellen Situation die tatsächliche Inanspruchnahme der Flexibilität von SOA durch die Integration von Services insbesondere anderer, neuer Technologien in bestehende Lösungen den Aufwand des Betriebs erheblich erhöht. Es wurde dagegen nicht untersucht, ob ein solcher auf Standards beruhender, zentraler Betriebsservice COCo derartige finanzielle Einsparungen ermöglichen würde, die die Investition in die Vereinheitlichung des Betriebs aufwiegen würden. Dies führt zu der Frage, ob die vollumfängliche Nutzung der Austauschbarkeit und Erweiterbarkeit von Services in SOA derartige finanzielle Einsparungen ermöglicht. Diese Arbeit untersuchte die technischen Voraussetzungen und entwickelte die theoretischen Grundlagen,

die zeigen, dass es möglich ist, ein geeignetes Betriebswerkzeug für diesen Zweck zu entwickeln. Gleichzeitig unterstreicht die Arbeit, wie bedeutend es ist, sich bereits beim Design und der Entwicklung von Systemen nicht nur auf die Bedürfnisse und Anforderungen der Anwender der Applikation zu fokussieren, sondern auch den späteren Betrieb einer darauf basierenden Lösung zu planen. Neben den funktionalen Anforderungen ergeben sich nicht-funktionale Anforderungen aus der Notwendigkeit, die Lösung gemäß den Anforderungen der Benutzer stabil und performant zu betreiben.

In den vorliegenden Untersuchungen wurde vorausgesetzt, dass alle in der SOA-basierenden Lösung verwendeten Services vom Benutzer bzw. im Auftrag der Benutzer von einem Outsourcer betrieben werden. Sämtliche Rechte der Nutzung und des Betriebs wurden von der nutzenden bzw. betreibenden Organisation per Lizenzen erworben. Aktuell gewinnen sogenannte Cloud-Lösungen zunehmend Verbreitung. Dabei werden lediglich die Nutzungsrechte an Lösungen bzw. Services erworben. Der Betrieb verbleibt weiterhin beim Anbieter der Lösung oder des Services. Der Benutzer einer Lösung mit integrierten Cloud-Services bleibt jedoch verantwortlich für die gesamte Lösung. Es ist zu erwarten, dass diese Architektur und der ihr innewohnenden Flexibilität den Druck zur Lösung der hier beschriebenen Herausforderungen im Betrieb SOA-basierter Lösungen erhöhen wird.

8 Literaturverzeichnis

Ali, S.; Iqbal, M. Z. ; Arcuri, A. ; Briand, L.: A Search-based OCL Constraint Solver for Model-based Test Data Generation. In: Proceedings 11th International Conference on Quality Software; Madrid, Spanien (2011).

Alter, S.; Börner, R. ; Goeken, M.: Operationalisierung der IT-Governance-Kernbereiche für die Identifizierung und Gestaltung von Services. In: Proceedings Business process, services computing and intelligent service management (BPSC 2009), Leipzig, Deutschland. Gesellschaft für Informatik, Bonn (2009).

Ault, M. R.: Oracle DBA made simple: Oracle database administration techniques. Rampant TechPress, Kittrell, North Carolina, USA (2003).

Barnevik, P.: Service Level Agreement: Mythos- Ist Leistung generell messbar? (2012). http://www.4managers.de/management/themen/service-level-agreement/ (Zuletzt geprüft am: 07.05.2014).

Bernhard, M. G.: Praxishandbuch Service-Level-Management: Die IT als Dienstleistung organisieren. Symposion, Düsseldorf, Deutschland, 1. Auflage (2003).

Brosig, F.; Huber, N. ; Reussner, R.: Towards Self-Aware Performance and Resource Management in Modern Service-Oriented Systems. In: Proceedings Services Computing (SCC), IEEE International Conference, Miami, Florida, USA. IEEE Computer Society, Piscataway, New Jersey, USA (2010), S. 621–624.

Buchwald, S.; Tiedeken, J.; Bauer, T. ; Reichert, M.: Anforderungen an ein Metamodell für SOA-Repositories. In: Proceedings 2. Zentral-europäischer Workshop über Services und ihre Komposition, Berlin, Deutschland (2010), S. 17–24.

Burleson, D. K.; Danchenkov, A. B.: Oracle tuning: The definitive reference. Rampant TechPress, Kittrell, North Carolina (2005).

Canfora, G.; Di Penta, M.: SOA: Testing and Self-Checking. In: Proceedings International Workshop on Web Services - Modeling and Testing, Palermo, Italien (2006).

Canfora, G.; Di Penta, M.: Service-Oriented Architectures Testing: A Survey. In: Lucia, A., Ferrucci, F. (eds.): Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg (2009), S. 78–105.

Chen, R.; Scheer, A.-W.: Modellierung von Prozessketten mittels Petri-Netz-Theorie. Institut für Wirtschaftsinformatik an der Universität des Saarlandes, Saarbrücken (1994).

Cisco Systems Inc.: Bandwidth, Packets Per Second, and Other Network Performance Metrics (2013).

Cummins, F.: Building the Agile Enterprise: With SOA, BPM and MBM. OMG Press, Burlington, USA (2009).

Datanamic Solutions EV: Data Generator for Oracle. http://www.datanamic.com/datagenerator-for-oracle/index.html (Zuletzt geprüft am: 07.05.2014).

DeLone, W.H.; McLean, E.R.: Information Systems Success: The Quest for the Dependent Variable. In: Information Systems Research (1992), Volume 3, Heft 1, S. 60-95.

DeLone, W.H.; McLean, E.R.: The DeLone and McLean Model of Information Systems Success: A Ten-Year Update. In: Journal of Management Information Systems (2003), Volume 19, S. 9-30.

Eicker, S.; Nagel, A; Schuler, P. M.: Flexibilität im Geschäftsprozessmanagement-Kreislauf. Institut für Informatik und Wirtschaftsinformatik (ICB), Essen, Deutschland (2007).

EMS Database Management Solutions: EMS Data Generator for Oracle. http://www.sqlmanager.net/en/products/oracle/datagenerator (Zuletzt geprüft am: 07.05.2014).

Erl,T.: Service-oriented architecture: Concepts, technology, and design. Prentice-Hall, Upper Saddle River, New Jersey, USA (2005).

Erl, T.: SOA design patterns. Prentice Hall, Upper Saddler River, New Jersey, USA (2009).

Evans, J. S.: Strategic Flexibility for high technology manoeuvres. In: Journal of Management Studies (1991), Volume 28, Heft 1, S. 69-89.

Franklin, S.; Graesser, A.: Is it an agent, or just a program?: A taxonomy for autonomous agents. In: Müller, J. P. (ed.): Agent theories, architectures, and languages Proceedings. Springer, Berlin, Deutschland, Volume 1193 (1997), S. 21–35.

Gebauer, J.; Schober, F.: Information System Flexibility and the Performance of Business processes (2005). www.business.uiuc.edu/working_papers/papers/05-0112.pdf (Zuletzt geprüft am: 13.05.2014).

Gernert, C.; Köppen, V.; Darkow, O.; Schwarz, T.: Von ARIS zur UML: Transformationen in der Prozessmodellierung. In: ObjektSpektrum, Jahrgang 6 (2005), S. 53-59.

Goyal, A.: Enterprise Services Repository and Registry . SAP AG, (2009) http://www.sdn.sap.com/irj/scn/index?rid=/library/uuid/00985388-6748-2c10-0d83-f17c3e768a8b&overridelayout=true (Zuletzt geprüft am: 07.05.2014).

Her, J. S.; Choi, S. W.; Oh, S. H. ; Kim, S. D.: A Framework for Measuring Performance in Service-Oriented Architecture. In: Proceedings International Conference on Next-Generation Web Services Practices. IEEE Computer Society, Los Alamitos, Kalifornien, USA (2006), S. 55–60.

Hevner, A.R.; March, S.T.; Park, J.; Ram, S.: Design Science in Information Systems Research. In MIS Quarterly, Volume 28, Heft 1 (2004), S. 75-105.

Hewlett Packard: HP Quality Center. http://www8.hp.com/us/en/software-solutions/software.html?compURI=1172141 (Zuletzt geprüft am: 07.05.2014).

IBM Inc.: Application Lifecycle Management. http://www-03.ibm.com/software/products/de/de/category/SW860 (Zuletzt geprüft am: 07.05.2014).

IBM Inc.: Rational Quality Manager. http://www-01.ibm.com/software/rational/products/rqm/ (Zuletzt geprüft am: 02.05.2014).

IBM Inc.: IBM Rational ClearCase, Version 8.0 (2011). http://public.dhe.ibm.com/common/ssi/ecm/en/rad10958usen/RAD10958USEN.PDF (Zuletzt geprüft am: 07.05.2013).

ISO/IEC: Software Testing, ISO/IEC 29119 (2012).

Jeng, J.-J.: Service-Oriented Business Performance Management for Real-Time Enterprise. In: Proceedings Joint Conference 8th IEEE International Conference on E-Commerce Technology (CEC 2006), 3rd IEEE; Piscataway, New Jersey. IEEE, Piscataway, New Jersey, USA (2006).

Josuttis, N.: SOA in der Praxis. System Design für verteilte Geschäftsprozesse. Dpunkt-Verlag, Heidelberg, (2008).

Köhler, P.T.: Applications Management. Springer, Berlin, Deutschland (2005).

Köppen, V.; Will, L.: Living SOA: Evolution des Betriebs von SOA-basierten Lösungen. In: ObjektSpektrum (2012), Heft 1, S. 42-46.

Krafzig, D.; Banke, K; Slama, D.: Enterprise SOA: Wege und Best Practices für serviceorientierte Architekturen: Einführung, Umsetzung, Praxis. mitp, Heidelberg, 1. Auflage (2007).

Liebhart, D.; SOA goes real: Service-orientierte Architekturen erfolgreich planen und einführen. Hanser, München u.a., Deutschland, 1. Auflage (2007).

Melzer, I.; Eberhard, S.: Service-orientierte Architekturen mit Web Services: Konzepte - Standards - Praxis. Spektrum Akad. Verl., Heidelberg, Deutschland, 3. Auflage (2008).

Oasis: UDDI Version 3.0.2: UDDI Spec Technical Committee Draft (2004). http://uddi.org/pubs/uddi-v3.0.2-20041019.pdf (Zuletzt geprüft am: 07.05.2014).

Oasis: Reference Model for Service Oriented Architecture (2011). http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.doc (Zuletzt geprüft am: 07.05.2014).

Office of Governance Commerce (Hrgs.): IT Infrastructure Library V3: information technology infrastructure library version 3 core. The Stationary Office, London (2007).

Oracle USA: Capacity Planning Guide v9.2.2.1 (2007). http://docs.oracle.com/cd/E11108_02/otn/pdf/install/E11092_01.pdf (Zuletzt geprüft am: 07.05.2014).

Papazoglou, M. G. D.: Service-Oriented Computing. In: *Communications of the ACM*, Jahrgang 46 (2003), Heft 10, S. 25–28.

Papazoglou, M.G.D.; Heuvel, W.-J.: Service oriented architectures: approaches, technologies and research issues. In: The VLDB Journal (2007), Volume 16/3, S. 389–415.

Pinto, G. H.; Vergilio, S. R.: A Multi-Objective Genetic Algorithm to Test Data Generation. In: Proceedings 24th International Conference on Tools with Artificial Intelligence (ICTAI); Athen, Griechenland. IEEE Computer Society, Piscataway, New Jersey, USA (2012).

Rapps, S.; Weyuker, E.: Selecting Software Test Data Using Data Flow Information. In: IEEE Transactions on Software Engineering, Jahrgang SE-11 (1985), Heft 4, S. 367–375.

Realtech AG: theGuard! CMDB: Automatisierte Datenerfassung in komplexen Infrastrukturen. http://www.realtech.com/wDeutsch/pdf/software/configuration-management/theGuard-CMDB.pdf (Zuletzt geprüft am: 07.05.2014).

Ribarov, L.; Manova, I.; Ilieva, S.: Testing in a service-oriented world. In: Proceedings International Conference on Information Technologies; St. Constantine and Elena, Bulgarien, Volume 2 (2007).

Robertson, S.; Robertson, J.: Mastering the requirements process. Addison-Wesley, Upper Saddle River, New Jersey, 2. Auflage (2006).

SAP AG: SAP Test Data Migration Server. http://help.sap.com/saptdm (Zuletzt geprüft am: 07.05.2014).

Schäfer, M. O.; Melich, M.: SAP Solution Manager. Galileo Press, Bonn, 1. Auflage (2012).

Scheer, A.-W.: ARIS – Vom Geschäftsprozess zum Anwendungssystem, Berlin, 4. Auflage (2002).

Schneider, T.: SAP-Performanceoptimierung: Analyse und Tuning von SAP-Systemen. Galileo Press, Bonn, Boston, 7. Auflage (2013).

Schulte, R.W.; Natis, Y.V.: Service Oriented Architectures Part 1. Gartner Inc., SSA Research Note SPA-401-068 (1996).

Schulte, R.W.; Natis, Y.V.: Service Oriented Architectures Part 2. Gartner Inc., SSA Research Note SPA-401-069 (1996).

Seidl, R.; Baumgartner, M.; Bucsics, T.: Basiswissen Testautomatisierung: Konzepte, Methoden und Techniken. dpunkt-Verlag, Heidelberg, (2012).

Simon, D.; Simon, F.: Testeffektivität vor Testeffizienz. In: ObjektSpektrum (2012), Heft 4, S. 8–12.

Software AG: Service-Oriented Architecture. https://www.softwareag.com/corporate/solutions/soa/soa_solution/overview/default.asp (Zuletzt geprüft am: 07.05.2014).

Sprott, D.: Business Flexibility through SOA. (2005). ftp://ftp.software.ibm.com/software/soa/pdf / CBDIWhitepaperBusinessFlexibiliyThroughSOA.pdf (Zuletzt geprüft am: 07.05.2014).

Starke, G.; Tilkov, S.: SOA-Expertenwissen: Methoden, Konzepte und Praxis serviceorientierter Architekturen. dpunkt-Verlag, Heidelberg (2007).

Starke, G.; Hruschka, P.: Software-Architektur kompakt. Angemessen und zielorientiert. Spektrum-Akademischer Verlag, Heidelberg (2011).

Teuber, L.; Weidmann, C; Will, L.: Monitoring und Betrieb mit dem SAP Solution Manager. Galileo Press, Bonn, 1. Auflage (2013).

The Open Group: Application Response Measurement: ARM 4.0 version 2 (2007). https://collaboration.opengroup.org/tech/management/arm/ (Zuletzt geprüft am: 07.05.2014).

Transaction Processing Performance Council: TPC Benchmarks. http://www.tpc.org/default.asp (Zuletzt geprüft am: 07.05.2014).

Trkman, P.; Kovačič, A.; Popovič, A.: Phasen der SOA-Einführung. In: Wirtschaftsinformatik, Jahrgang 53 (2011), Heft 4, S. 201–212.

van der Hoek, A. ; Hall, R. S. ; Heimbinger, D. ; Wolf, A. L.: Software Release Management. In: Proceedings 6th European Software Engineering Conference conference held jointly with the 5th ACM SIGSOFT, Zürich, Schweiz,. Springer, New York, USA (1997), S. 159–175.

Varshney, S.; Mehrotra, M.: Search based software test data generation for structural testing. In: *ACM SIGSOFT* Software Engineering Notes, Jahrgang 38 (2013), Heft 4, S. 1.

Will, L.: Operations requirements in SOA based solutions. In: Proceedings Fifth International Conference on Research Challenges in Information Science (RCIS), 19 - 21 Mai 2011; Gosier, Guadeloupe, Frankreich. IEEE, Piscataway, New Jersey, USA (2011), S. 1–10.

Will, L.: Requirements for Operations to Take Advantage of the Flexibility of SOA. In: Computer Science and Engineering, Jahrgang 2 (2012), Heft 5, S. 68–76.

Will, L. ; Köppen, V.: Zentrales, standardisiertes Monitoring als Grundlage des Service Level Managements in flexiblen SOA-Lösungen. In: Proceedings 42. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Braunschweig. Ges. für Informatik, Bonn. GI-Edition Thematics (2012), Band 6, S. 759–773.

Xie, L.; Luo, J.; Qiu, J.; Pershing, J.; Li, Y; Chen, Y.: Availability “weak point” analysis over an SOA deployment framework. In: Proceedings IEEE Network Operations and Management Symposium (NOMS 2008), Salvador, Bahia, Brasilien. IEEE Service Center, Piscataway, New Jersey, USA (2008), S. 473–480.

Zarnekow, R.; Brenner, W.; Scheeg, J.: Untersuchung der Lebenszykluskosten von IT-Anwendungen. In: Wirtschaftsinformatik, Jahrgang 46 (2004), Heft 3, S. 181-187.

Eigene Publikationen

Will, L; Köppen, V.: Zentrales, standardisiertes Monitoring als Grundlage des Service Level Managements in flexiblen SOA-Lösungen. In: Proceedings of Informatik 2012 GVS, Braunschweig, 2012.

Köppen, V.; Will, L.: Living SOA: Evolution des Betriebs von SOA-basierten Lösungen. In: OBJEKTspektrum, Nummer 1, Jan / Feb 2012

Will, L.: "Requirements for Operations to Take Advantage of the Flexibility of SOA. In Computer Science and Engineering, (2)5, S. 68–76, 2012.

Will, L.: Operations requirements in SOA based solutions. In Fifth International Conference on Research Challenges in Information Science (RCIS), 2011: 19 - 21 May 2011, Gosier, Guadeloupe ; proceedings: IEEE, Piscataway, NJ, S. 1–10, 2011.

Will, L.; Köppen, V.; Saake, G.: Flexibility in SOA Operations: The Need for a Central Service Component. In Process: SoEAEE'2014, Ulm (2014).

Teuber, L.; Weidmann, C.; Will, L.: Monitoring and Operations with SAP Solution Manager. Galileo Press, Boston, 1.Ausg. (2013).

Teuber, L.; Weidmann, C.; Will, L.: Monitoring und Betrieb mit dem SAP Solution Manager. Galileo Press, Bonn, 1. Ausg., (2013).

Föse, F.; Hagemann, S.; Will, L.: SAP NetWeaver AS ABAP – System Administration. Galileo Press, Boston, 4th ext. ed. (2012).

Föse, F.; Hagemann, S.; Will, L.: SAP NetWeaver AS ABAP - Systemadministration: Basiswissen für das SAP-Systemmanagement. Galileo Press, Bonn, 4., erw. Ausg. (2011).

Föse, F.; Hagemann, S.; Will, L.: SAP NetWeaver AS ABAP – System Administration: The Official SAP Guide. Galileo Press, Boston, 3rd ext.ed. (2009).

Föse, F.; Hagemann, S.; Will, L.: SAP NetWeaver AS ABAP - Systemadministration: Basiswissen für das SAP-Systemmanagement. Galileo Press, Bonn, 3., erw. Ausg. (2008).

Хагеман, С.; Вилл, Л.: SAP R/3. Системное администрирование. Лори (2010).

Hagemann, S.; Will, L.: SAP R/3系统管理. 东方出版社, Beijing, (2006).

Hagemann, S.; Will, L.: S.A.P. R/3システム管理ガイド. 日経BP社; 1版(2004).

Hagemann, S.; Will, L.: SAP R/3 Systemadministration. Galileo Press, Bonn, 2. Auflage (2002).

Hagemann, S.; Will, L.: SAP R/3 System Administration- The Official SAP Guide. Galileo Press, Boston, 2. Auflage (2003).

Вилл, Л.: SAP R/3. Системное администрирование- Официальное руководство SAP. Лори (1999).

Will, L.: SAP R/3 Administration Systeme- Configurez et administrez un systeme R/3. Campuspress by Pearson Education (2000).

Will, L.: SAP R/3 Gestion des Sistema-Conocimientos basicos para la gestion del sistema R/3. Gestion 2000, Madrid, (2001).

Will, L.: SAP R/3 System Administration- The Official SAP Guide. Sybex Inc. , 1. Auflage (1999).

Will, L.: SAP R/3 Systemadministration. Addison-Wesley. Bonn, 1. Auflage (1998).

Will, L.: SAP APO System Administration. Galileo Press, Boston (2003).

Will, L.: SAP APO Systemadministration. Galileo Press, Bonn (2002).

Schöler, S; Will, L; Schäfer, M. O. CobiT and the Sarbanes-Oxley Act: The SOX guide for SAP operations: Galileo Press, Bonn, Boston, 1. Auflage (2007).

Schöler, S; Will, L. SAP IT service & application management: [the ITIL guide for SAP operations]: Galileo Press, Fort Lee (NJ), Bonn, 1. Auflage (2006).

Ehrenerklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; verwendete fremde und eigene Quellen sind als solche kenntlich gemacht. Insbesondere habe ich nicht die Hilfe eines kommerziellen Promotionsberaters in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Ich habe insbesondere nicht wissentlich:

- Ergebnisse erfunden oder widersprüchliche Ergebnisse verschwiegen,
- statistische Verfahren absichtlich missbraucht, um Daten in ungerechtfertigter Weise zu interpretieren,
- fremde Ergebnisse oder Veröffentlichungen plagiiert,
- fremde Forschungsergebnisse verzerrt wiedergegeben.

Mir ist bekannt, dass Verstöße gegen das Urheberrecht Unterlassungs- und Schadensersatzansprüche des Urhebers sowie eine strafrechtliche Ahndung durch die Strafverfolgungsbehörden begründen kann. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form als Dissertation eingereicht und ist als Ganzes auch noch nicht veröffentlicht."

Magdeburg, den 04.09.2014

Liane Will