Volker Stiehl

Prozessgesteuerte Anwendungen entwickeln und ausführen mit BPMN

Wie flexible Anwendungsarchitekturen wirklich erreicht werden können

dpunkt.verlag

Dr. Volker Stiehl studierte Informatik an der Universität Erlangen-Nürnberg. Nach 12 Jahren als Entwickler und Berater bei Siemens begann er im Jahre 2004 seine Arbeit bei der SAP AG in Walldorf. Er ist heute als Chief Product Expert Teil des Produktmanagement-teams für SAP NetWeaver Process Integration, SAP's SOA Middle-wareprodukt für Systemintegrationen. Im Jahre 2011 promovierte er an der TU Darmstadt über die systematische Konstruktion von Anwendungen unter Verwendung von BPMN.

Volker Stiehl

Prozessgesteuerte Anwendungen entwickeln und ausführen mit BPMN

Wie flexible Anwendungsarchitekturen wirklich erreicht werden können

dpunkt.verlag

Volker Stiehl
volker.stiehl@sap.com

Lektorat: Christa Preisendanz
Copy Editing: Annette Schwarz, Ditzingen
Herstellung: Nadine Thiele
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

ISBN:
Buch 978-3-86490-007-5
PDF 978-3-86491-229-0
ePub 978-3-86491-230-6

1. Auflage 2013

Ringstraße 19B
69115 Heidelberg

5 4 3 2 1 0

Geleitwort von Sascha Bertsch

Die Idee hinter dem Prinzip *Separation of Concerns* – die Zerlegung von Anforderungen und Aufgaben einer Softwarearchitektur mit dem Ziel der Modularisierung und Kapselung von Verantwortlichkeiten – einschließlich der Sicherstellung einer arbeitsteiligen Strukturierung – führt nun auch in der Praxis prozessgesteuerter Anwendungen zu entscheidenden Vorteilen im Hinblick auf Organisation und Aufbau von Prozess- und Servicelandschaften.

Während eine rein serviceorientierte Ausrichtung dabei rückblickend oftmals aufgrund organisatorischer Hürden, technischer Restriktionen und hoher Abhängigkeiten einer direkten Nutzung im Wege steht, können die sich ergebenden fachlichen Anforderungen an eine prozessgesteuerte Anwendung deren Einsatz sogar noch verstärken und weiterentwickeln. Die konsequente fachliche Ausrichtung sowie die Fokussierung auf die tatsächlich notwendigen Daten, Dienste und Funktionalitäten beim Aufbau dieser Anwendungen führen bei den Beteiligten der Fach- und IT-Ebene zu einem besseren Verständnis sowohl über die tatsächlich notwendigen Prozessinhalte allgemein als auch über die Notwendigkeit zur Bereitstellung vollumfänglicher Dienste und Komponenten der führenden Systeme auf Grundlage eines längerfristigen Planungshorizonts.

Welche der im vorliegenden Buch erworbenen positiven Rückschlüsse und wichtigen Kernaussagen können wir nun unweigerlich aus der betrieblichen Praxis bei der Entwicklung geeigneter Prozess- und Anwendungsarchitekturen ableiten?

1. IT-gestützte Geschäftsabläufe gruppieren im Rahmen ihrer Zuständigkeiten fachlich-, technisch- und nachrichtenzentrische Prozesse. Die Vorteile und Gemeinsamkeiten werden in der Verwendung einheitlicher Notationen und Standards (z.B. *Business Process Model and Notation*) sowie einer lose gekoppelten, auf asynchroner Kommunikation basierenden Vertragsbeziehung erreicht. Der operative Betrieb, die Pflege, Wartbarkeit und Weiterentwicklung erfolgen komponentenbasiert, erhöhen die Agilität, reduzieren die Komplexität und können rollenspezifisch abgegrenzt werden.
2. Eine idealerweise gewachsene Landschaft von unternehmensweit akzeptierten Fachobjekten und Diensten für Stamm- und Bewegungsdaten vorausgesetzt, lassen sich durch Nutzung technisch sinnvoller Integrationsschnittstel-

len und Methoden der Serviceabstraktion flexible Prozessarchitekturen und -muster ableiten, die eine klare fachliche Herangehensweise bei der Bereitstellung von Anwendungen und ihren Anbindungen an die datenführenden Systeme erlauben und technologische Restriktionen wie beispielsweise die Schnittstellenbeschaffenheit minimieren. Sichergestellt wird dies durch die Berücksichtigung vertraglich festgelegter (Dienstleistungs-)Beziehungen zwischen den Prozess- und Anwendungsebenen der Gesamtarchitektur, die konsequent und durchgängig mithilfe des Top-down-Ansatzes definiert werden. Diese Vorgehensweise führt zu einer zielgerichteten und aufgabenfokussierten Umsetzung.

3. Eine prozessgesteuerte Anwendung benötigt neben einer klaren Prozessuntergliederung die Einbindung weiterer Integrationsebenen. Dienste müssen insbesondere auch für Ebenen der Geschäftslogik, der Benutzeroberflächen oder der Regelwerke bereitgestellt werden bzw. werden von diesen konsumiert. Dabei werden nur diejenigen Fachdaten in der Anwendung dauerhaft selbst gespeichert, die tatsächlich im Hoheitsgebiet der jeweils neu geschaffenen Anwendung liegen, während fremde Fachdaten konsistent über Schnittstellen disponiert werden.
4. Neben einer Gliederung lokaler Prozess- und Serviceebenen empfehlen wir aus unseren praktischen Erfahrungen heraus den Einsatz eines Mediators (z.B. Enterprise Service Bus). Innerhalb dieses auf Nachrichtenaustausch basierenden Systems werden Dienstleistungen ausgelagert, die den kontrollierten externen Zugriff auf Fachdaten führende Systeme gewährleisten, steuern und aufbereiten. Der Mediator gleicht durch seine Funktionen und Fähigkeiten aufkommende Änderungsanforderungen aus und schützt die vorgelagerte prozessgesteuerte Anwendung vor eigenen Anpassungen aufgrund von Fremdeinfluss. Der Mediator unterstützt ferner den Aufbau applikationsspezifischer Schnittstellen innerhalb und an den Nahtstellen einer Prozessanwendung.

Als ein Bestandteil des gesellschaftsübergreifenden Beratungs- und Dienstleistungsportfolios setzen wir als der IT-Dienstleister der EnBW AG solche querschnittsbezogenen Anwendungen und Lösungen durch die Themenfelder *Business Process Management* (BPM), *Application Integration* (AI) und *Application Development* (AD) um. Die Lösungsentwicklung und der operative Betrieb setzen dabei unter anderem auf den Plattformen *SAP NetWeaver Composition Environment* sowie *SAP NetWeaver Process Integration* der SAP AG auf. Beide Produkte kommunizieren mit einer Vielzahl heterogener Systeme in der Unternehmenslandschaft und bilden dabei annähernd tausend Szenarien in unterschiedlichen Ausprägungsstufen und Technologien ab. Entscheidende Kriterien bei der Wahl geeigneter Integrations- und Prozesswerkzeuge sind ihre Möglichkeiten in Bezug auf Anpassungsfähigkeit und Flexibilität, die sich unweigerlich

aus dem Änderungsbedarf in stark wandelnden Märkten und sich ständig veränderten Rahmenbedingungen ergeben.

Jedoch stellen die Positionierung und Verwendung der am Markt verfügbaren Werkzeuge für den Realisierungserfolg prozessgesteuerter Szenarien nur einen Teilaspekt dar. Weitere Gesichtspunkte betreffen den Aufbau und die Verfügbarkeit fachlicher Komponenten sowie die Anwendung qualifizierter Methoden und Verfahren, um von ihren uneingeschränkten Vorteilen zu profitieren. Die im vorliegenden Werk aufgezeigten Zusammenhänge und Mechanismen zur Umsetzung prozessgesteuerter Anwendungen enthalten aus unserer heutigen Sichtweise die wichtigsten und entscheidenden Architekturprinzipien und Richtlinien, um den Herausforderungen IT-gestützter Geschäftsprozesse zukunftssicher begegnen zu können.

Volker Stiehl liefert uns hierzu in einzigartiger Weise die fundamentalen Impulse und Ideen, die bei der Lösungsumsetzung und konkreten Problemstellungen im praktischen Alltag beachtet werden sollten. Seine in diesem Buch erläuterten Architekturansätze haben nicht nur uns rückblickend geholfen, einen entscheidenden Mehrwert zu leisten, sondern dürften auch allen interessierten Lesern und Anwendern von hohem Nutzen sein.

Sascha Bertsch
Professional Consultant für Integrationstechnologien der
EnBW Systeme Infrastruktur Support GmbH

Geleitwort von Prof. Dr. Erich Ortner

Volker Stiehl präsentiert uns in seinem vorliegenden Buch »Prozessgesteuerte Anwendungen entwickeln und ausführen mit BPMN« etwas Fundamentales, nämlich die klare Unterscheidung zwischen einem Systemprogrammierer, also einem klassischen *Software Engineer,* auf der einen und einem Organisationsprogrammierer oder, wie wir heute auf Neudeutsch sagen, einem *Enterprise Engineer* auf der anderen Seite. Damit läutet er auf über 350 Seiten eine neue Ära ein, und das über 30 Jahre nach dem ersten Aufkommen dieser grundsätzlichen Feststellung und ihrer daraus resultierenden Aspekte für den IT-Einsatz in Unternehmen.

Doch was ist denn nun der Unterschied und warum können wir hier mit Fug und Recht vom Einläuten einer neuen Ära sprechen? Die Antwort liegt auf der Hand. Der Software Engineer arbeitet formdominiert (operative Syntax) und denkt – Edsger W. Dijkstra sei Dank – in Blockstrukturen (z.B. »Programmieren im Großen« mit BPEL). Beim Enterprise Engineer hingegen dominieren der Inhalt (Business Logic) und ein Denken in Flussstrukturen (z.B. BPMN). Und noch eine weitere, sehr pragmatische Tatsache unterscheidet die beiden Entwickler: Die Nachfrage nach Enterprise Engineers wird durch das Angebot nicht annähernd gedeckt; der Bedarf nach (reinen) Software Engineers dagegen ist deutlich geringer und überdies rückläufig.

Und eine neue Ära wird deswegen eingeläutet, weil wir längst in unserem IT-unterstützten Arbeiten und auch Leben geprägt sind von drei Übergängen: der Fokussierung auf ein die komplette Organisation betreffendes Enterprise Engineering statt »nur« einem Software Engineering, der Gestaltung von Anwendungssystemen statt »nur« Informationssystemen und einer Ereignisverarbeitung statt »nur« einer reinen Datenverarbeitung. Ereignisse sind eben grundsätzlich etwas anderes als Daten und erfordern eine andere Aufnahme, Verarbeitung und Lenkung.

Und damit bietet uns das vorliegende Buch nun sowohl eine Orientierung als auch eine spezifische Basis für verschiedene in den nächsten Jahren zu bewältigende Aufgaben, die sich zwangsläufig aus den o.g. Feststellungen ergeben:

- Eine universelle *Methodenlehre*, Tool-unterstützt und mit einer Anwendungsentwicklungsumgebung ausgestattet, wie sie in diesem Buch für die Entwicklung von prozessgesteuerten Anwendungen angeregt bzw. von innovativen »Metaunternehmen« wie beispielsweise TECHNUM bereits zur Verfügung gestellt wird.
- *Lehrpläne* und *Seminare* für die Aus- und Weiterbildung von z.B. Software Engineers. Denn das neue Paradigma zur Entwicklung von Anwendungssystemen für das IT-unterstützte Arbeiten in Organisationen sowie in digitalen Gemeinschaften (z.B. Social Media) ist in den tradierten Ausbildungs- und Studiengängen der Informatik und Bindestrich-Informatiken noch deutlich zu wenig und wenn, dann selten konsequent genug verankert.
- Ein bis dato in der IT-Branche – bei Herstellern wie bei Anwendern – noch nie dagewesener, aber unausweichlicher Kraftakt zur *Umgestaltung* der bestehenden Verantwortlichkeiten und eingeführten IT-Systeme im Spannungsfeld »Mensch, Technik & Organisation«, ausgerichtet an Ereignissen und deren Lenkung. Dies ist dringend notwendig, um ein besseres Management der IT-Nutzung in Unternehmen zu gewährleisten. Aber nicht nur dort, denn Gleiches gilt selbstverständlich auch für eine sinnvolle und faire Nutzung der IT im Privaten.

Das *Ubiquitous Computing* und seine Möglichkeiten für uns Menschen wirken sich längst global und – thematisch gesehen – auch auf anderen Aufgabenfeldern nachhaltig aus. Es führt zu einem *Ubiquitous Modeling* und dementsprechend einer *Ubiquitous Compliance*, also der vollständigen Erfüllung des durch die Modellierung vorgegebenen Geschehens in den jeweiligen Anwendungsbereichen. Folgerichtig sind die Enterprise Engineers daher künftig auch für inhaltlich fehlerhafte Anwendungen – wie es rückwirkend für die globale Finanz- oder Schuldenkrise und die dort eingesetzten IT-Anwendungen natürlich nicht mehr möglich ist – mit zur Verantwortung zu ziehen.

Zum Zwecke einer effizienten, aber auch friedlichen Bewältigung dieser Herausforderungen sei diesem Buch eine möglichst große Verbreitung gewünscht. »Mehr Unterstützung und weniger Versklavung« heißt, ein wenig provokant, für die Zukunft die Devise – auch und vor allem vom Standpunkt der Algorithmik und IT aus gesehen. Wir Menschen hätten es verdient, trotz des in vielen Anwendungsbereichen übertriebenen Gebrauchs des Begriffs »Industrialisierung«.

Prof. Dr. Erich Ortner
Direktor TECHNUM
Steinbeis-Unternehmen für Prozesstechnologie

Vorwort

Der Beginn meiner Reise in die Welt der Anwendungsentwicklung unter Berücksichtigung komplexer verteilter Systemlandschaften liegt mehr als 10 Jahre zurück: Ich arbeitete seinerzeit bei Siemens Business Services (SBS) in einer Abteilung zur Entwicklung Java-basierter Anwendungen und konnte mitverfolgen, wie unser Entwicklungsteam an einer neuen Software zur Umsetzung neuer Geschäftsprozesse für die Verwaltung von Softwarelizenzen arbeitete. Es ging darum, die in einem Unternehmen eingesetzten Lizenzen zentral in einem Lizenz-Pool zu sammeln. Beim Kauf neuer Rechner installierte die IT-Abteilung entsprechend der Rolle des Mitarbeiters, für den der Rechner vorgesehen war, passende Software und entnahm sie dazu dem Lizenz-Pool. Im umgekehrten Fall, also immer dann, wenn ein Computer ausgemustert wurde oder ein Mitarbeiter das Unternehmen verließ, wanderte die Lizenz in den Pool zurück und stand folglich für weitere Installationen wieder zur Verfügung. Die neu zu entwickelnde Anwendung musste zudem mit einer Vielzahl bestehender Systeme interagieren: Die Bestellung neuer Lizenzen wickelten wir beispielsweise über ein SAP-System ab, die Informationen über Mitarbeiter steckten in einem LDAP-System (Lightweight Directory Access Protocol), der Versand der Workflow-Benachrichtigungen erfolgte über einen E-Mail-Server. Abgerundet wurde die Applikation um eine eigene JDBC-basierte Persistenz zur Speicherung der Informationen, welcher Benutzer welchen Rechner besitzt und welche Software sich auf dem Rechner befindet. Letztendlich handelte es sich also um eine klassische verteilte, webbasierte Anwendung, die neue Prozesse implementierte, die sich über mehrere Systeme hinweg erstreckte.

Das Team entwickelte die Lösung nach allen Regeln der damals bekannten JavaEE-Kunst: Session Beans, Entity Beans, Message-driven Beans, Servlets, Java Server Pages, Apache Struts als Web-Framework, Webservices, die gesamte Palette also. Zudem legte unser Architekt sehr viel Wert auf eine moderne Softwarearchitektur: Ich erinnere mich noch gut an den Tag, als er bei mir an meinem Schreibtisch vorbeikam und mir voller Stolz berichtete, wie viele Pattern der »Gang of Four« [Gamma et al. 1995] er in der erstellten Software sinnvoll einbringen konnte (obwohl aus der Anzahl der verwendeten Pattern allein natürlich

nicht auf die Qualität einer Software geschlossen werden kann). Die Software lief auf einem BEA Weblogic Server und fand schließlich auch ihren Weg auf einen SAP Web Application Server [Stiehl 2004]. Die Kundenresonanz auf die neue Lösung war äußerst positiv, doch es stellte sich alsbald heraus, dass potenzielle Kunden ihre Daten in anderen als den von uns bei der Erstellung der Software vorgesehenen Systemen ablegten. Wie sollten wir aber mit diesem Problem umgehen, wenn unsere Software weiterhin für einen möglichst großen Kundenkreis attraktiv sein sollte? Für jeden Kunden eine neue Variante erstellen? Bei der Anzahl möglicher Kombinationen von Backend-Systemen ist das sicherlich kein vielversprechender Ansatz. Die Anfälligkeit der Lösung bezüglich der Änderungen an der Systemlandschaft war offensichtlich, oder, um es mit anderen Worten auszudrücken, wir hatten uns zu sehr von einer konkreten IT-Landschaft abhängig gemacht. Sie werden sich jetzt vielleicht fragen, wie es dazu kommen konnte und warum die Lösung nicht von vornherein mehr von den Backend-Systemen abstrahierte? Aber wie setzt man eine solche Abstraktion konkret um? Wie weit geht die Abstraktion? Wie werden die Aufgaben auf die Abstraktionsebenen verteilt? Und überhaupt: Sollte die serviceorientierte Architektur nicht genau diese Probleme lösen? Letztendlich gipfeln all diese Fragestellungen in der einen zentralen Frage: Wie erstelle ich robuste Anwendungen für verteilte IT-Landschaften, die zukünftig durch die steigende Zahl von On-Demand- und Cloud-Angeboten sogar noch fragmentierter werden?

Je mehr ich mich damit auseinandersetzte, Projekte analysierte, Fachartikel las und auf Konferenzen Vorträge über ähnlich geartete Problemstellungen hörte, umso mehr musste ich feststellen, dass die dort vorgestellten Architekturen mit ähnlichen Unzulänglichkeiten zu kämpfen hatten. In Gesprächen mit den Projektverantwortlichen wurden oft weitere Gründe genannt, die zu derartigen Abhängigkeiten führten: Ein hoher Projektdruck verbunden mit fehlender Zeit bei einem knapp bemessenen Budget verhinderten die Ausarbeitung einer robusten Softwarearchitektur, obwohl wider besseren Wissens die Folgen eines derartigen Vorgehens hinlänglich bekannt sind: Die kurzfristig schnellere und damit kostengünstigere Entwicklung verschlingt später, auf lange Sicht betrachtet, ein wesentlich höheres Budget für die Wartung, Betreuung und Weiterentwicklung, als dies notwendig gewesen wäre. Ein langfristiges Ziel wurde also einem kurzfristigen geopfert! So dürfen wir uns nicht wundern, wenn allseits eine fehlende Agilität bei der Anpassung der Software an die vom Management vorgegebene Geschäftsstrategie beklagt wird, da niemand mehr in der Lage ist, die Verflechtungen aufgrund der gewachsenen Abhängigkeiten zwischen Endanwendungen und Systemlandschaft auseinanderzudividieren.

In diesem Buch stelle ich daher einen Ansatz zur Umsetzung einer Architektur für Endanwendungen vor, die eine vernünftige Balance zwischen Entwicklungs- und Wartungskosten, Nachhaltigkeit, Skalierbarkeit, Fehlertoleranz sowie die Erfüllung von Flexibilitätsanforderungen bei gleichzeitiger moderater Komplexi-

tät anstrebt und auf eine weitestgehende Unabhängigkeit der Endanwendung gegenüber der Systemlandschaft, gegen die sie operiert, setzt. Auf der Suche nach einer solchen Lösung bin ich überraschenderweise bei dem Prozessmodellierungsstandard BPMN (Business Process Model and Notation) fündig geworden: Gestartet als eine grafische Notation zur Modellierung von Geschäftsprozessen, entpuppte sie sich als geradezu vorzügliches Mittel für die *Implementierung* einer Anwendungsarchitektur, wie sie mir vorschwebte. Dabei verstehe ich unter *Anwendungsarchitektur* eine Lösung für Endanwendungen auf Basis heterogener IT-Landschaften, die sowohl fachliche als auch integrationszentrische Prozesse unterscheidet. Da mit der neuesten Version 2.0 des BPMN-Standards zudem die Ausführbarkeit derart erstellter Prozessmodelle aufgrund semantischer Erweiterungen gewährleistet wurde, begann ich damit, komplette Anwendungsarchitekturen mithilfe der BPMN zu entwerfen und ablaufen zu lassen. Dabei verwende ich die BPMN eben nicht nur zur Modellierung der fachlichen Prozesse der Endanwendung, wie das »B« in BPMN ja suggeriert, sondern ich nutze sie auch zur Modellierung und Ausführung der systemnahen Integrationsprozesse. Wie diese auf den ersten Blick widersprüchliche Verwendung der BPMN zusammenpasst, erfahren Sie ebenfalls in diesem Buch.

Mit der vorgeschlagenen Anwendungsarchitektur verfolge ich das Ziel, Architekten und Softwareentwicklern eine möglichst detaillierte Architektur-Blaupause in die Hand zu geben, nach deren Prinzipien zukünftig verteilte Anwendungen in den Unternehmen geplant und implementiert werden können. Wird die Anwendungsentwicklung vermehrt nach dieser Vorlage umgesetzt, trägt jede dieser Lösungen dazu bei, dass sich Unternehmen ganz allmählich aus dem eingangs erwähnten Würgegriff der Abhängigkeiten und Systemverbindungen lösen können. Von daher sehe ich die prozessgesteuerten Anwendungen als einen Beitrag zum Unternehmensarchitekturmanagement (Enterprise Architecture Management, EAM) an, womit auch gleichzeitig gesagt ist, dass dieses Buch sich vom großen Feld des EAM unterscheidet und EAM selbst nicht zum Gegenstand macht.

Das Buch richtet sich an Softwarearchitekten, technisch interessierte Manager, IT-Leiter, Softwareentwickler, Projektleiter, aber auch an Studenten der Informatik und Wirtschaftsinformatik. Ich setze Grundkenntnisse der BPMN voraus. Ich führe zwar kurz in die Notation ein, beschreibe allerdings nicht jedes BPMN-Element im Detail. Auch im Hinblick auf serviceorientierte Architekturen bin ich von einem grundlegenden Wissen bei meinen Lesern ausgegangen, da ich natürlich bei meinem Ansatz ebenfalls auf wiederverwendbare Dienste in den Backend-Systemen angewiesen bin.

Danksagungen

Ein solches Buch entwickelt sich über einen sehr langen Zeitraum, während dessen eine Vielzahl von Personen involviert ist. Ich möchte mich ganz herzlich bei all den Menschen bedanken, die mich auf diesem Weg begleitet haben: bei meinem Doktorvater Prof. Dr. Erich Ortner, bei meinen Gutachtern Prof. Em. Dr. Dr.-Ing. E.h. Hartmut Wedekind und Prof. Dr. Mathias Weske. Bei meinen Vorgesetzten Gunther Rothermel, Dr. Achim Kraiss und Sindhu Gangadharan sowie meinen Kollegen Dr. Udo Paltzer, Dr. Alexander Bundschuh und Waldemar Befort. Zudem bin ich folgenden Personen sehr zu Dank verpflichtet (in alphabetischer Reihenfolge): Sascha Alber, Prof. Dr. Thomas Allweyer, Dr. Dirk Ammermann, Sascha Bertsch, Dr. Gero Decker, Marcus Elzenheimer, Matthias Fischer, Prof. Dr. Elisabeth Heinemann, Nicolai Josuttis, Holger Koschek, Marco Link, Dr. Daniel Lübke, Christoph Neumann, Prof. Dr. Gunther Piller, Peter Schwab sowie Nicole Zeise.

Ein besonderer Dank geht an das Team vom dpunkt.verlag, vor allem an Christa Preisendanz. Doch all dies wäre undenkbar gewesen, gäbe es da nicht noch meine Familie, die mir in jeder Phase der Entstehung dieses Buches den Rücken gestärkt und viele Entbehrungen in Kauf genommen hat: Christa, Clara und Vera – ihr seid einfach wunderbar – danke!

Mit Sicherheit habe ich bei der Aufzählung den oder die eine(n) oder andere(n) vergessen. Man möge es mir bitte nachsehen.

Noch eine Anmerkung zum Schluss: Auch wenn ich zurzeit bei der SAP AG angestellt bin und in Kapitel 4 für die Implementierung der vorgeschlagenen Architektur auf das Softwarepaket SAP NetWeaver Process Orchestration zurückgreife, so handelt es sich bei diesem Buch dennoch um kein offizielles SAP-Buch. Ich möchte betonen, dass sämtliche im Buch gemachten Aussagen und Empfehlungen ausschließlich meine persönliche Meinung widerspiegeln. Zudem sind die Empfehlungen bewusst generisch gehalten und lassen sich daher mit jeder vergleichbaren Plattform umsetzen.

Mir bleibt jetzt nur noch, Ihnen, meinen Lesern, viel Freude bei der Lektüre zu wünschen. Verbunden damit ist natürlich die Hoffnung, dass Sie möglichst viele Anregungen mitnehmen und direkt in Ihrer praktischen Arbeit einsetzen können, um auch für Ihr Unternehmen oder für Ihre Kunden Anwendungen zu erstellen, die das Prädikat »flexibel« verdienen!

Volker Stiehl
Möhrendorf, im August 2012

Inhaltsverzeichnis

1 Einleitung

1.1 Unternehmenssoftware im Zeitalter der Globalisierung

Keine Frage: Heutige Unternehmen sehen sich aufgrund der Globalisierung einem erhöhten Wettbewerb ausgesetzt. Sie müssen im alltäglichen Kampf ums Überleben schneller und nachhaltiger als die Konkurrenz auf Veränderungen reagieren können. Besonders treffend fasst in diesem Zusammenhang das folgende von Charles Darwin stammende Zitat die Situation zusammen:

It is not the strongest of the species that survives,
nor the most intelligent that survives.
It is the one that is the most adaptable to change.

Frei übersetzt: Es ist weder der Stärkste noch der Intelligenteste, der überlebt.
Es ist derjenige, der sich am besten dem Wandel anpassen kann.

Es ist genau die Fähigkeit von Unternehmen, auf Veränderungen adäquate Antworten parat zu haben, die ihnen ihre Existenz letztendlich garantiert. Dazu kann die Informationstechnologie (IT) maßgeblich beitragen: Dank enormer Fortschritte im Software Engineering, wie z. B.

- der modellgetriebenen Softwareentwicklung,
- der Standardisierung von Kommunikations- und Schnittstellentechnologien,
- der Modularisierung fachlicher Funktionalitäten in Form von wiederverwendbaren Services,
- der Produktivitätssteigerungen für Softwareentwickler aufgrund hochintegrierter Entwicklungsumgebungen sowie
- dem Einsatz agiler Entwicklungsmethoden,

ist die effiziente Bereitstellung neuer Softwarelösungen weniger das Problem. Stattdessen erweisen sich aus der Gesamtkostenbetrachtung, der Total Cost of Ownership (TCO) also, die Pflege, die Erweiterung sowie die Anpassung der Software an veränderte Bedingungen als viel größere Hürden: Aufgrund erhöhten Zeit- und Kostendrucks entstehen Softwarearchitekturen, die genau der immer wichtiger werdenden Fähigkeit eines Softwaresystems zur Anpassbarkeit entge-

genwirken. Dies gilt insbesondere für Software, die bereits existierende fachliche Funktionalität zu neuen, system- und anwendungsübergreifenden Applikationen gemäß der SOA-Idee zusammenfügen soll. Der ursprüngliche Gedanke einer serviceorientierten Architektur (SOA), nämlich Potenziale wie gesteigerte Agilität, geringere Kosten, Wiederverwendbarkeit sowie eine zeitnahe Umsetzung der Unternehmensstrategie in Software heben zu wollen, sind in den Projekten nicht wie erwartet erfüllt worden. Fast schon legendär ist da schon der von Anne Thomas Manes veröffentlichte Blog geworden, in dem sie die Beerdigung von SOA ankündigt [Manes 2009]. Es gibt sicherlich eine Vielzahl von Gründen, warum die Erwartungen nicht in dem Maße wie erhofft erfüllt wurden, was zweifelsohne auch mit der nach wie vor nicht vorhandenen Definition von SOA zusammenhängt. Ich möchte an dieser Stelle auf einen Grund verweisen, der auf den ersten Blick gar nicht so offensichtlich ist und mit der Übernahme der aus der Programmierung eng gekoppelter Lösungen stammenden Erfahrungen beim Anwendungsentwurf in die serviceorientierte Welt zusammenhängt: Architekturen, die in eng gekoppelten Standalone-Anwendungen vollkommen richtig waren, sind in lose gekoppelten verteilten Anwendungen schlichtweg untauglich. Ob es um die Verwendung von Datentypen (Auseinandersetzung mit nur einem Datentypsystem vs. Umgang mit mehreren Datentypsystemen), der Kommunikation mit Systemen oder das Transaktionsverhalten geht – in einer verteilten Umgebung sind dazu andere Lösungen notwendig. Sie verlangen ein Umdenken bei den Entwicklern: So wie bei jedem Paradigmenwechsel sind veränderte Herangehensweisen die logische Konsequenz. Ob beim Wechsel von der maschinennahen zur strukturierten Programmierung, ob von der strukturierten zur objektorientierten Programmierung oder eben wie zurzeit der Wandel hin zur Entwicklung von serviceorientierten Architekturen: In jedem dieser Evolutionsschritte nimmt die Abstraktion zu und verlangt eine Auseinandersetzung des Entwicklers mit den neuen Paradigmen, um optimal von den neuen Möglichkeiten profitieren zu können. Alte Gewohnheiten in die neue Welt transportieren zu wollen hilft in den wenigsten Fällen und ist eher hinderlich. Prozessgesteuerte Anwendungen oder Composite Applications, kurz auch Composites genannt – diese Begriffe werde ich im Laufe dieses Buches synonym verwenden –, bilden hier keine Ausnahme: Sie kontrollieren eben nicht mehr alleine die beteiligten Ressourcen (wie zum Beispiel Datenbanksysteme oder ERP-Systeme). Wir bewegen uns vielmehr in einer heterogenen Systemlandschaft, in der es existierende Dienste zu neuer Geschäftslogik zu kombinieren gilt, was schlussendlich ein gesteigertes Problembewusstsein bei den Anwendungsentwicklern erfordert.

Damit sind die Anforderungen an derartige Systeme aber bei weitem nicht erschöpft. Der Flexibilitätsaspekt ist ebenfalls nicht zu unterschätzen und lässt sich konkret an zwei Anforderungen verdeutlichen:

1. Bedarf an Änderungen innerhalb der prozessgesteuerten Anwendung selbst, insbesondere Änderungen an den Geschäftsprozessen,
2. laufende Veränderungen innerhalb der Systemlandschaft, auf der die prozessgesteuerte Anwendung basiert.

Änderungen an den Geschäftsprozessen

Geschäftsprozesse (im klassischen Sinne zu verstehen als »eine Reihe von festgelegten Tätigkeiten (Aktivitäten, Aufgaben), die von Menschen oder Maschinen ausgeführt werden, um ein oder mehrere Ziele zu erreichen« (aus [EABPM 2009], siehe auch [Hammer & Champy 1993] oder [Gaitanides 1983]) spielen bei prozessgesteuerten Anwendungen eine besondere Rolle: Um sich von der Konkurrenz absetzen zu können, müssen Unternehmen ständig an genau den Geschäftsprozessen feilen, die ihnen einen Wettbewerbsvorteil sichern. Erfolgreiche Prozesse werden dabei unweigerlich von der Konkurrenz kopiert, so dass der Vorteil wieder schrumpfen wird. Hier sei beispielhaft an Fluggesellschaften erinnert: Die Bereitstellung von Selbstbedienungsterminals zum Einchecken brachte dem Kunden des Unternehmens, das diese Idee als Erstes umsetzte, den Vorteil verkürzter Wartezeiten (und damit verbunden eine höhere Kundenzufriedenheit) und dem Unternehmen selbst den Vorteil, Personal und damit Kosten einsparen zu können. Heute stellen nahezu alle Fluggesellschaften einen solchen Service zur Verfügung, so dass die Vorteile wieder verloren gingen. Auch geänderte Marktbedingungen verlangen von den Unternehmen eine schnellstmögliche Anpassung ihrer Prozesse. Was auch immer der Auslöser sein mag, die Folgen werden stets dieselben sein: Ein neuer Innovationszyklus zwingt Unternehmen schließlich dazu, weitere Optimierungen an den geschäftskritischen Prozessen vorzunehmen, um weiterhin erfolgreich agieren zu können. Je schneller solche Änderungen implementiert werden können, umso wahrscheinlicher wird das Vorgehen von Erfolg gekrönt sein. Folglich muss die Änderungsfreundlichkeit der Software in der Architektur von prozessgesteuerten Anwendungen bereits verankert sein, um den Bedarf nach Flexibilität und Agilität gerecht werden zu können.

Änderungen in der Systemlandschaft

Auf der anderen Seite müssen prozessgesteuerte Anwendungen auf eine sich ständig ändernde Systemlandschaft vorbereitet sein. Es gehört zum Kern von Composite Applications, dass sie in einem heterogenen Systemumfeld agieren: Der Wiederverwendungsgedanke, den wir ja bei SOA schon gefunden haben, spielt also auch hier eine wichtige Rolle. Das Rad soll eben nicht immer wieder neu erfunden werden, und wo etablierte, funktionierende Geschäftslogik bereits existiert, soll diese auch entsprechend genutzt werden. Allerdings ergibt sich daraus eine besondere Herausforderung: Derartige Systemlandschaften sind nicht stabil. Auch hier müssen Veränderungen von vornherein bei der Konzeption von pro-

zessgesteuerten Anwendungen berücksichtigt werden. Mindestens drei Gründe sprechen dafür, dass sich Systemlandschaften auch in Zukunft ständig, vielleicht sogar in einem noch höheren Tempo, verändern werden:

1. Systemkonsolidierungen
2. Firmenfusionen
3. Software as a Service (SaaS), On-Demand, Cloud

Schauen wir uns die drei Punkte im Detail an: IT-Abteilungen in Unternehmen unterliegen einem enormen Kostendruck. Die Unternehmensleitung verlangt von IT-Verantwortlichen eine ständige Kostenoptimierung. *Systemkonsolidierungen* sind da ein probates Mittel: Je mehr Systeme und Funktionalitäten zusammengelegt werden können, umso geringer werden die Belastungen. Allein aus diesem Grund werden Systemlandschaften sich ständig weiterentwickeln. Für prozessgesteuerte Anwendungen bedeutet dies, dass Aufrufe, die zuvor gegen mehrere Systeme erfolgten, nun gegen eine reduzierte Anzahl von Systemen abgewickelt werden müssen.

Doch das ist nur die eine Seite der Medaille: Durch das Abschalten von Systemen und die Überführung von ehemals differierenden Prozessen in Standardsoftware wird zwar eine Vereinfachung der IT-Landschaft erreicht. Diesem Trend entgegen wirkt jedoch die *Fusion von Unternehmen*. Zur Erreichung bestimmter Wachstumsziele für die Unternehmung wird das organische Wachstum durch Firmenakquisitionen unterstützt. Allerdings wächst durch derartige Firmenzusammenschlüsse naturgemäß der System- und Anwendungspark. In diesen Fällen ist die Composite Application gemäß der geänderten Rahmenbedingungen an die gestiegene Systemanzahl anzupassen.

Schließlich muss man dem neuen Trend der *Software-as-a-Service-*(SaaS-), On-Demand- sowie Cloud-Anbieter Rechnung tragen: Fachliche Dienste werden jetzt kostengünstig von Dienstleistern angeboten mit weiterem Optimierungspotenzial für IT-Leiter. Sie können nun unkritische Funktionalitäten auslagern und somit zu einer weiter optimierten Kostenstruktur für die IT beitragen. Für die Architektur von prozessgesteuerten Anwendungen bedeutet allerdings auch dieses Szenario einen erhöhten Bedarf nach Flexibilität. Welches der drei Szenarien auch immer zum Tragen kommt, ob Systemkonsolidierungen, Hinzufügen neuer Systeme oder die Verlagerung von Funktionalitäten: Für prozessgesteuerte Anwendungen gehören derartige Herausforderungen zum Alltag und müssen bei deren Konzeption integraler Bestandteil sein. Oder mit anderen Worten ausgedrückt: Trotz all dieser möglichen Veränderungen in der Systemlandschaft können und wollen Firmen nicht auf gewinnbringende Innovationsprozesse verzichten – und das sollen sie auch nicht: Eine geeignete Architektur gibt ihnen dazu die notwendige Rückendeckung und Sicherheit, dass sich die zwangsläufig kommenden nachgelagerten Änderungen an der Systemzusammenstellung nicht nachteilig auf die prozessgesteuerten Anwendungen auswirken! Die Innovationskraft des Unternehmens bleibt also erhalten.

Prozessgesteuerte Anwendungen für KMUs und ISVs

Auch wenn die bisherigen Ausführungen den Eindruck erwecken lassen, dass prozessgesteuerte Anwendungen ausschließlich für größere Unternehmen von Bedeutung sind, so ist dies keineswegs der Fall. Sicherlich spielt der Aspekt einer sich ständig ändernden Systemlandschaft für kleine und mittelständische Unternehmen (KMU) eine eher untergeordnete Rolle, obwohl SaaS sicherlich auch bei ihnen Spuren hinterlassen wird. Aber auch für sie gilt: Nur erfolgreiche, gegenüber der Konkurrenz differenzierende Prozesse garantieren das Überleben. Hier unterscheiden sich die Anforderungen im Vergleich zu Großunternehmen in keinster Weise, und damit gilt auch für sie hinsichtlich der Bedeutung von Änderungsfreundlichkeit als Bestandteil der Applikationsarchitektur das oben Gesagte.

Doch nicht nur innerhalb von Unternehmen spielt die neue Generation von prozessgesteuerten Anwendungen eine besondere Rolle. Auch Softwarehersteller (ISVs – Independent Software Vendors) können auf Basis von Composite Applications attraktive neue Märkte erschließen. In der Regel ist das Geschäftsmodell für ISVs darauf ausgerichtet, Lösungen zu entwickeln, die als Ergänzungen zu den Produkten bekannter Hersteller wie SAP, Oracle oder Microsoft zu sehen sind und die sich ausschließlich gegen diese Standardlösungen verbinden können (ich erinnere nur an das in der Einleitung erwähnte Beispiel einer Anwendung zur Verwaltung von Softwarelizenzen). Das Problem, das sich aufgrund dieses Modells ergibt, liegt auf der Hand: Zeigt ein Kunde Interesse an der fachlichen Lösung des Herstellers, so scheiterte das Zustandekommen eines Geschäfts ganz einfach an den fehlenden Voraussetzungen beim Kunden – die vorausgesetzte Standardanwendung ist dort eben nicht im Einsatz und ist auch kurzfristig nicht geplant. Viele potenzielle Kunden scheiden durch das Festlegen auf einen Hersteller von vornherein aus. Könnte also Software erstellt werden, die sich den anzutreffenden Systemlandschaften beim Kunden anpassen ließe, so würden sich gänzlich neue Perspektiven für ISVs ergeben. Und genau in diese Lücke stoßen prozessgesteuerte Anwendungen. Einmal erstellt, erlaubt deren Architektur die Anpassbarkeit an die beim Kunden angetroffene Landschaft, ohne dass die eigentliche fachliche Lösung davon betroffen ist. Dies kann nur dann gelingen, wenn wir es schaffen, eine weitestgehende Unabhängigkeit zwischen Fachanwendung und Systemlandschaft zu erzielen. Wie dies erreicht werden kann, werden wir in diesem Buch ausführlich besprechen.

Entwurfs- und Implementierungsansatz für prozessgesteuerte Anwendungen

Die zuvor genannten Eigenschaften, Probleme und Szenarien stellen den Ausgangspunkt dieses Buches dar. Es wurde nach Aufkommen der SOA-Idee und des damit verbundenen Hypes eine Vielzahl von Büchern und Artikeln zum SOA-Thema verfasst, die einzelne Aspekte teilweise auch sehr tiefgehend diskutieren.

Leider lösten sie die anstehenden Herausforderungen nicht wirklich, was aus meiner Sicht unter anderem auch eng mit dem eher Bottom-up geprägten Vorgehen bei der Schnittstellenermittlung sowie der starken Fokussierung auf Services und deren Wiederverwendbarkeit verknüpft ist. In diesem Buch wird dies durch eine Top-down-orientierte Methodik ersetzt und der Schwerpunkt auf die umzusetzenden Fachprozesse verlagert.

Was mir zudem immer fehlte, war eine detaillierte und dabei ganzheitliche Beschreibung der Architektur von fachlich orientierten (eben prozessgesteuerten) Anwendungen sowie deren konkrete Implementierung, die das volle Potenzial der serviceorientierten Ideen ausschöpft und dabei die SOA-Versprechen auch erfüllt. Die einzelnen für prozessgesteuerte Anwendungen relevanten Aspekte wie beispielsweise lose Kopplung, Transaktionshandling über Kompensation, Fehlertoleranz, Rolle von Service-Repositories, Anforderungen an Services, damit diese eine prozessgesteuerte Anwendungen optimal unterstützen, usw. müssen dafür zu einem harmonischen Ganzen zusammengefügt werden. In diesem Buch wird daher ein konkreter Architekturvorschlag erarbeitet, der die genannten Aspekte berücksichtigt und dabei auf eine konsequente Trennung von fachlichen und technischen Prozessen setzt. Dabei ist dieses Buch gleichzeitig ein Plädoyer für nachhaltige Architekturen, die sich im Alltag als robust bewähren und die rasche Anpassungen erlauben, und zwar so, wie die Geschäftsleitung im Idealfall ihr Geschäft abgewickelt sehen möchte. Die Zeitdifferenz zwischen der Vorgabe einer neuen (Geschäfts-)Strategie und deren Umsetzung wird dank einer durchdachten Architektur auf ein Minimum reduziert.

Auch bei der bereits angesprochenen Implementierung werde ich im Rahmen dieses Buches neue Wege beschreiten. Dabei soll die Verwendung und Eignung der Business Process Model and Notation (BPMN) in ihrer neuesten Version 2.0 für genau diesen Zweck erklärt werden, da gerade im Hinblick auf die Umsetzung sowohl kollaborativer, aber eben auch integrationszentrischer Prozesse interessante Neuerungen in den Standard eingeflossen sind. Diese neuen Möglichkeiten zu eruieren, in konkreten Beispielen anzuwenden sowie Chancen und Grenzen auszuloten wird ebenfalls Gegenstand der Betrachtungen sein.

Zusammengefasst basiert der in diesem Buch erarbeitete Entwurfs- und Implementierungsansatz letztendlich auf den folgenden drei Säulen:

1. Der Einsatz einer *Top-down-orientierten Methodologie* (ich benutze auch die Bezeichnung *Prozessgesteuerte Methodologie*) zur Bestimmung der wesentlichen Bestandteile/Artefakte der Lösung. Die Treiber der Lösung sind fraglos die fachlichen Prozesse – daher erklärt sich auch das Wort »*Prozessgesteuert*« im Titel des Buches. Ich werde im weiteren Verlauf des Buches derartig erstellte Lösungen daher auch mit *Prozessgesteuerte Anwendungen* bezeichnen, um diesem Aspekt Nachdruck zu verleihen.

2. Einer *nachhaltigen Architektur* (gleichzusetzen mit einer *prozessgesteuerten Architektur*) für prozessgesteuerte Anwendungen, die auf eine konsequente Trennung zwischen den eigentlichen Fachprozessen der Endanwendungsebene und den technischen Prozessen der sogenannten Servicevertrag-Implementierungsschicht setzt. Dabei wird gleichzeitig sowohl eine eindeutige Aufgabenzuordnung zu den beiden Schichten als auch deren Zusammenspiel festgelegt. Die Trennung über einen Servicevertrag bewirkt einen umfassenden Schutz der fachlichen Prozesse gegenüber den ständigen Veränderungen auf Systemebene.
3. Der *durchgängigen Verwendung der Business Process Model and Notation (BPMN)* sowohl zur Modellierung als auch zur Implementierung sämtlicher Prozessanteile auf fachlicher Anwendungsebene und der Ebene der Servicevertrag-Implementierungsschicht. Wie zu sehen sein wird, kann in der Servicevertrag-Implementierungsschicht zudem zwischen zustandsbehafteten und zustandslosen Prozessen unterschieden werden.

Buchaufbau

Damit unsere angestrebten Ziele Flexibilität, Skalierbarkeit und Fehlertoleranz auch wirklich erreicht werden können, wird es im weiteren Verlauf des Buches zunächst darum gehen, den Begriff der prozessgesteuerten Anwendung zu schärfen (Kapitel 2). Es geht um die typischen Eigenschaften derartiger Anwendungen und dabei gleichzeitig um die Abgrenzung zu anderen Anwendungstypen wie beispielsweise eng gekoppelten Einzelplatzprogrammen, projektspezifischen Lösungen oder reinen Integrationsapplikationen. In Form konkreter Beispiele werde ich die Ideen, die den Composite Applications zugrunde liegen, verdeutlichen.

Nachdem Sie auf diese Weise ein Verständnis von prozessgesteuerten Anwendungen entwickelt haben, wird sich Kapitel 3 mit der grundlegenden Architektur von Composite Applications sowie mit der Methode beschäftigen, wie die wesentlichen Bestandteile einer Composite (Prozesse, Benutzeroberflächen, Daten in Form von Geschäftsobjekten, die benötigten und bereitgestellten Dienste sowie der eigentliche Servicevertrag) ermittelt werden können. Kapitel 4 behandelt eine konkrete Implementierung einer prozessgesteuerten Anwendung. Eine wesentliche Rolle wird dabei, wie bereits erwähnt, die standardisierte grafische Notation zur Modellierung von Geschäftsprozessen BPMN spielen. Warum BPMN in diesem Zusammenhang so wichtig ist, wird Gegenstand einer ausführlichen Diskussion sein. Es wird aber nicht nur bei einer theoretischen Abhandlung bleiben. Vielmehr werde ich anhand eines kleinen Beispiels zeigen, dass die zu erwartenden Effekte auch tatsächlich erzielt werden können. Dafür wird anhand eines Einkaufsprozesses die Umsetzung in Software unter Verwendung von SAP NetWeaver Process Orchestration vorgestellt, die wesentlichen Implementierungsschritte werden kurz erklärt und abschließend die dabei gemachten Erfahrungen besprochen.

Prozessgesteuerte Anwendungen decken unternehmenskritische fachliche Prozessabläufe ab, und das trotz verteilter Architekturen und potenziell vielfältiger Fehlersituationen. Damit sie dabei dennoch eine ähnliche Robustheit und Zuverlässigkeit wie herkömmliche eng gekoppelte Anwendungen erreichen, reicht eine solide Architektur allein nicht aus. Begleitende technische Konzepte wie optimistisches Locking in den beteiligten Systemen, Idempotenz-Unterstützung bei den Services, Transaktionsverarbeitung über Kompensation, das Konzept der Fehlerbehandlung nahe bei den involvierten Backend-Systemen usw. helfen bei der Stabilisierung der Gesamtlösung. Wie diese begleitenden Maßnahmen dabei zusammenwirken und mit BPMN-Unterstützung modelliert werden können, ist Gegenstand von Kapitel 5.

Ebenfalls in Kapitel 5 werden typische BPMN-Prozessfragmente (Pattern) vorgestellt, die bei der Verbindung von prozessgesteuerten Anwendungen mit den Systemen der IT-Landschaften einsetzbar sind. In diesem Zusammenhang möchte ich auch auf einen Erweiterungsvorschlag für BPMN eingehen, um insbesondere Integrationsprozesse noch prägnanter und präziser modellieren zu können. Fortgesetzt wird Kapitel 5 mit einer Diskussion unterschiedlichster Ansätze zur Flexibilitätssteigerung sowohl der prozessgesteuerten Anwendung selbst als auch der Integration mit den Backend-Systemen. Regelwerke (Business Rules) und analytische Anwendungen werden hierbei eine wichtige Rolle übernehmen.

Abschließend werden im Ausblick des Kapitel 5 gegenwärtige Diskussionen rund um strukturierte und unstrukturierte Prozesse (Stichwort: Case Management) analysiert sowie mögliche Auswirkungen auf prozessgesteuerte Anwendungen erläutert.

Kapitel 6 fasst die Ergebnisse des Buches zusammen und behandelt zu guter Letzt das Thema, wie Composites als Produkt betrachtet durch Kunden modifikationsfrei erweitert werden können. Hier zeichnen sich erste Ansätze ab, auf die ich kurz eingehen werde.

Im Anhang finden Sie darüber hinaus neben einer Übersicht der BPMN-Elemente drei Exkursionen, in denen ich auf aktuelle Forschungsgebiete aufmerksam machen möchte, die in engem Zusammenhang mit prozessgesteuerten Anwendungen stehen: In Abschnitt A.4 geht es um unterschiedliche Ansätze zur Verwaltung von Services. In Abschnitt A.5 diskutiere ich Prozessvarianten, deren Komponentisierung sowie deren Konfiguration zu ausführbaren Lösungen. In Abschnitt A.6 schließlich beleuchte ich die Auswirkungen unterschiedlichster Aspekte auf die Ressourcenzuweisung für Prozesse, die bei deren Start, aber auch während der Prozessausführung zu berücksichtigen sind.

An dieser Stelle möchte ich nochmals betonen, dass es **nicht** Gegenstand dieses Buches ist, alle Facetten der Entwicklung serviceorientierter Software zu beleuchten. Auch eine umfassende Einführung in BPMN werden Sie in diesem Buch nicht finden. Dafür gibt es schon hervorragende Grundlagenliteratur wie das *BPMN 2.0*-Buch von Thomas Allweyer [Allweyer 2009a] bzw. das *Praxis-*

handbuch BPMN 2.0 von Jakob Freund und Bernd Rücker [Freund & Rücker 2012]. Bruce Silver beschreibt in seinem Buch *BPMN Method & Style* [Silver 2011] sehr schön, wie man BPMN auf fachlicher Seite strukturiert einsetzt und auf diese Weise selbst komplexe Prozesse in handhabbare Komponenten aufteilen und miteinander kommunizieren lassen kann. Wer sich darüber hinaus für die Ausführbarkeit von Prozessen jenseits von BPMN interessiert, dem sei das Buch *Geschäftsprozesse automatisieren mit BPEL* von van Lessen, Lübke, Nitzsche empfohlen [van Lessen, Lübke & Nitzsche 2011].

Der Schwerpunkt dieses Buches liegt eindeutig auf der Ausarbeitung einer konkreten Architektur für fachliche Anwendungen, die mehrere Geschäftsprozesse umfassen, basierend auf einer prozessgesteuerten Methodologie, wobei die daraus resultierenden Lösungen das von SOA versprochene Potenzial auch voll ausschöpfen, sowie deren Umsetzung auf Basis von BPMN. Natürlich können auch abweichend von der vorgeschlagenen Architektur Anwendungen entwickelt werden, die nur Teile des empfohlenen Ansatzes umsetzen. Architekten solcher Lösungen sind dann aber darüber informiert (und dazu möchte dieses Buch ebenfalls einen Beitrag leisten), welche möglichen Nachteile sich bei Abweichungen von der im Buch beschriebenen Lösung ergeben. Architekten können dementsprechend fundierte Entscheidungen treffen.

Themen, die sich hingegen weniger auf die Architektur einer prozessgesteuerten Anwendung auswirken, wie beispielsweise

- organisatorische Aspekte einer SOA
- Lebenszyklus von Services
- Versionsmanagement
- SOA und Sicherheit
- Enterprise Service Bus
- SOA Governance
- SOA-Projektmanagement
- Suchen und Finden von Services

sind bewusst ausgespart worden, da sie bereits als Einzelthemen in der SOA-Literatur ausgiebig diskutiert wurden. Ich möchte Sie in diesem Zusammenhang auf das (für mich beste) Buch in Sachen SOA mit dem Titel *SOA in der Praxis* von Nicolai Josuttis [Josuttis 2008] verweisen. Aber auch das Werk von Dirk Krafzig, Karl Banke und Dirk Slama mit dem Titel *Enterprise SOA – Service-Oriented Architecture Best Practices* [Krafzig, Banke & Slama 2004] kann ich Ihnen ans Herz legen.

Last but not least gehe ich zwar auf die Methodologie zur Ermittlung der wesentlichen Bestandteile des fachlichen Prozesses ein. Doch deckt dies bei Weitem keine allumfassende BPM-Methodik ab, wie dies beispielsweise äußerst detailliert in dem feinen Buch von Dirk Slama und Ralph Nelius über *Enterprise BPM* beschrieben ist [Slama und Nelius 2011].

1.2 SOA und prozessgesteuerte Anwendungen

Zwar teilen prozessgesteuerte Anwendungen und SOA viele Eigenschaften, Werte sowie Ziele, und Sie werden im Laufe des Buches auch immer wieder Gemeinsamkeiten mit SOA finden. Allerdings unterscheiden sich die beiden Ansätze in folgenden Punkten grundlegend: dem *Schwerpunkt*, der bei den beiden Ansätzen gelegt wird, und der Vorgehensweise, *wie* die Anwendungen und insbesondere der Servicevertrag entstehen. Da immer wieder die Frage gestellt wird, wo denn nun konkret die Unterschiede liegen, möchte ich diesen wichtigen Aspekt gleich zu Beginn des Buches klären. Allerdings muss ich eine Art »Ausschlussklausel« (Disclaimer) für den Vergleich mit SOA vorausschicken: Da es keine genaue Definition von SOA gibt, sondern SOA vielmehr als »ein Architektur-Paradigma (Denkmuster)« [Josuttis 2008, S. 30] betrachtet wird, kann ich prozessgesteuerte Anwendungen nur *mit meiner Interpretation von SOA* vergleichen (das gilt natürlich für sämtliche Vergleiche mit SOA im Verlauf dieses Buches). Und genau das macht es so schwierig: SOA ist ein bewegliches Ziel, und nahezu jeder versteht in dem einen oder anderen Punkt etwas ganz anderes darunter. Selbst die SOA-Experten haben Probleme, den Begriff SOA und dessen Charakterisierung in den Griff zu bekommen. Ganz offensichtlich wird dies beim SOA-Manifest [SOA-Manifest 2012], in dem »die weltweit führenden Autoren zu diesem Thema die Quintessenz von Service-Orientierung und von Service-orientierter Architektur (SOA) auf den Punkt bringen«, wie es in der Einleitung zu einer Broschüre von Nicolai Josuttis mit dem Titel *Das SOA-Manifest. Kontext, Inhalt, Erläuterung* [Josuttis 2010] heißt. Josuttis schreibt über die besondere Herausforderung bei der Suche nach geeigneten Formulierungen für das SOA-Manifest auf Seite 7 seiner Broschüre:

»Und wir konnten sogar hier und da unterschiedlicher Meinung bleiben, weil wir die gefundenen Formulierungen immer noch unterschiedlich interpretieren konnten.«

Genau diese Interpretationsmöglichkeit ist auch mein persönlicher Hauptkritikpunkt an SOA und wahrscheinlich einer der wichtigsten Unterschiede zu den in diesem Buch diskutierten prozessgesteuerten Anwendungen: Während SOA eben viele Interpretationsmöglichkeiten offen lässt (und diese Interpretationsmöglichkeiten führen leider immer wieder zu überflüssigen und zeitraubenden Grundsatzdiskussionen), möchte ich in diesem Buch so konkret und explizit wie möglich werden. Mir geht es um eine konkrete Anwendungskategorie (prozessgesteuerte Anwendung) mit einer konkreten Anwendungsarchitektur (prozessgesteuerte Architektur) zur Lösung eines konkreten Problems (prozessgesteuerte Anwendungsentwicklung auf Basis einer verteilten IT-Landschaft) verbunden mit einer konkreten Vorgehensweise (prozessgesteuerte Methodologie).

Da das SOA-Manifest nichtsdestotrotz noch immer zum Besten zählt, was im SOA-Umfeld hinsichtlich der Charakterisierung von SOA zu finden ist, beziehe ich mich daher in dem nun folgenden Vergleich darauf, ergänzt um die Erläuterungen von Josuttis aus der bereits genannten Broschüre [Josuttis 2010]. Dabei ist das SOA-Manifest selbst in drei Teile gegliedert: einer Präambel, einem Wertesystem sowie den SOA-Prinzipien, die sowohl die Ziele von SOA als auch das Wertesystem untermauern [Josuttis 2010, S. 20]. Die Präambel umfasst zum einen die Einführung der Begriffe *Service-Orientierung* und *SOA*, zum anderen die Festlegung der Ziele von SOA. Der Präambel schließen sich Werteaussagen an, in denen die Autoren bestimmte Werte vergleichen und priorisieren. Schließlich arbeiten 14 Prinzipien bestimmte Eigenschaften von SOA heraus, die die zuvor in der Präambel und dem Wertesystem herausgearbeiteten Aussagen stützen.

Gemeinsamkeiten

Lassen Sie mich zunächst die Gemeinsamkeiten herausarbeiten. Nun, diesen Teil kann ich relativ kurz halten, da ich mich mit fast allen Aussagen des SOA-Manifests anfreunden kann. So stimme ich beispielsweise vollkommen mit dem Ziel überein, das in der Präambel wie folgt zusammengefasst wird:

> Wir haben Service-Orientierung angewendet, um Organisationen zu helfen, kontinuierlich nachhaltigen Geschäftswert zu liefern mit höherer Agilität und Kosteneffizienz und im Einklang mit den sich ändernden fachlichen Bedürfnissen. [Josuttis 2010, S. 11]

Prägnanter kann man die Ziele wahrscheinlich nicht formulieren, und sie decken sich vollständig mit denen prozessgesteuerter Anwendungen. Lassen Sie mich zudem einige Beispiele aus dem Wertesystem herausgreifen, denen ich ebenfalls eine hohe Bedeutung beimesse und mit denen ich mich besonders gut identifizieren kann (ohne damit die Aussage treffen zu wollen, dass die nicht genannten Werte falsch oder unerheblich sind):

- Geschäftswert über technische Strategie [Josuttis 2010, S. 13]
- Strategische Ziele über projektspezifischen Nutzen [Josuttis 2010, S. 14]
- Flexibilität über Optimierung [Josuttis 2010, S. 17]
- Evolutionäre Vervollkommnung über Streben nach anfänglicher Perfektion [Josuttis 2010, S. 18]

Bei den Prinzipien gibt es ebenfalls große Übereinstimmungen. Auch hier möchte ich die für prozessgesteuerte Anwendungen wichtigsten Prinzipien kurz auflisten:

- Produkte und Standards alleine werden weder SOA liefern noch das Service-orientierte Paradigma umsetzen. [Josuttis 2010, S. 22]
- SOA kann mit unterschiedlichen Technologien und Standards umgesetzt werden. [Josuttis 2010, S. 22]
- Trenne die verschiedenen Aspekte eines Systems, die sich unterschiedlich häufig ändern. [Josuttis 2010, S. 26]
- Reduziere implizite Abhängigkeiten und publiziere alle externen Abhängigkeiten, um Robustheit zu fördern und die Auswirkungen von Veränderungen zu reduzieren. [Josuttis 2010, S. 26]

Diese Prinzipien können wir unverändert auch für prozessgesteuerte Anwendungen übernehmen. Sie müssen lediglich SOA durch PDA (process-driven application) ersetzen. Sie erkennen also schon die vielen Parallelen, die zwischen SOA und PDA gezogen werden können. Allerdings gibt es eben auch die im Folgenden diskutierten Unterschiede.

Unterschiede

Bereits zu Beginn dieses Abschnitts bin ich auf die beiden wesentlichen Punkte zu sprechen gekommen: die Schwerpunktsetzung und die Art und Weise, wie aufgrund der Schwerpunktsetzung die daraus resultierenden Anwendungen und der erforderliche Servicevertrag entstehen. Sehen wir uns dazu zunächst die SOA-Seite an. An mehreren Stellen sowohl des SOA-Manifests als auch der Broschüre von Nicolai Josuttis wird die Fokussierung auf Services klar zum Ausdruck gebracht. In der Präambel heißt es bereits:

> Service-orientierte Architektur (SOA) ist ein Architekturtyp, der aus der Anwendung von Service-Orientierung entsteht. [Josuttis 2010, S. 10]

Josuttis ergänzt in seinem Kommentar dazu:

»Zum anderen ist eine exakte Definition (von Service-Orientierung, Anm. des Autors) auch schwierig, wenn man einmal von der offensichtlichen Definition absieht, dass man sich auf Services bzw. den Dienstleistungsgedanken konzentriert.«

Diese Konzentration auf Services zieht sich wie ein roter Faden durch das Manifest. Weitere Beispiele: