Flexible Prozessapplikationen in Service-orientierten Architekturen – Ein Überblick

Stephan Buchwald[1], Thomas Bauer[2], Manfred Reichert[3]

[1]Group Research & Advanced Engineering, Daimler AG
stephan.buchwald@daimler.com

[2]Fakultät Informationsmanagement, Hochschule Neu-Ulm
thomas.bauer@hs-neu-ulm.de

[3]Institut für Datenbanken und Informationssysteme, Universität Ulm
manfred.reichert@uni-ulm.de

Zusammenfassung: Service-orientierte Architekturen (SOA) werden zunehmend in Unternehmen eingesetzt. Wichtige Ziele bilden die flexible IT-Unterstützung von Geschäftsprozessen, etwa deren rasche Anpassungsfähigkeit sowie die (Teil-) Automatisierung dieser Prozesse. Um die in der betrieblichen Praxis geforderte Flexibilität zu verwirklichen, sind jedoch eine Reihe von Maßnahmen vonnöten, die von der Dokumentation fachlicher Anforderungen über die Modellierung von Geschäftsprozessen bis hin zu dynamischen Service-Aufrufen reichen. Besonders wichtige Flexibilitätsmaßnahmen werden im vorliegenden Beitrag erörtert und in ein Rahmenwerk zur Erhöhung der Flexibilität in Service-orientierten Architekturen eingebettet.

1 Motivation

Service-orientierte Architekturen (SOA) und die sie unterstützenden Technologien spielen in Unternehmen eine immer bedeutsamere Rolle [9, 23, 29, 33, 52]. Insbesondere große Unternehmen versprechen sich durch die Einführung einer SOA mehr Flexibilität bei der Entwicklung und Inbetriebnahme von Unternehmenssoftware sowie sinkende Wartungs- und Änderungskosten für deren Betrieb. Mit Flexibilität meinen wir im vorliegenden Beitrag vor allem eine möglichst schnelle Umsetzung fachlicher Anforderungen und Prozesse in unterstützende IT-Applikationen. Einen wesentlichen Erfolgsfaktor bildet in diesem Zusammenhang die rasche Anpassungsfähigkeit der prozessunterstützende IT-Applikationen an fachliche Änderungen [33]. Nur wenn solche Änderungen schnell und kostengünstig umgesetzt werden können, ergibt sich ein Vorteil gegenüber Mitwettbewerbern.

Eine Umfrage unter IT-Führungskräften unseres Konzerns hat bestätigt, dass die skizzierte Art von Flexibilität eines der Hauptziele an eine SOA ist. Konkret ging aus dieser Umfrage hervor, dass sich der Bedarf nach Flexibilität primär auf die Anpassungsfähigkeit von IT-Applikationen sowie die Reaktionsfähigkeit auf neue bzw. geänderte fachliche und technische Anforderungen bezieht. Dieser Flexibilitätsbedarf wurde durch eine sich anschließende Literaturrecherche und Werkzeuganalyse bestätigt. Durch die Mitarbeit in verschiedenen IT-Projekten identifizierten wir weitere Kriterien für die Flexibilisierung einer SOA. In diesem Beitrag diskutieren wir darauf basierende Maßnahmen zur Flexibilisierung einer SOA. Wir zeigen einerseits, welche Maßnahmen zur Erhöhung der Flexibilität von prozessorientierten Anwendungen in einer SOA beitragen, andererseits diskutieren wir offene Fragestellungen für die bisher noch keine angemessenen Technologieangebote existieren. Der Fokus dieser Betrachtungen liegt bei IT-Themen, wohingegen organisatorische Aspekte weitgehend ausgeklammert werden.

Um IT-seitig möglichst flexibel auf fachliche Änderungen reagieren zu können, muss das Zusammenspiel der an einer SOA-Lösung beteiligten Rollen gut organisiert sein. Dazu gehört insbesondere die Verbesserung der Beziehungen zwischen Fach- und IT-Bereichen. Dies soll durch einen optimierten Softwareentwicklungsprozess, der fachliche Anforderungen mög-

lichst schnell, vollständig und unverfälscht in die IT-Implementierung überführt, erreicht werden.

In einer SOA existieren einerseits Anwendungsdienste (engl. Services), Prozesse, Geschäftsregeln, Organisationsmodelle und andererseits verschiedene Infrastrukturkomponenten, etwa Appikationsserver oder Prozess-Engines [9]. Bei Änderungen dieser Artefakte muss geeignet reagiert werden. Beispielsweise sollten vor Außerbetriebnahme eines Services die jeweiligen Service-Konsumenten rechtzeitig informiert werden. Vor allem prozessorientierte Applikationen – in der Folge Prozessapplikationen genannt – müssen flexibel gestaltet werden, so dass auf Änderungen, wie dem Abschalten eines Services mit möglichst geringem Anpassungsaufwand reagiert werden kann. Flexibilität beschreibt deshalb, neben einer möglichst schnellen und kostengünstigen Umsetzung fachlicher Anforderungen in Prozessapplikationen, auch die Reaktionsfähigkeit auf umgebungsinduzierte Änderungen, d.h. Änderungen der von einer Prozessapplikation genutzten Services oder Organisationsmodellen. Wir entwickeln im Projekt *Enhanced Process Management through Service Orientation (ENPROSO)* Gestaltungsrichtlinien für eine SOA, auf deren Grundlage sich Geschäftsprozesse und die zu unterstützenden Prozessapplikationen leichter realisieren lassen als mit heutigen Architekturen.

Dieser Beitrag betrachtet wichtige Maßnahmen bzw. Richtlinien zur Erhöhung der Flexibilität in einer SOA. Abschnitt 2 beschreibt Grundlagen die zum Verständnis dieses Beitrags erforderlich sind. Abschnitt 3 stellt ein Anwendungsszenario vor entlang dessen wir später verschiedene Maßnahmen zur Erhöhung der Flexibilität in einer SOA illustrieren. Anschließend wird in Abschnitt 4 der erwähnte Softwareentwicklungsprozess behandelt und phasenweise beschrieben. Im Vordergrund stehen die Flexibilität bei der Neuentwicklung von Prozessapplikationen sowie der Umgang mit späteren Änderungen [50]. Neben Änderungen, welche direkt die Prozessapplikation betreffen, beschreibt Abschnitt 5 den Umgang mit Änderungen in der SOA-Umgebung der Prozessapplikation (z.B. die Migration eines Services). Abschnitt 6 diskutiert verwandte Arbeiten, bevor der Beitrag mit einer Zusammenfassung und einem Ausblick schließt.

2 Grundlagen

Unter einer **Service-orientierten Architektur (SOA)** versteht man ein Architekturparadigma, das die Modellierung, Implementierung und Ausführen von Services (d.h. Anwendungsdiensten) unterstützt [9, 16, 23, 27]. Eine zentrale Idee besteht darin, die auf verschiedenen Plattformen ablaufenden bzw. von verschiedenen Herstellern stammenden Services einheitlich zu nutzen. Services sind Softwarebausteine, die Anwendungsfunktionalität kapseln und anderen Prozessapplikationen zur Nutzung anbieten. Durch ihre Nutzung entsteht eine lose Kopplung zwischen Anbieter und Nutzer. Das SOA-Paradigma erfordert neben Standards für Service-Integration, -Kommunikation und -Interoperabilität ein umfassendes Architekturmanagement (*Enterprise Architecture Management, EAM*), das geeignete Vorgehensweisen zur Gestaltung der SOA sowie Prozesse zur Umsetzung von Geschäfts- und IT-Strategien beinhaltet. Daneben werden Prozesse zur Kontrolle bzw. Planung der IT-Landschaft betrachtet [22]. Dabei werden Informationen über vorhandene Prozessapplikationen und IT-Systeme sowie deren Services und Beziehungen untereinander dokumentiert. Konkret wird festgehalten, welche IT-Systeme in den einzelnen Domänen existieren, welche Schnittstellen sie realisieren und wie lange sie in welcher Version im Betrieb sind. Dazu werden Gültigkeitszeiträume für Applikations- und Systemversionen dokumentiert.

Änderungen an Services können Auswirkungen auf die sie nutzenden Prozessapplikationen haben, etwa wenn ein Service nicht mehr korrekt ausgeführt werden kann. Aus diesem Grund sollten Prozesse für das Änderungsmanagement (Governance-Prozesse) definiert und opera-

tional unterstützt werden. Insbesondere sollten diese sicherstellen, dass Prozessapplikationen rechtzeitig angepasst werden.

Weitere Governance-Prozesse steuern den Entwurf und die Entwicklung von Services unter Beachtung von Prinzipien wie Kapselung, lose Kopplung, Standardisierung der Schnittstellen, Auffindbarkeit, Wiederverwendbarkeit und Autonomie [23]. Die Einführung neuer Services wird ebenfalls durch Governance-Prozesse geregelt. So müssen entsprechende Gremien prüfen, ob (fachliche) Services tatsächlich in der geplanten Form benötigt werden und – falls ja - den bestehenden Richtlinien entsprechen. Als Entscheidungsgrundlage für eine solche Prüfung muss der Service zunächst fachlich modelliert werden. Der erste Schritt dazu ist die Spezifikation der Funktionalität dieser fachlichen Services. In diesem Kontext werden z.B. Service-Name, Service-Beschreibung, Service-Eigenschaften (z.B. Quality of Service und der Service-Zustand), Service-Operationen und Service-Verantwortlichkeiten festgehalten [56, 61]. Service-Zustände beschreiben zum Beispiel ob ein Service fachlich spezifiziert, genehmigt, implementiert oder bereits freigegeben ist. Die einzelnen Zustände, die ein Service durchlaufen kann, werden im *Service-Lebenszyklus* (engl.: *Service-Lifecycle*) dokumentiert [29]. Entsprechend steuert und überwacht das *Service-Lifecycle-Management (SLM)* das Durchlaufen dieses Lebenszyklus [51] (vgl. Abbildung 1).

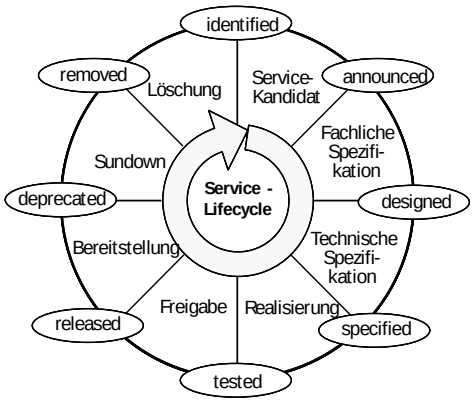

Abbildung 1 Vorschlag für einen Service-Lebenszyklus

Alle anfallenden Informationen zu einzelnen Services und der SOA werden in einem (logisch) zentralen *SOA-Repository* dokumentiert [13, 57]. Dies inkludiert Informationen, die zur Ausführung von Governance-Prozessen erforderlich sind, ebenso wie die für konkrete Service-Aufrufe zur Laufzeit erforderliche Daten (z.B. die Endpunktinformation eines Services).

Einen ebenfalls wichtigen Aspekt für ein umfassendes Business-IT-Alignment [15] bildet das **Geschäftsprozessmanagement** [58]. Eine wichtige Rolle nimmt dabei die Geschäftsprozessmodellierung [43] ein. Neben der Analyse und Optimierung der Geschäftsprozesse, dient sie vor allem der Dokumentation von Anforderungen seitens der Fachabteilungen und Prozessverantwortlichen. Da Letztere meist wenig oder gar keinen IT-Hintergrund besitzen, werden die Inhalte eines Geschäftsprozesses nicht formal beschrieben. Stattdessen werden in der Praxis meist einfache graphische Notationen oder textuelle bzw. tabellarische Beschreibungen verwendet, wie sie von Geschäftsprozess-Modellierungswerkzeugen angeboten werden. Beispielhaft seien an dieser Stelle erweiterte Ereignisgesteuerte Prozess-Ketten (eEPK) [55] und BPMN [37]. Die Verantwortung für die Erstellung des Geschäftsprozesses liegt meist bei den jeweiligen Fachbereichen, auch wenn die operative Umsetzung durch IT-Abteilungen oder externe Software-Häuser erfolgt. Für die Inhalte sind aber die Fachbereiche selbst verantwortlich, da nur sie das entsprechende Fachwissen haben. Bei der Modellierung werden primär der Prozessablauf (Kontroll- und Datenfluss) mit seinen Geschäftsprozessschritten beschrieben.

Zudem werden fachliche Services, deren Ein- und Ausgabedaten, das Organisationsmodell sowie zuständigen Bearbeiter festgelegt [43, 58].

Basierend auf Geschäftsprozessbeschreibungen können Prozessapplikationen implementiert werden. Dies erfolgt durch Software-Entwickler aus der IT-Abteilung. Diese treffen keine fachlich relevanten Entscheidungen bei der Erstellung des *ausführbaren Prozessmodells*, sondern übernehmen vielmehr die Struktur und Inhalte des Geschäftsprozesses als Grundlage für die Implementierung. Für das ausführbare Modell sind zahlreiche weitere zielplattformabhängige Informationen notwendig, wie konkrete Datenobjekte, Implementierung der von operational unterstützten Geschäftsprozessen benötigten Services, Benutzeroberflächen, Geschäftsregeln und zugrundeliegende Organisationsmodelle. Aufgrund der technischen Detaillierung kann diese Modellebene ausschließlich von Software-Entwicklern erstellt werden. Da solche in Fachbereichen typischerweise nicht verfügbar sind, liegt die Verantwortung hierfür beim IT-Bereich.

Eine Reduzierung des Implementierungsaufwand und eine bessere Anpassungsfähigkeit von Geschäftsprozessen bzw. Prozessapplikationen an notwendige Änderungen [55, 58, 62, 63, 64] lässt sich durch Verwendung von *Prozess-Management-Technologie* erreichen [43, 58]. Durch Trennung der verschiedenen Prozess-Aspekte, wie Kontroll- und Datenfluss, von der Implementierung der Anwendungslogik, entsteht vor allem eine besser strukturierte Prozessapplikation. Ein implementierter Prozess wird (in mehrfacher Instanz) von einer Prozess-Engine ausgeführt, weshalb die entsprechende technische Prozessbeschreibung (*ausführbares Prozessmodell*) vollständig, korrekt und ausführbar sein muss. Außerdem muss sie den Vorgaben des von der Prozess-Engine verwendeten Metamodells genügen [39, 44]. Prozess-Engines nehmen insbesondere in einer SOA eine zentrale Rolle ein, da sie die Orchestrierung von Services ermöglichen. So lassen sich durch Verwendung existierender Services leicht neue Prozessapplikationen realisieren. Moderne Prozess-Engines wie der IBM WebSphere Process Server [28] oder die Arista Flow BPM Suite [18, 19, 20] sind deshalb eng in die jeweilige Ausführungsinfrastruktur [9] für Prozessapplikationen integriert.

3 Anwendungsszenario

Wir skizzieren zunächst ein typisches Anwendungsszenario aus der Automobilindustrie, entlang dessen wir später Maßnahmen zur Erhöhung der Flexibilität in einer SOA diskutieren. Das in Abbildung 2 dargestellte Szenario zeigt einen abstrakten Geschäftsprozess für das Änderungsmanagement in der Fahrzeugentwicklung. Dieser stark vereinfacht dargestellte Ablauf stellt sicher, dass Änderungsvorhaben an Bauteilen vor ihrer eigentlichen Umsetzung bewertet, genehmigt und entsprechend dokumentiert werden.

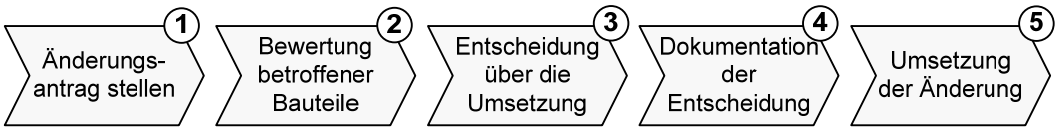

Abbildung 2 Geschäftsprozess zum Änderungsmanagement

Ein Änderungsvorhaben wird in Geschäftsprozessschritt (1) angelegt, indem die Änderung in Form eines Antrages beschrieben wird. Anschließend wird für die betroffenen Bauteile eine Stellungnahme eingeholt (2). Dabei werden insbesondere die technische Realisierbarkeit der Änderungen sowie die aus der Anpassung des jeweiligen Bauteils resultierenden Kosten bewertet. Abhängig von dieser Bewertung wird entschieden, ob das Änderungsvorhaben umgesetzt wird (3). Die Entscheidung über die Umsetzung des Änderungsvorhabens wird in Ge-

schäftsprozessschritt (4) dokumentiert, bevor die Änderung tatsächlich umgesetzt wird (5) (sofern sie genehmigt wurde).

4 Entwicklung von Prozessapplikationen

Dieser Abschnitt beschreibt, wie bei einer Entwicklung von Prozessapplikationen vorgegangen werden muss, um möglichst schnell und kostengünstig von fachlichen Anforderungen zur entsprechenden Prozessapplikation zu gelangen. Ausgehend von fachlichen Anforderungen werden Geschäftsprozesse modelliert, die anschließend durch eine IT-Implementierung realisiert werden. Wir identifizieren Maßnahmen, die dazu beitragen, die Flexibilität von Prozessapplikationen während der Entwicklung zu erhöhen.

Das in Abbildung 3 dargestellte Phasenmodell skizziert den zugehörigen Softwareentwicklungsprozess. Im Folgenden werden die einzelnen Phasen entlang des Anwendungsszenarios aus Abbildung 2 erörtert. Wir betrachten dazu sowohl Aspekte der Neuentwicklung von Prozessapplikationen als auch von deren späteren Änderungen [64].

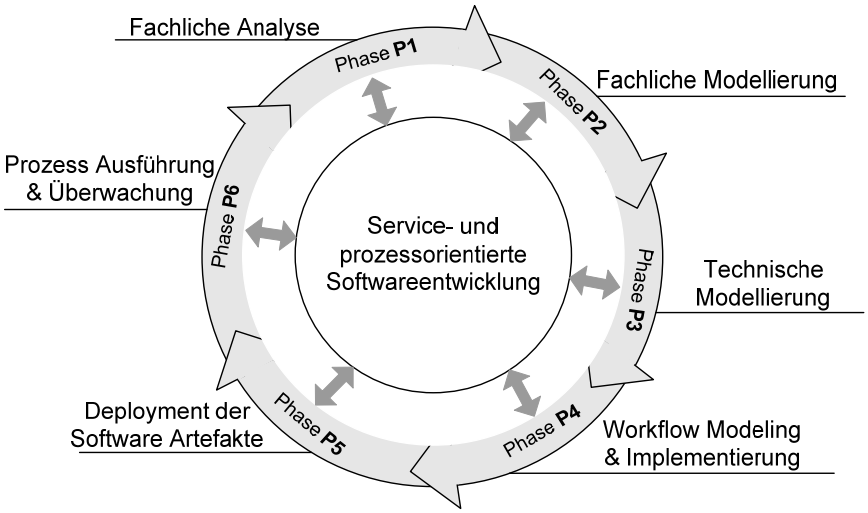

Abbildung 3 Prozess für Service- und Prozess-orientierte Softwareentwicklung

Im Folgenden werden die einzelnen Phasen entlang des Anwendungsszenarios aus Abbildung 2 erörtert. Wir betrachten dazu sowohl Aspekte der Neuentwicklung von Prozessapplikationen als auch von deren späteren Änderungen.

4.1. Beschreibung fachlicher Anforderungen (P1: *Fachl. Analyse*)

In der ersten Phase des Entwicklungsprozesses werden (fachliche) Anforderungen dokumentiert [53]. Diese beschreiben funktionale und nicht-funktionale Vorgaben an die zu entwickelnde Prozessapplikation. Zur Erfassung und Beschreibung dieser Anforderungen werden Software-Werkzeuge (z.B. [65]) eingesetzt, mittels denen ein Anforderungsingenieur (engl.: Requirements Engineer) die Anforderungen (meist textuell) dokumentiert und sie mit den Fachbereichen abstimmt. Für das in Abschnitt 3 beschriebene Anwendungsszenario etwa könnte sich der in Abbildung 4 dargestellte Anforderungskatalog ergeben. Dieser zeigt allerdings nur ausgewählte (funktionale) Anforderungen und keine vollständige Spezifikation aller Anforderungen.

Req.1)	Änderungsvorhaben an Bauteilen müssen vor der Genehmigung bewertet werden
Req.2)	Änderungsvorhaben an Bauteilen müssen genehmigt werden
Req.3)	Änderungsvorhaben werden in Form eines Änderungsantrages angelegt
Req.4)	Bauteildaten müssen aus dem Produktdaten-Management-System importiert werden
Req.5)	Prüfung auf technische Realisierbarkeit und Stellungnahme durch den entsprechenden Konstrukteur
Req.6)	Baureihenverantwortlicher entscheidet über die Umsetzung eines Änderungsvorhabens
Req.7)	Die Entscheidung über die Umsetzung muss in davon betroffenen Systemen dokumentiert werden
Req.8)	Möglichkeit zur Durchführung von Einzelbewertungen für eine beliebige Anzahl von Bauteilen

Abbildung 4 Anforderungskatalog für den Änderungsprozess (Ausschnitt)

Die erste von uns identifizierte Maßnahme zur Erhöhung der Flexibilität in einer SOA stellt sich folgendermaßen dar.

Maßnahme 1: Dokumentation der Beziehungen zwischen Anforderung und Implementierung

Neben den eigentlichen Anforderungen sollte dokumentiert werden, wie diese konkret implementiert werden [26]. Dazu müssen die Beziehungen zwischen den fachlichen Anforderungen und den Umsetzungsobjekten (z.B. Geschäftsprozessschritte, Datenobjekte oder Bearbeiter) im zugehörigen Geschäftsprozess explizit dokumentiert und konsistent verwaltet werden. Nur dann ist nachvollziehbar, welche Geschäftsprozesse und Umsetzungsobjekte von zukünftigen Änderungen fachlicher Anforderungen betroffen sein werden. Das folgende Beispiel zeigt, wie die Beziehung zwischen Anforderung Req. 3 aus Abbildung 4 und dem entsprechenden Geschäftsprozess aus Abbildung 6 dokumentiert werden kann.

Beispiel 1 (Dokumentation von Beziehungen)

Anforderung Req. 3 fordert, dass für jedes Änderungsvorhaben ein Änderungsantrag zu erstellen ist. Dies wird dadurch realisiert, dass ein Geschäftsprozessschritt zur Eingabe eines Änderungsvorhabens im Änderungsprozess modelliert wird (vgl. Abbildung 6). Anschließend speichert ein Datenobjekt die Informationen über das Änderungsvorhaben. Abbildung 5 zeigt die Beziehung zwischen Anforderung Req. 3 und den zugehörigen Umsetzungsobjekten im Änderungsprozess.

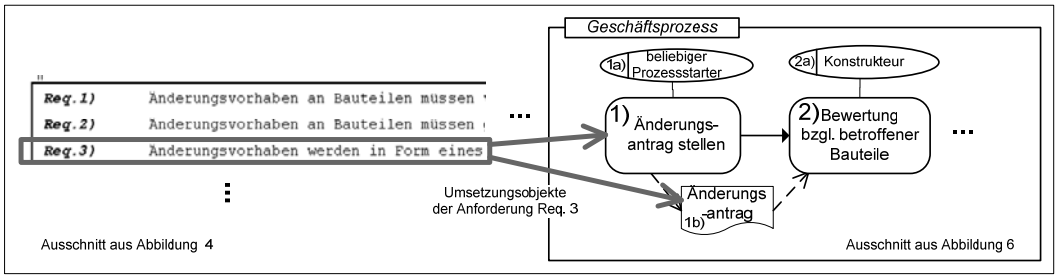

Abbildung 5 Beziehung zwischen Anforderung und Umsetzungsobjekten

Durch die Dokumentation solcher Beziehungen lassen sich Änderungen an fachlichen Anforderungen schneller umsetzen. Zudem wird sichergestellt, dass alle fachlichen Anforderungen tatsächlich durch den entsprechenden Geschäftsprozess realisiert werden.

4.2. Definition von Geschäftsprozessen (P2: *Fachliche Modellierung*)

Die Modellierung von Geschäftsprozessen erfolgt typischerweise durch die Fachabteilungen bzw. Fachmodellierer. Diese bilden fachliche Anforderungen des Unternehmens auf Geschäftsprozessmodelle ab. Dazu werden Geschäftsprozessschritte und deren Ablaufreihenfol-

ge sowie involvierte IT-Systeme modelliert und Verantwortlichkeiten festgelegt. Aufgrund der fehlenden bzw. mangelhaften IT-Kompetenz der Fachmodellierer werden einfache graphische Notationen verwendet. In der Praxis verbreitet sind z.B. *erweiterte Ereignisgesteuerte Prozess-Ketten (eEPK)* [55] und die *Business Process Model and Notation (BPMN)* [37]. Beispiel 2 beschreibt den in Abbildung 6 modellierten Änderungsprozess.

Beispiel 2 (Geschäftsprozessmodellierung)

Im Geschäftsprozessschritt (1) wird ein Änderungsvorhaben angelegt, indem die Änderung in Form eines Änderungsantrages (1b) beschrieben wird. Der Geschäftsprozessschritt (1) und das Datenobjekt Änderungsantrag (1b) realisieren zusammen die Anforderung Req. 3 (vgl. Abbildung 5), d.h. Req. 3 wird durch zwei Umsetzungsobjekte realisiert. Der Änderungsantrag enthält, neben der eigentlichen Änderungsbeschreibung, mögliche Maßnahmen zur Umsetzung der Änderung, eine Liste betroffener Bauteile sowie eine grobe Abschätzung resultierender Kosten. Ausgehend von dieser Information werden im Geschäftsprozessschritt (2) für alle vom Änderungsvorhaben betroffenen Bauteile die entsprechenden Bauteildaten aus dem Produktdaten-Management-System (2b) importiert (vgl. Anforderung Req. 4) und jeweils durch einen Konstrukteur (2a) bewertet (vgl. Anforderung Req. 1 und Req. 5). Anschließend entscheidet in (3) ein Baureihenverantwortlicher über die Umsetzung des Änderungsvorhabens (Req. 6). Diese Entscheidung muss in allen betroffenen Systemen (4) dokumentiert werden (Req. 7), bevor die Änderung tatsächlich umgesetzt wird (5).

Die vorangehend spezifizierten Anforderungen (vgl. Abbildung 4) führen zu dem in Abbildung 6 dargestellten Geschäftsprozess:

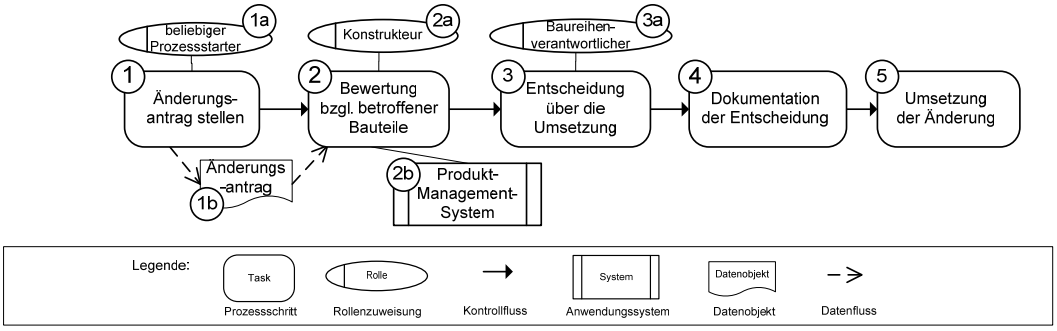

Abbildung 6 Modellierter Änderungsprozess in ARIS als eEPK

In der frühen Phase P2 des Softwareentwicklungsprozesses sind meist noch nicht alle Aspekte im Detail bekannt oder sollen aus Komplexitätsgründen noch nicht erfasst werden. Deshalb wird das Geschäftsprozessmodell an verschiedenen Stellen bewusst vage und offen gehalten. Dies betrifft neben der Struktur von Geschäftsprozessmodellen (d.h. Geschäftsprozessschritten und den Kontrollfluss zwischen ihnen) noch weitere Aspekte wie Datenstrukturen oder Bearbeiterzuordnungen [44, 47, 48, 58]. Dennoch ist eine frühe Modellierung dieser Information sinnvoll, um diese in einer späteren Phase des Entwicklungsprozesses (vgl. Abbildung 3) verwenden zu können, ohne im Fachbereich nachfragen zu müssen.

Maßnahme 2: Frühe Modellierung von Informationen durch Frontloading

Obwohl sich bei der fachlichen Modellierung nicht immer alle Aspekte detailliert beschrieben lassen, ist eine möglichst frühe Dokumentation von Informationen über die spätere Implementierung, prinzipiell anzustreben (*Frontloading*). Als Frontloading bezeichnen wir in diesem Zusammenhang die frühzeitige Modellierung von Information, die während der Geschäftsprozessmodellierung möglichst vollständig beschrieben werden soll. Somit lassen sich Aspek-

te einer späteren IT-Implementierung früh dokumentieren und mit den Verantwortlichen aus dem Fachbereich abstimmen. Dazu gehören Informationen, die erstmalig bei der Geschäftsprozessmodellierung dokumentiert werden. So sollte betrachtet werden, welche Personen einen bestimmten Schritt des Geschäftsprozesses bearbeiten sollen. Hierbei genügt es nicht, lediglich eine Rolle zuzuordnen. Stattdessen müssen die *Bearbeiterzuordnungen* auf fachlicher Ebene genau spezifiziert werden. Dies muss ausreichend detailliert erfolgen und zudem vollständig und formal sein. Dennoch sollte diese Aufgabe durch Fachanwender bewerkstelligt werden können. Eine Bearbeiterzuordnung kann folgendermaßen lauten:

Beispiel 3 (Frontloading)
Die Kreditgenehmigung soll durch eine Person aus der Filiale bearbeitet werden, in welcher auch der Kreditantrag entgegengenommen wurde. Diese Person darf nicht die Person sein, die bereits die Vorprüfung des Antrags durchgeführt hat.

Solche Bearbeiterzuordnungen werden meist auf fachlicher Ebene nicht in Geschäftsprozessen dokumentiert. Lediglich begleitende Textdokumente beschreiben ggf. ein solches Verhalten abstrakt. Damit Bearbeiterzuordnungen, wie etwa aus Beispiel 3, tatsächlich in die Implementierung übernommen werden können, müssen diese hinreichend detailliert beschrieben werden. Ein Beispiel dafür kann folgendermaßen aussehen: *(Rolle = Kreditprüfer)* $\wedge$ *(Filiale = $Antragsstellung.Filiale$)* $\wedge$ *(Person* $\neq$ *$Vorprüfung.Person$)*. Die dokumentierte Bearbeiterzuordnung ist aufgrund des formalen Karakters zur weiteren Verarbeitung im IT-Bereich gut geeignet. Fachmodellierer können jedoch eine solche Beschreibung meist nicht erstellen, da ihnen der notwendige IT-Hintergrund fehlt. Deshalb ist eine intelligente Lösung notwendig, welche einen Modellierer bei der Erstellung solcher Bearbeiterzuordnungen hinreichend unterstützt. Wird ein Frontloading durch Fachmodellierer in einer frühen Phase des Softwareentwicklungsprozesses realisiert, lässt sich eine zeitaufwendige Abstimmung mit verantwortlichen Fachbereichen bei der Implementierung der technischen Prozesse vermeiden. Die Schwierigkeit ist nun, auf fachlicher Ebene eine Beschreibungsform und Methodik zur Dokumentation von Bearbeiterzuordnungen zu finden, die ausreichend vollständig genug ist, um zur Weiterverarbeitung in der Implementierung zu dienen.

Nicht nur die Dokumentation von Bearbeiterzuordnungen, sondern auch eine detaillierte *Informations- und Datenmodellierung* kann bereits auf der fachlichen Ebene (P2) sinnvoll sein, um später bei der Implementierung der Prozessapplikation in einem Prozess-Management-System bestimmte Information (teil)automatisiert ableiten zu können (d.h. ohne zusätzliche Analyse-Phase im Fachbereich). Auch explizite Informationen von Fachanwendern, wie etwa zu möglichen *Eskalationen* oder die Funktionalität von *(Benutzer-) Masken* (insb. interessant für die Implementierung in Phase P5) sollen frühzeitig dokumentiert werden können. Zudem sind Informationen über *Ausnahmesituationen* im Geschäftsprozess notwendig, um geeignete Reaktionsmöglichkeiten bereits bei der Modellierung festzulegen, etwa bei fehlgeschlagenen Service-Aufrufen oder einer unzureichenden Datenqualität nach einer Benutzereingabe.

Ein weiterer Frontloading-Aspekt betrifft die *Markierung potentieller Flexibilitätsstellen* im Geschäftsprozessmodell. D.h. es sollten bereits bei der Modellierung festgelegt werden, welche Bereiche des Geschäftsprozesses in späteren Phasen möglichst flexibel zu implementieren sind. Entsprechende Kennzeichnungen können an unterschiedlichen Objekten, etwa Geschäftsprozessschritten, Verzweigungen, Service-Aufrufen oder Bearbeiterzuordnungen erfolgen. Beispiel 4 beschreibt die Kennzeichnung von Flexibilitätsstellen für einen Geschäftsprozessschritt.

Beispiel 4 (Frontloading)
Anforderung Req. 8 an unserem Beispielprozess (vgl. Abbildung 4) fordert, dass Einzelbewertungen für eine beliebige Anzahl von Bauteilen durchgeführt werden können. Eine solche Information kann durch eine explizite Kennzeichnung (bspw. als textuelle Beschreibung) am Geschäftsprozessschritt (2) dokumentiert werden (vgl. Abbildung 6). Weitere Beispiele für Kennzeichnungen im Geschäftsprozessmodell sind Markierungen an Verzweigungsbedingungen für häufig anzupassende Schwellwerte oder Service-Zuweisungen für das späte Binden von Service-Instanzen (Late-Binding, [17, 35]).

Grundlegendes Ziel des Frontloading ist es, zusätzliche Abstimmungen mit den Fachbereichen (in späten Phasen) zu vermeiden. Dadurch sollen Entwicklungszeit und -kosten reduziert sowie die Qualität der Prozessapplikation erhöht werden. Hierfür ist eine Methodik zu entwickeln, die die frühzeitige Modellierung von Informationen ermöglicht, d.h. Informationen für spätere Phasen bereitstellt. Diese Methodik muss zugleich auch für Fachmodellierer und -anwender verständlich sein [21, 54].

Maßnahme 3: Berücksichtigung von Informationen durch Look-ahead

Geschäftsprozesse werden oft ohne Rücksichtnahme auf die bereits existierende IT-Umgebung modelliert. Informationen über vorhandene Services, Datenobjekte oder Organisationsmodelle etwa werden deshalb nicht immer vollständig dokumentiert. Um einen möglichst hohen Grad an Wiederverwendung zu erreichen, ist es daher wichtig, den Geschäftsprozess so zu gestalten, dass existierende Services auch tatsächlich wiederverwendet werden können. Meist wird dies jedoch nicht berücksichtigt, da Fachmodellierer entweder keine Information über deren Existenz vorliegt oder diese Information nur schwer beschaffbar ist. Oft wird auch nicht explizit nach Services gesucht bzw. bewusst auf deren Verwendung verzichtet, insbesondere wenn die bereitgestellte Funktionalität nicht exakt passt. Dadurch wird wiederum die Modellierung der Geschäftsprozesse aufwendiger, da im Geschäftsprozess verwendete Funktionalität erst implementiert werden muss.

Es kann auch sinnvoll sein, einen Service zu verwenden, der nicht exakt die gewünschte Funktionalität realisiert. Dadurch muss ggf. der Geschäftsprozess geringfügig umgestaltet werden: Wenn hierdurch jedoch ein existierender Service wiederverwendet werden kann, spart dies Zeit und Aufwand bei der Implementierung. Um eine solche Wiederverwendung zu fördern, müssen im Unternehmen Governance-Prozesse etabliert werden, welche die Wiederverwendung existierender Services zuverlässig steuern und teilweise auch „erzwingen“. Zusätzlich müssen diese Services für Fachmodellierer zugänglich gemacht werden.

Neben der Berücksichtigung von Services (*Service-Look-ahead*) sind weitere Aspekte relevant. So sollten Rollen, die im Geschäftsprozessmodell verwendet werden, auch tatsächlich im Organisationsmodell des Unternehmens existieren. Bereits im Unternehmen existierende Datenobjekte, die als Eingabe oder Ausgabe von Geschäftsprozessschritten und Services verwendet werden, sollten ebenfalls im Geschäftsprozessmodell verwendet werden, etwa um Datenredundanz zu vermeiden. Weitere Frontloading- und Look-ahead-Aspekte werden in [21] diskutiert. Beispiel 5 beschreibt anhand unseres Szenarios wie ein Service-Look-ahead aussehen kann.

Beispiel 5 (Service-Look-ahead)

Für Geschäftsprozessschritt (2) unseres Beispiels (vgl. Abbildung 6) sucht der Fachmodellierer im SOA-Repository [9, 13, 57] nach einem bereits dokumentierten fachlichen Service der Bauteilinformationen bereitstellt. Der Fachmodellierer hinterlegt den fachlichen Service am Geschäftsprozessschritt (2), etwa in Form eines Textdokuments, einer Referenz auf das fachliche Service-Modell oder einer externen Referenz auf ein SOA-Repository-Objekt.

4.3. IT-orientierte Sichtweise auf Geschäftsprozesse (P3: *Technische Modellierung*)

Da Fachanwender und IT-Implementierer typischerweise einen unterschiedlichen fachlichen Hintergrund haben, kommt es oft zu Unstimmigkeiten zwischen Fach- und IT-Bereich. Informationen auf fachlicher Ebene werden anders dokumentiert und verstanden als Informationen auf technischer Ebene. Fachanwender verwenden meist andere Modelltypen für die Prozessdokumentation als IT-Implementierer: Häufig modellieren Fachanwender ihre Geschäftsprozesse in Geschäftsprozessmanagement-Werkzeugen, etwa in ARIS [55] als erweiterte EPK oder Signavio mittels BPMN [37], wohingegen IT-Implementierer CASE-Tools mit UML-Unterstützung verwenden. Es ist kein durchgängiges Vorgehen zur Abbildung fachlicher Modelle (Phase P2) auf deren technische Implementierung (Phase P4) etabliert.

All dies erschwert die Zusammenarbeit dieser beiden Rollen und Bereiche. Um die bestehende fachliche Distanz zu schließen (Business-IT-Alignment, [15]), sollte ein zusätzliches Modell (Systemmodell, Phase 3) eingeführt werden, das die Adaption fachlicher Modelle auf eine IT-Realisierung erleichtert. Ziel dabei ist es, Informationen nachvollziehbar und verständlich (erklärbar) für den Fachbereich sowie formal und vollständig genug für den IT-Bereich zu spezifizieren. Die Verantwortung für die Erstellung des Prozesses auf Systemmodell-Ebene (Systemprozess) liegt dann beim IT-Bereich. Die zugehörigen Inhalte sind dieselben wie im Geschäftsprozessmodell. Allerdings müssen diese ausreichend detailliert, vollständig erfasst und formal beschrieben sein, um die Basis einer plattformunabhängigen IT-Spezifikation bilden zu können. Das bedeutet, dass textuelle Beschreibungen und offene Aspekte aus dem Fachmodell ersetzt worden sind.

Maßnahme 4: Nachvollziehbarkeit zwischen Modellierungsebenen

Der Systemprozess detailliert diejenigen Aspekte aus dem Geschäftsprozessmodell, die technisch unterstützt werden sollen [10, 11, 12]. Das sind beispielsweise einzelne Geschäftsprozessschritte des fachlichen Geschäftsprozessmodells, die *direkt* in Schritte des Systemprozesses überführt werden:

Beispiel 6 (Nachvollziehbarkeit)

Die Benutzerinteraktion zum Stellen eines Änderungsantrages (Geschäftsprozessschritt (1) in Abbildung 7) kann beispielsweise als sog. *Human Task* [4] mit dem zu den IT-Vorgaben konformen Namen HT_*request_change* realisiert werden.

Geschäftsprozessschritte können zudem in mehrere *sequenziell* oder *parallel* ausgeführte (technische) Aktivitäten *aufgespalten* und ggf. durch zusätzliche Ablauflogik detailliert werden:

Beispiel 7 (Nachvollziehbarkeit)

Geschäftsprozessschritt (4) in Abbildung 7 ist ein Beispiel für eine Aufspaltung eines Prozessschrittes beim Übergang von einem fachlichen Geschäftsprozessmodell in das technische Modell: Die Dokumentation der Entscheidung besteht einerseits aus der Dokumentation im PDM-System (*service_doc_change_PDM*) und andererseits aus einer Dokumentation im Stücklistensystem (*service_doc_change_parts_list*) sowie der Ablauflogik (AND-Join und -Split) zur Parallelisierung dieser beider Prozessschritte.

Ferner wird dokumentiert, welche technisch nicht unterstützten Geschäftsprozessschritte im Systemprozess *weggelassen* werden, etwa wenn keine IT-Umsetzung gewünscht ist (vgl. Geschäftsprozessschritt (5) in Abbildung 7).
Die Durchführung von Einzelbewertungen für Bauteile liefert ein Beispiel für die Umsetzung der Anforderung Req. 8 aus Abbildung 4 im Systemprozess:

Beispiel 8 (Nachvollziehbarkeit)

Durch Einfügen einer Multi-Instanz Aktivität [7] im Systemprozess, wird dafür gesorgt, dass für alle von einer Änderung betroffenen Bauteile eine Einzelbewertung durch den entsprechenden Teileverantwortlichen erfolgt.

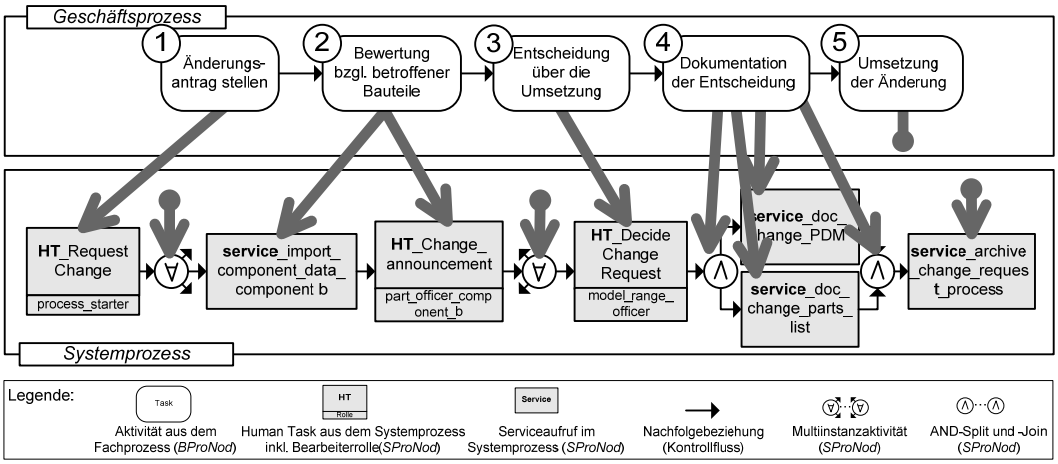

Abbildung 7 Überführung von Geschäftsprozessschritten in Schritte des Systemprozesses

Aufgrund dieser mitunter komplexen Umstrukturierungen beim Übergang vom Geschäfts- zum Systemprozess, ist es für Fachmodellierer keineswegs trivial, die richtigen Objekte aus dem Geschäftsprozess mit denen im Systemprozess in Beziehung zu setzen. Herausfordernd ist zudem die Art der Dokumentation dieser Beziehungen, die zum einen Fachmodellierer nicht überfordern darf und zum anderen lesbar und verständlich für Endanwender sein muss. Ein weiteres Problem entsteht dadurch, dass die Metamodelle der heutzutage eingesetzten Werkzeuge eine solche Dokumentation nicht explizit unterstützen.

Bezogen auf diese Problemstellung ermöglicht unser ENPROSO-Ansatz die Nachvollziehbarkeit der Beziehungen zwischen Geschäftsprozessschritten und deren IT-Implementierung im Systemprozess. Ebenso wird eine Zuordbarkeit in umgekehrter Richtung unterstützt: So ist es für die robuste Ausführung einer Prozessapplikation wichtig, die von Umgebungsänderungen betroffenen Prozesse und Aktivitäten identifizieren zu können, etwa

wenn ein laufender Service abgeschaltet werden soll. Nur so wird es möglich, entsprechend schnell auf anstehende Änderungen zu reagieren.

Neben der Detaillierung von Geschäftsprozessen, ist eine zu entwickelnde Prozessapplikation so zu gestalten, dass spätere Änderungen möglichst wenig Aufwand verursachen. So können bereits in dieser frühen Phase (vgl. Phase P3 in Abbildung 3) die mittels Frontloading beschriebenen Informationen berücksichtigt werden. Für sich häufig ändernde oder komplexe Verzweigungsbedingungen in Geschäftsprozessen sollen zudem Geschäftsregeln so definiert werden, dass sie unabhängig vom Geschäftsprozess verändert werden können [9]. Im Systemprozess werden dadurch bestimmte Verzweigungsentscheidungen durch Geschäftsregeln von der Geschäftsprozesslogik entkoppelt.

Maßnahme 5: Analyse von Inkonsistenzen zwischen den Modellebenen

Spätere Änderungen fachlicher Anforderungen, etwa infolge einer Geschäftsprozessoptimierung oder einer erforderlichen Prozessanpassung aufgrund von Änderungen gesetzlicher Rahmenbedingungen, wirken sich auf die Geschäftsprozessmodelle (Phase P2), die zugehörigen Systemprozesse (P3) sowie die ausführbaren Modelle (P4) aus. Diese müssen bei Änderungen entsprechend angepasst werden. Wir erläutern dies wieder anhand unseres Beispiels:

Beispiel 9 (Aus Änderung resultierende fachliche Anforderung)

Änderungsvorhaben betreffen häufig viele Bauteile. Da die Auswirkung eines solchen Vorhabens auf jedes einzelne Bauteil zu bewerten ist (vgl. (2) aus Abbildung 7), kann das Durchlaufen einer Instanz des Änderungsprozesses einen hohen Aufwand verursachen. Ein „Filter“ vor der eigentlichen Bewertung der Bauteile kann die Anzahl der Anträge jedoch reduzieren: Der Vorgesetzte des Antragstellers soll die Erfolgsaussichten des Änderungsvorhabens bewerten und bei Bedarf den Änderungsprozess vorzeitig stoppen. Als Ergebnis dieser Analyse entsteht eine neue fachliche Anforderung (Req. 9) an den Änderungsprozess:
Anforderung Req. 9 beschreibt die Notwendigkeit eines Vorfilters, bevor die Bewertung betroffener Bauteile durchgeführt wird. Die Verantwortung dafür liegt beim Vorgesetzten des Antragstellers.

Im Folgenden beschreiben wir, wie sich die neu hinzugekommene Anforderung Req. 9 in unserem Änderungsprozess umsetzen lässt. Abbildung 8 zeigt das resultierende Ergebnis.

Schritt A: Anforderung Req. 9 wird durch einen zusätzlichen Geschäftsprozessschritt (6) im Änderungsprozess sowie einer entsprechenden Abbruchbedingung (6a) realisiert. Dadurch entsteht eine neue Version des Geschäftsprozessmodells.

Schritt B: Aufgrund der nachvollziehbar dokumentierten Beziehungen zwischen Geschäftsprozess und Systemprozess (vgl. Maßnahme 4), können Inkonsistenzen erkannt werden. Dabei werden die Objekte im Fachmodell identifiziert, für die keine Objekte im Systemprozess existieren, d.h. für die keine Beziehung dokumentiert ist (vgl. (6) und 6a)).

Schritt C: Identifizierte Inkonsistenzen lassen sich dadurch beheben, dass fehlende Objekte im Systemprozess erzeugt und die Beziehungen zu korrespondierenden Objekten im Geschäftsprozess dokumentiert werden.

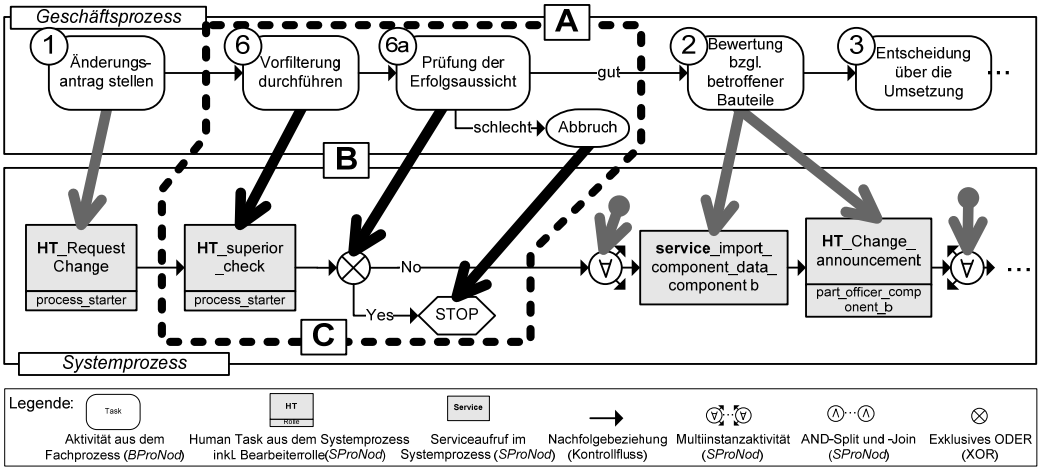

Abbildung 8 Änderungsprozess nach Optimierung

Die Speicherung der Beziehungen zwischen Geschäftsprozessschritten eines Geschäftsprozessmodells auf der einen Seite und des zugehörigen Systemmodells auf der anderen Seite, erhöht die Flexibilität bei späteren Anpassungen. D.h. durch Änderungen verursachte Abhängigkeitsverletzungen können leichter erkannt und schneller behoben werden, da die Beziehungen zwischen Objekten des Geschäftsprozesses und des Systemprozesses explizit ist. So können Änderungen fachlicher Anforderungen oder Geschäftsprozesse schneller in die IT-Implementierung überführt werden. Andererseits lassen sich die Effekte von Änderungen der Umgebung der Prozessapplikation (vgl. Abschnitt 5), etwa die Abschaltung eines technischen Services, bis zur verantwortlichen Person im Fachbereich zurückverfolgen. Diese kann über die Änderung informiert werden und aus fachlicher Sicht über eine geeignete Reaktion entscheiden.

4.4. IT-Implementierung (P4: Workflow Modellierung und Implementierung)

Die Übernahme des Systemprozesses in ein *ausführbares Prozessmodell* (Phase 4) soll sich möglichst einfach gestalten, da diese Aufgabe häufig von externen Dienstleistern ohne tiefgehende Kenntnis der eigentlichen Geschäftsprozesse durchgeführt wird. Deshalb sollten diejenigen Informationen im Systemmodell vorliegen, die für eine Implementierung durch einen IT-Dienstleister notwendig sind. Sobald das Systemmodell als Spezifikation das Unternehmen verlässt, ist eine Abstimmung von Änderungen mit den Fachbereichen nicht mehr oder nur noch sehr eingeschränkt bzw. mit hohem Aufwand möglich.

Die Überführung der Information aus dem Systemprozess in ein ausführbares Prozessmodell kann meist 1:1 erfolgen. Dabei werden alle im Systemmodell beschriebenen Objekte übernommen und entsprechend den Möglichkeiten der Zielplattform implementiert. Dies betrifft den Systemprozess (inklusive Aktivitäten), die dort verwendeten technischen Services, Benutzermasken und Geschäftsregeln. Der Systemprozess wird aus dem generischen Format in ein zielplattformabhängiges Format (z.B. WS-BPEL [1]) übertragen. Neben Benutzerinteraktionen (Human Tasks [4, 5]) und BPEL-Nachrichten [36] müssen noch geeignete BPEL-Konstrukte (z.B. While, Parallel-ForEach) verwendet werden, um den spezifizierten Kontrollfluss aus dem Systemprozess plattformabhängig zu realisieren. Dabei ist es wichtig, die Implementierung möglichst flexibel zu gestalten, um spätere Änderungen einfacher als bisher umsetzen zu können. Die Maßnahmen 6 und 7 ermöglichen eine solche Dynamik.

Maßnahme 6: Flexibilität durch Verwendung von Geschäftsregeln

Geschäftsregeln werden häufig eingesetzt, um Verzweigungsbedingungen in Geschäftsprozessen zu detaillieren und die eigentliche Verzweigungslogik vom Geschäftsprozess zu entkoppeln. Die Flexibilität entsteht durch die Auslagerung der Geschäftsregeln aus dem Geschäftsprozessmodell. Dadurch ist es möglich, sich häufig ändernde oder komplexe Verzweigungsbedingungen unabhängig vom Geschäftsprozess anzupassen [9].

Maßnahme 7: Serviceauswahl unter Nutzung v. Quality-of-Service-Kriterien

Damit Services zur Laufzeit anhand ihrer Attribute ausgewählt und gerufen werden können, ist bei ihrer Implementierung darauf zu achten, dass Service-Endpunkte nicht statisch im Code der Prozessapplikation gespeichert, sondern dynamisch ermittelt werden. Dies kann durch den Einsatz eines SOA-Repositories realisiert werden, in dem die Informationen über Service-Endpunkte und -Instanzen abgelegt werden. Eine Service-Instanz beschreibt eine konkrete Service-Installation auf einem Applikationsserver. Zur Identifikation der richtigen Service-Instanz werden zusätzlich zu der Endpunktinformation die Quality-of-Service (QoS)-Attribute (vgl. [14]) aller Service-Instanzen im Repository gespeichert. Im folgenden Beispiel zeigen wir, wie eine Service-Auswahl basierend auf QoS-Attributen durchgeführt werden kann, ohne die Implementierung des ausführbaren Modells anpassen zu müssen.

Beispiel 10 (Service Auswahl durch QoS-Attribute)

Die durch den Fachbereich festgelegten QoS-Attribute für Geschäftsprozessschritte oder fachliche Services werden zur Laufzeit genutzt, um die richtige Service-Instanz zu finden und aufzurufen. Dies ist deshalb notwendig, da für fachliche Services unterschiedliche technische Implementierungen existieren können, die von konsumierenden Prozessapplikationen zur Laufzeit unterschieden werden müssen. Dies ist insbesondere dann von Vorteil, wenn unterschiedliche Prozessapplikationen die gleiche Service-Instanz mit unterschiedlichen QoS-Eigenschaften verwenden wollen. So existiert beispielsweise der in Abbildung 9 verwendete Service Ser1 (*service_import_component_date_component b)* in unterschiedlicher Qualität, einerseits mit einer garantierten Antwortzeit t_x ($5ms \leq t_x \leq 15ms$) und andererseits mit einer schlechteren Antwortzeit $t_y \geq 25ms$. Damit zur Laufzeit die richtige Service-Instanz gefunden und gerufen werden kann, muss die Prozessapplikation so implementiert werden, dass der Aufruf der Service-Instanz abhängig von den QoS-Attributen (dynamisch) erfolgt. In Abbildung 9 wird ein solches Vorgehen beschrieben:

(1) Das QoS-Attribut *Antwortzeit* wird durch den verantwortlichen Modellierer aus dem Fachbereich festgelegt (z.B. Antwortzeit ≤ 15ms), indem das Attribut erzeugt und dem Geschäftsprozessschritt *Bewertung bzgl. betroffener Bauteile* zugewiesen wird.

(2) Der Systemprozess inklusive aller definierter QoS-Attribute wird erstellt.

(3) Für den in unserem Beispiel verwendeten Service Ser1 wird der Aufruf so implementiert, dass der Service-Endpunkt erst zur Ausführungszeit ermittelt werden muss. D.h. der Endpunkt wird anhand des Service-Namens und des geforderten QoS-Attribut in einem SOA-Repository zur Ausführungszeit ermittelt.

(4) Zunächst müssen dazu alle vorhandenen Service-Instanzen des Services Ser1 im zentralen SOA-Repository registriert werden. Dabei werden Endpunktinformationen sowie Eigenschaften (QoS-Attribute) der Service-Instanzen dokumentieren.

(5) Wird die Prozessapplikation nun ausgeführt, ermittelt sie den geforderten Endpunkt des Services, basierend auf den im Geschäftsprozess definierten QoS-Attribut (Antwortzeit ≤ 15ms). Dies erfolgt durch eine Anfrage am SOA-Repository, welches einen Dienst zur Auflösung solcher Anfragen implementiert.

(6) Dieser Repository-Dienst ermittelt die Service-Instanz, die die erforderlichen QoS-Attribute besitzt und gibt den zugehörigen Service-Endpunkt an die Prozessapplikation zurück.

(7) Der in (6) ermittelte Service-Endpunkt wir für den tatsächlichen Aufruf der Service-Instanz (Instanz A) durch die Prozessapplikation verwendet.

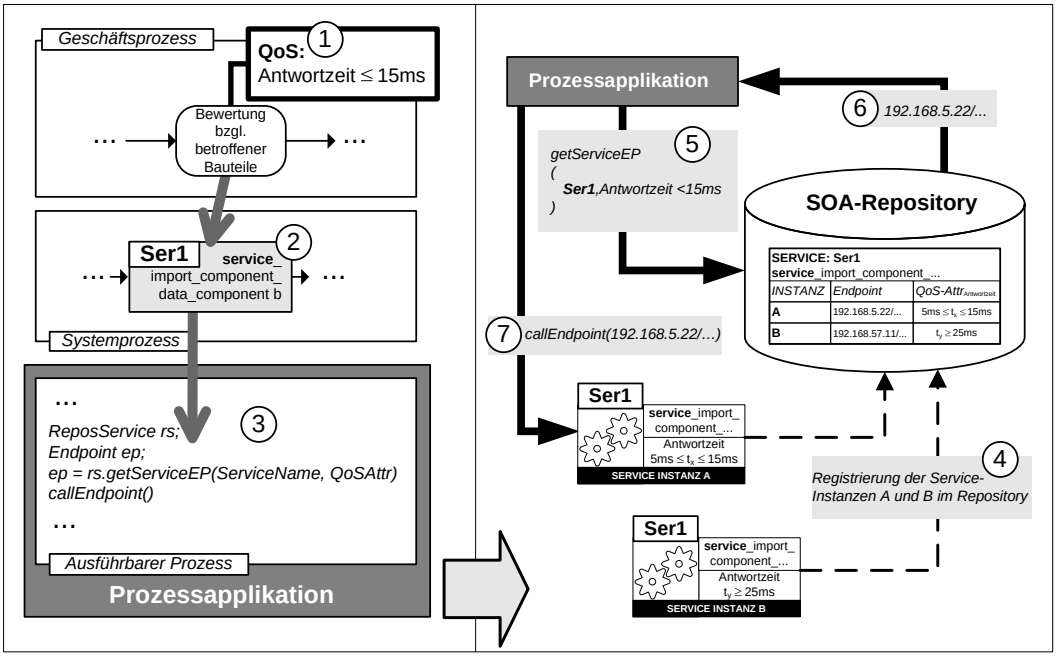

Abbildung 9 Service-Auswahl anhand QoS-Attributen

Ändert ein Fachmodellierer die QoS-Attribute, etwa die geforderte Antwortzeit, liefert das Repository den für die neuen QoS-Attribute passenden Service-Endpunkt (Instanz B). Die Prozessapplikation kann auf solche Änderungen reagieren, ohne die Implementierung anpassen zu müssen. Außerdem lassen sich zur Laufzeit neue Service-Instanzen einführen und im SOA-Repository speichern, die nun für Konsumenten direkt verwendbar sind.

4.5. Deployment (P5: Deployment der Software-Artefakte)

In der Deployment-Phase (vgl. Abbildung 3) wird das ausführbare Modell in Betrieb genommen, d.h. die Prozessapplikation wird zur Ausführung auf einem Applikations- oder Prozess-Server [43] bereitgestellt. Um die gewünschte Flexibilität bei der Ausführung zu erlangen, ist die Granularität der Deployment-Einheiten von besonderer Bedeutung. Jede aus dem Geschäftsprozessmodell ausgelagerte Funktionalität (z.B. Geschäftsregeln, vgl. Maßnahme 6) wird separat auf einem Applikationsserver installiert und ermöglicht dadurch eine vom Geschäftsprozessmodell unabhängige Anpassung. Konkret bedeutet das, dass beispielsweise bei Änderungen an Geschäftsregeln das Geschäftsprozessmodell nicht angepasst werden muss. Diese Entkopplung sorgt dafür, dass die Auswirkungen von Änderungen begrenzt werden können und somit die Prozessapplikation unverändert bleibt. Insbesondere sich in Ausführung befindliche Prozessinstanzen (vgl. Abschnitt 4.6) können ohne Unterbrechung das geänderte Verhalten übernehmen.

Maßnahme 8: Entkopplung beim Deployment durch Packaging

Packaging ist ein Vorgang, der beschreibt, in welche Einheiten eine Prozessapplikation zu „schneiden“ ist, um möglichst effizient und flexibel auf einem Applikationsserver betrieben werden zu können. Informationen aus frühen Phasen (P2, P3) dienen dabei als Indikatoren für eine solche Aufteilung. Dies sind z.B. Informationen zu Service-Implementierungen, Bearbeiterzuordnungen, Geschäftsregeln oder Sub-Prozessen. Entsprechende Artefakte werden unabhängig voneinander auf einer entsprechenden Ausführungskomponente [9] installiert.

4.6. Ausführung und Überwachung (P6: Prozessausführung und -überwachung)

Die Ausführung und Steuerung von Prozessapplikationen wird typischerweise durch eine Prozess-Engine realisiert (vgl. Abbildung 3).

Maßnahme 9: Prozesssteuerung durch Prozess-Engine

Die Nutzung einer adaptiven Prozess-Engine [19] ermöglicht Ad-hoc Instanz-Änderungen und Schemaevolution [42, 49, 50]. Zudem sind Funktionen wie Arbeitslistenverwaltung oder Prozess-Recovery in den meisten Prozess-Engines verfügbar und müssen daher nicht mehr in der Prozessapplikation realisiert werden. Durch Wiederverwendung solcher Grundfunktionen entsteht weniger Implementierungsaufwand für eine Prozessapplikation [41]. Darüber hinaus realisieren viele Prozess-Engines eine Integration in eine Monitoring-Komponente [9].

Maßnahme 10: Messung fachlicher Kennzahlen

Ein wichtiger Ausführungsaspekt von Prozessen ist die Messung der tatsächlichen Qualität durch Überwachung von Prozessapplikationen. Ziel des fachlichen Monitoring ist die Überwachung der Prozessapplikationen zur Sicherstellung der Geschäftsziele (oftmals als *Business Activity Monitoring* (BAM) bezeichnet). Eine solche Überwachung fordert zunächst die Definition von Leistungskennzahlen (*Key-Performance-Indicators* (KPI)) auf fachlicher Ebene [2], die z.B. auch Messstellen in Geschäftsprozessen kennzeichnen. Messstellen beschreiben bspw. Start- und Endzeitpunkte einer Zeitmessung während einer Geschäftsprozesssimulation. Ein solcher KPI könnte zum Beispiel die durch eine Fachabteilung vorgegebene maximale Durchlaufzeit des Änderungsprozesses beschreiben. Maßnahme 4 (Nachvollziehbarkeit zwischen Modellebenen) hilft nun, ausgehend von fachlich definierten KPIs, die tatsächlichen Messpunkte in der IT-Implementierung zu identifizieren. Diese Messpunkte beschreiben Ereignisse, die signalisieren, wann ein konkreter Prozessschritt im ausführbaren Modell gestartet bzw. beendet wird. Ausgehend von dieser Information kann nachvollzogen werden, welche Ereignisse in der IT-Implementierung zur Messung dieser KPI verwendet werden müssen. Die Flexibilität wird dadurch erhöht, dass neue KPIs leichter definierbar sind und vorhandene KPIs einfacher geändert werden können.

5 Änderungen der Umgebung einer Prozessapplikation

Ein besonders wichtiger Aspekt einer Prozessapplikation betrifft den Umgang mit Änderungen ihrer Umgebung. Letztgenannte beschreiben z.B. Veränderungen der von der Prozessapplikation genutzten Services, der Plattform und den IT-Infrastrukturkomponenten auf der die Prozessapplikation betrieben wird sowie des Organisationsmodells, das z.B. für die Auflösung der Rollen im Zusammenhang mit der Ausführung von Prozessschritten benötigt wird (vgl. Abbildung 10). Eine rasche und effektive Reaktion auf solche Umgebungsänderungen ist unabdingbare Vorraussetzung für jede flexible Prozessapplikation.

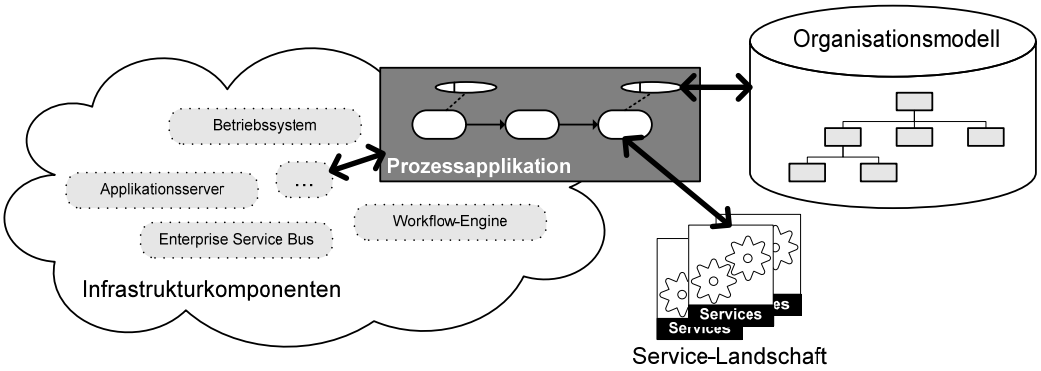

Abbildung 10 Umgebung einer Prozessapplikation

Einer Umgebungsänderung, etwa die Migration eines Services von einem Server A auf einen Server B, wirkt sich im Allgemeinen auf Prozessapplikationen aus. Allerdings kann durch geeignete Richtlinien vermieden werden, dass der Betrieb von Prozessapplikationen gefährdet wird. Im Folgenden beschreiben wir für typische Umgebungsänderungen, wie auf diese reagiert werden kann, ohne einen stabilen Betrieb der Prozessapplikationen zu beeinträchtigen.

5.1. Änderung der Lokation eines Services

Eine häufige Umgebungsänderung ist die Service-Migration: Wird ein Service an einen anderen Ort verschoben (Lokationsänderung), kann er für konsumierende Prozessapplikationen nicht mehr ohne weiteres gefunden werden, da diese lediglich Information über die ursprüngliche Service-Lokation besitzen. Dies führt zu einem Fehler bei der Ausführung der Prozessapplikation. Diese kann erst wieder nach Anpassung ihrer Implementierung in Betrieb genommen werden.

Maßnahme 11: Sicherstellung der Robustheit bei Lokationsänderungen

Die Robustheit einer Prozessapplikation bei Lokationsänderungen kann durch Einsatz von Proxies und Repositories sichergestellt werden. Ein Proxy ist eine Komponente, die einen definierten Proxy-Ablauf zur Modifikation von Daten beschreibt. Dies wird meist durch die Aneinanderreichung unterschiedlicher Ablaufknoten realisiert. Letztere transformieren Daten, rufen Services auf oder fragen ein Repository nach Informationen an. Prozessapplikationen können Proxy-Abläufe über eine standardisierte Schnittstelle (z.B. Web Service) verwenden. Folgende Szenarien beschreiben wie dies realisiert werden kann (s. Abbildung 11).

Szenario S1: Entkopplung mittels Proxy

Der Einsatz eines Proxies ermöglicht eine lose Kopplung von Prozessapplikationen und Service-Aufrufen. D.h. Prozessapplikationen rufen den konkreten Service nicht direkt auf, sondern verwenden einen Proxy (siehe (1) in Abbildung 11a). Der Proxy realisiert den eigentlichen Service-Aufruf mittels eines Proxy-Ablaufs (2) (*Call X @ A*), der den tatsächlichen Service aufruft (3). Wird der Service X nun von Lokation A auf Lokation B migriert, muss lediglich der Proxy-Ablauf angepasst werden und nicht die Service nutzenden Prozessapplikationen (vgl. Abbildung 11b). Konkret wird der Proxy-Ablauf so angepasst (2) (*Call X @* ***B***), dass er die *neue* Lokation B von Services X (3) verwendet.

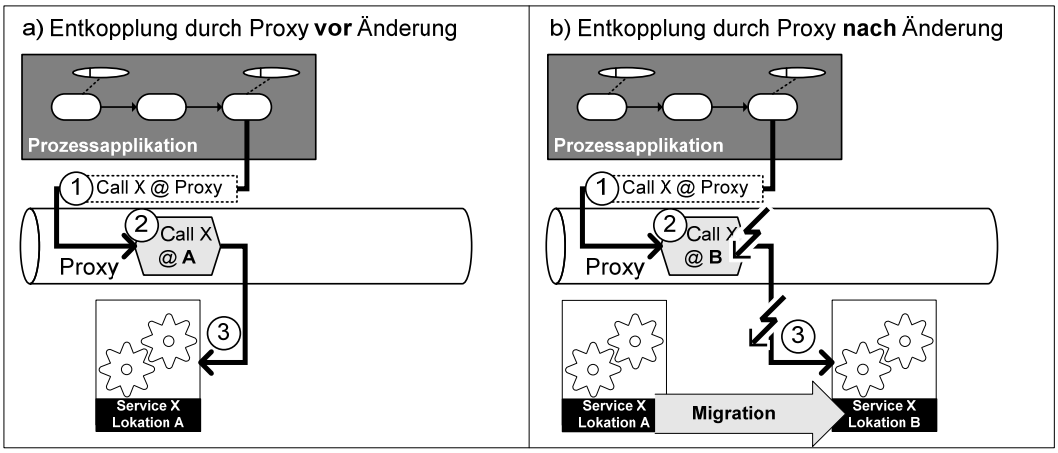

Abbildung 11 Szenarien zur Service-Migration

Szenario S2: Repository zur Endpunktsuche

Eine weitere Möglichkeit, Prozessapplikationen während einer Service-Migration stabil weiter betreiben zu können, ist der Einsatz eines SOA-Repositories (bzw. einer Registry) zwecks Verwaltung von Endpunktinformationen der Services. Ein solches SOA-Repository wird zur Endpunktauflösung eingesetzt (vgl. Abbildung 12a). Die Service-Lokation wird nicht im Proxy-Ablauf (hart) kodiert, sondern dynamisch zur Laufzeit ermittelt. Dazu ruft die Prozessapplikation analog zu Szenario S1 den Proxy auf (1). Anschließend wird mittels einer Repository-Anfrage (lookup) (2) der entsprechende Endpunkt (EP) des Services ermittelt (3). Hierzu müssen Informationen über Service-Endpunkte im Repository gespeichert und zur Laufzeit für den Proxy-Ablauf zur Verfügung gestellt werden (4). Als Ergebnis einer solchen Anfrage erhält der Proxy-Ablauf den konkreten Endpunkt des Services (5). Daraufhin führt ein weiterer Ablaufknoten (6) den eigentlichen Service-Aufruf durch (7).

Wird nun der Service X migriert (vgl. Abbildung 12b), muss lediglich der Eintrag im SOA-Repository angepasst werden, so dass die *neue* Service-Lokation B verwendet werden kann. Der Proxy-Ablauf muss nicht geändert werden, womit eine weitere Stufe der Endkopplung erreicht ist.

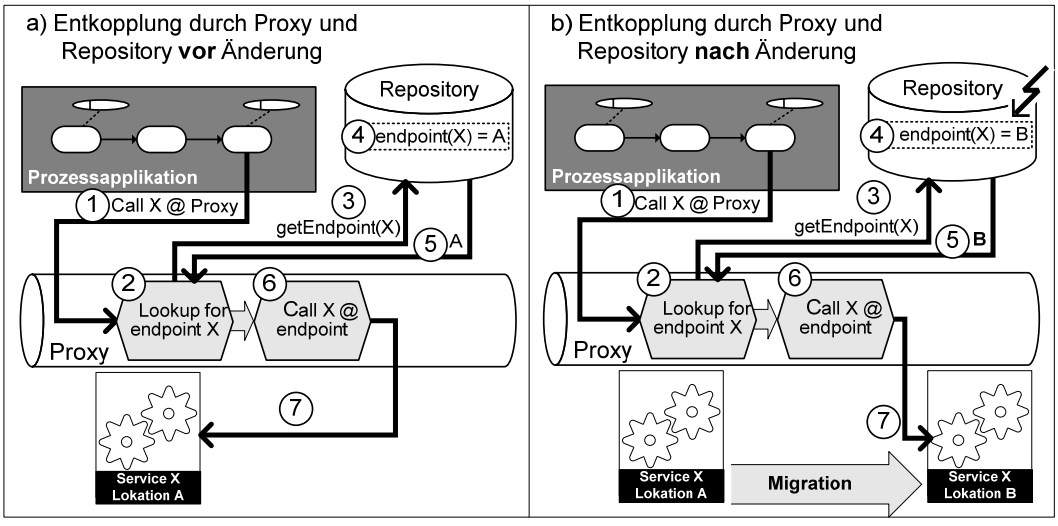

Abbildung 12 Szenarien zur Service-Migration

5.2. Service-Ausfall oder -Überlastung

Der Ausfall oder die Überlastung eines Services sind weitere relevante Ereignisse in der Umgebung einer Prozessapplikation: Ist ein Service spontan nicht mehr verfügbar, führt dies zu einer fehlerhaften Ausführung oder Blockierung der Prozessapplikation. Analog dazu führt eine Service-Überlastung dazu, dass garantierte Antwortzeiten ggf. nicht mehr eingehalten werden können. Dadurch wird die Ausführung der Prozessapplikation ebenfalls beeinträchtigt oder sogar fehlerhaft. Eine Vermeidung solcher Fehler wird durch Maßnahme 12 beschrieben.

Maßnahme 12: Parallelbetrieb von Services zur Ausfallsicherheit

Durch Parallelbetrieb mehrerer identischer Services (d.h. Service-Replikationen) sowie dem Einsatz eines Proxies und Repositories (vgl. Maßnahme 11) können solche Fehlersituationen oftmals vor der Prozessapplikation verborgen werden [40]. Dazu müssen zunächst die Service-Endpunkte für einen Parallelbetrieb im Repository registriert werden (0) (vgl. Abbildung 13). Analog zu den Maßnahmen 10 und 11 ruft die Prozessapplikation zunächst den Proxy-Ablauf (1), bevor dieser eine Anfrage an das SOA-Repository stellt (2), um den Service-Endpunkt des aufzurufenden Services X zu ermitteln (3). Im vorliegenden Beispiel sind im Repository für den Services X mehrere Endpunkte gespeichert (4). Als Ergebnis der Anfrage (3) erhält der Proxy-Ablaufknoten eine Liste aller im Repository registrierten Endpunkte des angefragten Services X (5). Aus dieser Liste wählt der Ablaufknoten (6) einen konkreten Service-Endpunkt aus und ruft daraufhin den Services X auf (7). Schlägt dieser Service-Aufruf fehl (8), wird der nächste Service-Endpunkt aus der Liste für einen Service-Aufruf herangezogen (9).

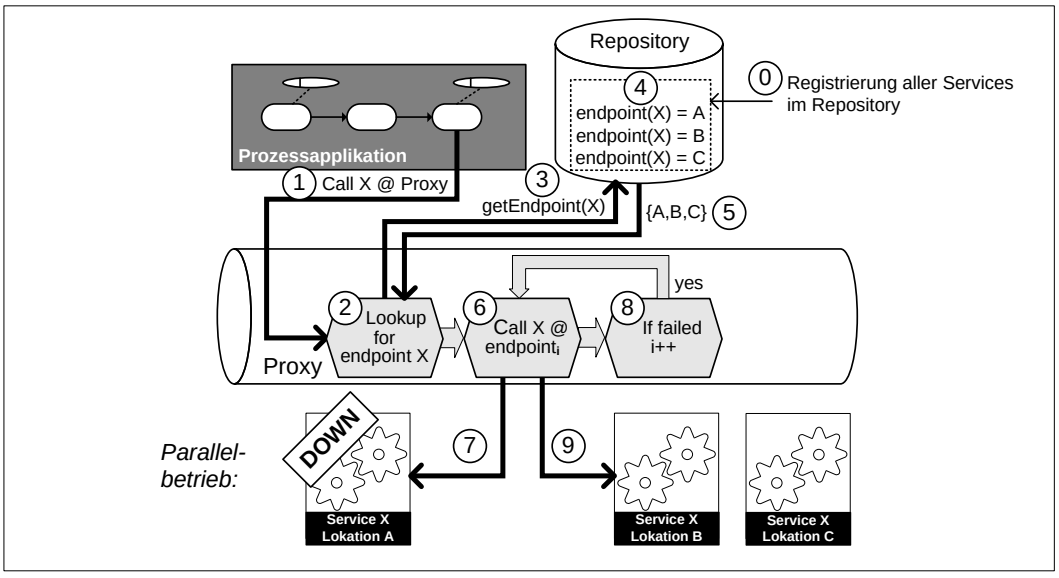

Abbildung 13 Proxy-Ablauf zum Verbergen von Service-Ausfällen

Darüber hinaus trägt der Parallelbetrieb von Service-Instanzen einer Balancierung der Last bei. Service-Aufrufe werden an denjenigen Applikationsserver weitergeleitet, der zum Zeitpunkt des Aufrufs am wenigsten ausgelastet ist. Die dazu benötigte Lastinformation kann z.B. in einem SOA-Repository als zusätzliche Information zur konkreten Service-Instanz gespeichert werden. Anfragen durch einen Proxy-Ablauf (3) liefern dann den Endpunkt der momentan am wenigsten belasteten Service-Instanz zurück.

5.3. Service-Versionswechsel und -Abschaltung

Änderungen über die Zeit werden durch Versionen beschrieben. So führen Änderungen an fachlichen Anforderungen oder Nachbesserungen in der Service-Implementierung (Bugfixes) zu neuen Service-Versionen. Letztere sollen alte Service-Versionen ablösen. Eine direkte Abschaltung alter Service-Versionen führt jedoch häufig zur aufwendigen Anpassung der Konsumenten, da die neue Service-Version nicht immer kompatibel zur alten Service-Version ist. Bei der Service-Versionierung wird deshalb prinzipiell zwischen kompatiblen und inkompatiblen Änderungen von Services unterschieden [24]. Eine inkompatible Service-Änderung erfordert immer eine Anpassung der Service-Konsumenten, in unserem Fall der Prozessapplikation. Ein Beispiel für eine inkompatible Änderung ist das Entfernen einer Service-Operation. Die folgenden Maßnahmen beschreiben, wie kompatible Änderungen vor Service-Konsumenten verborgen und inkompatible Änderungen frühzeitig analysiert werden können.

Maßnahme 13: Verbergen eines Service-Versionswechsels

Um einen Service-Versionswechsel für Service-konsumierende Prozessapplikation nachvollziehbar zu gestalten, verwenden wir analog zu Maßnahme 12 einen Proxy-Ablauf und eine SOA-Repository. Abbildung 14a zeigt wie ein konkreter Service-Aufruf (inkl. Datenobjekten) in einem Proxy-Ablauf realisiert werden kann. Die Prozessapplikation ruft den Endpunkt des Proxies auf (1) und übergibt ein Datenobjekt (Dat_{in}). Anschließend ermittelt der Proxy-Ablaufknoten (2) den Endpunkt (3) des aufzurufenden Services (4). Der Endpunkt des Services ist danach bekannt (5) und kann so für den tatsächlichen Service-Aufruf verwendet werden. Dazu ruft der Ablaufknoten (2) im Proxy den Service X (Lokation A) und übergibt an ihn das Datenobjekt Dat_{in} (6). Als Ergebnis des Aufrufs erhält der Proxy das Datenobjekt Dat_{out} (7), welches anschließend an die entsprechende Prozessapplikation zurückgegeben wird (8).

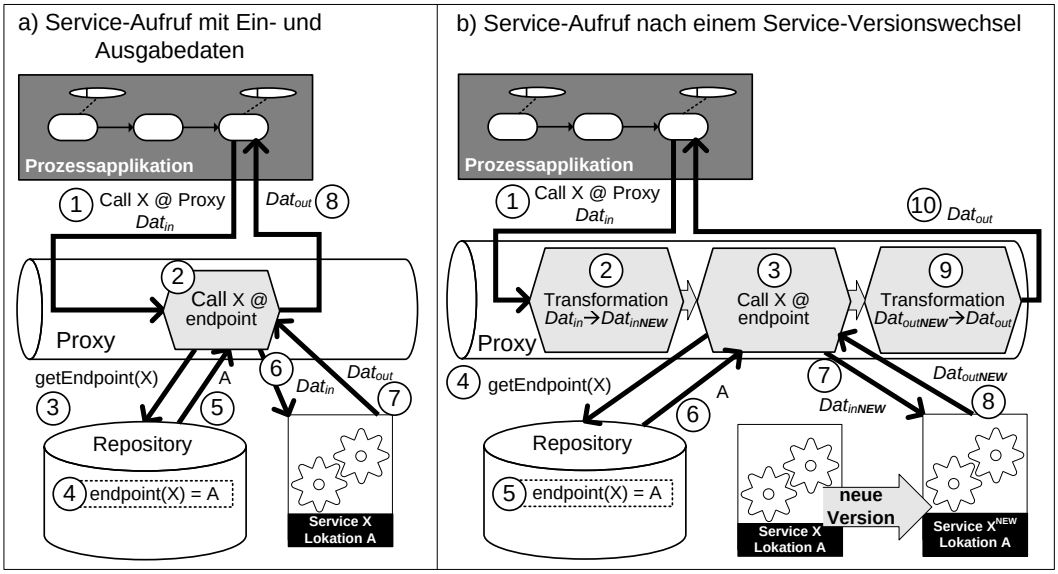

Abbildung 14 Service-Aufruf bei Versionswechsel

Abbildung 14b zeigt einen angepassten Proxy-Ablauf, wie er nach einem *Versionswechsel* von Service X aussehen kann: Eine Änderung der Implementierung von Service X führt zu einer neuen Version X^{NEW} des Services. Bedingt durch diese Anpassung hat sich die Schnittstelle des Services geändert. Eine zusätzlich notwendige Operation für Service X^{NEW} sorgt dafür, dass der bisherige Eingabedatentyp Dat_{in} angepasst werden muss (Dat_{inNEW}). Ebenfalls geändert hat sich der (Ausgabe-) Datentyp Dat_{out} in Dat_{outNEW}. Durch eine intelligente Anpassung des Proxy-Ablaufs, kann eine meist sehr aufwendigen Anpassung der Prozessapplikation

bei solchen Änderungen vermieden werden. Dazu werden zwei zusätzliche Ablaufkonten zur Datentransformation im Proxy-Ablauf hinzugefügt, die den alten in den neuen Datentyp konvertieren [38]. Dies muss einerseits für den veralteten Eingabedatentyp Dat_{in} (2) und andererseits für den Ausgabedatentyp Dat_{out} (9) realisiert werden. Durch diese zusätzlichen Ablaufknoten können Service-Versionswechsel vor der Prozessapplikation verborgen werden.

Maßnahme 14: Ermittlung der Auswirkung von Änderungen durch Vorfeldanalyse

Änderungen an Services lassen sich nicht immer verbergen. Insbesondere erfordern inkompatible Änderungen, Anpassungen von Service-konsumierenden Prozessapplikationen. Eine solche Anpassung ist meist mit hohem Aufwand und hohen Kosten verbunden. Damit sie planbar wird, sollten notwendige Änderungen frühzeitig identifiziert werden können, etwa indem die Gültigkeitszeiträume von Services in einem SOA-Repository verwaltet werden [13]. Durch Analysen kann nun frühzeitig festgestellt werden, welche Service-konsumierenden Prozessapplikationen von Änderungen betroffen sein können und zu welchen Zeitpunkten ein Service-Versionswechsel geplant ist. Weiter sollten Governance-Prozesse die Versionswechsel bzw. die Service-Abschaltungen koordinieren. Hierdurch werden vor Bewilligung bzw. Durchführung von Umgebungsänderungen konkrete Aussagen zu deren Auswirkungen möglich.

5.4. Austausch von Infrastrukturkomponenten

Prozessapplikationen und Services werden in einer IT-Infrastruktur teilweise auf Plattformen betrieben, die für den Betrieb und die Überwachung der Prozessapplikation mit dafür spezifischer Software ausgestattet sind. Dazu gehören neben Applikationsservern verschiedene Infrastrukturkomponenten, wie Prozess-Engine oder Enterprise Service Bus [9]. Meist sind jedoch Prozessapplikationen an die entsprechenden Infrastrukturkomponenten gebunden, da diese spezielle API-Funktionen verwenden. Wird eine Infrastrukturkomponente ausgetauscht oder eine neue Version dieser eingeführt, müssen die Prozessapplikationen auf diese Komponente angepasst werden, um einen stabilen Betrieb sicher zu stellen.

Maßnahme 15: Dokumentation der Betriebsumgebung

Informationen über Infrastrukturkomponenten müssen explizit dokumentiert werden. Nur dann ist nachvollziehbar, welche Prozessapplikation von welchen Infrastrukturkomponenten abhängig ist. Eine solche Dokumentation ist beispielsweise durch die Verwendung eines SOA-Repositories möglich. Nur wenn alle Abhängigkeiten zwischen Prozessapplikationen und Infrastrukturkomponenten gespeichert sind, kann analysiert werden, welche Auswirkung die Änderung einer Infrastrukturkomponente auf Prozessapplikationen hat. Für Änderungen ohne Auswirkungen kann die entsprechende Infrastrukturkomponente ausgetauscht werden, ohne den Betrieb von Prozessapplikationen zu beeinträchtigen. Die Dokumentation konkreter Termine für die Einführung neuer Infrastrukturkomponenten tragen dazu bei, von einer tatsächlichen Infrastrukturänderung, betroffene Prozessapplikationen zu ermitteln und Änderungseffekte abzuschätzen und - falls nötig - betroffene Prozessapplikationen anzupassen. Eine solche Dokumentation ermöglicht, analog zu Maßnahme 14, eine Analyse von Änderungen betroffener Infrastrukturkomponenten und Prozessapplikationen.

5.5. Änderungen am Organisationsmodell

Änderungen am Organisationsmodell werden meist durch den Fachbereich oder durch eine Umstrukturierung des Unternehmens ausgelöst [46, 47]: Abteilungen werden verschmolzen und Rollenzuordnungen ändern sich. Prozessapplikationen referenzieren bestimmte Organisa-

tionseinheiten aus einem Verzeichnisdienst (z.B. Rolle *project_manager* in Abbildung 7). Wird nun eine Rolle aus dem Organisationsmodell gelöscht, kann die Prozessapplikation nicht mehr korrekt ausgeführt werden.

Maßnahme 16: Verbergen von Änderungen durch Proxy-Elemente

Damit Prozessapplikationen bei der Anpassung von Organisationseinheiten unverändert bleiben, ist eine lose Kopplung zwischen Prozessapplikation und physischen Organisationsobjekten notwendig. Analog zu Abschnitt 5.1 kann hier die Entkopplung mittels eines Proxies realisiert werden. Dadurch wird die Rollenauflösung nicht durch die Prozessapplikation selbst realisiert, sondern durch den Proxy. Konkret bedeutet das, dass die Auflösung der Organisationseinheit durch einen Ablaufknoten realisiert wird und nicht in der Prozessapplikation implementiert ist [47].

5.6. Zusammenfassung

Abbildung 15 gibt einen Überblick über die in diesem Beitrag identifizierten Maßnahmen zur Erhöhung der Flexibilität in einer SOA. Zunächst werden Maßnahmen beschrieben (Maßnahme M1 bis M10), welche die Entwicklung von Prozessapplikationen flexibilisieren, bevor auf Änderungen der Umgebung von Prozessapplikationen eingegangen wird (Maßnahme M11 bis M16).

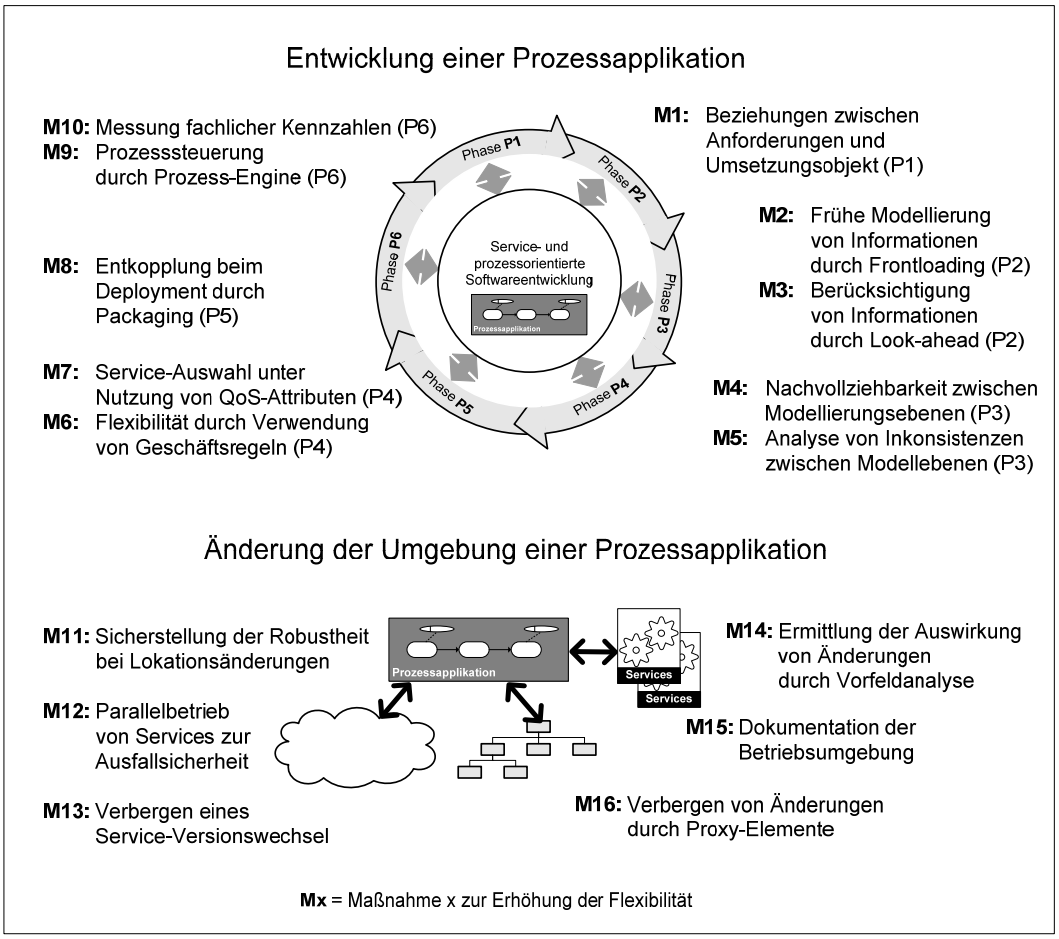

Abbildung 15 Maßnahmen zur Erhöhung der Flexibilität in einer SOA

6 Diskussion

Abschließend diskutieren wir weitere Ansätze zur Erhöhung der Flexibilität in einer SOA. Zunächst betrachten wir Methoden, die das grundlegende Vorgehen zur Entwicklung Service-orientierter Architekturen und Informationssysteme beschreiben. Anschließend diskutieren wir weitere Maßnahmen zur Erhöhung der Flexibilität.

[6, 31, 32, 56] beschreiben Methoden zur Entwicklung von Service-orientierten Architekturen und Informationssystemen. Dabei wird ein Vorgehen beschrieben, welches einen Leitfaden für die Einführung und den Betrieb Service-orientierter Anwendungslandschaften liefert. Konkrete Maßnahmen zur Erhöhung der Flexibilität in einer SOA, etwa nachvollziehbar dokumentierte Beziehungen zwischen fachlichen und technischen Prozessen (Maßnahme 4), werden allerdings meist vernachlässigt, was später zu erhöhten Betriebs- und Wartungskosten führen kann.

[23, 29] betrachten einzelne SOA-Aspekte, etwa die Bereitstellung eines SOA-Repositories für Service-Endpunkte oder die Verwendung eines Enterprise Service Bus (ESB) zum entkoppelten Aufruf von konkreten Service-Instanzen. Die Erhöhung der Flexibilität bei der Einführung und dem Betrieb einer SOA werden jedoch nicht betrachtet.

Einzelne der in diesem Beitrag vorgestellten Maßnahmen werden durch bereits existierende Ansätze abgedeckt. Für die Identifikation von Services gibt es unterschiedliche Methoden und Vorgehensweisen. Meist wird dabei zwischen einer fachlich (Top-down-Ansatz [60]) und einer technisch (Bottom-up-Ansatz [34]) orientierten Identifikation unterschieden. Auch eine Kombination beider Vorgehensweisen ist möglich [3, 22, 30]. Reine Top-down-Ansätze beschränken sich primär auf die fachliche Identifikation von Services. Dabei werden Services ohne Kenntnis der vorhandenen IT-Infrastruktur identifiziert, was die spätere Implementierung oft schwierig gestaltet. Andererseits betrachten Bottom-up-Ansätze lediglich technische Aspekte, was zur Implementierung feingranularer Services führt. Diese können meist nicht über Systemgrenzen hinweg eingesetzt werden, da sie für eine bestimmte Domäne konzipiert worden sind.

Darüber hinaus existieren Lösungsansätze [59] um fachliche Anforderungen und Geschäftsprozesse zu verzahnen (Maßnahme 1) und dadurch die Nachvollziehbarkeit zwischen ihnen sicherstellen. Weitere Arbeiten ermöglichen eine Entkopplung des Service-Aufrufs durch entsprechende Proxy- und Repository-Ansätze (Maßnahme 11) [45] sowie das Verbergen von Service-Versionswechseln (Maßnahme 13) [25]. Dabei klont der Service-Proxy das Service-Interface des Ziel-Services und wird als logischer Service für Nutzer veröffentlicht. Anfragen an den Service werden dann über die jeweilige Version an die betreffende Service-Implementierung weitergeleitet.

Der Umgang mit organisatorischen Änderungen, etwa infolge von Entlassungen oder Neueinstellungen von Mitarbeitern, wird in [46, 47] beschrieben. Dabei werden Fragestellungen im Zusammenhang mit der Entwicklung von Organisationsstrukturen diskutiert. Insbesondere wird ein Rahmenwerk vorgestellt, welches Änderungen an Organisationsmodellen sowie Zugriffsregeln ermöglicht und zugleich deren Auswirkungen analysiert. Diese Ansätze können zur Realisierung von Maßnahme 16 eingesetzt werden.

Insbesondere für die Modellierung des Fach- und IT-Modells (inkl. ihrer Beziehungen , vgl. Maßnahmen 2 bis 5 in Abbildung 15) finden sich in der Literatur keine geeigneten Ansätze, weshalb wir diese im Projekt *Enhanced Process Management through Service Orientation (ENPROSO)* detailliert bearbeiten. In ENPROSO wird ein Modellierungsansatz zur durchgängigen, nachvollziehbaren Dokumentation von Geschäftsprozessen entwickelt. Dabei werden ausgehend von fachlichen Anforderungen Geschäftsprozesse modelliert und in einer zu-

sätzlichen Modellierungsebene, dem sogenannten Systemmodell, technisch verfeinert. Anschließend wird das Systemmodell als Spezifikation zur Implementierung eines Prozess- und Service-orientierten Informationssystems an den IT-Bereich übergeben [10, 11, 12].

7 Zusammenfassung und Ausblick

In diesem Beitrag haben wir Maßnahmen diskutiert, welche die Flexibilität einer SOA bzgl. der Entwicklung, Inbetriebnahme und Wartung von Service- und Prozess-orientierten Applikationen erhöhen. Ein entsprechendes Architekturmanagement legt die Basis für eine stabile und flexible SOA. Darüber hinaus lassen sich auf dieser Grundlage Governance-Prozesse zur Steuerung und Verwaltung von Services und deren Lebenszyklen definieren. Um möglich schnell und kostengünstig von fachlichen Anforderungen zur entsprechenden Prozessapplikation zu gelangen, sind weitere Maßnahmen, etwa zur Dokumentation der Beziehungen zwischen Anforderungen und der konkreten Umsetzung, essentiell.

Erfolgt eine Änderung in der Umgebung einer Prozessapplikation, etwa die Migration eines Services auf einen anderen Server, so wirkt sich dies auf existierende Prozessapplikationen aus. Geeignete Mechanismen, welche die Auswirkung von Änderungen vor der betroffenen Prozessapplikation verbergen, ermöglichen einen Betrieb der unveränderten Prozessapplikation nach einer Umgebungsänderung.

Literatur

[1] Alves A., Arkin A., Askary S., Bloch B., Curbera F., Goland Y., Kartha N., Liu C.K., Mehta V., Thatte S., Yendluri P., Yiu A. (2007) Web Services Business Process Execution Language Version 2.0. OASIS Standard

[2] Amnajmongkol S., Angani J., Che Y., Fox T., Lim A., Keen M. (2008) Business Activity Monitoring with WebSphere Business Monitor V6.1. IBM Redbooks

[3] Arsanjani A., Ghosh S., Allam A., Abdollah T., Ganapathy S., Holley K. (2008) SOMA: A method for developing service-oriented solutions. IBM Systems J, 47(3):377-396

[4] Agrawal A. et al. (2007) Web Services Human Task. Technical Report, Active Endpoints, Adobe, BEA, IBM, Oracle, SAP AG

[5] Agrawal A. et al. (2007) WS-BPEL Extension for People Specification. Technical Report, Active Endpoints, Adobe, BEA, IBM, Oracle, SAP AG

[6] Arsanjani A.(2004) Service-oriented Modeling and Achitecture -How to identify, specify, and realize services for your SOA. IBM developerWorks, SOA and Web services, Technical library

[7] van der Aalst W.M.P, ter Hofstede A.H.M., Kiepuszewski B., Barros A.P. (2003) Workflow Patterns. Distributed and Parallel Databases, Springer, 14(3):5-51

[8] Barry D. K. (2003) Web Services and Service-oriented Architectures. Morgan Kaufmann

[9] Buchwald S., Bauer T., Pryss R. (2009) IT-Infrastrukturen für flexible, service-orientierte Anwendungen – ein Rahmenwerk zur Bewertung. Proc. 13. GI-Fachtagung Datenbanksysteme in Business, Technologie und Web (BTW'09), Münster, pp. 524–543

[10] Buchwald S., Bauer T., Reichert M. (2009) Durchgängige Modellierung von Geschäftsprozessen durch Einführung eines Abbildungsmodells: Ansätze, Konzepte, Notationen. Technical Report UIB-2009-12, Universität Ulm

[11] Buchwald S., Bauer T., Reichert M. (2010) Durchgängige Modellierung von Geschäftsprozessen in einer Service-orientierten Architektur. Proc. GI-Fachtagung Modellierung'10, Klagenfurt, pp. 203-211

[12] Buchwald S., Bauer T., Reichert M. (2011) Bridging the Gap Between Business Process Models and Service Composition Specifications. In: Int'l Handbook on Service Life Cycle Tools and Technologies: Methods, Trends and Advances (to appear)

[13] Buchwald S., Tiedeken J., Bauer T., Reichert M. (2010) Anforderungen an ein Metamodell für SOA-Repositories. CEUR Workshop on Services and their Composition (ZEUS'10), Berlin, Volume 563, pp. 17-24

[14] Bodenstaff L., Wombacher A., Reichert M., Jaeger M.C. (2008) Monitoring Dependencies for SLAs: The MoDe4SLA Approach. Proc. IEEE 5th Int'l Conf. on Services Computing (SCC'08), Honolulu, Hawaii, USA, pp. 21-29

[15] Chen H.-M. (2008) Towards Service Engineering, Service Orientation and Business-IT Alignment. Proc. 41st Hawaii Int'l. Conf. on System Sciences (HICSS'08), pp. 114-124.

[16] Chinnici R., Moreau J. J., Ryman A., Weerawarana S. (2007) Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. www.w3.org/TR/wsdl20/

[17] Penta M., Esposito R., Villani M. L., Codato R., Colombo M., Nitto E. D. (2006) WS binder: A Framework to Enable Dynamic Binding of Composite Web Services. Proc. Int'l Workshop on Service-oriented Software Engineering (SOSE'06), New York, NY, USA, pp. 74-80

[18] Dadam P., Reichert M. (2009) The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support - Challenges and Achievements. Springer, Computer Science - Research & Development, 23(2):81-97

[19] Dadam P., Reichert M., Rinderle-Ma S., Goeser K., Kreher U., Jurisch M. (2009) Von ADEPT zur AristaFlow BPM Suite - Eine Vision wird Realität: "Correctness by Construction" und flexible, robuste Ausführung von Unternehmensprozessen. EMISA Forum, 29(1):9-28

[20] Dadam P., Reichert M., Rinderle-Ma S. (2011) Prozessmanagementsysteme: Nur ein wenig Flexibilität wird nicht reichen. Informatik-Spektrum, 34(4):364-376

[21] Enderle R. (2009) Frühe fachliche Modellierung ausführungsrelevanter Prozess-Aspekte, Prozessmodellierung in Zeiten von SOA. Diplomarbeit, Universität Ulm

[22] Engels G., Hess A., Humm B., Juwig O., Lohmann M., Richter J.P., Voß M., Willkomm J. (2008) Quasar Enterprise: Anwendungslandschaften serviceorientiert gestalten. dpunkt.verlag

[23] Erl T. (2005) Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall

[24] Fang R., Lam L., Fong L., Frank D., Vignola C., Chen Y., Du N. (2007) A Version-aware Approach for Web Service Directory. Proc. IEEE Int'l Conf. on Web Services (ICWS'07), pp. 406-413

[25] Frank D., Lam L., Fong L., Fang R., Khangaonkar M. (2008) Using an Interface Proxy to Host Versioned Web Services. Proc. IEEE Int'l Conf. on Services Computing (SCC'08), Los Alamitos, CA, USA, pp. 325-332

[26] Gotel O. C. Z., Finkelstein A. C. W. (1994) An analysis of the requirements traceability problem. Proc. of the First Int'l Conf. on Requirements Engineering (RE'94), pp. 94-101.

[27] Heffner R. (2005) Your Strategic SOA Platform Vision. Forrester Research

[28] IBM (2008) WebSphere Process Server, Version 6.2, White Paper

[29] Josuttis N.M. (2007) SOA in Practice, The Art of Distributed System Design. O'Reilly

[30] Klose K., Knackstedt R., Beverungen D. (2007) Identification of Services - A Stakeholder-Based Approach to SOA Development and Its Application in the Area of Production Planning. Proc. 15th Europ. Conf. on Inf Sys, St. Gallen, pp. 1802-1814

[31] Klückmann J. (2007) 10 Steps to Business-Driven SOA. IDS Scheer AG

[32] Kohlmann F. (2007) Service Identification and Design - A Hybrid Approach in Decomposed Financial Value Chains. Proc. M. Reichert, S. Strecker, K. Turowski (Hrsg.): Proc. EMISA'07, St. Goar, Germany, pp. 205-218

[33] Mutschler B., Reichert M., Bumiller J. (2008) Unleashing the Effectiveness of Process-oriented Information Systems: Problem Analysis, Critical Success Factors and Implications. IEEE Transactions on Systems, Man, and Cybernetics, 38(3):280-291

[34] Nadham E.G. (2004) Seven Steps To a Service-Oriented Evolution. Business Integration Journal, pp. 41-44

[35] Nakajima S. (2002) Model-Checking Verification for Reliable Web Service. Proc. OOPS-LA Workshop on Object-Oriented Web Services (OOWS'02), Seattle, Washington, USA, pp. 1-5

[36] OASIS (2007) Web Services Business Process Execution Language. Ver. 2.0

[37] OMG (2009) Business Process Model and Notation (BPMN). Beta 1, Ver. 2.0

[38] Pokraev S., Quartel D., Steen M.W.A., Reichert M. (2006) Semantic Service Modeling - Enabling System Interoperability. Proc. Int'l Conf. on Interoperability for Enterprise Software and Applications (I-ESA'06), Bordeaux, France, pp. 221-231

[39] Reichert M., Bauer T., Dadam P. (1999) Enterprise-Wide and Cross-Enterprise Workflow Management: Challenges and Research Issues for Adaptive Workflows. Proc. Workshop Informatik '99, Paderborn, pp.56-64

[40] Reiser H. P., Danel M. J., Hauck F. J. (2005) A flexible replication framework for scalable and reliable .NET services. Proc. Int'l Conf. Applied Computing'05, pp. 161-169.

[41] Reichert M., Dadam P., Kreher U., Jurisch M., Göser K. (2008) Architectural Design of Flexible Process Management Technology. Proc. PRIMIUM Subconference at the Multikonferenz Wirtschaftsinformatik (MKWI'08), Garching, Germany, CEUR Workshop Proceedings, Vol. 328

[42] Reichert M., Dadam P. (1998) ADEPTflex - Supporting Dynamic Changes of Workflows without Losing Control. Journal of Intelligent Information Systems, 10(2):93-129.

[43] Reichert M., Dadam P. (2000) Geschäftsprozessmodellierung und Workflow-Management - Konzepte, Systeme und deren Anwendung. Industrie Management, 16(3):23-27

[44] Reichert M. (2000) Dynamische Ablaufänderungen in Workflow-Management-Systemen. Dissertation, Fakultät für Informatik, Universität Ulm

[45] Rohmann M. (2008) Managing services dynamically using WebSphere DataPower SOA Appliances with WebSphere Service Registry and Repository. IBM developerWorks, WebSphere, Technical library

[46] Rinderle-Ma S., Reichert M. (2007) A Formal Framework for Adaptive Access Control Models. Journal on Data Semantics IX, Vol. LNCS 4601, pp. 82-112

[47] Rinderle-Ma S., Reichert M. (2008) Managing the Life Cycle of Access Rules in CEOSIS. Proc. 12th IEEE Int'l Enterprise Computing Conference (EDOC'08), Munich, Germany, pp. 257-266

[48] Rinderle-Ma S., Reichert M. (2009) Comprehensive Life Cycle Support for Access Rules in Information Systems: The CEOSIS Project. Enterprise Information Systems, 3(3):219-251

[49] Rinderle S., Reichert M., Dadam P. (2004) Flexible Support of Team Processes by Adaptive Workflow Systems. Distributed and Parallel Databases, 16(1):91-116

[50] Reichert M., Rinderle-Ma S., Dadam P. (2009) Flexibility in Process-aware Information Systems. LNCS Transactions on Petri Nets and Other Models of Concurrency (ToPNoC), LNCS 5460, Springer, pp. 115-135

[51] Rinderle-Ma S., Reichert M., Jurisch M. (2011) On Utilizing Web Service Equivalence for Supporting the Composition Life Cycle. Int'l Journal of Web Services Research, 8(1):41-67

[52] Reichert M., Stoll D. (2004) Komposition, Choreograhpie und Orchestrierung von Web Services: Ein Überblick. EMISA Forum, 24(2):21-32

[53] Rupp C. (2007) Requirements-Engineering und Management. Hanser, 2007

[54] Schmitzl J. (2010) Durchgängige Modellierung von prozessorientierten Anwendungen mit BPMN 2.0. Diplomarbeit, Universität Ulm

[55] Scheer A.-W. (2001) ARIS - Modellierungsmethoden, Metamodelle, Anwendungen. Springer

[56] Stein S. (2009) Modelling Method Extension for Service-Oriented Business Process Management. Dissertation, Universität Kiel

[57] Tiedeken J. (2011) Konzeption und Realisierung eines logisch zentralen SOA-Repositories. Diplomarbeit, Fakultät für Ingenieurwissenschaften und Informatik, Universität Ulm

[58] Weske M. (2007) Business Process Management - Concepts, Languages, Architectures. Springer

[59] Weidlich M., Grosskopf A., Lübke D., Schneider K., Knauss E., Singer L. (2009) Verzahnung von Requirements Engineering und Geschäftsprozessdesign. REBPM Workshop, Software Engineering 2009, Kaiserslautern, pp. 229-136

[60] Winkler V. (2007) Identifikation und Gestaltung von Services. Vorgehen und beispielhafte Anwendung im Finanzdienstleistungsbereich. Wirtschaftsinformatik, pp. 257-266

[61] Werth D., Leyking K., Dreifus F., Ziemann J., Martin A. (2006) Managing SOA through Business Services - A Business-Oriented Approach to Service-Oriented Architectures. ICSOC Workshop, Springer, pp. 3-13

[62] Weber B., Reichert M., Rinderle-Ma S. (2008) Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. Data and Knowledge Engineering, 66(3):438-466

[63] Weber B., Reichert M., Wild W., Rinderle-Ma S. (2009) Providing Integrated Life Cycle Support in Process-Aware Information Systems. Int'l Journal of Cooperative Information Systems, 18(1):115-165

[64] Weber B., Sadiq S., Reichert M. (2009) Beyond Rigidity - Dynamic Process Lifecycle Support: A Survey on Dynamic Changes in Process-aware Information Systems. Computer Science - Research and Development, 23(2):47-65

[65] Zielczynski P. (2007) Requirements management using rational requisitepro, First Edition, IBM Press