ulm university universität
uulm

Institut für Datenbanken und Informationssysteme
(Leiter: Prof. Dr. Peter Dadam)

Erhöhung der Durchgängigkeit und Flexibilität prozessorientierter Applikationen mittels Service-Orientierung

Dissertation
zur Erlangung des Doktorgrades Dr. rer. nat.
der Fakultät für Ingenieurwissenschaften und Informatik
der Universität Ulm

vorgelegt von
Stephan Buchwald
aus Ulm

2012

Amtierender Dekan:	Prof. Dr. Klaus Dietmayer
Gutachter:	Prof. Dr. Manfred Reichert Prof. Dr. Franz Schweiggert Prof. Dr. Thomas Bauer
Tag der Promotion:	23. Oktober 2012

für Sabrina...

Zusammenfassung

Höhere Flexibilität für IT-gestützte Prozesse ist eine der zentralen Erwartungen, die von Anwenderseite an eine Service-orientierte Architektur (SOA) gestellt wurden. Insbesondere sollen fachliche Anforderungen an Geschäftsprozesse rasch in betriebliche Informationssysteme, d.h. die technische Implementierung der Prozesse, überführt werden können. Des Weiteren ist die Fähigkeit, auf Änderungen der fachlichen oder technischen Ebene schnell und korrekt zu reagieren, unabdingbare Voraussetzung für den Betrieb prozessorientierter Applikationen in einer SOA. Eine Herausforderung ist in diesem Zusammenhang die Diskrepanz zwischen den Anforderungen der Fachbereiche und den vom IT-Bereich realisierten technischen Implementierungen (sog. *Business-IT-Gap*). Um den genannten Herausforderungen gerecht zu werden, bedarf es einer durchgängigen Definition, Verwaltung und Pflege von Prozessen, Services und Datenobjekten, sowohl auf fachlicher als auch auf technischer Ebene.

Informationen zum Beziehungsgeflecht zwischen fachlichen und technischen Prozessen, Services und Datenobjekten sind in heutigen Unternehmensarchitekturen meist nicht vorhanden, was zu weiteren Problemen führt. So ist etwa bei Außerbetriebnahme eines Services nicht immer nachvollziehbar, welche (prozessorientierten) Applikationen davon betroffen sind. Dadurch ist es wiederum schwierig sicherzustellen, dass die Deaktivierung einzelner Services oder Service-Versionen in der Folge nicht zu unerwarteten Fehlern führt, etwa dass ein implementierter Geschäftsprozesses nicht mehr ausführbar ist.

Die vorliegende Arbeit adressiert mit ENPROSO *(Enhanced Process Management through Service Orientation)* diese Problemfelder und stellt einen Ansatz zur Verbesserung der Konsistenz zwischen fachlichen Anforderungen und implementierten Prozessen dar. Die Verwaltung und Konsistenzsicherung des komplexen Beziehungsgeflechts fachlicher und technischer Artefakte wird durch geeignete Methoden und Vorgehensmodelle für eine durchgängige Prozessmodellierung unterstützt. So lassen sich bereits bei der fachlichen Modellierung benötigte Informationen (z.B. über wiederverwendbare Services) explizit dokumentieren. Dadurch entsteht bereits während der fachlichen Analyse und Konzeptentwicklung eine detaillierte Beschreibung des zu implementierenden Sachverhalts. Zudem ist es möglich, fachliche Anforderungen schon in frühen Phasen der Softwareentwicklung vollständig zu dokumentieren und dadurch Aufwände für die Implementierung in späteren Phasen zu reduzieren. Zur Verwaltung der von einer SOA benötigten Artefakte ist ein umfassendes und generisches Repository-Metamodell notwendig, das die konsistente Speicherung aller Artefakte mit allen relevanten Beziehungen ermöglicht. Auf diese Weise kann die Konsistenz der gegenwärtig im Repository dokumentierten Artefakte sichergestellt werden.

Inhaltsverzeichnis

Teil I

Motivation

1 Einleitung

In großen Unternehmen und Organisationen müssen heute eine Vielzahl von Prozessen unterstützt werden, um Unternehmensziele zu erreichen. Durch Einsatz von Prozess-Management-Technologie werden die Modellierung und IT-gestützte Ausführung von (Geschäfts-) Prozessen (sog. Geschäftsprozessmanagement oder *Business Process Management, BPM*) ermöglicht [RW12, Wes07]. Gleichzeitig haben sich De-facto-Standards etabliert. Diese ermöglichen einerseits die Modellierung von Prozessen, etwa auf Grundlage der *Business Process Modeling Notation (BPMN)* [Obj09], andererseits die technische Umsetzung der fachlich modellierten Prozesse (z.B. mittels *WS-BPEL* [AAA+07]). Im Allgemeinen findet die Modellierung von Geschäftsprozessen durch einen Verantwortlichen aus dem Fachbereich statt. Er dokumentiert die Aktivitäten, die zur Erreichung eines bestimmten Geschäftsziels notwendig sind. Dadurch werden die Unternehmensabläufe transparent und sie können verschiedenen Analysen unterzogen werden. Des Weiteren wird die Grundlage für eine Prozessautomatisierung geschaffen. Dabei werden die dokumentierten Geschäftsprozessmodelle an den IT-Bereich übergeben, wo Implementierer ausführbare und mit Anwendungskomponenten und Ressourcen verknüpfte Prozessmodelle erstellen. Diese werden anschließend in einem Prozess-Management-System zur Ausführung gebracht. Die zugehörige Prozess-Engine steuert dann die Ausführung des Prozessmodells entsprechend der definierten Abfolge seiner Prozessschritte. In diesem Zusammenhang wird zwischen Prozessschritten mit und ohne Benutzerinteraktionen unterschieden. Letztgenannte werden automatisch ausgeführt.

Neue Anforderungen oder geänderte Unternehmensstrategien sowie die stetige Weiterentwicklung des Produktportfolios erfordern von Unternehmen die kontinuierliche Anpassung ihrer Geschäftsprozesse [WRR07, WRR08]. Änderungen an Geschäftsprozessen verursachen allerdings aufwendige Anpassungen der korrespondierenden IT-Implementierung (im ausführbarem Prozessmodell), da sie durch den Fachbereich genehmigt und anschließend durch den IT-Bereich umgesetzt, getestet und freigegeben werden müssen. Vielversprechende Perspektiven bietet in diesem Zusammenhang die Einführung *Service-orientierter Architekturen (SOA)* [Erl07, Jos07,

Erl09, EKW+09]. Ihr Ziel ist es, Aufwände für Entwicklung und Wartung, etwa durch Wiederverwendung oder Entkopplung von Services, zu reduzieren. SOA wird zudem mit der Erwartung eingeführt, mehr Flexibilität bei der Entwicklung und Inbetriebnahme von Unternehmenssoftware sowie sinkende Wartungskosten für den Anwendungsbetrieb zu erzielen. Dies soll durch ein verbessertes Zusammenspiel zwischen Fach- und IT-Bereich während der Entwicklung service- und prozessorientierter Applikationen erreicht werden.

Ziel des in dieser Arbeit entwickelten ENPROSO-Rahmenwerks ist es, die Flexibilität service- und prozessorientierter Applikationen zu erhöhen, indem SOA-Ansätze und -Methoden entsprechend weiterentwickelt werden. Mit Flexibilität ist eine rasche Umsetzbarkeit fachlicher Anforderungen und Geschäftsprozesse in service- und prozessorientierte Applikationen gemeint.

1.1. Hintergrund

Zunächst werden grundlegende Begriffe vorgestellt. Unter **Geschäftsprozessmanagement** wird sowohl die fachliche Modellierung von Geschäftsprozessen als auch deren technische Umsetzung und Betrieb verstanden. Die fachliche Modellierung umfasst die Beschreibung fachlicher Anforderungen an das zu entwickelnde Informationssystem in Form der zu unterstützenden Geschäftsprozesse. Letztgenannte sind sich wiederholende Abläufe von Prozessschritten (Aktivitäten), die gewissen Regeln unterliegen. Gewöhnlich ist ein Geschäftsprozess mit betriebswirtschaftlichen Zielsetzungen und Geschäftsbeziehungen (Kunden-Lieferanten-Beziehungen) verbunden.

Ein Fachmodellierer erstellt einen Geschäftsprozess (im folgenden *Fachprozess* genannt) und gibt ihn an den IT-Bereich weiter. Dieser Bereich erstellt daraufhin eine IT-Spezifikation (im folgenden *Systemprozess* genannt) für die Implementierung der entsprechenden prozessorientierten Applikation (siehe 1 in Abbildung 1.1). Fachprozesse werden meist abstrakt beschrieben, wodurch der Aufwand für ihre Interpretation im IT-Bereich hoch ist. Oftmals gibt es auch Fehler bei der Erstellung des Systemprozesses. Dies liegt daran, dass für die technische Umsetzung der im Fachprozess dokumentierten Prozessschritte häufig mehrere technische Prozessschritte im Systemprozess notwendig sind und somit eine direkte Abbildung von Prozessschritten zwischen Fach- und Systemprozess nicht möglich ist. Der Systemprozess wiederum liefert die Grundlage für eine spätere Ausführung, weshalb alle hierfür benötigten Informationen (z.B. technische Service-Aufrufe im Kontext von Prozessschritten) möglichst vollständig im Systemprozess dokumentiert werden sollten. Der Systemprozess selbst wird unabhängig von der Zielplattform in einer für den Fach- und IT-Bereich verständlichen Sprache beschrieben (siehe 2 in Abbildung 1.1). Im Anschluss daran wird ein ausführbarer Prozess erstellt. Dieser wird direkt vom Systemprozess abgeleitet und durch zielplattformabhängige Informationen ergänzt (siehe 3 in Abbildung 1.1): Beispielhaft seien hier Datenobjekte, implementierte Services und Benutzerschnittstellen genannt. Diese Prozessbeschreibung wird dann zur Erzeugung entsprechender Prozessinstanzen und deren Ausführung an eine Prozess-Engine übergeben. Während der Ausführung von Prozessinstanzen werden bei der Bearbeitung einzelner Prozessschritte Services aufgerufen, etwa zur Ermittlung von Bauteilinformationen aus einem Produktdaten-Management-System. Die Reihenfolge der bei der Ausführung einer Prozessinstanz aufgerufenen Services wird durch den Kontroll- und Datenfluss der jeweiligen Prozessbeschreibung festgelegt. Ihre Steuerung durch die Prozess-Engine zur Laufzeit wird in einer SOA auch als Service-Orchestrierung bezeichnet [Pap08].

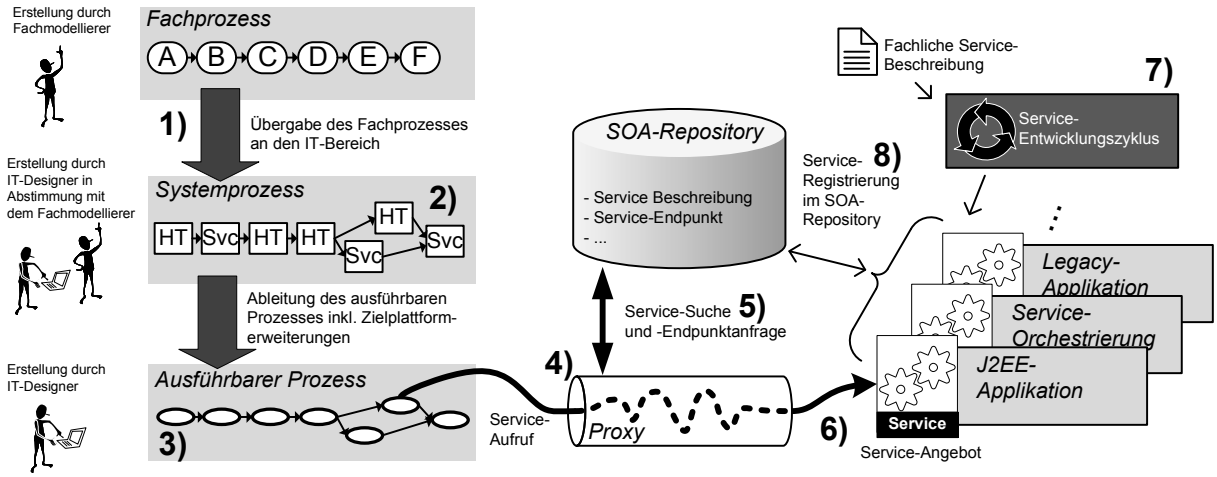


Abbildung 1.1.: Prozessorientierte Applikationsentwicklung in einer SOA

Unter einer **Service-orientierten Architektur (SOA)** versteht man nicht nur eine Technologie zum Aufruf von Services, sondern auch ein Vorgehensmodell zur Entwicklung und Inbetriebnahme service- und prozessorientierter Applikationen. Hierzu gehören auch Konzepte für das Management der SOA. In einer SOA werden Service-Aufrufe idealerweise durch einen sog. Proxy (siehe 4 in Abbildung 1.1) von der eigentlichen Applikation entkoppelt, d.h. ein Service-Aufruf wird nicht direkt in einem ausführbaren Prozess dokumentiert, sondern der tatsächlich aufzurufende Service wird von einem Proxy ermittelt. Dieser sucht in einem SOA-Repository nach dem passenden Service (siehe 5 in Abbildung 1.1) und ruft diesen mittels der dort dokumentierten Service-Endpunktinformation auf (siehe 6 in Abbildung 1.1). Das Repository stellt dazu alle notwendigen Informationen bereit. Ist ein Service noch nicht vorhanden, muss er neu spezifiziert und implementiert werden. Dazu muss ein Service-Entwicklungszyklus durchlaufen werden (siehe 7 in Abbildung 1.1). Dieser legt fest, in welcher Form der Service fachlich spezifiziert und technisch umgesetzt werden soll, bevor er im SOA-Repository registriert werden kann (siehe 8 in Abbildung 1.1).

1.2. Problemstellung

An der Erstellung und Umsetzung von Fachprozessen sind meist mehrere Personen des Fachbereichs beteiligt. Diese nehmen in der Regel verschiedene Sichtweisen ein und besitzen unterschiedliche Kompetenzen. Eine Herausforderung besteht darin, solche Fachprozesse mit möglichst geringem Aufwand durch eine service- und prozessorientierte Applikation auf technischer Ebene zu realisieren. Durch divergente Detailierungsgrade bei der Beschreibung von Fachprozessen auf der einen Seite und deren technischer Spezifikation auf der anderen Seite, entsteht eine Lücke zwischen Fach- und IT-Bereich, die auch *Business-IT-Gap* genannt wird [CGH+05]. Problematisch ist, dass Fachmodellierer die Fachprozesse üblicherweise abstrakt und unvollständig beschreiben, weshalb die Realisierung seitens des IT-Bereichs mit aufwendigen Anpassungen des Fachprozesses verbunden ist. Die Umstrukturierung von Fachprozessen durch Hinzufügen zusätzlicher Aktivitäten oder Entfernen bzw. Zusammenführen von Aktivitäten werden notwendig, um ein vollständi-

ges und durch eine Prozess-Engine ausführbares Modell zu erhalten. Zusätzlich müssen Informationen zu Services verfügbar sein, d.h. alle durch ein ausführbares Modell konsumierten Services müssen zur Laufzeit aufrufbar sein. Dazu ist eine service- und prozessorientierte IT-Infrastruktur [BBP09] notwendig, die neben einer zentralen Speicherung von Service-Informationen in einem SOA-Repository auch Infrastrukturkomponenten anbietet, um Service-Aufrufe entkoppelt von der service- und prozessorientierten Applikation (ausführbarer Prozess) zu realisieren. Außerdem sind Methoden für eine durchgängige Realisierung service- und prozessorientierter Applikationen (kurz: *Prozessapplikationen*) nötig. Neben der initialen Erstellung von Prozessapplikationen ist deren Änderung wichtig. Eine Herausforderung ist es, dass alle durch Änderungen verursachten Auswirkungen modellübergreifend behandelt werden. Ändert sich z.B. ein Fachprozess, müssen die Änderung bis zum ausführbaren Prozess propagiert und alle davon betroffenen Objekte (etwa konsumierte Services oder Datenobjekte) ermittelt werden. Im Projekt *Enhanced Process Management through Service Orientation (ENPROSO)* wird eine Erhöhung der Durchgängigkeit von Prozessapplikationen mittels Service-Orientierung angestrebt. In diesem Kontext ergeben sich folgende Punkte:

- **Service-orientierte IT-Infrastruktur:** Um Prozessapplikationen durchgängig und flexibel in einer SOA betreiben zu können, muss geklärt sein, welche Infrastrukturkomponenten mit welcher Funktionalität benötigt werden. Ferner sind diese Komponenten bzgl. der Erhöhung der Flexibilität von Prozessapplikationen zu bewerten.
- **Business-IT-Gap:** Ein weiteres Problem ergibt sich bei der Übergabe von Fachprozessen an den IT-Bereich (vgl. 1 in Abbildung 1.1). Fachprozesse sind meist unvollständig und ungenau dokumentiert. Zudem müssen Verantwortliche aus dem IT-Bereich hohe Aufwände in deren Interpretation stecken, wodurch hohe Kosten bei der Entwicklung der Systemprozesse resultieren. Dieser Business-IT-Gap erfordert neue Konzepte und Methoden für die Entwicklung von Prozessapplikationen. Insbesondere muss die Prozessapplikation die durch die fachlichen Anforderungen festgelegte Qualität erfüllen.
- **Durchgängige Modellierung von Prozessapplikationen:** Heutige Modellierungswerkzeuge unterstützen die Beschreibung von Fachprozessen und ausführbaren Prozessen. Was fehlt, ist ein Konzept für die durchgängige Modellierung von Prozessen. Dieses erstreckt sich von Fachprozessen, über die Erstellung zugehöriger Systemprozesse, bis zur Spezifikation und Implementierung der ausführbaren Prozesse auf technischer Ebene. Wichtig ist die nachvollziehbare Dokumentation der Abhängigkeiten zwischen den einzelnen Objekten der unterschiedlichen Modellierungsebenen. Beispielhaft genannt seien Abhängigkeiten zwischen Prozessschritten des Fach- und Systemprozesses. Insbesondere bei Änderungen an Fachprozessen oder technischen Artefakten (z.B. Services) muss feststellbar sein, welche Auswirkungen jeweils für das andere Modell entstehen. Nur so kann sichergestellt werden, dass Änderungen schnell und vollständig umgesetzt werden.
- **Zentralisierte Informationsbereitstellung:** Ein Repository soll alle in einer SOA notwendigen Informationen zentralisiert verwalten. Voraussetzung hierfür ist, dass ein einheitliches Metamodell existiert, das sowohl alle Anforderungen an eine SOA als auch an die Entwicklung von Prozessapplikationen abdeckt. Dazu gehören, neben der Beschreibung von Fachprozessen sowie der Beziehungen zwischen Prozessschritten aus Fach- und Systemprozessen, Service-Endpunkte und Informationen zum Betrieb von Prozessapplikationen. Das Repository muss zudem in der Lage sein, diese Informationen zu speichern.

- **Gesamtmethodik für SOA:** Eine ganzheitliche Vorgehensweise für die flexible Entwicklung, Inbetriebnahme, Ausführung und Wartung von Prozessapplikationen ist notwendig, um eine SOA dauerhaft stabil zu halten. Dazu gehören auch Governance-Prozesse für den Entwurf und die Entwicklung von Services sowie Methoden und Konzepte zur durchgängigen Modellierung von Prozessapplikationen.

1.3. Vorgehen

Das grundlegende Vorgehen dieser Arbeit lehnt sich an die in [MS95] beschriebene Methodik an. Letztgenannte unterteilt sich in einen Design-Prozess (*Erstellung* und *Evaluation*) und in die Design-Artefakte *Konstrukte*, *Modelle*, *Methoden* und *Instanzen*. Abbildung 1.2 begrenzt den Design-Prozess und die Design-Artefakte im Hinblick auf die Erhöhung der Durchgängigkeit und Flexibilität von Prozessapplikationen.

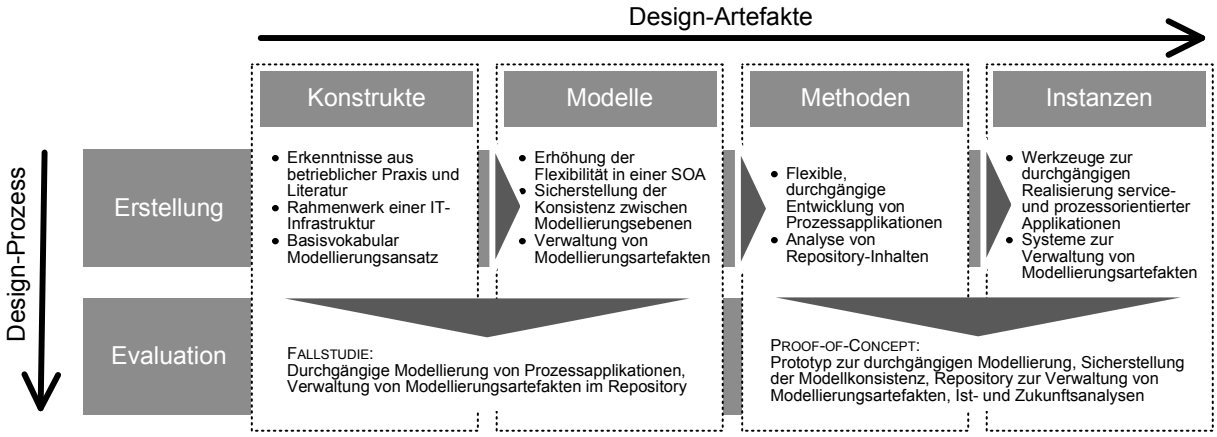


Abbildung 1.2.: Vorgehen angelehnt an [MS95]

Um Anforderungen an die Erhöhung der Durchgängigkeit und Flexibilität von Prozessapplikationen mittels Service-Orientierung zu identifizieren, werden Anwendungsszenarien aus der betrieblichen Praxis sowie aus der Literatur untersucht (*Konstrukte*). Es wird gezeigt, dass u.a. Modelle zur Entwicklung von Prozessapplikationen sowie Konzepte zur durchgängigen Modellierung benötigt werden (*Modelle*). Ebenso wichtig im Kontext dieser Arbeit ist die Entwicklung von Konzepten für ein zentral bereitgestelltes Repository sowie die Modellierung einer einheitlichen IT-Infrastruktur. Basierend auf diesen Erkenntnissen werden Methoden und Konzepte zur Realisierung von Prozessapplikationen entwickelt (*Methoden*), welche durch Werkzeuge und Systeme realisiert werden (*Instanzen*).

Die technische Realisierbarkeit und praktische Anwendbarkeit der in ENPROSO entwickelten Methoden und Konzepte wird in Teil III der Arbeit validiert. Dazu werden die entwickelten Konzepte prototypisch umgesetzt. Des Weiteren werden sie in einer Fallstudie evaluiert, um die praktische Anwendbarkeit der Konzepte zu untermauern.

1.4. Beitrag der Arbeit

In ENPROSO werden Konzepte und Vorgehensmodelle für Service-orientierte Architekturen entwickelt. Ein Ziel ist es, eine hohe Konsistenz zwischen fachlichen Anforderungen einerseits und der IT-Unterstützung von Prozessapplikationen andererseits zu erreichen. Zugleich soll für Prozessapplikationen erhöhte Flexibilität erzielt werden. Im Detail:

Erstens definiert ENPROSO ein Rahmenwerk, das **Flexibilitätsmaßnahmen** entlang des Entwicklungsprozesses für Prozessapplikationen beschreibt.

Zweitens spezifiziert ENPROSO eine **mehrstufige Modellierungsmethode** zur Entwicklung, Inbetriebnahme und Wartung von Prozessapplikationen. Die Modellierungsebenen beschreiben verschiedene Prozessaspekte, etwa Kontrollfluss, Datenobjekte, Geschäftsregeln oder Services.

Drittens werden in ENPROSO Konzepte zur **durchgängigen Modellierung von Prozessapplikationen** in einer SOA entwickelt. Neben methodischen Vorgaben werden Lösungen entwickelt, welche die Nachvollziehbarkeit zwischen einzelnen Elementen unterschiedlicher Modellierungsebenen ermöglichen und deren Konsistenz sicherstellen. Generell ist es eine komplexe Aufgabe, die Beziehungen zwischen fachlichen Aktivitäten und technischen Aktivitäten im Systemprozess bzw. ausführbaren Prozess zu erkennen, da diese meist nicht explizit dokumentiert vorliegen. Daher ist eine **nachvollziehbare Dokumentation solcher Abhängigkeiten** von grundlegender Bedeutung, um die Lücke zwischen Fach- und IT-Bereich zu schließen.

Viertens unterstützt ENPROSO Konzepte zur **frühzeitigen Definition implementierungsrelevanter Sachverhalte** in Fachprozessen (*Frontloading*). Dadurch soll bereits während der Geschäftsprozessmodellierung eine möglichst vollständige Beschreibung der zu implementierenden Prozessapplikation entstehen. Es werden nicht nur die Implementierungsaufwände reduziert, sondern auch Mehrdeutigkeiten bzw. falsche Interpretationen bei der Verfeinerung von Fachprozessen im IT-Bereich reduziert.

Schließlich werden grundlegende Konzepte für die Realisierung eines **SOA-Repositories** entwickelt, um Informationen für alle Unternehmensbereiche und Modellierungsebenen bereitzustellen. Das SOA-Repository legt die notwendige Basis für die unternehmensweite Governance von Prozessapplikationen und SOA-Komponenten. Insbesondere werden verschiedene Versionen von Fachprozessen, Systemprozessen und ausführbaren Prozessen, fachlichen und technischen Services sowie Datenobjekten im Repository dokumentiert. Darüber hinaus speichert und verwaltet das Repository umfassende Informationen bezüglich des gesamten Lebenszyklus von Prozessen und Services [WRWR09, Jos07]; dies inkludiert auch alle Beziehungen und Abhängigkeiten zwischen den dokumentierten Artefakten.

1.5. Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich in vier Teile (vgl. Abbildung 1.3): Teil I umfasst vier Kapiteln und beschreibt die grundlegenden Herausforderungen hinsichtlich der Erhöhung der Durchgängigkeit und Flexibilität von Prozessapplikationen in einer SOA. Zunächst werden in Kapitel 2 die Grundlagen eingeführt, die für das Verständis der Arbeit notwendig sind. In Kapitel 3 werden

Anwendungsbeispiele aus der betrieblichen Praxis analysiert, um reale Probleme bei der Entwicklung von Prozessapplikationen zu identifizieren. Dem folgt eine ausführliche Diskussion des Stands der Technik. Darauf basierend werden konkrete Anforderungen an den zu entwickelnden ENPROSO-Ansatz diskutiert. Diese dienen als Basis für die Konzeptentwicklung in Teil II der Arbeit. In Kapitel 4 wird eine SOA IT-Infrastruktur, die konkrete Infrastrukturkomponenten in unterschiedlichen Ausbaustufen detailliert und den jeweiligen Grad der durch die Komponente erreichbaren Flexibilität beschreibt, vorgestellt.

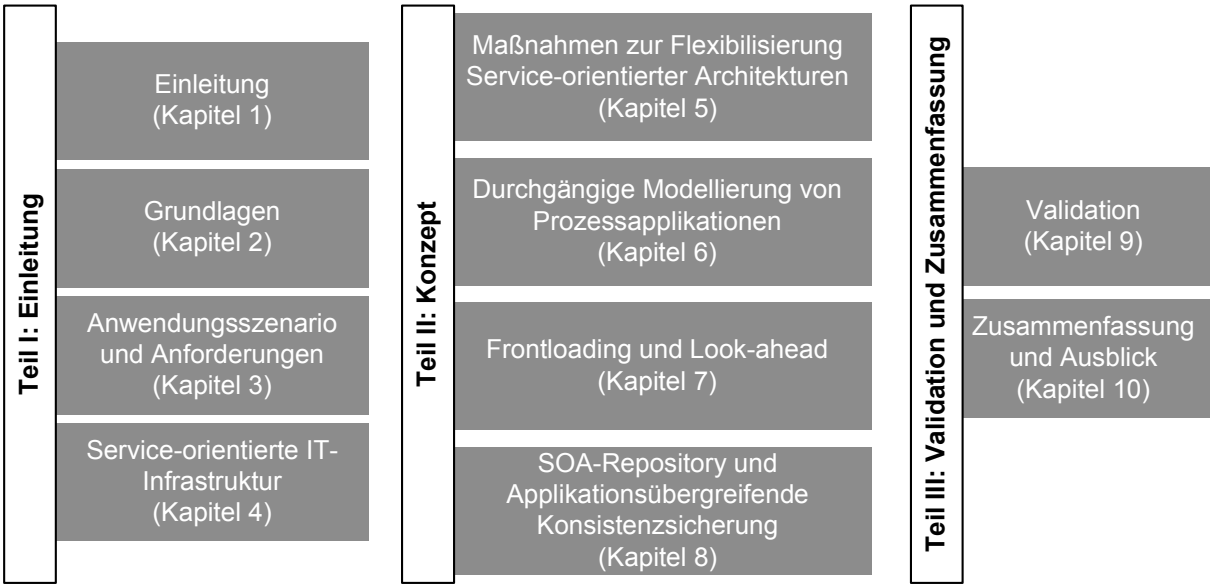


Abbildung 1.3.: Aufbau der Arbeit

Teil II der Arbeit stellt die entwickelten ENPROSO-Konzepte vor. Kapitel 5 entwickelt ein Rahmenwerk zur Flexibilisierung von Service-orientierten Architekturen. Konkret werden Maßnahmen beschrieben, wie sich die Flexibilität bei der Entwicklung einer Prozessapplikation oder Änderungen ihrer Umgebung steigern lässt. Anschließend beschreibt Kapitel 6 Ansätze für eine durchgängige Modellierung. Dazu werden sowohl Konzepte zur Transformation von Fachprozessen als auch Ansätze zur konsistenten Verwaltung und Pflege der Beziehungen zwischen Modellierungsebenen entwickelt. Im Anschluss werden Vorgehensweisen zur mehrstufigen Modellerstellung (von Fachprozess, Systemprozess und ausführbarem Prozess) vorgestellt und bewertet. Kapitel 7 beschäftigt sich mit der frühen Modellierung implementierungsrelevanter Informationen (*Frontloading*), d.h. diese Informationen sollen bereits während der Geschäftsprozessmodellierung möglichst vollständig erfasst und beschrieben werden, um spätere Implementierungsaufwände zu reduzieren. Nicht nur die frühe Dokumentation implementierungsrelevanter Informationen reduziert die Implementierungsaufwände, sondern auch die Wiederverwendung existierender Artefakte (*Look-ahead*). Hierfür werden in Kapitel 7 ebenfalls geeignete Konzepte entwickelt. Alle in einer Service-orientierten Architektur erstellten oder verwendeten Artefakte sollten zentral verwaltet werden. Das in Kapitel 8 entwickelte SOA-Repository-Metamodell bietet die Basis für eine solche zentrale Verwaltung der Artefakte. Zugleich liefern die dokumentierten Informationen die Basis für Repository-Analysen.

Teil III befasst sich mit der Validation der ENPROSO-Konzepte. In Kapitel 9 wird deren technische Realisierbarkeit durch prototypische Umsetzung gezeigt. Im Anschluss daran diskutiert eine Fallstudie die Anwendbarkeit der Konzepte in der Praxis. Kapitel 10 schließt mit einer Zusammenfassung und einem Ausblick auf zukünftige Arbeiten.

2
Grundlagen

Dieses Kapitel führt Begriffe und Konzepte ein, die für das Verständnis der Arbeit vonnöten sind. In Kapitel 2.1 wird zunächst der Begriff Service-Orientierung eingeführt, bevor in Kapitel 2.2 auf das Thema Geschäftsprozess-Management eingegangen wird.

2.1. Service-Orientierung

Unter einer *Service-orientierten Architektur* (SOA) [ACKM04, RS04, Erl05, AC05, Jos07, Pap08, BBP09, Mel10] versteht man ein Architekturparadigma, das die Modellierung, Implementierung und Ausführung von Services (Anwendungsdiensten) unterstützt. Eine zentrale Idee besteht darin, die Services verschiedener Anbieter einheitlich nutzbar zu machen. Services sind Softwarebausteine, die Anwendungsfunktionalität kapseln und anderen Applikationen als Dienstleistung zur Nutzung anbieten. Wird ein Service von mehreren Konsumenten genutzt, muss die Kopplung zwischen Service und Konsument lose sein, d.h. sie muss möglichst geringe Abhängigkeiten aufweisen. Das SOA-Paradigma erfordert darüber hinaus ein umfassendes Architekturmanagement (*Enterprise Architecture Management*, EAM), das Vorgehensweisen zur Gestaltung der SOA und Prozesse zur Umsetzung entsprechender Geschäfts- und IT-Strategien beinhaltet. Daneben werden Prozesse zur Kontrolle bzw. Planung der IT-Anwendungslandschaft betrachtet [EHH+08]. Dabei werden Informationen über vorhandene Applikationen und IT-Systeme sowie deren Services und Beziehungen zueinander dokumentiert. So wird z.B. festgehalten, welche IT-Systeme in den einzelnen Fachbereichen existieren und welche Services diese nutzen bzw. anbieten. Änderungen an Services wiederum können Auswirkungen auf die sie nutzenden Applikationen haben, etwa wenn diese nicht mehr korrekt ausführbar sind. Aus diesem Grund sollten Änderungsmanagement-Prozesse definiert und unterstützt werden. Sie müssen sicherstellen, dass konsumierende Prozessapplikationen rechtzeitig angepasst werden.

Definition 2.1 (Service-orientierte Architektur, SOA)
SOA ist ein Architekturparadigma, das Service-Orientierung unterstützt, d.h. das Denken in Services und eine service-basierte Softwareentwicklung. Das erfordert

- die Ausrichtung der IT an den fachlichen Anforderungen (Business-IT-Alignment), d.h. ein EAM zur strategischen Planung und Weiterentwicklung der IT, Governance-Prozesse zur Steuerung des Designs der Services und eine schnelle Reaktionsfähigkeit auf geänderte fachliche Anforderungen sind Bestandteile einer SOA.
- Gestaltungsprinzipien für Services [Erl07], z.B. Kapselung, lose Kopplung, standardisierte Schnittstellen, Protokolle, Auffindbarkeit, Wiederverwendbarkeit und Autonomie von Services.
- Prozessorientierung, d.h. Ausrichtung der Services an fachlichen Prozessen sowie IT-Unterstützung für Prozesse und Service-Orchestrierung [RS04, RRD04a, RR06, AAA+07].
- Vereinheitlichung der Service-orientierten IT-Infrastruktur [BBP09]

2.1.1. Service-Modellierung

Die Einführung neuer Services sollte durch Governance-Prozesse geregelt sein, d.h. es ist durch Entscheidungsträger zu prüfen, ob Services in der geplanten Form benötigt werden und den existierenden Richtlinien entsprechen. Als Entscheidungsgrundlage für eine solche Prüfung muss der Service zunächst fachlich beschrieben werden. Ausgangspunkt für eine solche Modellierung sind Service-Kandidaten. Ein Service-Kandidat beschreibt eine Service-Idee oder eine abstrakte Beschreibung der Funktionalität, welche als Dienstleistung realisiert werden soll.

Fachliche Services

In einem ersten Schritt erfolgt die fachliche Spezifikation des Services. Dabei werden fachliche Aspekte des Services dokumentiert: Service-Name, Kurzbeschreibung, Service-Level-Agreement (SLA), Quality of Service (QoS), Service-Status, Service-Operationen und Service-Verantwortlichkeiten [Ste08a, Ste08b, WLD+07]. Zudem werden fachliche Datenobjekte in Form von Ein- und Ausgabeparametern der Service-Operationen beschrieben. Die Festlegung fachlicher Information zu Services erfolgt früh, d.h. vor der Service-Implementierung. Dadurch können Konsumenten ihre Fach- und Systemprozesse so gestalten, dass der später umgesetzte Service auch tatsächlich verwendbar ist. Die Realisierung service- und prozessorientierter Applikationen kann dann schneller erfolgen, was die Flexibilität des Unternehmens erhöht.

Technische Services

Technische Services realisieren die Schnittstelle zu den IT-Systemen der Unternehmenslandschaft. Teile der Funktionalität, sofern sie für andere Anwendungen oder Domänen ebenfalls relevant sind, werden durch technische Services bestimmter IT-Systeme angeboten. Technische Services haben standardisierte Schnittstellen. Da nicht alle technischen Services direkt auf bestehende IT-Systeme zugreifen, nehmen wir folgende Unterteilung vor [ELK+06] (vgl. Abbildung 2.1):

- **Elementare Services:** Unternehmenslandschaften bestehen aus einer Vielzahl von IT-Systemen. Dazu zählen Datenbanken, Legacy-Anwendungen und Individual-Software. Um IT-Systeme in service-orientierten IT-Infrastrukturen nutzen zu können, muss ihre Funktionalität auch service-orientiert gestaltet werden. Dazu wird diese entweder mittels Adapter-Technik als Service publiziert oder es wird eine existierende Schnittstelle (Programmierschnittstelle des IT-Systems) durch bspw. Transformation an einen Service-Standard angepasst [ACKM04]. Elementare Services beschreiben demnach die von einem IT-System angebotene Funktionalität in abstrakter Form.
- **Zusammengesetzte Services:** Durch Komposition von Services [RS04] können Funktionen über mehrere IT-Systeme realisiert und als zusammengesetzter Service weiteren IT-Systemen zur Nutzung angeboten werden. Eine solche Komposition kann z.B. durch eine J2EE-Session-Bean oder durch einen (graphisch definierten) Ablauf entsprechend eines Orchestrierungsstandards [RS04], etwa WS-BPEL [RRD04a, RR06, AAA+07], realisiert werden.
- **Öffentliche Services:** Dies sind entweder elementare oder zusammengesetzte Services. Im Folgenden wird von öffentlichen Services als Grundabstraktion der IT-Systeme ausgegangen. Somit sind öffentliche Services die technischen Repräsentionen, die in Systemprozessen verwendet werden, nachfolgend *technische Services* genannt.

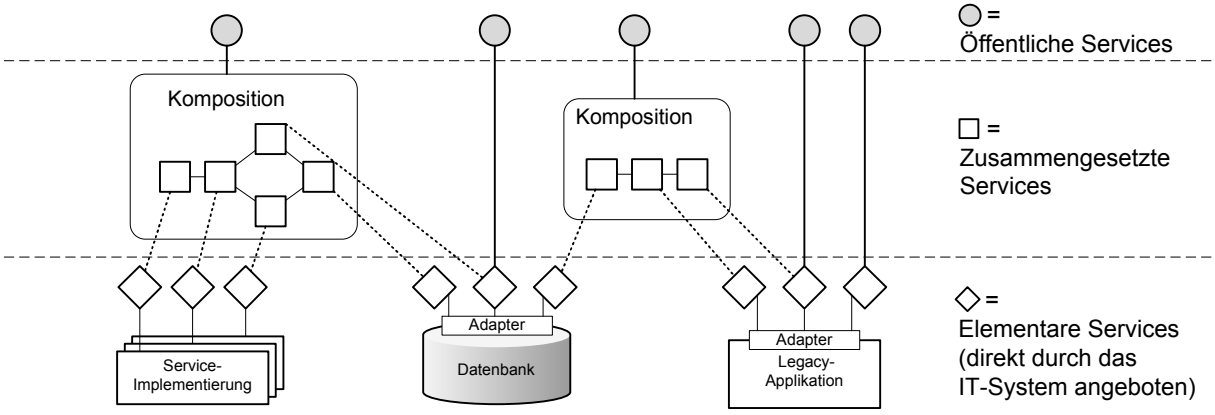


Abbildung 2.1.: Kategorisierung technischer Services nach [ELK+06]

2.1.2. SOA-Komponenten

Einen entscheidenden Aspekt einer jeden SOA stellt die Standardisierung der verwendeten IT-Infrastruktur dar. Diese ist wichtig, da eine solche Infrastruktur durch Einsatz unterschiedlicher SOA-Komponenten die Basis für die Entwicklung service- und prozessorientierter Applikationen ist. Dazu gehören neben Applikationsservern für die Bereitstellung technischer Services ein Kommunikationsbus (Proxy), der einen von der Anwendung entkoppelten Service-Aufruf ermöglicht, sowie ein Service-Repository, in dem Systeme die von ihnen angebotenen Services registrieren.

Eine detaillierte Beschreibung der in einer IT-Infrastruktur notwendigen SOA-Komponenten erfolgt in Kapitel 4.

2.2. Geschäftsprozess-Management

Fachliche Anforderungen an eine zu entwickelnde Prozessapplikation werden häufig in Form von Geschäftsprozessen (im weiteren Verlauf der Arbeit als *Fachprozesse* bezeichnet) dokumentiert. Eine solche Dokumentation wird durch Fachmodellierer (Geschäftsprozess-Designer) realisiert - die unterstützt durch Fachprozessmodellierungswerkzeuge - Prozessabläufe modellieren. Anschließend werden die meist abstrakt beschriebenen Fachprozesse an den IT-Bereich übergeben, der dann die IT-Spezifikation der zu implementierenden Applikation vornimmt. Konkret wird dies durch Systemmodellierer (IT-Designer) realisiert, die basierend auf dem Fachprozessmodell ein Systemprozessmodell in Abstimmung mit dem Fachmodellierer erstellen. Dieses Systemprozessmodell dient als IT-Spezifikation für die Implementierung der zu entwickelnden service- und prozessorientierten Applikation (ausführbares Modell).

Wir definieren im Folgenden die Umgebung der für ein Geschäftsprozess-Management relevanten Aspekte.

2.2.1. Grundlagen

Zunächst geben wir grundlegende Begriffsdefinitionen, die zum Verständnis von ENPROSO nötig sind. Dazu wird der Begriff Modell (engl.: *Model*), mit relevanten Komponenten, eingeführt (vgl. Definition 2.2).

Definition 2.2 (Model)
Ein $Model = (ProSet, DatSet, SerSet)$ ist das Abbild eines Ausschnitts der realen Welt und wird durch ein 3-Tupel beschrieben.

- $ProSet = \{Pro_1, ..., Pro_n\}$, $n \in \mathbb{N}$ ist die Menge der Prozesse,
- $DatSet = \{Dat_1, ..., Dat_m\}$, $m \in \mathbb{N}$ ist die Menge der Datenelemente und
- $SerSet = \{Ser_1, ..., Ser_k\}$, $k \in \mathbb{N}$ ist die Menge genutzter Services.

Außerdem:

- $released(Model) = true :\Leftrightarrow Model$ wurde durch einen Verantwortlichen freigegeben. Als Verantwortlicher wird eine Person bezeichnet, die durch einen Fach- oder IT-Bereich autorisiert ist, das Modell freizugeben.

Um *Model* genauer zu beschreiben, werden nun seine einzelnen Bestandteile definiert. Definition 2.3 führt den Begriff Prozess (engl.: *Process*) ein. Er dient als Grundlage für die Definition der Begriffe Fach- und Systemprozess sowie ausführbarem Prozess. Ein Prozess beschreibt dabei, welche Prozessschritte zur Erreichung eines bestimmten Ziels in welcher Abfolge durchgeführt werden müssen. Die Reihenfolge der Prozessschritte kann mittels Modellierung von Sequenzen,

parallelen, bedingten Verzweigungen oder Schleifen vorgegeben werden. Entsprechende Festlegungen sind Gegenstand der Prozessmodellierung [Rei00, Wes07].

Definition 2.3 (Process)
Ein *Process* $Pro = (ProNodSet, ProEdgSet) \in ProSet$ ist ein 2-Tupel mit:

- $ProNodSet = \{ProNod_1, ..., ProNod_e\}$ $e \in \mathbb{N}$ ist die Menge der Knoten. Ein Knoten $ProNod_i = (NodeType_i, ProNodAttSet_i) \in ProNotSet$ ist ein 2-Tupel mit:
  - $NodeType_i \in \{Start, End, Activity, ANDSplit, ORSplit, XORSplit, ANDJoin, ORJoin, XORJoin, LoopEntry, LoopExit, DataObj\}$ beschreibt den Knotentyp
  - $ProNodAttSet_i = \{ProNodAtt_1, ..., ProNodAtt_s\}$, $s \in \mathbb{N}$ beschreibt die Menge den Knoten zugeordneten Attributen.
- $ProEdgSet = \{ProEdg_1, ..., ProEdg_l\}$, $l \in \mathbb{N}$ beschreibt die Menge der Kanten. Für eine Kante $ProEdg = (n1, n2, EdgType, ProEdgAttSet) \in ProEdgSet$ gilt:
  - $EdgType \in \{ControlFlow, DataFlow, Loop\}$ ist der Kantentyp
  - $n1$ direkter Vorgänger von $n2$ bzgl. Kantentyp $EdgeType$
  - $n2$ direkter Nachfolger von $n1$ bzgl. Kantentyp $EdgeType$
  - $ProEdgAttSet = \{ProEdgAtt_1, \ldots, ProEdgAtt_n\}$ ist eine Menge von Kantenattributen

Ferner:

- $proNodType(ProNod)$ sei der konkrete Knotentyp $NodeType$ des Knotens $ProNod \in ProNodSet$,
- $proEdgType(ProEdg)$ sei der konkrete Kantentyp $EdgeType$ der Kante $ProEdg \in ProEdgSet$,
- $nodes(Pro)$ sei die Knotenmenge $ProNod \in ProNodSet$ des Prozesses $Pro \in ProSet$,
- $edges(Pro)$ sei die Kantenmenge $ProEdg \in ProEdgSet$ des Prozesses $Pro \in ProSet$,
- $attr_{ProNodAtt}(ProNod)$, $proNodType(ProNod) = Activity$ liefert den Wert für Knotenattribut $ProNodAtt$ des Knotens $ProNod$, mit $ProNod \in ProNodSet$ und $ProNodAtt \in ProNodAttSet$,
- $attr_{ProEdgAtt}(ProEdg)$, $ProEdg \in ProEdgSet$, $ProEdgAtt \in ProEdgAttSet$, liefert den Wert für Kantenattribut $ProEdgAtt$ der Kante $ProEdg$.

Der Datenaustausch zwischen Knoten wird durch Datenelemente realisiert. Die Gesamtmenge $DatSet = \{Dat_1, ..., Dat_n\}$, $n \in \mathbb{N}$ der Datenelemente bildet den Datenkontext des Modells (vgl. Definition 2.2).

Definition 2.4 (Data Element)
Ein Datenelement $Dat \in DatSet$ ist im Allgemeinen komplex strukturiert (vgl. Definition 2.2). Die Funktionen $value_a(Dat)$ liefert den Wert des Attributes a dieses Datenelements. Des Weiteren liefert $value_{a.b}(Dat)$ den Wert des Sub-Attributes b des Attributes a, usw. Wenn ein Da-

tenelement $Dat \in attr_{InPar}(ProNod)$ mit $InPar \in ProNodAttSet$ existiert (vgl. Definition 2.3), dann hat der Knoten $ProNod$ den Eingabeparameter Dat. Andererseits hat der Knoten $ProNod$ das Datenelement Dat als Ausgabeparameter, genau dann, wenn gilt: $Dat \in attr_{OutPar}(ProNod)$ mit $OutPar \in ProNodAttSet$.

Defintion 2.5 beschreibt einen Service in einer Service-orientierten Architektur.

Definition 2.5 (Service)
Für Service $Ser = (SerOpSet, SerQoSAttSet, SerEP) \in SerSet$ (vgl. Definition 2.2) beschreibt

- $SerOpSet = \{SerOp_1, ..., SerOp_f\}$, $f \in \mathbb{N}$ die Menge von Service-Operationen des Service Ser: Für $SerOp_i = (OpType_i, OpInSet_i, OpOutSet_i) \in SerOpSet$ bezeichnet
  - $OpType_i \in \{OneWay, RequestResponse\}$ die Art der Service-Operation
  - $OpInSet_i = \{OpIn_1, ..., OpIn_n\}$, $n \in \mathbb{N}$ die Menge der Eingabedatenobjekte der Service-Operation
  - $OpOutSet_i = \{OpOut_1, ..., OpOut_m\}$, $m \in \mathbb{N}$ die Menge der Rückgabedatenobjekte der Service-Operation
- $SerQoSAttSet = \{SerQoSAtt_1, ..., SerQoSAtt_k\}$, $k \in \mathbb{N}$ die Menge zum Service Ser gehöriger Quality-of-Service-Attribute
- $SerEP$ den Service-Endpunkt des Services Ser

2.2.2. Fachmodell

Ein Fachmodell wird durch einen Fachmodellierer erstellt. Dieser verwendet ein Werkzeug zur Modellierung der zugehörigen Fachprozesse. Letztere dokumentieren die Abläufe aus Sicht eines Fachbereichs. Diese Abläufe werden später durch eine IT-Applikation realisert. Im Folgenden werden die Eigenschaften eines Fachmodells zunächst informell beschrieben.

Eigenschaft 2.1 (Verständlichkeit für Fachbereiche)

Ein Fachmodell muss verständlich für den Fachbereich festgelegt werden. Nur dann können Fachmodellierer alle Anforderungen des Fachbereiches an die zu entwickelnde Prozessapplikation im Fachmodell dokumentieren.

Eigenschaft 2.2 (Werkzeugunterstützung)

Die Definition eines Fachmodells soll für den Fachmodellierer möglichst komfortabel gestaltet werden. Dies beinhaltet auch eine Werkzeugunterstützung für die Modellierung von Fachprozessen sowie deren Simulation.

Eigenschaft 2.3 (Vollständige Modellierung relevanter Prozessaspekte)

Das Fachmodell dient der fachlichen Beschreibung der zu entwickelnden Prozessapplikation. Dazu gehören die fachliche Dokumentation der Fachprozesse und ihrer verschiedenen Aspekte. Letztere beschreiben den Kontroll- und Datenfluss des Fachprozesses und legen Bearbeiterzuordnungen für Prozessschritte und zeitliche Vorgaben (z.B. Zeitdauern) fest. Die Dokumentation organisatorischer Entitäten (z.B. Gruppen, Rollen, Abteilungen oder Mitarbeiterkompetenzen) trägt ebenfalls zur Vollständigkeit des Fachmodells bei, ebenso die Datenmodellierung. Letztgenannte dient der Beschreibung und Strukturierung von Datenobjekten und dokumentiert deren Verwendung im Fachprozess (z.B. Ein-/Ausgabeparameter von Prozessschritten).

Eigenschaft 2.4 (Modellierung fachlicher Services)

Die Einführung Service-orientierter Architekturen erfordert die fachliche Modellierung von Services. Diese werden durch Service-Anbieter (Provider) angeboten und durch Service-Konsumenten (Consumer) genutzt (vgl. Abschnitt 2.1). Bevor ein Service von einer IT-Applikation konsumiert werden kann, muss er fachlich beschrieben und technisch umgesetzt werden. Dazu gehört die Beschreibung seiner Funktionalität, der von ihm angebotenen Operationen, der Quality-of-Service-Attribute, der Service-Level-Agreements (SLAs) sowie der Abhängigkeiten zu anderen Services oder IT-Applikationen [WLD+07, Ste08b, SR08, ST07] (vgl. Definition 2.5).

Eigenschaft 2.5 (Eignung zur Prozessoptimierung)

Ein Fachmodell muss zur Prozessoptimierung geeignet sein, d.h., Fachprozesse müssen durch Angabe von Kennzahlen (Key Performance Indicators, KPIs) auswertbar sein.

Ein Fachmodell wird wie folgt definiert:

Definition 2.6 (Business Model ($\mathcal{B}Model$))
Ein Fachmodell $\mathcal{B}Model = (\mathcal{B}ProSet, \mathcal{B}DatSet, \mathcal{B}SerSet)$ ist ein 3-Tupel mit:

- der Menge von Fachprozessen $\mathcal{B}ProSet = \{\mathcal{B}Pro_1, \ldots, \mathcal{B}Pro_n\}$, $\mathcal{B}Pro_i = (\mathcal{B}ProNodSet_i, \mathcal{B}ProEdgSet_i)$ für $i = 1...n$ (vgl. Definition 2.3)
- der Menge fachlicher Datenelemente $\mathcal{B}DatSet$ (vgl. Definition 2.4)
- der Menge fachlicher Services $\mathcal{B}SerSet$ (vgl. Definition 2.5)

wobei alle in Definition 2.2 bis Definition 2.5 eingeführten Objekttypen (und Funktionen) jeweils mit $\mathcal{B}$ als der fachlichen Ebene (engl.: business level) zugeordnet gekennzeichnet werden.

Ein Fachprozess $\mathcal{B}Pro \in \mathcal{B}ProSet$ ist ein Arbeitsablauf, der einem bestimmten Unternehmenszweck dient. Fachprozesse sind sich wiederholende Sequenzen fachlicher Prozessschritte (*Fachprozessaktivitäten*), die gewissen Regelungen unterliegen. Gewöhnlich ist ein Fachprozess mit betriebswirtschaftlichen Zielsetzungen verbunden. Ein Arbeitsablauf kann innerhalb einer Organisationseinheit oder organisationsübergreifend durchgeführt werden [DR99, WFM99]. Fachprozessmodelle werden üblicherweise durch gerichtete Graphen repräsentiert. Zu diesem Zweck

existieren graphische Beschreibungssprachen für Fachprozessmodelle: *Business Process Modeling Notation (BPMN)* [OMG09a, Obj09], *erweiterte Ereignisgesteuerte Prozessketten (eEPK)* [Sch00, NR02], *UML-Aktivitätendiagramme* [Par98, Esh02, Sch04, JRH+04, OMG09b] und *Petri-Netze* [Pet81, Aal98] sowie die entsprechenden Werkzeuge zur Erstellung solcher Modelle [Sch00, MID04, Pet05]. Fachprozessmodelle dienen der Dokumentation von Unternehmensabläufen und stellen die Grundlage für eine technische Realisierung dar.

Beispiel 2.1 (Fachprozess in BPMN-Notation)
Ein Beispiel für einen Fachprozess zeigt Abbildung 2.2. Zur Modellierung wurde BPMN 2.0 verwendet. Dieser stark vereinfachte Prozess beschreibt den Umgang mit Produktänderungen in der Automobilbranche. Dabei stellt er sicher, dass Änderungen an Bauteilen nicht ohne explizite Verifizierung und Freigabe potenziell betroffener Bauteile realisiert werden (vgl. Abschnitt 6.1.1).

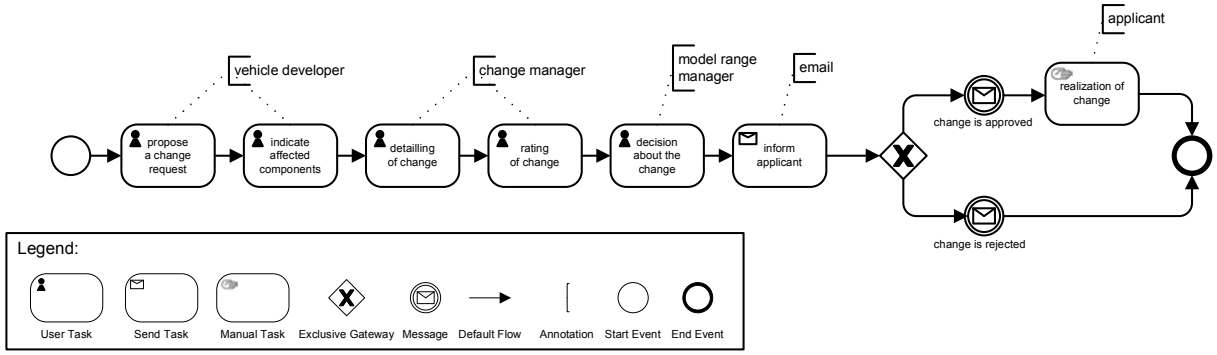


Abbildung 2.2.: Beispiel für einen Fachprozess in BPMN-Notation

2.2.3. Ausführbares Modell

Ein ausführbares Modell (siehe Definition 2.2) wird durch eine Prozess-Engine orchestriert [Pap08]. Dazu sind eine vollständige Beschreibung aller zielplattformabhängigen Sachverhalte nötig, d.h. es ist der ausführbare Prozess zu beschreiben, ebenso wie die verwendeten technischen Datenobjekte und implementierten Services, Benutzerschnittstellen (z.B. als Maskenentwurf), Geschäftsregeln und Organisationsmodelle. Aufgrund der hierfür notwendigen technischen Detaillierung ist es von *IT-Designern* zu erstellen. Dadurch liegt die Verantwortung für diese Modellebene beim *IT-Bereich*. In vielen Organisationen gibt es solche Verantwortliche aber nicht. Deshalb wird die Erstellung des ausführbaren Prozessmodells häufig an externe Software-Hersteller vergeben. Ein ausführbares Modell wird im Folgenden auch als *Prozessapplikation* bezeichnet.

Definition 2.7 (Executable Model ($\mathcal{EM}odel$))
Ein ausführbares Modell $\mathcal{EM}odel = (\mathcal{EP}roSet, \mathcal{ED}atSet, \mathcal{ES}erSet)$ ist ein 3-Tupel mit:

- der Menge ausführbarer Prozesse $\mathcal{EP}roSet = \{\mathcal{EP}ro_1, \ldots, \mathcal{EP}ro_n\}$, $\mathcal{EP}ro_i = (\mathcal{EP}roNodSet_i, \mathcal{EP}roEdgSet_i)$ für $i = 1...n$ (vgl. Definition 2.3)
- der Menge technisch spezifizierter Datenelemente $\mathcal{ED}atSet$ (vgl. Definition 2.4)

- der Menge ausführbarer Services $\mathcal{E}SerSet$ (vgl. Definition 2.5)

wobei alle in Definition 2.2 bis Definition 2.5 eingeführten Objekttypen (und Funktionen) jeweils mit $\mathcal{E}$ als der ausführbaren Ebene (engl.: executable level) zugeordnet gekennzeichnet werden.

Ein ausführbarer Prozess $\mathcal{E}Pro \in \mathcal{E}ProSet$ stellt die IT-Realisierung eines Fachprozesses $\mathcal{B}Pro \in \mathcal{B}ProSet$ dar [Rei00, Wes07]. Analog zum Fachprozess besteht ein ausführbarer Prozess aus einer Menge ausführbarer Prozessschritte. Letztere können als maschinell ausgeführte Programme (bspw. technische Services) oder menschliche Interaktionen realisiert werden. Abweichend vom Fachprozess muss ein ausführbarer Prozess vollständig detailliert sein [RS04, RW12]. Die Beschreibung ausführbarer Modelle kann durch unterschiedliche Modellierungssprachen realisiert werden, etwa die *Web Services Business Process Execution Language (WS-BPEL)* [AAA+07]. Abbildung 2.3 zeigt einen Teilausschnitt des in Abbildung 2.2 beschriebenen Fachprozesses in WS-BPEL.

Die Überführung eines Fachprozesses in einen ausführbaren Prozess wird durch den verantwortlichen IT-Designer realisiert. Dieser interpretiert den Fachprozess und definiert daraus einen ausführbaren Prozess. Ein Problem entsteht dadurch, dass zwischen beiden Ebenen häufig Transformationen durchgeführt werden müssen. So ist die Auswirkung einer Bauteiländerung in Abbildung 2.2 durch *eine* Fachprozessaktivität (*indicate affected components*) modelliert, wohingegen dieser Vorgang im ausführbaren Prozess durch *zwei* ausführbare Prozessschritte (*HT_ ChangeAppl_ ProductDevelopment_ InputPartNumber* und *Service_ ChangeAppl_ ProductDevelopment_ GetPartData*) realisiert wird. Neben einer solchen Aufspaltung kann es Fachprozessaktivitäten geben, für die keine technische Realisierung vorgesehen ist. Ein Beispiel dafür ist eine Fachprozessaktivität, die lediglich zur Dokumentation dient (z.B. das Begrüßen eines Kunden). Ebenso kann es notwendig sein, zusätzliche ausführbare Prozessschritte im ausführbaren Prozess zu realisieren, auch wenn für diese keine Entsprechung im Fachprozess existiert. Erschwerend kommt hinzu, dass Fachprozesse und ausführbare Prozesse unterschiedliche Namenskonventionen besitzen, etwa *inform applicant* in Abbildung 2.2 und *eMail_ ChangeAppl_ ProductDevelopment_ NotifyRequestor* in Abbildung 2.3. Die Zuordnung zwischen den Prozesschritten beider Modellebenen ist somit nicht ohne Weiteres erkennbar.

Die Abbildung von Fachprozessen auf ausführbare Prozesse erfordert meist komplexe Modelltransformationen. Dadurch gehen fachliche Anforderungen verloren und es entsteht ein hoher Aufwand bei späteren Prozessanpassungen. Deshalb wird eine zusätzliche Ebene zwischen Fachmodell und ausführbarem Modell eingeführt, um Prozesstransformationen besser zu unterstützen und den Ergebnisprozess besser mit den Fachbereichen abstimmen zu können.

2.2.4. Systemmodell

Um die Lücke zwischen Fach- und IT-Bereich zu schließen, ist eine zusätzliche Modellebene notwendig, welche die Beziehungen zwischen fachlichen Anforderungen und deren IT-Realisierung nachvollziehbar dokumentiert. Es existieren einige Ansätze, welche die Verwendung einer solchen Modellebene vorsehen. Beispielhaft seien an dieser Stelle M3 [PR05], SOMA [AGA+08, Ars04], AVE [BEH+07] und Quasar Enterprise [EHH+08] genannt. ENPROSO verwendet hierfür ein Zwischenmodell, im Folgenden auch *Systemmodell* genannt.

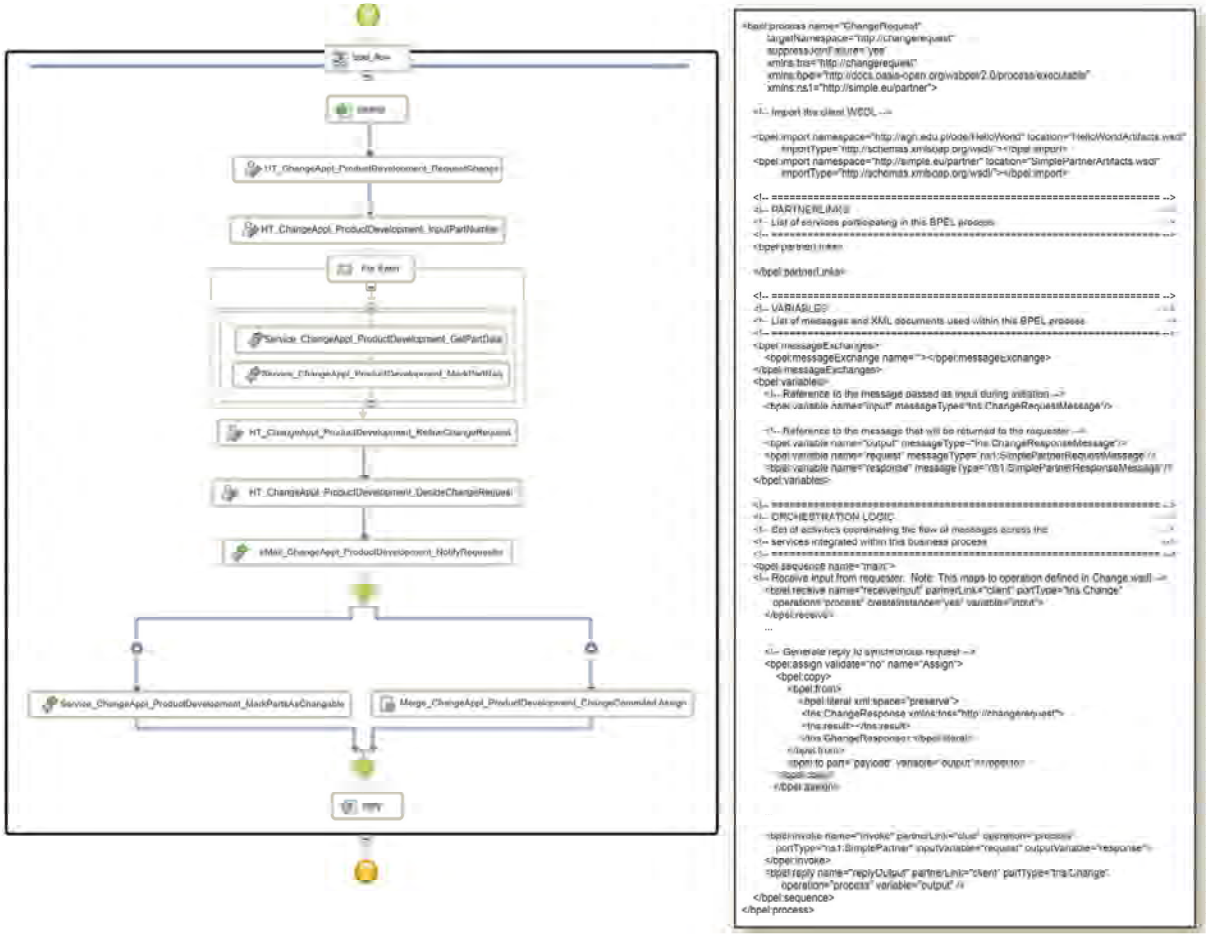


Abbildung 2.3.: Beispiel für ein ausführbares Prozessmodell

Definition 2.8 (System Model ($\mathcal{SModel}$))
Ein Systemmodell $\mathcal{SModel} = (\mathcal{SProSet}, \mathcal{SDatSet}, \mathcal{SSerSet})$ ist ein 3-Tupel mit:

- der Menge der Systemprozesse $\mathcal{SProSet} = \{\mathcal{SPro}_1, \ldots, \mathcal{SPro}_n\}$, $\mathcal{SPro}_i = (\mathcal{SProNodSet}_i, \mathcal{SProEdgSet}_i)$ für $i = 1...n$ (vgl. Definition 2.3)
- der Menge technischer Datenelemente $\mathcal{SDatSet}$ (vgl. Definition 2.4)
- der Menge technischer Services $\mathcal{SSerSet}$ (vgl. Definition 2.5)

wobei alle in Definition 2.2 bis Definition 2.5 eingeführten Objekttypen (und Funktionen) jeweils mit $\mathcal{S}$ als der Systemmodellebene (engl.: system level) zugeordnet gekennzeichnet werden.

Ein Systemmodell muss folgende Eigenschaften aufweisen:

Eigenschaft 2.6 (Formalitätsgrad)

Zu einem Systemmodell gehörende Systemprozesse müssen ausreichend beschrieben sein, so dass sie im Pflichtenheft für die IT-Umsetzung verwendet werden können. Hierzu muss die Bedeutung aller modellierbaren Objekte im Systemmodell eindeutig definiert sein, so dass ein ausführbares Model abgeleitet werden kann. D.h., Kontrollflussstrukturen im Systemprozess müssen besipielsweise verboten werden [Obj09], um eine technische Ausführbarkeit sicherzustellen. So ist etwa bei einem Rücksprung aus einem einzelnen Zweig einer Parallelität unklar, was dies für die anderen Zweige bedeuten soll (Abbruch oder Beendigung mit oder ohne erneute Ausführung der betroffenen Prozessschritte) [Rei00].

Eigenschaft 2.7 (Vollständigkeit)

Der Systemprozess muss vollständig sein, d.h. alle zur Modellierung notwendigen Kontrollfluss-Konstrukte müssen im zugrundeliegenden Metamodell enthalten sein. So wird neben AND-/OR-/XOR-Aufspaltungen und -Zusammenführungen auch ein Konstrukt für eine unbekannte Anzahl paralleler Zweige benötigt [AHKB03, AAA+07].

Eigenschaft 2.8 (Prozess-Aspekte)

Das Systemmodell muss alle relevanten Prozess-Aspekte abbilden und nicht auf den Kontrollfluss beschränkt sein. So müssen auch Datentypen, Organisationseinheiten und deren Beziehungen (Rollen, Abteilungen, Kompetenzen), Geschäftsregeln und Services beschreibbar sein. Diese Objekttypen müssen bei der Definition des Prozessgraphen referenzierbar sein.

Eigenschaft 2.9 (Verständlichkeit)

Da der Systemprozess mit dem Fachbereich abgestimmt werden muss, muss die Notation für diesen Anwenderkreis gut verständlich sein.

Eigenschaft 2.10 (Prozesstransformation)

Der Aufwand für die Transformation des existierenden Fachprozesses in Systemprozesse sollte möglichst gering sein. Dies ist der Fall, wenn die beiden Prozess-Metamodelle für diese zwei Ebenen identisch sind. Dann kann der Fachprozess übernommen und angepasst werden, anstatt dass er vollständig neu erstellt werden muss.

Eigenschaft 2.11 (Generierung des ausführbaren Modells)

Die Art der Prozessdokumentation im Systemmodell soll so gestaltet werden, dass eine direkte Ableitung des ausführbaren Modells möglich ist, d.h. alle im Systemmodell beschriebenen Artefakte sollten 1:1 in das ausführbare Modell übertragen sein. Die dazu notwendige Modelltransformation soll manuell oder automatisch durch entsprechende Werkzeuge (Generatoren) erfolgen.

Abbildung 2.4 einen stark vereinfachten Systemprozess. Dieser spezifiziert den in Abbildung 2.2 modellierten Fachprozess.

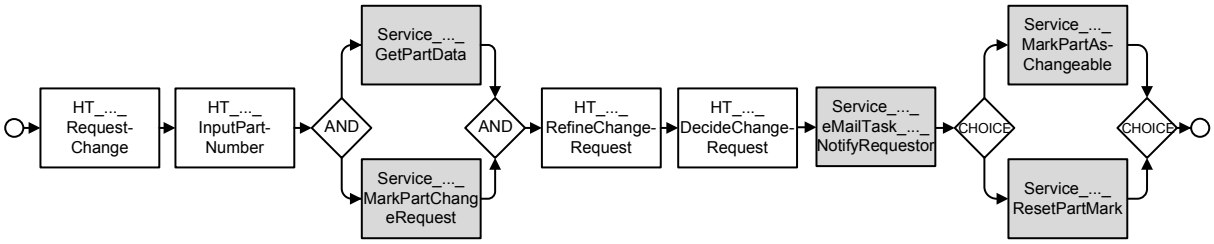


Abbildung 2.4.: Beispiel eines Systemprozesses

2.2.5. Modellierungsebenen und -aspekte

Als Basis zur Erhöhung der Durchgängigkeit und Flexibilität von Prozessapplikationen wird eine mehrstufige Modellierungsmethodik eingeführt, welche aus den zuvor beschriebenen drei Modellierungsebenen besteht. Abbildung 2.5 zeigt diese Modellierungsebenen.

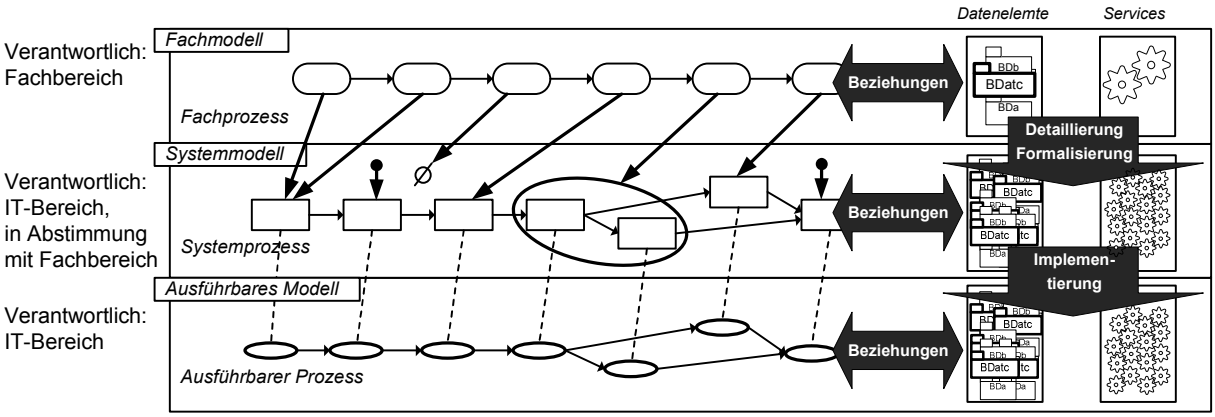


Abbildung 2.5.: Ebenen der Modellierung von Prozessen und auch anderer Aspekte

Die oberste Modellierungsebene wird durch einen Fachmodellierer aus einem bestimmten Fachbereich detailliert. Dieser erstellt ein Fachmodell, bestehend aus Fachprozessen, Datenobjekten und fachlichen Services sowie den Beziehungen dieser Artefakte. Anschließend wird dieses Modell an den IT-Bereich übergeben und von einem IT-Designer interpretiert. Dies geschieht in Abstimmung mit dem Verantwortlichen aus dem Fachbereich. Analog zum Fachmodell müssen im Systemmodell der Systemprozess (inkl. aller notwendiger Prozesstransformationen) und die zugehörigen Daten und Services technisch definiert werden. Die Abhängigkeiten zwischen einzelnen Objekten im Systemmodell (horizontal abgebildet in Abbildung 2.5) sind ebenso zu dokumentieren, wie die Beziehungen zum Fachmodell (vertikal). Anschließend wird das Systemmodell als IT-Spezifikation zur Implementierung freigegeben. Dann wird es als ausführbares Modell umgesetzt und durch eine Prozess-Engine zur Ausführung gebracht.

Die in den folgenden Kapitel entwickelten ENPROSO-Konzepte setzen die hier vorgestellte mehrstufige Modellierungsmethodik voraus.

3

Anwendungsszenario und Anforderungen

In diesem Kapitel werden spezifische Anforderungen an die Entwicklung und Inbetriebnahme von Prozessapplikationen erläutert. Abschnitt 3.1 beschreibt Anwendungsszenarien aus der betrieblichen Praxis, um tatsächlich auftretende Probleme bei der Entwicklung von Prozessapplikationen aufzuzeigen. Die anschließende Diskussion verwandter Arbeiten ermöglicht die Einordnung identifizierter Probleme (vgl. Abschnitt 3.2). Dazu wird analysiert, ob für diese Probleme bereits adäquate und in der Praxis anwendbare Lösungen existieren. Darauf basierend werden in Abschnitt 3.3 die Anforderungen an den zu entwickelnden ENPROSO-Ansatz abgeleitet. Diese dienen in den darauffolgenden Kapiteln als Basis für die ENPROSO-Konzeptentwicklung.

3.1. Anwendungsszenarien aus der betrieblichen Praxis

Im Folgenden werden unterschiedliche Anwendungsszenarien aus der Praxis vorgestellt, die zu Beginn des ENPROSO Projektes in verschiedenen Bereichen eines Automobilherstellers identifiziert wurden. Diese schließen die Personenkraftwagen- und Nutzfahrzeugentwicklung, den Vertrieb sowie Finanzdienstleistungen ein. Durch detaillierte Untersuchung verschiedener Projekte im Umfeld für Prozessapplikationen konnten konkrete Probleme identifiziert werden. Aus diesen wiederum lassen sich konkrete Anforderungen ableiten. Im Folgenden werden die Anwendungsszenarien in drei Kategorien eingeordnet (vgl. Abbildung 3.1):

- **Umsetzungsprojekte:** Entwicklungsprojekte, in denen die Implementierung service- und prozessorientierter Applikationen eine konkrete Problemstellung löst.
- **Vorgaben aus Konzern-Zentralbereichen und -Strategie:** Allgemeine, nicht projektspezifische Vorgaben aus der zentralen IT beschreiben Methoden zur Gestaltung von Prozessapplikationen. Konkrete Projekte berücksichtigen diese Vorgaben während der Software-Entwicklung.

- **Bereitstellung von Infrastrukturkomponenten:** Infrastrukturkomponenten und -plattformen bilden die Basis für die technische Realisierung von Umsetzungsprojekten entlang der Vorgaben aus der zentralen IT. Hierbei wird eine möglichst zentrale und standardisierte Lösung angestrebt.

Abbildung 3.1.: Kategorisierung der Anwendungsszenario

3.1.1. Umsetzungsprojekte für Prozessapplikationen

Die Implementierung von Prozessapplikationen erstreckt sich von der fachlichen Beschreibung bis zur tatsächlichen Implementierung. Dabei werden Fachprozesse definiert, die später als ausführbare Prozesse in den technischen Betrieb übergehen. Im Folgenden werden unterschiedliche Anwendungsszenarien aus der betrieblichen Praxis vorgestellt.

Anwendungsszenario I: Änderungsmanagement in der PKW-Entwicklung

Das erste Anwendungsszenario beschreibt ein Anwendungssystem zur Realisierung des Produktänderungsmanagements in der Automobilindustrie [HBR08b, HBR08c]. Der Umgang mit Produktänderungen gehört zu den zentralen Aufgabenbereichen in der Fahrzeugentwicklung. Die Steuerung des Produktänderungsmanagements wird durch Änderungsprozesse und unterstützende IT-Systeme realisiert. Änderungsideen werden in sogenannten Produktänderungsvorhaben (kurz: PÄV) frühzeitig erfasst und klassifiziert. Eine Klassifizierung kann etwa zur Einordnung eines Produktänderungsvorhabens als kosten-, zertifizierungs- und qualitätsrelevant verwendet werden. Der in Abbildung 3.2 dargestellte Fachprozess beschreibt den Ablauf einer solchen Produktänderung, beginnend mit der Erfassung eines PÄV, dessen Detaillierung und Bewertung, bis hin zur produktiven Umsetzung. Die einzelnen Phasen werden im Folgenden beschrieben.

Die erste Phase im Änderungsprozess beschreibt die Auslösung eines PÄV. Dort dokumentiert derjenige, der den Antrag ausgelöst hat, alle relevanten Informationen zum Änderungsvorhaben, etwa eine Beschreibung der Änderung und ihrer Ursachen oder Bauteil- und Kosteninformationen. Wurde im PÄV eine Baureihe vorgegeben, wird diese einem Gremium zur Klassifizierung und Freigabe (meist durch den Vorgesetzten des Auslösers) vorgelegt und ein PÄV-Verantwortlicher (PÄV-V) festgelegt. Ist eine Freigabe erteilt, detailliert der PÄV-V das Änderungsvorhaben in

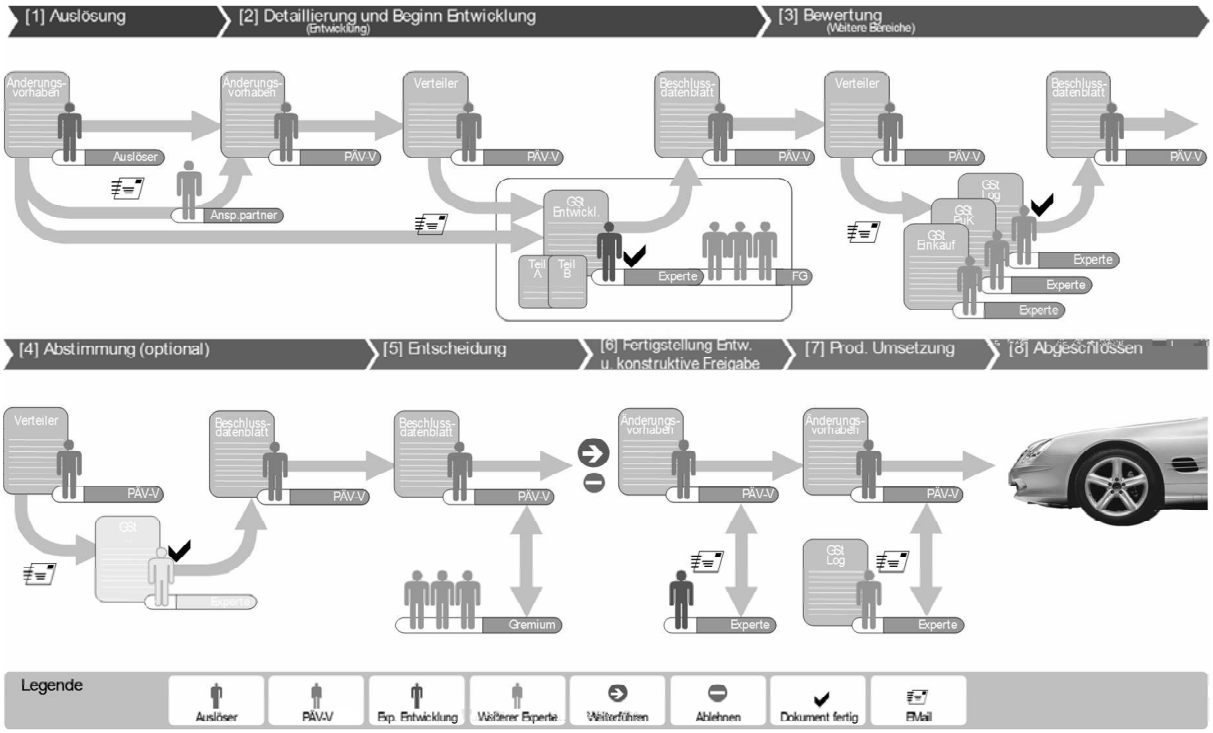


Abbildung 3.2.: Fachprozess: Produktänderungsmanagement in der Automobilindustrie [Dai10]

Phase 2. Anschließend bestimmt dieser PÄV-V geeignete Experten für alle von der Änderung betroffenen Bauteile zur Abgabe einer Stellungnahme für das PÄV. Weitere Experten werden, abhängig von ihren Tätigkeitsbereichen, zur Bewertung des PÄV herangezogen (Phase 3), etwa Experten aus dem Einkauf oder der Logistik. Nach Ablauf der Rückmeldefrist erhält der PÄV-V eine Auflistung aller Stellungnahmen, erstellt daraus eine Gesamtstellungnahme für die Entwicklung und prüft diese auf Vollständigkeit. Dies ist erforderlich, um Informationen zu Kosten, zeitlichen Rahmenbedingungen und baulichen Umfängen zu erhalten sowie durch entsprechende Gremien bewerten zu können. In Phase 4 findet eine optionale Abstimmung statt, die es ermöglicht, nach Einsicht der anderen Stellungnahmen die eigene Meinung final festzulegen. Wurden durch alle Experten Stellungnahmen zurückgemeldet, wird das PÄV zur Entscheidung vorgelegt (Phase 5). Das Ergebnis der Entscheidung wird in einem Beschlussdatenblatt zusammengefasst und einem weiteren Gremium zur Genehmigung vorgelegt. Wird das PÄV abgelehnt, endet der Änderungsprozess an dieser Stelle. Nach erfolgter Genehmigung werden eine konstruktive Umsetzung sowie eine Freigabe im Produktdokumentationssystem durchgeführt (Phase 6), bevor die Änderung direkt im Fahrzeug bzw. als Prototyp realisiert wird (Phase 7).

Dieser Fachprozess wurde durch den Fachbereich modelliert und als Dokument an den IT-Bereich übergeben. Letztgenannter muss diesen Fachprozess interpretieren und soweit detaillieren, dass ein technisch ausführbarer Prozess entsteht, der durch eine Prozess-Engine gesteuert werden kann. Aufgrund der Komplexität des dadurch entstandenen ausführbaren Prozesses, wird in Abbildung 3.3 lediglich ein Ausschnitt gezeigt.

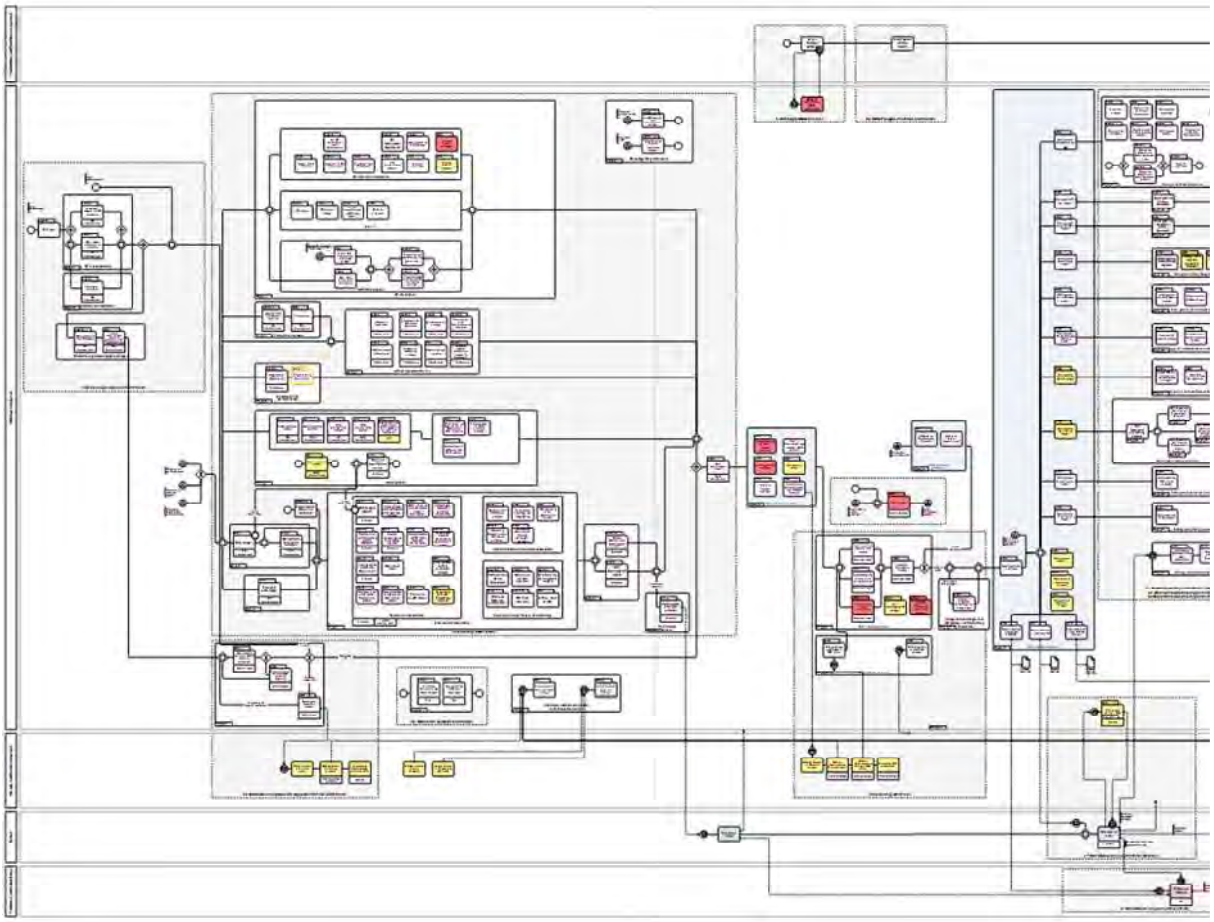


Abbildung 3.3.: Ausschnitt des ausführbaren Prozesses zum Produktänderungsmanagement in der Automobilindustrie (dokumentiert als BPMN-Diagramm)

Fazit: Aufgrund der zahlreichen ausführbaren Prozessaktivitäten (hunderte), ist eine Zuordnung zwischen Fachprozessaktivitäten (siehe Abbildung 3.2) und Aktivitäten des ausführbaren Prozesses (siehe Abbildung 3.3) nicht mehr erkennbar. Dadurch können spätere Änderungen im Fachprozess oder im ausführbaren Prozess nur schwer umgesetzt und abgestimmt werden. Einerseits werden Fachprozessaktivitäten in vielen Fällen in mehrere ausführbare Prozessaktivitäten verfeinert oder es kommen zusätzliche Aktivitäten im ausführbaren Prozess hinzu. Andererseits tauchen Fachprozessaktivitäten, für die keine IT-Unterstützung erfolgen soll, im ausführbaren Prozess nicht mehr auf. Erschwerend kommt hinzu, dass Fach- und IT-Bereiche unterschiedliche Granularitätsstufen zur Beschreibung ihrer Prozesse aufweisen und dazu verschiedene Beschreibungsformen wählen. Die initiale Übergabe von Fachprozessen in den IT-Bereich ist durch keine Methodik definiert, wodurch insbesondere der Umgang mit späteren Änderungen am Fachprozess bzw. am technischen Prozess nicht trivial ist, da betroffene Objekte im korrespondierenden Modell nicht unmittelbar identifizierbar sind. Problematisch ist insbesondere die Tatsache, dass Umstrukturierungen im Fachprozessmodell notwendig werden, um einen adäquaten und technisch umsetzbaren Systemprozess auf technischer Ebene zu erhalten.

Anwendungsszenario II: Werkstattprozesse

Werkstattprozesse beschreiben Abläufe, die im Falle einer Fahrzeugreparatur oder eines Kundendienstes in der Werkstatt durchgeführt werden müssen [HBR09, HBR10a]. Die Abläufe sind meist standardisiert und werden durch einen Fachbereich definiert und systemseitig implementiert. Das im Folgenden vorgestellte Anwendungsszenario beschreibt ein Projekt, welches einen abstrakten Werkstattprozess (vgl. Abbildung 3.4) durch eine Prozessapplikation realisieren soll.

Abbildung 3.4.: Abstrakte Beschreibung eines Prozesses aus der Werkstatt

Zentrale Anforderungen an das Projekt waren die Entwicklung eines Vorgehens zur Umsetzung von Werkstattprozessen sowie eine vollautomatische Generierung ausführbarer Prozesse, basierend auf abstrakt beschriebenen Werkstattprozessen. Ziel war es, für zukünftig entwickelte Fahrzeuge und Baureihen, eine einfache und werkzeuggestützte Generierung erforderlicher technischer Werkstattprozesse zu realisieren. Dies erwies sich als schwierig, weshalb ein solches Vorgehen nach kurzer Zeit verworfen wurde. Die neue Zielsetzung bestand anschließend darin, lediglich bestimmte Einzelfunktionen des zu entwickelnden IT-Systems automatisch zu generieren, jedoch nicht den gesamten technischen Prozessablauf. Das dazu verwendete Vorgehen sieht sechs Phasen vor (vgl. Abbildung 3.5).

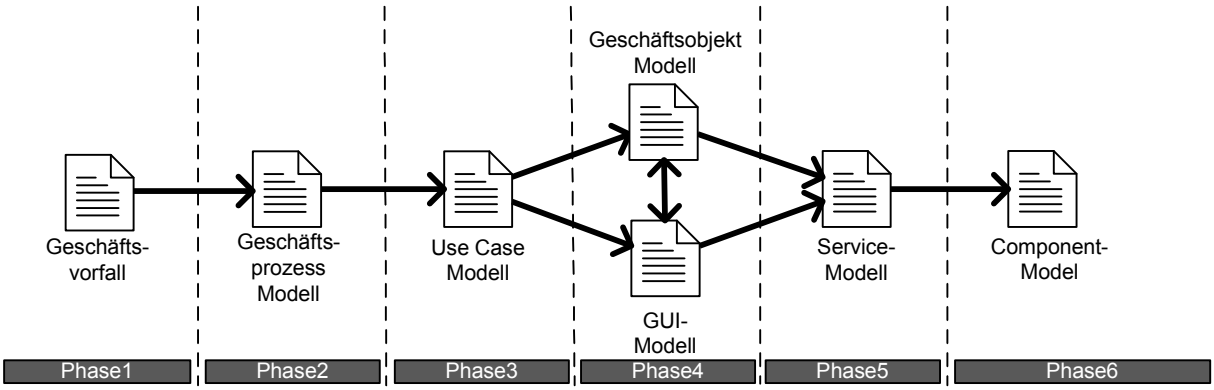


Abbildung 3.5.: Vorgehen zur Implementierung eines Werkstattprozesses

Geschäftsvorfall (Phase 1): Geschäftsvorfälle beschreiben spezifische Abläufe in einer Werkstatt, etwa das Auffüllen des Spritzwassers durch einen Mechaniker oder das Anlegen eines Neukunden in der Kundendatenbank. Ziel dieser Vorfallbeschreibung ist es, möglichst alle Abläufe in der Werkstatt zu erfassen.

Geschäftsprozessmodell (Phase 2): Basierend auf diesen Vorfällen werden anschließend abstrakte Fachprozessmodelle erstellt. Lediglich die Geschäftsvorfälle, die durch eine technische Umsetzung realisiert werden sollen, werden über mehrere Detaillierungsstufen verfeinert (vgl. Abbildung 3.6). Diese Detaillierung wird durch einen Fachmodellierer realisiert (als eEPK), welcher die zusätzlich zum Prozessablauf notwendigen Datenobjekte (z.B. *Kunde*) beschreibt.

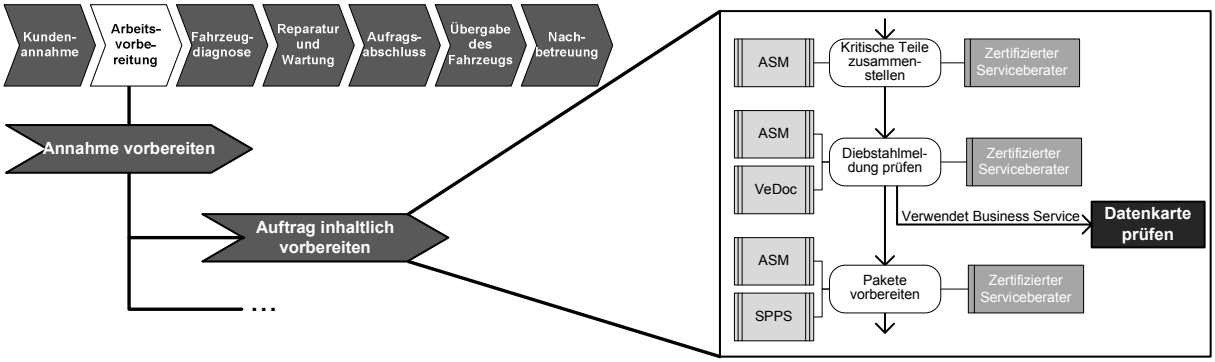
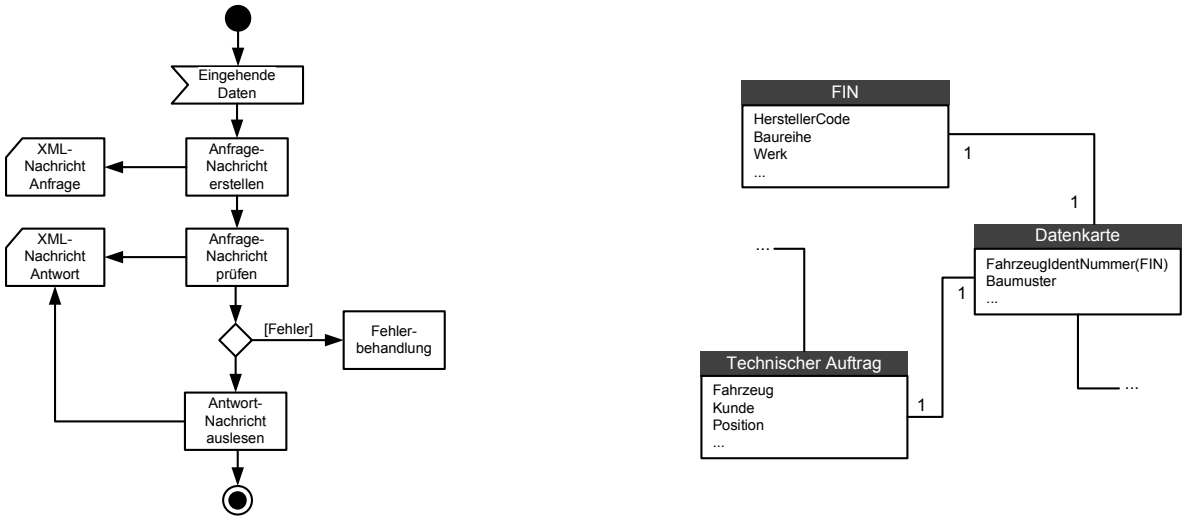


Abbildung 3.6.: Detaillierung des Fachprozesses in Phase 2

Use-Case Modell (Phase 3): Die zuvor beschriebenen eEPKs enthalten noch keine technischen Details. Deshalb werden in Phase 3 alle detaillierten Geschäftsvorfälle in mehrere Use-Case Modelle zerlegt und technisch verfeinert (vgl. Abbildung 3.7a).



a) Verfeinerung des Use-Case - Datenkarte ermitteln

b) Auszug aus dem Geschäftsobjektmodell

Abbildung 3.7.: Beispiel für die Detaillierung eines Use Cases

Geschäftsobjekt- und GUI-Modell (Phase 4): In Phase 4 werden alle für die Implementierung der Use-Cases relevanten Datenobjekte (vgl. Abbildung 3.7b) spezifiziert, und es werden zugehörige Eingabemasken entworfen.

Service-Modell (Phase 5): Alle als Service implementierbaren Use-Cases werden in dieser Phase spezifiziert, d.h. es werden Service-Eigenschaften wie Name, Beschreibung und Operationen dokumentiert.

Component-Modell (Phase 6): In der letzten Phase werden aus Use-Case-Modellen, inklusive der dort definierten Geschäftsobjekte und Service-Modelle, abstrakte technische IT-Objekte generiert. Diese können dann als Basis für die Implementierung verwendet werden.

Fazit: Die in diesem Projekt implementierten Werkstattprozesse werden durch keine gesamtheitliche Prozesssteuerung realisiert. Dies liegt daran, dass lediglich Einzelfragmente des abstrakt beschriebenen Fachprozesses durch EPKs und Use-Cases verfeinert und technisch umgesetzt werden. Die Beziehung zwischen diesen Fragmenten geht verloren, wodurch eine Prozessorientierung und -steuerung auf technischer Ebene nicht mehr realisierbar ist.

Aufgrund der schrittweisen Verfeinerung ist für einzelne Geschäftsvorfälle eine Nachvollziehbarkeit gegeben. Dadurch kann bei Änderungen am abstrakten Fachprozess identifiziert werden, welche Use-Cases und EPKs betroffen sind und ggf. angepasst werden müssen. Voraussetzung dafür ist, dass die Änderung an den Geschäftsvorfällen durchgeführt wird, die auch technisch verfeinert wurden (vgl. Phase 2).

Anwendungsszenario III: Verkäuferarbeitsplatz (Point of Sales)

Im Rahmen dieses IT-Projektes sollen für den Vertrieb alle Kundenprozesse in einer zentralen Portallösung für den *Point of Sales* gebündelt werden. Diese Prozesse beschäftigen sich mit der Beschaffung von Fahrzeuginformationen, der Konfiguration von Neufahrzeugen, der Suche von Gebrauchtfahrzeugen sowie der Fahrzeugfinanzierung. Ziel des Projektes ist es, die genannten Prozesse in einer zentralen Prozessapplikation zu realisieren. Die Basis für die Implementierung bildet die von Automobilherstellern eingesetzte *PAI PI Plattform* (vgl. Abschnitt 3.1.3) sowie derder hier verwendeten WebSphere Process Server Software [IBM08c], die eine technische Ausführung von Prozessen und eine Orchestrierung von Web-Services ermöglicht.

Um dieses Projekt zu realisieren, wurde ein Vorgehensmodell spezifiziert, welches zunächst eine Beschreibung der Kundenprozesse in ARIS vorschlägt, bevor diese über eine Detaillierung in UML verfeinert werden. Im Folgenden wird das in diesem Projekt verwendete Vorgehensmodell vorgestellt (vgl. Abbildung 3.8).

Fachprozessmodellierung als EPK (Schritt 1): Aufgrund der Vielzahl bereits zuvor als EPK modellierter Kundenprozesse, erfolgt die Beschreibung der Fachprozesse im ARIS Business Architect [Sch01]. Diese Art der Beschreibung ist in vielen Fachbereichen bekannt, wodurch Aufwände für die Einarbeitung und Kosten für Schulungen gering gehalten werden. Alle Fachprozessmodelle werden in der internen ARIS-Datenbank gespeichert.

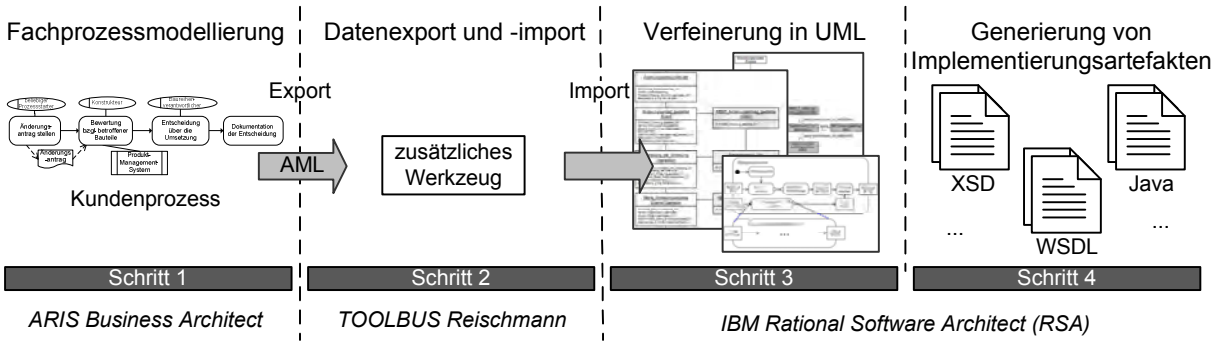


Abbildung 3.8.: Vorgehensmodell zur Implementierung eines Verkäuferarbeitsplatz

Datenexport und -import (Schritt 2): Im ARIS Business Architect modellierte Prozessmodelle können auf Grundlage der ARIS Markup Language (AML) [MN04] exportiert werden. AML ist eine XML-basierte Sprache zur Serialisierung von in ARIS modellierten Fachprozessen. Diese Sprache wird eingesetzt, um EPK-Modelle für andere Werkzeuge bereitzustellen. Als Hilfsmittel dient ein Werkzeug, das die exportierten AML-Dateien in UML-Diagramme (Activity Diagrams und Use Case Diagrams [JRH+04]) überführt. Dies wird durch spezifische Transformationen realisiert, d.h. für jeden verwendeten Objekttyp im Fachprozess ist klar definiert, in welchen Objekttyp eines UML-Diagramms er zu konvertieren ist. Beispielsweise wird das EPK-Objekt *Business Object* in UML als *Klasse* realisiert.

Verfeinerung in UML (Schritt 3): Nachdem die Prozessmodelle aus ARIS in einem Case-Tool verfügbar gemacht worden sind, können sie detailliert werden. Dabei werden Services technisch spezifiziert oder Klassen durch Attribute und Beziehungen zu anderen Klassen detailliert. Zusätzlich dazu werden sog. *UML Traces* dokumentiert, um Auswirkungen bei späteren Änderungen identifizieren zu können. So wird die ObjektID jedes ARIS-Objektes im korrespondierenden UML-Aktivitätendiagramm als Attribut vermerkt. Dadurch können Änderungen am UML-Diagramm bis zum entsprechenden Objekt im Fachprozess nachvollzogen werden.

Generierung von Implementierungsartefakten (Schritt 4): In Schritt 4 werden verschiedene Implementierungsartefakte generiert, etwa WSDL- und XSD-Beschreibungen oder Java-Code.

Fazit: Diese Methodik beschreibt, wie aus Fachprozessen durch mehrfache Verfeinerung unterschiedlicher UML-Diagramme, Implementierungsartefakte abgeleitet werden können. Dadurch wird die Nachvollziehbarkeit zwischen Fachprozess und einzelnen technischen Fragmenten erreicht. So werden etwa IDs aus ARIS-Modellen (EPKs) in Attribute von UML-Diagrammen überführt, um die Beziehung zwischen fachlichen und technischen Artefakten zu speichern. Das eingangs erwähnte Ziel, Kundenprozesse durch eine zentrale prozessorientierte IT-Lösung zu realisieren, wird jedoch nicht erreicht, da lediglich einzelne Implementierungsartefakte als Ergebnis weiterverwendet werden können. Die eigentliche Prozesslogik ist auf technischer Ebene nicht mehr vorhanden, weshalb eine prozessorientierte Steuerung des Gesamtprozesses durch ein

Prozess-Management-System nicht unmittelbar möglich ist. Insbesondere diese Einschränkung gestaltet die geforderte Implementierung auf dem WebSphere Process Server aufwendig, da der dazu notwendige technische Prozess (Service-Orchestrierung, BPEL) von Hand neu spezifiziert und erstellt werden muss.

3.1.2. Zentrale Vorgaben

Durch zentrale Vorgaben wird sichergestellt, dass Anwendungssysteme nach bestimmten Richtlinien erstellt werden. Dies ist wichtig, um die Qualität der entwickelten Software zu garantieren und durch Vereinheitlichung des Vorgehens und der eingesetzten Basistechnologien eine Kostenreduzierung zu erzielen. Die im Folgenden vorgestellten Anwendungsszenarien erörtern unterschiedliche Vorgaben zur Entwicklung von Anwendungssystemen.

Anwendungsszenario IV: Vorgehensweise für die Systemgestaltung und den Systembetrieb

Das Handbuch für Systemgestaltung und -betrieb [Dai07] beschreibt eine standardisierte Vorgehensweise und Methodik zur Gestaltung von Anwendungssystemen. Konkret umfasst dies die Vorgehensweise bei der Abwicklung von Projekten in der Softwareentwicklung sowie die Überführung dieser in den IT-Betrieb. Zielsetzung ist es, einheitlich bzgl. des Vorgehens und der Dokumentation zu realisierender Projekte zu sein. Zudem sollen unternehmensweit gültige IT-Standards, Richtlinien und Konventionen zum Einsatz kommen sowie bereits definierte Software-Artefakte bei der Entwicklung (wieder-) verwendet werden. Abbildung 3.9 beschreibt das Vorgehen zur Systemgestaltung. Die einzelnen Schritte werden nachfolgend erörtert.

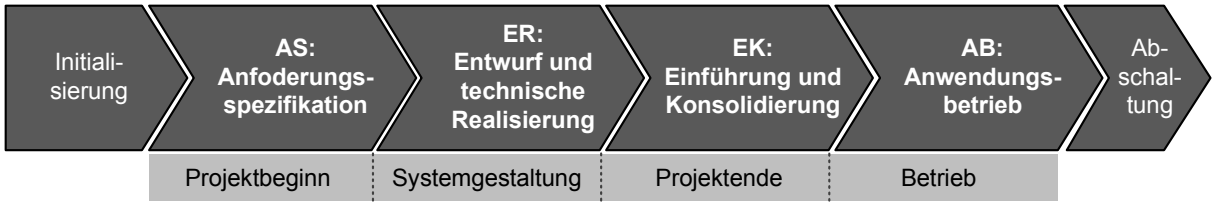


Abbildung 3.9.: Vorgehen zur Systemgestaltung (aus [Dai07])

Anforderungsspezifikation (AS): Die Erstellung des Lasten- bzw. Pflichtenhefts sowie die Entwicklung von Lösungsalternativen sind wichtige Aufgaben dieser Phase. Fachliche und technische (nicht-funktionale) Anforderungen werden aus Sicht des Anwendungssystems beschrieben, um eine Detaillierung des fachlichen Leistungsumfangs zu erhalten. Letztgenannter besteht aus einer Vielzahl unterschiedlicher Spezifikationsdokumente, etwa organisatorischen Rahmenbedingungen, Soll-Geschäftsprozessmodell, Zugriffs- und Berechtigungskonzepten, System-Qualitätszielen, Benutzeroberflächendesign und Teststrategien (vgl. Abbildung 3.10). Für alle weiteren Phasen des Vorgehens existiert ebenfalls eine detaillierte Beschreibung(vgl. Abbildung 3.10).

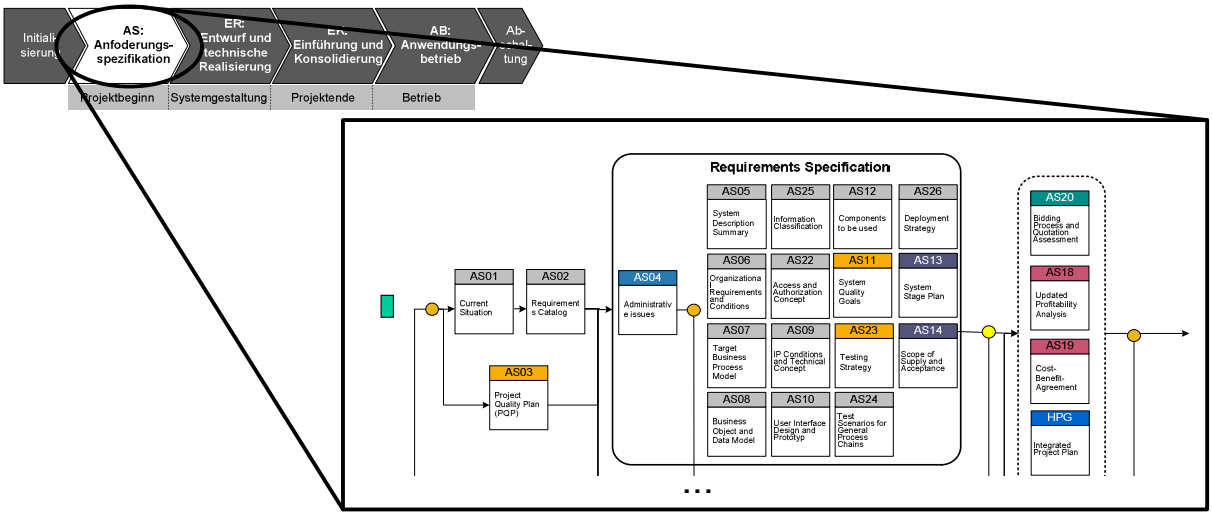


Abbildung 3.10.: Detaillierung der Phase Anforderungsspezifikation (aus [Dai07])

Entwurf und technische Realisierung (ER): Die Phase des Entwurfs und der technischen Realisierung behandelt primär die Umsetzung des Pflichtenhefts in ein Anwendungssystem. Der technische Entwurf und die Implementierung beinhalten z.B. die System- und Anwendungsarchitektur sowie das logische Objekt- und Datenmodell. Neben der Implementierung und des Systemtests wird in dieser Phase die Systemdokumentation erstellt.

Einführung und Konsolidierung (EK): Die Einführung und Konsolidierung umfasst die technische und organisatorische Implementierung des zu realisierenden Anwendungssystems. Nach erfolgreicher Übergabe des Anwendungssystems an den Fachbereich, wird der Projektabschluss durchgeführt.

Anwendungsbetrieb (AB): Der Anwendungsbetrieb hat zum Ziel, das realisierte Anwendungssystem dem Nutzer bereitzustellen. Letzteres wird durch *Service-Level-Agreements (SLA)* im Pflichtenheft festgehalten.

Fazit: Jede einzelne Phase detailliert Spezifikationsdokumente und Richtlinien, die bei der klassischen Softwareentwicklung eingehalten werden müssen. Service-orientierte Architekturen (SOA) und die Entwicklung service- und prozessorientierter Applikationen werden hier jedoch nicht berücksichtigt. Des Weiteren existieren in diesem Anwendungsszenario keine Vorgaben darüber, dass ein Fachprozess oder ein ausführbarer Prozess (bspw. Service-Orchestrierung) vollständig grafisch modelliert und strukturiert werden soll, so dass ein Prozess-Management-System diesen steuern kann.

Das Vorgehen beschreibt jedoch auch Beziehungen zwischen unterschiedlichen Artefakten, etwa die Beziehungen zwischen Anforderungen und konkreten technischen Implementierungsartefakten. So kann vereinzelt Nachvollziehbarkeit dokumentiert werden, diese wird jedoch nicht für Fachprozessesaktivitäten und deren Überführung in Systemprozessaktivitäten gefordert.

Anwendungsszenario V: Technische Repository-Vorgabe

Anwendungsszenario V beschreibt eine technische Vorgabe für ein Service-Repository. Es wurde in einem IT-Projekt entwickelt, um einen zentralen Informationsspeicher für Services anbieten zu können. Neben einer zentralisierten Speicherung von Services, sind die Suche nach Services sowie deren Entwicklung wichtige Ziele. Um diese zu realisieren, muss zunächst der Aufbau des Service-Repositories vereinheitlicht werden. Abbildung 3.11 skizziert das in diesem Projekt konzipierte Metamodell eines Service-Repositories [Dai09a]. Hier werden lediglich Objekte und Beziehungen visualisiert, auf eine Darstellung der Attribute wird verzichtet.

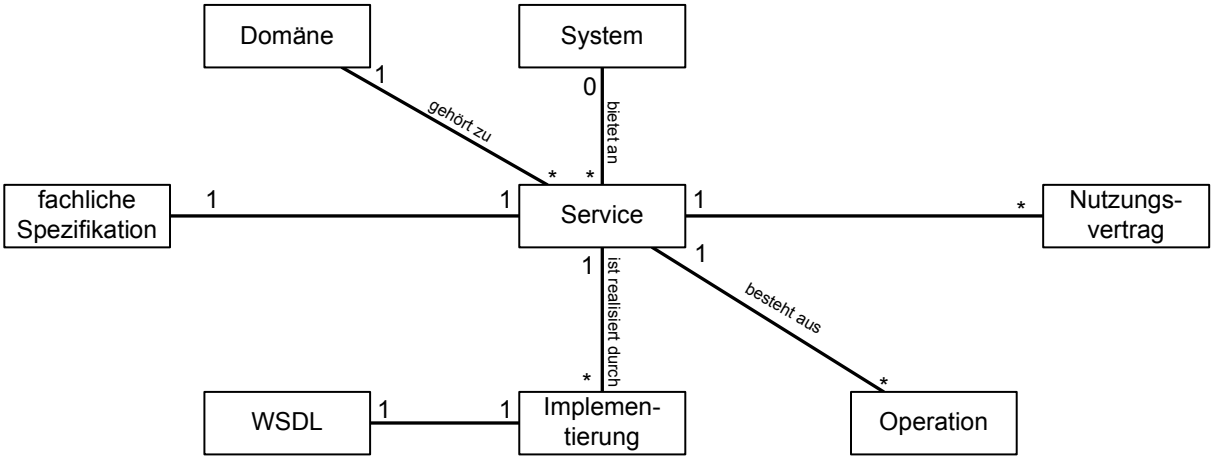


Abbildung 3.11.: Vorschlag eines technischen Repository (aus [Dai09a])

Zentrales Objekt dieses Metamodells ist der *Service*. Er gehört zu einer bestimmten *Domäne* und wird durch ein *System* (z.B. Produktdaten-Management-System) angeboten. Seine Nutzung wird durch einen *Nutzungsvertrag* definiert, der z.B. Angaben über Gültigkeitszeitraum, Betriebszeiten und Service-Kosten dokumentiert. Die Umsetzung eines Services erfolgt durch technische *Implementierung*, die basierend auf einer *fachlichen Spezifikation* realisiert wird. Das Ergebnis ist eine *WSDL*-Datei mit mehreren *Service-Operationen*.

Fazit: Das Metamodell ermöglicht die Beschreibung technischer Services und zugehöriger Artefakte (z.B. Nutzungsverträge). Für einen Service können mehrere Nutzungsverträge existieren, die im Repository dokumentiert werden. Ein Vertrag wird zwischen Service-Konsumenten und -Anbietern geschlossen. Da kein eigenes Objekt für Service-Konsumenten und -Anbieter im Repository existiert, muss die Information im Nutzungsvertrag dokumentiert werden (etwa als zusätzliches Attribut). Des Weiteren kann nicht dokumentiert werden, welches System welchen Service nutzt oder welche Geschäftprozesse die Service-Entwicklung beeinflusst haben. Dieser Vorschlag liefert einen ersten Entwurf für ein technisches Repository. Erkannt wurde, dass eine fachliche Sichtweise notwendig ist, um beispielsweise Fachprozesse oder einzelne Use-Cases bei der Service-Entwicklung berücksichtigen und dokumentieren zu können. Diese fachliche Sichtweise ist hier jedoch nur schwach ausgeprägt (nur ein Objekt im Metamodell vorhanden).

Anwendungsszenario VI: Fachliche Repository-Methodik

Die fachliche Repository-Methodik [Dai09b] macht Vorgaben zur Service-Entwicklung. Ziel ist es, eine im zentralen IT-Bereich verankerte Methodik zu definieren, welche die Service-Entwicklung in allen IT-Projekten unternehmensweit standardisiert und steuert. Dazu müssen zunächst die fachlichen Eigenschaften eines Services beschrieben werden, bevor die fachliche Schnittstelle des Services, d.h. seine Funktionalität und Nutzungsmöglichkeiten, detailliert werden. Die fachliche Service-Beschreibung und die Service-Registrierung sind die ersten Schritte bei der Service-Entwicklung. Abbildung 3.12 skizziert das methodische Vorgehen.

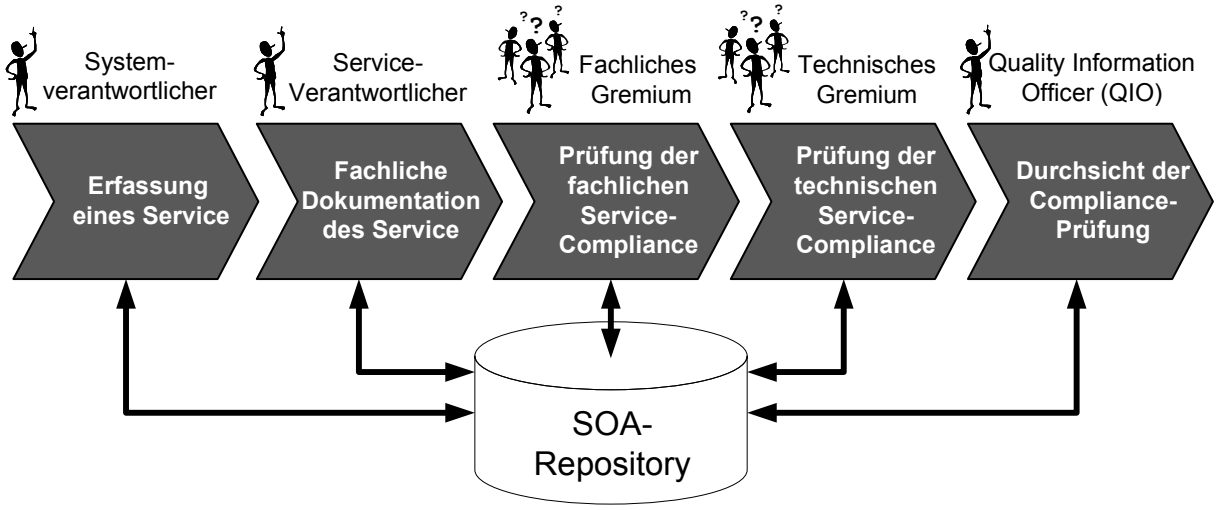


Abbildung 3.12.: Vorschlag eines technischen Repository (angelehnt an [Dai09b])

Erfassung eines Service: Die Entscheidung, ob ein Service durch ein System angeboten werden soll, wird durch den Systemverantwortlichen getroffen. Dieser erfasst die Basisdaten zum Service und benennt einen Service-Verantwortlichen.

Fachliche Dokumentation des Services: Der Service-Verantwortliche detailliert den Service fachlich. Diese Dokumentation besteht aus der Einbettung des Services in die Domänen- und Architekturlandschaft einerseits und der Beschreibung von Abhängigkeiten zu anderen Services oder Systemen andererseits. Außerdem werden Service-Operationen nach bestimmten Richtlinien (Service-Guidelines) benannt (z.B. *getProductInformation* oder *setServiceResponsible*) und mit einer ausführlichen fachlichen Beschreibung versehen, welche die Art der Verwendung oder den Zweck der Service-Operation dokumentiert. Anschließend werden alle Fachobjekte beschrieben, deren Verständnis für die Verwendung des Services Voraussetzung sind (z.B. Ein- und Ausgabeparameter von Service-Operationen). Die fachliche Beschreibung eines Service dient als Grundlage für eine spätere technische Verfeinerung und Implementierung. Zunächst muss diese jedoch auf Vollständigkeit geprüft werden.

Prüfung der fachlichen Service-Compliance: Bevor ein fachlicher Service implementiert werden kann, ist eine Compliance-Prüfung durch Experten aus dem Fachbereich notwendig. Sie entscheiden, ob der fachliche Service in der beschriebenen Form realisiert werden soll. Maßgeblich für diese Entscheidung sind die vollständige Dokumentation der fachlichen Services nach definierten Richtlinien, ein Finanzierungs- und Nutzungsplan für den fachlichen Service sowie die Einordnung in die Unternehmensstrategie. Beschließen die Experten aus dem Fachbereich die technische Realisierung des fachlichen Services, wird die fachliche Dokumentation des Services als Basis für die Erstellung der technischen Service-Spezifikation verwendet.

Prüfung der technischen Service-Compliance: Die technische Prüfung findet nach der Spezifikation des technischen Services statt. Ein Gremium, bestehend aus IT-Experten, entscheidet darüber, ob die technische Spezifikation den vom Unternehmen vorgegebenen technischen Richtlinien entspricht. Experten aus dem IT-Bereich entscheiden darüber, ob der geplante Service für die Implementierung freigegeben werden kann.

Durchsicht der Compliance-Prüfung: Der *Quality Information Officer (QIO)* prüft einerseits, ob eine fachliche und technische Compliance-Prüfung durchgeführt worden ist, andererseits beleuchtet er alle vorliegenden Artefakte (z.B. Fachspezifikation, Entwicklungshandbuch, etc.) aus neutraler Sicht. Diese zusätzliche Prüfung ermöglicht es, Compliance-Verletzungen aufzudecken und darauf zu reagieren. Anschließend wird der Service freigegeben und durch den IT-Bereich implementiert.

Fazit: Es hat sich gezeigt, dass fachliche Aspekte bei der Service-Entwicklung wichtig sind. Sie sollten deshalb zusätzlich zu den technischen Aspekten (vgl. Anwendungsszenario V) im Service-Repository dokumentiert werden. Darauf basierend können z.B. Prüfungen auf Vollständigkeit der Spezifikationen durchgeführt werden. Eine explizite Betrachtung technischer Services ist in diesem Anwendungsszenario lediglich auf eine Phase (Prüfung der technischen Compliance) beschränkt. Dies erklärt sich dadurch, dass die Entwicklung der in diesem Anwendungsszenario skizzierten Vorgehensmethodik (vgl. Abbildung 3.12) durch den Fachbereich geprägt ist. Eine technische Betrachtung von Services und ihre Verwaltung durch ein Repository, wie etwa in Anwendungsszenario V, wird durch die Vorgehensmethodik nicht unterstützt.

3.1.3. Bereitstellung einer IT-Infrastruktur

Eine IT-Infrastruktur legt die Basis für die effiziente Durchführung von IT-Projekten. Die im Folgenden vorgestellten Anwendungsszenarien zeigen, wie IT-Infrastrukturkomponenten (z.B. ein Service-Repository) bei der Entwicklung von Prozessapplikationen eingesetzt werden.

Anwendungsszenario VII: Metamodellerweiterung für ein Service-Repository

Ein Service-Repository dient zur Speicherung und Verwaltung angebotener Services. Ziel des nachfolgend vorgestellten Anwendungsszenarios ist es, alle Services eines bestimmten Unternehmensbereichs zentral zu dokumentieren und für autorisierte Nutzer zugreifbar zu machen. Als

Basis für die Implementierung in diesem Anwendungsszenario wird die Infrastrukturkomponente *WebSphere Registry and Repository (WSRR)* verwendet [DRS+07]. Da das zugrundeliegende Metamodell für die in diesem Szenario definierten Anforderungen nicht ausreichend ist, muss es angepasst werden. Das daraus resultierende Repository dokumentiert primär technische Schnittstellenbeschreibungen, Service-Operationen, Ein- und Ausgabeparameter sowie nicht-technische Aspekte (z.B. Verfügbarkeit des Service, Laufzeit oder Ansprechpartner).

Fazit: Das in diesem Projekt umgesetzte Service-Repository dient primär zur Dokumentation technischer Services. Fachliche Aspekte, etwa eine fachliche Beschreibung des Services wie im Anwendungsszenario IV (vgl. Abschnitt 3.1.2), sind nicht vorgesehen. Demzufolge können Änderungen an Services nur schwer dem zugehörigen Fachbereich oder den jeweiligen Fachprozessen zugeordnet werden. Dadurch geht die Nachvollziehbarkeit zwischen technischen und fachlichen Services verloren, was insbesondere bei Änderungen entsprechende Anpassungen davon betroffener Services und Prozesse schwierig gestaltet.

Anwendungsszenario VIII: Repository als Shared Service

In diesem Anwendungsszenario wird ein Service-Repository beschrieben, das in einer Multi-Tenancy-Architektur [MLP08] betrieben ist und von allen in der zugrundeliegenden IT-Infrastruktur existierenden Komponenten genutzt werden kann. Ziel dieses Projekts ist es, eine zentrale Repository-Lösung für die Unternehmens-IT zu realisieren. So sollen alle im Konzern durchgeführten IT-Projekte das Repository direkt verwenden.

Als Basis-Repository wird WSRR eingesetzt. WSRR besitzt ein Metamodell, das jedoch für die in diesem Projekt geforderte Funktionalität nicht ausreichend ist. Insbesondere die Anbindung an existierende Unternehmenskomponenten kann nicht realisiert werden. Zudem ist die Mächtigkeit des implementierten Metamodells nicht ausreichend genug, um fachliche Informationen aus IT-Projekten speichern zu können. Deshalb muss ein erweitertes Metamodell entwickelt werden, das sowohl fachliche als auch technische Artefakte dokumentiert. Zudem muss die Anbindung an existierende Infrastrukturkomponenten, Anwendungssysteme und IT-Projekte möglich sein (vgl. Abbildung 3.13).

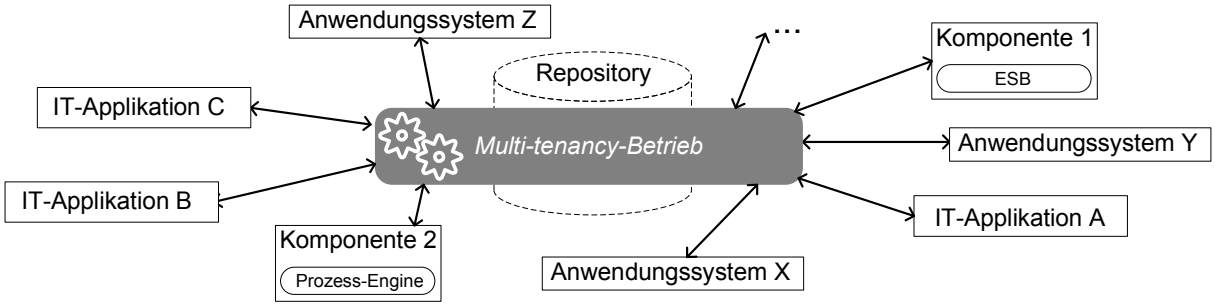


Abbildung 3.13.: Repository Shared Service

Das in diesem Anwendungsszenario beschriebene Projekt wurde aufgrund von Schwierigkeiten bei der Entwicklung des notwendigen Metamodells vorzeitig eingestellt.

Fazit: Neben der reinen Informationsspeicherung, müssen existierende Infrastrukturkomponenten, etwa ein *Enterprise Service Bus* [Cha04, KAB+04] oder ein *Message Broker* [Men07], das Repository verwenden können. Nur dann lassen sich Services entkoppelt von der eigentlichen Anwendungslogik aufrufen oder bereits implementierte Funktionalität weiterverwenden. Dies erfordert die Entwicklung eines unternehmensweit einheitlichen Metamodells für ein Service-Repository, das in einer Multi-Tenancy-Architektur angeboten werden kann und die benötigte Funktionalität abdeckt, d.h. welche die Anbindung an alle IT-Applikationen und Komponenten im Unternehmen realisiert (vgl. Abbildung 3.13). Nur wenn ein solches Metamodell existiert, können Services so dokumentiert werden, dass eine Verwendung durch andere Services, Komponenten oder IT-Applikationen möglich wird.

Anwendungsszenario IX: ProActive Infrastructure

Die *ProActive Infrastructure (PAI)* [Sch09] beschreibt eine Sammlung unternehmensweiter Software-Plattformen, auf denen (als einheitliche Basis) Applikationen realisiert werden können. Ziel ist es, eine unternehmensweite Basis zur Umsetzung von IT-Projekten zu etablieren. Dies wird unterstützt, indem PAI-Plattformen mit unterschiedlicher Funktionalität entwickelt werden. Im Folgenden werden die wichtigsten Plattformen skizziert [Sch09]:

PAI Java 2 Enterprise Edition (J2EE) Plattform: Basierend auf der *PAI J2EE Plattform* findet die Entwicklung von J2EE-kompatiblen Anwendungen statt. Dies sind z.B. Enterprise Java Beans, Web Applikationen (JSP, JSF) oder Eclipse RCP basierte Application Clients. Der WebSphere Application Server [AHP+09], HTTP Server [BB05] und UDB [IBM11] sowie Eclipse sind Komponenten, die bei der Plattformauslieferung bereits vorinstalliert und vorkonfiguriert sind und deshalb einheitlich verwendet und betrieben werden.

PAI Process Integration (PI) Plattform: Die PAI PI Platform [Sch09] bietet eine Grundlage zur Integration und Entwicklung service- und prozessorientierter Applikationen. Die Plattform ermöglicht die Definition, Implementierung und Ausführung von BPEL-Prozessen basierend auf der Service Component Architecture (SCA) und dem Service Data Objects (SDO) Programmiermodell. Durch Verwendung des WebSphere Process Server [IBM08c], des WebSphere Integration Developer [Pet05] und der PAI J2EE Plattform, ermöglicht die PI-Plattform eine service-orientierte Entwicklung von Prozessapplikationen.

PAI Business Information Broker (BIB) Plattform: Die BIB-Plattform [Sch09] bietet eine Lösung zur Nachrichten-basierten Integration von Applikationen. Die Plattform basiert auf WebSphere Message Broker [DCGP05], WebSphere MQ [IBM05] und UDB. Die BIB-Plattform ermöglicht die Integration von Anwendungen durch klare Trennung der Integrations- von der Geschäftslogik. Dies wird einerseits durch grafische Modellierung unterstützt, andererseits durch umfassende Datentransformationsmöglichkeiten sowie verschiedene Kommunikationsprotokolle. Zudem bietet die BIB-Plattform einen ESB, der eine lose gekoppelte Architektur ermöglicht und Funktionen wie dynamisches Nachrichten-Routing oder Service-Bindung bietet.

Die Plattformen können miteinander kombiniert werden, wodurch die nutzbare Funktionalität

in der Anwendungsentwicklung erhöht wird. Die vorgestellten Plattformen J2EE, PI und BIB sind auch im Kontext der vorliegenden Arbeit interessant, da sie in Kombination die Entwicklung service- und prozessorientierter Applikationen sowie die Integration in eine bestehende IT-Infrastruktur ermöglichen (vgl. Abbildung 3.14). Die BIB-Plattform dient als Kommunikationseinheit zwischen der J2EE- und PI-Plattform, indem sie das Routing von Nachrichten sowie die dazu notwendigen Datentransformationen realisiert. Die J2EE-Plattform dient einerseits der Umsetzung von Einzelapplikationen und andererseits der Spezifikation und Realisierung von Services, die durch die Einzelapplikationen konsumiert werden. Die PI-Plattform orchestriert die Services und ermöglicht deren service- und prozessorientierte Ausführung.

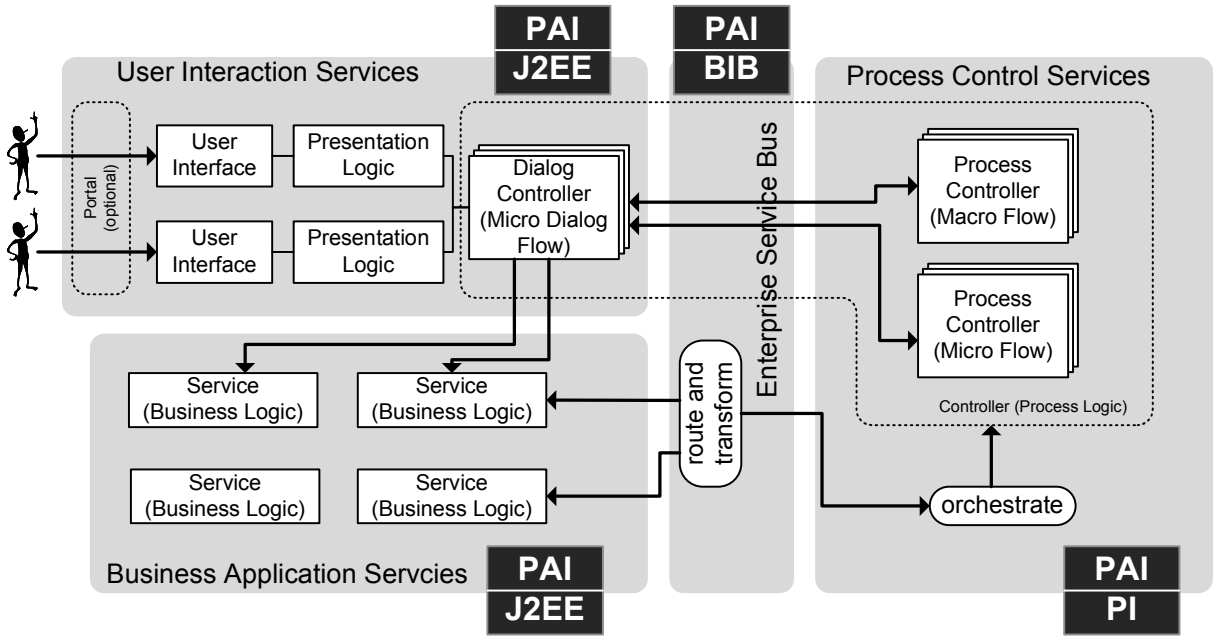


Abbildung 3.14.: Kombination von PAI-Plattformen für eine SOA (aus [Sch09])

Fazit: Die strategische Plattformentwicklung ist bei Automobilherstellern weit fortgeschritten, was den Einsatz unterschiedlicher Plattformen in IT-Projekten ermöglicht. Was hier jedoch außer acht gelassen wurde, ist einerseits die fachliche Prozessmodellierung und andererseits die Berücksichtigung fachlicher SOA-Aspekte, etwa die durch den Fachbereich definierten Vorgaben. Des Weiteren fehlt eine durchgängige Methodik, die beschreibt, wie fachliche Anforderungen konkret durch eine IT-Implementierung umgesetzt werden sollen. Dies ist schwierig, wenn fachliche Anforderungen entsprechende Änderungen des Anwendungssystem erfordern, die Beziehung zwischen fachlichen Anforderungen und ihrer Implementierung aber nicht dokumentiert ist. Was hingegen gut dokumentiert ist, sind Vorgaben dazu, in welcher Form die Plattformen und die dort verfügbaren Produkte zur service- und prozessorientierten Entwicklung von Applikationen eingesetzt werden sollen. Erkannt wurde zudem, dass eine einheitliche IT-Infrastruktur inklusive einer Prozesssteuerung (Prozess-Engine), eines Enterprise Service Bus (ESB) und eines Nachrichten-Routing (Message Broker) notwendig ist und dass die Entwicklung von Prozessapplikationen durch entsprechende Methoden und Werkzeuge unterstützt werden muss.

3.1.4. Diskussion

In Abschnitt 3.1 wird eine Reihe von Anwendungsszenarien aus der Automobilindustrie analysiert. Mit ihrer Hilfe konnten mehrere in der Praxis vorkommende Anforderungen und Lösungsansätze identifiziert werden.

Aus den Anwendungsszenarien wird klar, dass nicht die technische IT-Infrastruktur zu Problemen in der Umsetzung service- und prozessorientierter Applikationen führt, sondern vor allem die fehlende Methodik und die zur Umsetzung notwendigen Konzepte. Es existiert eine Vielzahl an Ansätzen, die eine Prozessausführung durch eine Orchestrierungs-Engine (z.B. WebSphere Process Server [IBM08c]) realisieren oder technische Services (z.B. WebSphere Registry and Repository [DRS+07]) speichern. Der Großteil davon ist anpassbar und erweiterbar. Eine Anpassung kann jedoch nur dann durchgeführt werden, wenn bekannt ist, nach welcher Methodik eine service- und prozessorientierte Anwendung entwickelt werden soll und wie ein Repository-Metamodell zur Speicherung von Services und deren Beziehungen zu Anwendungen aussehen muss. Die beschriebenen Anwendungsszenarien zeigen, dass eine solche Methodik fehlt. Insbesondere fällt die Lücke zwischen dem Fach- und IT-Bereich auf, etwa bei der Übergabe von Fachprozessmodellen in den IT-Bereich. Zudem wurden in den Anwendungsszenarien VII und VIII partiell nutzbare Ansätze für die Realisierung eines Service-Repository identifiziert, jedoch sind diese nicht umfangreich genug, um fachliche und technische Services sowie ihre Beziehungen untereinander vollständig zu dokumentieren. Die Analyse der vorangehenden Anwendungsszenarien ergibt folgende Erkenntnisse:

- **Erkenntnis A (Methodik zur Entwicklung service- und prozessorientierter Applikationen):** Eine Methodik zur Entwicklung service- und prozessorientierter Applikationen ist für mehrere Anwendungsszenarien erforderlich. So wird in Anwendungsszenario IV (vgl. Abschnitt 3.1.2) zwar ein Handbuch zur Systemgestaltung beschrieben und als zentrale Vorgehensmethodik für die Software-Entwicklung eingesetzt, jedoch ohne Berücksichtigung von Prozess- und Service-Aspekten. Insbesondere Anwendungen, die durch eine Prozess-Engine ausgeführt werden, bleiben ebenso unberücksichtigt wie das Management von Services und ihren Lebenszyklen. Bestätigt wird dies durch Anwendungsszenario V (vgl. Abschnitt 3.1.2), in welchem Vorgaben zur Dokumentation technischer Services unter Verwendung eines Service-Repository existieren, jedoch fachliche Aspekte (fachliche Services oder Prozesse) nicht berücksichtigt werden. Die Anwendungsszenarien aus der betrieblichen Praxis ergaben keine zufriedenstellenden Ergebnisse hinsichtlich eines zentralen Vorgehensmodells zur Entwicklung flexibler service- und prozessorientierter Applikationen.
- **Erkenntnis B (Konzepte zur durchgängigen Modellierung):** Eine durchgängige Modellierung, beginnend beim Fachprozess, über dessen schrittweise Verfeinerung, bis hin zur Spezifikation und Orchestrierung ausführbarer Prozessmodelle, wird in den zuvor beschriebenen Anwendungsszenarien nicht realisiert. Genausowenig betrachtet wird die Nachvollziehbarkeit der Auswirkungen von Änderungen an Fachprozessaktivitäten bzw. Systemprozessaktivitäten. Die in Anwendungsszenario II dokumentierten Werkstattprozesse (vgl. Abschnitt 3.1.1) werden mittels schrittweiser Verfeinerung detailliert. Dabei werden einzelne Prozessfragmente durch Use-Case Diagramme und EPKs detailliert, bevor sie an den IT-Bereich zur Implementierung weitergegeben werden. Die schrittweise Verfeinerung sorgt dafür, dass eine nachvollziehbare Dokumentation der Beziehungen zwischen Fachprozess-

und Systemprozessaktivitäten realisiert wird. Da lediglich Teile des Fachprozesses auf diese Weise verfeinert werden, geht die Gesamtablaufstruktur des ausführbaren Prozesses verloren, weshalb eine unmittelbare Ausführung durch eine Prozess-Engine nicht möglich ist. Das Fehlen adäquater Konzepte zur Realisierung einer durchgängigen Überführung von Fachprozessen in ausführbare Prozesse führt dazu, dass lediglich Teilaspekte des Gesamtprozesses technisch umgestetzt werden.

- **Erkenntnis C (Bereitstellung eines zentralen Repository):** Neben einer Methodik zur Entwicklung service- und prozessorientierter Applikationen sowie der dazu erforderlichen IT-Konzepte zur durchgängigen Modellierung, muss eine entsprechende IT-Infrastruktur existieren, die in der Lage ist, solche Konzepte umsetzen zu können. So muss ein zentrales Service-Repository bereitgestellt werden, das alle geforderten fachlichen und technischen Services, inklusive deren Eigenschaften, verwaltet. Dazu verwendet Anwendungsszenario VII (vgl. Abschnitt 3.1.3) eine Metamodellerweiterung des *WebSphere Registry and Repository* mit dem Schwerpunkt auf der Speicherung technischer Services. Fachliche Service-Aspekte sowie die Abhängigkeiten zwischen fachlichen und technischen Repository-Artefakten bleiben ausgeklammert. Anwendungsszenario VIII (vgl. Abschnitt 3.1.3) zeigt die Notwendigkeit eines unternehmensweiten zentralen Service-Repository.
- **Erkenntnis D (Vereinheitlichung):** Eine weitere Erkenntnis aus den betrachteten Anwendungsszenarien ist die Notwendigkeit einer Vereinheitlichung der IT-Infrastruktur und der von ihr betriebenen Komponenten, etwa der Einsatz eines Enterprise Service Bus oder einer einheitlichen Prozess-Engine. Anwendungsszenario IX liefert dazu eine Infrastrukturplattform, welche eine Reihe von Komponenten beinhaltet und bereits in mehreren IT-Projekten eingesetzt wird. Die jeweiligen Plattformen sind jedoch technisch orientiert, wodurch eine Beschreibung fachlicher Services oder von Fachprozessen meist nicht möglich ist (vgl. Erkenntnis A).

3.2. Stand der Technik

Wie die beschriebenen Anwendungsszenarien zeigen, gibt es bei der Entwicklung service- und prozessorientierter Applikationen eine Vielzahl offener Fragestellungen. Neben einem Vorgehensmodell zur Entwicklung von Anwendungssystemen in einer SOA, werfen die flexible Anpassung von Services und Fachprozessen sowie der Betrieb eines zentralen SOA-Repository zur Dokumentation fachlicher und technischer SOA-Artefakte wichtige Fragestellungen auf. Basierend auf den in Abschnitt 3.1 beschriebenen Anwendungsszenarien sowie der in diesem Abschnitt diskutierten wissenschaftlichen Ansätze, werden in Abschnitt 3.3 Anforderungen an das zu entwickelnde ENPROSO-Konzept abgeleitet.

[Erl05] führt Grundlagen service-orientierter Architekturen ein, beschreibt Service-Prinzipien (z.B. Service-Wiederverwendung) und gibt eine Übersicht über WS*-Standards. Weitere Arbeiten von Erl fokussieren auf bestimmte Aspekte einer SOA [EKW+09, Erl09][1]. In [Erl09] etwa werden sog. *SOA Design Patterns* beschrieben. Beispielhaft sei an dieser Stelle das *Metadata Centralization*-Pattern genannt, das die Verwendung eines Service-Repository voraussetzt. Hier

[1] Weitere Bücher von Erl sind unter http://www.soabooks.com zu finden.

sollen sowohl existierende als auch sich in Entwicklung befindliche Services (inkl. deren Funktionalität) dokumentiert werden. Eine konkrete Beschreibung, wie ein Metamodell für eine solches Repository aussehen muss, wird nicht gegeben. Weitere Pattern schlagen die Verwendung eines *Enterprise Service Bus* vor oder regeln die Service-Versionierung. Die definierten Patterns liefern eine Basis, um die Entwicklung und den Betrieb von Services besser zu gestalten. Was jedoch nur am Rande betrachtet wird, ist die gesamtheitliche Vorgehensmethodik, um in einer SOA Prozessapplikationen entwickeln und betreiben zu können. So wird nicht beschrieben, wie Fachprozesse in konkrete ausführbare Prozesse überführt werden und wie letztere durch eine Prozess-Engine ausgeführt werden. Darüber hinaus werden Abhängigkeiten zwischen Fach- und Systemprozessen nicht explizit dokumentiert. Folglich sind deren Änderungen nur schwer nachvollziehbar.

[Jos07] beschreibt sowohl fundamentale Konzepte einer SOA als auch praktische Details. Letztere sind hilfreich bei der Realisierung einer SOA. Zahlreiche technische Aspekte, etwa die Verwendung eines *Enterprise Service Bus*, werden erkannt und in den Kontext einer service-orientierten Architektur gesetzt. Weitere Betrachtungsgegenstände betreffen das Design und die Einführung eines Service-Repository sowie die Relevanz eines (Repository-) Metamodells bei der Realisierung einer SOA. Empirische Untersuchungen zeigen jedoch, dass eine Service-Wiederverwendung in der Praxis nur selten realisierbar ist. [Jos07] beschreibt nach einer Analyse mehrerer SOA-Projekte, bei denen nach ein bis zwei Jahren pro Service lediglich ein Nutzer existiert und nach drei bis fünf Jahren die Zahl der Nutzer auf lediglich zwei bis vier ansteigt. Methodische Fragestellungen werden hier nur am Rande betrachtet, lediglich eine abstrakte schrittweise Einführung einer SOA wird skizziert.

[Heu07] gibt einen Überblick über Prinzipien service-orientierter Architekturen und detailliert diese anhand verschiedener Fallbeispiele. Dies inkludiert einen Erfahrungsbericht zur Identifikation von Nutzenpotentialen sowie Einführung einer SOA im Unternehmen. Ein einheitliches und strukturiertes Vorgehensmodell für die Entwicklung service- und prozessorientierter Applikationen wird nicht deutlich. Die Notwendigkeit eines Service-Repository wird angesprochen, jedoch ohne einen tatsächlichen Vorschlag für ein Repository-Metamodell zu machen.

[Lie07] liefert eine Einführung in technische Grundlagen einer SOA. Es werden SOA-Referenzmodelle beschrieben und verglichen. Sehr fundiert wird das Zusammenspiel von SOA mit anderen Konzepten, etwa Portalen und Integrationsarchitekturen (Enterprise Application Integration, CORBA) diskutiert. Ein weiterer Schwerpunkt liegt auf der Integration bereits existierender Alt-Applikationen und deren Migration. Letztere beschreibt eine vollständige oder teilweise Ablösung der Alt-Applikationen durch Services. Durch die eher technische Ausrichtung der in [Lie07] beschriebenen Konzepte wird die Gestaltung fachlicher Services oder Fachprozesse vernachlässigt. Darüber hinaus fehlt eine Dokumentation der Beziehungen zwischen Artefakten aus Fachprozessen und deren Implementierung. Eine durchgängige Gesamtmethodik, die beschreibt, wie fachliche Services und Fachprozesse in service- und prozessorientierten Applikationen realisiert werden können, wird nicht dokumentiert.

[Pap08] gibt einen technischen Überblick zu Service-orientierten Architekturen. Der Fokus liegt auf Themen wie Web-Services, Middleware-Technologien, SOAP, UDDI, ESB und BPEL. Aufgrund der starken Fokussierung auf Web-Services, ist das Gesamtbild einer SOA schwer erkennbar, weshalb eine durchgängige Modellierung von Fachprozessen ebenso wenig dokumentiert wird, wie eine entsprechende Methodik. Ähnlich bietet auch [Mel10] nur eine sehr technische Sichtweise auf eine SOA.

[ST07] liefert einen breiten Überblick über eine SOA, der aber technisch wenig fundiert ist. So fehlt eine Behandlung von ESB und Orchestrierungswerkzeugen, ebenso wie eine Auseinandersetzung mit BPEL, BPMN oder anderen Modellierungs- und Ausführungsaspekten für Services sowie Prozessen.

Weitere Arbeiten, etwa von [Mas07], beschäftigen sich mit einem Vorgehensmodell zur Entwicklung von Software mittels Services, das mit *Service Oriented System Engineering (SOSE)* bezeichnet wird, d.h. der Fokus liegt auf dem Software-Entwicklungsprozess. Der Vorgehensmodell sieht eine Dreiteilung der Methodik vor:

- Abstrakte Beschreibungen von Unternehmensabläufen sind ebenso Bestandteil des **Geschäftsmodells** wie die Spezifikation von Erwartungen an den zu entwickelnden Service bzw. die service-orientierte Applikation.
- Das **Servicemodell** legt, neben der notwendigen Semantik von Services, die Service-Level-Agreements sowie Nutzungsverträge zwischen Service-Anbieter und -Konsument fest.
- Das **Implementierungsmodell** steht für die technische Beschreibung des Services bzw. die Implementierung service-orientierter Applikationen.

Die in [Mas07] vorgestelle Methodik beschreibt die Entwicklung service-orientierter Applikationen. Was jedoch fehlt, ist eine prozessorientierte Sichtweise, die auf technischer Ebene eine Service-Komposition und -Orchestrierung ermöglicht. Darüber hinaus ist nicht klar definiert, welches Abstraktionsniveau für ein Geschäfts-, Service- oder Implementierungsmodell notwendig ist und was jeweils konkrete Inhalte sind. Die Notwendigkeit eines zentralen Service-Repository wird erkannt, jedoch lediglich für die Dokumentation technischer Service-Informationen und nicht für fachliche Artefakte (z.B. Verantwortlichkeiten oder fachliche Anforderungen an Services).

Das OrVia-Projekt [OrV09] befasst sich mit der Realisierung einer robusten und zuverlässigen Architektur zur Orchestrierung geschäftskritischer E-Business-Umgebungen. OrVia nutzt dazu eine teilautomatische Abbildung von Fachprozessen auf ausführbare Prozesse. Dazu werden Patterns eingesetzt. Abbildung 3.15 zeigt die OrVia-Architektur. Die Vorgehensmethodik bzw. eine dazugehörige Arbeitspaketstruktur des Projektes werden in [KTS05, KKT06] beschrieben. [SKD+08] evaluiert die in [KKT06] beschriebene Vorgehensweise anhand einer Fallstudie aus dem E-Government Bereich. Weitere Details zum OrVia-Projekt, etwa zur Erweiterung von Fachprozessen durch domänenspezifische Regeln, sind in [FF08, PD08] zu finden.

OrVia liefert eine Architektur zur Orchestrierung service-orientierter Umgebungen. Was nicht betrachtet wird, ist die Verwendung eines zentralen SOA-Repository.

[Off08] beschäftigt sich ebenfalls mit einer Methodik zur Entwicklung einer Service-orientierten Architektur und bezeichnet das entsprechende mehrstufige Vorgehen als *Service-Oriented Architecture Method (SOAM)*: Ausgehend von der Unternehmensanalyse, in deren Rahmen fachliche Anforderungen in Fachprozesse abgebildet werden, werden Service-Operationen abgeleitet. Abbildung 3.16 zeigt die einzelnen Phasen:

- **Unternehmensanalyse:** Die erste Phase umfasst die Aufnahme von Anforderungen an das zu entwickelnde Anwendungssystem. Ferner werden auf Grundlage eines sog. *Business Motivation Model* [OMG06], abstrakte Geschäftsfunktionalität und -strategien festgehalten, bevor konkrete Fachprozessmodelle erstellt werden. Die Fachprozesse müssen soweit

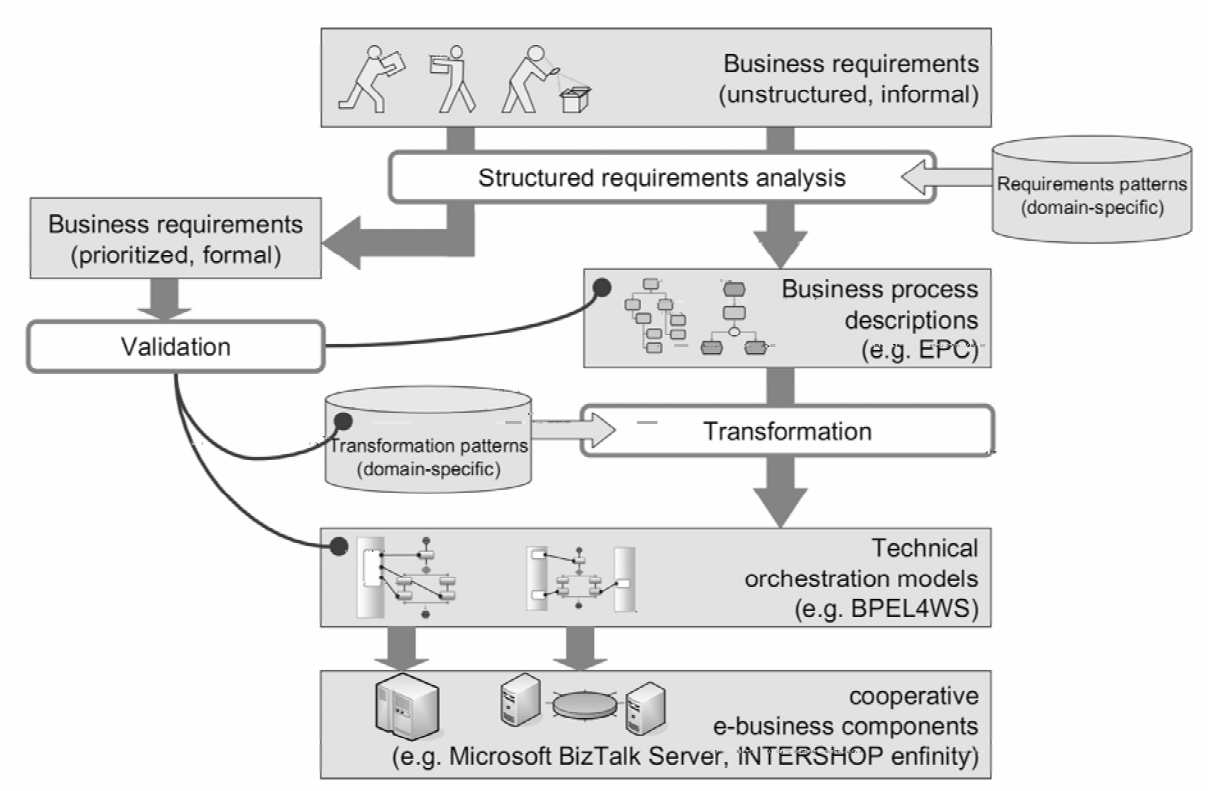


Abbildung 3.15.: Architektur zur Orchistrierung und Validation in OrVia (aus [KTS05])

verfeinert werden, dass jede Fachprozessaktivität genau einer technischen Funktion (z.B. Service-Operation) zugeordnet werden kann.

- **Identifikation von Service-Operationen:** In dieser Phase werden Services bzw. konkrete Service-Operationen aus den zuvor modellierten Fachprozessen abgeleitet. Anwendungsfalldiagramme [OMG05] helfen bei der Beschreibung dieser Services.
- **Bestandssystemanalyse:** Zusätzlich zur bisherigen (Top-Down) Vorgehensweise bzgl. Ableitung von Services, werden auch bereits existierende Systeme betrachtet und Schnittstellen identifiziert (Bottom-Up), welche die beschriebenen Fachprozesse unterstützen und zugleich als Services bereitgestellt werden können.
- **Konsolidierung:** Die Konsolidierung beschreibt die Vereinigung der Top-Down und Bottom-Up Analyse, indem Service-Operationen und Datenobjekte derselben Art redundanzfrei zusammengefasst werden.
- **Service-Design:** Im Anschluß an die Konsolidierung werden Service-Operationen zu Services zusammengefasst und auf Autonomie bzw. Zustandslosigkeit geprüft.
- **Prozessaufbereitung:** Services werden nach ihrer Spezifikation implementiert. Analog dazu werden in dieser Phase auch die technischen Prozessmodelle definiert.

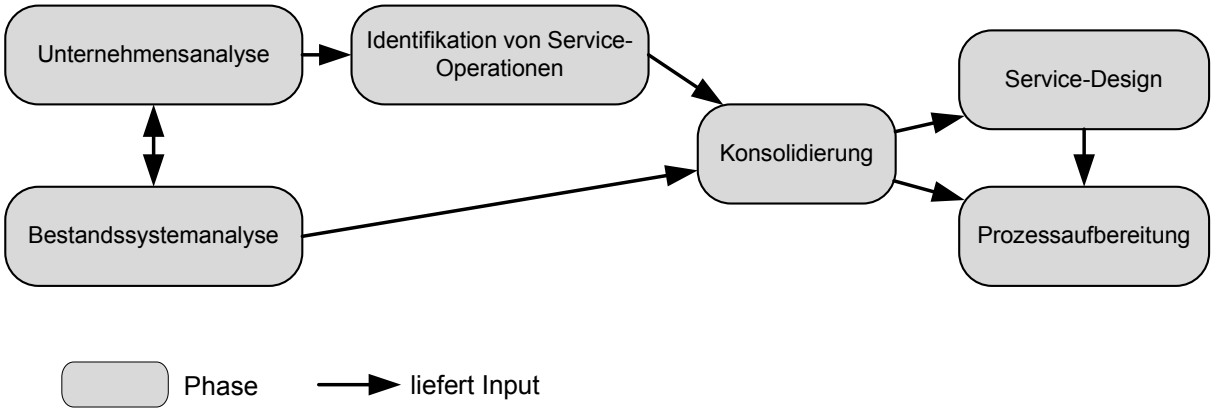


Abbildung 3.16.: Phasen der Service-Oriented Architecture Method (aus [Off08])

SOAM liefert eine umfangreiche Methodik zur Entwicklung einer Service-orientierten Architektur. Die Anwendungsszenarien aus Abschnitt 3.1 zeigen jedoch, dass eine durchgängige Modellierung in der Praxis meist nicht vorhanden ist. So ist die in der ersten Phase (Unternehmensanalyse) beschriebene Verfeinerung der Fachprozesse nicht trivial, weshalb ENPROSO einen Ansatz mit zusätzlicher Modellierungsebene (vgl. Systemmodell in Abschnitt 2.2.5) zwischen Fach- und IT-Bereich anstrebt. Darüber hinaus wird die Verwendung eines SOA-Repository nicht explizit beschrieben, obwohl dies bei der Ermittlung bereits existierender Funktionalität (Phase: Bestandssystemanalyse) helfen würde.

[SI07, Ste08a, Ste08b, SR08] stellen einen Ansatz zur Beschreibung fachlicher Services aus Sicht von Fachmodellierern und Geschäftsexperten vor. Die fachliche Beschreibung eines Services wird nicht als dessen technische Realisierung gesehen, sondern vielmehr als Erwartung an einen Service. In diesem Ansatz wird ein Service durch seine Eigenschaften bzgl. Fähigkeiten erbrachter Leistungen, Ausführungsdauer, maximale Antwortzeit und die Kosten pro Aufruf eines Services charakterisiert. [Ste08b] führt ein fachliches Repository ein, welches die zuvor beschriebenen Services verwaltet und im Rahmen der Fachprozessmodellierung verfügbar macht. Dieses Repository verwaltet primär fachliche Services; technische Aspekte, etwa konkrete Service-Schnittstellen oder Abhängigkeiten zu anderen technischen Services, werden nicht dokumentiert. Dadurch sind Auswirkungen von Änderungen an fachlichen Services nur schwer nachvollziehbar.

Die SOA-Entwicklungsmethode *Service-Oriented Modeling & Architecture (SOMA)* [AGA+08, Ars04] wird durch das Werkzeug Rational Software Architect [BDE+05] unterstützt und zur Entwicklung von Software eingesetzt. SOMA bietet eine integrierte Entwurfs- und Entwicklungsunterstützung für die modellgetriebene Software-Entwicklung basierend auf einer Geschäfts- und IT-Sicht [FWGH09]. In der Geschäftssicht werden Fachprozesse modelliert und Services abstrakt spezifiziert, bevor sie durch den IT-Bereich in der IT-Sicht realisiert werden. Abbildung 3.17 zeigt die unterschiedlichen Phasen, die bei Anwendung der SOMA-Methodik während der Software-Entwicklung durchlaufen werden [AGA+08]. Die Phasen *Business modeling and transformation*, *Solution management*, *Identification* und *Specification* werden durch den Fachbereich bewerkstelligt, die restlichen Phasen durch den IT-Bereich.

SOMA ist eine umfangreiche Methodik zur Entwicklung von Anwendungen in einer SOA. Auf-

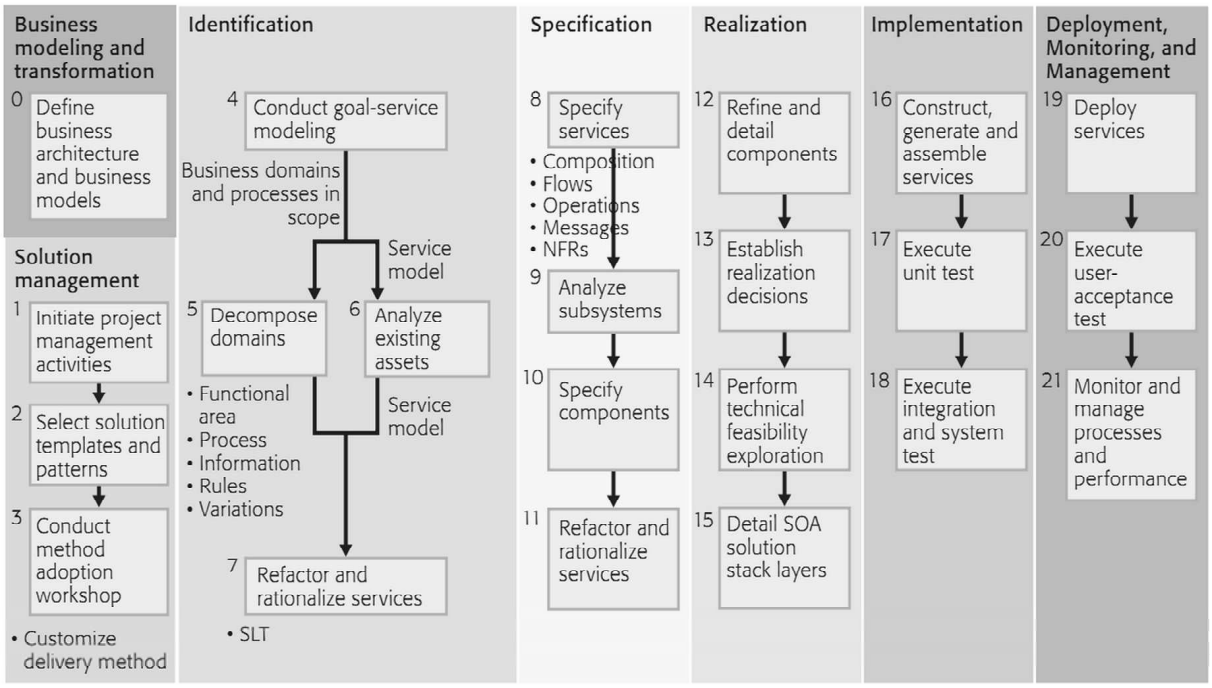


Abbildung 3.17.: Phasen der SOMA-Methodik (aus [AGA+08])

grund der abstrakten Vorgehensbeschreibung ist die Realisierung technischer Details, etwa die Orchestrierung von technischen Services nur sehr aufwendig oder mit beratender Unterstützung möglich. Ein SOA-Repository wird zwar erwähnt, ist jedoch ohne Spezifikation eines Metamodells nicht einsetzbar.

Eine in der Praxis gängige Vorgehensweise ist *Quasar Enterprise* [EHH+08]. Die Gestaltung service-orientierter Anwendungslandschaften wird durch Methoden, Regeln, Muster und Referenzarchitekturen unterstützt. Quasar Enterprise definiert eine Landkarte zur strategischen Umsetzung service-orientierter IT-Projekte. Die Basis hierfür ist das *Integrated Architecture Framework (IAF)* [MM06], das die Reihenfolge der Aufgaben bei der Umsetzung eines IT-Projektes vorgibt. Abbildung 3.18 zeigt die Bestandteile von Quasar Enterprise. Die Quasar Enterprise Methode besteht aus fünf Phasen, die beim Entwurf einer SOA durchlaufen werden. Die Analyse der *Geschäftsarchitektur* dient dazu, ein Verständnis für die Unternehmensstrategie und -ziele zu gewinnen sowie einen hohen Grad an Service-Wiederverwendung zu ermöglichen [HHV06]. Eine *ideale Anwendungslandschaft* dient als Vorgabe für das Architekturmanagment. Dadurch wird die *Integration* von Komponenten in die Anwendungslandschaft vereinfacht und zugleich die Grundlage zur Auswahl von *Integrationsplattformen* (bspw. ein Enterprise Service Bus) gelegt. Nachdem der Ist-Zustand der Geschäftsarchitektur analysiert und dokumentiert ist, wird die Soll-Architektur erstellt (*Evolution*).

ADONIS ist ein Fachprozessmodellierungswerkzeug zur modellgetriebenen Entwicklung von prozessorientierten Applikationen [MM03, Bro04]. Schwerpunkte bilden die Integration von Benutzerschnittstellen, die graphische Visualisierung von Ressourcenauslastungen [BOC05], die Verknüpfung von Fachprozessen mit Daten [BOC07] und die technische Speicherung von Services in

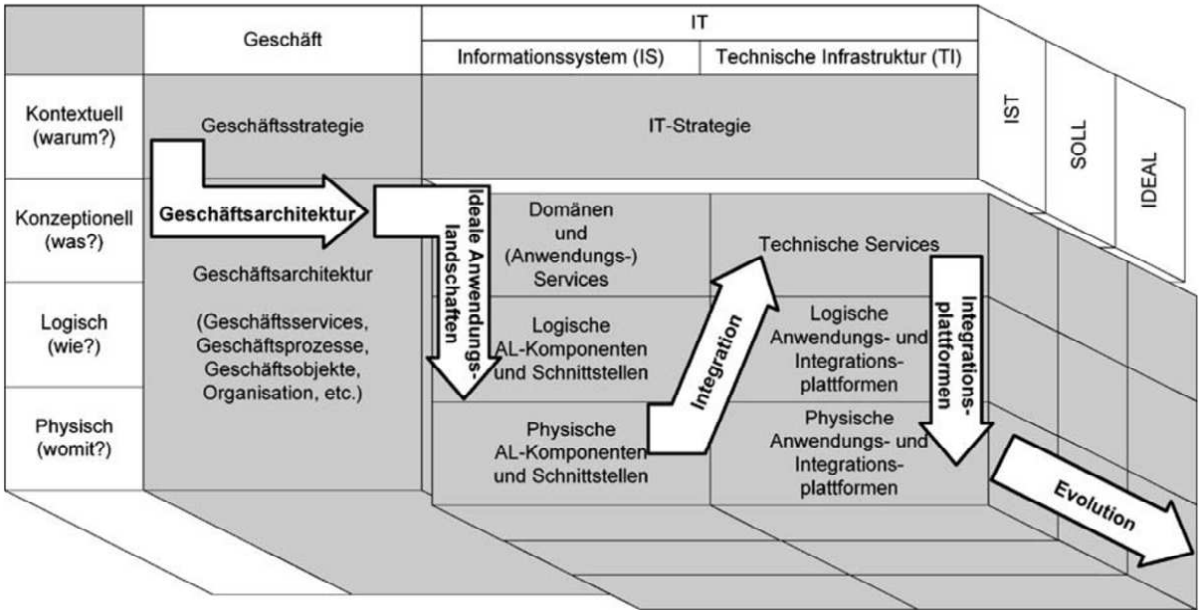


Abbildung 3.18.: Quasar Enterprise basierend auf dem IAF [AGA+08]

einem Repository [BOC06]. Nicht betrachtet wird, wie ausgehend von konkreten Fachprozessen, eine service- und prozessorientierte Applikation implementiert werden kann. Weitere Fachprozessmodellierungswerkzeuge ermöglichen eine modellgetriebene Entwicklung prozessorientierter Anwendungen [MID07, MID08a, Dee07, DPRW08, MID04, Sof09, IDS05, FFO05].

Nicht nur die in Abschnitt 3.1 identifizierten Erkenntnisse, sondern auch die diskutierte Literatur bestätigen, dass ein konkretes Vorgehensmodell zur durchgängigen Modellierung sowie Konzepte für die Entwicklung service- und prozessorientierter Applikationen in einer SOA nicht in ausreichendem Maße existieren. Auffällig ist, dass bei der bisherigen Analyse insbesondere Probleme bei dem Umgang von Änderungen entstehen. Dies erklärt sich dadurch, dass der Fach- und der IT-Bereich aufgrund unterschiedlicher Modellierungswerkzeuge und unterschiedlicher Kenntnisstände der jeweiligen verantwortlichen Modellierer nur unzureichende Möglichkeiten haben, notwendige Informationen auszutauschen. Zudem wurde die Notwendigkeit für den Einsatz eines zentralen SOA-Repository erkannt, ohne aber dass es konkrete Vorschläge gibt, wie ein solches in einer SOA umgesetzt werden soll.

3.3. Anforderungsanalyse

Basierend auf den diskutierten Erkenntnissen, werden nun funktionale Anforderungen an die Entwicklung service- und prozessorientierter Applikationen beschrieben.

Anforderung 3.1 (Business-IT-Alignment)
Ein wichtiges Ziel von Service-orientierten Architekturen ist das *Business-IT-Alignment*. Es beschreibt die Ausrichtung der Geschäftsziele an die Informationstechnologie des Unternehmens.

Aktuell stellt der Abgleich zwischen den Fach- und IT-Bereichen, insbesondere zwischen Fachprozessen und deren IT-Implementierung, eine Herausforderung für Unternehemen dar [Che08]. So sollten Fachprozesse schnell und vollständig in technische Lösungen (ausführbare Prozesse) umgesetzt werden können.

Anforderung 3.2 (Bidirektionale Nachvollziehbarkeit)
Eine bidirektionale Nachvollziehbarkeit von Beziehungen zwischen Objekten aus dem Fachmodell (z.B. Fachprozessaktivitäten) und Elementen aus dem Systemmodell (z.B. Systemprozessaktivitäten) ist eine wichtige Anforderung. Sie untergliedert sich wie folgt:

- **Transparente Dokumentation von Beziehungen:** Beziehungen zwischen unterschiedlichen Modellierungsebenen (Fachmodell, Systemmodell und ausführbares Modell) bzw. Werkzeuggrenzen müssen explizit dokumentiert werden. D.h. für alle fachlichen Objekte muss definiert sein, durch welche technischen Objekte sie realisiert werden. Nur dann lässt sich identifizieren, an welchen Stellen im korrespondierenden Modell Änderungen an Objekten Auswirkungen haben.
- **Versionsübergreifende Transparenz:** Sowohl Fach- und Systemmodelle als auch ausführbare Modelle entwickeln sich stetig weiter, wodurch neue Modell-Versionen entstehen. So führen neue oder geänderte fachliche Anforderungen zur Anpassung von Fach- und Systemprozessen. Eine versionsübergreifende Dokumentation zusammengehörender Fach- und Systemmodelle sowie ausführbarer Modelle ist notwendig, um sicherzustellen, das Änderungen im richtigen Modell (in der richtigen Modellversion) realisiert werden. Dies gilt nicht nur für die Modellierungsebenen und die dort beschriebenen Prozesse, sondern auch für alle definierten Prozess-Aspekte (z.B. Aktivitäten, Datenobjekte und Services).
- **Abhängigkeiten zwischen Applikation und Repository-Inhalten:** Ein wichtiger Aspekt einer bidirektionalen Nachvollziehbarkeit betrifft die Dokumentation der Abhängigkeiten zwischen Prozessapplikationen und Repository-Inhalten. Letztere dokumentieren bereits existierende Services oder Datenobjekte, die in den unterschiedlichen Modellierungsebenen verwendet werden. ENPROSO fordert eine explizite Dokumentation dieser Abhängigkeiten, um bei einem plötzlichen Service-Ausfall die konkreten Modellversionen identifizieren zu können, die diesen Service verwenden.

Anforderung 3.3 (Änderbarkeit)
Die Änderbarkeit von Fach- und Systemmodellen sowie ausführbaren Modellen ist zu unterstützen. Diese Modelle entwickeln sich weiter und erfordern eine entsprechende Anpassung. Letztere muss für alle betroffenen Modelle mit geringem Aufwand realisierbar sein. In einer SOA können nicht nur Änderungen an den verschiedenen Modellen, sondern auch an Services auftreten. Diese sollten ebenfalls mit geringem Arbeitsaufwand vorgenommen werden können.

- **Identifikation und Lokalisierung von Änderungen an Modellartefakten:** Um Änderungen schnell und mit geringem Aufwand zu realisieren, sind deren Identifikation und die Ermittlung ihrer Auswirkung(en) wichtig. So müssen Änderungen am Fachprozess entdeckt und konkreten Objekten im zugehörigen Systemprozess zugeordnet werden. Analog dazu müssen Änderungen am Systemprozess identifiziert und konkreten Objekten im Fachprozess zuordbar sein.

- **Adaption von Änderungen:** Nachdem eine Änderung identifiziert und lokalisiert worden ist, wird sie umgesetzt. Diese Adaption muss so realisierbar sein, dass alle betroffenen Anwendungen ohne Fehler weiter betrieben werden können.

Anforderung 3.4 (Konsistenz)
Eine zentrale Rolle bei der Entwicklung und Inbetriebnahme von Prozessapplikationen spielt die Konsistenthaltung von Fach- und Systemmodellen sowie zugehöriger Artefakte (z.B. Datenobjekte oder Services). Es muss sichergestellt werden, dass Fachmodelle durch Systemmodelle spezifiziert und freigegeben werden, bevor diese implementiert werden.

- **Konsistenzsicherung zwischen den Modellierungsebenen:** Die Konsistenz zwischen Fach- und Systemmodell muss sichergestellt werden. Dies ist auch dann zu bewerkstelligen, wenn Anpassungen an Fach- oder Systemmodellen unabhängig voneinander durchgeführt werden.
- **Konsistenz zwischen Fach- und Systemprozess:** Alle in einem Fachprozess dokumentierten Fachprozessaktivitäten werden im Systemprozess technisch beschrieben. Letzterer dient somit als Spezifikation der zu implementierenden Prozessapplikation.
- **Konsistenz zwischen Applikationen:** In einer SOA werden Services unterschiedlicher Applikationen verwendet, d.h. sie werden in unterschiedlichen Versionen konsumiert und angeboten, wodurch ein komplexes Beziehungsgeflecht zwischen Service-Anbietern und -Konsumenten entsteht. Es muss sichergestellt werden, dass durch eine Prozessapplikation konsumierte Services nicht ausfallen und zu Fehlern in der Ausführung der Prozessapplikation führen.
- **Impact-Analysen:** Um die Konsistenz zwischen unterschiedlichen Modellen oder Applikationen sicherstellen zu können, sind Analysemechanismen notwendig, die im Falle von Veränderungen eventuelle Inkonsistenzen ermitteln.
- **Modellierer-geführte Behandlung von Inkonsistenzen:** Entdeckte Inkonsistenzen zwischen Fach- und Systemmodell, Fach- und Systemprozess oder zwischen Applikationen, müssen durch Modellierer möglichst einfach behoben werden können.

Anforderung 3.5 (Weiterentwicklung von SOA-Artefakten)
In einer SOA existiert eine Vielzahl von Artefakten, etwa Services und Datenobjekte die von verschiedenen Applikationen verwendet werden, IT-Infrastrukturkomponenten und Organisationsmodelle. SOA-Artefakte sollten mit wendig Aufwand weiter entwickelt werden können:

- **Möglichkeit zur Weiterentwicklung und Abschaltung von Sevices:** Ein Service, der in einer SOA von anderen konsumiert wird, muss abschaltbar sein. Dabei dürfen keine Fehlersituationen entstehen, etwa dass eine Applikation aufgrund der Service-Abschaltung oder -Weiterentwicklung nicht mehr korrekt funktioniert.
- **Organisatorische Veränderungen:** Zentral bereitgestellte Organisationsmodelle (z.B. in einem Corporate Directory) müssen weiterentwickelt werden können, damit Änderungen an der Organisationsstruktur des Unternehmens umsetzbar sind.
- **Austausch und Weiterentwicklung von Infrastrukturkomponenten:** Eine IT-

Infrastruktur bildet die Grundlage zur Ausführung von Applikationen [BBP09]. Infrastrukturkomponten wie Applikationsserver entwickeln sich weiter oder werden durch neue Komponentenversionen ausgetauscht. Solche Veränderungen dürfen keinerlei Auswirkungen auf den Anwendugsbetrieb nach sich ziehen.

- **Vorfeldanalysen:** Durch Analysen soll die Möglichkeit entstehen, Änderungen an SOA-Artefakten im Vorfeld zu analysieren und deren Auswirkungen zu identifizieren.

Anforderung 3.6 (Wiederverwendung von SOA-Artefakten)
Die Wiederverwendung von SOA-Artefakten bezieht sich primär auf Services, Datenobjekte, Geschäftsregeln und Infrastrukturkomponenten. Je größer der Wiederverwendungsgrad dieser Artefakte ist, desto geringer sind die Aufwände für (redundante) Neuentwicklungen.

Anforderung 3.7 (Flexibilität)
Unter Flexibilität wird die Bereitstellung neuer IT-Implementierungen für Fachprozesse einerseits und die Anpassungen bestehender IT-Implementierungen als Reaktion auf Änderungen andererseits [MRB08] verstanden. Änderungen werden meist durch die Geschäftsseite ausgelöst, d.h. durch Fachprozesse, z.B. aufgrund von Optimierungen oder Änderungen gesetzlicher Rahmenbedingungen. Flexibilität bedeutet weiter, Prozesse und Services bei Veränderung von verwendeten Services, Infrastrukturkomponenten, Geschäftsregeln oder Datenobjekten funktionsfähig zu halten.

Anforderung 3.8 (Logisch zentrale Datenverwaltung)
Für die zentrale Speicherung von Daten ist es notwendig, ein unternehmensweites, einheitliches, proaktives und standardisiertes Metamodell eines SOA-Repository zu entwickeln. Dieses muss sowohl fachliche als auch technische Artefakte speichern und die notwendigen Beziehungen zwischen diesen Artefakten herstellen können.

Anforderung 3.9 (Service-orientierte IT-Infrastruktur)
Nicht nur ein zentrales SOA-Repository ist wichtig, sondern eine komplette Service-orientierte IT-Infrastruktur, die alle für einen flexiblen Betrieb von Prozessapplikationen notwendigen IT-Komponenten bereitstellt.

Im folgenden Kapitel wird deshalb beschrieben, wie eine grundlegende SOA IT-Infrastruktur aussieht.

4

Service-orientierte IT-Infrastruktur

Ein wichtiger Erfolgsfaktor für jedes Unternehmen ist seine Anpassungsfähigkeit bei Umgebungsänderungen [MRB08, Rei00, RRD04a, RR07]. Diese Fähigkeit wird zum Wettbewerbsvorteil, wenn die nötigen Anpassungen schneller und kostengünstiger realisierbar sind als bei Konkurrenten. Um diesen Vorteil zu erlangen, wird der Einsatz flexibler Informationstechnologie (IT) zunehmend zum Schlüsselfaktor. Unternehmen haben mittlerweile erkannt, dass die geforderte Anpassungsfähigkeit mit monolithischen IT-Systemen nicht realisierbar ist. Deshalb wurden im Laufe der Zeit verschiedene Technologien und Methoden in Unternehmen etabliert, mittels denen sich die Abläufe zwischen den IT-Systemen abbilden und damit deren prozessorientierte Integration realisieren lassen. Das Anwenden dieser Technologien führt zu immer komplexer werdenden Unternehmenslandschaften. Diese Komplexität ist in der Nutzung unterschiedlicher Integrationslösungen begründet, was zu intransparenten Abläufen geführt hat [RD00]. Neben der fehlenden Transparenz der Abläufe im eigenen Unternehmen, kommt erschwerend hinzu, dass auch die Beziehungen mit Partnern zunehmend komplexer werden. Um die Transparenz dieser Abläufe zu verbessern, werden diese heute losgelöst von den IT-Systemen dokumentiert, d.h. auf einer fachlichen Ebene, oftmals auch in Verbindung mit einer Prozessoptimierung. Konkret stellen diese Abläufe die fachlichen Anforderungen dar, die von den IT-Systemen zu erfüllen sind, um die Geschäftsfähigkeit des Unternehmens sicherzustellen.

Getrieben durch diese Erkenntnis versuchen Unternehmen, die vorhandene Systemlandschaft service-orientiert auszurichten [RS04, Erl05, Wes07]. Service-orientierte Architekturen (SOA) werden in der Literatur oft als Architekturparadigma verstanden (vgl. Abschnitt 2.1), das durch die Einführung von Services und dem zu ihrer Realisierung und ihrem Betrieb notwendigen Technologien und Methoden geprägt sind. Um Unternehmenslandschaften und deren IT-Systeme flexibel im Kontext einer SOA betreiben zu können, ist eine IT-Infrastruktur notwendig, deren Komponenten Funktionen zur service-orientierten Entwicklung und Inbetriebnahme von Prozessapplikationen bereitstellen. Um zu verstehen, welche Funktionen an welcher Stelle in der IT-Infrastruktur benötigt werden, werden im Folgenden einzelne Infrastrukturkomponenten und

deren Zusammenspiel im Detail betrachtet [BBP09]. Dazu werden unterschiedliche Ausbaustufen einer IT-Infrastruktur entwickelt und verschiedenen Funktionen konkreter Infrastrukturkomponenten zugeordnet. Beispielhaft hierfür stehen die Funktionen eines zentralen Informationsmanagements, das durch Verwendung eines SOA-Repository realisiert werden kann.

In Abschnitt 4.1 wird ein Anwendungsszenario vorgestellt. Dieses führt einen vereinfachten Fachprozess ein, der zur Illustration des Übergangs von „hart verdrahteten“ IT-Systemen zu Service-orientierten Architekturen dient. Abschnitt 4.2 greift dieses Anwendungsszenario bei der Beschreibung von Infrastrukturkomponenten auf, um darzulegen, weshalb eine IT-Infrastruktur die dargestellten Komponenten umfassen sollte. Die IT-Infrastruktur wird unterteilt in zwei Entwicklungsstufen, mittels denen sich existierende IT-Systeme service-orientiert weiter entwickeln lassen. Abschnitt 4.3 betrachtet den Stand der Technik, bevor Abschnitt 4.4 mit einer Zusammenfassung und einem Ausblick schließt.

4.1. Anwendungsszenario und Ziele

Abbildung 4.1 zeigt einen Prozess für das Änderungsmanagement in der Fahrzeugentwicklung. Dieser ist vereinfacht dargestellt, beinhaltet aber realitätsnahe Anforderungen. Er stellt sicher, dass Änderungsvorhaben an Bauteilen vor ihrer eigentlichen Umsetzung bewertet, genehmigt und entsprechend dokumentiert werden.

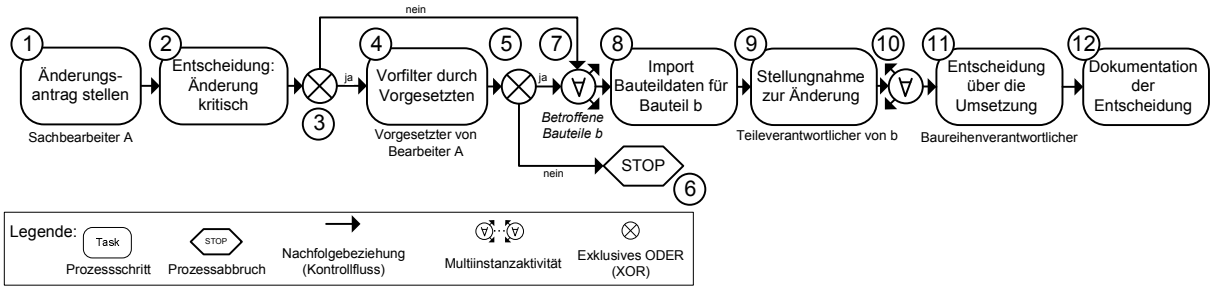


Abbildung 4.1.: Fachprozess zum Änderungsmanagement

Beispiel 4.1 (Fachprozess zum Änderungsmanagement)
Im Fachprozessschritt (1) wird ein Änderungsvorhaben angelegt, indem die Änderung in Form eines Änderungsantrages beschrieben wird. Dieser enthält neben der Änderungskategorie eine Beschreibung der Änderung, mögliche Maßnahmen zu deren Umsetzung, Angaben zu betroffenen Bauteilen sowie eine grobe Abschätzung der resultierenden Kosten.

Eine Änderung wirkt sich oft auf viele Bauteile aus. Das Durchlaufen des Änderungsprozesses kann deshalb hohe Kosten (für Stellungnahmen) verursachen. Diese sollten vermieden werden, wenn eine Umsetzung der Änderung unwahrscheinlich ist. Aus diesem Grund bewertet im Fachprozessschritt (4) der Vorgesetzte des Antragstellers die Erfolgsaussichten des Änderungsvorhabens. Gegebenenfalls kann er den Änderungsprozess vorzeitig beenden (6). Allerdings sind auch für den Vorgesetzten sowohl unnötiger Aufwand als auch unnötige Verzögerungen durch den Fachprozessschritt (4) zu vermeiden. Deshalb wird Schritt (4) nur ausgeführt, wenn ein solcher

Vorfilter sinnvoll ist. Wann dies der Fall ist, entscheidet ein automatischer Fachprozessschritt mittels vordefinierter Geschäftsregeln (2). So kann ein Vorfilter z.B. bei hoher Anzahl zu bewertender Bauteile, in Kombination mit hohen erwarteten Kosten für die Umsetzung der Änderung, sinnvoll sein.

Für alle vom Änderungsvorhaben betroffene Bauteile werden entsprechende Bauteildaten (z.B. Teilenummer, Beschreibung und Material) automatisch aus dem jeweiligen Produktdaten-Management-System (PDM) importiert (8). Anschließend geben die Bauteilverantwortlichen eine Stellungnahme zur Änderung ab (9). Hierbei werden insbesondere die technische Realisierbarkeit der Änderung sowie die tatsächlich resultierenden Kosten aufgrund der Bauteiländerung bewertet. Der Baureihenverantwortliche entscheidet im Fachprozessschritt (11) über die Umsetzung des Änderungsvorhabens. Diese Entscheidung wird automatisiert dokumentiert (12). Es werden sowohl Informationen an das PDM zurück übermittelt als auch an alle IT-Systeme übertragen, die später an der Umsetzung des Änderungsvorhabens beteiligt sind.

Das beschriebene Anwendungsszenario aus Abbildung 4.1 wurde bereits einer fachlichen Optimierung unterzogen, d.h. es hatte bei seiner ersten Realisierung eine andere Struktur. Ursprünglich waren die Fachprozessschritte (3) bis (6) nicht vorhanden, d.h. sie sind erst später hinzugekommen. Aufgrund der nachträglichen Änderung am Fachprozess musste dieser erneut auf die IT-Systeme angepasst werden.

Ziel einer SOA ist es zum einen, aus solchen Fachprozessen schnell und kostengünstig eine IT-Implementierung abzuleiten. Zum anderen sollten Änderungen, wie beispielsweise das Einfügen der Fachprozessschritte (3) bis (6), einfach realisiert werden können, um eine vollständige Neuimplementierung des gesamten Fachprozesses zu vermeiden (vgl. Anforderung 3.3 aus Abschnitt 3.3). Deshalb sollte die Automatisierung durchgängig entlang des gesamten Prozesslebenszyklus gestaltet werden, so dass eine möglichst vollständige Informationsweitergabe von der fachlichen Prozessbeschreibung bis zur technischen Implementierung realisiert wird (vgl. Anforderung 3.2). Insbesondere die Änderbarkeit von Fachprozessen sowie die zugehörige Synchronisation mit der konkreten IT-Implementierung sind essentiell, sowohl aus Sicht des Fachbereichs in Richtung der IT-Implementierung als auch in entgegengesetzter Richtung (vgl. Anforderung 3.4). Dadurch wird es möglich, Fachprozesse in einer SOA zu etablieren und später auf Änderungen zu reagieren. Ein Faktor zur Verringerung der Kosten für die Implementierung von Fachprozessen ist die Wiederverwendung existierender Funktionen. Beispielsweise sollte im Fachprozessschritt (8) ein Service verwendet werden, der bereits existiert und die notwendigen Informationen über ein Bauteil bereitstellt.

Die zuvor beschriebenen Ziele lassen sich nur erreichen, wenn dafür geeignete Rahmenbedingungen geschaffen werden. Dazu gehört unter anderem, dass Informationen über fachliche Anforderungen und Fachprozesse sowie Informationen, die für die konkrete Implementierung benötigt werden, zentral dokumentiert sein müssen (vgl. Anforderung 3.8). Dadurch können beispielsweise geeignete Services für die Fachprozessschritte (8) und (12) bereits in frühen Phasen des Prozesslebenszyklus gefunden und ihre geplante Verwendung dokumentiert werden. Sowohl *Enterprise Architecture Management* [MBL07] als auch Governance-Prozesse und -Gremien nutzen diese zentral abgelegten Informationen, etwa als Grundlage für Entscheidungen. Neben der reinen Informationsspeicherung müssen die Abhängigkeiten zwischen Fachprozessen, Services, IT-Systemen und Bearbeitern explizit verwaltet werden. Unter Nutzung dieser Informationen

erkennen Analysemethoden auftretende Fehler frühzeitg und erlauben so geeignete Reaktionen. Beispielsweise kann analysiert werden, welche Services von welchen Fachprozessen verwendet werden. Diese Information ist wichtig, um bei Änderungen an Services die davon betroffenen Fachprozesse identifizieren und ggf. anpassen zu können.

4.2. Ausbaustufen einer IT-Infrastruktur für SOA

Um die Funktionsfähigkeit der Unternehmenslandschaft sicherzustellen, ist eine Integration der IT-Systeme unabdingbar. Diese Fähigkeit wird technisch durch IT-Infrastrukturen unterstützt. Darunter verstanden wird die Datenverarbeitung sowie die dafür benötigten Hard- und Softwarekomponenten sowie unterstützende Technologien zur Kommunikation zwischen ihnen [IBM93, IBM08a, FN08]. Da die Komplexität der IT-Systeme hoch sein kann, ist schwer bewertbar, ob die im Unternehmen vorhandene IT-Infrastruktur in Bezug auf die Anpassungsfähigkeit adäquat ist. Um diese Bewertung vornehmen zu können, werden nachfolgend zwei verschiedene Ausbaustufen vorgeschlagen, die eine detaillierte Betrachtung der Komponenten und deren Zusammenspiel im Vergleich ermöglichen. Die Ausbaustufen beginnen mit der Beschreibung klassischer Integration auf Basis von Punkt-zu-Punkt-Verbindungen, und enden mit der prozessorientierten Integration von Services. Dabei werden Aspekte wie Governance und Security ausgeklammert.

Zunächst beschrieben wird die Entwicklung von IT-Infrastrukturen vor Einführung von Services, d.h. IT-Infrastrukturen wie sie heute in der Praxis häufig anzutreffen sind. Aus service-orientierter Sichtweise sind diese meist unzureichend und werfen bei der Einführung von Services bzw. einer SOA teils schwerwiegende Probleme auf. Diese werden analysiert und dienen zugleich als Motivation für die nachfolgend vorgestellten Ausbaustufen.

4.2.1. IT-Infrastrukturen vor einer SOA-Einführung

Durch Eigenentwicklungen oder Zukauf von IT-Systemen hat sich in vielen Unternehmen eine heterogene IT-Infrastruktur entwickelt. Insbesondere die Wartung und Pflege dieser zunehmend komplexeren IT-Infrastrukturen gestalten sich als schwierig [Pry05]. Hauptsächlich wird dies durch eine „harte Verdrahtung“ der IT-Systeme verursacht. Diese ist meist als Punkt-zu-Punkt-Verbindung (d.h. Bridge) zwischen jeweils zwei IT-Systemen realisiert. Dies führt bei kontinuierlichem Hinzufügen neuer IT-Systeme zu zahlreichen Abhängigkeiten, die für Anwender nur schwer nachvollziehbar sind. Erschwerend kommt hinzu, dass IT-Infrastrukturen mit dem Technologiefortschritt gewachsen sind, d.h. es werden unterschiedliche Standards und Technologien für die Integration verwendet. Oftmals erfolgt der Datenaustausch manuell. Ferner kann der gemeinsame Zugriff von IT-Systemen auf die Daten über den Austausch von Dateien oder über eine gemeinsame Datenbank erfolgen. Hierbei stellt sich das Problem, dass die Integration ausschließlich über die Datenbasis erfolgt und der eigentliche Ablauf oft nicht transparent wird. Eine direkte Analyse der Abläufe ist somit aufgrund der komplexen Beziehungen zwischen den IT-Systemen kaum möglich.

Diese Betrachtung einer IT-Infrastruktur behandelt bisher nur innerbetriebliche Aspekte. Die

Probleme verschärfen sich im unternehmensübergreifenden Kontext, da hier nicht nur die unternehmensinternen IT-Systeme betrachtet werden müssen, sondern auch die der Partnerunternehmen [RWR06a, RWR06b]. Um diesem Problem zu begegnen, wird eine abstraktere Sicht auf die Gesamt-IT-Infrastruktur benötigt. Dazu wird mit Ausbaustufe 1 eine erste Beschreibung einer Service-orientierten IT-Infrastruktur vorgestellt.

4.2.2. Ausbaustufe 1: Minimale Service-orientierte IT-Infrastruktur

Ziel von Ausbaustufe 1 ist es, hart verdrahteten, heterogenen IT-Infrastrukturen und den mit ihnen einhergehenden Problemen zu begegnen und in Richtung einer Service-orientierten Architektur zu gelangen. Deshalb werden Services zur Kapselung vorhandener IT-Systemfunktionalität eingeführt. Dadurch wird es möglich, standardisierte Technologien für die Kommunikation zwischen Services einzusetzen, wodurch Punkt-zu-Punkt-Verbindungen ersetzt werden. Neben einer solchen Kapselung ist die Wiederverwendung von Services [Erl05] ein wichtiger Schritt in Richtung einer Service-orientierten Architektur. Damit IT-Systeme existierende Services wiederverwenden können, müssen die Informationen über diese zentralisiert abgelegt und auffindbar sein, d.h. eine zentrale Speicherung entsprechender Service-Daten ist nötig (vgl. Anforderung 3.8).

Um dies zu unterstützen, sind verschiedene Komponenten für eine IT-Infrastruktur notwendig, etwa zur Modellierung von Fachprozessen oder zur Ausführung technischer Services. Infrastrukturkomponenten lassen sich in zwei Gruppen untergliedern (vgl. Abbildung 4.2): Erstens sind dies Komponenten, welche die Entwicklung von Informationssystemen unterstützen. Beispielhaft seien die fachliche Service-Modellierung oder die Implementierung technischer Services genannt. Zweitens gibt es Infrastrukturkomponenten, die zur Ausführungszeit der Informationssysteme vonnöten sind, etwa um einen konkreten technischen Service aufzurufen oder den richtigen Bearbeiter für eine bestimmte Aufgabe zu ermitteln. Wir unterscheiden diese beiden Bereiche bei der Entwicklung der Ausbaustufen und bezeichnen sie im Folgenden mit *Build-* und *Runtime*. Die Buildtime lässt sich weiter in einen fachlichen und technischen Teil untergliedern. Infrastrukturkomponeten auf fachlicher Ebene sind primär für Verantwortliche aus Fachbereichen relevant, während die technische Ebene typischerweise in der Verantwortung der IT-Bereiche liegt. Als Runtime bezeichnen wir die Laufzeitumgebung eines Informationssystems. Dazu gehören der Betrieb des Informationssystems, ebenso wie das zugehörige Routing und die richtige Rollenauflösung. Abbildung 4.2 zeigt die Infrastrukturkomponenten der ersten Ausbaustufe.

Geschäftsprozessmodellierung

Eine wichtige Infrastrukturkomponente dient der Modellierung von Fachprozessen. Letztere unterstützen die Dokumentation fachlicher Anforderungen (vgl. Abschnitt 2.2). Die einzelnen Fachbereiche des Unternehmens definieren Fachprozesse (vgl. Definition 2.6)zwecks der Dokumentation fachlicher. Daher ist bereits in dieser Ausbaustufe eine Infrastrukturkomponente notwendig, mit der sich Fachprozesse auf eine für den Fachmodellierer intuitive Art und Weise dokumentieren lassen. Dazu können beispielsweise Modellierungswerkzeuge wie ARIS Business Architect (kurz: ARIS) [Sch01] oder WebSphere Business Modeler (kurz: WBM) [IBM08c] eingesetzt werden (vgl. Abbildung 4.3). Beide Werkzeuge ermöglichen die Modellierung von Fachprozessen, verwenden aber unterschiedliche Notationen.

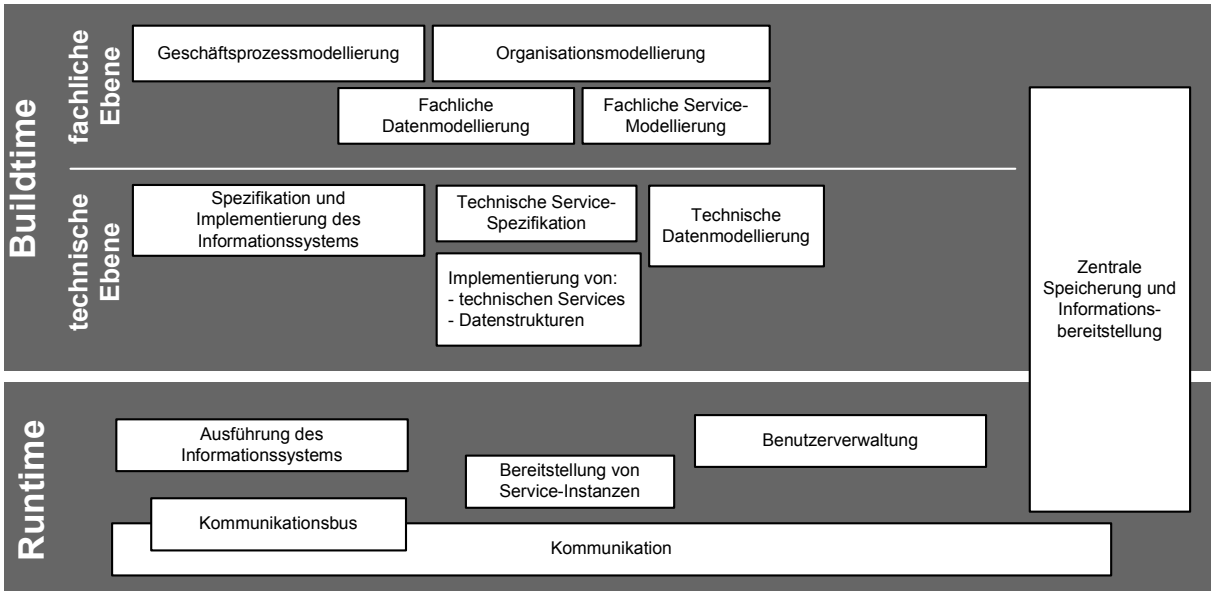
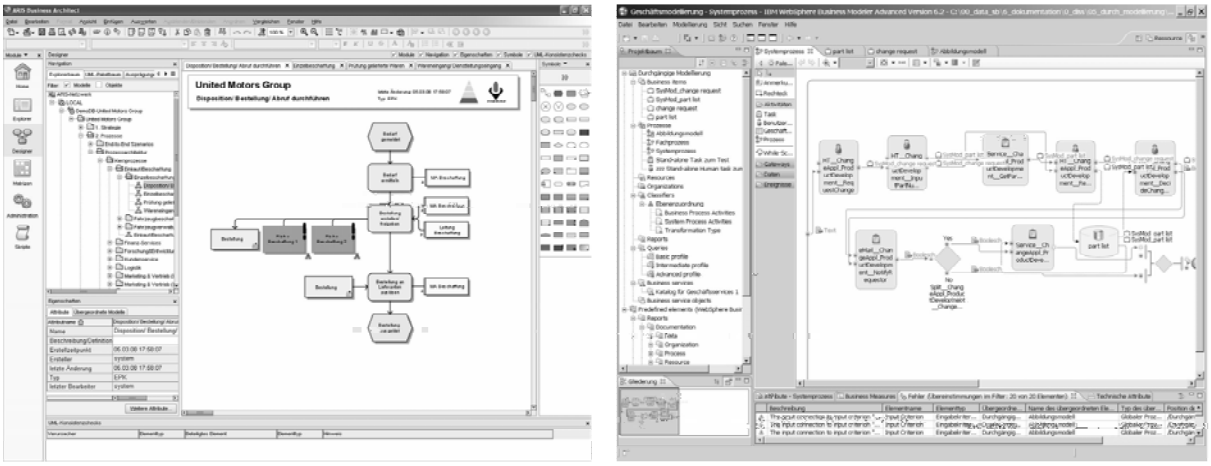


Abbildung 4.2.: Ausbaustufe 1 - Minimale Service-orientierte IT-Infrastruktur



a) ARIS Business Architect b) IBM WebSphere Modeler

Abbildung 4.3.: Werkzeuge zur Geschäftsprozessmodellierung

Fachliche Service-Modellierung

Fachliche Services sind Dienste, die in einem Unternehmen angeboten oder konsumiert werden. Bei der Modellierung fachlicher Services geht es in erster Linie um die fachliche Dokumentation einer Dienstleistung, oft auch fachliche *Service-Spezifikation* genannt. Fachliche Services besitzen nicht-funktionale Eigenschaften, die im Fall einer Nutzung durch *Service-Level-Agreements (SLA)* [BTZ99, OEH02, BWRJ08] spezifiziert werden. Dazu gehören Verfügbarkeit, Nutzungskosten, Sicherheitsaspekte und Laufzeit der Nutzung. Fachliche Services verwenden Datenelemente, die als Ein-/Ausgabeparameter dienen. Sie strukturieren Fachprozessschritte und müssen

daher für Fachbereiche verwendbar sein. Schließlich lassen sich Fachprozesse als fachliche Services kapseln und in anderen Fachprozessen wiederverwenden. Beispielsweise kann der fachliche Service zum Importieren von Bauteildaten (vgl. Fachprozessschritt (8) in Abbildung 4.1) oder derjenige für die Dokumentation über die Entscheidung einer Umsetzung (Fachprozessschritt (12)) in verschiedenen Fachprozessen wiederverwendet werden.

Der Nutzen einer fachlichen Modellierung liegt in der Unterstützung beim Auffinden fachlicher Services und in deren Komposition [Ste08a, Ste08b]. Um diesen Nutzen zu realisieren, werden Anforderungen auf fachlicher Ebene definiert, wie sie der fachliche Service erfüllen muss. Dies betrifft etwa die Vergabe fachlicher Service-Bezeichner sowie die Unterstützung von SLAs und das Aufstellen von Quality-of-Service (QoS)-Kriterien [Pap03].

In unserem Beispiel etwa muss für den fachlichen Service, der die Bauteildaten bereitstellt, definiert sein, welche Daten er benötigt, um im Fachprozessschritt (6) verwendet zu werden (vgl. Abbildung 4.1). Grundsätzliches Ziel der fachlichen Service-Modellierung ist es darüber hinaus, die Geschäftsfunktionen innerhalb des Unternehmens sowie gegenüber Partnerunternehmen gekapselt anzubieten. Eine Möglichkeit für eine Modellierung fachlicher Services beschreibt [Ste08a, Ste08b]. Hierbei wird, wie in Abbildung 4.4 dargestellt, für das Objekt „Service Type“ definiert, welche zusätzlichen Objekte zur Beschreibung eines fachlichen Services im Geschäftsprozessmodellierungswerkzeug ARIS notwendig sind.

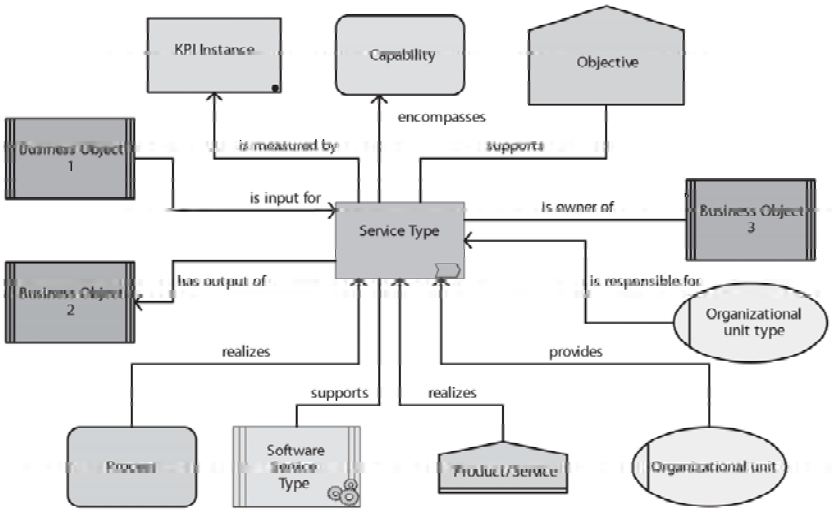


Abbildung 4.4.: Fachliche Service-Modellierung nach [Ste08b]

Organisationsmodellierung

Jedes Unternehmen besteht aus einer Vielzahl von Mitarbeitern und Fachabteilungen mit meist hierarchischer Organisationsstruktur (vgl. Abbildung 4.5). Rollen müssen während der Modellierung von Fachprozessen bestimmten Prozessschritten zugewiesen werden. Dadurch können Rollen bei einer späteren Ausführung dieser Prozesse eindeutig ausführbaren Prozessschritten

zugeordnet werden. Deshalb werden Rollen ebenso wie andere Organisationseinheiten im Organisationsmodell des Unternehmens definiert [RR07, RR08].

Die Organisationsstruktur, inklusive der entsprechenden Rollen, sollte zentral vorgegeben und in allen IT-Systemen verwendet werden. Dadurch reduziert sich der Pflegeaufwand für das Organisationsmodell. Werden diese Informationen zentral gespeichert, können sie bei der Geschäftsprozessmodellierung sowie der fachlichen Service-Modellierung verwendet werden, um Zuständigkeiten und Berechtigungen für die Ausführung von Fachprozessschritten oder den Aufruf von Services zu definieren. Heute existiert (noch) keine standardisierte Methode für die Definition von Organisationsmodellen, dennoch sollte eine Komponente in der IT-Infrastruktur existieren, um Organisationmodelle fachlich zu dokumentieren. Realisiert sind Organisationsmodelle z.B. in bestehenden Anwendungssysstemen [DEJL10].

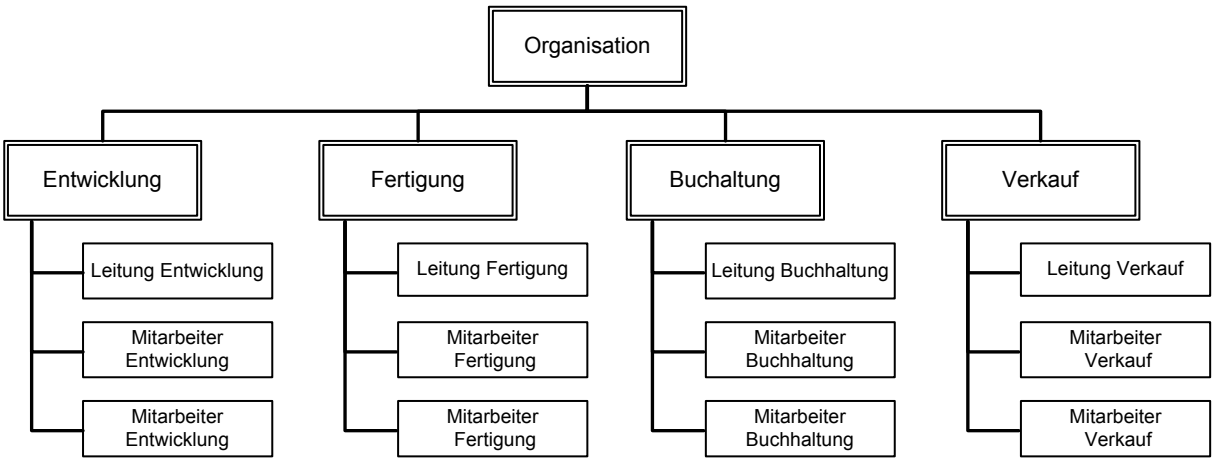


Abbildung 4.5.: Beispiel für ein Organisationsmodell

Fachliche Datenmodellierung

Neben Fachprozessschritten und fachlichen Services spielen Datenelemente in Fachprozessen eine wichtige Rolle. Sie beschreiben nicht nur Informationen innerhalb von Fachprozessen (vgl. Kapitel 2), sondern dienen zugleich als Ein-/Ausgabeparameter für fachliche Services (vgl. *Business Objects* in Abbildung 4.4). Die Modellierung fachlicher Datenelemente muss durch eine Infrastrukturkomponente realisiert werden.

Technische Service-Spezifikation und Implementierung

Technische Services realisieren fachliche Services. Ein technischer Service implementiert die Schnittstelle zu den IT-Systemen in der Unternehmenslandschaft. Diese Schnittstelle muss technisch spezifiziert und implementiert werden, bevor Informationssysteme sie nutzen können. Dazu wird die fachliche Service-Beschreibung entsprechend der technischen Vorgaben verfeinert. Eine sog. technische Service-Spezifikation beschreibt verbindlich, welche Eigenschaften eine Schnittstelle realisiert und detailliert die in der fachlichen Service-Spezifikation beschriebenen

fachlichen Anforderungen technisch. Einen Überblick über die konkreten Eigenschaften technischer Services gibt [Erl05]. Demnach sind diese durch standardisierte Schnittstellen beschrieben, was sie einfach auffindbar und nutzbar macht. Ein weiterer Vorteil der einheitlichen Beschreibung stellt die lose Kopplung mit den Services dar, die aus den darunterliegenden IT-Systemen stammen. Services sind autonom und abstrahieren Funktionslogik, was ihre Wiederverwendung ermöglicht. Weiter lassen sich Services zu neuen Services komponieren [Jos07]. Eine Möglichkeit technische Services zu realisieren, stellen Web Services dar. Letztere werden durch Standards (z.B. XML [CCMW01], WSDL [W3C06a], HTTP [FGM+99, RS95], SOAP [SOA00]) sowie Profile (etwa WS-I [WSI06], WS-Policy [W3C06b] und WS-RM [OAS09]) beschrieben, die durch unterschiedliche Organisationen (Open Group, OASIS, W3C, OMG) definiert werden. Web Services besitzen Dienstgüteeigenschaften (QoS), die für die Service-Nutzung relevant sind. Zu ihnen gehören zum Beispiel die Antwortzeit eines Services, die mögliche Nutzungslast (mögliche Aufrufe pro Zeiteinheit) oder verwendete Sicherheitsstandards (Veschlüsselung).

Beispielsweise könnte ein technischer Service *BauteilDaten* die Funktionalität bereit stellen, Bauteilinformationen zu einer Bauteilnummer zu ermitteln (vgl. Fachprozessschritt (8) in Abbildung 4.1). So wird durch die Operation *getBauteilDaten(Bauteilnummer)* die für den Fachprozess notwendige Information auf technischer Ebene zur Verfügung gestellt. Die Spezifikation und Implementierung technischer Services wird in der Praxis meist durch IT-Verantwortliche und entsprechende Werkzeuge, etwa CASE-Tools, unterstützt.

Technische Datenmodellierung

Die auf fachlicher Ebene dokumentierten Datenobjekte werden durch eine weitere Infrastrukturkomponente verfeinert. Dabei werden insbesondere abstrakt beschriebene fachliche Datenobjekte in technische Datenobjekte überführt und durch technische Details vervollständigt. CASE-Tools unterstützen die Modellierer bei der Erstellung komplexer Datenstrukturen, etwa mittels UML-Klassendiagramm.

Zentrale Dokumentation modellierter Artefakte

In einer SOA wird ein zentrales Repository zur Speicherung von Informationen und zur Dokumentation der Abhängigkeiten zwischen fachlichen und technischen Artefakten benötigt. Eine wichtige Frage in diesem Zusammenhang ist, mit welchen Informationen das Repository befüllt werden soll.

- **Fachliche Ebene:** Daten des fachlichen Teils des SOA-Repository entstehen bei der Modellierung der Fachprozesse, des Organisationsmodells und der fachlichen Services. Es werden Informationen wie Prozessversionen, Prozessbeschreibung, Gültigkeitszeitraum des Fachprozesses und konkrete Bearbeiterzuordnung einzelner Fachprozessschritte gespeichert. Die Speicherung organisatorischer Informationen im Repository dient der zentralen Dokumentation (meist durch ein LDAP-fähiges Verzeichnis [YHK95] realisiert) und Pflege von Organisationsmodellen. Dadurch wird die Auflösung von Rollen, Personen und Verantwortlichkeiten zur Laufzeit ermöglicht. Darüber hinaus wird die Information über die

Verwendung fachlicher Services in Fachprozessen abgelegt. Zu diesen Informationen gehören Servicequalität (z.B. Reaktionszeit), Funktionalität (welchen Zweck erfüllt der Service) und Rahmenbedingungen des Services sowie die von ihm erzeugten Geschäftsobjekte.

- **Technische Ebene:** Der technische Teil des Repository speichert Informationen über technische Services, die während der Entwicklung von Informationssystemen benötigt und später für die Ausführungsphase notwendig sind. Diese Informationen können z.B. den aktuellen Status (Service ist technisch spezifiziert und implementiert) beschreiben. Es handelt sich um Informationen, welche die konkreten Schnittstellen (z.B. WSDL), die zugehörigen Datenstrukturen sowie Laufzeitinformationen betreffen [ACKM04].

Spezifikation und Implementierung des Informationssystems

Die Spezifikation eines Informationssystems definiert die Anforderungen, denen die zu entwickelnde Applikation genügen muss. Diese resultieren aus den dokumentierten Fachprozessen, deren Bearbeitung durch das Informationssystem wiederum unterstützt werden soll. Die Implementierung des Informationssystems findet in einer weiteren Infrastrukturkomponente statt. IT-Implementierer setzten die Spezifikation unter Einbeziehung von technischen Services, Organisationsmodellen und technischen Datenstrukturen.

Benutzerverwaltung

Damit das zuvor implementierte Informationssystem eine Bearbeiterzuordnung zur Laufzeit auflösen kann, legt der Administrator (oder eine autorisierte Person) sowohl die hierarchische Struktur als auch die Benutzer und Rollen des Unternehmens im Repository an. Eine Berechnung konkreter Mitarbeiter zu einer vorgegebenen Bearbeiterzuordnung erfolgt zur Ausführungszeit auf Basis des Repositories [RD00]. Das bei der Implementierung des Informationssystems zugrundeliegende Organisationsmodell dient einer solchen Berechnung. So wird Fachprozessschritt (5) aus Abbildung 4.1 nur von denjenigen Personen bearbeitet, die der Rolle „Teileverantwortlicher“ zugeordnet sind. Eine Infrastrukturkomponente übernimmt die Anfragen an das SOA-Repository und die Auflösung der Bearbeiterzuordnungen zur Ausführungszeit.

Technische Services Instanz

Eine Service-Instanz beschreibt einen konkreten technischen Service, der für Informationssysteme über einen Service-Endpunkt aufrufbar ist. Um dies zu realisieren, muss eine Infrastrukturkomponte eingesetzt werden, welche die Installation und den Betrieb von Service-Instanzen ermöglicht, etwa ein Applikationsserver.

Kommunikationsbus

Wie erwähnt, sind bei der Implementierung von Informationssystemen technische Services essentiell. Diese müssen aufrufbar sein, damit das Informationssystem sie zur Ausführungszeit nutzen kann. Deshalb wird ein Kommmunikationsbus eingeführt, der einen solchen Aufruf weiterleitet. Der Kommunikationsbus ist eine Infrastrukturkomponente, die zur Ausführungszeit Informationen über technische Services sowie die für einen Service-Aufruf notwendigen Parameter (z.B. XSD und WSDL) aus dem SOA-Repository bezieht. Mittels dieser Information wird der spätere Service-Aufruf statisch realisiert, d.h. es wird immer derselbe Service-Endpunkt gerufen. Informationssysteme rufen schließlich über den Kommunikationsbus die entsprechenden Service-Instanzen auf.

4.2.3. Ausbaustufe 2: Service-orientierte IT-Infrastruktur

In diesem Abschnitt wird die aus ENPROSO-Sicht notwendige IT-Infrastruktur vorgestellt, um Fachprozesse technisch umzusetzen und flexibel auf Änderungen reagieren zu können.

Aufbauend auf Ausbaustufe 1 beschreibt Ausbaustufe 2 weitere Infrastrukturkomponenten, welche die geforderte Funktionalität bereitstellen (vgl. Abbildung 4.6). Die verschiedenen Infrastrukturkomponenten werden anschließend in kombinierbaren *Unterstufen* organisiert. Letztere dienen dem Zweck, Infrastrukturkomponenten und deren Abhängigkeiten untereinander zu beschreiben. Wenn beispielsweise technische Services erstmals zur Ausführungszeit des implementierten Informationssystems bekannt sind, reicht die Funktionalität des Kommunikationsbusses aus Ausbaustufe 1 nicht aus. Somit muss weitere Funktionalität geschaffen werden, die das dynamische Binden technischer Service-Instanzen zur Ausführungszeit ermöglicht. Die Fachprozessschritte (8) und (12) aus Abbildung 4.1 sind hierfür Beispiele: Die Lokation (Endpunkte) der für diese beiden Schritte verwendeten Services sollten nicht fest vorgegeben sein. Erst zur Laufzeit steht fest, welche Bauteildaten zu importieren sind. Eventuell sind für die Anfrage unterschiedliche Produktdaten-Management-Systeme (PDM-System) notwendig, die in verschiedenen Geschäftsbereichen angesiedelt sein können. Unter Umständen sind sogar die Schnittstellen der verschiedenen PDM-Systeme unterschiedlich. Diese Funktionalität kann jedoch nicht alleine durch Erweiterung des Kommunikationsbusses erreicht werden. Deshalb werden zusätzliche Komponenten notwendig und bestehende müssen erweitert werden. Die resultierenden Abhängigkeiten werden durch eine detaillierte Beschreibung der Komponenten transparent gemacht. Erst dadurch lässt sich bewerten, welche Komponenten für eine Unternehmenslandschaft in welchem Detaillierungsgrad sinnvoll sind. Abbildung 4.6 zeigt die Erweiterungen gegenüber Ausbaustufe 1. Die neuen und teilweise erweiterten Infrastrukturkomponten sind weiß dargestellt.

Geschäftsregelmodellierung

Die erste Erweiterung in Ausbaustufe 2 ist eine Infrastrukturkomponente zur Beschreibung von Geschäftsregeln. In Fachprozessen sind Geschäftsregeln häufig mit Verzweigungsknoten verknüpft. Hierzu werden die Verzweigungsentscheidungen in Fachprozessen (d.h. der gewählte Pfad) mittels Geschäftsregeln berechnet. Die Flexibilität des Fachprozesses wird hierdurch erhöht, da auch Fachanwender ohne IT-Kenntnisse diese Regeln zur Ausführungszeit anpassen können. Im

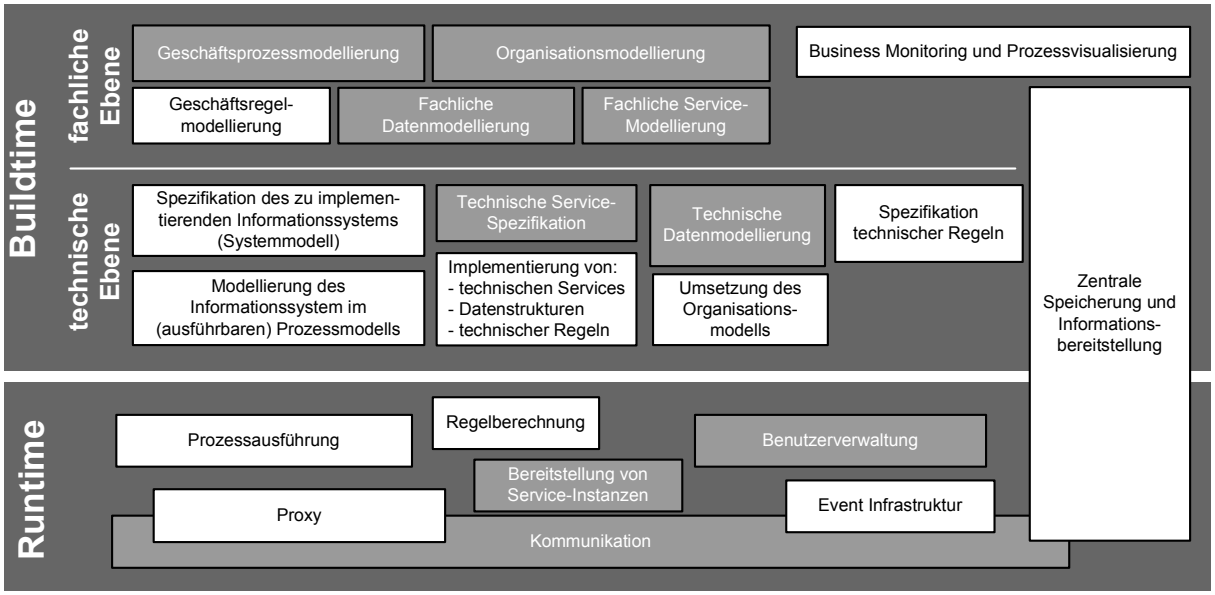


Abbildung 4.6.: Ausbaustufe 2 der SOA-IT-Infrastruktur

Anwendungsszenario aus Abbildung 4.1 geschieht die Auswertung bei Fachprozessaktivität (2) mittels einer Geschäftsregel. Dazu wertet die Geschäftsregel aus, inwieweit sich der Aufwand für den Fachprozessschritt (4) rechnet.

Außerdem werden Geschäftsregeln für die Zuordnung von Ressourcen und Bearbeitern verwendet. Geschäftsregeln können unternehmensweit, für einen bestimmten Fachprozess oder für genau einen Prozessschritt definiert werden.

Die Geschäftsregeldefinition erfolgt auf fachlicher Ebene. Die Auswertung der Regel erfolgt instanzbezogen zur Ausführungszeit. Die definierten Geschäftsregeln sind explizit als solche erkennbar, können im Fachprozess verwendet werden und sind flexibel änderbar. Geschäftsregeln lassen sich auf unterschiedliche Art und Weise beschreiben:

- **Textbasierte Beschreibung:** Eine rudimentäre Definition von Geschäftsregeln kann z.B. informell mittels Freitext erfolgen.
- **Grafische Spezifikation:** Bestenfalls existiert zur Definition der Geschäftsregeln eine grafische Werkzeug-Unterstützung (z.B. ein Regeleditor wie ARIS Business Rule Designer). Idealerweise werden die Geschäftsregeln zentral abgelegt, etwa in einem Repository [HM96], so dass diese wiederverwendet und technisch verfeinert werden können.
- **Automatische Modellprüfung:** Als Erweiterung dieser Funktionalität können syntaktische und semantische Prüfungen der Geschäftsregeln bereits zur Modellierungszeit vorgenommen werden.

Business Activity Monitoring

Ein wichtiger Aspekt bei der Messung der Qualität einer Unternehmenslandschaft ist das Überwachen (Monitoring) laufender Applikationen und Fachprozesse. Überwachung bedeutet nicht das technische Monitoring der IT-Infrastruktur, etwa zwecks Feststellung der Verfügbarkeit von Komponenten. Es geht vielmehr um eine geeignete Überwachung der Applikationen zur Sicherstellung der Geschäftsziele, auch *Business Activity Monitoring* (BAM) genannt. Informationslieferant für BAM ist die zugrundeliegende technische Infrastruktur. Sie übermittelt die relevanten Ereignisse an die Monitoring-Komponente. Die auf fachlicher Ebene definierten Key-Performance-Indikatoren (KPI) werden dann auf Basis dieser Ereignisse berechnet [AAC+08].

- **Manuelle Definition von KPIs:** In der 1. Unterstufe des BAM müssen KPIs und die zu ihrer Bestimmung benötigten bzw. zu überwachenden Ereignisse manuell in der Monitoring-Komponente definiert werden [BWRJ08]. Ein Beispiel eines KPI ist die maximale Durchlaufzeit eines Fachprozesses, die von der Fachabteilung vorgegeben wird. Durch die technische Infrastruktur werden anschließend Ereignisse definiert, die sich auf die einzelnen Aktionen (z.B. Start eines Service-Aufrufs) beziehen und die zur Ausführungszeit an die BAM-Komponente übermittelt werden. Letztere berechnet die jeweiligen KPIs. In unserem Beispiel etwa kann überprüft werden, wie sich der Vorfilter aus Abbildung 4.1 auf die Durchlaufzeit der Fachprozessschritte (2) bis (11) ausgewirkt hat.
- **Ableitung von KPIs:** Als Erweiterung unterstützt die Infrastrukturkomponente Benutzer bei der Definition von KPIs. Dazu müssen, ausgehend von der Geschäftsprozessmodellierung und den fachlich modellierten KPIs, die für die technische Infrastruktur notwendigen Ereignisse abgeleitet werden. Ferner müssen in der BAM-Komponente geeignete Algorithmen zur Berechnung der KPIs zur Laufzeit existieren.

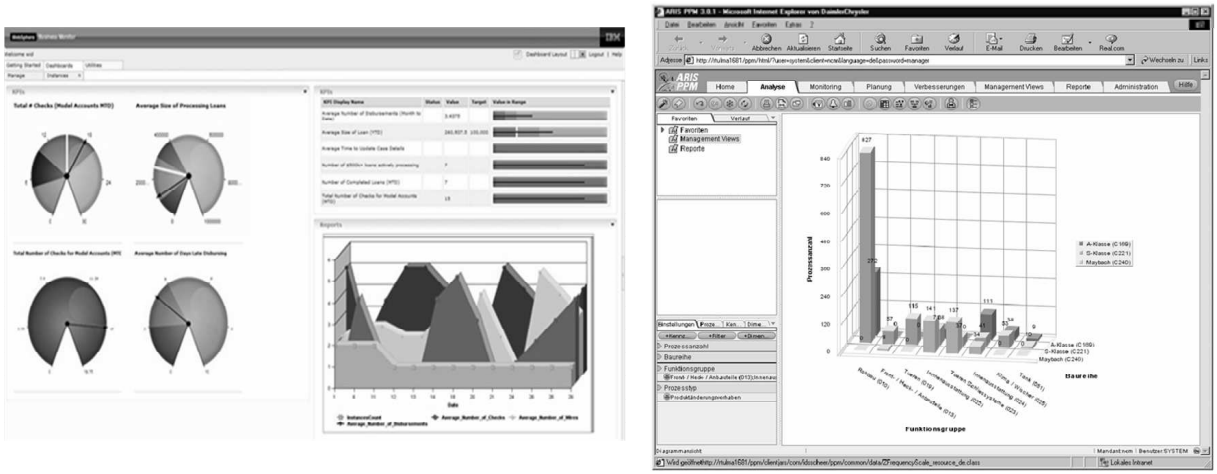
In der Praxis werden Monitoring-Werkzeuge für das Überwachen von KPIs eingesetzt. Abbildung 4.7 illustriert dies an dem Beispiel des IBM WebSphere Business Monitors [AAC+08] und ARIS Process Performance Managers [Sof05].

Prozessvisualisierung

Im Gegensatz zu dem vereinfachten Fachprozess aus Abbildung 4.1 sind reale Prozesse komplexer. Da IT-unterstützte Fachprozesse flexibel an Prozessänderungen angepasst werden können müssen, muss z.B. die Ablaufstruktur analysierbar sein. Eine adäquate Visualisierung von sich in Ausführung befindlichen Prozessen ist für das Monitoring und die Analyse der Fachprozesse essentiell [BRB05]. Hierbei benötigt nicht jeder Nutzer oder Analyst den kompletten Fachprozess, sondern angepasste bzw. personalisierte Sichten (Views) auf diesen [BRB07, Bob08, RBBB10, RKBB12, KRW12]. Durch Einsatz von Prozess-Views kann zum Beispiel bei zu langen Durchlaufzeiten erkannt werden, bei welchen Fachprozessschritten Probleme auftreten.

Spezifikation technischer Regeln

Fachliche Geschäftsregeln müssen technisch verfeinert und formal beschrieben werden, da sie zur Ausführungszeit durch eine Infrastrukturkomponente (eine Rule-Engine) ausgewertet wer-



a) IBM WebSphere Business Monitor b) ARIS Process Performance Manager

Abbildung 4.7.: Werkzeuge zum Business Activity Monitoring

den. Dazu müssen die Regeln in eine maschinenlesbare Sprache überführt werden. Sollte auf fachlicher Ebene lediglich eine textuelle Beschreibung der Geschäftsregeln erfolgt sein, muss diese in maschinenlesbare Form transformiert werden. Beispiele für maschinenlesbare Sprachen sind die *Business Rules Markup Language (BRML)* [Cov02], *Rule Markup Language (RuleML)* [BPS10, BAP+11] und *DARPA Agent Markup Language (DAML)* [HM00, A+01]. Als Ergebnis erhält man eine Spezifikation einer technischen Regel, die durch eine Rule-Engine ausgewertet werden kann [Lie07].

Umsetzung des Organisationsmodells

In vielen Unternehmen wird zur Speicherung und Verwaltung der Organisationsstruktur ein Verzeichnisdienst eingesetzt, etwa ein *Lightweight Directory Access Protocol* fähiges Directory (LDAP Directory) oder ein *Active Directory* [YHK95].

Spezifikation des zu implementierenden Informationssystems

In Ausbaustufe 2 sollte die Spezifikation und Implementierung des Informationssystems service- und prozessorientiert stattfinden, d.h. bisher dokumentierte fachliche Aspekte werden technisch verfeinert. Dies wird mit Hilfe des Systemmodells (vgl. Definition 2.8 aus Abschnitt 2.2.4) realisiert, das die Spezifikation der Implementierung auf technischer Ebene repräsentiert. Diese Spezifikation dokumentiert alle für die Implementierung und spätere Ausführung des Informationssystems relevanten Informationen. Dazu gehören, neben einer plattformunabhängigen Beschreibung der Prozesslogik, alle zuvor beschriebenen Aspekte, etwa technische Services, die im Systemprozess automatisierte Systemprozessschritte realisieren, oder zugehörige Datenstrukturen für einen Service-Aufruf. Das Systemmodell umfasst alle für die Implementierung des Systemprozesses notwendigen Informationen in Form einer technischen Spezifikation.

Modellierung des Informationssystems

Die in dieser Ausbaustufe verwendete Implementierung des Systemmodells wird durch *Prozess-Management-Technologie* realisiert [Rei00, RW12]. Dabei wird ein ausführbares Modell (vgl. Definition 2.7) erstellt, welches die technische Spezifikation (Systemmodell) auf eine beliebige Zielplattform abbildet. D.h. der Systemprozess, die technischen Services und Datenstrukturen sowie die technischen Regeln werden in eine zielplattformabhängige Darstellung überführt. Durch Einsatz von Prozess-Management-Technolgie wird die Ablauflogik aus den IT-Systemen herausgelöst [RD00]. Aufgrund dieser Entkopplung von Prozesslogik und Anwendungslogik wird die Unternehmenslandschaft flexibler, da nicht nur auf fachlicher Ebene durch den Fachprozess, sondern auch auf technischer Ebene die Transparenz erhöht wird.

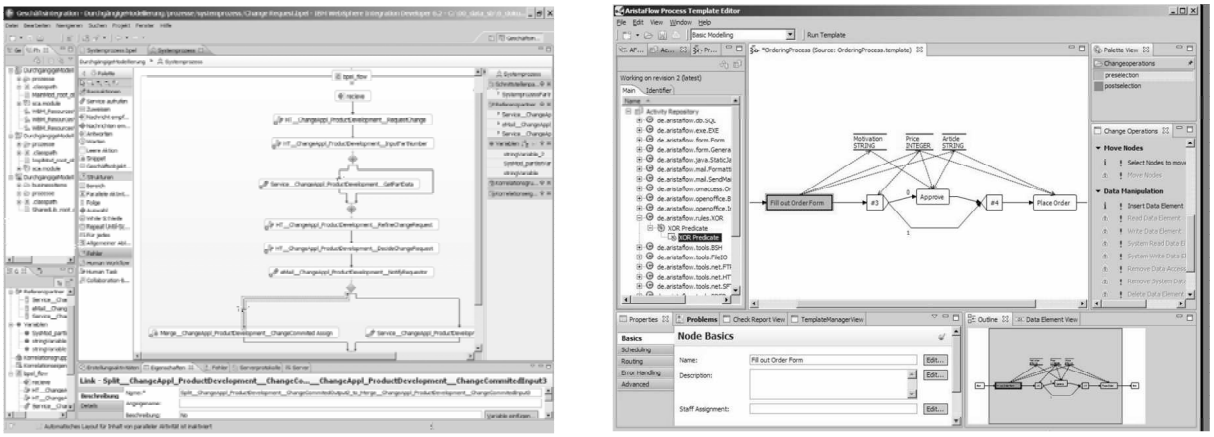
Bei der Modellierung des ausführbaren Modells (auch *Prozessschema* genannt), werden die im Systemprozess dokumentierten Prozessschritte technisch abgebildet. Dazu werden meist grafische Editoren zur Erstellung des Prozessschemas eingesetzt. Letztere beschreiben die Prozesslogik, mittels der die technischen Services orchestriert werden sollen [RS04]. Dadurch wird die Prozess- bzw. Orchestrierungslogik, die bisher in den einzelnen IT-Systemen versteckt war, transparent und zentral änderbar. Es existieren zwei grundlegende Unterstufen zur Modellierung eines Prozessschemas:

- **Schemamodellierung:** Die erste Unterstufe beschreibt die Erstellung eines Prozessschemas durch einen Modellierer aus dem IT-Bereich. Dieser spezifiziert das Prozessschema zielplattformabhängig unter Verwendung des Systemprozesses. Dazu sind z.B. die Modellierung von Kontrollfluss, Datenfluss und Ressourcenzuteilung einzelner Prozessschritte erforderlich. Das resultierende Prozessschema kann später durch Ergänzung von Laufzeitinformationen von einer Prozess-Engine ausgeführt werden.
- **Schemagenerierung:** Das Prozessschema wird automatisch generiert, d.h. mittels automatischer Transformation des Systemprozesses. In einem manuellen Schritt muss das Prozessschema ggf. noch um Informationen ergänzt werden, die für die Ausführung notwendig sind, z.B. um Informationen, die zum Aufruf technischer Services benötigt werden (z.B. Nachrichtenformate) [AAA+07].

Heutige Modellierungswerkzeuge unterstützen die zuvor beschriebenen Unterstufen. Beispielhaft stehen dafür der WebSphere Integration Developer [Pet05] oder die AristaFlow BPM Suite [DRRM+09, DR09, RDRM+09] (vgl. Abbildung 4.8). Die Modellierungsnotationen unterscheiden sich dabei abhängig vom Werkzeug. So wird im WebSphere Integration Developer BPEL verwendet, wohingegen die AristaFlow BPM Suite eine erweiterte blockstrukturierte Notation [RD97, Rei00] als Orchestrierungssprache verwendet.

Prozessausführung

Die Prozess-Engine ist verantwortlich für die Erzeugung und Ausführung von Prozessinstanzen. Sie steuert sowohl die Aufrufe von Service-Instanzen als auch die Zuordnung von konkreten Bearbeitern zu manuell ausgeführten Fachprozessschritten (vgl. Abbildung 4.1). Generell unterscheiden sich Prozess-Engines in ihrer Funktionalität:



a) IBM WebSphere Integration Developer b) AristaFlow BPM Suite

Abbildung 4.8.: Werkzeuge zur Modellierung ausführbarer Prozessmodelle

- **Ausführung von Prozessen:** Die Prozess-Engine muss in der Lage sein, Instanzen von Prozessschemata abzuleiten und auszuführen. Dazu gehört z.B. das Aufrufen technischer Service-Instanzen. Diese Unterstufe repräsentiert die Grundfunktionalität einer flexiblen Prozessausführung.
- **Ad-hoc-Änderungen:** In der Praxis muss flexibel auf Ausnahmesituationen reagiert werden können [RD97, RD98, WRR08, RRD09]. Beispielsweise kann der Fachprozessschritt (11) aus Abbildung 4.1 ausnahmebedingt entfallen, etwa wenn die Umsetzung bereits durch einen Vorstandsbeschluss genehmigt worden ist. Ad-hoc Abweichungen ermöglichen die Behandlung solcher Ausnahmen während der Ausführung von Prozessinstanzen [RD98, Rei00].
- **Schemaevolution:** Infolge von Prozessoptimierungen und -änderungen kann erforderlich werden, ein Prozessschema auf Typebene zu ändern (und nicht nur eine einzelne laufende Prozess-Instanz). In einem solchen Fall muss die Prozess-Engine prinzipiell die Migration laufender Prozess-Instanzen auf ein geändertes Schema unterstützen [RRD03, Rin04, RRD04b, RRD04a]. In diesem Kontext ist zu entscheiden, ob laufende Prozess-Instanzen migriert werden oder auf dem alten Prozessschema weiterlaufen sollen. Auch für noch nicht gestartete Prozess-Instanzen kann die Prozess-Engine unterscheiden, ob das alte oder das neue Prozessschema verwendet werden soll.

Heutige Prozess-Management-Systeme unterstützen die vorgestellten Unterstufen gut. So kann z.B. die AristaFlow BPM Suite eingesetzt werden, um flexibel auf Prozessänderungen reagieren zu können.

Proxy

Ein hohes Maß an Laufzeit-Flexibilität ist für jede SOA essentiell, auch weil die Auswahl der aufzurufenden Service-Instanzen erst zur Ausführungszeit erfolgt. Neben dem dynamischen Binden von Services zur Ausführungszeit ist der Umgang mit Änderungen, etwa einer Service-

Abschaltung oder einem Service-Lokationswechsel, notwendig. Um solchen Flexibilitätsansprüchen gerecht zu werden, erweitern wir den Kommunikationsbus aus Ausbaustufe 1 zu einem Proxy mit folgenden Funktionen:

- **Dynamisches Binden:** Beim Kommunikationsbus in Ausbaustufe 1 werden technische Service-Instanzen statisch gebunden. Der Proxy realisiert dagegen das dynamische Binden von Service-Instanzen: Endpunkte für Service-Instanzen werden vom Proxy zur Ausführungszeit mittels Informationen aus dem Repository ermittelt und gebunden. Der Proxy bekommt lediglich eine technische Service-Spezifikation vom Service-Consumer (z.B. einer Prozess-Engine) übergeben und ruft mittels dieser den passenden technischen Service auf. Durch das Zusammenspiel von Proxy und Repository ist es möglich, technische Service-Instanzen physisch zu verschieben, ohne laufende Prozess-Instanzen zu beeinträchtigen. Die eigentliche Information über die Lokation der technischen Service-Instanz ist im Repository hinterlegt und wird vom Proxy zur Ausführungszeit angefragt.
- **Endpunktsuche:** Ein Proxy ist in der Lage, erweiterte Suchanfragen an das Repository zu stellen. Diese können über Service-Eigenschaften und -Merkmale (z.B. Service Level Agreements) verfügen. Ein SLA etwa könnte fordern, dass der Service-Aufruf verschlüsselt erfolgen muss. Findet der Proxy zu einer technischen Service-Spezifikation mehr als einen passenden Service, existieren ferner Strategien zur Auflösung dieses Szenarios (z.B. der erste passende Service wird ausgewählt).
- **Datentransformation:** Eine weitere Aufgabe des Proxy ist die Transformation von Daten zwischen Service-Consumer und -Provider. Dazu enthält der Proxy-Ablauf meist einen Knoten, der die notwendige Transformation definiert.

Um den beschriebenen Aufruf zwischen Service-Consumer und -Provider zu realisieren, werden im Proxy die Abläufe statisch definiert. Mittels Komposition technischer Services und der dazu notwendigen Transformation der Daten, realisieren Proxies weitere technische Services, die wiederum von ausführbaren Prozessen oder einem anderen Proxy-Ablauf verwendet werden können.

In der Praxis wird ein Proxy oft durch einen *Enterprise Service Bus (ESB)* realisiert.

Regelauswertung

Fachlich modellierte und technisch spezifizierte Geschäftsregeln müssen zur Ausführungszeit ausgewertet werden. Voraussetzung dafür ist die Speicherung dieser Regeln in einem Repository. Die Regelauswertung findet zur Ausführungszeit statt und kann z.B. als technischer Service oder durch eine Rule-Engine realisiert werden [Lie07]. Im Folgenden werden drei Unterstufen beschrieben:

- **Regelauflösung zur Laufzeit:** Die Komponente zur Regelauswertung greift zur Ausführungszeit auf im Repository hinterlegte Schwellwerte zu. Ein Schwellwert beschreibt einen Grenzwert, ab dem eine bestimmte Regel erfüllt ist. Das heißt, zur Ausführungszeit erfolgt eine Anfrage durch das Informationssystem bei der Rule-Engine zwecks Evaluation einer bestimmten Geschäftsregel. Diese liefert das Ergebnis zurück, das die Basis für eine Verzweigungsbedingung liefert.

- **Dynamische Schwellwerte:** Ausgehend von der statischen Evaluation der Regeln zur Ausführungszeit, ergänzt diese Unterstufe die Evaluation um dynamische Aspekte: Die Regelauswertung berücksichtigt dynamische Schwellwertänderungen innerhalb der Regelbedingungen. Das ermöglicht Fachanwendern eine Einflußnahme auf die Fachprozesse, da Schwellwerte auch noch während der Ausführung von Prozessen geändert werden können.
- **Dynamische Regelkomposition:** Im Vergleich zu den vorherigen Unterstufen können zusätzlich Regeln zur Ausführungszeit ausgetauscht werden. Ein solcher Austausch wird mittels vordefinierter Regel-Fragmente realisiert. Dies bedeutet, dass Regel-Fragmente definiert und zur Ausführungszeit durch Fachanwender kombiniert und verwendet werden. Dadurch kann z.B. im Prozessschritt (2) aus Abbildung 4.1 berücksichtigt werden, ob die beantragte Änderung das Gewicht des Fahrzeugs positiv oder negativ beeinflusst und dadurch ein Vorfilter notwendig wird.

Ereignis-Infrastruktur

Durch den Einsatz von Ereignissen können Anwendungen untereinander Daten austauschen. Ferner können Ereignisse zum Überwachen von Anwendungen genutzt werden. Technisch wird dies meist durch einen Nachrichtenaustausch über den Proxy realisiert. Mittels Publish- & Subscribe-Mechanismus [Pry05] können sich Empfänger für relevante Ereignisse registrieren, etwa Start- und Endezeitpunkte (z.B. wann wurde Proxy-Instanz x auf Proxy-Engine z gestartet und beendet) [Buc05]. So könnte ein technisches Ereignis zum Fachprozessschritt (9) aus Abbildung 4.1 definiert werden, das anzeigt, ob der zuständige Teileverantwortliche die Freigabe schon erteilt hat. Dauert letztere zu lange, kann entsprechend reagiert werden.

Durch Überwachen (Business Monitoring) von Ereignissen ist es möglich, Engpässe bei der Bearbeitung von Aufgaben (z.B. sehr lange Arbeitslisten von Benutzern oder größere Verzögerungen bei einzelnen Aufgaben) zu erkennen. Dies bildet die Grundlage dafür, bei Problemen spontan eingreifen oder Maßnahmen zur Prozessoptimierung durchführen zu können.

Ein Beispiel für eine Ereignis-Infrastruktur ist die *Common Event Infrastructure* [IBM08b].

Zentrale Dokumentation von Artefakten (Repository)

In Ausbaustufe 2 speichert das Repository Daten, die für den Proxy, die Regelevaluation, die Prozess-Engine und das Business Activity Monitoring notwendig sind. So werden aufzurufende Service-Endpunkte für die technische Realisierung der Fachprozessschritte (8) und (12) aus Abbildung 4.1 oder Schwellwerte für Geschäftsregeln (vgl. Fachprozessschritt (2)) verwaltet. Außerdem kann ein Repository noch weitergehende Funktionalität realisieren:

- **Speicherung von Beziehungen zwischen Objekten:** Ein zusätzlicher Aspekt betrifft die Speicherung und das Management von Objektversionen sowie die Dokumentation von Abhängigkeiten zwischen fachlichen und technischen Services. Zusätzlich dazu sind Beziehungen zwischen Fachprozessen und ausführbaren Prozessen (inkl. Daten) zu dokumentieren. Nur wenn diese Beziehungen explizit dokumentiert werden, ist eine bidirektionale Nachvollziehbarkeit zwischen fachlichen und technischen Artefakten möglich.

- **Automatische Analyse und Notifikation:** Basierend auf der Dokumentation von Abhängigkeiten werden Analyseverfahren bereitgestellt, die Problemsituationen erkennen (z.B. explizites Suchen verletzter Abhängigkeiten). Außerdem werden Notifikationsmechanismen angeboten, welche die von Änderungen betroffenen Objekte (bspw. Fachprozess- oder Service-Verantwortliche) informieren.

4.3. Diskussion

SOA ist ein Thema, mit dem sich zahlreiche Softwarehersteller, Gremien und Wissenschaftler befassen. Seit der Definition wichtiger Grundprinzipien einer SOA durch Gartner [Nat03] hat sich viel getan, jedoch mangelt es an grundlegenden Betrachtungen zu Service-orientierten Architekturen. Dies hat dazu geführt, dass für Unternehmen die Bewertung der eigenen IT-Infrastruktur im Hinblick auf die Entwicklung und den Betrieb service- und prozessorientierter Applikationen nur schwer möglich ist.

Um die in dieser Arbeit entwickelten Konzepte und Methoden strukturieren und einordnen zu können, ist ein generelles Verständnis einer SOA-Infrastruktur und ihrer Komponenten erforderlich. Einige Softwarehersteller haben zwar mächtige ESBs [FN08] und Prozess-Engines [IBM08a, IBM08c] in ihrem Portfolio, bieten jedoch keine adäquaten Lösungen im Bereich der Modellierung von Fachprozessen oder Geschäftsregeln an. In der Praxis werden dafür meist Geschäftsprozessmodellierungswerkzeuge bevorzugt. Dies ist nur ein Beispiel für die Kombination von Komponenten verschiedener Hersteller aufgrund der unterschiedlichen Funktionalität. Die entwickelte IT-Infrastruktur bietet die Möglichkeit zu bewerten, welche Funktionalität eine Komponente aufweisen muss, um der Forderung, nach Reaktionsfähigkeit auf Änderungen gerecht zu werden.

Gremien wie OASIS [MLM+06] haben eine anderen Sichtweise auf SOA. Diese befasst sich nicht mit einer funktionalen Beschreibung der Komponenten. Auch eine detaillierte Beschreibung, wie und in welcher Form welche Informationen im Repository abgelegt werden, existiert nicht.

In der SOA-Literatur finden sich ferner die Begriffe Registry und Repository. Eine Zusammenfassung der Unterschiede und Gemeinsamkeiten sich in [GR07, Mat05] dokumentiert. Dort werden insbesondere die Eigenschaften und Ziele von Repository und Registry zur Entwicklungs- und Laufzeit beschrieben. Eine Registry ist während der Entwicklungsphase für das Auffinden technischer und fachlicher Services anhand ihrer Beschreibungen zuständig. Während des Service-Betriebs ist sie für die Einhaltung von Kontrakten für Service-Policies und für Versionierung verantwortlich. Im Gegensatz dazu verwaltet ein Repository die zur Entwicklungszeit anfallenden Artefakte (Services, Prozesse, Datenschemata und Dokumentation). [Jos07] unterscheidet zwischen einer Registry für Metadaten und einem Repository für Software-Artefakte. Hersteller von IT-Infrastrukturen folgen dieser Aufteilung [DRS+07, HR08]. Wir unterscheiden in der vorgestellten IT-Infrastruktur zwischen fachlichem Teil für die Dokumentation von Fachprozess-, Service-Beschreibungen und Metadaten sowie technischem Teil für die Bereitstellung von zur Ausführungszeit benötigten Informationen.

Zusammengefasst sind sowohl Repository als auch Registry für die Verwaltung von Objekten und Metadaten zuständig. Der größte Unterschied besteht darin, dass ein Repository auch Berichts-

und Suchfunktionen anbietet, während bei einer Registry die Performance zur Laufzeit vordergründig ist. Für die erfolgreiche Umsetzung einer SOA wird eine Kombination aus Registry und Repository benötigt.

4.4. Zusammenfassung

Der Weg zu einer Service-orientierten IT-Infrastruktur für die technische Realisierung flexibler Prozessapplikationen gestaltet sich schwierig. Damit eine Strukturierung und Zuordnung der in dieser Arbeit entwickelten Konzepte und Methoden möglich wird, muss zunächst beschrieben werden, welche Komponenten einer IT-Infrastruktur in einer SOA notwendig sind.

Dazu wurde ein Rahmenwerk vorgestellt, das in unterschiedlichen Ausbaustufen eine Referenz für eine Service-orientierte IT-Infrastrukur definiert. Unternehmen sollen in der Lage sein, Fachprozesse effizient umzusetzen und in einer SOA, selbst bei häufigen Änderungen, flexibel zu bleiben, etwa wenn sich eine Service-Lokation ändert. Das Rahmenwerk betrachtet Komponenten zur Verwaltung von Services, Prozessen und Geschäftsregeln in verschiedenen Ausbaustufen. Ferner wird sichergestellt, dass Abhängigkeiten zwischen den Komponenten bewertbar sind.

SOA-Hersteller bieten mächtige SW-Werkzeuge an, die mittlerweile einen ausgereiften Stand erreicht haben. Was noch fehlt, ist eine konzeptionelle Sicht, die das Zusammenspiel der einzelnen Komponenten in einer IT-Infrastruktur betrachtet. Dies ist aufgrund fehlender Richtlinien und Standards aber schwierig. Das Rahmenwerk ermöglicht es den Herstellern, die Werkzeugfunktionalität oder Integrationsfähigkeit ihrer Produkte zu erhöhen.

Teil II

Konzept

5

Maßnahmen zur Flexibilisierung Service-orientierter Architekturen

Die grundlegenden Anforderungen an eine flexible, service- und prozessorientierte Architektur wurden in Kapitel 3 vorgestellt. Davon ausgehend wurde der Stand der Technik diskutiert und bewertet. Dabei hat sich gezeigt, dass aktuelle Ansätze den identifizierten Anforderungen nicht gerecht werden.

Eine detaillierte Analyse von Anwendungsbeispielen hat bestätigt, dass Flexibilität eines der Hauptziele einer SOA ist (vgl. Abschnitt 3.1). Konkret ging hervor, dass sich der Bedarf nach Flexibilität in einer SOA primär auf die Anpassungsfähigkeit von IT-Applikationen sowie deren Reaktionsfähigkeit auf neue bzw. geänderte fachliche und technische Anforderungen bezieht. Dieser Flexibilitätsbedarf wurde durch eine umfassende Literaturstudie und Analyse bestehender Software-Werkzeuge bestätigt. Durch eine zusätzliche Analyse realer IT-Projekte wurden weitere Kriterien für die Flexibilisierung einer SOA identifiziert. In diesem Kapitel werden darauf basierende Flexibilisierungsmaßnahmen diskutiert. Es wird einerseits gezeigt, welche bekannten Konzepte zur Erhöhung der Flexibilität service- und prozessorientierter Applikationen in einer SOA beitragen, andererseits werden offene Fragestellungen erörtert, für die noch keine angemessenen Technologieangebote existieren.

Dieses Kapitel betrachtet wichtige Maßnahmen, um die Flexibilität in einer SOA zu verbessern [BBR11c, BBR11b]. Abschnitt 5.1 stellt zunächst ein charakteristisches Anwendungsszenario vor, entlang dessen wir die verschiedenen Maßnahmen illustrieren. Anschließend wird in Abschnitt 5.2 ein Softwareentwicklungsprozess vorgestellt. Dieser wird phasenweise beschrieben, wobei die Flexibilität bei der Neuentwicklung service- und prozessorientierter Applikationen – in der Folge *Prozessapplikationen* genannt – sowie der Umgang mit späteren Änderungen der Prozessapplikation im Vordergund stehen. Zusätzlich beschreibt Abschnitt 5.3 den Umgang mit Änderungen, die in der SOA-Umgebung der Prozessapplikation stattfinden, die also außerhalb der eigentlich betrachteten Prozessapplikation liegen. Abschnitt 5.4 diskutiert verwandte Arbeiten, bevor das

Kapitel mit einer Zusammenfassung und einem Ausblick in Abschnitt 5.5 schließt.

5.1. Motivation

Um von IT-Seite möglichst flexibel auf fachliche Änderungen reagieren zu können, muss das Zusammenspiel der an der Entwicklung, dem Betrieb und der Wartung einer service- und prozessorientierten Applikation beteiligten Rollen gut organisiert sein. Dazu gehört insbesondere eine möglichst optimale Integration von Fach- und IT-Bereichen. Diese soll durch einen optimierten Softwareentwicklungsprozess, der fachliche Anforderungen schnell, vollständig und unverfälscht in die IT-Implementierung überführt, erreicht werden (vgl. Anforderung 3.1).

In einer SOA existieren einerseits Services, Prozesse, Geschäftsregeln und Organisationsmodelle sowie andererseits eine Reihe von Infrastrukturkomponenten, etwa Appikationsserver oder Prozess-Management-Systeme [BBP09]. Entsprechend Anforderung 3.5 (vgl. Abschnitt 3.3) muss auf Änderungen der von einer Prozessapplikation konsumierten SOA-Artefakte geeignet reagiert werden. Beispielsweise sollten vor Außerbetriebnahme eines Services die jeweiligen Service-Konsumenten informiert werden. Vor allem Prozessapplikationen müssen flexibel gestaltet werden, so dass auf Änderungen, wie die Außerbetriebnahme eines Services, mit möglichst geringem Anpassungsaufwand reagiert werden kann [DRR11, RRD09]. Flexibilität beschreibt 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 Organisationsmodelle.

Wir skizzieren zunächst ein typisches Anwendungsszenario aus der Automobilindustrie, entlang dessen Maßnahmen zur Erhöhung der Flexibilität in einer SOA erörtert werden. Das in Abbildung 5.1 dargestellte Szenario zeigt einen abstrakten Fachprozess für das Änderungsmanagement in der Fahrzeugentwicklung (vgl. Abbildung 4.1). Dieser stark vereinfacht dargestellte Prozess stellt sicher, dass Änderungsvorhaben an Bauteilen vor ihrer eigentlichen Umsetzung bewertet, genehmigt und dokumentiert werden.

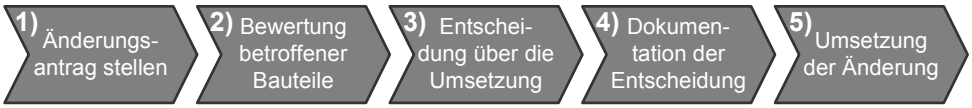


Abbildung 5.1.: Fachprozess zum Änderungsmanagement

Beispiel 5.1 (Änderungsmanagement)
Ein Änderungsvorhaben wird in Fachprozessschritt (1) angelegt, indem die Änderung in Form eines Antrages beschrieben wird. Anschließend wird für die betroffenen Bauteile eine Stellungnahme eingeholt (2). Insbesondere werden die technische Realisierbarkeit der Änderung sowie die aus der Anpassung des jeweiligen Bauteils resultierenden Kosten bewertet. Davon abhängig wird entschieden, ob das Änderungsvorhaben umgesetzt wird (3). Die Entscheidung über die Umsetzung des Änderungsvorhabens wird in Fachprozessschritt (4) dokumentiert, bevor die Änderung, sofern sie genehmigt wurde, tatsächlich umgesetzt wird (5).

5.2. Entwicklung von Prozessapplikationen

Wie muss nun bei der Entwicklung von Prozessapplikationen vorgegangen werden, um möglichst schnell und kostengünstig von fachlichen Anforderungen zur Prozessapplikation zu gelangen. Ausgehend von fachlichen Anforderungen werden Fachprozesse 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 5.2 dargestellte Phasenmodell skizziert einen idealtypischen Softwareentwicklungsprozess für Prozessapplikationen auf abstrakter Ebene. Dieser ist angelehnt an die in der Literatur im Umfeld des Geschäftsprozessmanagements und der Service-Orientierung diskutierten Softwareentwicklungsprozesse [PH07, Wes07].

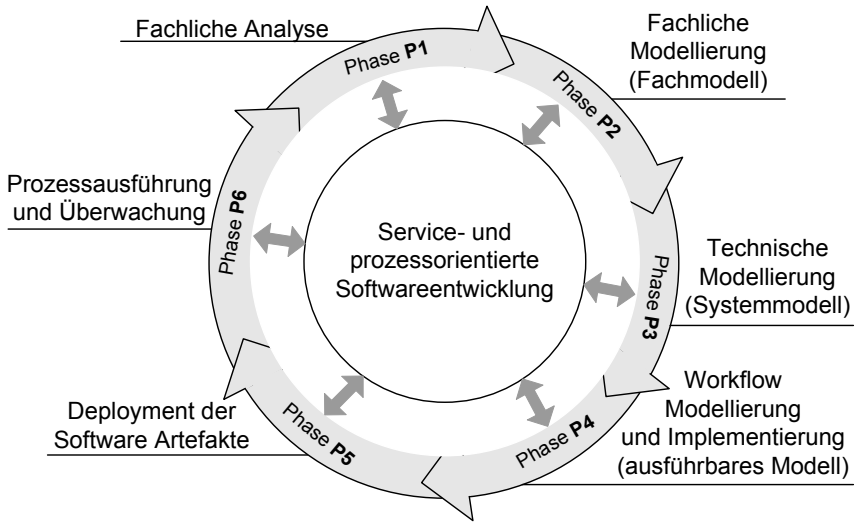


Abbildung 5.2.: Service- und prozessorientierte Softwareentwicklung

Ausgehend von einer Analyse wurden Fachprozesse modelliert und anschließend durch einen Verantwortlichen des IT-Bereichs technisch spezifiziert und implementiert. Dies wird unter Verwendung von in einer SOA angebotenen Services realisiert. Die Ausführung technischer Prozesse findet mittels eines Prozess-Management-Systems statt, das zugleich die Grundlage für eine Überwachung solcher darstellt.

Im Folgenden werden die einzelnen Phasen entlang des Anwendungsszenarios aus Abbildung 5.1 erörtert. Wir betrachten dazu sowohl die Neuentwicklung von Prozessapplikationen als auch deren spätere Änderung [WSR09].

5.2.1. Beschreibung fachlicher Anforderungen (P1: Fachliche Analyse)

In der ersten Phase des Entwicklungsprozesses werden (fachliche) Anforderungen dokumentiert [Rup07]. Diese beschreiben funktionale und nicht-funktionale Vorgaben an die zu entwickelnde Prozessapplikation. Zur Erfassung und Beschreibung dieser Anforderungen werden Software-Werkzeuge (z.B. Rational RequisitePro [Zie07] oder DOORS [Cor10]) eingesetzt, mittels denen

ein *Requirements Engineer* die Anforderungen (meist textuell) dokumentiert und mit den Fachbereichen abstimmt. Für das beschriebene Anwendungsszenario etwa könnte sich der in Tabelle 5.1 dargestellte Anforderungskatalog ergeben. Dieser zeigt lediglich ausgewählte (funktionale) Anforderungen und bietet keine vollständige Spezifikation.

Anforderung	**Beschreibung**
Anforderung 1	Änderungsvorhaben an Bauteilen müssen bewertet werden.
Anforderung 2	Bauteile müssen explizit genehmigt werden.
Anforderung 3	Änderungsvorhaben werden in Form eines Änderungsantrages angelegt.
Anforderung 4	Bauteildaten müssen aus dem Produktdaten-Management-System importiert werden.
Anforderung 5	Prüfung auf technische Realisierbarkeit und Stellungnahme erfolgt durch den entsprechenden Konstrukteur.
Anforderung 6	Baureihenverantwortlicher entscheidet über die Umsetzung eines Änderungsvorhabens.
Anforderung 7	Die Entscheidung über die Umsetzung muss in den davon betroffenen Systemen dokumentiert werden.
Anforderung 8	Die Durchführung von Einzelbewertungen für eine beliebige Anzahl von Bauteilen muss möglich sein.

Tabelle 5.1.: Anforderungskatalog für den Änderungsprozess (Ausschnitt)

Maßnahme 1: Beziehungen zwischen Anforderungen und Fachprozess

Neben den Anforderungen sollte dokumentiert werden, wie diese konkret umgesetzt werden sollen [GF94]. Dazu müssen die Beziehungen zwischen fachlichen Anforderungen und ihren Umsetzungsobjekten (z.B. Prozessschritte, Datenobjekte oder Bearbeiter) im zugehörigen Fachprozess explizit dokumentiert und konsistent verwaltet werden. Nur dann ist nachvollziehbar, welche Fachprozesse und Umsetzungsobjekte von zukünftigen Änderungen fachlicher Anforderungen betroffen sein werden. Das folgende Beispiel zeigt, wie die Beziehung zwischen Anforderung 3 (vgl. Tabelle 5.1) und dem Fachprozess aus Abbildung 5.4 dokumentiert werden kann.

Beispiel 5.2 (Dokumentation von Beziehungen)
Anforderung 3 erfordert, dass für jedes Änderungsvorhaben ein Änderungsantrag erstellt wird. Dies wird dadurch realisiert, dass ein entsprechender Prozessschritt zur Eingabe eines Änderungsvorhabens im Änderungsprozess modelliert wird (vgl. Abbildung 5.4). Anschließend speichert ein Datenobjekt die Informationen über das Änderungsvorhaben. Abbildung 5.3 zeigt die Beziehung zwischen Anforderung 3 und den zugehörigen Umsetzungsobjekten im Änderungsprozess.

Durch Dokumentation solcher Beziehungen lassen sich Änderungen an fachlichen Anforderungen

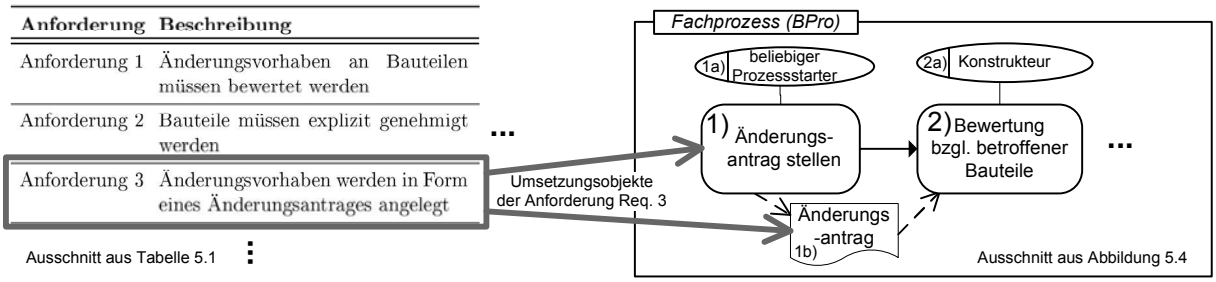


Abbildung 5.3.: Beziehung zwischen Anforderungen und Umsetzungsobjekt

schneller umsetzen, da die davon betroffenen Objekte im zugehörigen Fachprozess bekannt sind. Zudem wird sichergestellt, dass alle fachlichen Anforderungen tatsächlich durch den entsprechenden Fachprozess realisiert werden.

5.2.2. Definition von Fachprozessen (P2: Fachliche Modellierung)

Die Modellierung von Fachprozessen erfolgt typischerweise durch die Fachabteilungen bzw. -modellierer. Diese bilden fachliche Anforderungen des Unternehmens auf Fachprozessmodelle ab. Dazu werden Prozessschritte, deren Abhängigkeiten (z.B. Reihenfolge) und involvierte IT-Systeme modelliert sowie Verantwortlichkeiten festgelegt. Aufgrund der meist fehlenden bzw. mangelnden IT-Kompetenz der Fachmodellierer werden hierzu einfache graphische Notationen verwendet. In der Praxis verbreitet sind z.B. *erweiterte Ereignisgesteuerte Prozess-Ketten (eEPK)* [KNS92] und *Business Process Model and Notation (BPMN)* [Obj09]. Beispiel 5.3 zeigt einen modellierten Änderungsprozess (vgl. Abbildung 5.4).

Beispiel 5.3 (Fachprozessmodellierung)
Im Fachprozessschritt (1) wird ein Änderungsvorhaben angelegt, indem die Änderung in Form eines Änderungsantrages (1b) beschrieben wird. Der Fachprozessschritt (1) und das Datenobjekt Änderungsantrag (1b) realisieren zusammen Anforderung 3 (vgl. Abbildung 5.1), d.h. Anforderung 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 Fachprozessschritt (2) für alle vom Änderungsvorhaben betroffenen Bauteile die entsprechenden Bauteildaten aus dem Produktdaten-Management-System (2b) importiert (vgl. Anforderung 4) und jeweils von einen Konstrukteur (2a) bewertet (vgl. Anforderung 1 und Anforderung 5). Anschließend entscheidet in (3) ein Baureihenverantwortlicher über die Umsetzung des Änderungsvorhabens (Anforderung 6). Diese Entscheidung muss in allen betroffenen Systemen (4) dokumentiert werden (7), bevor die Änderung umgesetzt wird (5).

In dieser frühen Phase P2 des Softwareentwicklungsprozesses sind meist noch nicht alle Aspekte im Detail bekannt, sollen noch nicht erfasst werden, oder sind Fachmodellierern aufgrund mangelnder IT-Kompetenz fremd. Deshalb ist das Fachprozessmodell an verschiedenen Stellen vage bzw. offen und unvollständig. Dies betrifft neben der Struktur von Fachprozessmodellen

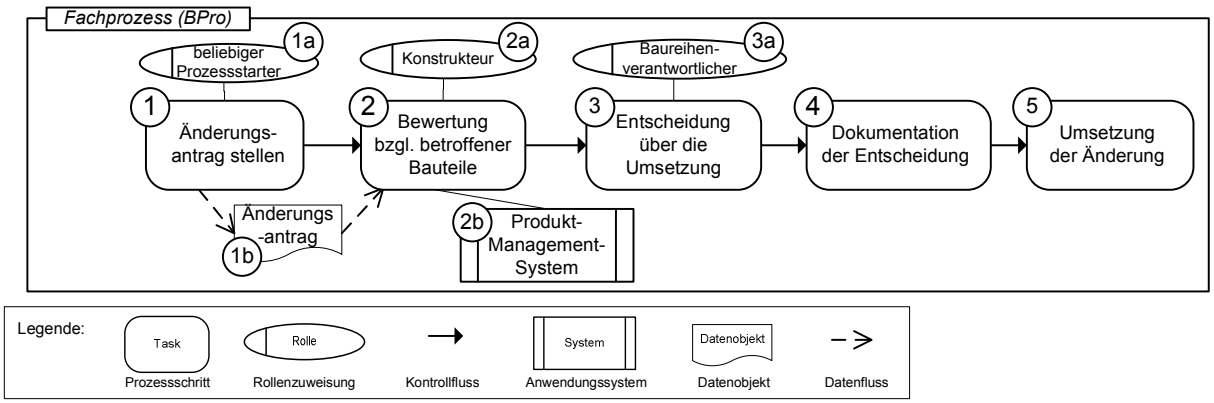


Abbildung 5.4.: Modellierter Änderungsprozess

(d.h. Fachprozessschritte und den Kontrollfluss zwischen ihnen) noch weitere Aspekte, wie Datenstrukturen oder Bearbeiterzuordnungen [Rei00, RR09, RR08, Wes07, WRR08]. Diese vage Beschreibung wird zum Problem, wenn die Fachprozessmodelle zur technischen Umsetzung an einen IT-Bereich übergeben werden und dort hohe Aufwände für die Interpretation entstehen. Deshalb sind Maßnahmen notwendig, die eine frühe Modellierung von Informationen ermöglichen und dadurch die Verwendung dieser in einer späteren Phase des Entwicklungsprozesses (vgl. Abbildung 5.2) arrangieren, ohne im Fachbereich nachfragen zu müssen.

Maßnahme 2: Frühe Modellierung von Informationen durch Frontloading

Als *Frontloading* bezeichnet wird die frühzeitige Modellierung von Information, d.h. diese soll während der Fachprozessmodellierung möglichst vollständig beschrieben werden (siehe Kapitel 7). Somit lassen sich Aspekte einer späteren IT-Implementierung früh dokumentieren und mit den Verantwortlichen aus dem Fachbereich abstimmen. Dazu gehören Informationen, die erstmalig bei der Fachprozessmodellierung dokumentiert werden. So sollte betrachtet werden, welche Personen bzw. Rollen einen bestimmten Schritt des Fachprozesses bearbeiten sollen. Hierbei genügt es nicht, lediglich eine Rolle (bspw. Kreditprüfer) zuzuordnen. Stattdessen müssen die *Bearbeiterzuordnungen* auf fachlicher Ebene genau beschrieben werden, um Aufwände für die Implementierungen gering zu halten (vgl. Abschnitt 7.3). Dies muss ausreichend detailliert und vollständig erfolgen. Dennoch sollte diese Aufgabe durch Fachanwender bewerkstelligt werden können. Ein Beispiel für eine Bearbeiterzuordnung kann folgendermaßen aussehen:

Beispiel 5.4 (Frontloading)
Die Entscheidung über die Umsetzung eines Änderungsantrags soll durch eine Person bearbeitet werden, die für die Baureihe des geänderten Bauteils verantwortlich ist. Ferner darf diese Person nicht mit der Person übereinstimmen, die den Änderungsantrag gestellt hat.

Solche Bearbeiterzuordnungen werden meist auf fachlicher Ebene und nicht in Fachprozessmodellen erfasst und dokumentiert. Damit Bearbeiterzuordnungen tatsächlich in die Implementierung übernommen werden können, müssen sie jedoch hinreichend detailliert beschrieben werden

[Bau09]. Ein Beispiel dafür könnte folgendermaßen aussehen: *(Rolle = Baureihenverantwortlicher) ∧ (Baureihe = $ÄnderungsantragStellen.Baureihe$) ∧ (Person ≠ $ÄnderungsantragStellen.Person$)*. Diese technisch beschriebene Bearbeiterzuordnung kann direkt in das Systemmodell übernommen und dort als Basis für die Implementierung eingesetzt werden. Für Fachmodellierer ist es jedoch schwierig eine solche Beschreibung zu erstellen, da ihnen meist der IT-Hintergrund fehlt. Eine Schwierigkeit ist nun, eine Beschreibungsform und Methodik zur Dokumentation von Bearbeiterzuordnungen zu finden, die für Fachmodellierer verständlich und zugleich vollständig genug ist, um zur Weiterverarbeitung im Systemmodell zu dienen.

Nicht nur die Dokumentation von Bearbeiterzuordnungen, sondern auch eine detaillierte *Informations- und Datenmodellierung* kann bereits auf fachlicher Ebene (P2) sinnvoll sein, um später bei der Implementierung des Prozesses bestimmte Information (teil)automatisiert ableiten zu können, d.h. ohne zusätzliche Analyse-Phase im Fachbereich.

Außerdem sollen Informationen zum Aussehen von späteren *(Benutzer-) Masken* (insbesondere hilfreich für die Implementierung in Phase P5) frühzeitig im Fachprozessmodell dokumentiert werden können. Des Weiteren sind Informationen über *Ausnahmesituationen* im Fachprozess notwendig, um geeignete Reaktionsmöglichkeiten bereits bei der Modellierung festzulegen, etwa bei fehlgeschlagenen Service-Aufrufen oder unzureichender Datenqualität nach einer Benutzereingabe.

Ein weiterer Frontloading-Aspekt betrifft die *Markierung potentieller Flexibilitätsstellen* im Fachmodell. D.h. es sollte bereits bei der Modellierung des Fachmodells festgelegt werden, welche Bereiche des Modells in späteren Phasen möglichst flexibel zu implementieren sind. Entsprechende Markierungen können an unterschiedlichen Objekten, etwa Fachprozessschritten, Verzweigungen, Service-Aufrufen oder Bearbeiterzuordnungen erfolgen. Beispiel 5.5 beschreibt die Kennzeichnung von Flexibilitätsstellen für einen Fachprozessschritt.

Beispiel 5.5 (Frontloading)
Anforderung 8 unseres Beispielprozesses (vgl. Tabelle 5.1) fordert, dass Einzelbewertungen für eine beliebige Anzahl von Bauteilen durchgeführt werden können. Eine solche Information kann durch eine explizite Kennzeichnung (etwa als textuelle Beschreibung) am Fachprozessschritt (2) dokumentiert werden (vgl. Abbildung 5.4). Weitere Beispiele für Kennzeichnungen im Fachprozessmodell sind Markierungen an Verzweigungsbedingungen für häufig anzupassende Schwellwerte oder Service-Zuweisungen für das späte Binden von Service-Instanzen [Nak02, DEV+06].

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

Maßnahme 3: Berücksichtigung von Informationen durch Look-ahead

Fachprozesse werden oft ohne Rücksichtnahme auf die bereits existierende IT-Umgebung modelliert. Informationen etwa über bereits vorhandene Services, Datenobjekte oder Organisationsmodelle werden deshalb nicht immer vollständig dokumentiert. Um einen möglichst hohen Grad

an Wiederverwendung zu erreichen, ist es wichtig, das Fachprozessmodell so zu gestalten, dass existierende Services tatsächlich wiederverwendet werden können. Meist wird dies jedoch nicht berücksichtigt, da Fachmodellierern keine Information über deren Existenz vorliegt und diese nur schwer beschaffbar ist. Oftmals wird nicht explizit nach Services gesucht oder auf deren Verwendung verzichtet, insbesondere wenn die bereitgestellten Funktionen nicht exakt passen. Dadurch wird die Implementierung der Fachprozesse in Phase 4 aufwendiger.

Werden Services verwendet, die nicht die exakt gewünschte Funktionalität realisieren, kann ggf. der Fachprozess geringfügig angepasst werden, um den existierenden Service dennoch wiederzuverwenden. Dies spart Zeit und Aufwand bei der Implementierung. Um eine solche Wiederverwendung zu fördern, müssen Governance-Prozesse etabliert und die Services für Fachmodellierer zugänglich gemacht werden.

Neben der Berücksichtigung von Services (*Service-Look-ahead*) sind weitere Aspekte relevant. So sollten Rollen, die im Fachprozessmodell verwendet werden, auch tatsächlich im Organisationsmodell des Unternehmens existieren. Bereits im Unternehmen existierende Datenobjekte, die als Eingabe oder Ausgabe von Fachprozessschritten und Services verwendet werden, sollten ebenfalls im Fachprozessmodell vorliegen, etwa um redundante Implementierungen zu vermeiden.

Beispiel 5.6 (Look-ahead)
Für Fachprozessschritt (2) unseres Beispiels (vgl. Abbildung 5.4) sucht der Fachmodellierer im SOA-Repository nach einem bereits dokumentierten fachlichen Service, der Bauteilinformationen bereitstellt. Der Fachmodellierer hinterlegt den fachlichen Service am Fachprozessschritt (2), etwa in Form eines Textdokuments, einer Referenz auf das fachliche Service-Modell oder einer externen Referenz auf ein SOA-Repository-Objekt.

Frontloading und Look-ahead werden in Kapitel 7 im Detail behandelt.

5.2.3. IT-orientierte Sichtweise auf Fachprozesse (P3: Technische Modellierung)

Informationen auf fachlicher Ebene werden oftmals auf unterschiedliche Weise dokumentiert, als Informationen auf technischer Ebene. Fachanwender verwenden meist andere Modelltypen für die Prozessdokumentation als IT-Implementierer: Häufig modellieren Fachanwender ihre Fachprozesse mit gängigen Fachprozessmodellierungswerkzeugen (z.B. ARIS [Sch01] als EPK), wohingegen IT-Implementierer CASE-Tools (z.B. mit UML-Unterstützung) verwenden. Es ist meist kein durchgängiges Vorgehen zur Abbildung fachlicher Modelle (Phase P2) auf deren technische Implementierung (Phase P4) etabliert. Dies erschwert die Zusammenarbeit der beiden Rollen und Bereiche. Um diese inhaltliche Distanz zu schließen, wird ein zusätzliches Modell eingeführt (vgl. Abschnitt 2.2.4), das die Überführung fachlicher Modelle in eine IT-Realisierung erleichtert. Ziel ist es, Informationen nachvollziehbar und verständlich (erklärbar) für den Fachbereich sowie formal und detailliert genug für den IT-Bereich zu spezifizieren (vgl. Anforderung 3.2 in Abschnitt 3.3). Die Verantwortung für die Erstellung des Systemprozesses liegt beim IT-Bereich. Die zugehörigen Inhalte sind dieselben wie im Fachprozessmodell. Allerdings müssen diese ausreichend detailliert, vollständig erfasst und formal beschrieben sein, um die Basis einer plattformunabhängigen IT-Spezifikation bilden zu können.

Maßnahme 4: Nachvollziehbarkeit zwischen Modellierungsebenen

Das Systemmodell detailliert diejenigen Aspekte aus dem fachlichen Modell, die technisch unterstützt werden sollen. Das sind beispielsweise einzelne Fachprozessschritte (vgl. Definitionen 2.3 und 2.6) des Fachprozessmodells, die *direkt* in Systemprozessschritte überführt werden:

Beispiel 5.7 (Nachvollziehbarkeit)
Die Benutzerinteraktion zum Stellen eines Änderungsantrages (Fachprozessschritt (1) in Abbildung 5.5) kann beispielsweise als sog. *Human Task* mit dem zu den IT-Vorgaben konformen Namen **HT**_*request*_*change* realisiert werden.

Die Menge der Fachprozessschritte kann zudem in mehrere (technische) Systemprozessschritte *aufgespalten* und ggf. durch zusätzliche Ablauflogik detailliert werden:

Beispiel 5.8 (Nachvollziehbarkeit)
Fachprozessschritt (4) in Abbildung 5.5 ist ein Beispiel für eine Aufspaltung eines Fachprozessschrittes beim Übergang von einem Fachprozessprozess in einen ausführbaren Prozess:

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*). Hinzu kommt die Ablauflogik (AND-Join und -Split) zur Parallelisierung dieser beider Systemprozessschritte.

Ferner wird dokumentiert, welche Fachprozessschritte im Systemprozess *weggelassen* werden sollen, weil keine IT-Umsetzung gewünscht ist (vgl. Fachprozessschritt (5) in Abbildung 5.5).

Die Durchführung von Einzelbewertungen für Bauteile liefert ein Beispiel für die Umsetzung der Anforderung 8 (Tabelle 5.1) im Systemprozess:

Beispiel 5.9 (Nachvollziehbarkeit)
Durch Einfügen einer Multi-Instanz Aktivität [AHKB03, RR06] wird dafür gesorgt, dass für alle von einer Änderung betroffenen Bauteile eine Einzelbewertung durch den entsprechenden Teileverantwortlichen erfolgt.

Aufgrund dieser mitunter komplexen Umstrukturierungen beim Übergang vom Fach- zum Systemprozess ist es für Fachmodellierer keineswegs trivial, die Objekte aus dem Fachprozess mit Objekten im Systemprozess korrekt in Beziehung zu setzen. Herausfordernd etwa ist die Art der Dokumentation der Beziehungen, die zum einen Fachmodellierer nicht überfordern darf und zum anderen für Endanwender verständlich sein sollte. Ein weiteres Problem entsteht dadurch, dass die Metamodelle heutiger Geschäftsprozessmodellierungswerkzeuge eine solche Dokumentation nicht explizit unterstützen. Eine detaillierte Betrachtung zur Dokumentation der Beziehungen zwischen Fach- und Systemprozess erfolgt in Kapitel 6.

Bezogen auf diese Problemstellung ermöglicht ENPROSO die Nachvollziehbarkeit der Beziehungen zwischen Fachprozessschritten 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 konkrete Prozessschritte identifizieren zu können, etwa wenn ein laufender Service abgeschaltet

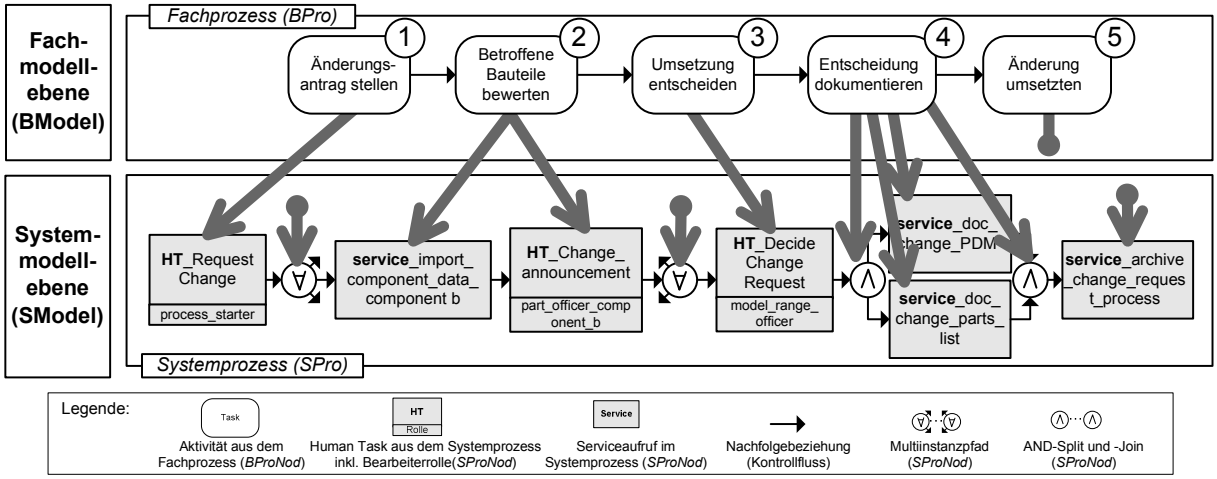


Abbildung 5.5.: Überführung fachlicher Aktivitäten in das Systemmodell

werden soll. Nur dann ist es möglich, rasch auf Änderungen zu reagieren (vgl. Anforderung 3.5 aus Abschnitt 3.3).

Neben der Detaillierung von Fachprozessen, ist eine zu entwickelnde Prozessapplikation so zu gestalten, dass spätere Änderungen wenig Aufwand verursachen. So können bereits in dieser frühen Phase (vgl. Phase P3 in Abbildung 5.2) die mittels *Frontloading* beschriebenen Informationen berücksichtigt werden. Für sich häufig ändernde oder komplexe Verzweigungsbedingungen in Fachprozessen sollen Geschäftsregeln so definiert werden, dass sie unabhängig vom Fachprozess verändert werden können [BBP09]. Im Systemprozess werden dadurch bestimmte Verzweigungsentscheidungen durch Geschäftsregeln von der Geschäftsprozesslogik entkoppelt.

Maßnahme 5: Analyse von Inkonsistenzen

Spätere Änderungen an fachlichen Anforderungen, etwa infolge einer Fachprozessoptimierung oder einer erforderlichen Prozessanpassung aufgrund von Änderungen gesetzlicher Rahmenbedingungen, wirken sich auf die Fachprozesse (Phase P2), die zugehörigen Systemprozesse (P3) und die ausführbaren Prozesse (P4) aus. Diese müssen bei Änderungen angepasst werden.

Beispiel 5.10 (Neue fachliche Anforderung)
Änderungsvorhaben betreffen häufig viele Bauteile. Da die Auswirkung eines solchen Vorhabens auf jedes einzelne Bauteil zu bewerten ist (vgl. (2) aus Abbildung 5.6), 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 an den Änderungsprozess:

Anforderung 9 beschreibt die Notwendigkeit eines Vorfilters, der anzuwenden ist, bevor die Bewertung betroffener Bauteile durchgeführt wird. Die Verantwortung dafür liegt beim Vorgesetzten des Antragstellers.

Im Folgenden wird beschrieben, wie sich die neu hinzugekommene Anforderung 9 im Änderungsprozess umsetzen lässt. Abbildung 5.6 und 5.7 zeigen das resultierende Ergebnis:

Schritt A: Anforderung 9 wird durch einen zusätzlichen Fachprozessschritt (6) im Änderungsprozess zuzüglich einer Abbruchbedingung (6a) realisiert. Dadurch entsteht eine neue Version des Fachprozesses (vgl. Abbildung 5.6).

Schritt B: Aufgrund der nachvollziehbar dokumentierten Beziehungen zwischen Geschäft- und Systemprozess (vgl. Maßnahme 4) lassen sich Inkonsistenzen erkennen. Dabei werden die Objekte im Fachprozess identifiziert, für die keine Objekte im zugehörigen Systemprozess existieren, d.h. für die keine Beziehung dokumentiert ist (vgl. (6) und (6a)).

Schritt C: Identifizierte Inkonsistenzen werden dadurch behoben, dass fehlende Objekte im Systemprozess ergänzt und die Beziehungen zu korrespondierenden Objekten im Fachprozess dokumentiert werden (vgl. Abbildung 5.7).

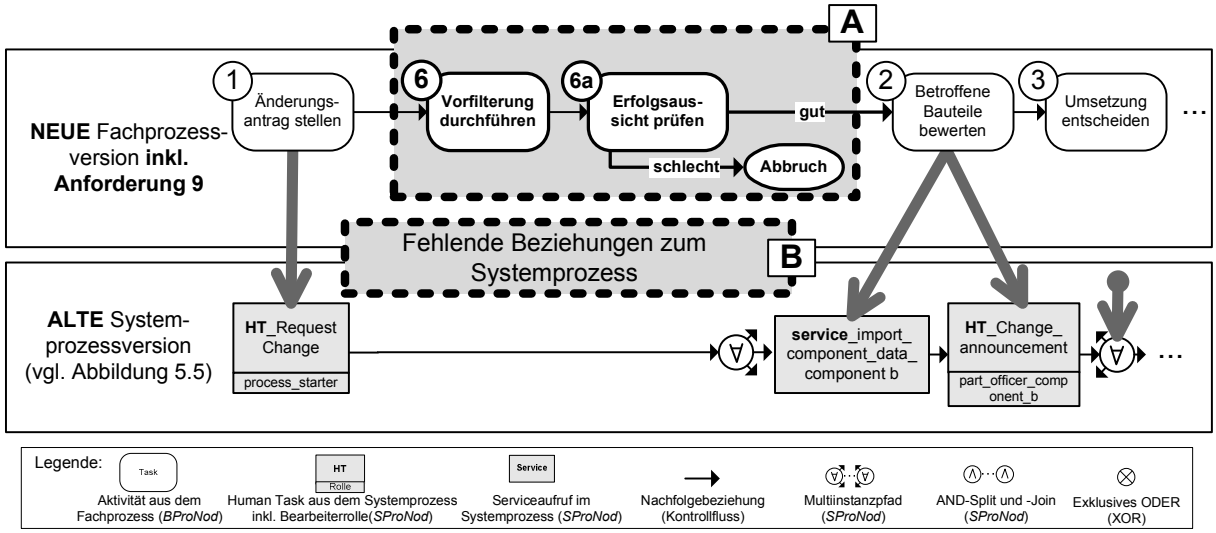


Abbildung 5.6.: Neue Fachprozessversion inklusive Anforderung 9

Die Speicherung der Beziehungen zwischen den Schritten eines Fachprozessmodells auf der einen Seite und den Aktivitäten des zugehörigen Systemprozessmodells auf der anderen Seite erhöht die Flexibilität bei späteren Anpassungen. D.h. durch Änderungen verursachte Abhängigkeitsverletzungen können erkannt und schneller behoben werden, da die Beziehungen zwischen Objekten des Fach- und Systemprozesses explizit gespeichert werden. So können Änderungen fachlicher Anforderungen bzw. Änderungen der Fachprozesse rasch in die IT-Implementierung überführt werden. Andererseits lassen sich Änderungen der Umgebung der Prozessapplikation (vgl. Abschnitt 5.3), etwa die Abschaltung eines Services, bis zur verantwortlichen Person im Fachbereich zurückverfolgen. Diese kann über die Änderung informiert werden und aus fachlicher Sicht über eine geeignete Reaktion entscheiden.

Es bleibt zu diskutieren, wie Inkonsistenzen erkannt und aufgelöst werden können (vgl. Anforderung 3.4 in Abschnitt 3.3). Die Identifikation von Inkonsistenzen alleine reicht nicht aus, um

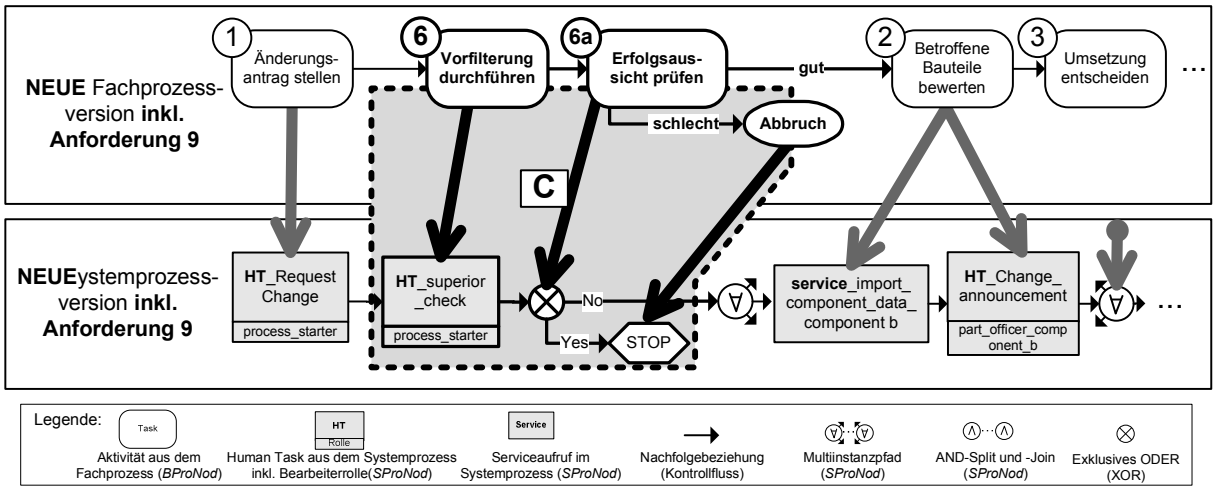


Abbildung 5.7.: Neue Systemprozessversion inklusive dokumentierter Beziehungen

den verantwortlichen Modellierern passende Hinweise für deren Auflösung zu geben. Ein Grund ist, dass unbekannt ist, welche Art von Änderung in welchem Prozess (Fach- oder Systemprozess) durchgeführt worden ist. Details zur Erkennung und Auflösung von Inkonsistenzen zwischen Fach- und Systemprozess werden in Kapitel 6 gegeben.

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

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

Die Überführung der Information aus dem Systemprozess in einen ausführbaren Prozess 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 Systemprozessschritten), die dort verwendeten technischen Services, Benutzermasken und Geschäftsregeln. Der Systemprozess wird aus dem generischen Format in ein zielplattformabhängiges Format (z.B. WS-BPEL) übertragen. Neben Benutzerinteraktionen (Human Tasks) und BPEL-Nachrichten müssen noch geeignete BPEL-Konstrukte (z.B. While-Schleifen, Parallel-ForEach-Konstrukte) 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. Maßnahmen 6 und 7 ermöglichen eine solche Dynamik.

Maßnahme 6: Flexibilität durch Verwendung von Geschäftsregeln

Geschäftsregeln werden häufig eingesetzt, um Verzweigungsbedingungen in Fachprozessen zu beschreiben und die eigentliche Verzweigungslogik vom Fachprozess zu entkoppeln. Die Flexibilität entsteht durch die Auslagerung der Geschäftsregeln aus dem Fachprozess. Dadurch ist es möglich, sich häufig ändernde oder komplexe Verzweigungsbedingungen unabhängig vom Fachprozess und damit auch unabhängig von der Implementierung anzupassen [BBP09].

Maßnahme 7: Service-Auswahl unter Nutzung von Quality-of-Service-Kriterien

Damit technische Services zur Laufzeit anhand ihrer fachlichen Attribute ausgewählt und gerufen werden können, ist bei ihrer Implementierung darauf zu achten, dass Service-Endpunkte nicht statisch im Code der Prozessapplikation verborgen sind, sondern dynamisch ermittelt werden. Dies kann durch Einsatz eines SOA-Repository (vgl. Kapitel 8) realisiert werden [BTBR10], in dem die Informationen über Service-Endpunkte und -Instanzen abgelegt werden. Eine Service-Instanz beschreibt eine konkrete technische Service-Installation auf einem bestimmten Applikationsserver. Zur Identifikation der richtigen Service-Instanz werden zusätzlich zur Endpunktinformation die Quality-of-Service (QoS)-Attribute aller Service-Instanzen im Repository gespeichert. Im folgenden Beispiel wird gezeigt, wie eine Service-Auswahl basierend auf QoS-Attributen durchgeführt werden kann, ohne die Implementierung des ausführbaren Modells (vgl. Abschnitt 2.7) anpassen zu müssen.

Beispiel 5.11 (Service-Auswahl durch QoS-Attribute)
Die durch den Fachbereich festgelegten QoS-Attribute für Fachprozessschritte 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 die konsumierenden Prozessapplikationen zur Laufzeit unterscheiden können müssen. Dies ist z.B. nützlich, wenn unterschiedliche Prozessapplikationen die gleiche Service-Instanz mit unterschiedlichen QoS-Eigenschaften verwenden wollen. So existiert beispielsweise der in Abbildung 5.8 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 $t_y \leq 25$ms. 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. Abbildung 5.8 beschreibt ein solches Vorgehen:

1) Das QoS-Attribut *Antwortzeit* wird durch den verantwortlichen Modellierer aus dem Fachbereich festgelegt (z.B. „Antwortzeit $\leq 15ms$“), indem das Attribut erzeugt und dem Fachprozessschritt *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 bestimmt.

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 dokumentiert.

5) Wird die Prozessapplikation nun ausgeführt, ermittelt sie den geforderten Endpunkt des Services basierend auf den im Fachprozess definierten QoS-Attribut („Antwortzeit $\leq 15ms$“). Dies erfolgt durch eine Anfrage an das SOA-Repository, das 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. Falls keine Service-Instanz existiert, kann auch kein Service-Endpunkt gefunden werden, und der Aufruf bricht mit einem definierten Fehler-Code ab.

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

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 [EM08].

Die hier beschriebene Maßnahme erhöht die Flexibilität bei Service-Aufrufen entsprechend Anforderung 3.7 aus Abschnitt 3.3.

5.2.5. Deployment (P5: Deployment der Softwareartefakte)

In der Deployment-Phase (vgl. Abbildung 5.2) wird das ausführbare Modell in Betrieb genommen, d.h. die Prozessapplikation wird zur Ausführung auf einem Applikations- oder Prozess-Server [RD00] bereitgestellt. Um eine möglichst hohe Flexibilität bei der Ausführung zu erlangen, ist die Granularität der Deployment-Einheiten von besonderer Bedeutung. Jede aus dem Fachmodell ausgelagerte Funktionalität (z.B. Geschäftsregeln, vgl. Maßnahme 6) wird separat auf einem Applikationsserver installiert und ermöglicht dadurch eine vom Fachprozess unabhängige Anpassung. Konkret bedeutet dies, dass etwa bei Änderungen an Geschäftsregeln das Fachprozess nicht angepasst werden muss (vgl. Anforderung 3.7 aus Abschnitt 3.3). 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 5.2.6) können ohne Unterbrechung das geänderte Verhalten übernehmen.

Maßnahme 8: Entkopplung beim Deployment durch Packaging

Packaging beschreibt, in welche Einheiten eine Prozessapplikation zu „schneiden“ ist, um diese möglichst effizient und flexibel auf einem Applikationsserver betreiben zu können. Informationen aus frühen Phasen (P2, P3) dienen dabei als Indikatoren für eine solche Aufteilung. Dies können z.B. Informationen zu Service-Implementierungen, Bearbeiterzuordnungen, Geschäftsregeln oder

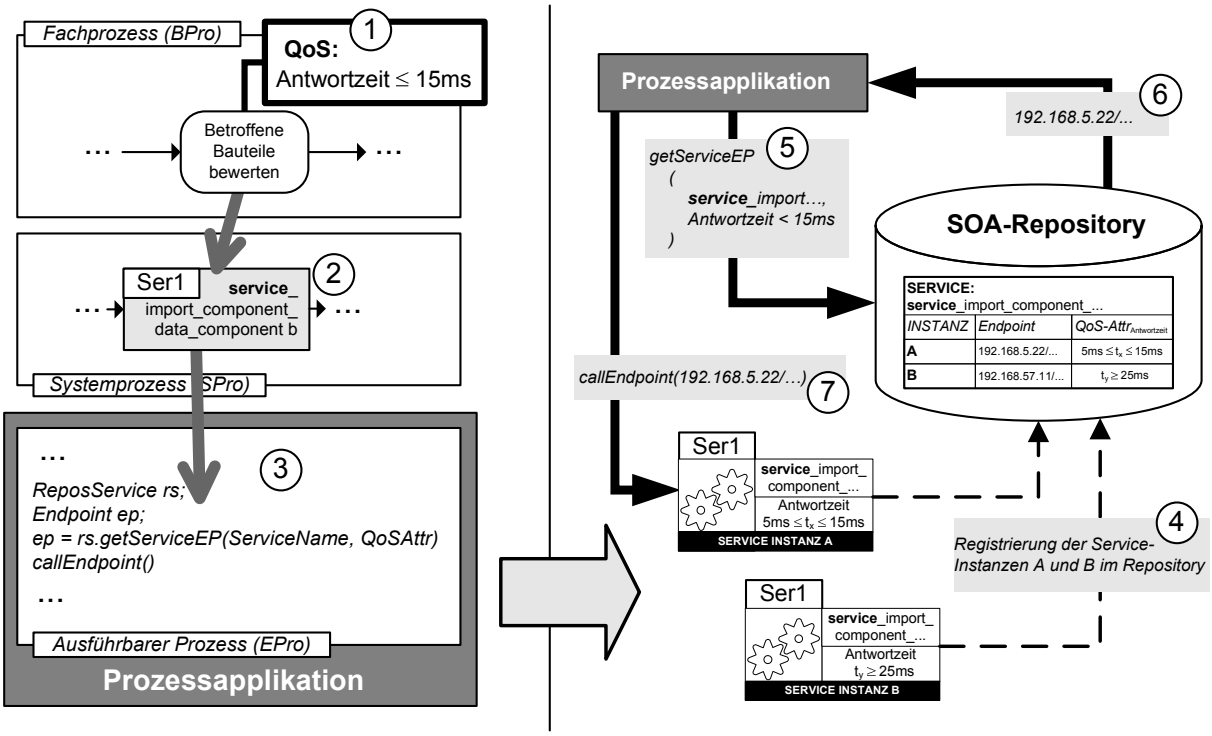


Abbildung 5.8.: Auswahl anhand QoS-Attributen

Sub-Prozessen sein. Solche Artefakte werden unabhängig voneinander auf einer entsprechenden Ausführungskomponente installiert.

5.2.6. Ausführung und Überwachung (P6: Process Execution and Monitoring)

Die Ausführung und Steuerung von Prozessapplikationen wird idealerweise durch eine Prozess-Engine realisiert (vgl. Abbildung 5.2).

Maßnahme 9: Prozesssteuerung durch Prozess-Engine

Funktionen wie Arbeitslistenverwaltung oder Prozess-Recovery sind bereits in den meisten Prozess-Engines verfügbar und müssen daher nicht mehr in der Prozessapplikation realisiert werden. Die Nutzung einer adaptiven Prozess-Engine [DRRM+09, DR09] ermöglicht darüber hinaus Ad-hoc-Instanz-Änderungen und Schemaevolution [RD98, RRD04a, RRD09]. Die Handhabung von Prozessvarianten wird in [Hal10, HBR10a, HBR10b] beschrieben. Der dort entwickelte Provop-Ansatz ermöglicht es, ausgehend von einem Basisprozessmodell Varianten eines bestimmten Prozesstyps abzuleiten und durch einen Prozess-Engine auszuführen. Weitere Ansätze beschreiben spezielle datengetriebene Ausführungsmodelle [MRH07, MRH08, KR11a, KR11b]. Letztere beschäftigen sich etwa mit der Bereitstellung einer integrierten Sichtweise auf Daten

und einer Daten-basierten Prozessmodellierung und -ausführung. Durch Wiederverwendung solcher Grundfunktionen entsteht weniger Implementierungsaufwand für eine Prozessapplikation [RDK+08]. Darüber hinaus realisieren viele Prozess-Engines eine Integration in eine Monitoring-Komponente [BBP09, BRBB09].

Maßnahme 10: Messung fachlicher Kennzahlen

Ein wichtiger Ausführungsaspekt von Prozessen ist die Messung ihrer tatsächlichen Qualität durch Überwachung von Prozessapplikationen. Ziel des fachlichen Monitoring ist die Überwachung der Prozessapplikation zwecks 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 [AAC+08], die sich auf Messstellen in Fachprozessen beziehen. Messstellen beschreiben z.B. Start- und Endzeitpunkte einer Zeitmessung während einer Fachprozesssimulation. 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 KPIs verwendet werden müssen. Die Flexibilität wird dadurch erhöht, dass neue KPIs leichter definierbar sind und vorhandene KPIs einfacher geändert werden können.

5.3. 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 respektive den IT-Infrastrukturkomponenten, auf der die Prozessapplikation betrieben wird, sowie des Organisationsmodells, das zum Beispiel für die Auflösung von Rollen bei der Ausführung von Prozessschritten benötigt wird (vgl. Abbildung 5.9). Eine rasche und effektive Reaktion auf solche Umgebungsänderungen ist unabdingbare Voraussetzung für jede flexible Prozessapplikation (vgl. Anforderung 3.5 in Abschnitt 3.3).

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

5.3.1. Änderung an der Service-Lokation

Eine häufige auftretende Umgebungsänderung ist die Service-Migration: Wird ein Service auf einen anderen Server verschoben (Lokationsänderung), kann er durch konsumierende Prozessapplikationen nicht mehr ohne weiteres gerufen werden, wenn diese Prozessapplikationen nur

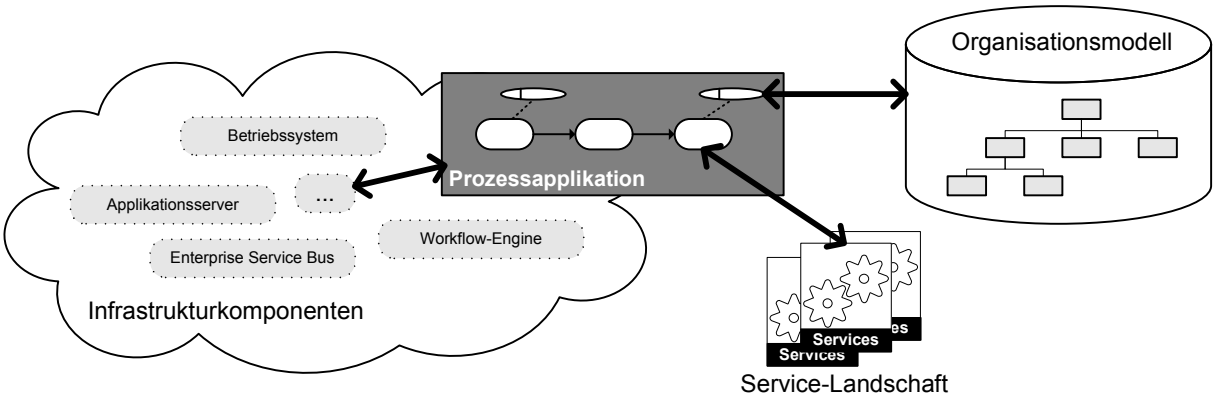


Abbildung 5.9.: Umgebung einer Applikation

Informationen zur ursprünglichen Service-Lokation besitzen.

Maßnahme 11: Sicherstellung der Robustheit bei Lokationsänderungen

Die Robustheit einer Prozessapplikation bei Lokationsänderungen kann durch Einsatz von Proxies und Repository sichergestellt werden. Ein Proxy ist eine Komponente, die einen definierten Proxy-Ablauf zur Modifikation von Daten beschreibt. Dies wird meist durch die Aneinanderreihung unterschiedlicher Ablaufknoten realisiert. Letztere transformieren Daten, rufen Services auf oder fragen ein Repository nach Informationen an. Prozessapplikationen können Proxy-Abläufe über eine standardisierte Schnittstelle (bspw. Web Service) verwenden. Folgende Szenarien beschreiben, wie dies realisiert werden kann (vgl. Abbildung 5.10).

Szenario S1: Entkopplung mittels Proxy. Der Einsatz eines Proxy 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 5.10a). Dieser Proxy realisiert dann 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, nicht aber die den Service nutzenden Prozessapplikationen (vgl. Abbildung 5.10b). Konkret wird der Proxy-Ablauf so angepasst (2) (*Call X @* ***B***), dass er die neue Lokation B von Services X (3) verwendet. Einige bereits häufig in der Praxis verwendete Proxy-Werkzeuge (z.B. der IBM WebSphere Enterprise Service Bus [IBM08a]) unterstützen eine solche Entkopplung.

Szenario S2: Repository verwaltet Endpunkte. Eine weitere Möglichkeit, Prozessapplikationen auch während bzw. nach einer Service-Migration stabil weiter betreiben zu können, ist der Einsatz eines SOA-Repository zur Verwaltung von Endpunktinformationen der Services. Ein solches SOA-Repository wird zur Endpunktauflösung eingesetzt (vgl. Abbildung 5.11a). Die Service-Lokation wird nicht im Proxy-Ablauf kodiert, sondern dynamisch zur Laufzeit ermittelt. Dazu ruft die Prozessapplikation analog zu Szenario S1 den Proxy auf (1). Anschließend wird

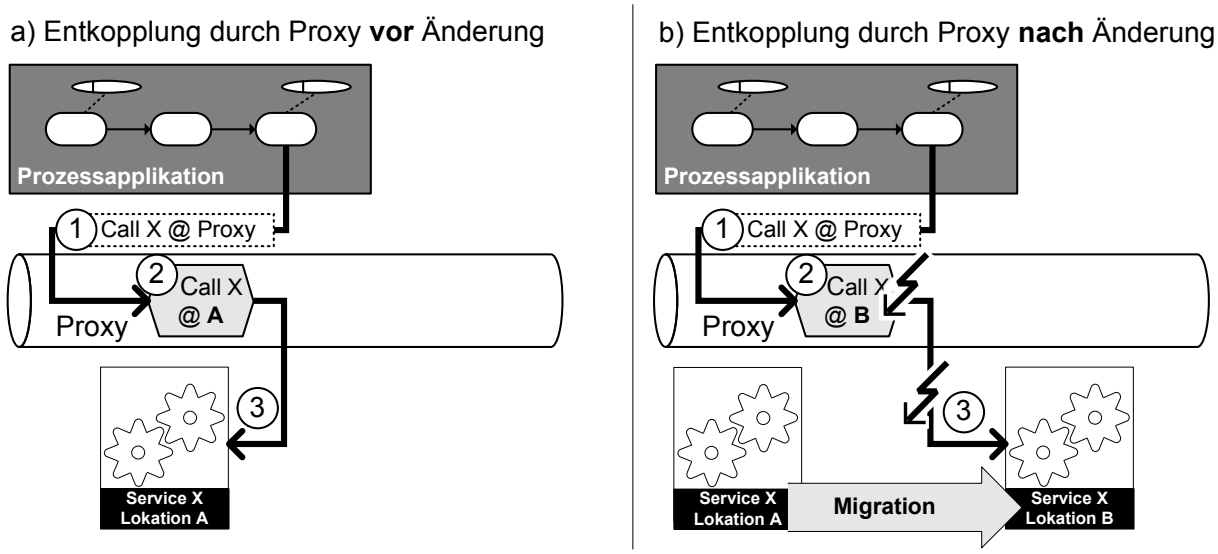


Abbildung 5.10.: Service-Migration durch Einsatz eines Proxies

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 der Service X migriert (vgl. Abbildung 5.11b), 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 Entkopplung erreicht wird.

5.3.2. 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. Entlang der Anforderung 3.5 aus Abschnitt 3.3 beschreiben wir die Vermeidung solcher Fehler durch Maßnahme 12.

Maßnahme 12: Parallelbetrieb von Services zur Ausfallsicherheit

Durch Parallelbetrieb mehrerer identischer Services (d.h. Service-Replikationen) sowie Einsatz von Proxy und Repository (vgl. Maßnahme 11), können solche Fehlersituationen oftmals vor der Prozessapplikation verborgen werden [RDH05]. Dazu müssen zunächst die Service-Endpunkte für

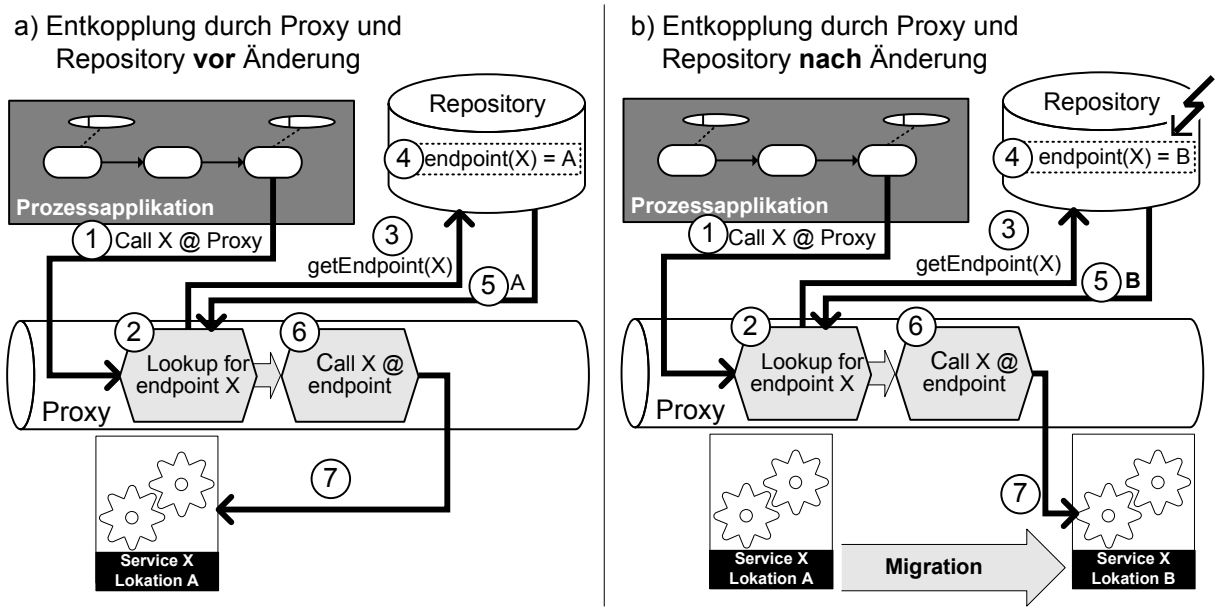


Abbildung 5.11.: Service-Migration durch den Einsatz von Proxy und Respository

einen Parallelbetrieb im Repository registriert werden (0) (vgl. Abbildung 5.12). 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 Service 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 Service X auf (7). Schlägt dieser Service-Aufruf fehl (8), wird der nächste Service-Endpunkt aus der Liste für einen Service-Aufruf herangezogen (9).

Darüber hinaus trägt der Parallelbetrieb von Service-Instanzen zu einer Balancierung der Last bei [ZT09]. 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.

5.3.3. Versionswechsel und Abschaltung von Services

Änderungen über die Zeit werden durch Versionen beschrieben [Stu05]. So führen Änderungen an fachlichen Anforderungen oder Nachbesserungen in der Service-Implementierung (Bugfixes) zu neuen Service-Versionen. Letztgenannte sollen alte Service-Versionen ablösen. Eine direkte Abschaltung alter Service-Versionen führt jedoch häufig zur aufwendigen Anpassung der Konsumenten dieser Services, da die neue Service-Version nicht immer kompatibel zur alten ist. Bei

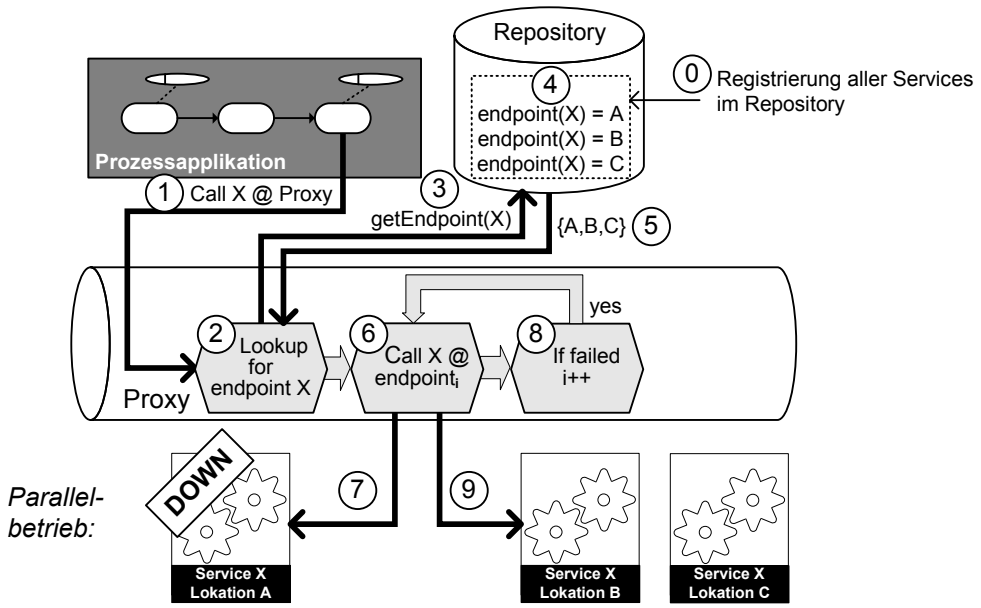


Abbildung 5.12.: Proxy-Ablauf zum Verbergen von Service-Ausfällen

der Service-Versionierung wird deshalb prinzipiell zwischen kompatiblen und inkompatiblen Änderungen von Services unterschieden [FLF+07]. 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 Prozessapplikationen nachvollziehbar zu gestalten, wird analog zu Maßnahme 12 einen Proxy-Ablauf und ein SOA-Repository verwendet. Abbildung 5.13a zeigt wie ein konkreter Service-Aufruf (inkl. Datenobjekten) in einem Proxy-Ablauf realisiert werden kann. Die Prozessapplikation ruft den Endpunkt des Proxy 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 5.13b 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 Anpassung des Proxy-Ablaufs kann eine meist sehr aufwendige 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 [PQSR06b]. 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.

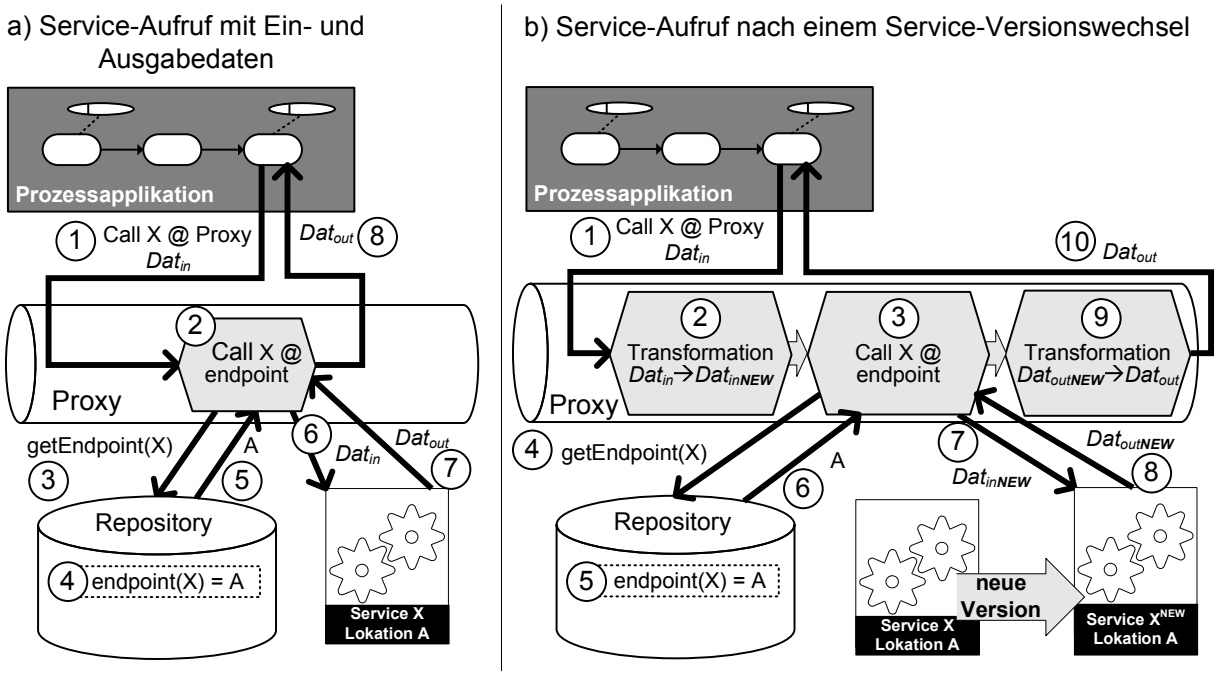


Abbildung 5.13.: Proxy-Ablauf zum Verbergen von Service-Ausfällen

Maßnahme 14: Ermittlung von Änderungsauswirkungen durch Vorfeldanalyse

Änderungen an Services (z.B. entfernte Service-Operationen) 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, indem etwa die Gültigkeitszeiträume von Services in einem SOA-Repository verwaltet werden [BTBR10]. Durch Analysen kann nun frühzeitig festgestellt werden, welche Service-konsumierenden Prozessapplikationen ggf. von Änderungen betroffen sind und zu welchen Zeitpunkten ein Service-Versionswechsel geplant ist (vgl. Anforderung 3.5 in Abschnitt 3.3). Weiter sollten Governance-Prozesse die Versionswechsel bzw. Service-Abschaltungen koordinieren. Hierdurch werden vor Bewilligung bzw. Durchführung von Umgebungsänderungen konkrete Aussagen zu deren Auswirkungen möglich. Diese Probleme werden ausführlich in Kapitel 8 diskutiert und es werden entsprechende Lösungskonzepte aufgezeigt.

5.3.4. 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, etwa ein Prozess-Management-System [RW12] oder Enterprise Service Bus [BBP09]. Meist sind jedoch Prozessapplikationen an die entsprechenden Infrastrukturkomponenten gebunden, da diese spezielle Programmierschnittstellen verwenden. Wird eine Infrastrukturkomponente ausgetauscht oder eine neue Version einer solchen Komponente eingeführt, müssen die Prozessapplikationen auf diese Komponente angepasst werden, damit ein weiterer Betrieb möglich ist.

Maßnahme 15: Dokumentation der Betriebsumgebung

Die Dokumentation von Informationen über Infrastrukturkomponenten ist beispielsweise durch Verwendung eines SOA-Repository möglich. Nur wenn alle Abhängigkeiten zwischen Prozessapplikationen und Infrastrukturkomponenten gespeichert sind, kann analysiert werden, welche Auswirkung die Änderung einer Infrastrukturkomponente auf Prozessapplikationen hat. Dadurch lassen sich Infrastrukturkomponenten austauschen, ohne den Betrieb von Prozessapplikationen zu beeinträchtigen. Eine solche Dokumentation ermöglicht, analog zu Maßnahme 14, eine Analyse von Änderungen betroffener Infrastrukturkomponenten und Prozessapplikationen.

5.3.5. Änderungen am Organisationsmodell

Änderungen am Organisationsmodell werden meist durch den Fachbereich oder durch eine Umstrukturierung des Unternehmens ausgelöst [RR07, RR08]: Abteilungen werden verschmolzen und Rollenzuordnungen ändern sich. Prozessapplikationen referenzieren bestimmte Organisationseinheiten aus einem Verzeichnisdienst (z.B. Rolle *model_range_officer* in Abbildung 5.5). 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 5.3.1 kann hier die Entkopplung mittels eines Proxy realisiert werden. Dadurch wird die Rollenauflösung nicht durch die Prozessapplikation selbst realisiert, sondern durch den Proxy. Konkret bedeutet dies, dass die Auflösung der Organisationseinheit durch einen Ablaufknoten realisiert wird und nicht in der Prozessapplikation implementiert ist [RR08].

5.4. Diskussion

Abschließend diskutieren wir weitere Ansätze zur Erhöhung der Flexibilität in einer SOA. Zunächst betrachten wir Methoden, die das grundlegende Vorgehen zur Entwicklung von Service-orientierten Architekturen und Informationssystemen beschreiben. Anschließend wird untersucht, durch welche Einzelansätze bestimmte Maßnahmen zur Flexibilisierung einer SOA realisierbar werden. Eine Ergebnisübersicht liefert Tabelle 5.2.

Bereits in Kapitel 3 wurden verschiedene Methoden zur Entwicklung von Service-orientierten Architekturen und Informationssystemen detailliert beschrieben. So liefert etwa die von IBM entwickelte Methodik SOMA [AGA+08, Ars04] eine Vorgehensweise zur Erstellung von Software in einer SOA. Die Erstellung von Software wird dabei in sieben größere aufeinander folgende Phasen gegliedert (vgl. Abbildung 3.17). Hinsichtlich der hier beschriebenen Maßnahmen interessant ist die Phase *Identification*, in welcher Services eines SOA-Projektes identifiziert und Beziehungen dokumentiert werden. Ein sogenannter *Service Litmus Test (SLT)* prüft Service-Kandidaten auf Ähnlichkeiten und filtert redundante Kandidaten vor deren Implementierung aus. Dies erhöht die Flexibilität dahingehend, dass bereits in einer frühen Phase des Softwareentwicklungsprozesses Redundanzen späterer Service-Implementierungen vermieden werden können.

[Klü07] beschreibt in zehn Schritten, wie Fachprozesse (EPKs) durch Service-Informationen angereichert werden sollten, um daraus automatisiert technische Prozesse (BPEL-Prozesse) generieren zu können. Die in [Klü07] vorgestellte Methodik beginnt mit der fachlichen Modellierung eines Geschäftprozesses im ARIS SOA Architect [Ste07]. Anschließend findet eine Anreicherung des Fachprozesses mit Service und Daten statt. Bereits existierende Services werden dabei durch die Analyse ihrer Ein- und Ausgabedaten oder über dokumentierte Funktionalitäten identifiziert, wodurch eine frühe Dokumentation bereits existierender Services realisiert werden kann (Teilsaspekt von Maßnahme 3).

Es existieren eine Vielzahl weiterer Vorgehensmodelle zur Entwicklung Service-orientierter Anwendungen. [TLS09] führt dazu eine detaillierte Analyse zahlreicher (17) Vorgehensmodelle durch und liefert eine tabellarische Zusammenfassung ihrer Klassifikationsmerkmale. Eine konkrete Beschreibung, hinsichtlich der in einer SOA notwendigen Flexibilität, wird dabei nicht gegeben. Wie bereits durch die oben beschriebenen Ansätze skizziert, werden lediglich vereinzelt Maßnahmen zur Flexibilisierung unterstützt.

Neben Vorgehensmodellen existieren weitere Ansätze, die einige der in diesem Kapitel beschriebenen Maßnahmen realisieren. So beschreibt etwa [Erl09] durch unterschiedliche *Design Patterns* einzelne SOA-Aspekte, etwa die Bereitstellung zentralisierter Datenspeicher (*Metadata Centralization*) oder die Verwendung zentralisierter Geschäftsregeln (*Rule Centralization*). Durch diese Art der Zentralisierung werden Geschäftsregeln von der eigentlichen Applikation (ausführbarer Prozess) entkoppelt und können dadurch zur Ausführungszeit unabhängig von der Applikation geändert werden (vgl. Maßnahme 6). Ein weiteres Pattern beschreibt den Parallelbetrieb von Service-Instanzen (*Redundant Implementation*), wodurch Maßnahme 12 umgesetzt werden kann. In [EKW+09] werden die Grundlagen gelegt, um Services zu versionieren und basierend darauf sowie durch den Einsatz eines Proxy Maßnahme 13 zu realisieren.

[Erl05, Jos07, EKW+09] betrachten weitere einzelne SOA-Aspekte, etwa die Bereitstellung eines SOA-Repository für Service-Endpunkte oder die Verwendung eines Enterprise Service Bus

(ESB) zum entkoppelten Aufruf konkreter Service-Instanzen. Eine explizite Betrachtung der hier beschriebenen Maßnamen zur Flexibilisierung einer SOA findet nicht statt.

Einzelne der in diesem Kapitel 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 [WB07]) und einer technisch (Bottom-up-Ansatz [Nad04]) orientierten Identifikation unterschieden. Auch eine Kombination beider Vorgehensweisen ist möglich [AGA+08, KKB07, EHH+08]. 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 [WGL+09], die fachliche Anforderungen und korrespondierende Fachprozesse in Beziehung setzen (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) [Roh08] sowie das Verbergen von Service-Versionswechseln (Maßnahme 13) [FLF+08]. 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. Außerdem existieren Ansätze zur Steuerung von Prozessen mittels einer adaptiven Prozess-Engine [DRRM+09, DR09], durch die Maßnahme 9 realisiert werden.

Der Umgang mit organisatorischen Änderungen, etwa infolge von Entlassungen bzw. Neueinstellungen von Mitarbeitern, wird im Projekt CEOSIS [RCR04, RR07, RR08] adressiert. Dabei werden Fragestellungen im Zusammenhang mit der Entwicklung von Organisationsstrukturen diskutiert. Es wird ein Rahmenwerk vorgestellt, welches Änderungen an Organisationsmodellen sowie Zugriffsregeln ermöglicht und zugleich deren Auswirkungen analysiert. Diese Ansätze fokussieren auf die Realisierung von Maßnahme 16.

Maßnahmen	**SOMA** [AGA+08, Ars04]	**10 Steps to Business-Driven SOA** [Kli07, Sof05, Hil07, Sof09]	**T. Erl** [Erl05, Erl09, EKW+09]	**CEOSIS** [RCR04, RR07, RR08]	**Einzelaspekte** [Roh08, FLF+07, FLF+08, WGL+09] [DRRM+09, DR09]
M1: Beziehungen zwischen Anforderung und Fachprozess	/	o	-	/	+ [WGL+09]
M2: Frühe Modellierung von Informationen durch Frontloading	o	o	-	/	/

Fortsetzung auf nächster Seite

Maßnahmen	**SOMA** [AGA+08, Ars04]	**10 Steps to Business-Driven SOA** [Kli07, Sof05, Hi107, Sof09]	**T. Erl** [Erl05, Erl09, EKW+09]	**CEOSIS** [RCR04, RR07, RR08]	**Einzelaspekte** [Roh08, FLF+07, FLF+08, WGL+09] [DRRM+09, DR09]
M3: Berücksichtigung von Informationen durch Look-ahead	o	o	o	/	/
M4: Nachvollziehbarkeit zwischen Modellierungsebenen	-	-	-	/	/
M5: Analyse von Inkonsistenzen	-	-	-	/	/
M6: Flexibilität durch Verwendung von Geschäftsregeln	o	-	+	+	/
M7: Service-Auswahl unter Nutzung von Quality-of-Service-Kriterien	-	o	-	/	/
M8: Entkopplung beim Deployment durch Packaging	/	-	o	/	/
M9: Prozesssteuerung durch Prozess-Engine	-	-	o	/	+ [DRRM+09, DR09]
M10: Messung fachlicher Kennzahlen	+	o	-	/	/
M11: Sicherstellung der Robustheit bei Lokationsänderungen	/	-	o	/	+ [FLF+07, FLF+08]
M12: Parallelbetrieb von Services zur Ausfallsicherheit	/	-	+	/	/
M13: Verbergen eines Service-Versionswechsels	-	-	o	/	+ [Roh08]
M14: Ermittlung von Änderungsauswirkungen durch Vorfeldanalyse	-	o	-	/	/
M15: Dokumentation der Betriebsumgebung	-	o	o	/	/
M16: Verbergen von Änderungen durch Proxy-Elemente	-	-	o	+	/

Legende: + = wird unterstützt; **-** = wird nicht unterstützt; **o** = mit Anpassung unterstützt; / = Bewertung nicht möglich

Tabelle 5.2.: Übersicht zu Maßnahmen und Ansätzen in der Literatur

5.5. Zusammenfassung

In diesem Kapitel wurden Maßnahmen diskutiert, die die Flexibilität einer SOA bzgl. der Entwicklung, Inbetriebnahme und Wartung von Prozessapplikationen erhöhen. Ein entsprechendes Architekturmanagement legt die Basis für eine stabile und flexible SOA, indem Governance-Prozesse zur Steuerung und Verwaltung von Services und deren Lebenszyklen definiert werden. Durch die frühe Identifikation potentieller Services wird die Grundlage für eine spätere Entwicklung von Prozessapplikationen gelegt. Dazu ist jedoch auch eine frühe Modellierung implementierungsrelevanter Informationen (Maßnahme 2) notwendig, die heute in der Regel nicht stattfindet. Fachmodellierer haben meist sehr detailliertes Fachwissen, jedoch existiert keine adäquate Methodik, dieses Wissen in Fachmodellen zu beschreiben. Wir entwickeln in Kapitel 7 geeignete Lösungskonzepte und -methoden für ein sogenanntes *Frontloading*, um Fachwissen möglichst frühzeitig zu identifizieren und bereits im Fachprozess zu dokumentieren. Außerdem werden Lösungskonzepte für *Look-ahead* (vgl. Maßnahme 3) in Kapitel 7 entwickelt, um bereits existierende Artefakte (z.B. Services) während der Erstellung des Fachmodells nutzen zu können.

Für die Dokumentation der Beziehungen zwischen Fach- und Systemmodell (inkl. ihrer Beziehungen, vgl. Maßnahmen 4 und 5 in Abbildung 5.14) finden sich in der Literatur keine geeigneten Lösungsansätze.

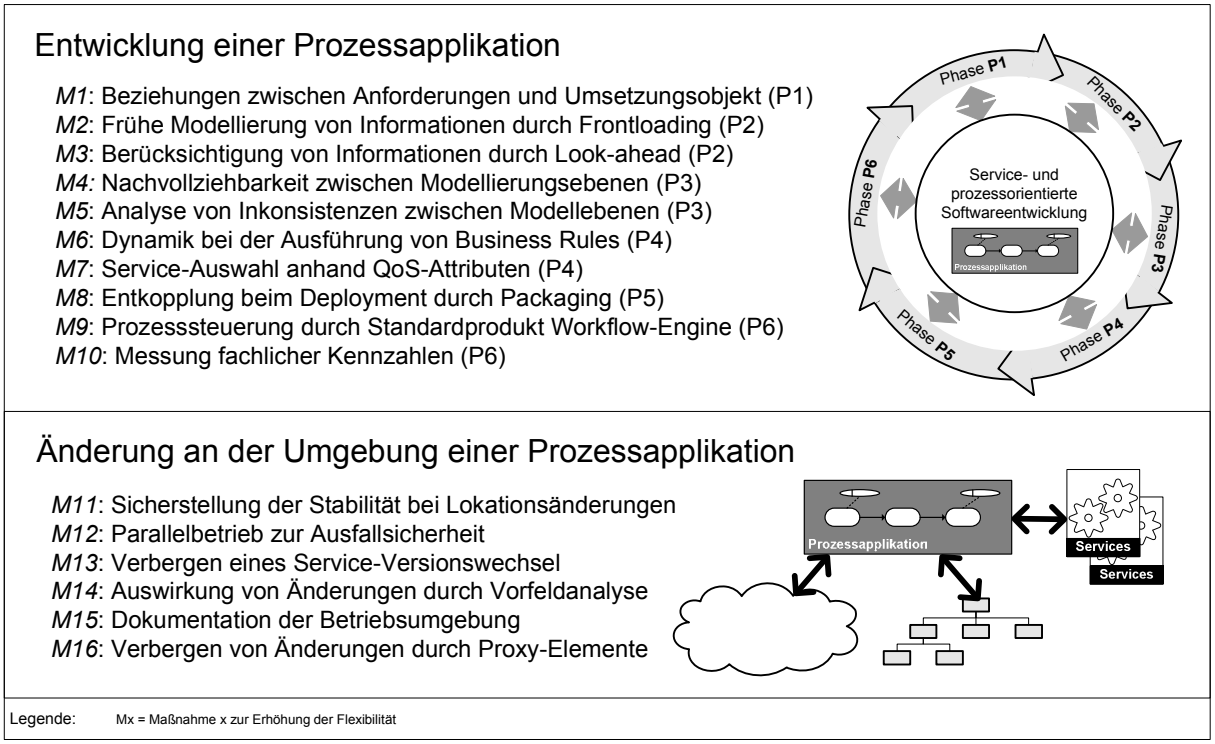


Abbildung 5.14.: Maßnahmen zur Erhöhung der Flexibilität in einer SOA

Eine zentrale Dokumentation von Informationen durch ein Repository wird in mehreren Maßnahmen als notwendig identifiziert, eine adäquate Lösung in der Literatur ist jedoch nicht verfügbar.

Deshalb werden diese Themen in den folgenden Kapiteln aufgegriffen und entsprechende Lösungskonzepte entwickelt. Wir entwickeln einen Modellierungsansatz zur durchgängigen, nachvollziehbaren Dokumentation von Fachprozessen. Dabei werden Fachprozesse ausgehend von fachlichen Anforderungen modelliert und in einer zusätzlichen Modellierungsebene, dem sogenannten Systemmodell, technisch verfeinert. Anschließend wird das Systemmodell als Spezifikation zur Implementierung einer service- und prozessorientierten Applikation an den IT-Bereich übergeben [BBR09, BBR10, BBR11a]. Zusätzlich dazu werden Methoden konzipiert, die eine durchgänge Modellierung solcher Applikationen ermöglichen und zugleich Frontloading- und Look-Ahead-Aspekte berücksichtigen.

Neben einer durchgängigen Methodik und Konzepten zur Beschreibung der Beziehungen zwischen den unterschiedlichen Modellierungsebenen, werden Analysen entwickelt, die Inkonsistenzen zwischen den Modellierungsebenen automatisch aufdecken und die verantwortlichen Modellierer darüber informieren. Ein großes Problem stellt in diesem Zusammenhang die große Diskrepanz zwischen den Anforderungen der Fachbereiche und den vom IT-Bereich angebotenen technischen Implementierungen dar.

Um den oben genannten Herausforderungen gerecht zu werden, bedarf es einer durchgängigen Definition, Verwaltung und Pflege von Prozessen, Services und Datenobjekten. Dies gilt sowohl für die fachliche als auch die technische Ebene. Informationen zum komplexen Beziehungsgeflecht zwischen fachlichen und technischen Prozessen, Services und Datenobjekten sind in heutigen Unternehmensarchitekturen jedoch meist nicht vorhanden, was zu weiteren Problemen führt. So ist bei Außerbetriebnahme eines Services nicht ohne weiteres nachvollziehbar, welche Prozessapplikationen davon betroffen sind. Dadurch ist es wiederum schwierig sicherzustellen, dass die Deaktivierung einzelner Services oder Service-Versionen in der Folge nicht zu unerwarteten Fehlern führt, etwa dass die Implementierung eines Fachprozesses nicht mehr funktioniert. In Kapitel 8 entwickeln wir ein SOA-Repository, welches die genannten Abhängigkeiten verwaltet und eine applikationsübergreifende Konsistenz zwischen den einzelnen Artefakten sicherstellt.

10

Zusammenfassung und Ausblick

Service-orientierte Architekturen bieten für Unternehmen vielversprechende Perspektiven. Ein wesentliches Ziel das mit ihrer Einführung in ein Unternehmen verfolgt wird, bildet die flexible IT-Unterstützung von Geschäftsprozessen (Fachprozessen). Dies wiederum erfordert eine (Teil-) Automatisierung dieser Prozesse sowie die rasche Anpassbarkeit der implementierten Prozessapplikationen. Um die in der betrieblichen Praxis geforderte Flexibilität zu verwirklichen, sind verschiedene Konzepte, Methoden und Maßnahmen erforderlich. Diese reichen von der Dokumentation fachlicher Anforderungen, über die fachliche und technische Modellierung von Fachprozessen, bis zu dynamischen Service-Aufrufen während der Prozessausführung. Existierende Konzepte und Werkzeuge bieten keine ausreichende Unterstützung für diese Aufgaben. Sie erlauben lediglich eine einfache Modellierung von Fachprozessen bzw. die Spezifikation ausführbarer Prozessmodelle, beinhalten aber keine explizite Vorgehensweise zur Erstellung der Fachprozesse und deren IT-Realisierung. Ferner liefern sie keine adäquate Lösung zur Dokumentation der zahlreichen Abhängigkeiten zwischen Modellen aus dem Fach- und IT-Bereich. Einer der Gründe für diese Defizite ist die Lücke, die zwischen Fachprozessmodellen einerseits und deren technischen Spezifikationen und Implementierungen andererseits besteht. Mit der Zeit führt dies zu einer Divergenz der verschiedenen Modelle und Spezifikationen. Dies erhöht den Wartungsaufwand und erfordert teure Umgestaltungen der Spezifikation und Implementierung bei Änderungen an Fachprozessen.

Die vorliegende Arbeit liefert Konzepte und Methoden zur Erhöhung der Durchgängigkeit und Flexibilität prozessorientierter Applikationen mittels Service-Orientierung:

- **Service- und prozessorientierte IT-Infrastruktur:** Wichtig für jede SOA ist die Standardisierung der IT-Infrastruktur von Unternehmen. Nur dann sind IT-Anbieter leichter austauschbar und Entwicklungsaufwände beherrschbar. Ferner wird die Verwendung und Funktionalität der IT für Nutzer vereinheitlicht, d.h. für jede im Unternehmen benötigte Funktionalität wird ein Produkt ausgewählt, das für IT-Applikationen als Implementierungsplattform fest vorgegeben ist. Da jedes Unternehmen unterschiedliche Anforderungen

an eine SOA bzw. an die durch eine SOA erreichbare Flexibilität hat, ist eine Betrachtung unterschiedlicher Ausbaustufen der einzelnen SOA-Komponenten erforderlich. Die Anforderungen der Unternehmen erstrecken sich von erhöhter Funktionalität einzelner Komponenten für Nutzer, bis hin zu mehr Funktionalität bei der Prozesssteuerung. Ferner sollten Kosten infolge redundanter Implementierungen vermieden werden. Aus diesem Grund beschreibt ENPROSO das Zusammenspiel typischer Funktionen einer SOA durch ein Rahmenwerk. Letzteres bietet unterschiedliche Ausbaustufen und ermöglicht eine Einordnung und Bewertung existierender IT-Infrastrukturkomponenten. Dies liefert die Basis für zukünftige Erweiterungen in Richtung einer flexiblen, service-orientierten IT-Infrastruktur.

- **Maßnahmen zur Erhöhung der Flexibilität:** Unternehmen versprechen sich durch SOA mehr Flexibilität bei der Entwicklung und Inbetriebnahme ihrer IT-Applikationen. Was jedoch konkret mit Flexibilität gemeint ist, bleibt meist unklar. ENPROSO gibt ein Rahmenwerk vor, das Flexibilitätsmaßnahmen entlang des Software-Entwicklungsprozesses beschreibt. Insbesondere wird die Flexibilität einer SOA bzgl. der Entwicklung, Inbetriebnahme und Wartung von Prozessapplikationen erhöht. Dabei wird auch eine Vorgehensweise zur flexiblen Entwicklung von Prozessapplikationen fixiert. Die identifizierten Maßnahmen geben einen Gesamtüberblick über die mittels SOA realisierbare Flexibilität, um service- und prozessorientierter Applikationen effizienter zu betreiben und kostengünstiger entwickeln zu können. Einige dieser Maßnahmen lassen sich bereits durch existierende Konzepte und Werkzeuge realisieren. Für den Großteil von ihnen existieren jedoch noch keine adäquaten Methoden und Lösungskonzepte.
- **Durchgängige Modellierung:** Wichtige Ziele einer SOA sind die bessere Unterstützung und Anpassbarkeit von Fachprozessen sowie das Business-IT-Alignment. Diese werden heute nicht angemessen umgesetzt, da die bei der Implementierung eines Fachprozesses notwendigen Transformationen in einen ausführbaren Prozess kaum nachvollziehbar sind. Dadurch gehen fachliche Anforderungen verloren, und es entsteht ein hoher Aufwand bei späteren Prozessanpassungen. In ENPROSO wird zu diesem Zweck ein Abbildungsmodell eingeführt, das die Zugehörigkeit von Fach- zu Systemprozessschritten dokumentiert. Dadurch werden automatisierte Konsistenzprüfungen zwischen den Modellebenen möglich. Werden später Prozessanpassungen erforderlich, lassen sich die zu einem Fachprozessschritt gehörenden Systemprozessschritte direkt erkennen, was die Durchführung der Anpassung erleichtert. Ein Vorteil dieses Ansatzes besteht darin, dass die Erstellung des Abbildungsmodells nur einen minimalen Aufwand verursacht, da keine komplexen Regeln, sondern lediglich einfache Beziehungen definiert werden müssen. ENPROSO unterstützt die dazu notwendigen Konzepte und stellt eine Methodik für eine durchgängige Modellierung bereit.
- **Frühzeitige Dokumentation von Information durch Frontloading:** ENPROSO ermöglicht die frühzeitige Modellierung implementierungsrelevanter Information; diese kann bereits bei der Fachprozessmodellierung beschrieben werden (Frontloading). Dadurch entsteht während der fachlichen Analyse und Konzeptentwicklung eine detaillierte Beschreibung der zu implementierenden Sachverhalte. Letztere umfassen konkrete Bearbeiterzuordnungen für einzelne Prozessschritte, fachliche Services, fachliche Datenobjekte und speziell markierte Stellen im Fachprozess (Flexiblitätsmarkierungen), die später bei der Implementierung berücksichtigt werden. Durch dieses *Frontloading* lassen sich späte Interviews mit Fachbereichen vermeiden und Fehler bei der Implementierung reduzieren.

- **Wiederverwendung existierender Artefakte (Look-ahead):** Ein weiteres ENPROSO-Konzept ist *Look-ahead*. Es beschreibt methodisch und konzeptionell, wie existierende Artefakte (z.B. Services oder Datenobjekte) bereits während der Fachprozessmodellierung durch Fachmodellierer verwendet werden können. Bisherige SOA-Ansätze lassen dies nicht zu, so dass bei der Fachprozessmodellierung bereits spezifizierte Services oder technisch umgesetzte Datenobjekte nicht berücksichtigt werden können. Oftmals werden Services oder andere SOA-Artefakte redundant entwickelt, was durch Look-Ahead vermieden werden kann. Die in ENPROSO entwickelten Konzepte und Methoden beschreiben, welche Art von Look-Ahead in welchen Phasen des Softwareentwicklungsprozesses durchzuführen ist und beschreiben konkrete Lösungen für deren Realisierung.
- **Zentrale Dokumentation von SOA-Artefakten:** Durch die Vielzahl von Services und Prozessen in unterschiedlichen Varianten sowie deren gleichzeitige Verwendung durch eine Vielzahl von Nutzern entstehen hohe Kosten für den Betrieb und die Wartung. Services ohne oder mit nur wenig Nutzern sollten zeitnah *abgeschaltet* werden. Um nicht mehr benötigte Services identifizieren zu können, muss bekannt sein, welche Services aktuell von welcher Applikation benutzt werden. Zudem entstehen durch unterschiedliche Service-Versionen komplexe Abhängigkeiten, die durch eine geeignete Informationsverwaltung verwaltet werden müssen. Nur so kann sichergestellt werden, dass ein Service-Versionswechsel und die dadurch notwendig werdende Migration der Services auf die Folgeversion fehlerfrei durchführbar sind. Dies wird durch entsprechende Governance-Prozesse realisiert. Heutige Repository-Werkzeuge können lediglich Teile der in einer SOA zu verwaltenden Artefakte speichern. Insbesondere Abhängigkeiten zwischen fachlichen und technischen Artefakten, z.B. Services oder Prozessen, bleiben unberücksichtigt, was eine durchgängige Modellierung unmöglich macht. Dementsprechend sind die Auswirkungen von Änderungen an solchen Artefakten aufgrund fehlender Abhängigkeiten nicht nachvollziehbar. ENPROSO stellt ein umfassendes Metamodell für ein SOA-Repository bereit, welches alle SOA-Artefakte (inklusive deren Abhängigkeiten) verwaltet.
- **Ist- und Zukunftsanalyse dokumentierter Abhängigkeiten:** Basierend auf Informationen aus dem SOA-Repository werden Ist- und Zukunftsanalysen unterstützt. Erstere untersuchen den gegenwärtigen Stand der im Repository dokumentierten Artefakte, wohingegen letztere die Auswirkungen von Änderungen zu einem bestimmten Zeitpunkt in der Zukunft betrachten. So kann festgestellt werden, welche Konsumenten von einer Service-Abschaltung betroffen sein werden. Zusätzlich können Nutzungsverträge bestimmt werden, welche zu einem bestimmten Zeitpunkt auslaufen. Dies ermöglicht die Konsistenthaltung der dokumentierten Artefakte. Außerdem können ggf. entstehende Fehlersituationen frühzeitig erkannt werden.

ENPROSO bietet eine Reihe weiterer Themen zur Bearbeitung an:

- **Werkzeugintegration:** In Kapitel 6 wurden Konzepte entwickelt und Methoden beschrieben, wie eine durchgängige Modellierung in einer Service-orientierten Architektur realisiert werden kann. Dazu wurden in Kapitel 9 eine prototypische Realisierung und eine Fallstudie vorgestellt. Um diese Konzepte und Methoden für eine durchgängige Modellierung in großen Projekten verankern zu können, ist eine Integration in die bestehende Projektinfrastruktur,

insbesondere in die entsprechenden IT-Werkzeuge notwendig. Für die in ENPROSO konzipierten Modelle, etwa das Abbildungsmodell, muss ein zusätzlicher Modelltyp vorhanden sein, um die Beziehungen zwischen Fach- und Systemprozess dokumentieren zu können. Nicht nur die Modellintegration in bestehende Werkzeuge ist wichtig für die Realisierung einer durchgängigen Modellierung, sondern auch eine dazu passende Methodik. Erst wenn eine Modellintegration in vorhandene Werkzeuge realisiert ist, kann eine Erprobung der ENPROSO-Konzepte und -Methoden in großen Projekten erfolgen.

- **Benutzerfreundliche Visualisierung:** System- und Abbildungsmodelle werden durch Systemmodellierer in Abstimmung mit Fachmodellierern erstellt, weshalb die Modelle für beide Personengruppen visualisiert werden müssen [BRB07, Bob08].
- **Unterstützung weiterer Aspekte einer durchgängigen Modellierung:** Die in Kapitel 6 beschrieben Konzepte und Methoden zur durchgängigen Modellierung fokussieren auf Prozessaspekte, d.h. Fach- und Systemprozesse (inkl. zugehörige Service und Datenobjekte). Darüber hinaus existieren Aspekte wie Geschäftsregeln oder Transaktionsgrenzen, welche ebenfalls durchgängig modelliert werden sollten.
- **Unterstützung weiterer Frontloading- und Look-Ahead-Aspekte:** In Kapitel 7 wurden Anforderungen an Frontloading und Look-ahead beschrieben und für drei Aspekte konkrete Lösungen zur Umsetzung entwickelt. Für weitere Aspekte ist ebenfalls eine detaillierte Methodik zur Umsetzung zu entwickeln und in konkreten Projekten zu erproben. Beispielhaft dafür stehen das Frontloading für fachliche Services oder ein Look-Ahead für implementierte Geschäftsregeln.
- **Synchronisierung verteilter Repositories:** In Kapitel 8 wurde ein Metamodell für ein SOA-Repository vorgestellt, das unterschiedliche Applikationen (CASE-Entwicklung, Fachprozessmodellierung), die verschiedene Artefakte lesen und erzeugen, unterstützt. Viele dieser Artefakte, etwa technische Service-Spezifikationen oder Fachprozesse, existieren bereits in applikationseigenen Datenbanken und heterogenen Repositories. So können in ARIS definierte fachliche Services und modellierte Fachprozesse in einer ARIS-internen Datenbank gespeichert, wohingegen Web Services in einem WebSphere Service Registry and Repository gespeichert werden. Die Basis für ein einheitliches Metamodell wurde in Kapitel 8 gelegt, geeignete Konzepte zur Synchronisation unterschiedlicher heterogener Datenbanken und Repositories sind Herausfoderungen zukünfiger Forschungsarbeiten.

Literaturverzeichnis

[A+01] A. Ankolekar et al. DAML-S: Semantic Markup for Web Services. volume 13, pages 411–430. July, 2001.

[A+07a] A. Agrawal et al. Web Services Human Task WS-HumanTask, Version 1.0, 2007.

[A+07b] A. Agrawal et al. WS-BPEL Extension for People Specification, 2007.

[AAA+07] A. Alves, A. Arkin, S. Askary, B. Bloch, F. Curbera, Y. Goland, N. Kartha, C.K. Liu, V. Mehta, S. Thatte, P. Yendluri, and A. Yiu. Web Services Business Process Execution Language, Version 2.0. OASIS, 2007.

[AAC+08] S. Amnajmongkol, J. Angani, Y. Che, T. Fox, A. Lim, and M. Keen. Business Activity Monitoring with WebSphere Business Monitor V6.1. IBM Corporation, 2008.

[AAD+04] H. Acker, C. Atkinson, P. Dadam, S. Rinderle, and M. Reichert. Aspekte der komponentenorientierten Entwicklung adaptiver prozessorientierter Unternehmenssoftware. In: *Proc. 1st Verbundtagung AKA'04*, number 57 in Lecture Notes in Informatics (LNI), pages 7–24. Koellen-Verlag, 2004.

[Aal98] W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

[AB93] R.S. Arnold and S.A. Bohner. Impact Analysis - Towards a Framework for Comparison. In: *Proc. of the Conference on Software Maintenance*, ICSM '93, pages 292–301, Washington, DC, USA, 1993.

[Abr74] J.R. Abrial. Data Semantics. In: *IFIP Working Conference Data Base Management'74*, pages 1–60, 1974.

[AC05] G. Alonso and F. Casati. Web Services and Service-Oriented Architectures. In: *ICDE*, page 1147. IEEE Computer Society, 2005.

[ACKM04] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services Concepts, Architectures and Applications*. Springer-Verlag, Berlin, 2004.

[AGA+08] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, and K. Holley. SOMA - A Method for Developing Service-oriented Solutions. *IBM Systems Journal*, 47(1):377–396, 2008.

[AHKB03] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

[AHP+09] A. Agopyan, H. Huebler, T. Puah, T. Schulze, D.S. Vilageliu, and M. Keen. *WebSphere Application Server V7.0: Concepts, Planning, and Design*. IBM Technical Support Organisation, 2009.

[AL05] W.M.P. van der Aalst and K.B. Lassen. Translating Workflow Nets to BPEL4WS. Technical Report BPM-05-16, BPM Center, 2005.

[All07] T. Allweyer. Erzeugung detaillierter und ausführbarer Geschäftsprozessmodelle durch Modell-zu-Modell-Transformationen. In: *Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (EPK 2007)*, volume 303, pages 23–38. Citeseer, 2007.

[Ars04] A. Arsanjani. Service-oriented Modeling and Architecture. IBM developer works, 2004.

[ASA+06] C. Atkinson, D. Stoll, H. Acker, P. Dadam, M. Lauer, and M. Reichert. Separating Per-client and Pan-client Views in Service Specification. In: *Proc. Int'l Workshop on Service Oriented Software Engineering (IW-SOSE '06)*, pages 47–52. ACM Press, 2006.

[BAP+11] H. Boley, T. Athan, A. Paschke, S. Tabet, B. Grosof, N. Bassiliades, G. Governatori, D. Hirtle, O. Shafiq, and M. Machunik. Schema Specification of RuleML 1.0, 2011. http://ruleml.org/1.0/.

[Bau09] T. Bauer. Stellvertreterregelungen für Task-Bearbeiter in prozessorientierten Applikationen. *Datenbank-Spektrum*, 9(31):40–51, 2009.

[BB05] T. Barlen and W. Blankertz. *HTTP Server (powered by Apache): an integrated solution for IBM @server iSeries servers*. IBM Int'l Technical Support Organization, 2005.

[BB10] S. Buchwald and T. Bauer. Modellierung von Service-Aufrufbeziehungen zwischen prozessorientierten Applikationen. *EMISA Forum*, 30(2):32–48, 2010.

[BBP09] S. Buchwald, T. Bauer, and R. Pryss. IT-Infrastrukturen für flexible, serviceorientierte Anwendungen - ein Rahmenwerk zur Bewertung. In: *GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW'09)*, volume 13 of *Lecture Notes in Informatics (LNI)*, pages 524–543. Koellen-Verlag, 2009.

[BBR09] S. Buchwald, T. Bauer, and M. Reichert. Durchgängige Modellierung von Geschäftsprozessen durch Einführung eines Abbildungsmodells: Ansätze, Konzepte, Notationen. Technical Report UIB-2009-12, University of Ulm, 2009.

[BBR10] S. Buchwald, T. Bauer, and M. Reichert. Durchgängige Modellierung von Geschäftsprozessen in einer Service-orientierten Architektur. In: *Modellierung'10*, volume 161 of *Lecture Notes in Informatics (LNI)*, pages 203–211. Koellen-Verlag, 2010.

[BBR11a] S. Buchwald, T. Bauer, and M. Reichert. Bridging the Gap Between Business Process Models and Service Composition Specifications. In: *Service Life Cycle Tools and Technologies: Methods, Trends and Advances*, pages 124–153. Idea Group Reference, 2011.

[BBR11b] S. Buchwald, T. Bauer, and M. Reichert. Flexibilisierung Service-orientierter Architekturen. Technical Report UIB-2011-01, University of Ulm, 2011.

[BBR11c] S. Buchwald, T. Bauer, and M. Reichert. Flexible Prozessapplikationen in Service-orientierten Architekturen - Ein Überblick. *EMISA Forum*, 31(3):32–58, 2011.

[BDE+05] A.W. Brown, M. Delbaere, P. Eeles, S. Johnston, and R. Weaver. Realizing Service-oriented Solutions with the IBM Rational Software Development Platform. *IBM Systems Journal*, 44(4):727–752, 2005.

[BEH+07] A. Bösl, J. Ebell, S. Herold, C. Linsmeier, D. Peters, and A. Rauch. Modellbasierte Software-Entwicklung von Informationssystemen: Vom Geschäftsprozess zum servicebasierten Entwurf. *OBJEKTspektrum*, 4(1):46–53, 2007.

[Ber05] M. Berroth. Konzeption und Entwurf einer Komponente für Organisationsmodelle. Master's thesis, Universität Ulm, 2005.

[Beu02] T. Beuter. *Workflow-Management für Produktentwicklungsprozesse.* PhD thesis, Universität Ulm, 2002.

[BFF+08] P. Bauler, F.Feltz, E. Frogneux, B. Renwart, and C. Thomase. Usage of Model Driven Engineering in the Context of Business Process Management. In: *Proc. 4th GI Workshop XML Integration and Transformation for Business Process Management*, volume 4, pages 1963–1974. Citeseer, 2008.

[BGLS06] H. Bedoya, F. Gruz, D. Lema, and S. Singkorapoom. *Stored Procedures, Triggers, and User-defined Functions on DB2 Universal Database for Iseries.* Vervante, 2006.

[Bob08] R. Bobrik. *Konfigurierbare Visualisierung komplexer Prozessmodelle.* PhD thesis, Universität Ulm, 2008.

[BOC05] BOC Group. *Werkzeuggestützte Prozesskostenrechnung, White Paper*, 2005.

[BOC06] BOC Group. *Wege zu einer serviceorientierten Architektur (SOA), White Paper*, 2006.

[BOC07] BOC Group. *Prozessorientierte Anwendungsentwicklung mit ADONIS, White Paper*, 2007.

[BPS10] H. Boley, A. Paschke, and O. Shafiq. RuleML 1.0: The Overarching Specification of Web Rules. In: *Proc. of the 2010 Int'l Conference on Semantic Web Rules*, RuleML'10, pages 162–178, Berlin, Heidelberg, 2010.

[BRB05] R. Bobrik, M. Reichert, and T. Bauer. Requirements for the Visualization of System-Spanning Business Processes. In: *Proc. 1st Int'l Workshop on Business Process Monitoring and Performance Management (BPMPM'05)*, pages 948–954. IEEE Computer Society Press, 2005.

[BRB07] R. Bobrik, M. Reichert, and T. Bauer. View-Based Process Visualization. In: *5th Int'l Conf. on Business Process Management (BPM'07)*, number 4714 in LNCS, pages 88–95. Springer, 2007.

[BRBB09] S. Bassil, M. Reichert, R. Bobrik, and T. Bauer. Access Control for Monitoring System-Spanning Business Processes in Proviado. In: *3rd Int'l Workshop on Enterprise Modelling and Information Systems Architectures (EMISA'09)*, number P-152 in Lecture Notes in Informatics (LNI), pages 125–139. Koellen-Verlag, 2009.

[Bro04] A. W. Brown. Model Driven Architecture: Principles and Practice. In: *Software and Systems Modeling*, pages 314–327. Springer, 2004.

[BS06] J. Bloomberg and R. Schmelzer. *Service Orient or Be Doomed? How Service Orientation Will Change Your Business*. Wiley, 2006.

[BTBR10] S. Buchwald, J. Tiedeken, T. Bauer, and M. Reichert. Anforderungen an ein Metamodell für SOA-Repositories. In: *2nd Central-European Workshop on Services and their Composition (ZEUS'10)*, number 563, pages 17–24, 2010.

[BTZ99] J. Bouman, J. Trienekens, and M. van der Zwan. Specification of Service Level Agreements, Clarifying Concepts on the Basis of Practical Research. In: *Proc. Software Technology and Engineering Practice (STEP'99*, pages 169–178, 1999.

[Buc05] S. Buchwald. Konzeption und Realisierung einer Infrastruktur zur prozessorientierten Integration von Applikationen in Workflows. Master's thesis, Hochschule Ulm, 2005.

[Bud04] F. Budinsky. *Eclipse Modeling Framework: A Developer's Guide*. Prentice Hall Ptr, 2004.

[BWR11] L. Bodenstaff, A. Wombacher, and M. Reichert. Empirical Validation of MoDe4SLA; Approach for Managing Service Compositions. In: *14th Int'l Conference on Business Information Systems (BIS'11)*, volume 87 of *LNBIP*, pages 98–110. Springer, 2011.

[BWRJ08] L. Bodenstaff, A. Wombacher, M. Reichert, and M.C. Jaeger. Monitoring Dependencies for SLAs: The MoDe4SLA Approach. In: *IEEE 5th Int'l Conference on Services Computing (SCC 2008)*, pages 21–29. IEEE Computer Society Press, 2008.

[BWRJ09] L. Bodenstaff, A. Wombacher, M. Reichert, and M.C. Jaeger. Analyzing Impact Factors on Composite Services. In: *6th Int'l Conf. on Services Computing (SCC'09)*, pages 218–226. IEEE Computer Society Press, 2009.

[Car01] D. Carlson. Modeling the UDDI Schema with UML, 2001.

[CCMW01] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1, 2001.

[CCN+06] W. Chen, J. Chun, N. Ngan, R. Ranjan, and M.K. Sardana. *DB2 Express-C: The Developer Handbook for XML, PHP, C/C++, Java, and .NET*. IBM Corporation, 2006.

[CGH+05] L. Cherbakov, G. Galambos, R. Harishankar, S. Kalyana, and G. Rackham. Impact of Service Orientation at the Business Level. *IBM System Journal*, 44(4):653–668, 2005.

[Cha04] D.A. Chappell. *Enterprise Service Bus*. O'Reilly Media, Inc., 2004.

[Che08] H.M. Chen. Towards Service Engineering: Service Orientation and Business-IT Alignment. In: *Hawaii Int'l Conference on System Sciences, Proc. of the 41st Annual (HICSS'08)*, page 114. IEEE Computer Society, 2008.

[Cor10] IBM Corporation. Using Rational DOORS, White Paper, 2010.

[Cov02] R. Cover. Business Rules Markup Language, 2002. http://xml.coverpages.org/brml.html.

[CWMR07] R. Chinnici, S. Weerawarana, J. Moreau, and A. Ryman. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C recommendation, W3C, 2007. http://www.w3.org/TR/2007/REC-wsdl20-20070626.

[Dai07] Daimler AG. *Handbuch der Systemgestaltung und des Systembetriebs (HBSG+B)*, version 3.6 edition, 2007.

[Dai09a] Daimler AG. Service Repository and SOA Governance, 2009. Interner Bericht.

[Dai09b] Daimler AG. SOA-Governance-Prozesse, 2009. Interner Bericht.

[Dai10] Daimler AG. Produktänderungsmanagement in der Nutzfahrzeugentwicklung, 2010. Interner Bericht.

[DCGP05] S. Davies, L. Cowen, C. Giddings, and H. Parker. *WebSphere Message Broker Basics, IBM Redbook*. IBM Int'l Technical Support Organization, 2005.

[Dee07] M. Deeg. SOA Fängt weit vor BPEL an - Serviceorientierte Geschäftsprozessmodellierung als Basis für eine SOA, OBJEKTspektrum - Onlineausgabe, 2007.

[DEJL10] C. Dinnus, S. Ettrich, M. Jansen, and M. Lange. *Run SAP: Das umfassende Handbuch*. SAP PRESS, 2010.

[DEV+06] M. Di Penta, R. Esposito, M.L. Villani, R. Codato, M. Colombo, and E. Di Nitto. WS Binder: A Framework to Enable Dynamic Binding of Composite Web Services. In: *Proc. of the 2006 Int'l Workshop on Service-oriented Software Engineering*, SOSE '06, pages 74–80, New York, USA, 2006.

[DMW09] V. De Castro, E. Marcos, and R. Wieringa. Towards a Service-Oriented MDA-Based Approach to the Alignment of Business Processes with IT Systems: From the Business Model to a Web Service Composition Model. *Int'l Journal of Cooperative Information Systems*, 18(2):225–260, 2009.

[Dow98] T.B. Downing. *Java RMI: Remote Method Invocation*. IDG Books Worldwide, Inc. Foster City, CA, USA, 1998.

[DPRW08] A. Ditze, D. Peters, G. Rempp, and L. Wiegmann. M3 modellbasiert zum Ziel, Modeling Magazin, 2008.

[DR99] P. Dadam and M. Reichert, editors. *Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications*, volume 24 of *CEUR Workshop Proceedings*. 1999.

[DR09] P. Dadam and M. Reichert. The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support - Challenges and Achievements. *Computer Science - Research and Development*, 23(2):81–97, 2009.

[DRR11] P. Dadam, M. Reichert, and S. Rinderle-Ma. Prozessmanagementsysteme: Nur ein wenig Flexibilität wird nicht reichen. *Informatik-Spektrum*, 34(4):364–376, 2011.

[DRRM+09] P. Dadam, M. Reichert, S. Rinderle-Ma, K. Goeser, U. Kreher, and M. Jurisch. Von ADEPT zur AristaFlow BPM Suite - Eine Vision wird Realität: "Correctness by Constructionünd flexible, robuste Ausführung von Unternehmensprozessen. *EMISA Forum*, 29(1):9–28, 2009.

[DRS+07] C. Dudley, L. Rieu, M. Smithson, T. Verma, and B. Braswell. *Websphere Service Registry and Repository Handbook, IBM Redbook*. IBM, 2007.

[DS08] M. Deeg and E. Schmidt. SOA fängt vor BPEL an, White Paper. MID, 2008.

[EHH+08] G. Engels, A. Hess, B. Humm, O. Juwig, M. Lohmann, J.P. Richter, M. Voß, and J. Willkomm. *Quasar Enterprise: Anwendungslandschaften serviceorientiert gestalten*. Dpunkt-Verlag, 2008.

[EKW+09] T. Erl, A. Karmarkar, P. Walmsley, H. Haas, L.U. Yalcinalp, K. Liu, D. Orchard, A. Tost, and J. Pasley. *Web Service Contract Design and Versioning for SOA*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1 edition, 2009.

[ELK+06] C. Emig, K. Langer, K. Krutz, S. Link, C. Momm, and S. Abeck. The SOA's Layers, C+M Research Report, University of Karlsruhe. 2006.

[EM08] A. Erradi and P. Maheshwari. Dynamic Binding Framework for Adaptive Web Services. In: *3rd Int'l Conference on Internet and Web Applications and Services*, pages 162–167, Washington, DC, USA, 2008. IEEE Computer Society.

[End09] R. Enderle. Frühe fachliche Modellierung ausführungsrelevanter Prozess-Aspekte-Prozessmodellierung in Zeiten von SOA. Master's thesis, Universität Ulm, 2009.

[Erl05] T. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design, Book*. Prentice Hall Press Upper Saddle River, NJ, USA, 2005.

[Erl07] T. Erl. *SOA: Principles of Service Design*. Prentice Hall Press Upper Saddle River, NJ, USA, 2007.

[Erl09] T. Erl. *SOA Design Patterns*. Prentice Hall Press Upper Saddle River, NJ, USA, Upper Saddle River, NJ, USA, 1st edition, 2009.

[Erl10] T. Erl. *SOA with .NET*. Prentice Hall Press Upper Saddle River, NJ, USA, Upper Saddle River, NJ, USA, 2010.

[Esh02] E. Eshuis. *Semantics and Verification of UML Activity Diagrams for Workflow Modelling*. PhD thesis, University of Twente, 2002.

[FF08] S. Feja and D. Fötsch. Ein Framework für grafische Validierungsregeln. In: *Model Driven Integration Engineering*, pages 67 – 80. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, 2008.

[FFO05] H. Fischer, A. Fleischmann, and S. Obermeier. Subjektorientierte Modellierung und automatische Codegenerierung bei ERP-Prozessen. In: *ERP Management*, volume 4, pages 22–24, 2005.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and H. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, 1999.

[Fin11] F. Finkenzeller. Versionsübergreifende Konsistenzsicherung - Konzeption und Realisierung von Konsistenzanalysen in einer SOA. Master's thesis, Universität Ulm, 2011.

[FKT08] K.P. Fähnrich, S. Kühne, and M. Thränert. Model-Driven Integration Engineering-Modellierung, Validierung und Transformation zur Integration betrieblicher Anwendungssysteme. Technical report, Leipziger Informatikverbund (LIV), 2008.

[FLF+07] R. Fang, L. Lam, L. Fong, D. Frank, C. Vignola, Y. Chen, and N. Du. A Version-Aware Approach for Web Service Directory. In: *IEEE Int'l Conf. on Web Services (ICWS)*, pages 406–413, 2007.

[FLF+08] D. Frank, L. Lam, L. Fong, R. Fang, and M. Khangaonkar. Using an Interface Proxy to Host Versioned Web Services. In: *Services Computing, 2008. SCC'08. IEEE Int'l Conference on Service Computing*, volume 2, pages 325–332. IEEE, 2008.

[FN08] T. Freund and P. Niblett. ESB Interoperability Standards. White Paper, IBM Corporation, 2008.

[Ful06] L. Fulton. SOA-Repositories: Crucial To SOA Governance Success, forrester research, 2006.

[FWGH09] A. Fuhr, A. Winter, R. Gimnich, and T. Horn. Extending SOMA for Model-Driven Software Migration into SOA. In: *11th Workshop Software-Reengineering, Bad Honnef*, volume 29, pages 11–12, 2009.

[Gar03] T. Gardner. UML Modelling of Automated Business Processes with a Mapping to BPEL4WS. In: *Proc. of the First European Workshop on Object Orientation and Web Services at ECOOP*, volume 2003. Citeseer, 2003.

[GF94] O. C. Z. Gotel and A. C. W. Finkelstein. An Analysis of the Requirements Traceability Problem. In: *Proc. first Int'l Conference on Requirements Engineering*, pages 94–101. IEEE Computer Society Press, 1994.

[GP06] S. Guckenheimer and J.J. Perez. *Software Engineering with Microsoft Visual Studio Team System (Microsoft .NET Development Series)*. Addison-Wesley Professional, 2006.

[GR07] N. Gronau and M. Rohloff. Managing Change: Business / IT Alignment and Adaptability of Information Systems. In: *ECIS 2007, 15th European Conference on Information Systems, St. Gallen*, pages 1741–1753, 2007.

[Hal10] A. Hallerbach. *Management von Prozessvarianten*. PhD thesis, University of Ulm, 2010.

[Han10] I. Hanschke. *Strategic IT Management - A Toolkit for Enterprise Architecture Management*. Springer, 2010.

[HBR08a] A. Hallerbach, T. Bauer, and M. Reichert. Anforderungen an die Modellierung und Ausführung von Prozessvarianten. In: *Datenbank Spektrum*, volume 24, pages 48–58. dpunkt verlag, 2008.

[HBR08b] A. Hallerbach, T. Bauer, and M. Reichert. Context-based Configuration of Process Variants. In: *3rd Int'l Workshop on Technologies for Context-Aware Business Process Management (TCoB 2008)*, pages 31–40, 2008.

[HBR08c] A. Hallerbach, T. Bauer, and M. Reichert. Managing Process Variants in the Process Lifecycle. In: *10th Int'l Conf. on Enterprise Information Systems (ICEIS'08)*, pages 154–161, 2008.

[HBR08d] A. Hallerbach, T. Bauer, and M. Reichert. Modellierung und Darstellung von Prozessvarianten in Provop. In: *Modellierung'08 Conference*, number P-127 in Lecture Notes in Informatics (LNI), pages 41–56. Koellen-Verlag, 2008.

[HBR09] A. Hallerbach, T. Bauer, and M. Reichert. Issues in Modeling Process Variants with Provop. In: *4th Int'l Workshop on Business Process Design (BPD'08), Workshop held in conjunction with BPM'08 conference.*, number 17 in LNBIP, pages 56–67. Springer, 2009.

[HBR10a] A. Hallerbach, T. Bauer, and M. Reichert. Capturing Variability in Business Process Models: The Provop Approach. *Journal of Software Maintenance and Evolution: Research and Practice*, 22(6-7):519–546, 2010.

[HBR10b] A. Hallerbach, T. Bauer, and M. Reichert. Configuration and Management of Process Variants. In: M. Rosemann and J. von Brocke, editors, *Int'l Handbook on Business Process Management I*, pages 237–255. Springer, Berlin, 2010.

[Heu07] R. Heutschi. *Serviceorientierte Architektur Architekturprinzipien und Umsetzung in der Praxis*. Springer, 2007.

[Hew10] Hewlett-Packard Development Company. HP SOA Systinet Software - Data sheet, 2010.

[HHV06] A. Hess, B. Humm, and M. Voß. Regeln für serviceorientierte Architekturen hoher Qualität. *Informatik Spektrum*, 29(6):395 – 411, 2006.

[Hil07] B. Hilt. *Vorgedacht & angepasst: Referenzmodelle bieten als Lebende Leitfaden erprobtes und leicht umzusetzendes Prozesswissen für jedermann.* IDS Scheer AG, 2007.

[HM96] H. Herbst and T. Myrach. A Repository System for Business Rules. In: *Proc. of the 6th IFIP TC-2 Working Conference on Data Semantics: Database Applications Semantics*, DS-6, pages 119–139, London, UK, UK, 1996. Chapman & Hall, Ltd.

[HM00] J. Hendler and D.L. McGuinness. The DARPA Agent Markup Language. *IEEE Intelligent systems*, 15(6):67–73, 2000.

[Hol04] S. Holzner. *Eclipse*. O'Reilly Germany, 2004.

[HR08] M. Herzog and A. Richter. SOA benötigt Governance - Eine virtuelle Diskussion zwischen IBM und SAP über das optimale Repository. Technical report, IBM SAP Competence Center Walldorf, IBM WebSphere Technical Sales BPM, 2008.

[HSS01] A. Heuer, G. Saake, and K.U. Sattler. *Datenbanken kompakt.* MITP, 2001.

[HSS08] A. Heuer, G. Saake, and K.U. Sattler. *Datenbanken: Konzepte und Sprachen.* MITP bei Redline, 2008.

[HW07] S. Huth and T. Wieland. SOA Meets EPK - Serviceorientierte Geschäftsprozesse durchgängig modellieren. *OBJEKTspektrum - Onlineausgabe*, 2007.

[HW08] S. Huth and T. Wieland. Geschäftsprozessmodellierung mittels Software-Services auf Basis der EPK. In: *Service-orientierte Architekturen.* Gabler, 2008.

[IBM93] IBM Corporation. Message Queue Interface: Technical Reference, 1993.

[IBM05] IBM Corporation. IBM WebSphere MQ Workflow: Getting Started with Buildtime 3.6, Produktdokumentation, 2005.

[IBM07] IBM Corporation. Basic Governance Profile Sample Specification (IBM WSRR). IBM Corporation, 2007.

[IBM08a] *Getting Started with IBM WebSphere Process Server and IBM WebSphere Enterprise Service Bus Part 1: Development.* IBM Corp., Riverton, NJ, USA, 2008.

[IBM08b] IBM Corporation. Common Event Infrastructure, Version 6.1, White Paper, 2008.

[IBM08c] IBM Corporation. IBM WebSphere Process Server, Version 6.2, White Paper, 2008.

[IBM11] IBM Corporation. IBM DB2 UDB, Version 8, Product Manual, 2011.

[IDS05] IDS Scheer AG. *Business Process Management: ARIS Value Engineering-Concept, White Paper*, 2005.

[jCo09] jCom1. jPass! - Subjektorientierte Geschäftsprozessmodellierung, 2009.

[JF10] A. Jones and A. Freeman. Windows Presentation Foundation. *Visual C# 2010 Recipes*, page 800, 2010.

[Jos07] N.M. Josuttis. *SOA in Practice; The Art of Distributed System Design.* O'Reilly, 2007.

[JRH+04] M. Jeckle, C. Rupp, J. Hahn, B. Zengler, and S. Queins. *UML 2 glasklar.* Hanser Fachbuchverlag, 2004.

[KAB+04] M. Keen, A. Acharya, S. Bishop, A. Hopkins, S. Milinski, C. Nott, R. Robinson, J. Adams, and P. Verschueren. *Patterns: Implementing an SOA using an Enterprise Service Bus.* IBM Int'l Technical Support Organization, 2004.

[KBS05] D. Krafzig, K. Banke, and D. Slama. *Enterprise SOA.* Prentice Hall PTR, 2005.

[KGV00] M. Koetsier, P. Grefen, and J. Vonk. Contracts for Cross-Organizational Workflow Management. In: *Electronic Commerce and Web Technologies*, volume 1875 of *Lecture Notes in Computer Science*, pages 110–121. Springer Berlin, 2000.

[KK05] T. Kahl and F. Kupsch. Transformation und Mapping von Prozessmodellen in verteilten Umgebungen mit der ereignisgesteuerten Prozesskette. *EPK 2005*, page 54, 2005.

[KKB07] K. Klose, R. Knackstedt, and D. Beverungen. Identification of Services - A Stakeholder-based Approach to SOA Development and its Application in the Area of Production Planning. In: *ECIS*, volume 7, pages 1802–1814, 2007.

[KKL+02] A. Keller, G. Kar, H. Ludwig, A. Dan, and J.L. Hellerstein. Managing Dynamic Services: A Contract Based Approach to a Conceptual Architecture. In: *Network Operations and Management Symposium (NOMS'02), IEEE/IFIP*, pages 513 – 528, 2002.

[KKL+05] M. Kloppmann, D. Koenig, F. Leymann, G. Pfau, A. Rickayzen, C. von Riegen, P. Schmidt, and I. Trickovic. WS-BPEL Extension for People - BPEL4People. IBM Corporation, 2005.

[KKT06] S. Kühne, H. Kern, and M. Thränert. OrViA - Orchestrierung und Validierung integrierter Anwendungssysteme. Poster zur Statuskonferenz Forschungsoffensive Software Engineering 2006, 26.-28. Juni 2006 in Leipzig, 2006.

[Klü07] J. Klückmann. 10 Steps to Business-Driven SOA. IDS Scheer AG, 2007.

[KM10] O. Kiese and J. Müller. Pragmatisches IT-Landscaping: In großen Systemlandschaften zügig Transparenz schaffen. *OBJEKTspektrum*, pages 34–41, 2010.

[KNS92] G. Keller, M. Nüttgens, and A.W. Scheer. Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89, University of Saarland, Saarbrücken, 1992.

[Kop05] O. Kopp. Abbildung von EPKs nach BPEL anhand des Prozessmodellierungswerkzeugs Nautilus. Master's thesis, Institut für Architektur von Anwendungssystemen, Universität Stuttgart, 2005.

[KR09] V. Künzle and M. Reichert. Towards object-aware process management systems: Issues, challenges, benefits. In: *Proc. 10th Int'l Workshop on Business Process Modeling, Development, and Support (BPMDS'09)*, number 29 in LNBIP, pages 197–210. Springer, 2009.

[KR11a] V. Künzle and M. Reichert. PHILharmonicFlows: Towards a Framework for Object-aware Process Management. *Journal of Software Maintenance and Evolution: Research and Practice*, 23(4):205–244, 2011.

[KR11b] V. Künzle and M. Reichert. Striving for Object-aware Process Support: How Existing Approaches Fit Together. In: *1st Int'l Symposium on Data-driven Process Discovery and Analysis (SIMPDA'11)*, 2011.

[KRW12] J. Kolb, M. Reichert, and B. Weber. Using Concurrent Task Trees for Stakeholder-centered Modeling and Visualization of Business Processes. In: *S-BPM ONE 2012*, number 284 in CCIS, pages 237–251. Springer, 2012.

[KTS05] S. Kühne, M. Thränert, and A. Speck. Towards a Methodology for Orchestration and Validation of Cooperative E-Business Components. In: Matthew J. Rutherford, editor, *7th GPCE YRW 2005 Proceedings*, pages 29–34, Tallinn, Estonia, 2005.

[Kub98] M. Kubicek. Organisatorische Aspekte in flexiblen Workflow-Management-Systemen. Master's thesis, Universität Ulm, 1998.

[KWR11] V. Künzle, B. Weber, and M. Reichert. Object-aware Business Processes: Fundamental Requirements and their Support in Existing Approaches. *Int'l Journal of Information System Modeling and Design (IJISMD)*, 2(2):19–46, 2011.

[Lie07] D. Liebhart. *SOA goes real: Service-orientierte Architekturen erfolgreich planen und einführen.* Hanser, München, 2007.

[LKR+10] L.T. Ly, D. Knuplesch, S. Rinderle-Ma, K. Goeser, H. Pfeifer, M. Reichert, and P. Dadam. SeaFlows Toolset - Compliance Verification Made Easy for Process-Aware Information Systems. In: *Proc. CAiSE'10 Forum - Information Systems Evolution*, number 72 in LNBIP, pages 76–91. Springer, 2010.

[LR00] F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques.* Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 2000.

[LR11] M. Lohrmann and M. Reichert. Understanding Business Process Quality. In: *Advances in Business Process Management.* Springer, 2011.

[LWR10] A. Lanz, B. Weber, and M. Reichert. Workflow Time Patterns for Process-aware Information Systems. In: *Proc. Enterprise, Business-Process, and Information Systems Modelling: 11th Int'l Workshop BPMDS and 15th Int'l Conference EMMSAD at CAiSE 2010*, number 50 in LNBIP, pages 94–107. Springer, 2010.

[Mar05] F. Marschall. *Modelltransformationen als Mittel der modellbasierten Entwicklung von Software-Systemen.* PhD thesis, Institut für Informatik Technische Universität München, 2005.

[Mas07] D. Masak. *SOA? Serviceorientierung in Business und Software.* Springer-Verlag, Berlin, 2007.

[Mat05] M. Matsumura. Registry vs. Repository - A World of Difference, 2005.

[MBL07] D. Maurer, P. Büch, and M. Linke. Vom Geschäftsprozess zur Enterprise Architecture - ARIS Solution for Enterprise Architecture Management. Technical report, 2007.

[MD05] T. Marrs and S. Davis. *JBoss at Work: A Practical Guide.* O'Reilly Media, 2005.

[Mel10] I. Melzer. *Service-orientierte Architekturen mit Web Services: Konzepte-Standards-Praxis.* Springer, 2010.

[Men07] F. Menge. Enterprise Service Bus. In: *Free and Open Source Software Conference*, 2007.

[MID04] MID. *Mit Innovator und MDA vom Geschäftsprozess zum IT-System, White Paper*, 2004.

[MID07] MID. *EJB 3.0-Anwendungsentwicklung mit MDA - Anwendungsbeispiel der MID ModellierungsMethodik M3, White Paper*, 2007.

[MID08a] MID. *Embedded Engineering mit der Modellierungsplattform Innovator, White Paper*, 2008.

[MID08b] MID. Innovator Object - Objektorientiertes Software Engineering mit der UML, White Paper. White Paper, 2008.

[MLM+06] C.M. MacKenzie, K. Laskey, F. McCabe, P.F. Brown, and R. Metz. *Reference Model for Service Oriented Architecture. OASIS Committee Specification 1*, 2006.

[MLP08] R. Mietzner, F. Leymann, and M.P. Papazoglou. Defining Composite Configurable SaaS Application Packages Using SCA, Variability Descriptors and Multi-tenancy Patterns. In: *ICIW*, pages 156–161, 2008.

[MM03] J. Miller and J. Mukerji. MDA Guide Version 1.0.1. Technical report, Object Management Group (OMG), 2003.

[MM06] A. Mulholland and A.L. Macaulay. *Architecture and the Integrated Architecture Framework, White Paper*, 2006.

[MN04] J. Mendling and M. Nüttgens. Transformation of ARIS Markup Language to EPML. In: *Proc. of the 3rd GI Workshop on Event-Driven Process Chains (EPK 2004)*, pages 27–38. Citeseer, 2004.

[MN06] J. Mendling and M. Nüttgens. EPC Markup Language (EPML): An XML-based Interchange Format for Event-Driven Process Chains (EPC). *Information Systems and e-Business Management*, 4(3):245–263, 2006.

[MRB08] B. Mutschler, M. Reichert, and J. Bumiller. Unleashing the Effectiveness of Process-oriented Information Systems: Problem Analysis, Critical Success Factors, and Implications. *IEEE Transactions on Systems, Man, and Cybernetics (Part C)*, 38(3):280–291, 2008.

[MRH07] D. Müller, M. Reichert, and J. Herbst. Data-driven Modeling and Coordination of Large Process Structures. In: *15th Int'l Conf. on Cooperative Information Systems (CoopIS'07)*, number 4803 in LNCS, pages 131–149. Springer, 2007.

[MRH08] D. Müller, M. Reichert, and J. Herbst. A New Paradigm for the Enactment and Dynamic Adaptation of Data-driven Process Structures. In: *20th Int'l Conf. on Advanced Information Systems Engineering (CAiSE'08)*, number 5074 in LNCS, pages 48–63. Springer, 2008.

[MS95] S. T. March and G. F. Smith. Design and Natural Science Research on Information Technology. *Decis. Support Syst.*, 15(4):251–266, 1995.

[Nad04] E.G. Nadham. Seven Steps to a Service-Oriented Evolution. In: *Business Integration Journal*, volume 1, pages 41–44, 2004.

[Nak02] S. Nakajima. Model-Checking Verification for Reliable Web Service. In: *Proc. OOPSLA Workshop on Object-Oriented Web Services*, Seattle, Washington, 2002.

[Nat03] Y.V. Natis. Service-Oriented Architecture Scenario. Technical Report, Gartner Research ID Number: AV-19-6751, 2003.

[NR02] M. Nüttgens and F. Rump. Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In: *Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen (Promise 2002)*, 2002.

[OAS01a] OASIS. ebXML Business Process Specification Schema Version 1.01. Technical report, UN/CEFACT and OASIS, 2001.

[OAS01b] OASIS. ebXML Registry Information Model v2.0. Technical report, Organization for the Advancement of Structured Information Standards, 2001.

[OAS01c] OASIS. ebXML Registry Services Specification v2.0. Technical report, Organization for the Advancement of Structured Information Standards, 2001.

[OAS01d] OASIS. ebXML Technical Architecture Specification. Technical report, Organization for the Advancement of Structured Information Standards, 2001.

[OAS02a] OASIS. ebXML Message Service Specification v2.0. Technical report, Organization for the Advancement of Structured Information Standards, 2002.

[OAS02b] OASIS. OASIS: Universal Description, Discovery, and Integration (UDDI), Version 3.0, 2002.

[OAS05] OASIS. ebXML Registry Information Model v3.0 - Committee Draft Specification 02. Technical report, Organization for the Advancement of Structured Information Standards, 2005.

[OAS09] OASIS. Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.2, 2009.

[Obj09] Object Management Group. Business Process Modeling and Notation (BPMN) Specification 2.0. Technical report, 2009.

[ODv+09] C. Ouyang, M. Dumas, W.M.P. van der Aalst, A.H.M. Hofstede, and J. Mendling. From Business Process Models to Process-oriented Software Systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 19(1):1–37, 2009.

[OEH02] J. O'Sullivan, D. Edmond, and A.H.M Hofstede. What's in a Service? In: *Distributed and Parallel Databases 12*, volume 12, pages 117 – 133. Springer Netherlands, 2002.

[Off08] P. Offermann. SOAM - Eine Methode zur Konzeption betrieblicher Software mit einer Serviceorientierten Architektur. *Wirtschaftsinformatik*, 6:461–471, 2008.

[OMG05] OMG. Unified Modeling Language Specification, 2005.

[OMG06] OMG. Business Motivation Model (BMM) Specification. Technical report, Technical Report dtc/06–08–03, Object Management Group, Needham, Massachusetts, USA, 2006.

[OMG09a] OMG. *Business Process Model and Notation (BPMN) Specification 2.0, V0.9.14, revised submission draft*, 2009.

[OMG09b] OMG. Unified Modeling Language (UML) 2.2, 2009.

[Ora08] Oracle. Oracle Enterprise Repository and Oracle Service Registry for the SOA-Lifecycle. Oracle Data Sheet, 2008.

[Ora09] Oracle. Oracle Fusion Middleware Service Registry 11g (11.1.1). Product Documentation, 2009.

[OrV09] OrVia. OrVia Projektidee, 2009.

[OvDT06] C. Ouyang, W.M.P. van der Aalst, M. Dumas, and A.H.M. Ter Hofstede. Translating BPMN to BPEL. *BPM Center Report BPM-06-02, BPMcenter. org*, 2006.

[Pap03] M.P. Papazoglou. Service-Oriented Computing: Concepts, Characteristics and Directions. In: *Proc. of the 4th Int'l Conference on Web Information Systems Engineering*. IEEE Computer Society Washington, DC, USA, 2003.

[Pap08] M.P. Papazoglou. *Web Services - Principles and Technology*. Prentice Hall, 2008.

[Par98] H. Partsch. *Requirements-Engineering systematisch*. Springer Verlag, 1998.

[PD08] O. Pape and J. Drawehn. Anwendungsszenario Elektronische Steuererklärung. In: *Model Driven Integration Engineering*, pages 157 – 170. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, 2008.

[Pet81] J.L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.

[Pet05] R. Peterson. Get started with WebSphere Integration Developer. *IBM WebSphere Developer Technical Journal*, 7, 2005.

[PH07] M.P. Papazoglou and W.v.d. Heuvel. Business Process Development Life Cycle Methodology. *Commun. ACM*, 50(10):79–85, 2007.

[Pla02] D.S. Platt. *Introducing Microsoft. Net*. Microsoft Press Redmond, WA, USA, 2002.

[PNT10] M. Proctor, M. Neale, and E. Tirelli. JBoss Enterprise SOA Platform 5.0.1 JBoss Rules 5 Reference Guide. 2010.

[PQS+07] S. Pokraev, D. Quartel, M.W.A. Steen, A. Wombacher, and M. Reichert. Business Level Service-Oriented Enterprise Application Integration. In: *Proc. 3rd Int'l Conf. on Interoperability for Enterprise Software and Applications (I-ESA'07)*, pages 507–518. Springer, 2007.

[PQSM06] S. Pokraev, D. Quartel, M.W.A. Steen, and M.Reichert. A Method for Formal Verification of Service Interoperability. In: *Proc. IEEE Int'l Conf. on Web Services (ICWS'06)*, pages 895–900. IEEE Computer Press, 2006.

[PQSR06a] S. Pokraev, D. Quartel, M.W.A. Steen, and M. Reichert. Requirements and Method for Assessment of Service Interoperability. In: *Proc. 4th Int'l Conf. on Service Oriented Computing (ICSOC'06)*, number 4294 in LNCS, pages 1–14. Springer, 2006.

[PQSR06b] S. Pokraev, D. Quartel, M.W.A. Steen, and M. Reichert. Semantic Service Modeling - Enabling System Interoperability. In: *Proc. Int'l Conf. on Interoperability for Enterprise Software and Applications (I-ESA'06)*, 2006.

[PR05] O. Pera and B. Rintelmann. Von betrieblichen Geschäftsprozessen zu einer SOA. 18. Deutsche ORACLE-Anwenderkonferenz, 2005.

[Pry05] R. Pryss. Enterprise Application Integration. Master's thesis, 2005.

[RBBB10] M. Reichert, S. Bassil, R. Bobrik, and T. Bauer. The Proviado Access Control Model for Business Process Monitoring Components. *Enterprise Modelling and Information Systems Architectures*, 5(3):64–88, 2010.

[RBFD02] M. Reichert, T. Bauer, T. Fries, and P. Dadam. Modellierung planbarer Abweichungen in Workflow-Management-Systemen. In: *Proc. Modellierung 2002, Arbeitstagung der GI*, number P-12 in Lecture Notes in Informatics (LNI), pages 183–194. Koellen-Verlag, 2002.

[RCR04] M. Reichert, U. Catrinescu-Wiedemuth, and S. Rinderle. Evolution von Zugriffsregelungen in Informationssystemen. In: *Proc. Conf. eBusiness Processes (EBP'04)*, pages 100–114, 2004.

[RD97] M. Reichert and P. Dadam. A Framework for Dynamic Changes in Workflow Management Systems. In: *Proc. 8th Int'l Workshop on Database and Expert Systems Applications*, pages 42–48, 1997.

[RD98] M. Reichert and P. Dadam. ADEPTflex - Supporting Dynamic Changes of Workflows Without Losing Control. *Journal of Intelligent Information Systems, Special Issue on Workflow Management Systems*, 10(2):93–129, 1998.

[RD00] M. Reichert and P. Dadam. Geschäftsprozessmodellierung und Workflow-Management - Konzepte, Systeme und deren Anwendung. *Industrie Management*, 16(3):23–27, 2000.

[RDB03] M. Reichert, P. Dadam, and T. Bauer. Dealing with Forward and Backward Jumps in Workflow Management Systems. *Int'l Journal Software and Systems Modeling (SOSYM)*, 2(1):37–58, 2003.

[RDH05] H.P. Reiser, M.J. Danel, and F.J. Hauck. A Flexible Replication Framework for Scalable and Reliable .NET Services. In: *Proc. IADIS Intl. Conf. on Applied Computing*, pages 161–169, 2005.

[RDK+08] M. Reichert, D. Dadam, U. Kreher, M. Jurisch, and K. Göser. Architectural Design of Flexible Process Management Technology. In: *Proc. PRIMIUM Subconference at the Multikonferenz Wirtschaftsinformatik (MKWI)*, number 328 in CEUR Workshop Proceedings, 2008.

[RDRM+09] M. Reichert, P. Dadam, S. Rinderle-Ma, A. Lanz, R. Pryss, M. Predeschly, J. Kolb, L. Thao Ly, M. Jurisch, U. Kreher, and K. Goeser. Enabling Poka-Yoke Workflows with the AristaFlow BPM Suite. In: *Proc. BPM'09 Demonstration Track*, number 489 in CEUR Workshop Proceedings, 2009.

[Rec10] S. Rechtenbach. Durchgängige Modellierung: Vorgehen und Funktionalität im GPM-Tool. Master's thesis, Universität Ulm, 2010.

[Rei00] M. Reichert. *Dynamische Ablaufänderungen in Workflow-Management-Systemen.* PhD thesis, Universität Ulm, 2000.

[Rin04] S. Rinderle. *Schema Evolution in Process Management Systems.* PhD thesis, Universität Ulm, 2004.

[RKBB12] M. Reichert, J. Kolb, R. Bobrik, and T. Bauer. Enabling Personalized Visualization of Large Business Processes through Parameterizable Views. In: *27th ACM Symposium On Applied Computing (SAC'12), 9th Enterprise Engineering Track*, pages 1653–1660. ACM Press, 2012.

[Rog06] S. Rogers. CentraSite: An Integrated SOA Registry and Repository. *Elektronische Veröffentlichung. www.softwareag.com*, 11, 2006.

[Roh08] M. Rohmann. Managing Services Dynamically using WebSphere DataPower SOA Appliances with WebSphere Service Registry and Repository. Technical report, IBM developerWorks, 2008.

[RR06] M. Reichert and S. Rinderle. On Design Principles for Realizing Adaptive Service Flows with BPEL. In: *Proc. Workshop Methoden, Konzepte und Technologien für die Entwicklung von dienstbasierten Informationssystemen (EMISA'06)*, number P-95 in Lecture Notes in Informatics (LNI), pages 133–146. Koellen-Verlag, 2006.

[RR07] S. Rinderle-Ma and M. Reichert. A Formal Framework for Adaptive Access Control Models. *Journal on Data Semantics IX*, LNCS 4:82–112, 2007.

[RR08] S. Rinderle-Ma and M. Reichert. Managing the Life Cycle of Access Rules in CEOSIS. In: *Proc. of the 12th IEEE Int'l Enterprise Computing Conference (EDOC'08)*, pages 257–266. IEEE Computer Society Press, 2008.

[RR09] S. Rinderle-Ma and M. Reichert. Comprehensive Life Cycle Support for Access Rules in Information Systems: The CEOSIS Project. *Enterprise Information Systems*, 3(3):219–251, 2009.

[RRD03] M. Reichert, S. Rinderle, and P. Dadam. On the Common Support of Workflow Type and Instance Changes under Correctness Constraints. In: *Proc. 11th Int'l Conf. Cooperative Information Systems (CooplS '03)*, number 2888 in LNCS, pages 407–425. Springer, 2003.

[RRD04a] M. Reichert, S. Rinderle, and P. Dadam. On the Modeling of Correct Service Flows with BPEL4WS. In: *EMISA 2004 - Informationssysteme im E-Business und E-Government*, number P-56 in Lecture Notes in Informatics (LNI), pages 117–128, 2004.

[RRD04b] S. Rinderle, M. Reichert, and P. Dadam. On Dealing with Structural Conflicts between Process Type and Instance Changes. In: *Proc. 2nd. Int'l Conf. Business Process Management (BPM'04)*, number 3080 in LNCS, pages 274–289. Springer, 2004.

[RRD09] M. Reichert, S. Rinderle-Ma, and P. Dadam. Flexibility in Process-Aware Information Systems. *LNCS Transactions on Petri Nets and Other Models of Concurrency (ToPNoC), Special Issue on Concurrency in Process-Aware Information Systems*, 2:115–135, 2009.

[RRJ11] S. Rinderle-Ma, M. Reichert, and M. Jurisch. On Utilizing Web Service Equivalence for Supporting the Composition Life Cycle. *Int'l Journal of Web Service Research*, 8(1):41–67, 2011.

[RS95] E. Rescorla and A. Schiffman. The Secure HyperText Transfer Protocol. Technical report, Enterprise Integration Technologies, 1995.

[RS04] M. Reichert and D. Stoll. Komposition, Choreographie und Orchestrierung von Web Services - Ein Überblick. *EMISA Forum*, 24(2):21–32, 2004.

[Rup07] C. Rupp. *Requirements-Engineering und-Management: professionelle, iterative Anforderungsanalyse für die Praxis*. Hanser, 2007.

[RW12] M. Reichert and B. Weber. *Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies*. Springer, 2012.

[RWR06a] S. Rinderle, A. Wombacher, and M. Reichert. Evolution of Process Choreographies in DYCHOR. In: *Proc. 14th Int'l Conf. on Cooperative Information Systems (CooplS'06)*, volume 14 of *LNCS*, pages 273–290. Springer, 2006.

[RWR06b] S. Rinderle, A. Wombacher, and M. Reichert. On the Controlled Evolution of Process Choreographies. In: *Proc. 22nd Int'l Conf. on Data Engineering (ICDE'06)*, page 124. IEEE Computer Society Press, 2006.

[Sac07] N. Sachdeva. Customize the WebSphere Servcie Registry und Repository Governance Profil. IBM Corporation, 2007.

[Sch00] A.W. Scheer. *ARIS: Business Process Modelling*. Springer-Verlag, Berlin, 2000.

[Sch01] A.W. Scheer. *ARIS-Modellierungsmethoden, Metamodelle, Anwendungen*. Springer, 2001.

[Sch04] R. Scheuch. Mit der UML 2.0 Geschäftsprozesse modellieren. *OBJEKTspektrum*, 3:30–33, 2004.

[Sch09] M. Scheuermann. PAI bei der Daimler AG - Warum proaktive Produktlinienentwicklung SOA erst möglich macht, Java Forum Stuttgart, 2009.

[Sch10] J. Schmitzl. Durchgängige Modellierung von prozessorientierten Anwendungen mit BPMN 2.0. Master's thesis, Universität Ulm, 2010.

[Shu08] L. Shuster. Project-Oriented SOA. *SOA Magazin*, XXI, 2008.

[SI07] S. Stein and K. Ivanov. Fachliche Beschreibung eines Services. In: *SOA-Expertenwissen: Methoden, Konzepte und Praxis serviceorientierter Architekturen*, pages 213 – 227. G. Starke and S. Tilkov, 2007.

[SKD+08] S. Stein, S. Kühne, J. Drawehn, S. Feja, and W. Rotzoll. Evaluation of OrViA Framework for Model-Driven SOA Implementations: An Industrial Case Study. In: Marlon Dumas, Manfred Reichert, and Ming-Chien Shan, editors, *Business Process Management: 6th Int'l Conference*, pages 310–325. Springer, LNCS 5240, Milan, Italy, 2008.

[SOA00] Simple Object Access Protocol (SOAP) 1.1, 2000.

[Sof05] Software AG. ARIS Process Performance Manager, White Paper, 2005.

[Sof06] Software AG. *ARIS Platform - Methode 7.0 Toolset Dokumentation*, 2006.

[Sof09] Software AG. ARIS Value Engineering, 2009.

[SR08] S. Stein and U. Roediger. Servicebeschreibung am Beispiel eines virtuellen Autokonzerns. *OBJEKTspektrum*, 2008.

[ST07] G. Starke and S. Tilkov, editors. *SOA-Expertenwissen: Methoden, Konzepte und Praxis serviceorientierter Architekturen.* dpunkt, Heidelberg, 2007.

[Sta02] S. Staab. Wissensmanagement mit Ontologien und Metadaten. *Informatik-Spektrum*, 25(3):194–209, 2002.

[Ste07] S. Stein. *Von EPK nach BPEL – Technische Einblicke in ARIS SOA Architect.* Saarbrücken, Germany, 2007. ARIS Online Tutorial.

[Ste08a] S. Stein. Fachliche Servicebeschreibung am Beispiel eines virtuellen Autokonzerns, Expert Paper. Technical report, IDS Scheer, 2008.

[Ste08b] S. Stein. *Modelling Method Extension for Service-Oriented Business Process Management.* PhD thesis, Christian-Albrechts-Universität zu Kiel, 2008.

[Stu05] A. Stuckenholz. Component Evolution and Versioning State of the Art. *ACM SIGSOFT Software Engineering Notes*, 30(1):7, 2005.

[Tie10] J. Tiedeken. Konzeption und Realisierung eines logisch zentralen SOA-Repositories. Master's thesis, Universität Ulm, 2010.

[TLD+07] O. Thomas, K. Leyking, F. Dreifus, M. Fellmann, and P. Loos. Serviceorientierte Architekturen: Gestaltung, Konfiguration und Ausführung von Geschäftsprozessen. *Veröffentlichungen des Instituts für Wirtschaftsinformatik im Deutschen Forschungszentrum für künstliche Intelligenz*, 189, 2007.

[TLS09] O. Thomas, K. Leyking, and M. Scheid. Vorgehensmodelle zur Entwicklung Serviceorientierter Softwaresysteme. In: *Wirtschaftsinformatik (1)*, volume 246 of *books@ocg.at*, pages 181–192. Österreichische Computer Gesellschaft, 2009.

[vD05] O. von Susani and P. Dugerdil. Contract-based cross-organizational automated processes. In: *7th IEEE Int'l Conference on E-Commerce Technology, 2005. CEC 2005.*, pages 540–543. IEEE, 2005.

[W3C06a] Web Services Description language (WSDL) Version 2.0, W3C Working Draft, 2006.

[W3C06b] Web Services Policy 1.2 - Framework (WS-Policy), 2006.

[WB07] V. Winkler and H.U. Buhl. Identifikation und Gestaltung von Services. *Wirtschaftsinformatik*, 49(4):257–266, 2007.

[Wes07] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag New York Inc, 2007.

[WFM99] WFMC. Workflow Management Coalition Terminology Glossary, 1999.

[WGL+09] M. Weidlich, A. Grosskopf, D. Lübke, K. Schneider, E. Knauss, and L. Singer. Verzahnung von Requirements Engineering und Geschäftsprozessdesign. In: *Workshops für Requirements Engineering und Business Prcess Management (REBPM 2009), Kaiserslautern, Deutschland*, 2009.

[Whi05] S. White. Using BPMN to Model a BPEL Process. *BPTrends*, 3(3):1–18, 2005.

[WHMN07] I. Weber, J. Hoffmann, J. Mendling, and J. Nitzsche. Towards a Methodology for Semantic Business Process Modeling and Configuration. In: *SeMSoC-07: Proc. of the 2nd Int'l Workshop on Business Oriented Aspects concerning Semantics and Methodologies in Service-oriented Computing*, 2007.

[WLD+06] D. Werth, K. Leyking, F. Dreifus, J. Ziemann, and A. Martin. Managing SOA Through Business Services - A Business-Oriented Approach to Service-Oriented Architectures. In: *ICSOC Workshops*, volume 4652 of *Lecture Notes in Computer Science*, pages 3–13. Springer, 2006.

[WLD+07] D. Werth, K. Leyking, F. Dreifus, J. Ziemann, and A. Martin. Managing SOA Through Business Services–A Business-Oriented Approach to Service-Oriented Architectures. *Service-Oriented Computing ICSOC 2006*, pages 3–13, 2007.

[WMW12] M. Weidlich, J. Mendling, and M. Weske. Propagating Changes between Aligned Process Models. *Journal of Systems and Software*, 85(8):1885–1898, 2012.

[WRR07] B. Weber, S. Rinderle, and M. Reichert. Change Patterns and Change Support Features in Process-Aware Information Systems. In: *Proc. 11th Int'l Conf. on Advanced Information Systems Engineering (CAiSE'07)*, number 4495 in LNCS, pages 574–588. Springer, 2007.

[WRR08] B. Weber, M. Reichert, and S. Rinderle-Ma. Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. *Data and Knowledge Engineering*, 66(3):438–466, 2008.

[WRWR09] B. Weber, M. Reichert, W. Wild, and S. Rinderle-Ma. Providing Integrated Life Cycle Support in Process-Aware Information Systems. *Int'l Journal of Cooperative Information Systems (IJCIS)*, 18(1):115–165, 2009.

[WSI06] Web Service Interoperability (WS-I) Basic Profile 1.1, 2006.

[WSR09] B. Weber, S. Sadiq, and M. Reichert. 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, 2009.