Modellierung von Service-Aufrufbeziehungen zwischen prozessorientierten Applikationen

Stephan Buchwald, Thomas Bauer
Group Research & Advanced Engineering, Daimler AG,
{stephan.buchwald, thomas.tb.bauer}@daimler.com

Zusammenfassung: In Unternehmen existiert oftmals eine Vielzahl von heterogenen Informationssystemen für die Bereitstellung und Verarbeitung von Geschäftsdaten. Die Integration dieser Informationssysteme stellt eine große Herausforderung dar, insbesondere aufgrund fehlender Standardisierung und Detaillierung der Dokumentation solcher IT-Landschaften. Problematisch ist, dass in vielen Unternehmen die verschiedenen Applikationen und Geschäftsprozesse sowie die Abhängigkeiten zwischen ihnen (d.h. angebotene Schnittstellen und deren Verwendung) nicht vollständig bekannt sind. Dadurch entstehen sehr unübersichtliche, heterogene und nur schwer erweiterbare IT-Landschaften. Dieser Beitrag entwickelt eine Methode zur Modellierung und Spezifikation der Abhängigkeiten insbesondere zwischen prozessorientierten Applikationen.

1 Einleitung

Infolge der Integration und Konfiguration heterogener und nur wenig standardisierter Systeme weisen IT-Landschaften heutiger Unternehmen eine hohe Komplexität auf. Aufgrund der Vielzahl an Applikationen (oft mehrere hundert) sowie der fehlenden oder nur unzureichenden Dokumentation der Beziehungen zwischen ihnen und den im Unternehmen ablaufenden Geschäftsprozessen, entsteht eine nur schwer wartbare IT-Landschaft. Verstärkt wird dies durch die ständig wachsende Anzahl von Applikationen und dem regen Datenaustausch zwischen ihnen. Die Kopplung dieser Systeme ist oft entweder unvollständig oder sehr heterogen. Auch wird häufig auf eine solche verzichtet und es findet eine manuelle Datenübernahme statt. In Abbildung 1a wird eine sehr vereinfachte Unternehmenslandschaft dargestellt. Applikationen und Prozesse kommunizieren dabei bspw. über einen Nachrichtenaustausch (per Middleware wie z.B. IBM WebSphere MQ), eine gemeinsame Datenbank, einen File-Transfer und oftmals auch über eine manuelle Datenübernahme, was leicht zu inkonsistenten und fehlerhaften Daten führt. Deshalb ist eine einheitliche Dokumentation aller Informationen über die Unternehmenslandschaft essentiell. Im Einzelnen sind das Informationen darüber, welche Applikationen und Geschäftsprozesse an welchen Ereignissen interessiert sind, welche Operationen in Folge eines Ereignisses ausgeführt werden sollen und wie die Daten hierfür transformiert werden müssen. Abbildung 1b zeigt zwei Ereignisse die während der Ausführung des Entwicklungs- und Änderungsprozesses erzeugt werden: Der Entwicklungsprozess erstreckt sich von der Einreichung einer Idee zur Verbesserung eines Bauteils bis hin zur abschließenden Bewertung. Während

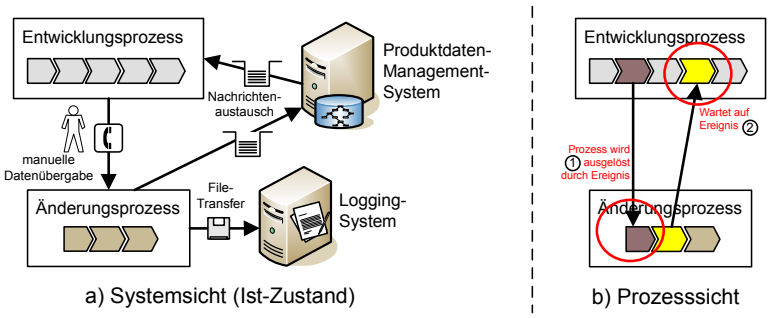

Abbildung 1: Datenaustausch zwischen Prozessen und Applikationen

der Ausführung des Entwicklungsprozesses, wird ein Änderungsprozess gestartet, um zu prüfen, ob die Verbesserungsidee Auswirkung auf andere Bauteile hat. Der Änderungsprozess wird durch ein Ereignis aus dem Entwicklungsprozess gestartet (vgl. ① in Abbildung 1b). Der Entwicklungsprozess wartet anschließend, bis eine entsprechende Genehmigung durch einen Verantwortlichen aus dem Änderungsprozess kommuniziert wird (vgl. ② in Abbildung 1b). Diese Abhängigkeiten beschreiben Aufrufbeziehungen zwischen Applikationen und Geschäftsprozessen, die mittels Ereignissen in Zielapplikationen definierte Operationen ausführen. Damit die verschiedene Applikationen untereinander Daten austauschen können, ist je Interaktion zwischen zwei Systemen eine Konvertierung der Daten notwendig. Transformationsregeln müssen applikationsspezifische Datenobjekte (ASBO) in globale Datenobjekte (GBO) transformieren, um diese dann in einer einheitlichen Form anderen Applikationen zur Verfügung zu stellen. Somit können sich Applikationen und Geschäftsprozesse für bestimmte Ereignisse registrieren und zugleich ihre Daten entsprechend dem GBO transformieren. Um die Modellierung der skizzierten Abhängigkeiten zu ermöglichen, ist eine werkzeugseitige Unterstützung erforderlich. Hierbei ist es wichtig, bereits dokumentierte Geschäftsprozessmodelle und Datentypen bei der Modellierung von Abhängigkeiten weiter verwenden zu können. Außerdem müssen die in den Systemen auftretenden relevanten Ereignissen und die Abhängigkeiten zwischen Ereignisse und Schnittstellen (Aufruf einer Operation oder Funktion) modellierbar sein.

Dieser Beitrag wurde im Rahmen des Projektes *Enproso (Enhanced Process Management by Service Orientation)* erstellt. Dementsprechend liegt der Schwerpunkt auf Geschäftsprozessen und prozessorientierten Applikationen. Der Beitrag beschreibt, wie Ereignisse und Schnittstellen explizit dokumentiert werden können. Anschließend zeigen wir, wie Aufrufbeziehungen zwischen Applikationen und Geschäftsprozessen detailliert modelliert werden. Aus diesen Informationen kann ein Gesamtüberblick der Aufrufbeziehungen zwischen Applikationen und Geschäftsprozessen abgeleitet werden, welcher als Basis für die Planung der zukünftigen IT-Landschaft des Unternehmens oder zur Implementierung weiterer Applikationen verwendet werden kann. Andererseits kann durch solch eine Dokumentation sichergestellt werden, dass Änderungen an Applikationen und Geschäftsprozessen sowie den zugehörigen Aufrufbeziehungen keine unerwarteten Folgen haben. D.h. Änderungen sind transparent und nachvollziehbar. Die Gesamtübersicht dient zudem als Basis für Analysen und Optimierungen der IT-Landschaft des Unternehmens, im Sinne

eines detaillierten *Enterprise Architecture Management* [MBL07, EHH+08, BELM08]. So geben die modellierten Informationen Auskunft darüber, welche Personen z.B. bei der Veränderung eines Geschäftsprozesses zustimmen müssen oder welche laufenden Applikationen dadurch beeinflusst werden. Schließlich kann aus der dokumentierten Information ein Modell zur Implementierung (z.B. als J2EE-Implementierung), Generierung (z.B. Message-Broker-Abläufe) oder Konfiguration einer generischen Integrationsinfrastruktur (vgl. [Bau05, Buc05]) abgeleitet werden.
In Abschnitt 2 wird ein Anwendungsszenario eingeführt. Unterschiedliche Sichtweisen auf die IT-Landschaft werden in Abschnitt 3 (fachliche Sicht für Prozess-Modellierer) und Abschnitt 4 (IT-Sicht für IT-Abteilungen oder IT-Architekten) detailliert. Nach einer Diskussion zur Nutzung der modellierten Information in Abschnitt 5, wird der aktuelle Stand der Technik betrachtet, bevor dieser Beitrag mit einer Zusammenfassung und einem Ausblick schließt.

2 Anwendungsszenario

Wir skizzieren zunächst ein typisches Anwendungsszenario aus der Automobilindustrie, entlang dessen wir später die Modellierung von Service-Aufrufbeziehungen zwischen prozessorientierten Applikationen diskutieren. Das Anwendungsbeispiel in Abbildung 2 stellt eine vereinfachte IT-Landschaft dar. Dabei werden Beziehungen zwischen Applikationen und Geschäftsprozessen explizit durch Richtungspfeile gekennzeichnet.
Der Entwicklungsprozess erstreckt sich von der Einreichung einer Verbesserungsidee bis hin zu deren abschließenden Bewertung [HBR08] (① in Abbildung 2). Er wird durch Einreichung einer Idee zur Verbesserung eines Bauteils oder einer Baugruppe gestartet. Nach detaillierter Spezifikation der Verbesserungsidee wird automatisch ein Änderungsantrag gestartet. Änderungsanträge werden in einem separaten, ebenfalls prozessorientierten, Änderungs-Management-System bearbeitet. Genauer betrachtet führt das vom Entwicklungsprozess erzeugte Ereignis „Detaillierung abgeschlossen“ zur Instanziierung eines Prozesses im Änderungs-Management-System (② in Abbildung 2). Hierzu wird die Schnittstelle („Starte Änderungsprozess“, ③) des Änderungs-Management-Systems verwendet, die beim Aufruf eine Instanz des Änderungsprozesses startet.
Der erstellte Änderungsantrag wird durch einen Fahrzeugentwickler detailliert (④ in Abbildung 2). Dann werden durch den Prototypenbau (PT), die Produktionsplanung (PP) und ggf. weitere Bereiche Stellungnahmen zu den jeweiligen Anträgen abgegeben. Anschließend entscheidet ein Gremium über die Genehmigung oder Ablehnung des gestellten Antrags. Im Entwicklungsprozess stehen nach der Detaillierung der Verbesserungsidee zwei Aktivitäten („Angebot beim Zulieferer einholen“⑤ und „Konstruktive Umsetzung“⑥) zur Ausführung bereit. Anzumerken ist, dass das Einholen eines Angebotes beim Zulieferer unabhängig von der Genehmigung im Änderungsprozess abläuft. Lediglich die Beendigung der Aktivität „Detaillierung der Änderung“④ muss abgewartet werden. Um möglichst frühzeitig mit ⑤ im Entwicklungsprozess fortfahren zu können, muss die Information über die Beendigung von ④ an den Entwicklungsprozess gemeldet werden. Dies geschieht durch das Auslösen des Ereignisses „Detaillierung beendet“ ⑦ und des Aufrufs

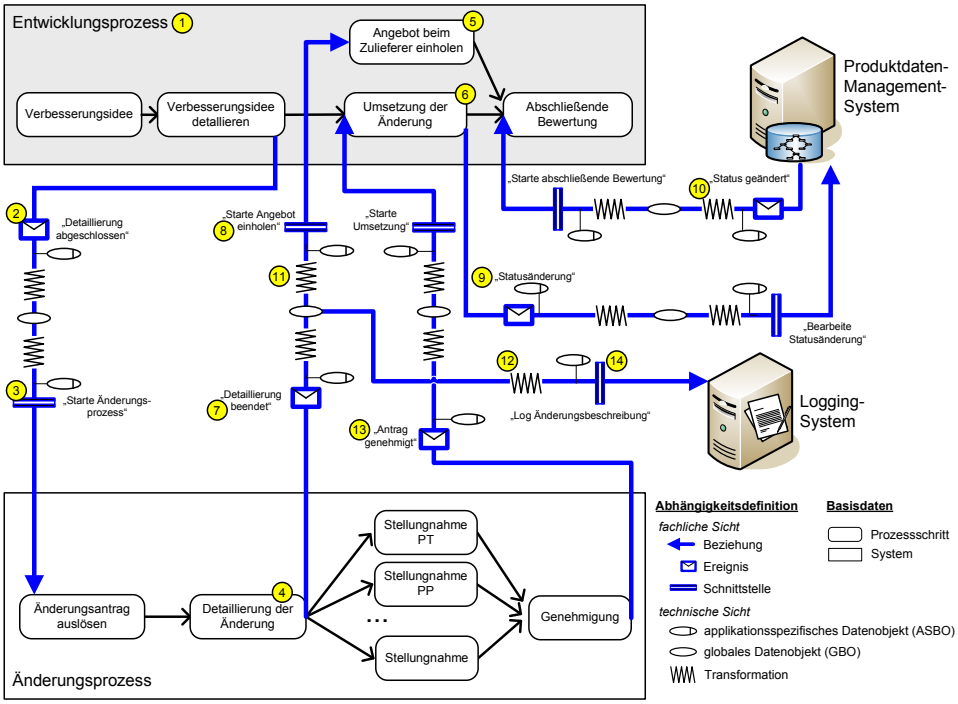

Abbildung 2: Beispielszenario für Abhängigkeiten zwischen Applikationen

der Schnittstelle „starte Angebot einholen“ ⑧. Die Umsetzung der Verbesserungsidee ⑥ im Entwicklungsprozess kann erst nach Abschluss der Genehmigung des Änderungsprozesses ausgeführt werden.

Das Produktdaten-Management-System, Basissystem für die Umsetzung von Änderungen, repräsentiert schließlich eine Alt-Applikation, die verschiedene Freigabestati (z.B. Bauteilstatus „Gesperrt“ oder „Hüllgeometrie freigegeben“) durchläuft. Eine Statusänderung ⑨ aus dem Entwicklungsprozess geht an das Produktdaten-Management-System. Dort wird der neue Status gespeichert und die eigentliche Arbeit der konstruktiven Umsetzung findet statt. Nach deren Abschluss meldet das Produktdaten-Management-System den geänderten und bearbeiteten Status zurück an den Entwicklungsprozess ⑩, der daraufhin mit der abschließenden Bewertung fortfahren kann. Das Logging-System ist ebenfalls eine Alt-Applikation, die bestimmte Informationen (hier das Beenden der Aktivitäten „Detaillierung der Änderung“④) explizit protokolliert.

Um die bisher beschriebenen Aufrufbeziehungen abbilden zu können, ist ein entsprechendes Werkzeug notwendig. Das in diesem Beitrag vorgestellte Modellierungswerkzeug ermöglicht es, solche Abhängigkeiten zwischen beliebigen Applikationen und Geschäftsprozessen vollständig und übersichtlich zu dokumentieren. Hierfür werden getrennte Modelltypen für fachliche und technische Aspekte eingeführt, um es verschiedenen Nutzergruppen zu ermöglichen, die jeweils ihnen bekannten Informationen zu dokumentieren.

Fachliche Sicht: Modellierer sind in der Lage, Abhängigkeiten zwischen Geschäftsprozessen und sonstigen Applikationen auf einer eher abstrakten Ebene zu dokumentieren. Dabei gestalten sie Modelle, in denen Applikationen, (bereits existierende) Geschäftsprozesse bzw. einzelne Prozessschritte untereinander in Beziehung gesetzt werden. Dies geschieht durch die Modellierung von Aufrufbeziehungen zwischen Ereignissen und Schnittstellen unterschiedlicher Applikationen und Geschäftsprozesse.

Technische Sicht: Basierend auf der fachlichen Sicht werden in einem nachgelagerten Arbeitsschritt technische Details dokumentiert. Dies wird durch einen IT-Spezialist realisiert. Hierbei stehen nicht nur die Applikationen und Geschäftsprozesse selbst im Fokus, sondern insbesondere Datenformate und Datentransformationen zwischen diesen. Beispielsweise resultiert aus dem Ereignis „Detaillierung beendet“(⑦ in Abbildung 2) resultiert ein applilkationsspezifisches Datenobjekt (ASBO) mit einem für das Änderungs-Management-System typischen Datenformat. In der technischen Sicht wird das ASBO selbst und die Transformation dieses in ein globales Datenobjekt (GBO) beschrieben. Ausgehend davon, finden weitere Transformationen (⑪ und ⑫) für die spezifischen Zielsysteme (Entwicklungsprozess und Logging-System) statt.

3 Modellierung der fachlichen Sicht

In der fachlichen Sicht gestalten Modellierer *Abhängigkeitsmodelle*, in denen Applikationen, (bereits existierende) Geschäftsprozesse sowie einzelne Prozessschritte miteinander in Beziehung gesetzt werden: Quellsysteme (Applikationen oder Geschäftsprozesse) senden Ereignisse, die von Modellierern über Beziehungskanten mit den von den Zielsystemen (Applikationen oder Geschäftsprozesse) bereitgestellten Schnittstellen verbunden werden. So entstehen Aufrufbeziehungen zwischen Quell- und Zielsystem. Ereignisse publizieren in Systemen eingetretene Datenänderungen. Eine Schnittstelle wiederum repräsentiert einen Teil einer Applikation, welche die Kommunikation und den Austausches von Daten mit anderen Applikationen spezifiziert (bspw. Web-Services). Die zu entwerfende Modellierungsmethodik muss also Objekttypen und Kanten bereitstellen, um Aufrufbeziehungen zwischen Systemen, Ereignisse und Schnittstellen modellieren zu können. Neben der Modellierung erwähnter Objekte ist die Wiederverwendung bereits existierender Geschäftsprozessmodelle wichtig. Geschäftsprozessmodelle die bereits dokumentiert sind, sollenin die Abhängigkeitsmodelle integriert werden können.

3.1 Existierende Methoden und Werkzeuge zur Modellierung

Es gibt eine Vielzahl von Ansätzen und Methodiken zur Definition von Prozessmodellen. Jedoch existiert kein Ansatz, der die zur Modellierung von Abhängigkeitsmodellen benötigte Funktionalität vollständig bereitstellt (vgl. [Buc07]). Deshalb ist die Erweite-

rung einer existierenden Methodik notwendig. Die Herausforderung besteht nun darin, einen geeigneten Ansatz auszuwählen und diesen mit Hilfe zusätzlicher Objekt-, Kanten- und Symboltypen um Ereignisse, Schnittstellen und spezielle Beziehungskanten zu erweitern. Dadurch sollen zum einen Geschäftsprozesse modellierbar werden und zum anderen diese mit Applikationen oder anderen Geschäftsprozessen in Beziehung gesetzt werden können. Insbesondere die Wiederverwendung von bereits dokumentierten Geschäftsprozessen steht dabei im Vordergrund. Ebenso wichtig ist die explizite Modellierung von Aufrufbeziehungen zwischen Applikationen und Geschäftsprozessen auf eine für Modellierer verständliche Weise.

Text-basierte Technologien (XML, HTML, etc.) sowie graphische, aber nicht prozessorientierte Ansätze (z.B. RDF [MM04], UML-Sequenzdiagramme [Obj07b], klassische Petrinetze [Pet62]) sind ungeeignet, da die Integration und Wiederverwendung bereits existierender Geschäftsprozessmodelle schwierig ist. Andere Technologien wie etwa Workflow-Modelle (z.B. erweiterte Petrinetze [Aal98], Service-Orchestrierung via BPEL [AAA+07], FDL [LR00], etc.) wiederum weisen eine sehr technische und für Modellierer unverständliche Sicht auf. Deshalb sind sie für die Modellierung von Abhängigkeiten auf der fachlichen Ebene ebenfalls ungeeignet. Eine detaillierte Diskussion unterschiedlicher Technologien und Ansätze wird in [Buc07] geführt.

In unserem Kontext interessant sind Geschäftsprozessmodelle. Sie bieten den Modellierern eine verständliche und vertraute Methodik zur Beschreibung von Geschäftsprozessmodellen. Durch die Erweiterung von Geschäftsprozess-Metamodellen wird eine komfortable Modellierung von Aufrufbeziehungen möglich, und das in einer für Modellierer bekannten Umgebung. Kandidaten für entsprechende Metamodelle sind beispielsweise erweiterte ereignisgesteuerte Prozessketten (eEPK), UML-Aktivitätsdiagramme oder BPMN-Diagramme [BPM06]. Wir wählen hier eEPKs als Basis-Methodik zur Modellierung von Aufrufbeziehungen aufgrund ihrer großen Menge an Grundobjekten und ihrer weiten Verbreitung in der Praxis. Durch die leicht verständliche Dokumentationsmethodik wird eine einfache Gestaltung semi-formaler Geschäftsprozessmodelle möglich. Erweiterte EPKs erlauben zudem die Verwendung einer Vielzahl unterschiedlicher Objekt- und Modelltypen zur Visualisierung von Aufrufbeziehungen, Ereignissen und Schnittstellen [Buc07].

Im Folgenden wird ein Ansatz vorgestellt, der mit eEPKs als Basis-Notation und ARIS [IDS06] als Entwicklungsumgebung, Aufrufbeziehungen modellierbar macht und die Weiterverwendung bereits existierender Geschäftsprozessmodellen ermöglicht.

Eine Alternative dazu ist die Verwendung von Aktivitätsdiagrammen oder anderen Geschäftsprozess-Metamodellen. Dies ist insbesondere dann zu bevorzugen, wenn bereits Geschäftsprozessmodelle in der entsprechenden Notation (z.B. als UML-Aktivitätsdiagramme) vorliegen und somit wieder verwendet werden können. Die im Folgenden vorgestellten Konzepte können vom Prinzip her auch auf andere Geschäftsprozess-Metasprachen übertragen werden.

3.2 Modellierungsmethodik zur Definition von Abhängigkeiten

Neben den ARIS-Standardobjekttypen spielen Ereignisse und Schnittstellen eine zentrale Rolle bei der Modellierung von Abhängigkeitsmodellen: Ereignisse realisieren ausgehende Aktionen zu anderen Applikationen, indem sie geschäftsrelevante Informationen wie die Freigabe eines Bauteils oder die Genehmigung eines Änderungsantrages (⑬ in Abbildung 2) symbolisieren. Schnittstellen hingegen repräsentieren Dienste bzw. Operationen, die ein Geschäftsprozess oder eine Applikation anbietet, etwa das Starten eines Änderungsprozesses (③ in Abbildung 2) oder die Protokollierung einer Änderungsbeschreibung (⑭). Um Aufrufbeziehungen modellieren zu können, müssen für die Modellierer ansprechende und verständliche Objekttypen definiert werden. Abbildung 3 zeigt die von uns in dieser Arbeit neu eingeführten Objekttypen *Ereignis* (1), *Ereigniskante* (2), *Schnittstelle* (3) und *Schnittstellenkante* (4) am Beispiel des Entwicklungsprozesses ① aus Abbildung 2.

Modellierer können entweder bereits bei der Entwicklung von Geschäftsprozessmodellen die entsprechenden Objekte modellieren oder diese nachträglich in bereits existierende Geschäftsprozessmodelle einfügen. Ein Quellobjekte ist entweder ein einzelner Prozessschritt in einem Geschäftsprozessmodell, das Geschäftsprozessmodell selbst oder eine beliebige Applikationen. Abbildung 3 zeigt Ereignisse (1) die über gerichtete Ereigniskanten (2) mit Quellobjekten („Verbesserungsidee detaillieren“) verbunden werden. Ereigniskanten besitzen meist eine Bedingung, die Auskunft darüber gibt, unter welchen Umständen das Ereignis tatsächlich ausgelöst wird. Schnittstellenkanten (4) werden verwendet, um Schnittstellen (3) mit Zielobjekten („Umsetzung der Änderung“) zu verbinden. Zur Unterscheidung von Ereigniskanten werden Schnittstellenkanten durch gestrichelte Pfeile symbolisiert. Die in Abbildung 3 dargestellte eEPK zeigt ein für Modellierer übersichtliches und einfaches Geschäftsprozessmodell. In der Realität sind Geschäftsprozessmodelle jedoch sehr viel komplexer und zudem meist hierarchisch strukturiert. Zudem existieren Aufrufbeziehungen zwischen einer Vielzahl von Geschäftsprozessen und Applikationen. Aufrgund der dadurch entstehenden Komplexität führen wir im Folgenden einen weiteren Diagrammtyp ein, welcher eine abstrahierte Sichtweise auf Applikationen, Geschäftsprozesse, Schnittstellen und Ereignisse sowie deren Zusammenspiel ermöglicht. Für diese Sichtweise sind Hinterlegungen für Objekte in ARIS notwendig. Dadurch wird es möglich, Objekte durch detaillierte Modelle zu verfeinern und somit die Sichtweise zu abstrahieren. Gekennzeichnet sind Hinterlegungen durch ein kleines Symbol rechts unter dem jeweiligen Objekt (vgl. Entwicklungsprozess in Abbildung 4). Des Weiteren können Objekte als sogenannte Ausprägungskopien in ARIS in mehreren Modellen verwendet werden, d.h. es handelt sich dabei um graphische Symbole, die alle dasselbe bereits existierende Objekt referenzieren. Somit lassen sich bereits vorhandene Modelle (bspw. Geschäftsprozessmodelle) über Referenzen integrieren. Die grundlegende Idee unseres Ansatzes besteht nun darin, mit Hilfe von Hinterlegungen, Ausprägungskopien und zusätzlichen Symbolen, einen neuen Modelltyp in ARIS zu spezifizieren, durch den die zuvor beschriebene Komplexität reduziert wird. Ein solches Modell wird als *GlobalView* bezeichnet und im Folgenden detailliert vorgestellt.
Das grundlegende Diagramm *GlobalView* ist abgeleitet vom ARIS Modelltyp „EPK“. In-

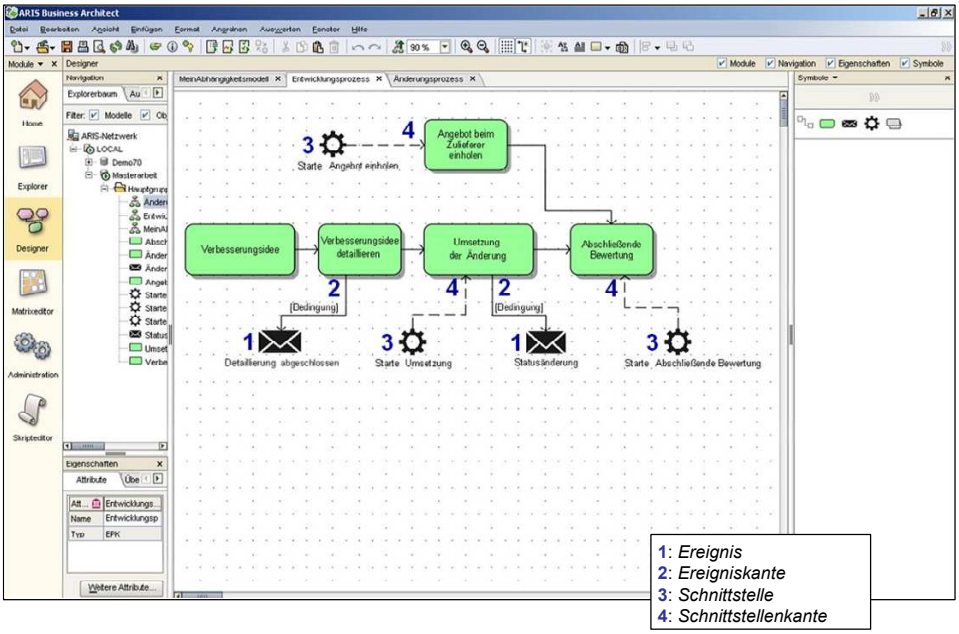

Abbildung 3: Vereinfachtes Prozessmodell in ARIS, erweitert um Schnittstellen und Ereignisse

nerhalb der *GlobalView* werden Filter und Vorlagen definiert, welche die Sichtweise auf das Modell für Modellierer soweit einschränken, dass nur noch die hier relevanten Objekttypen angeboten werden. Für die Modellierung von Aufrufbeziehungen werden zusätzliche Typen (für Objekte und Kanten) benötigt. Da in ARIS keine benutzerdefinierten Objekttypen realisiert werden können, sind diese von ARIS Standard-Objekten abzuleiten. Neue Objekttypen für Ereignisse und Schnittstellen werden deshalb vom Objekttyp „Function“ abgeleitet. Der Übersichtlichkeit halber, wird jeweils ein gesamter Prozess als ein gekerbter Pfeil dargestellt (vgl. Entwicklungsprozess in Abbildung 4). Dieser besitzt eine Hinterlegung, die auf das detaillierte Geschäftsprozessmodell (vgl. Abbildung 3) verweist, so dass dieses vom Modellierer bei Bedarf im Detail betrachtet werden kann. Ereignisse und Schnittstellen hinterlegter Prozesse werden an das abstrakte Prozessobjekt als Ausprägungskopie des jeweiligen Originalobjektes angehängt. Alt-Applikationen wie etwa das Logging System in Abbildung 4 können in dieser *GlobalView* explizit modelliert werden. Ereignisse und Schnittstellen solcher Systeme werden erstmalig in dieser aggregierten Sichtweise definiert.

Da nun alle Geschäftsprozesse, Applikationen, Ereignisse und Schnittstellen in der *GlobalView* enthalten sind, können Aufrufbeziehungen definiert werden. Diese gerichteten Kanten (dick gekennzeichnete Kanten in Abbildung 4) verbinden Ereignisse mit Schnittstellen und legen somit Aufrufbeziehung fest. Letztere haben einen Namen und besitzen optional eine Beschreibung.

Basierend auf der *GlobalView* können detaillierte Sichten, z.B. für eine bestimmte Gruppe von Applikationen, mittels ARIS-Reports generiert werden. Das resultierende Modell

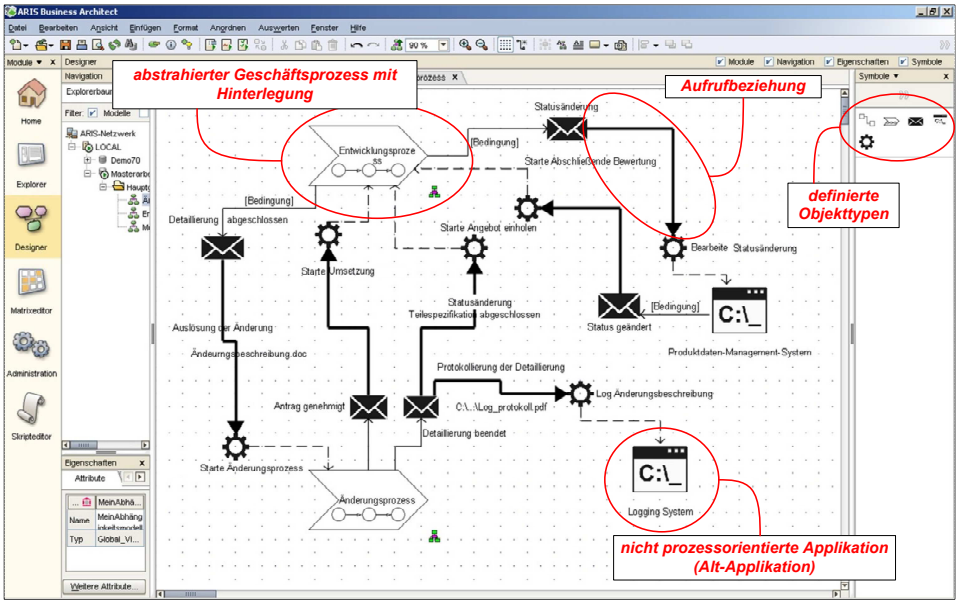

Abbildung 4: Aggregierte Sichtweise auf eine IT-Landschaft: *GlobalView*

enthält dann wieder die „expandierte“ Sicht (vgl. Abbildung 3) auf die Detailprozesse sowie die Abhängigkeiten zwischen bestimmten Ereignisse und Schnittstellen. Basierend auf dieser Information lassen sich Analysen zur Ermittlung von Optimierungspotentialen sowie zur Fehlererkennung durchführen. Zudem kann bei zyklischen Abhängigkeiten zwischen Applikationen analysiert werden, ob diese tatsächlich eine problematische Verklemmung darstellen.

4 Definition der technischen Sicht

Mit dem bisher vorgestellten Ansatz ist es möglich, Aufrufbeziehungen in einer abstrakten und fachlichen Sicht zu definieren. Dabei werden lediglich die Beziehungen zwischen Ereignissen und Schnittstellen dokumentiert. Technische Details die zur konkreten Implementierung von Aufrufbeziehungen notwendig sind, werden dabei nicht modelliert.
Zur Kommunikation zwischen unterschiedlichen Applikationen und Geschäftsprozessen muss ein Datenaustauschformat festegelegt werden. Da sich das Datenformat der Quellapplikation i.A. von dem der Zielapplikation unterscheidet, ist eine entsprechende Konvertierung der Daten notwendig. Dies wird in der technischen Sicht (IT-Sicht) dokumentiert. Neben der Konvertierung werden hier weitere Elemente definiert:

- Datentypen und Datenstrukturen (z.B. als XSD) zum Austausch von Daten zwischen Quell- und Zielsystem,

- konkrete Datentransformation zwischen Quell- und Zielsystemformaten (z.B. als XSLT) und
- IDs (z.B. IP-Adressen) und physikalische Endpunktinformationen

Analog zur fachlichen Sicht muss für die technische Sicht eine geeignete Modellierungsmethodik entwickelt werden. Modellierer verfügen meist nicht über die erforderliche IT-Kompetenz, um technische Details zu spezifizieren. Deshalb wird eine neue Rolle für die Modellierung der technischen Sicht benötigt: IT-Architekten verwenden typischerweise CASE-Tools mit UML-Unterstützung und keine eEPKs zur Datenmodellierung und Softwareentwicklung. Wir verwenden deshalb UML-Kommunikationsdiagramme zur technischen Beschreibung von Aufrufbeziehungen und Objektdiagramme zur Verfeinerung der Detailinformation (vgl. Abbildung 6). Eine technisch detaillierte Aufrufbeziehung bezeichnen wir im weiteren als *Interaktion*.
Die Datenübernahme aus der fachlichen Sicht in die IT-Sicht findet über ein standardisiertes Modellaustauschformat statt (bspw. XMI [Obj07a]). Über einen Extraktionsmechanismus wird die Information aus der fachlichen Sicht (GlobalView) exportiert und dient der IT-Sicht als Grundlage zur technischen Detaillierung. Da ARIS-Modelle (EPK) nicht direkt im XMI-Format abgespeichert werden können, ist ein entsprechender Work-around notwendig, um die Modellinformation weiterverwenden zu können. Die erste Möglichkeit besteht darin, über einen Standard XML-Export die Modellinformation in eine XML-Datei zu schreiben (vgl. Abbildung 5). Diese Datei repräsentiert dann das komplette Abhängigkeitsmodell inkl. zusätzlicher Informationen wie bspw. Name der Quelldatenbank, Erstellungsdatum, Name des Benutzers der den Export initiiert hat, Objektpositionierung und vieles mehr. Interessant für die technische Sicht sind lediglich die Informationen, die für die Darstellung von Interaktionsdiagrammen notwendig sind. Hierzu gehören bspw. Objekt- und Attributdefinitionen, Kantentypen und Kantendefinitionen, Objekt-, Kanten- und Attributausprägungen, usw.
Die Modellinformation aus der fachlichen Sicht liegt nun als XML-Datei vor und kann in das XMI-Format konvertiert werden. Die Konvertierung wird entweder in Form einer XSL Transformation (XSLT) oder mittels eines „Parsers“ (bspw. Java-Applikation) realisiert. Beim Einsatz einer Transformationssprache wie XSLT, ist zunächst das in ARIS modellierte Abhängigkeitsmodell in eine XML-Repräsentation dieses Modells exportiert werden, um anschließend die eigentliche Konvertierung mittels XSLT durchzuführen. Anderenfalls kann zur Konvertierung eine beliebige Applikation (bspw. Java) implementiert werden, welche die aus ARIS exportierte XML-Repräsentation einliest, ins entsprechende Format transformiert und als XMI-Datei speichert. Ein weiterer Weg um Informationen aus Abhängigkeitsmodellen zu exportieren führt über die Verwendung von ARIS-Reports und -Makros. Die Idee dabei ist, ARIS-interne Skripte zu implementieren die Modellinformationen aus der ARIS-Datenbank auslesen, aufbereiten und direkt als XMI-Datei speichern. In Abbildung 5 werden die zuvor beschriebenen Ansätze zur Extraktion von Modellinformationen aus der fachlichen Sicht illustriert.
Basierend auf dieser Information können nun UML-Diagramme erzeugt werden, die dem IT-Architekten als Grundlage für die technische Detaillierung und spätere Implementierung der Aufrufbeziehungen dienen. Dazu haben wir die exportierten Daten aus der fachlichen Sicht in sofern erweitert, dass für jedes Ereignis, jede Schnittstelle und insbesondere für die Datentransformationen dazwischen ein Objekt im zugehörigen UML-Diagramm

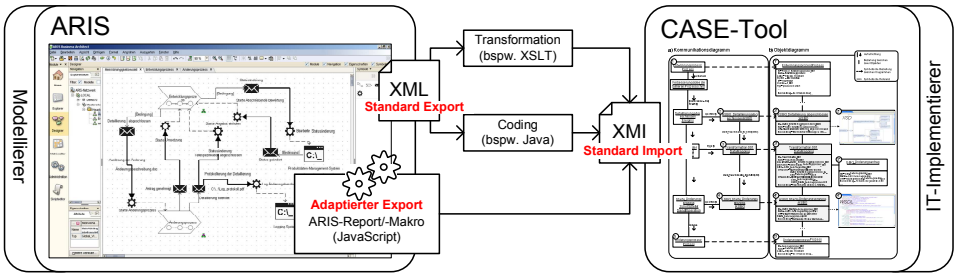

Abbildung 5: Export- und Import von Modellinformationen

angelegt wird. Dadurch werden beim Importieren in ein CASE-Tool bereits „leere“ Objekte im UML-Objekt- bzw. UML-Kommunikationsdiagramm angelegt. Diese dienen als Platzhalter und beschreiben zusätzliche Objekte, die durch den IT-Architekten verfeinert werden müssen.

Das Objekt- und das Kommunikationsdiagramm in Abbildung 6 zeigt die (importierte) Modellinformation aus der fachlichen Sicht, sowie zusätzlich generierte Objekte (grau hinterlegt). Diese ursprünglich leeren Objekte wurden bereits befüllt (②'③'④' in Abbildung 6), so dass eine verfeinerte Interaktion zwischen zwei Systemen definiert wird. Das Modell beschreibt die erste Beziehung aus Abbildung 2. Dabei wird ein Ereignis durch einen Prozessschritt in dem Entwicklungsprozess ausgelöst, wodurch ein Prozess im Änderungs-Management-System gestartet wird. Das Änderungs-Management-System bietet hierfür den Dienst „Starte Änderungsprozess“ an. Die entsprechende Detaillierung der generierten Objekte aus dem Kommunikationsdiagramm (①②③④⑤) wird im Objektdiagramm (①'②'③'④'⑤') vorgenommen. Dabei werden die Objekte mit Informationen wie Name, Beschreibung oder Schnittstellenreferenz (②"③"④") angereichert. Um die Übersichtlichkeit zu erhalten, werden lediglich die Objekte ① bis ⑤ im Objektdiagramm detailliert. Die Datenübernahme aus der fachlichen Sicht erfolgt mittels einer Exportfunktionalität und die Datenergänzung in der IT-Sicht manuell durch einen IT-Architekten. Dabei entsteht das Problem, dass wenn sich in der fachlichen Sicht nachträglich was ändert, nachdem in der IT-Sicht das zugehörige Objekt schon verfeinert wurde, die Änderung explizit nachgeflegt werden muss. Zweitens müssen Fehler, die in der IT-Sicht entdeckt und behoben werden, in der fachlichen Sicht entsprechend nachdokumentiert werden. Dies lässt sich kaum vermeiden, da unterschiedliche Werkzeuge für die einzelnen Rollen (Modellierer und IT-Architekten) eingesetzt werden. Durch die getrennte Datenbasis existieren zudem redundante Datenbestände, die konsistent gehalten werden müssen. Dieses generelle Problem ist vergleichbar mit der heute üblichen Trennung zwischen Geschäftsprozess- und Workflow-Modellierung, Applikationsimplementierung und Prozess-Monitoring.

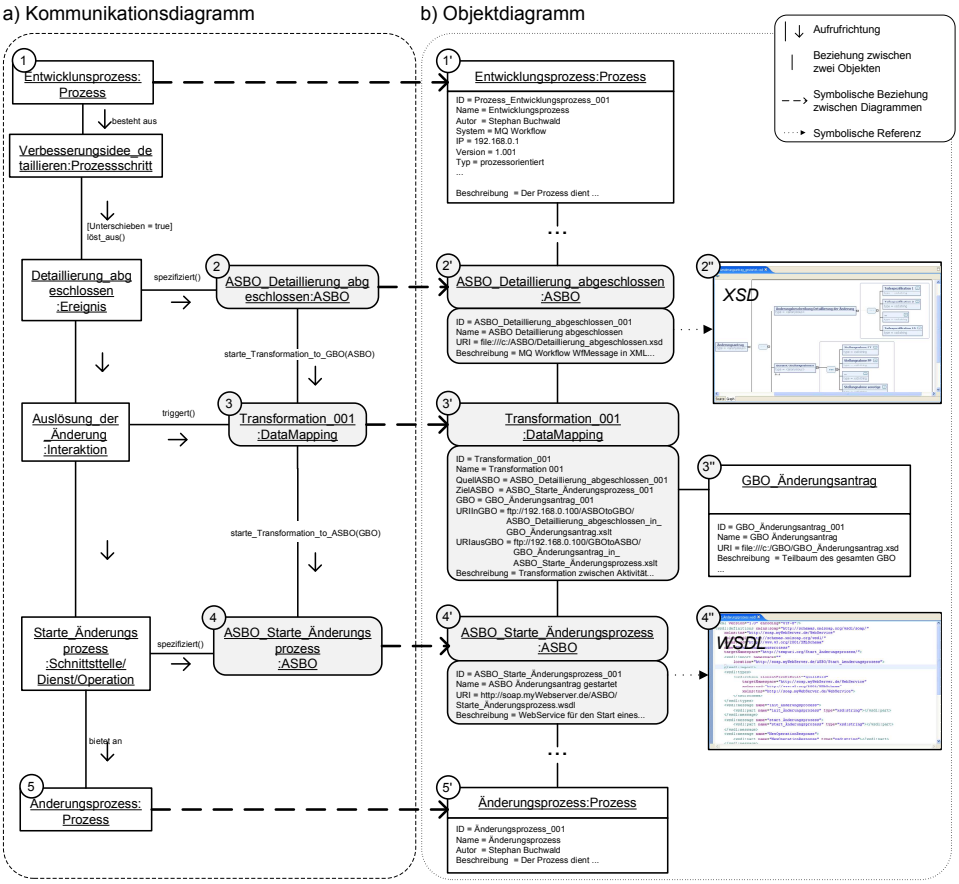

Abbildung 6: Verfeinerung einer Interaktion zwischen zwei Systemen

5 Nutzung der Modellinformation zur Ausführungszeit

Die vorgestellte Modellierungsmethodik ermöglicht die Dokumentation von Aufrufbeziehungen, um Optimierungspotentiale für Systeminteraktionen zu identifizieren und Änderungen an der IT-Landschaft besser abstimmen zu können. Der logisch nächste Schritt ist nun, das in der IT-Sicht detaillierte Modell als Basis für die Implementierung der Beziehungen zwischen Systemen zu verwenden. Die Integration neuer Applikationen kann dann direkt modellbasiert realisiert werden. Die modellierten Beziehungen können dann direkt als Spezifikation für die Implementierung der System-Interaktionen dienen. Hierfür existieren mehrere Möglichkeiten. Ein konventioneller Ansatz besteht darin, z.B. mittels SOAP-Kommunikation und in einem Application Server realisierte Funktionalität, Dienste entfernter Applikationen aufzurufen. Etwas moderner sind Ansätze, bei denen diese Aufgabe durch Nachrichtenflüsse in Message-Brokern realisiert wird. In beiden Fällen empfängt eine solche Komponente ein Ereignis und leitet es an die jeweils zuständige

Funktion bzw. den entsprechenden Nachrichtenfluss weiter. Die Interaktionskomponente realisiert dies als Programm-Code (Java, C++, etc.) oder Messge-Broker-Ablauf (z.B. ESQL-Knoten im IBM WebSphere Message Broker [IBM07]). Informationen über Quellapplikation werden zuvor ggf. einer Datentransformation unterzogen und Funktionen der entsprechenden Zielapplikationen werden aufgerufen.
Zudem existieren Ansätze die alle Systeminteraktionen durch eine generische Integrationskomponente realisieren [Bau05]. Die Interaktionsmodelle (vgl. Abbildung 6) werden dann automatisch transformiert und das Ergebnis wird in eine Steuerungsdatenbank der Komponente abgelegt. Diese verwendet die abgeleitete Information dann zur Durchführung von Applikationsinteraktionen. Da Interaktionen zwischen Applikationen stets denselben grundlegenden Ablauf aufweisen, kann eine solche Komponente generisch realisiert werden (vgl. [Buc05]):

(1) Die Komponente empfängt alle relevanten Ereignisse,
(2) nutzt die Information aus der Steuerungsdatenbank, um Zielsysteme und die notwendigen Datentransformationen zu ermitteln,
(3) führt die Transformationen durch und
(4) ruft die entsprechenden Schnittstellen der Zielsysteme.

Die mit unserem Ansatz definierten Modelle können eine solche Interaktionskomponente konfigurieren, so dass keine explizite Implementierung mehr notwendig ist. Der einmalig anfallende Aufwand für die Implementierung der Interaktionskomponente ist jedoch hoch, vor allem wenn diese unterschiedliche Kommunikationsprotokolle bedienen soll.
Die in Abschnitt 3 und 4 vorgestellten Modelle enthalten alle benötigten Informationen, unabhängig davon, welcher der beschriebenen Ansätze verfolgt wird. Deshalb bilden sie die Basis für eine Generierung von Programmrümpfen für Java-Implementierungen, von Message-Broker-Abläufen und von Informationen einer Steuerungsdatenbank für eine generische Integrationskomponente.

6 Stand der Technik

Die Integration von IT-Systemen wird vielfach durch eine direkte Verbindung zwischen den einzelnen Systemen realisiert. Die eigentliche Kommunikation und der Datenaustausch ist dabei oftmals im Programm-Code fest implementiert. Da die Applikationen selbst meist keine geeigneten API-Funktionen anbieten (z.B. den Zugriff auf interne Funktionalität oder Datenbanken), müssen „Wrapper“ implementiert werden, die eine Kommunikation unterstützen. Dieser Ansatz ist aufgrund gewachsener IT-Landschaften stark verbreitet.
Message-Broker sind Komponenten, die den Informationsaustausch zwischen unterschiedlichen Systemen realisieren und zur zentralen Applikationsintegration eingesetzt werden können [CHKT05]. Die Hauptaufgabe eines Brokers besteht darin, Informationen (Nachrichten) von Applikationen entgegenzunehmen, diese entsprechend zu transformieren und an Zielapplikationen weiterzuleiten. Die Kommunikation zwischen Applikationen wird mittels eines (Message-)Brokers realisiert, der sich u.a. um die notwendige Konvertierung

der Nachrichten aus den unterschiedlichen Applikationen kümmert. Vorteil eines Broker-Ansatzes ist die Reduktion der Anzahl der Schnittstellen und die einfachere Erweiterung der IT-Landschaft, weil neue Interaktionen als Nachrichtenflüsse basierend auf existierenden Schnittstellen realisiert werden können. Diese Nachrichtenflüsse müssen dennoch explizit modelliert werden. Es existieren Standards für die Kommunikation [TS02] (Corba, RMI, DCOM, etc.) zwischen Applikationen, jedoch keine für deren explizite Modellierung.
Andererseits wird durch EAM oft nur auf abstrakter Ebene modelliert [MBL07, EHH+08, BELM08]. Details über die unterschiedlichen Abhängigkeiten (fachlich und technisch) zwischen Systemen und prozessorientierten Applikationen werden dagegen vernachlässigt. Es gibt keine Standards, mit denen die Modellierung von Abhängigkeiten so beschrieben werden kann, dass diese möglichst nah an den realen Systemen ist.
Wie die zuvor beschriebenen Ansätze zeigen, werden bei der Integration von Applikationen heterogene Technologien eingesetzt und es existieren keine allgemein akzeptierten Modellierungs-Standards. Zwar gibt es für die Modellierung von Abhängigkeiten im Allgemeinen viele Ansätze, die beschreiben wie diese definiert werden können, jedoch mit dem Schwerpunkt auf Daten und nicht auf der Modellierung von Applikationen und Prozessen. [Mei02] etwa stellt einen Abhängigkeitsmanager vor, welcher die Definition von Datenabhängigkeiten in heterogenen Informationssystem-Landschaften beschreibt. Dabei werden die Definition und Verwaltung von Abhängigkeiten, z.B. das Erzeugen neuer, das Modifizieren bestehender oder das Löschen vorhandener Abhängigkeiten beschrieben. Zur Speicherung der Abhängigkeitsinformation wird ein Repository eingeführt, welches unterstützende Information zur Interaktion zwischen den Systemen verwaltet. Eine graphische Modellierung der Abhängigkeiten wird nicht diskutiert. Des Weiteren existieren Ansätze, die mittels Web-Service- oder Enterprise-Service-Bus-Technologien [Mel05, Erl05] entsprechende Integrationslösungen in Service-orientierten Architekturen (SOA) technisch beschreiben. Dabei können bspw. Applikationen über abstrakte BPEL-Schnittstellen gekapselt und über Orchestrierung (vgl. [TF06]) in Beziehung gesetzt werden. Diese Ansätze bieten jedoch keine fachliche Sicht zur Modellierung von Abhängigkeiten zwischen Applikationen.
[Fra02] beschreibt eine Unternehmensmodellierungsmethode, die insbesondere betriebswirtschaftliche und technische Konzepte beim Entwurf von Informationssystemen betrachtet. Diese Methode unterscheidet zwischen verschiedenen Sichtweisen auf das Unternehmen [Fra94]. Eine detaillierte Betrachtung einzelner Aufrufbeziehungen zwischen Applikationen und Geschäftsprozessen und insbesondere die entsprechende technische Detaillierung wird dabei nicht explizit betrachtet. Weitere Ansätze wie [TLD+06] verwenden Landscape Maps zur Beschreibung von abstrakten Beziehungen zwischen verschiedenen Domänen einer Unternehmenslandschaft. Der Schwerpunkt hierbei liegt darin, eine möglichst verständliche und vollständige Sichtweise für Manager, Geschäftsprozess- und Systemverantwortliche zu erhalten. Eine technische Detaillierung von Aufrufbeziehungen wir auch hier nicht betrachtet.

7 Zusammenfassung und Ausblick

Dieser Beitrag beschreibt eine erweiterte Methodik zur Modellierung von Abhängigkeiten zwischen Applikationen und Geschäftsprozessen - so genannten Aufrufbeziehungen. Insbesondere diese werden meist unzureichend dokumentiert, da für die rasante Entwicklung von IT-Landschaften in Unternehmen keine festgelegten Entwicklung- und Entwurfsprozesse existieren. Generell kann davon ausgegangen werden, dass die Entstehung einer IT-Landschaft nicht modellbasiert realisiert wird. Dies führt dazu, dass es keine graphische und formale Dokumentation über die Abhängigkeit zwischen verschiedenen Applikationen und Prozesse existiert.
Bei der Modellierung von Geschäftsprozessen ist es wichtig, Meta-Informationen über Prozessverantwortliche, Bearbeiter, Bearbeitungszeiten oder Kosten einzelner Prozessschritte festzuhalten. Analog dazu ist die Dokumentation von Abhängigkeiten zwischen Systemen wichtig, um z.B. Gültigkeitszeiträume für Beziehungen, Versionsnummern eingebundener Transformationsfunktionen zwischen Applikationen oder Verantwortliche für bestimmte Beziehungen zu dokumentieren. Deshalb sollte zum einen die IT-Landschaft und zum anderen die Abhängigkeiten zwischen den Applikationen und Geschäftsprozessen modelliert werden. Der in dieser Arbeit vorgestellte Ansatz zur Modellierung von Aufrufbeziehungen behandelt beide Aspekte. Die Modellierung erfolgt durch zwei unterschiedliche Rollen „Prozess-Modellierer“ und „IT-Implementierer“, die sich in unterschiedlichen Sichten (fachliche Sicht und technische Sicht) bewegen und Abhängigkeiten, Ereignisse, Schnittstellen sowie die entsprechenden Datentransformationen zwischen Systemen modellieren.
Die graphischen Modelle der fachlichen Sicht dokumentieren eine abstrakte Sicht auf das Zusammenspiel einzelner Applikationen und Geschäftsprozesse. Dadurch entsteht eine übersichtliche Darstellung der unterschiedlichen Interaktionen. Interessant ist die Information darüber, welche Abhängigkeiten zu welchen Applikationen existieren und unter welchen Bedingungen diese aktiv werden. Auf Basis dieser Information kann geprüft werden, welche Applikationen z.B. bei Änderungen an Geschäftsprozessen oder Applikationen beeinträchtigt werden und welche Personen im Unternehmen deshalb zustimmen müssen. Zudem liefert die modellierte Information über die IT-Landschaft eine transparente Grundlage für Optimierungen.
Auch die IT-Sicht ist modellbasiert dokumentiert und spezifiziert technische Detailinformationen. Diese können als Grundlage für die Implementierung einer Steuerungskomponente verwendet werden. Die Modelle definieren Aufrufbeziehungen zu Diensten entfernter Applikationen. Damit stellt die Methodik einen Baustein in einer SOA dar. Allerdings werden in diesem Beitrag Management-Aspekte einer SOA wie Governance-Prozesse oder -Gremien nicht betrachtet. Die vorgestellte Methodik bietet jedoch eine Basis, um entsprechenden Gremien die benötigten Informationen bereitzustellen.
Wünschenswert wäre eine Absicherung des Konzeptes durch eine umfangreiche Fallstudie. Dabei sollte die entsprechende Domäne zwar überschaubar, aber auch nicht zu klein sein. Es muss eine ausreichende Anzahl an Applikationen vorhanden sein, zwischen denen so viele Abhängigkeiten existieren, dass eine aussagekräftige Bewertung möglich wird. Als Ergebnis werden dann Aussagen über den Nutzen des Ansatzes zur Erhöhung der Transparenz bzgl. Abhängigkeiten, als Planungs- und Entscheidungsgrundlage für Gremi-

en, sowie als Implementierungsgrundlage entsprechender Abhängigkeiten denkbar. Voraussetzung für eine solche Fallstudie ist allerdings die Verfügbarkeit entsprechender Konzepte in einem Geschäftsprozessmanagement-Tool wie z.B. ARIS und in einem CASE-Tool wie etwa Rational Software Architect [BB03]. Da solche Werkzeuge heutzutage diese Funktionalität nicht bieten, kann eine Fallstudie alternativ auf Basis einer prototypischen Implementierung durchgeführt werden. Hierbei ist die für die Modellierung der Abhängigkeiten getätigte Investition allerdings nicht langfristig geschützt, weil Prototypen meist keine dauerhafte Lösung darstellen. Dazu kommt, dass Funktionalität und Bedienkomfort von Prototypen für einen größeren Pilotbetrieb eigentlich nicht ausreichend sind. Deshalb sollten solche Konzepte mittelfristig in kommerzielle Modellierungswerkzeuge aufgenommen werden.
Die fachliche Sicht auf Geschäftsprozesse, deren technische Umsetzung sowie die Beziehung zwischen diesen beiden Modellierungsebenen ist ein extrem wichiger, aber bisher nicht ausreichend betrachteter Aspekt. Deshalb betrachten wir dieses Thema im Projekt Enproso (zusätzlich zu den hier vorgestellten) auch unter dem Aspekt der Nachvollziehbarkeit von erforderlichen Prozess-Umstrukturierungen [BBR09, BBR10]. Nur wenn auch diese gegeben ist, lässt sich das im SOA-Umfeld häufig erwähnte Business-IT-Alignment [Che08] tatsächlich erreichen.

Literatur

[AAA+07] A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, A. Guízar, N. Kartha, C. K. Liu, R. Khalaf, Dieter Koenig, M. Marin, V. Mehta, S. Thatte, D. Rijn, P. Yendluri und A. Yiu. *Web Services Business Process Execution Language Version 2.0 (OASIS Standard)*, 2007.

[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.

[Bau05] T. Bauer. Integration von prozessorientierten Anwendungen. *In: Proc. Workshop on Enterprise Application Integration*, 2005.

[BB03] M. Boggs und W. Boggs. *UML and Rational Rose*. Mitp-Verlag, 2003.

[BBR09] S. Buchwald, T. Bauer und M. Reichert. Durchgängige Modellierung von Geschäftsprozessen durch Einführung eines Abbildungsmodells: Ansätze, Konzepte, Notationen. Bericht, Universität Ulm. Fakultät für Ingenieurwissenschaften und Informatik, April 2009.

[BBR10] S. Buchwald, T. Bauer und M. Reichert. Durchgängige Modellierung von Geschäftsprozessen in einer Service-orientierten Architektur. In *Modellierung'10*, Lecture Notes in Informatics (LNI). Koellen-Verlag, March 2010.

[BELM08] Sabine Buckl, Alexander M. Ernst, Josef Lankes und Prof. Dr. Florian Matthes. Enterprise Architecture Management Pattern Catalog Release 1.0, 2008.

[BPM06] BPMI. Business Process Management Initiative, Business Process Modeling Notation Specification (BPMN), 2006.

[Buc05] S. Buchwald. Konzeption und Realisierung einer Infrastruktur zur prozessorientierten Integration von Applikationen in Workflows. Diplomarbeit, Fachhochschule Ulm, 2005.

[Buc07] S. Buchwald. Modellierung von Abhängigkeiten zwischen prozessorientierten Applikationen. Masterarbeit, Hochschule Ulm, 2007.

[Che08] H.-M. Chen. Towards Service Engineering: Service Orientation and Business-IT Alignment. *Hawaii International Conference on System Sciences*, Seite 114, 2008.

[CHKT05] S. Conrad, W. Hasselbring, A. Koschel und R. Tritsch. *Enterprise Application Integration: Grundlagen - Konzepte - Entwurfsmuster - Praxisbeispiele*. Spektrum Akademischer Verlag, 2005.

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

[Erl05] T. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall, 2005.

[Fra94] U. Frank. Multiperspektivische Unternehmensmodellierung: Theoretischer Hintergrund und Entwurf einer objektorientierten Entwicklungsumgebung, 1994.

[Fra02] U. Frank. Multi-perspective Enterprise Modeling (MEMO) - Conceptual Framework and Modeling Languages. In *HICSS*, Seite 72, 2002.

[HBR08] A. Hallerbach, T. Bauer und M. Reichert. Anforderungen an die Modellierung und Ausführung von Prozessvarianten. *Datenbank Spektrum*, 2008.

[IBM07] IBM. *IBM WebSphere Message Broker: ESQL Version 6*, 2007.

[IDS06] IDS Scheer AG. *ARIS Platform - Methode 7.0 Toolset Dokumentation*. IDS Scheer AG, 2006.

[LR00] F. Leymann und D. Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall, 2000.

[MBL07] D. Maurer, P. Büch und M. Linke. From Business Process to Enterprise Architecture, ARIS Solution for Enterprise Architecture Management, ARIS Expert Paper, 2007.

[Mei02] H. Meinecke. Entwurf und Implementierung eines Abhängigkeits-Managers. Diplomarbeit, Universität Stuttgart, 2002.

[Mel05] I. Melzer. *Service-orientierte Architekturen mit Web Services - Konzepte, Standards, Praxis*. Spektrum Akademischer Verlag, 2005.

[MM04] F. Manola und E. Miller. *RDF Primer*. World Wide Web Consortium, Februar 2004.

[Obj07a] Object Management Group. *MOF 2.0/XMI Mapping Specification, v2.1.1*, 2007.

[Obj07b] Object Management Group. *UML Superstructure Proposal v2.1.2*, 2007.

[Pet62] C. A. Petri. *Kommunikation mit Automaten*. Dissertation, Institut für instrumentelle Mathematik, Bonn, 1962.

[TF06] S. Ross- Talbot und T. Fletcher. Web Services Choreography Description Language: Primer. World Wide Web Consortium, Working Draft, 2006.

[TLD+06] L. W. N. van der Torre, M. M. Lankhorst, H. W. L. ter Doest, J. T. P. Campschroer und F. Arbab. Landscape Maps for Enterprise Architectures. In Eric Dubois und Klaus Pohl, Hrsg., *Advanced Information Systems Engineering, 18th International Conference, CAiSE*, Jgg. 4001 of *Lecture Notes in Computer Science*, Seiten 351–366. Springer, 2006.

[TS02] A. S. Tanenbaum und M. van Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002.