Anforderungen an ein Metamodell für SOA-Repositories

Stephan Buchwald[1], Julian Tiedeken[1,2], Thomas Bauer[1], Manfred Reichert[2]

[1]Abteilung für Daten- und Prozessmanagement, Daimler AG,
{stephan.buchwald, julian.tiedeken, thomas.tb.bauer}@daimler.com
[2]Institut für Datenbanken und Informationssysteme, Universität Ulm
{manfred.reichert, julian.tiedeken}@uni-ulm.de

Abstract: Service-orientierte Architekturen (SOA) gewinnen in Unternehmen zunehmend an Bedeutung. Insbesondere die lose Kopplung von Services verspricht mehr Flexibilität. Durch die Vielzahl an Services und Prozessen in unterschiedlichen Varianten sowie deren gleichzeitige Verwendung durch Service-Nutzer, entstehen hohe Kosten für Betrieb und Wartung. Services, die nicht mehr genutzt werden, sollten deshalb zeitnah „abgeschaltet" werden. Um solche nicht mehr benötigten Services identifizieren zu können, muss u.a. bekannt sein, welche Services aktuell von wem benutzt werden. Zudem entstehen durch unterschiedliche Versionen von Services komplexe Abhängigkeiten, die durch eine geeignete Informationsverwaltung im SOA-Repository beherrscht werden müssen. Dieser Beitrag stellt die in diesem Zusammenhang bestehenden Anforderungen an ein Metamodell dar.

1 Motivation

Service-Orientierung ist ein wichtiges Architekturprinzip für Unternehmen. Die Ziele einer Service-orientierten Architektur (SOA) [1, 7, 9] sind vielschichtig. Sie reichen von einer Erhöhung der Flexibilität durch Adaptierbarkeit der Geschäftsprozesse an geänderte Rahmenbedingungen [5, 13] bis hin zu kürzeren Entwicklungszeiten sowie reduzierten Kosten für Service- und Prozess-orientierte Informationssysteme. Generell beschreibt eine SOA nicht nur ein technisches Paradigma. Vielmehr versucht sie das Zusammenspiel zwischen Fachbereichen und IT-Abteilungen zu verbessern, was in der Literatur auch als Business-IT-Alignment bezeichnet wird [3]. D.h. Informationssysteme sollen fachliche Anforderungen exakter treffen als bisher.

In einer SOA wird die Geschäftsprozessmodellierung durch fachliche Services und fachliche Datenobjekte unterstützt, deren Dokumentation aber abstrakt gehalten ist. Sie werden auf Implementierungsebene ggf. durch mehrere technische Services und Datenobjekte verfeinert. Das Zusammenspiel und die Beziehung zwischen fachlichen und technischen Services, Prozessen und Datenobjekten sind in der Praxis meist nicht ausreichend dokumentiert. Deshalb entstehen zahlreiche Probleme: So kann bei Außerbetriebnahme eines Services nicht immer nachvollzogen werden, welche Prozesse oder Applikationen davon betroffen sind. Dadurch ist es schwierig, sicherzustellen, dass die Abschaltung nicht zu unerwarteten Fehlern führt. Zusätzliche Komplexität entsteht dadurch, dass sowohl fachliche als auch technische Services (und Datenobjekte) in unterschiedlichen Versionen existieren. Um die Metainformation zu all die-

sen Objekten sowie relevante Beziehungen zwischen ihnen beherrschen zu können, ist ein Repository notwendig, welches die Daten speichert. Durch deren logisch zentrale Verwaltung wird mehr Transparenz zu Objektabhängigkeiten geschaffen, was Inkonsistenzen nach Änderungen an Objekten (z.B. fachliche und technische Services oder Datenobjekte) erkennbar macht sowie eine Reaktion darauf zulässt. Außerdem muss die Entwicklung neuer Applikationen dahingehend unterstützt werden, dass relevante Informationen in jeder Phase des Entwicklungsprozesses durch das SOA-Repository bereitgestellt werden. Um möglichst schnell von fachlichen Anforderungen zu deren IT-Implementierung zu gelangen, ist eine durchgängige Modellierungsmethodik [2] notwendig, die ebenfalls durch das SOA-Repository unterstützt werden muss.

Dieser Beitrag stellt erstmalig und vollständig Anforderungen an das Metamodell eines zentralen SOA-Repositories aus Praxissicht vor. Dies legt die Basis zur Entwicklung eines Gesamt-Metamodells für SOA-Repositories. Dazu betrachten wir in den Kapiteln 2 bis 4 Ziele einer SOA und leiten daraus Anforderungen an das Metamodell ab. Kapitel 5 diskutiert verwandte Arbeiten, bevor mit einer Zusammenfassung geschlossen wird.

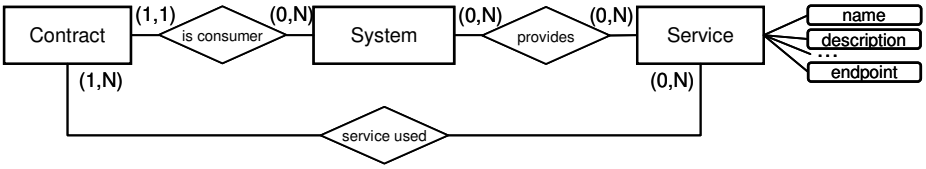
2 Nutzung angebotener Services

Eine zentrale Idee von SOA besteht darin, die Services verschiedener Anbieter einheitlich zu nutzen. Services abstrahieren dabei Funktionalitäten, die von anderen Applikationen verwendet werden können. Durch ihre Nutzung entsteht eine lose Kopplung zwischen Anbieter (engl.: *provider*) und Nutzer (*consumer*). Damit Service-Nutzer die richtigen Services finden und anschließend verwenden können, müssen Service-Informationen geeignet dokumentiert werden.

Anforderung 1: Service-Publikation und -Kontrakt

Bevor ein Service genutzt werden kann, sollte er in einem SOA-Repository publiziert werden. Anschließend kann ein potentieller Nutzer im Repository nach Services suchen, z.B. unter Verwendung von Suchkriterien wie angebotene Funktionalität oder verwendete Datenobjekte. Zur Ausführungszeit muss die physische Lokation eines Services mittels des Repositories ermittelbar sein, damit der Service-Aufruf durchgeführt werden kann. Der Einsatz eines Proxies (bspw. Enterprise Service Bus) ermöglicht die Entkopplung des Service-Aufrufs, d.h. Service-Nutzer rufen nicht direkt den konkreten Service, sondern einen Proxy-Service. Letzterer ermöglicht ein dynamisches Binden sowie eine Endpunktauswahl des tatsächlichen Services.

Für die Verwendung eines Services ist eine Nutzungsvereinbarung (*contract*) zwischen Anbieter und Nutzer notwendig. Diese klärt Ansprüche und Pflichten und beinhaltet Informationen zu Nutzungskosten, Antwortzeiten, Verfügbarkeit und Sicherheitsaspekten. Dem entsprechend sind die im folgenden ER-Diagramm in Min-Max-Notation skizzierten Objekte und Beziehungen im SOA-Repository zu verwalten (Attribute nur teilweise dargestellt):

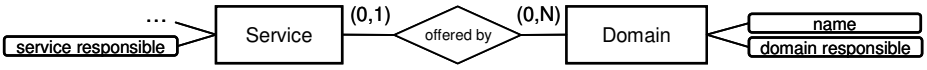


Ein *System* innerhalb eines Unternehmens kann sowohl ein Prozess oder eine Applikation (etwa J2EE [14]) sein, welches seine atomaren Funktionalitäten in Form von Services anbietet oder benötigte Funktionalität über Services konsumiert. Ein *Contract* ist dabei stets exakt einem *System* zugeordnet, wohingegen ein *System* beliebig viele *Contracts* hält.

Anforderung 2: Domänen zur Strukturierung

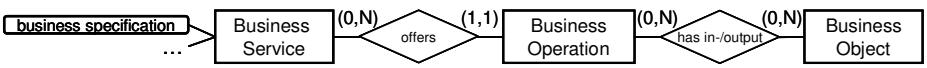
Die Untergliederung eines Unternehmens in Domänen auf organisatorischer Ebene dient u.a. dazu, einen Überblick über Funktionalität und Geschäftsobjekte verschiedener Fachbereiche zu erhalten [6]. Dabei werden unternehmensrelevante Funktionen sowie bestehende fachliche und technische Services der verschiedenen (Sub-) Domänen erfasst. Dadurch wird transparent, welche Funktionalität durch welche (Sub-) Domänen realisiert wird und wo Redundanzen bestehen.

Um Services auch Domänen-basiert suchen zu können, muss das SOA-Repository explizit die Beziehungen zwischen (Sub-)Domänen und angebotenen Services speichern. Damit sich Änderungen an Services effizient koordinieren lassen, werden Domänenverantwortliche im Repository verwaltet.



Anforderung 3: Informationserzeugung und -verwendung

Bei der Entwicklung eines Services sind neben unterschiedlichen Personengruppen verschiedene Werkzeuge im Einsatz. Diese dienen der fachlichen Beschreibung bzw. technischen Implementierung von Services. Sie erzeugen neue Informationen, verwenden aber auch existierende Dokumente aus dem SOA-Repository. Während der fachlichen Prozessmodellierung etwa werden neue fachliche Services erzeugt und bereits vorhandene weiterentwickelt. Diese verwenden Geschäftsobjekte, welche die Ein- und Ausgabeparameter der fachlichen Services darstellen. Die fachliche Service-Spezifikation (Fachspezifikation) bildet die Grundlage für die technische Service-Implementierung.



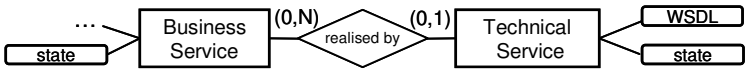
Analog wird eine technische Service-Spezifikation (WSDL) inklusive technischer Operationen und den verwendeten Parametern (Datenschemata z.B. als XSD) im Repository gespeichert. Als Attribute werden zu einem technischen Service ein Nutzungshandbuch und nach der Implementierung und Installation ein Endpunkt bereitgestellt.

3 Gewährleistung der langfristigen Stabilität einer SOA

Auch im Kontext von Änderungen der Service-Landschaft müssen Service-nutzende Applikationen zuverlässig funktionieren. Jede freigegebene Änderung an Repository-Objekten sollte deshalb wieder zu einer gültigen Service-Landschaft führen: Es dürfen keine Inkonsistenzen entstehen, etwa durch Abschalten eines Services, für den eine Nutzung noch vertraglich garantiert wird. Im Folgenden betrachten wir Anforderungen an das Repository-Metamodell, welche die Stabilität einer SOA unterstützen.

Anforderung 4: Service-Lifecycle-Management

Während der Entwicklung einer Prozessapplikation durchlaufen Services unterschiedliche Zustände. Sie beschreiben z.B., ob für einen Service bereits eine Fachspezifikation vorliegt oder ob er bereits realisiert bzw. freigegeben ist. Die einzelnen Zustände, die ein Service durchlaufen kann, werden in einem Service-Lebenszyklus (*Service Lifecycle*) dokumentiert und beschrieben [9]. Ein Wechsel des Service-Zustands erfolgt erst, wenn alle Vorraussetzungen an den Nachfolgezustand erfüllt sind, etwa das Vorliegen einer Fachspezifikation oder eines Entwicklungshandbuchs. Die Kontrolle und Überwachung der Zustandsübergänge wird durch Governance-Prozesse realisiert. Sie regeln unter anderem das Änderungsmanagement von Services und Prozessen, indem sie festlegen, von wem Änderungen zu genehmigen sind. Entscheidungsgrundlage dafür ist die im SOA-Repository gespeicherte Information.

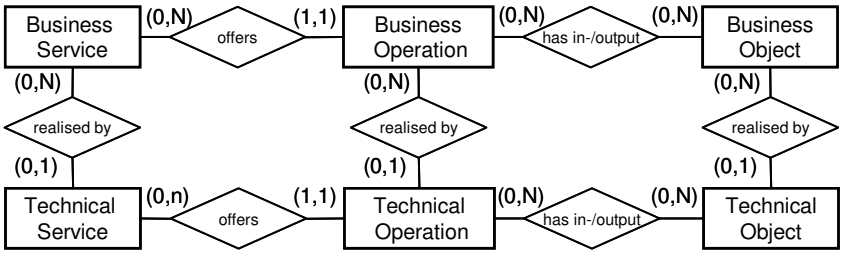


Zur Sicherstellung der Qualität von Services werden Compliance-Prüfungen während ihrer Entwicklung durchgeführt, die eine korrekte und kontrollierte Realisierung gewährleisten sollen. Die dazu notwendigen Dokumente, etwa Fachspezifikationen oder WSDL-Beschreibungen, sind im SOA-Repository abzulegen.

Anforderung 5: Speicherung der Beziehungen zwischen Repository-Objekten

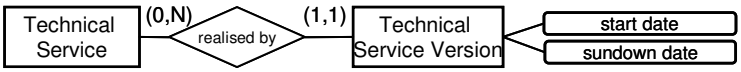
Fachliche und technische Artefakte, wie Service- oder Datenobjekte, stehen in direkter Beziehung zueinander: Ein fachlicher Service wird durch einen oder mehrere technische Services realisiert. Dabei verwendet eine Operation eines fachlichen Services Business-Objekte, die technisch auf ein oder mehrere Datenobjekte abgebildet werden. Um nach Änderungen an fachlichen Services, Operationen oder Business-

Objekten festzustellen, ob bzw. welche technischen Artefakte angepasst werden müssen, sollten die Beziehungen zwischen diesen Objekten im Repository explizit verwaltet werden:



Anforderung 6: Service-Versionen und Plantermine

Services werden in einer SOA kontinuierlich weiter entwickelt, wodurch neue Service-Versionen (sowohl fachlich als auch technisch) entstehen und veraltete „abgeschaltet" werden müssen. Durch die Vielzahl Service-konsumierender Applikationen ist eine Abschaltung jedoch nicht immer einfach durchzuführen, sondern macht für viele Applikationen eine aufwendige Anpassung erforderlich. Um Inkompatibilitäten zwischen Service-Anbieter und -Nutzer transparent zu machen, sollen neben den Abhängigkeitsbeziehungen auch Plantermine für die Abschaltung von Service-Versionen im Repository dokumentiert werden. Dadurch kann die Konsistenz überprüft und frühzeitig auf geplante Abschaltungen reagiert werden. Analog dazu können Plantermine eingeführt werden, welche die Nutzbarkeit einer neuen Service-Version anzeigen. Neue Service-Nutzer können dann z.B. prüfen, ob die neue Service-Version rechtzeitig zur Verfügung stehen wird oder ob eine ältere Version zu verwenden ist.

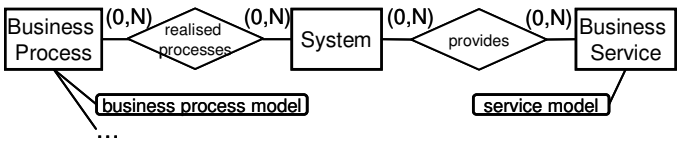


4 Permanente und durchgängige Speicherung von Modelldaten

Fachliche Prozesse und Services werden mit Werkzeugen (z.B. ARIS bzw. erweiterten Ereignisgesteuerten Prozess-Ketten) modelliert, wohingegen technische Prozesse häufig in CASE-Tools mit UML-Unterstützung dokumentiert werden. Infolge der unterschiedlichen Kenntnisse von Fach- und IT-Personal, werden fachliche Anforderungen oft unzureichend umgesetzt und es entsteht eine Lücke zwischen Fachbereich und IT-Abteilung. Deshalb ist es unabdingbar, die modellierte Information dauerhaft zu archivieren und diese Lücke zu schließen (Business-IT-Alignment [3]).

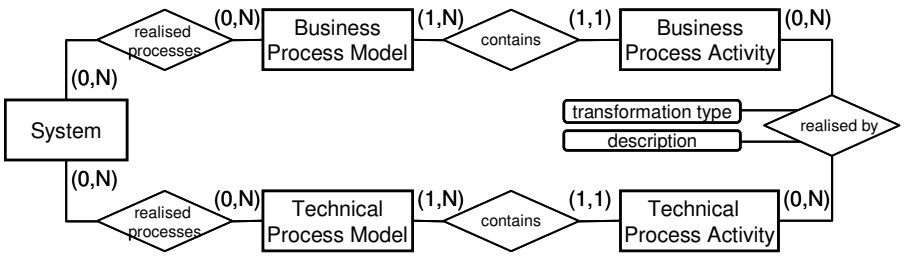
Anforderung 7: Langfristige Archivierung von Prozess- und Service-Modellen

Die langfristige Dokumentation von fachlichen und technischen Services sowie von Prozessen auf den Ebenen Fachmodell und technischem Modell ist essentiell. Hierdurch kann später nachvollzogen werden, welche konkreten Anforderungen und Fachprozesse durch eine Applikation bereits realisiert sind. Diese Information ist die Ausgangsbasis für eine Überarbeitung der Applikation (Application-Reengineering), nachdem sich Anforderungen geändert haben.



Anforderung 8: Beziehungen zwischen fachlichen und technischen Aktivitäten

Werden im SOA-Repository einzelne Aktivitäten der fachlichen und technischen Prozessmodelle und deren Beziehungen verwaltet, erhöht sich die Nachvollziehbarkeit, da Beziehungsrelationen zwischen fachlichen Anforderungen, fachlichen Services und ihrer Implementierung dokumentiert sind. Hierbei muss für jede Aktivität des fachlichen Modells (und damit für ihre Eigenschaften und Anforderungen) dokumentiert werden, durch welche Aktivität des technischen Modells diese realisiert wird [2]. Dadurch ist bei einer späteren Änderung fachlicher Aktivitäten leicht erkennbar, welche technischen Aktivitäten bzw. Services angepasst werden müssen. Zudem ist die Speicherung des Prozessmodells (*Business Process Model*) möglich, auch wenn die eigentliche Systemauswahl noch nicht abgeschlossen ist.



5 Verwandte Arbeiten

Heutige Repositories legen ihren Schwerpunkt entweder auf die Speicherung und Verwaltung technischer Artefakte (z.B. UDDI [12], WSRR [4]) oder Software- Komponenten [10, 11]. Eingeschränkt betrachtet werden dabei die Möglichkeiten zur kon-

sistenten und durchgängigen Modellierung sowie Dokumentation von fachlichen und technischen Services. So ist etwa die ARIS-interne Datenbank [8] in der Lage, fachliche und technische Artefakte zu verwalten. Die Beziehungen zwischen fachlichen und technischen Services werden im Regelfall aber nicht explizit verwaltet.

Einige existierende Repositories bieten Erweiterungsmöglichkeiten zur Realisierung eines umfassenden Metamodells. [7, 9] betonen die Notwendigkeit eines Repositories in einer SOA und fordern Funktionalitäten, wie die Bereitstellung von Service-Endpunkten für einen Enterprise Service Bus (ESB) oder die Möglichkeit für das Suchen und Auffinden von Services. Nicht betracht wird, welche konkreten Anforderungen an ein SOA-Repository existieren, d.h. welche konkreten Objekte in einem SOA-Repository verwaltet werden sollen und wie diese untereinander in Beziehung stehen.

6 Zusammenfassung und Ausblick

In diesem Beitrag haben wir wichtige Anforderungen an ein SOA-Repository diskutiert, die für eine konsistente Modellierung, Dokumentation, Verwaltung und Speicherung von fachlichen und technischen Services unverzichtbar sind. Aus diesen Anforderungen werden wir im Projekt *Enhanced Process Management by Service Orientation (ENPROSO)* ein umfassendes und in sich konsistentes Gesamt-Metamodell für SOA-Repositories ableiten. Bei der Realisierung eines SOA-Repositories sollten aufgrund der mannigfaltigen wechselseitigen Abhängigkeiten zwischen Objekten ggf. nicht alle vorgestellten Anforderungen in der ersten Version des SOA-Repositories realisiert werden. Eine Umsetzung kann etwa durch eine relationale Datenbank oder durch eine Erweiterung vorhandener SOA-Repositories (bspw. WSRR [4]) realisiert werden.

Literatur

1. S. Buchwald, T. Bauer, R. Pryss: IT-Infrastrukturen für flexible, service-orientierte Anwendungen – Ein Rahmenwerk zur Bewertung; In. Proc. 13. GI-Fachtagung Datenbanksysteme in Business, Technologie und Web, S. 524-543, Münster; 2009
2. S. Buchwald, T. Bauer, M. Reichert: Durchgängige Modellierung von Geschäftsprozessen in einer Service-orientierten Architektur; In. Proc. GI-Fachtagung Modellierung'10, Klagenfurt, 2010 (forthcoming)
3. H.-M. Chen: Towards Service Engineering, Service Orientation and Business-IT Alignment; In. Proc. 41st Hawaii Inf. Conf. on System Sciences; 2008
4. C. Dudley et al.: WebSphere Service Registry and Repository Handbook; 2007
5. P. Dadam, M. Reichert: The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support. Springer, 2009
6. G. Engels et al.: Quasar Enterprise: Anwendungslandschaften serviceorientiert gestalten, dpunkt.verlag, 2008.
7. T. Erl: SOA: Concepts, Technology, and Design; Prentice Hall, 2005
8. IDS Scheer: ARIS SOA Architect - Geschäftsprozesse als Grundlage für Service-orientierte Architekturen; IDS Scheer, 2008
9. N. M. Josuttis: SOA in Practice; The Art of Distributed System Design. O'Reilly, 2007

10. B. Li et al.: Building Interoperable Software Components Repository Based on MMF, Grid and Cooperative Computing (GCC) Workshops, 2004
11. M.J. Morel, J. Faget: The REBOOT Environment. Proceedings of the 2nd International Workshop on Software Reusability Advances in Software, 1993
12. OASIS: Universal Description, Discovery, and Integration (UDDI), Version 3.0, 2002
13. M. Reichert, D. Stoll: Komposition, Choreograhpie und Orchestrierung von Web Services: Ein Überblick. EMISA Forum, Vol. 24, No. 2, pp. 21-32, 2004
14. J. Keogh. J2ee: The Complete Reference. Osborne/McGraw-Hill, Berkeley, CA, USA, 2002.