Realisierung fachlicher Services auf Basis von Android-Konzepten

Steffen Dienst, Stefan Kühne*

Betriebliche Informationssysteme, Universität Leipzig
Johannisgasse 26, 04103 Leipzig
{dienst, kuehne}@informatik.uni-leipzig.de

Abstract: Web Services sind eine verbreitete Technik zur Realisierung von serviceorientierten Architekturen (SOA). In dezentral organisierten Umgebungen behindert die Notwendigkeit statisch getypter Schnittstellen und die mangelnde Unterstützung fachlicher Intensionen durch technische Realisierungskonzepte die Herausbildung agiler SOA-Communities. In diesem Beitrag wird diskutiert, inwiefern die Implementierungskonzepte des rasant wachsenden Android-Ökosystems Ansatzpunkte für eine höhere Flexibilät und bessere Wiederverwendung von lose gekoppelten SOA-Umgebungen eröffnen und welche Gestaltungshinweise an etablierte Techniken (z. B. Web Services) sich hieraus ableiten lassen.

1 Einleitung

Das Paradigma der Serviceorientierung bzw. die serviceorientierte Architektur (SOA) repräsentiert einen Ansatz zur inner- und überbetrieblichen Integration von verteilten Anwendungssystemen. Der SOA-Begriff wurde 1996 von Gartner eingeführt, besitzt jedoch keine allgemeingültige Definition [ST07, S. 9]. OASIS betrachtet SOA als „paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains“ [MLM+06, S. 8]. Damit wird SOA als technologieunabhängiges Konzept definiert, dass sich in unterschiedlichen Bereichen und auf unterschiedlichen Abstraktionsebenen anwenden lässt.

Auf betriebswirtschaftlicher Ebene meint dies bspw. die Aufteilung und Organisation einer Dienstleistung in verschiedene Teilleistungen, die durch unterschiedliche Akteure erbracht werden können. Exemplarisch sei folgendes Szenario gegeben: Im Rahmen einer Instandhaltungsdienstleistung sind Aufträge für Wartungs-, Entstörungs- und Vorsorgemaßnahmen zur Gewährleistung der Verfügbarkeit einer technischen Anlage zu koordinieren. Hierbei werden Ersatzteil-, Werkzeug-, Hilfs- und Betriebsstofflieferanten sowie Servicetechniker einbezogen. Ein konkreter fachlicher Service in diesem Szenario ist bspw. die Abfrage von Verfügbarkeiten und Angeboten für bestimmte Leistungen zu bestimmten

*Dieser Beitrag entstand im Rahmen des BMBF-geförderten Verbundprojekts EUMONIS (Förderkennzeichen: 01IS10033K).

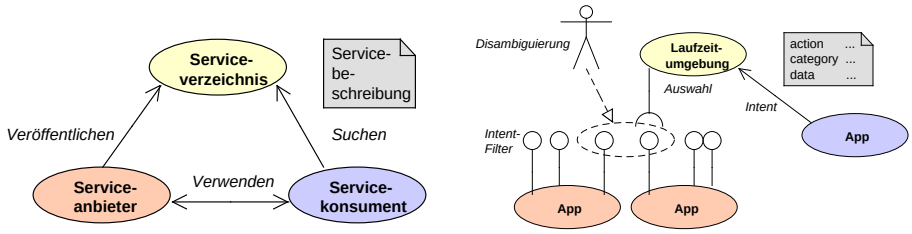
Terminen, wie z. B. die Lieferung eines Ersatzteils, die Bereitstellung eines Krans oder die Durchführung eines Ölwechsels, bei alternativen Anbietern.

Den verbreiteten technischen Realisierungskonzepten für SOA werden in diesem Beitrag die Konzepte der Android-Plattform gegenübergestellt. Android ist eine seit 2008 mit wenigen Anwendungen (Apps) gestartete Plattform und hat sich seitdem zu einem hochaktiven und rasant wachsenden Software-Ökosystem für mobile Anwendungen entwickelt. Allein der beobachtbare Anteil quelloffener Anwendungen umfasst derzeit 270 Mio. Quellcodezeilen. Täglich entstehen hunderte neue Apps; tausende werden aktualisiert [BPT+11].

Android ist auf den ersten Blick nicht als SOA zu erkennen. Die in Android unterstützten Aspekte hinsichtlich Wiederverwendbarkeit und flexibler Rekonfigurierbarkeit von lose gekoppelten Systemen sind jedoch kompatibel zu den SOA-Zielsetzungen. Vor dem Hintergrund des enormen Erfolgs des Ökosystems stellen sich somit die Fragen, inwiefern SOA-Konzepte in Android ausgeprägt und wie diese Ausprägungen mit etablierten SOA-Techniken (hier Web Services) zu vergleichen sind. Hieraus lassen sich Gestaltungshinweise hinsichtlich Erweiterungsbedarf bzw. Anwendungsrichtlinien an etablierte SOA-Techniken zur Herausbildung lebendiger SOA-Communities ableiten.

2 Ausprägung von SOA-Konzepten in Android

Die SOA-Konzepte Serviceanbieter, -konsument, -verzeichnis und -beschreibung sowie deren Zusammenspiel sind in Abbildung 1a dargestellt. Abbildung 1b zeigt deren konkrete Ausprägungen im Android-Kontext.



(a) SOA – Bestandteile

(b) SOA-Konzepte – Realisierungen in Android

Abbildung 1: SOA-Konzepte und deren Realisierung in Android

Applikationen (*Apps*) in Android bestehen aus vier Komponententypen (vgl. [BP10]): *Activity* (benutzerorientierte Funktionalitäten mit grafischer Repräsentation), *Service* (benutzerorientierte Funktionalitäten im Hintergrund)[1], *BroadcastReceiver* (Listener-Funktionalität, die auf verschiedene Ereignisse reagiert) und *ContentProvider* (tabellenorientierte

[1] Der Begriff *Service* soll im Weiteren im generischen Sinne verwendet werden, d. h. nicht speziell im Sinne der Android-spezifischen Komponente.

Datenspeicherung). Alle Komponenten kommunizieren ausschließlich mit den in diesem Kapitel beschriebenen Mechanismen miteinander.

Servicebeschreibung Services werden in Android durch drei Arten von Zeichenketten spezifiziert:

- *ACTION* ist ein „Verb“, das beschreibt, WAS getan werden soll. Dazu zählen sowohl vordefinierte Stringwerte im Android SDK mit fester Semantik (etwa VIEW oder EDIT), als auch frei definierbare Werte aus Drittquellen.
- *CATEGORY* Jedes *Intent* kann von konkreten Kontexten abhängig sein, etwa um Sprachausgaben nur zu aktivieren, wenn sich der Benutzer im Auto befindet. Spezifiziert werden diese Kontexte durch eine Liste von Strings.
- *DATA* besteht aus zwei Teilen: Einem Mimetype zur Beschreibung des Datentyps und einer URI, die wahlweise für die Auffindung der eigentlichen Daten dient oder diese Daten selbst enthält. Die URI enthält dabei entweder die Daten selbst oder einen Verweis auf diese.

Zur Beschreibung von bereitgestellten Services enthält jede *App* jeweils eine Manifest-Datei im XML-Format. Diese beschreibt alle Komponenten dieser *App* sowie weitere Metadaten. Komponenten können wahlweise extern verfügbar gemacht werden. Dazu werden *IntentFilter* definiert. *IntentFilter* beschreiben, auf welche *Intents* eine Komponente reagieren kann. Dazu können *Action*- und *Category*-*Werte* aufgelistet werden sowie die Art der kompatiblen Daten spezifiziert werden (etwa durch Auflistung von Mimetypes, URI-Schemabestandteilen wie Protokoll, Host etc.).

Serviceaufruf Der Aufruf von Services in Android erfolgt über die Bereitstellung von *Intents*. Dabei handelt es sich um Datenstrukturen, die einen Service adressieren. Es wird zwischen expliziten- und impliziten *Intents* unterschieden. Explizite *Intents* enthalten eine Angabe der exakten Implementierungsklasse, die angesprochen werden soll und werden im Allgemeinen verwendet, um Kontrollflüsse innerhalb von *Apps* zu realisieren. Implizite *Intents* hingegen adressieren beliebige Services. Dazu werden genau eine *Action*, optional mindestens eine *Category* und eine URI mit servicespezifisch enkodierten zusätzlichen Daten spezifiziert.

Servicerepository Zum Aufruf eines Services in Android ruft eine *App* eine entsprechende Methode auf und übergibt eine befüllte *Intent*-Struktur. Die Laufzeitumgebung gleicht dann die *IntentFilter* aller installierten *Apps* mit den Daten des übergebenen *Intent* ab. Gibt es genau einen passenden Service, wird dieser aufgerufen. Gibt es keinen wird dies dem Aufrufer durch eine entsprechende Fehlermeldung mitgeteilt und muss entsprechend behandelt werden. Stehen mehrere Services als Ziel eines *Intents* zur Auswahl wird im Allgemeinen der Benutzer durch einen Dialog aufgefordert zu entscheiden, welcher Service aufgerufen werden soll. Dieser kann dabei angeben, dass die getroffene Auswahl in Zukunft bei Konflikten bevorzugt werden soll.

Lose Kopplung Jeder Aufruf eines Services wird neu gegen die der Laufzeitumgebung bekannten *IntentFilter* geprüft. Es gibt keinen Mechanismus, der statische Konfigurationen von Services realisiert. Dieses Prinzip erlaubt eine sehr flexible Rekonfiguration der gesamten Funktionalität eines Androidsystems zur Laufzeit, bis hin zum Austausch von Kernkomponenten, etwa zur Ersetzung von Tastaturlayouts, der Oberfläche oder der Kontaktverwaltung.

Der Aufruf selbst kann synchron oder asynchron erfolgen. Für die Androidkomponente *Service* steht zusätzlich noch die Verwendung von *IBinder* zur Verfügung. Damit lassen sich direkte Zwei-Wege-Kommunikationen zwischen *Service*-Instanzen und *Service*-Konsumenten festlegen, welche zuvor initial über *Intents* etabliert wurden.

Die Android-Konzepte seien an folgendem Szenario illustriert: Eine *App* verlangt die Darstellung der aktuellen geographischen Position auf einer Karte. Diese „Absicht“ wird als *Intent* wie folgt formuliert: *ACTION*=„view“, *CATEGORY*=„default“, *DATA*= „geo://latitude,longitude?zoom=x“ bzw. *DATA*=„geo://0,0?q=adresse“ (vgl. Spezifikationsdraft unter [MS07]). Je nach Auswahl installierter *Apps* kann dieser *Intent* von unterschiedlichen Implementierungen etwa von Google Maps, OpenStreet Maps oder Bing Maps erfüllt werden. Ist mehr als eine passende *App* installiert, wird der Benutzer aufgefordert zu entscheiden, welche verwendet werden soll.

3 Vergleich von Realisierungsmöglichkeiten für fachliche Services

Die technischen Realisierungen (Verfeinerungen) von Services sind im Vergleich zu fachlichen Sichten oft deutlich ausführlicher und komplexer. Eine Umsetzung auf Basis von Web Services bedingt dies durch mehrere Faktoren. Neben der Ausführlichkeit der zugrunde liegenden XML-Strukturen in WDSL-Schnittstellenbeschreibungen und XML-Schemas zur Definition von Datenstrukturen enthalten Web-Service-Schnittstellen weitere Konstrukte, die für die technische Verarbeitung notwendig sind, jedoch das Verständnis und die Anpassbarkeit der Schnittstelle erschweren.

Das einführende Instandhaltungsszenario (vgl. Abschnitt 1) zeigt aus fachlicher Sicht, wie eine Anfrage an externe Dienstleister aussehen kann: Es wird angegeben, welche Art von Dienst in einem bestimmten Kontext mit einer Reihe von Qualitäten, Einschränkungen und Bedingungen erbracht werden kann. Aus fachlicher Sicht ist eine vorgegebene Syntax mit festen Operationen und Parametern für den Aufruf des Services nebensächlich. Interessant ist primär: WAS soll unter WELCHEN UMSTÄNDEN mit WELCHEN DATEN geschehen?

Servicebeschreibungen und -aufrufe mit Android-Konzepten erlauben hier eine direktere technische Repräsentation dieser Intention. Das Beispiel kann als Intent wie in Tabelle 1 beschrieben werden.

Damit könnten bspw. alle Instandhaltungsdienstleister gefunden werden, die in der Lage sind, eine Wartung an einer Offshore-Windenergieanlage vom Typ „WEA Gen 2“ in der 27. Kalenderwoche durchzuführen. Zusätzlich ist ein Verweis auf den für die Wartung relevanten Ausschnitt der Lebenslaufakte der betreffenden Windanlage als Datum angegeben. Ein derartiger Aufruf identifiziert alle Anbieter, die unter den unter *Category* aufgeführten

Action	org.eumonis.maintenance.wea
Category	offshore, KW27, WEA Gen 2, 500kW
Data	`eumonis://.../weagen2/...`

Tabelle 1: Exemplarischer Aufruf eines Services

Einschränkungen eine Wartung durchführen können. Diese können für weitere Interaktionen herangezogen werden.

Zur Abbildung des fachlichen Service im skizzierten Szenario auf Basis von Web Services sind verschiedene Service-Typen mit entsprechenden Operationen und Ein- und Ausgabeparametern erforderlich. Diese sind im Vorfeld durch die beteiligten Akteure unter Beachtung existierender Standards im Vorfeld festzulegen. Da die Menge der beteiligten Parteien jedoch schwer abzugrenzen ist (Peer-to-Peer-Szenario), ist dieser Abstimmungsprozess u. U. langwierig und aufwändig. Die Herausbildung einer lebendigen SOA-Community wird somit erschwert. Eine Serviceanfrage im Android-Stil hingegen kann durch beliebige weitere Details in Form von zusätzlichen Kontextelementen angereichert werden, ohne dass die Serviceanbieter ihre angebotene Serviceleistung anpassen müssen.

Die Spezifizierung derartiger Kontexte ist mit Serviceverzeichnissen für Web Services wie UDDI nicht möglich. Die praktische Einsetzbarkeit von UDDI selbst ist fraglich (etwa wegen der Existenz von Inkonsistenzen selbst in der aktuellsten Version 3.0.2 der Spezifikation [Col05]). Dokumentierte Ansätze zur Erweiterung, wie etwa in [LCLC08], beschäftigen sich zwar mit unterschiedliche Repräsentationsarten von QoS in UDDI, jedoch haben sich diese Ansätze nicht in der Praxis durchgesetzt.

Essentiell notwendig für eine servicebasierte Interaktion nach dem Vorbild von Android ist eine Ontologie von *Actions*, die es erlaubt, konkrete Services ohne weiteren Abstimmungsbedarf zwischen Serviceanbieter und -konsument anzubieten und nachzufragen.

Eine weiterer Unterschied liegt in der Komponierbarkeit von Services. Web Services lassen sich nur nach Kenntnis der konkreten Schnittstellen gemeinsam nutzen. Diese müssen für die Erstellung von zusammengesetzten Services (z. B. mit Hilfe von BPEL) bekannt sein. Dagegen sind Servicebeschreibungen in Android rein funktional ausgeführt, sodass die Komposition von Services einfacher möglich wird. Ein Beispiel wäre ein Service zur Verteilung von Wartungsaufträgen, der auf Funktionalitäten wie Terminplanung, Warenverfügbarkeit und den zuvor skizzierten Services zur Identifizierung von passenden Dienstleistern zurückgreift, indem er die jeweiligen Services über die beschriebenen Intent-Tupel aus Aktion, Kategorien und einer Datenreferenz aufruft.

4 Zusammenfassung und Ausblick

Web Services als Realisierungstechnik für SOA ist in betrieblichen Kontexten etabliert und soll an dieser Stelle nicht grundsätzlich in Frage gestellt werden. Dennoch ist fraglich, inwiefern die über Serviceverzeichnisse, wie etwa UDDI, suggerierte Flexibilität tatsächlich

besteht und Punkt-zu-Punkt-Verbindungen zwischen Serviceanbietern und -konsumenten durch dynamische Bindungen entkoppelt sind.

Die Rekonfigurierbarkeit, d. h. Austauschbarkeit und Anpassbarkeit, von Services erscheint mit Android-artigen Servicebeschreibungen deutlich höher. Neben Konzepten zur dynamischen Bindung von Services wird dies durch die Verwendung dynamisch getypter Schnittstellen (im Gegensatz zu statischen Schnittstellen bspw. bei Web Services) unterstützt. Untersuchungen wie [BPT+11] implizieren, dass für die Förderung des Wachstums offener Systeme die Beschreibung von Abhängigkeiten bzw. Services mit Hilfe von „Capabilities“, also einer Beschreibung von Fähikeiten statt fixer Schnittstellen notwendig ist.

Eine abschließende Bewertung ist im Rahmen dieses Beitrags nicht möglich. Für vertiefende Untersuchungen sind folgende Forschungsfragen relevant:

- Inwiefern ist die Rekonfigurierbarkeit von SOAs nach dem Vorbild von Android im industriellen Kontext sinnvoll?
- Wie sind etablierte SOA-Techniken hinsichtlich Android-Konzepte zur Bildung lebendiger SOA-Communities zu ergänzen bzw. zu ersetzen?
- Welche Usecase-Typen eignen sich eher für die technische Umsetzung mit Hilfe von Webservices bzw. Android-Prinzipien?

Literatur

[BP10] Arno Becker und Marcus Pant. *Android 2: Grundlagen und Programmierung*. dpunkt.verlag, 2010.

[BPT+11] Thorsten Berger, Rolf-Helge Pfeiffer, Reinhard Tartler, Steffen Dienst, Krzysztof Czarnecki, Andrzej Wąsowski und Steven She. Variability in Software Ecosystems: How to Manage and How to Encourage?, 2011. Under review.

[Col05] John Colgrave. Generating a JAX-RPC Client for UDDI 3.0.2. `http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-jax-rpc-20050126.htm`, 2005.

[LCLC08] Chi-Chun Lo, Ding-Yuan Cheng, Ping-Chi Lin und Kuo-Ming Chao. A Study on Representation of QoS in UDDI for Web Services Composition. In *Proceedings of the 2008 International Conference on Complex, Intelligent and Software Intensive Systems*, CISIS '08, Seiten 423–428, Washington, DC, USA, 2008. IEEE Computer Society.

[MLM+06] C. Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter F. Brown und Rebekah Metz. Reference Model for Service Oriented Architecture 1.0. Public review draft, OASIS, May 31 2006.

[MS07] A. Mayrhofer und C. Spanring. A Uniform Resource Identifier for Geographic Locations ('geo' URI) draft-mayrhofer-geo-uri-00, 2007.

[ST07] Gernot Starke und Stefan Tilkov, Hrsg. *SOA-Expertenwissen – Methoden, Konzepte und Praxis serviceorientierter Architekturen*. dpunkt.verlag, Heidelberg, 2007.