Jens Knoop, Uwe Zdun (Hrsg.): Software Engineering 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 41

Eine Domänenspezifische Sprache für die technologieübergreifende Bereitstellung von Web Services

Florian Rademacher[1] Martin Peters[1] Sabine Sachweh[1]

Abstract: Web Services realisieren geräteunabhängige Datenschnittstellen zur Kommunikation von Web-Applikationen mit Internet-Clients, wie Smartphone-Apps, und Maschinen im Industrie-4.0-Kontext. In der Mehrzahl werden diese Schnittstellen entweder mit Hilfe des REST-Paradigmas und Standards des World Wide Web (WWW) oder des SOAP-Protokolls und XML-Nachrichten implementiert.
Der Beitrag stellt eine Domänenspezifische Sprache (Domain-Specific Language; DSL) für die effiziente, technologieübergreifende Entwicklung von Web Services vor. Ein Codegenerator überführt die DSL-Angaben in Java-Code, welcher auf einem erweiterbaren Framework basiert, das Unterschiede zwischen verschiedenen Web-Service-Technologien abstrahiert.

Keywords: Domänenspezifische Sprachen, Codegenerierung, Web Services

1 Einführung

Die Gruppe der über das Internet kommunizierenden Geräte wächst in zunehmendem Maße und wird zugleich immer heterogener. Neben Computern und Smartphone-Apps nutzen mittlerweile auch Maschinen und Sensoren im Rahmen einer Industrie 4.0 Technologien für den Datenaustausch über das Internet. Web Services stellen ein etabliertes Mittel für den geräteunabhängigen Datenaustausch dar. Das auf WWW-Standards aufbauende REST-Paradigma und das XML-basierte SOAP-Protokoll sind dabei die am weitesten verbreiteten Web-Service-Technologien [GK13]. Während SOAP in Szenarien mit hohen Anforderungen bspw. an die Übertragungsqualität zum Einsatz kommt, werden REST-Schnittstellen für eine effiziente Kommunikation über das HTTP eingesetzt [PZL08].
Der Beitrag stellt eine DSL für die technologieübergreifende Entwicklung von Web Services vor. Sie basiert auf dem in [RPS15] eingeführten Framework, mit dem Entwickler Geschäftslogik parallel über beliebige Web-Service-Technologien, bspw. für Smartphone-Apps via REST und für Maschinen via SOAP, anbieten können.

2 Spezifikation der Domänenspezifischen Sprache

Abbildung 1 zeigt das semantische UML-Modell der DSL. Es basiert auf den Abstraktionen des Frameworks aus [RPS15] und beschreibt die Sprachkonstrukte als Klassen und ihre Beziehungen als Assoziationen. Pakete kapseln semantische Sprachbereiche. So enthält das *Types*-Paket das Typsystem der Sprache. Es erlaubt die Konstruktion strukturierter

[1] Fachhochschule Dortmund, Fachbereich Informatik, Otto-Hahn-Straße 23, 44227 Dortmund, vorname.nachname@fh-dortmund.de

Datentypen und Listen aus primitiven Basistypen. Das *Services*-Paket definiert Konzepte zur Modellierung von Web Services. Im Kontext der DSL ist ein *Service* ein benanntes Element, welches einen spezifischen *Request* entgegennimmt und eine bestimmte *Response* produziert. Das *Interfaces*-Paket ermöglicht die Assoziation einer technologieneutralen Service-Beschreibung mit Web-Service-Technologien wie REST oder SOAP.

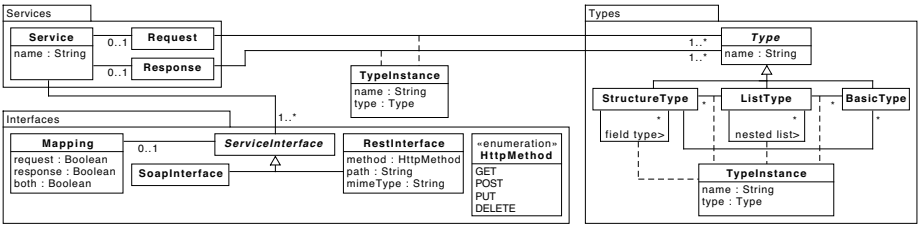

Abb. 1: Semantisches Modell der DSL

Listing 1 enthält Teile der aus dem Modell abgeleiteten DSL-Grammatik. Listing 2 zeigt einen mit der DSL modellierten Web Service, der per REST und SOAP verfügbar und Teil der Fallstudie aus [RPS15] ist. Ein Codegenerator überführt in der DSL vorliegenden

```
Service: 'service' name = ID ':' otoName = 'receives' otoVariables =
  TypeInstances itoName = 'returns' itoVariables = TypeInstances
  interfaces += ServiceInterface (interfaces += ServiceInterface)* ';' ;
ServiceInterface: name = 'interface' type = (RestInterface | SoapInterface)
  (mapping = MappingSpec)? ;
RestInterface: name = 'rest' 'method' method = HttpMethod 'path'
  path = STRING 'handles' mime = MimeSpec ;
SoapInterface: {SoapInterface} name = 'soap' ;
```

List. 1: Auszug der DSL-Grammatik

```
service UpdateParameterValue:
receives long wtsId, String paramName,
         Date timestamp, String value
returns int returnCode
interface rest method put
          path "wts/{wtsId}/{paramName}"
          handles "application/json"
interface soap;
```

List. 2: Beispiel-Service

Code in Framework-basierten Java-Code. Hierbei findet eine Transformation für jede konkrete Klasse aus Abbildung 1 statt. Die Geschäftslogik eines Services muss anschließend in einer Platzhaltermethode implementiert werden. Sie ist dann durch das Framework implizit über alle spezifizierten Web-Service-Technologien verfügbar. In einer erweiterten Fallstudie mit 25 Web Services, die parallel mittels REST und SOAP aufrufbar sein sollten, konnten aus 252 Zeilen DSL-Code 4384 Zeilen Java-Code generiert werden [RPS15]. Die im Folgenden implementierte Geschäftslogik umfasste 789 Zeilen Java-Code. Somit konnten rund 81% des Gesamtsystems automatisch erzeugt werden.

Literaturverzeichnis

[GK13] Gulden, Markus; Kugele, Stefan: A concept for generating simplified RESTful interfaces. In: Proceedings of the 22nd international conference on World Wide Web. International World Wide Web Conferences Steering Committee, S. 1391–1398, 2013.

[PZL08] Pautasso, Cesare; Zimmermann, Olaf; Leymann, Frank: RESTful Web Services vs. "Big"Web Services: Making the Right Architectural Decision. In: Proceedings of the 17th international conference on World Wide Web. ACM, S. 805–814, 2008.

[RPS15] Rademacher, Florian; Peters, Martin; Sachweh, Sabine: Design of a Domain-Specific Language Based on a Technology-Independent Web Service Framework. In: Software Architecture, S. 357–371. Springer, 2015.