Modellbasierte Entwicklung von Web Services mit Design by Contract

Gregor Engels[1,2], Marc Lohmann[1], Stefan Sauer[1,2]

[1]Institut für Informatik, [2]Software Quality Lab (s-lab)
Universität Paderborn, 33095 Paderborn
{engels, mlohmann, sauer}@upb.de

Abstract: Die Qualität Service-orientierter Softwaresysteme hängt wesentlich von der Einbindung der richtigen Dienste ab. Zwei grundsätzliche Aspekte kommen hierbei zum Tragen: (1) Passen die Anforderungen eines Service Requestors und die Servicebeschreibung eines Service Providers zusammen und (2) ist die Implementierung der Web Services korrekt gegenüber der Servicebeschreibung. Wir verwenden *Design by Contract* Techniken auf der Modellebene zur semantischen Beschreibung von Web Services und führen ein Matching-Konzept für den Vergleich von Requestor- und Provider-Kontrakten ein. Damit wird eine automatisierte, semantische Suche nach Web Services möglich. Wir erläutern zudem eine modellbasierte Vorgehensweise zur Entwicklung von Web Services. Hierbei werden aus den Modellen der Anwendung (1) die auswertbaren semantischen Beschreibungen und (2) Runtime-Assertions erzeugt, um die Korrektheit der Implementierung eines Web Services gegenüber seiner Spezifikation zu überprüfen. So wird eine konstruktive und prüfende Qualitätssicherung unterstützt.

1. Einleitung

Um die Qualität Service-orientierter Anwendungen zu gewährleisten, müssen Komponenten und Dienste gefunden werden, (1) deren Schnittstellenbeschreibung die Anforderungen der sie verwendenden Anwendungssoftware erfüllt und (2) deren Implementierung korrekt gegenüber der eigenen Schnittstellenspezifikation ist.

Zum Finden und Binden geeigneter Komponenten und Dienste ist eine möglichst präzise, semantische und mechanisch auswertbare Beschreibung des Verhaltens der angebotenen Dienste und Suchanfragen notwendig. Eine syntaktische Beschreibung von Web Services, wie sie in standardisierten Technologien für Service-orientierte Architekturen wie WSDL und UDDI bisher gewöhnlich verfolgt wird, allein genügt nicht.

Ein verbreiteter Ansatz zur semantischen Beschreibung von Operationen auf Implementierungsebene ist *Design by Contract*. Bei diesem Ansatz wird das Verhalten durch Vor- und Nachbedingungen (Kontrakte) beschrieben. Wir schlagen eine Verwendung von Kontrakten bereits zur Spezifikation der Operationen bzw. Schnittstellen auf der Modellebene vor. Im Rahmen einer modellbasierten Softwareentwicklung verwenden wir eine graphische Repräsentation der Kontrakte in einer Modellierungssprache. Die Vor- und Nachbedingungen dieser *visuellen Kontrakte* werden als UML-Objektdiagramme spezifiziert, welche über einem Klassendiagramm getypt sind. Die Modelle können dann

auf verschiedene Technologien abgebildet werden. Für Web Services bedeutet das, dass ein Service Provider bzw. Service Requestor seine angebotenen bzw. gesuchten Dienste mit Hilfe von Kontrakten beschreiben kann. Auf Basis dieser Beschreibungen kann ein Kompatibilitätsbegriff zwischen angebotenen und gesuchten Services entwickelt werden.

Damit Dienste in verschiedenen Domänen angeboten bzw. gesucht werden können, ist es notwendig, die Kontrakte gegenüber der Ontologie der Domäne zu spezifizieren. Wir zeigen, wie mithilfe einer Relation zwischen dem lokalen Modell, das die Implementierung eines Requestors oder Providers beschreibt, und der Ontologie der Domäne die gegenüber der Ontologie formulierten Kontrakte automatisch berechnet werden können.

Um die Korrektheit der Implementierung eines Dienstes sicherzustellen, generieren wir aus den Kontrakten Runtime-Assertions und unterstützen so ein modellbasiertes Testen.

2. Semantische Beschreibung von Web Services und Anfragen

Grundlage für ein automatisiertes Finden von Web Services ist eine semantische Beschreibung der angebotenen Dienste eines Providers sowie der Suchanfrage eines Requestors. Für diese Beschreibung verwenden wir Kontrakte. In unserem Ansatz besteht ein Kontrakt aus einem Paar von Objektdiagrammen, die über einem Klassendiagramm getypt sind. Der Kontrakt beschreibt das Verhalten einer Operation. Er spezifiziert die Änderungen des Datenzustandes mittels Vor- und Nachbedingungen. So kann ein Service für die Bestellung von Büchern durch den Provider-Kontrakt in Abb. 1 beschrieben werden. Die einfache, intuitive Interpretation eines solchen Kontrakts besagt, dass der Service die Daten auf der linken Seite als Eingabe benötigt. Das zugesicherte Ergebnis der Ausführung wird auf der rechten Seite angegeben.

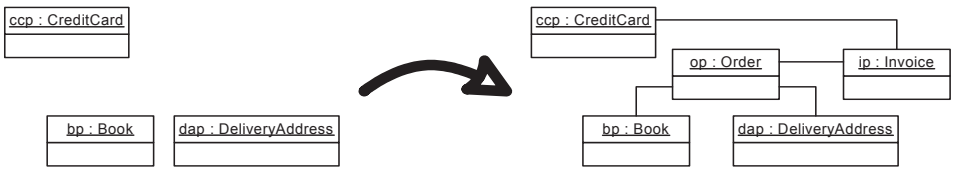

Abbildung 1: Kontrakt zur Beschreibung eines Web Services

Im Gegenzug muss ein Requestor die Semantik eines gesuchten Services beschreiben. Entsprechend beschreibt die linke Seite eines Requestor-Kontrakts, welche Informationen der Requestor an einen Web Service zu schicken bereit ist. Die rechte Seite beschreibt die Situation, welche ein Requestor durch den Aufruf eines Services erreichen möchte. Das zugehörige Klassendiagramm stellt ein gemeinsames Datenmodell bzw. eine Ontologie dar und definiert einen gemeinsamen Sprachraum. Mit Hilfe der visuellen Kontrakte kann also die Semantik von Services über einer Ontologie spezifiziert werden.

Aufbauend auf den visuellen Kontrakten haben wir in [HHL04, HHL05] ein Matching-Konzept definiert, welches mittels Subgraph-Relationen beschreibt, wann ein Provider die Anforderungen eines Requestors erfüllt. Dieses Matching-Konzept haben wir in einer Werkzeugkette mit aktuellen Semantic-Web-Technologien umgesetzt.

3. Einbettung des Matching-Prozesses in SOA

Das auf Kontrakten basierende Matching-Konzept erfordert ein gemeinsames Datenmodell für Provider- und Requestor-Kontrakte. Da aber Requestor und Provider prinzipiell (1) unabhängig entwickelt werden und (2) Dienste für unterschiedliche Domänen anbieten bzw. nutzen, kann die Existenz eines gemeinsamen Datenmodells nicht vorausgesetzt werden. Provider und Requestor können unterschiedliche Klassendiagramme für ihre Implementierung benutzen, die nicht identisch mit einer Ontologie sein müssen. Die Schnittstellen der Services und Requests sind dann nicht direkt über der Ontologie formuliert. Ein solches Szenario zeigt Abb. 2. Die Klasse *User* im Klassendiagramm des Requestors entspricht z.B. der Klasse *Customer* in der Ontologie, d.h. die Klasse *Customer* ist bei einem Request in der Domäne für das Bestellen von Büchern zu verwenden.

Im Folgenden zeigen wir, wie eine geeignete Servicebeschreibung während der Entwicklung einer Anwendung generiert werden kann. Das Vorgehen für Service Requests ist entsprechend. Abbildung 3 zeigt einen Überblick über unsere Vorgehensweise.

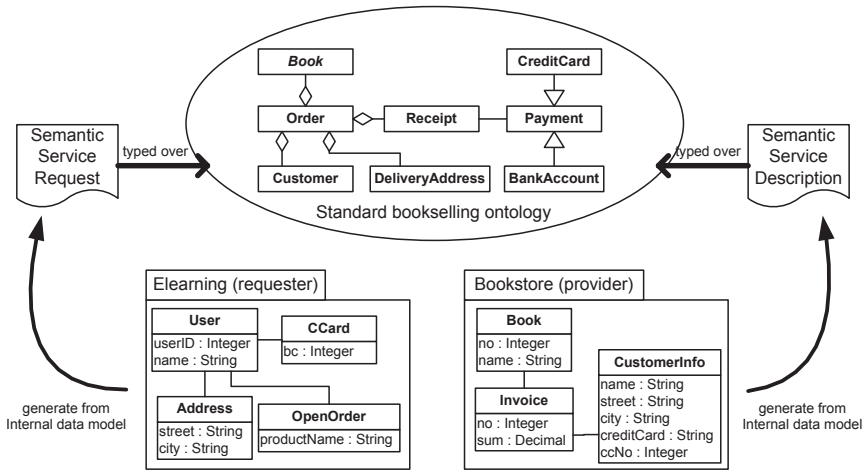

Abbildung 2: Szenario mit 3 verschiedenen Problemdomänen

Im ersten Schritt berechnet ein Algorithmus aus dem vom Softwareentwickler erstellten Entwurfsmodell (Implementation Level Diagram) automatisch ein Interface Information Model (IIM) für jede Schnittstelle, die als Web Service angeboten wird. Das IIM enthält alle Klassen, die zur Spezifikation des als Web Service angebotenen Verhaltens auf Implementierungsebene benötigt werden. Obwohl das IIM immer noch den individuellen Sprachgebrauch des Providers repräsentiert, erlaubt diese Projektion eine Konzentration auf die für die öffentlichen Beschreibungen relevanten Informationen. Nur Klassen des IIM müssen mit den Begriffen einer Ontologie in Beziehung gebracht werden.

In einem zweiten Schritt bildet ein Entwickler das IIM auf eine standardisierte Ontologie ab, indem er die Modellelemente der beiden Klassendiagramme in Beziehung setzt. Dies erlaubt eine einfache Überprüfung von Konsistenzbedingungen und eine einheitliche Generierung von konkreten Schnittstellen für Web Services.

Diese Generierung ist der dritte Schritt unserer Methodik. Wenn feststeht, wie die statischen Konzepte der Klassendiagramme aufeinander abgebildet werden, können Schema-Evolutionstechniken verwendet werden, um private Kontrakte in öffentliche, semantische Kontrakte zu übersetzen. Die Beziehung zwischen internen Strukturen und der Ontologie muss nur einmal definiert werden. Bei Änderungen der Implementierung sind nur inkrementelle Anpassungen nötig. Selbst bei der Veröffentlichung einer Schnittstelle in mehreren Domänen erlaubt dieser Ansatz eine andauernd konsistente Spezifikation.

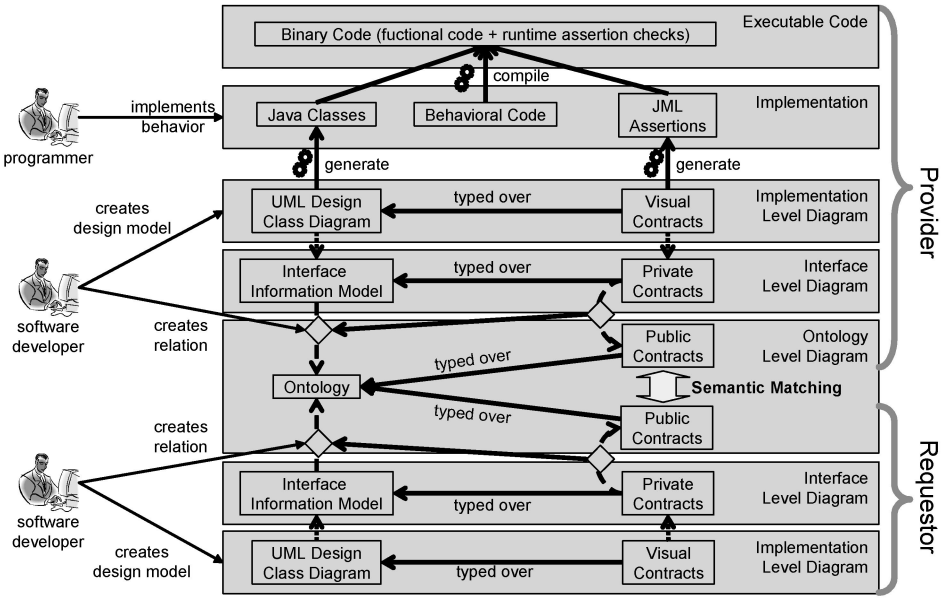

Abbildung 3: Kontraktbasierte Modellierung, Suche und Validierung von Web Services

Wenn sowohl Provider als auch Requestor ihre privaten Kontrakte auf öffentliche Kontrakte, die über einer identischen Domäne (Ontologie) formuliert sind, abgebildet haben, kann überprüft werden, ob der Provider alle Anforderungen des Requestors erfüllt.

4. Modellbasierte Entwicklung der Web Services

Für die Implementierung der Web Services erfüllen die visuellen Kontrakte zwei Aufgaben (siehe Abb. 3, vgl. [ELS05]): (1) Sie werden als Verhaltensmodelle interpretiert, aus denen Code für das Testen und die Laufzeitüberprüfung generiert werden kann. (2) Sie dienen als Spezifikation des Verhaltens, welches vom Programmierer als funktionaler Code zu implementieren ist.

Aus dem Entwurfsmodell der Service-Implementierung wird Java-Code generiert: Java-Klassen werden aus dem Klassendiagramm erzeugt; zu jedem visuellen Kontrakt werden *Assertions* in der Schnittstellenspezifikationssprache JML (Java Modeling Language) generiert, mit denen wir die Java-Operationen aus dem Klassendiagramm annotieren. Modellierung und Codegenerierung werden durch ein Eclipse-Plug-In unterstützt.

Anschließend implementiert ein Programmierer eine vollständige und lauffähige Anwendung, indem er den fehlenden funktionalen Code in die generierten Java-Fragmente einfügt. Hierbei werden die visuellen Kontrakte als Spezifikation benutzt. Der Programmierer darf zusätzliche Operationen oder Klassen hinzufügen, allerdings darf er die generierten JML-Kontrakte nicht verändern. So wird die Konsistenz zwischen visuellen Kontrakten und JML-Kontrakten sichergestellt.

Nachdem der Programmierer den funktionalen Code implementiert hat, kann er mit dem JML-Compiler ausführbaren Binärcode generieren. Dieser Binärcode besteht aus dem funktionalen Code und den Laufzeittests, die aus den JML-Assertions generiert wurden. Die Laufzeittests überwachen die Vor- und Nachbedingungen während der Ausführung der Anwendung. Sie erlauben so die automatische Überprüfung, ob das manuell implementierte Verhalten einer Operation die JML-Spezifikation und transitiv die Verhaltensspezifikation der visuellen Kontrakte auf Modellebene erfüllt. Auf diese Weise kann die Korrektheit der Implementierung überprüft werden.

5. Zusammenfassung und Ausblick

Wir haben einen modellbasierten Ansatz für die *semantische* Beschreibung, den Vergleich und die Validierung von Web Services vorgestellt. Zur Beschreibung der Semantik von Web Services und Service Requests werden Kontrakte auf Modellebene eingesetzt. Die Vor- und Nachbedingungen dieser *visuellen Kontrakte* sind mit UML-Objektdiagrammen beschrieben, die über einem Klassendiagramm getypt sind. Auf Basis dieser Verhaltensbeschreibung haben wir einen Kompatibilitätsbegriff eingeführt, der sich auf die Überprüfung von Sub-Graphen zurückführen lässt. Weiterhin haben wir eine Methodik für die Generierung einer Abbildung entwickelt, um die über den Klassendiagrammen des Providers und Requestors getypten Kontrakte automatisch auf eine Semantikbeschreibung gegenüber der Ontologie einer Domäne abzubilden. So kann ein Web Service Domänen übergreifend angeboten bzw. genutzt werden. Durch die Generierung von Runtime-Assertions aus Kontrakten auf Modellebene kann die Korrektheit der Implementierung der Web Services zur Laufzeit getestet werden. Abgerundet wird der Ansatz durch Werkzeuge für den Vergleich von Kontrakten und die Generierung des Java/JML-Codes. Durch Verwendung dieses Ansatzes, der sowohl den semantischen Vergleich von Anforderungen und Servicebeschreibungen als auch die modellbasierte Validierung von Service-Implementierungen unterstützt, kann die Qualität von Service-orientierten (oder Komponenten-basierten) Anwendungen verbessert werden. Der Ansatz wird gegenwärtig in einem Projekt des s-lab mit einem industriellen Partner eingesetzt und evaluiert.

Literaturverzeichnis

[ELS05] Engels, G.; Lohmann, M.; Sauer, S.: Executable visual contracts. In: Proc. IEEE Symposium on Visual Languages and Human-centric Computing (VL/HCC'05); to appear.

[HHL04] Hausmann, J.H.; Heckel, R.; Lohmann, M.: Model-based discovery of Web Services. In: Proc. IEEE Intl. Conference on Web Services (ICWS'04), 2004; S. 324-331.

[HHL05] Hausmann, J.H.; Heckel, R.; Lohmann, M.: Model-based development of Web service descriptions enabling a precise matching concept, International Journal of Web Services Research, Vol. 2, No. 2, 2005.