Globale Referenzen basierend auf SOAP

Gerhard Austaller, Eric Roth
Fachgebiet Telekooperation, Technische Universität Darmstadt
Hochschulstraße 10, 64289 Darmstadt
gerhard@tk.informatik.tu-darmstadt.de

Abstract: SOAP wurde als leichtgewichtiges Protokoll zum Austausch von strukturierten Informationen in einer verteilten Umgebung konzipiert und bildet so die Basis der „Web Service"-Technologie. Zur Zeit bietet SOAP nur die Möglichkeit lokale Objekte, also innerhalb einer Nachricht zu referenzieren, nicht aber global über Nachrichtengrenzen hinweg. Dieses Paper beschreibt die globale Referenzierung von Web Services in SOAP Nachrichten und deren Implementierung.

1 Motivation

In der verteilten objektorientierten Programmierung ist das Konzept von Remotereferenzen („remote (object) reference") ein schon lange bekanntes und bewährtes Konzept. Objekte können dabei nicht nur innerhalb des gleichen Adressraums einer Anwendung, sondern auch über Anwendungs- und Rechnergrenzen hinweg referenziert werden. Bekannte Beispiele für derartige Plattformen sind Voyager [voy], CORBA [cor] und JAVA RMI [jav].

In diesem Paper stellen wir ein Modell zur globalen Referenzierung von Web Services vor und wie die Darstellung dieser in SOAP-Nachrichten kodiert wird. Dazu wurde das SOAP Element `href` erweitert. In Methodenaufrufen oder in Rückgabewerten können damit Referenzen auf Web Services übergeben werden. Web Services oder Anwendungen, die die Semantik dieser SOAP-Nachrichten verstehen, können aus den Informationen eine Verbindung zum referenzierten Web Service rekonstruieren.

Zusätzlich stellen wir eine Implementierung in .NET vor, die Web Services als Objekte kapselt und aus Remotereferenzen automatisch .NET-Proxies erstellt.

2 Verwandte Arbeiten

Die vorgestellte Arbeit verbindet das beste aus zwei Welten: Remotereferenzen zur transparenten Referenzierung entferntes Objekte und statusbehaftete Web Services.

Remotereferenzen sind aus vielen Programmiersprachen bekannt. Nachdem eine Remotereferenz auf ein entfernten Objekt erlangt wurde, kann damit (fast) transparent wie mit einem lokalen Objekt gearbeitet werden. Gegebenenfalls muss Fehlerbehandlung für den Fall von Verbindugsabbrüchen vorgesehen sein. Je nach Umfang der Programmierspra-

che bleiben OO-Konzepte wie Vererbung und Polymorphismus erhalten. Auch die Verwendung von Remotereferenzen beim Aufruf entfernter Methoden als Parameter wird oft unterstützt. Die bekanntesten Beispiele für derartige Programmiersprachen/-umgebungen sind Voyager [voy], CORBA [cor] und JAVA RMI [jav].

Traditionell sind Web Services zustandslos. In vielen Situationen sind aber zustandsbehaftete Dienste sinnvoll. Dabei ist aber zu unterscheiden, ob jeder Benutzer des Web Services eine eigene Instanz bekommt oder alle Benutzer auf ein und demselben (zustandsbehafteten) Dienst arbeiten. Toolkits, die derartige Web Services unterstützen, sind GLUE [glu], das .NET Framework [mic] und die Implementierung von Apache [apa].

Die vorgestellten Web Service Toolkits bieten oft eine objektorientierte Sicht auf die zugrunde liegenden Web Services. Sie ermöglichen zum Beispiel das Erstellen programiersprachenspezifischer Schnittstellenklassen aus den WSDL-Beschreibungen. Nach dem Binden eines Proxyobjektes, das aus der WSDL-Beschreibung erzeugte Interface implementiert, an den Web Services, können auf dem Web Service transparent Methoden aufgerufen werden. Zu einer „echten“ Remotereferenz fehlt vielen Toolkits aber die Möglichkeit, die Proxies als Parameter beim Aufruf anderer (in Proxies gekappselter) Web Services zu verwenden. Die wenigen Toolkits, die das unterstützen, wie das .NET Remoting Framework, kodieren die Informationen in binärer Form und nicht in SOAP Nachrichten.

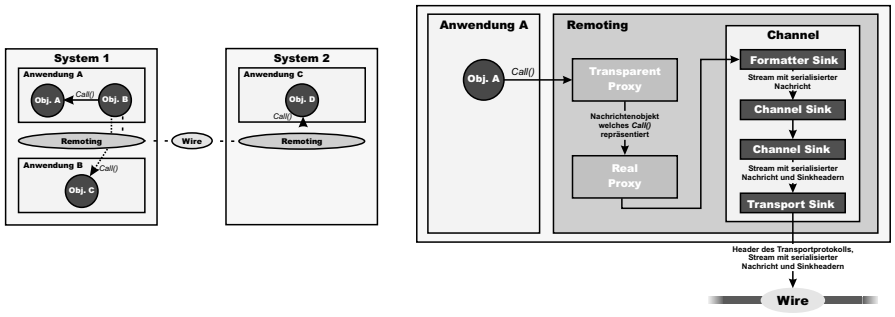
Unsere Arbeit setzt dort fort, wo viele Toolkits aufhören. Wir schlagen eine Erweiterung des `href` Elements in SOAP Nachrichten zur globalen Referenzierung von Web-Diensten vor. Diese Erweiterung enthält alle Informationen, die ein Klient benötigt um einen Web Service anzusprechen.

3 Das .NET Remoting Framework

.NET stellt eine sehr mächtige, flexible und erweiterbare Plattform zur Realisierung von verteilten Anwendungen dar. Dem Programmierer wird ein durchgängiges Modell zur Verfügung gestellt wird, wie Objekte und Methodenaufrufe in SOAP-Nachrichten serialisiert, übertragen und auf der Empfängerseite deserialisiert werden. Zum besseren Verständnis der im Paper vorgestellten Implementierung der Remotereferenzen auf Basis von SOAP muss kurz auf das .NET Remoting Framework eingegangen werden.

In .NET werden alle Daten durch Objekte dargestellt. Remoting erlaubt den Zugriff auf entfernte Objekte nur über Methoden oder Properties. Der Remotingmechanismus tritt immer dann in Kraft, wenn ein Objekt auf ein anderes Objekt in einem anderen Adressbereich in Form eines Methodenaufrufs zugreift. Die Objekte müssen sich dabei nicht zwangsweise auf einem anderen System befinden. Abbildung 1(a) verdeutlicht das in vereinfachter Form.

In Remoting gibt es drei verschiedene Arten von Objekten. Objekte, die das `Serializable` Attribut besitzen, können durch Serialisierung eine Anwendungsgrenze überschreiten. Ein Objekt, das von `MarshalByRefObject` erbt, wird in Form einer Remotereferenz (`ObjRef`) über eine Anwendungsgrenze weitergegeben. Alle anderen Objekte dürfen nicht im Rahmen eines RPCs vorkommen.



(a) Anwendungen von .NET Remoting. (b) .NET Remoting Architektur.

Abbildung 1: .NET Remoting.

Ein Objekt verwendet in Remoting zum Aufruf einer Methode auf einem Remoteobjekt einen „Transparent Proxy“. Dieser wird aus der Klassenbeschreibung des referenzierten Objekts automatisch zur Laufzeit erzeugt und hat exakt die gleiche Schnittstelle wie das Originalobjekt. Dieser erzeugt bei einem Methodenaufruf ein Nachrichtenobjekt, das alle notwendigen Informationen, wie den Methodennamen und die übergebenen Parameter, enthält. Dieses Nachrichtenobjekt wird an den ebenfalls automatisch erzeugten „Real Proxy“ übergeben der das Nachrichtenobjekt an einen der verfügbaren Channels übergibt.

Ein Channel sorgt für die Übermittlung eines Nachrichtenobjektes zur Zielanwendung. Dabei wird immer jeweils ein Channel auf der Clientseite und auf der Serverseite durchlaufen. Ein Channel besteht aus einer Reihe von gekoppelten Sinks. Ein Sink ist eine Komponente, die die übergebene Nachricht manipulieren kann und dann an den nächsten Sink in der Kette weiterreicht. Der Formatter Sink bringt die Nachricht auf der Clientseite unter Verwendung eines Formatters in die serialisierte Darstellung. Auf der Serverseite sorgt er für die Deserialisierung des Nachrichtenobjekts in seine ursprüngliche Form.

Dem Formatter Sink können mehrere spezialisierte Channel Sinks folgen, die die Verarbeitung der Nachricht nach der Serialisierung kontrollieren. Der letzte Sink sorgt für den Aufbau einer Transportverbindung, und das Senden der im Stream erhaltenen Daten.

Die Serialisierung von Objekten geschieht in .NET unter Verwendung eines so genannten Formatters. Der Formatter ist in Remoting durch den Formatter Sink beliebig austauschbar.

4 Anforderungen an den Objektkodierer für SOAP

Bei der Übertragung von Objekten zwischen Client und Server wird genau genommen nur der Zustand des Objektes übertragen und das Objekt anhand der Zustandinformationen und der Typinformation rekonstruiert. Dazu muss der Zustand in eine kodierte Darstellung gebracht werden. Bei Web Services wird der Zustand eines Objektes durch eine XML Struktur dargestellt, die in einer SOAP Message übertragen wird. Die Serialisierung und Deserialisierung wird dabei meistens durch klassenspezifische Kodierer durchgeführt.

Der in .NET Remoting vorhandene Kodierer unterstützt bisher nur die Kodierung von Objekten im XML Namespace des .NET Frameworks und ist somit nicht für offene Systeme verwendbar. Ferner enthält er eine Reihe von Fehlern, die den Gebrauch in der Praxis unmöglich machen.So wurden Probleme mit der Serialisierung von zweidimensionalen Arrays und beim Codieren von Attributwerten. Es wurde daher ein eigener Objektkodierer erstellt, der alle zustandsrelevanten Eigenschaften eines Objektes in eine serialisierte Darstellung überführen kann und aus dieser Darstellung eine neue Instanz des Objektes erzeugt, dessen Eigenschaften und Verhalten identisch zum Ursprungsobjekt sind. In unserem Fall wird als Darstellung SOAP Version 1.1 benutzt.

Der Zustand eines Objektes wird eindeutig durch die Belegung seiner Objektvariablen bestimmt. Eine Objektvariable wird bei der Kodierung abstrakt als ein Name / Wert Tupel betrachtet und häufig als Member des Objekts bezeichnet. In Verbindung mit SOAP ist es oft notwendig der Objektvariablen einen anderen Membernamen zuzuordnen, um die XML Darstellung anpassen zu können. Aus einem Objekt muss deshalb ein Satz von Memberinformationen erzeugbar sein, die für jedes Member dessen Namen in XML Darstellung enthält. Ferner muss enthalten sein, ob ein Member als Attribut serialisiert werden soll und ob der Wert des Members eingebettet werden soll. Eingebettete Objekte werden nicht als lokale Referenz serialisiert und dadurch wie ein „call by value“ Objekt behandelt. Die Serialisierung eines Objektes führt in der Regel zur Serialisierung von weiteren Objekten, die Bestandteil des gleichen Objektgraphs sind. Der Kodierer muss sich darum kümmern, dass der Objektgraph vollständig und eindeutig serialisiert wird. Um die Deserialisierung von Objekten durchführen zu können, müssen deren Typen in der serialisierten Darstellung enthalten sein.

Der Kodierer muss die internen Typnamen der Objekte in XML Typnamen umsetzen. In XML besteht ein Typname aus dem lokalen Namensteil und dem zugehörigen Namespace. Hier ist zu berücksichtigen, dass eine Reihe von internen Typen auf existierende einfache Standardtypen von XML Schema umgesetzt werden sollten. Bei der Deserialisierung ist eine Funktion notwendig, welche für einen XML Typnamen den zu verwendenden internen Typ bestimmt.

Der durch die SOAP Message durchgeführte Methodenaufruf stellt einen Spezialfall bei der Serialisierung und Deserialisierung dar. Die Informationen eines Methodenaufrufs müssen durch den Kodierer korrekt verarbeitet werden, obwohl es sich nicht um ein reguläres Objekt handelt.

5 Globale Referenzen

SOAP bietet bisher kein Konzept, welches die Referenzierung von Objekten außerhalb einer Message erlaubt. Daher erweitert die Map2Carry Plattform den existierenden SOAP Standard um eine XML Struktur zur Beschreibung solcher Remotereferenzen.

Zur Referenzierung von XML Strukturen wird häufig XLink [xli] eingesetzt. XLink beschreibt einen Satz von Attributen, die in einem XML Element zur Referenzierung eines anderen Elements verwendet werden können. XLink hat sich jedoch für Remotereferen-

zen im Rahmen einer eigenen Evaluierung als ungeeignet erwiesen. Dazu betrachten wir kurz die in Tabelle 1 aufgeführten Anforderungen an die Kodierung einer Remotereferenz. Da XLink nicht die vollständige Kodierung dieser Informationen ermöglicht, verwendet Map2Carry eine eigenständige Struktur, die an das existierende SOAP Encoding angelehnt ist.

Information	Beschreibung	XML Element Name	Schema Typ
Objekt ID	Die ID des durch die Remotereferenz referenzierten Objekts.	objectID	string
URIs der verfügbaren Channels	Die URIs aller Wege auf denen das Objekt erreichbar ist.	channelUris	string[]
Objekttyp	Der XML Typname des Remoteobjektes.	name	qName
Implementierte Schnittstellen	Die XML Typnamen der implementierten Schnittstellen.	implements	qName[]
Basistypen	Die XML Typnamen der Basistypen von denen das Remoteobjekt erbt.	extends	qName[]
Zusatzinformationen	Array von beliebigen Objekten.	details	anyType[]

Tabelle 1: Globale Referenzen.

Die Objekt ID identifiziert das Objekt eindeutig auf dem Zielsystem. Das Objekt ist möglicherweise auf mehreren Wegen erreichbar. SOAP kann zum Beispiel über HTTP oder über SMTP verwendet werden. Auch die Verwendung mehrerer HTTP Ports ist denkbar. Jeder Weg zum Remoteobjekt sollte im Rahmen einer Remotereferenz übermittelt werden, um dem Client die Wahl des besten Protokolls zu ermöglichen. Zur Adressierung der Channels werden URIs verwendet. Die vollständige Adresse zum Remoteobjekt ergibt sich durch die Verknüpfung von Objekt ID und Channel URI. Für ein Objekt mit der ID object1 ergibt sich beim Channel http://www.Map2Carry.de/objects/ die Zieladresse http://www.Map2Carry.de/objects/object1.

Der Typ des Objekts wird auf der Clientseite benötigt, um das Remoteobjekt durch lokale Referenzen an lokale Objekte zu koppeln. Die Klassenhierarchie des Objektes und die implementierten Schnittstellen müssen ebenfalls bekannt sein, da der Client nur so die möglichen Typkonvertierungen ermitteln kann.

Zusätzlich sollen erweiterte Informationen als Objekte beliebigen Typs an die Referenz koppelbar sein. Dies ermöglicht die Erweiterung der Referenz etwa für die Übertragung häufig benötigter Informationen des Remoteobjektes für die Zwischenspeicherung auf dem Client.

Der Typname einer Remotereferenz lautet `href` und ist im SOAP Encoding Namespace eingeordnet. Mit den in Tabelle 1 aufgelisteten Elementnamen ergibt sich die in Abbildung 2 dargestellte Struktur zur Kodierung einer Remotereferenz in SOAP.

```
<soap-enc:href xsi:type="soapenc-href">
  <objectID>/W3O1rcKKm6CrqBJdOBmlqRou_1.rem</objectID>
    <type name="m2c:M2CFeatureCollectionRemotingType">
        <implements xsi:type="soap-enc:Array"
            arrayType="xsd:qName[1]">
            <item>m2c:M2CFeature</item>
            <item>m2c:M2CFeatureCollection</item>
        </implements>
        <extends xsi:type="soap-enc:Array"
            arrayType="xsd:qName[1]">
            <item>m2c:M2CFeatureType</item>
        </extends>
    </type>
    <channelUris xsi:type="soap-enc:Array"
        arrayType="xsd:string[1]">
        <item>http://www.Map2Carry.de/service</item>
    </channelUris>
    <details xsi:null="1" />
</soap-enc:href>
```

Abbildung 2: Kodierungsbeispiel einer Remotereferenz in SOAP.

6 Implementierung des SOAP Formatters

Zur Realisierung wurde ein Formatter für das SOAP Protokoll Version 1.1 implementiert. Die Architektur des Formatters lässt sich in mehrere Bereiche unterteilen. Die zentrale Klasse `SoapFormatter` implementiert `IRemotingFormatter` und stellt dadurch die Methoden `Serialize` und `Deserialize` zur Verfügung, die den Ablauf der Serialisierung und Deserialisierung steuern. Diese werden für die einzelnen Objekte des Objektgraphen von Klassen übernommen, die die Schnittstelle `ISoapObject` implementieren.

Sämtliche Informationen und Dienste zu XML und SOAP, wie die XML Typnamen der Objekte oder die Namen der SOAP Protokollelemente und -attribute, werden über einen `ISoapInfoProvider` bereitgestellt. Dieser ermöglicht über einen `ISoapCoder` den Zugriff auf Informationen und Dienste der aktuell verwendeten SOAP-Protokollversion. Die eigentliche Verarbeitung der Objekte des Objektgraphen geschieht durch Klassen, die `ISoapObject` implementieren.

6.1 De-/Serialisierung durch ISoapObject

SOAP unterstützt die Kodierung verschiedener Arten von Datenstrukturen. Die Verwendung von XML Schema ermöglicht in SOAP einfache Datentypen, komplexe Datentypen und Aufzählungen zu kodieren. Zusätzlich definiert SOAP die Kodierung von Arrays, Feh-

lermeldungen und natürlich Methodenaufrufen. Die Map2Carry Plattform erweitert SOAP um die in Kapitel 5 beschriebenen Remotereferenzen, welche ebenfalls eine eigene Struktur haben. Jede dieser unterschiedlichen Datenstrukturen muss in einer anderen Form kodiert werden. Aus diesem Grund wird einer Datenstruktur jeweils eine eigene Klasse zur Kodierung zugeordnet, die die Serialisierung und Deserialisierung über die Schnittstelle `ISoapObject` ermöglicht.

Die `SoapObjectFormatterFactory` kennt die in Tabelle 2 aufgelisteten Datenstrukturen und verfügt je Datenstruktur über eine Klasse zur Kodierung der Datenstruktur. Die Factory kann nachträglich um weitere Datenstrukturen erweitert werden.

Datenstruktur	Klasse	Beschreibung
Array	SoapArrayObject	Kodiert einfache, rechteckige und fragmentierte Arrays.
Einfacher Datentyp	SoapPrimitiveObject	Kodiert Datenstrukturen, die in einem XML Schema einem SimpleType entsprechen.
Komplexer Datentyp	SoapComplexObject	Kodiert Instanzen einer Klasse.
Aufzählung	SoapEnumObject	Kodiert Enumerations.
ISerializable	SoapISerializableObject	Kodiert Instanzen einer Klasse, welche die Schnittstelle ISerializable implementiert.
Methodenaufruf	SoapMessageObject	Kodiert die Informationen zu einem Methodenaufruf.
Fehlermeldung	SoapFaultObject	Kodiert einen SoapFault, welcher im Rahmen eines Methodenaufrufs auftritt.
Remotereferenz	SoapReferenceObject	Kodiert die in Kapitel 5 definierten Remotereferenzen.

Tabelle 2: Unterstützte Datenstrukuren und die zugehörigen ISoapObjecte.

Zur Kodierung einer SOAP Nachricht werden neben den eigentlichen Objekten noch eine Reihe weitere Informationen benötigt. Diese sind über einen `ISoapInfoProvider` abrufbar, der im nächsten Abschnitt beschrieben wird.

6.2 Metainformation durch ISoapInfoProvider und IsoapCoder

Zur Serialisierung einer Datenstruktur benötigt ein `ISoapObject` eine Reihe von Metainformationen, die über Mechanismen des .NET Frameworks abrufbar sind. Ein Beispiel hierfür ist die Umsetzung des internen CLR Typs in den entsprechenden XML Typ.

Alle Mechanismen, die zur Abfrage und Verarbeitung von Metainformationen notwen-

dig sind werden im Map2Carry Formatter durch den `ISoapInfoProvider` gekapselt. Bei der Serialisierung müssen die XML spezifischen Informationen zum Typ und den Members eines Objektes bekannt sein, was druch das `ISoapInfoProvider` ermöglicht wird. Daneben ermöglicht es auch die Extraktion der internen Memberobjekte eines Objekts, sowie einen Test auf Arrays, `SoapFaults` und ob ein Objekt in Form einer lokalen Referenz kodiert wurde.

Neben den beschriebenen Methoden bietet der `ISoapInfoProvider` auch den Zugriff auf versionsspezifische Informationen des aktuellen SOAP Protokolls. Dies geschieht durch Bereitstellung eines austauschbaren `ISoapCoders`. Durch die absehbare Weiterentwicklung von SOAP werden auch in Zukunft neue Formatter für die jeweilige Version erforderlich. Der `ISoapCoder` übernimmt die Kodierung der SOAP spezifischen Strukturen und die Bereitstellung der definierten Namen und Namespaces einer Nachricht. Er hält somit den Formatter flexibel.

7 Zusammenfassung

Das Paper beschreibt eine Methode zur Referenzierung von Web Services und deren Abbildung in SOAP-Nachrichten. Dazu wurde das `href` Element um zusätzliche Elemente erweitert. Weiters wurde die Implementierung in .NET erläutert und eine darauf aufbauende Anwendung erläutert.

Unsere Implementierung erlaubt nun Web Services in SOAP-Nachrichten eindeutig zu referenzieren. Wie in OOP Sprachen kann in Methodenaufrufen oder in Rückgabewerten direkt eine Referenz auf einen Web Service übergeben werden. Aus den in der Remotereferenz kodierten Informationen kann der Empfänger eine direkte Verbindung zum referenzierten Web Service herstellen.

Literatur

[apa] Web Services Project @ Apache. `http://ws.apache.org/`, gesehen am 22.11.2004.

[cor] The OMG's CORBA Website. `http://www.corba.org/`, gesehen am 22.11.2004.

[glu] Glue. `http://www.webmethods.com/`, gesehen am 22.11.2004.

[jav] Java RMI Specification. `http://java.sun.com/products/jdk/rmi/reference/whitepapers/index.html`, gesehen am 22.11.2004.

[mic] Microsoft .NET Remoting: A Technical Overview. `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/hawkremoting.asp`, gesehen am 22.11.2004.

[voy] Voyager. `http://www.recursionsw.com/voyager.htm`, gesehen am 22.11.2004.

[xli] XML Linking Language (XLink) Version 1.0. `http://www.w3.org/TR/xlink/`, gesehen am 22.11.2004.