Dokumenten- und Schema-basierte Services mit WCF

Hartmut Wilms, 26.02.2007

BASTA! .NET, VISUAL STUDIO & MORE

dot.net konferenz 2007

Hartmut Wilms

- Principal Consultant
- innoQ Deutschland GmbH
- SOA, Web Services, WCF

- hartmut.wilms@innoq.com
- http://www.innoq.com/blog/hw
- http://www.innoq.com

BASTA! .NET, VISUAL STUDIO & MORE

dot.net konferenz 2007

1

Was kann ich hier erfahren?

- Warum Dokumenten- & Schema-basierte Services?
  - SO Grundsätze & Prinzipien
  - SO Konzepte
- WCF
  - Service-Modelle
  - XML Messaging in WCF
  - Schema-Validierung

SO(A)

SOA

- Service
  - Schlüsselkonzept
- Orientiert
  - Philosophie (Paradigma)
  - Prinzipien und Grundsätze
- Architektur
  - Umsetzung der Philosophie in einem IT-Konzept

Four Tenets

- Boundaries are explicit
- Services are Autonomous
- Services share Schema and Contract, not Class
- Compatibility is based upon Policy

Don Box: “A Guide to Developing and Running Connected Systems with Indigo”

Explizite Grenzen

- Jeglicher Service-Zugriff erfolgt über das explizite, publizierte Interface.
- Keine versteckten Annahmen, Interaktion ist explizit.
- Kein gemeinsamer Status/Kontext, sondern Stateless Services.
- ➢ Ein Service bietet seine Funktionalität über ein explizites Interface, welches die Interna kapselt, an.
  Die Interaktion mit einem Service ist ein expliziter Akt, der auf einem Nachrichtenaustausch zwischen Anbieter und Konsument basiert.

Services sind autonom

- Die einzige Beziehung zur Außenwelt – zumindest aus der SOA Perspektive – ist das publiziertes Interface.
- Service Provider & Consumer können unabhängig voneinander geändert, versioniert oder deployt werden.
- Die Laufzeitumgebung eines Service muss ohne Auswirkungen auf die Konsumenten geändert werden können (und umgekehrt).

Schema & Contract

- Alle notwendigen Informationen zur Interaktion mit dem Service sind in Schema & Contract definiert.
- Provider darf nicht annehmen, dass ein Consumer fähig ist, Code aus der Service-Implementation wiederzuverwenden.
- Idealerweise werden Schema & Contract in Form von XML Dokumenten definiert , da XML in jeder erdenkbaren Programmierumgebung und Plattform unterstützt wird.

Kompatibilität basiert auf Policy

- Die technischen Fähigkeiten und Anforderungen von Anbieter und Konsument müssen kompatibel sein.

Lose Kopplung

- Kopplung ist der Grad der (nicht-funktionalen) Abhängigkeit zwischen zwei Artefakten
- Lose Kopplung bedeutet Autonomie
- Kopplung hat mehrere Dimensionen
- Nicht alle Dimensionen können/sollten immer gleichzeitig lose gekoppelt sein

Kopplungsdimensionen

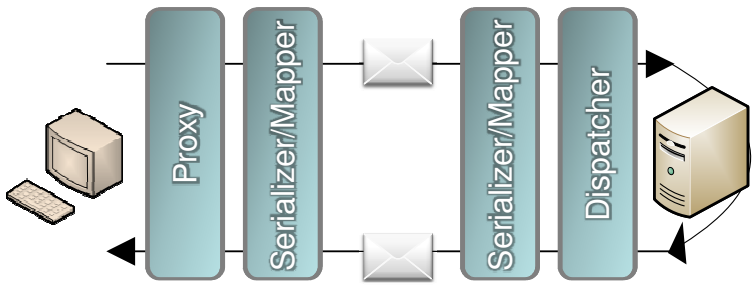
Dimension	Feste Kopplung	Lose Kopplung
Schnittstelle	Klassen und Methoden	Schema, Vertrag
Kommunikation	RPC	Nachricht, Dokument
Bindung	Fix, abgestimmt	Variabel, unabhängig
Zeit	Synchron	Asynchron
Verträge	Nicht-technische Absprache, implizit	Selbstbeschreibend, explizit

(vgl. [Carloz Perez: http://www.manageability.org/blog/stuff/loosely-coupled-dimensions])

Lose Kopplung durch

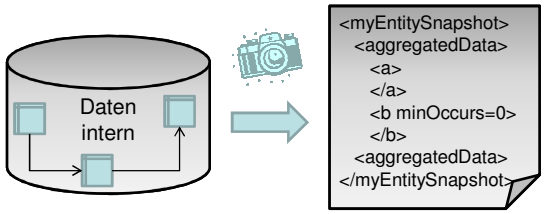
- Nachrichten-Metapher
  - XML Messaging, SOAP
- Dokumenten-basiert
  - XML, XML Schema
- Metadaten
  - Contract & Schema (WSDL & XSD)
  - Policy
- Separation of Concerns

Nachricht



Dokument

- XML, XML Schema
- Interne vs. Externe Daten



- Unabhängigkeit zwischen Contract, Schema & Verarbeitung
  - Optionale Elemente
  - Teilbäume
- Explizite Constraints

XML-Dokumente

- **Explizite Constraints**

```
<xs:simpleType name=„PLZ“>
    <xs:restriction base="xs:integer">
        <xs:pattern value="[0-9]{5}"/>
    </xs:restriction>
</xs:simpleType>
```

- **Optionale Elemente**

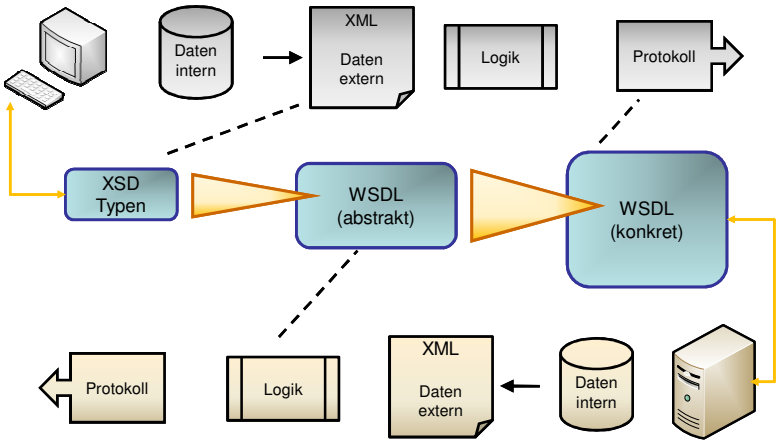
```
<xs:complexType name=„OptionalB">
  <xs:sequence>
    <xs:element name=„a" type="xs:string"/>
    <xs:element name=„b" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

- **Teilbäume**

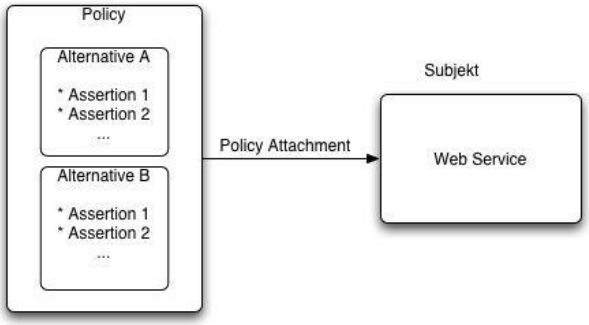
„PurchaseOrder/LineItems/Item“

„PurchaseOrder/LineItems/Item[Price>1000]“

Metadaten – Contract & Schema

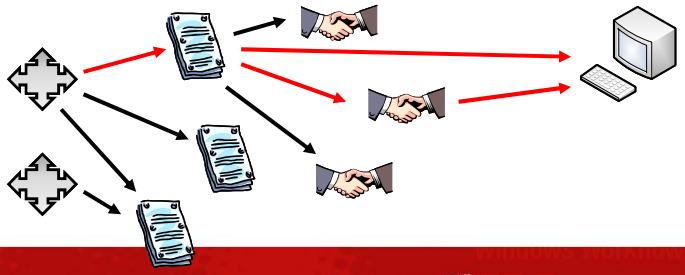


Metadaten - Policy

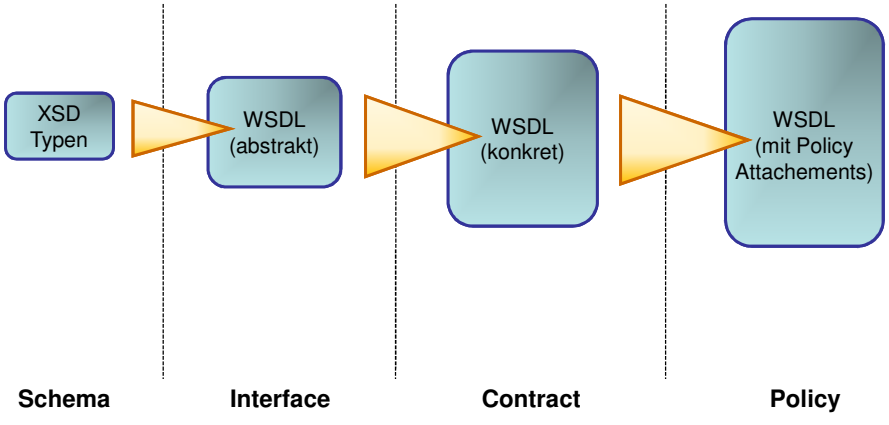


Separation of Concerns

- Beispiel: Trennung nicht-funktionaler Aspekte voneinander und von der Funktionalität
  - Trennung von Implementation & Contract
  - Trennung von Contract & Policy

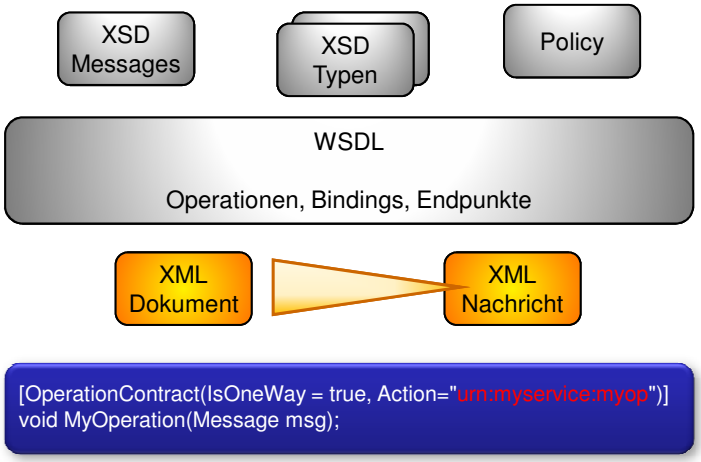


Struktur des Service-Vertrags



WCF

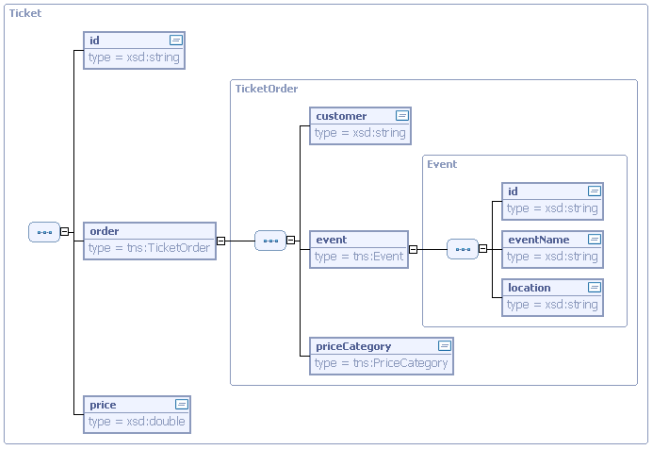
Metadaten, Nachrichten & Code



WCF Service-Modelle

- Typed Service
- Typed Message Service
- Untyped Service

Beispiel TicketService



Typed Services

Dokumente

```
[DataContract(Name = "TicketOrder")]
public class TicketOrder
{
   [DataMember(Name = "event")]
   Event requestedEvent;
   [DataMember(Name =
     "priceCategory")]
   string priceCategory;
    …
}
```

Operationen

```
[ServiceContract(Name = "TicketService")]
public interface ITicketService
{
   [OperationContract(Name =
     "OrderTicket")]
   Ticket OrderTicket(TicketOrder req);
}

public class TicketService : ITicketService
{
    …
}
```


Typed Services - Review

Pro

- Gewohntes Programmiermodell (Contract in Form von Code)
- Einfluss auf XSD/WSDL Generierung durch Attribute
- Kein XML (Vorteil?)

Contra

- DataContract
  - Fixes Mapping von XSD-Typen
  - XSD Attribute nicht unterstützt
  - Ausweg: XMLSerializer
- Implizite Message-Definition
- WSDL und XSD werden generiert
- Schlechte, komplizierte Unterstützung von WSDLs und (existierender) XSDs

Typed Message Services

Dokumente/Nachrichten

```
[DataContract(Name = "TicketOrder")]
public class TicketOrder {…}

[MessageContract]
public class OrderTicketRequest {
    [MessageBodyMember]
    public TicketOrder
            requestedTicket {
            …
    }
}
```

Operationen

```
[ServiceContract(Name = "TicketService")]
public interface ITicketService {
    [OperationContract(Name =
      "OrderTicket")]
    OrderTicketResponse
      GetBook(OrderTicketRequest req);
}
```


Typed Message Services - Review

Pro

- Gewohntes Programmiermodell (Contract in Form von Code)
- *Verbesserter* Einfluss auf XSD/WSDL Generierung durch (*Message*-)Attribute
- *Unterstützung von Messages*
- Kein XML (Vorteil?)

Contra

- DataContract
  - Fixes Mapping
  - XSD Attribute nicht unterstützt
  - Ausweg: XMLSerializer
- WSDL und XSD werden generiert
- *Mäßige, komplizierte* Unterstützung von WSDLs und schlechte Unterstützung (existierender) XSDs

Untyped Services

Dokumente/Nachrichten

```
<xs:complexType name="TicketOrder">
 <xs:sequence>
  <xs:element name="event" type="Event" />
  <xs:element name="customer„ type="xs:string" />
  <xs:element name="priceCategory"
     type="PriceCategory" />
 </xs:sequence>
</xs:complexType>
 <xs:complexType name=„OrderTicketRequest“>
    <xs:sequence>
     <xs:element name=„requestedTicket"
     minOccurs="1" maxOccurs="1" nillable="false"
     type="types:TicketOrder"/>
    </xs:sequence>
  </xs:complexType>
```

Operationen

```
[ServiceContract(Name = "TicketService")]
public interface ITicketService {
   [OperationContract(Name = "OrderTicket")]
   Message OrderTicket(Message request);
}
```


Untyped Services - Review

Pro

- Volle Unterstützung von XML
  - dadurch alle Vorteile
- Direkter Zugriff auf Dokument
- Reines Messaging
- Lose Kopplung

Contra

- Untyped im wahrsten Sinne des Wortes
- Kein Schema bzw. <xsd:any>
- Keine Validierung
- XML Know-how notwendig (Nachteil ?)

➢ Eigentlich das „Service-Orientierteste“ Modell …

XML Messaging mit WCF

- Untyped Services
- Message
  - XSD & WSDL
  - XML & XPath

Request auswerten

```
XPathNavigator nav = new
    XPathDocument(request.GetReaderAtBodyContents()).CreateNavigator();
XmlNamespaceManager nmMan = new
    XmlNamespaceManager(nav.NameTable);
nmMan.AddNamespace("m", "http://.../ns/xsd/TicketService/2007-02");

string customer =
    nav.SelectSingleNode("/m:OrderTicket/m:request/m:customer",
        nmMan).Value;
string priceCategory =
    nav.SelectSingleNode("/m:OrderTicket/m:request/t:priceCategory",
        nmMan).Value;
```

Response erzeugen

```
private const string ORDER_TICKET_RESPONSE =
  @"<OrderTicketResponse xmlns=""" + NAMESPACE + @""">
        <responseTicket xmlns=""" + NAMESPACE + @""">
          <id xmlns=""" + NAMESPACE + @""">{0}</id>
            …
        </response>
      </OrderTicketResponse>";
StringBuilder contentString = new StringBuilder();
using (StringWriter writer = new StringWriter(contentString)) {
  writer.WriteLine(ORDER_TICKET_RESPONSE, Guid.NewGuid(), ticketOrder, price);
}
XmlReader content = XmlReader.Create(new StringReader(contentString.ToString()));
return Message.CreateMessage(MessageVersion.Soap11,
  http://.../ns/wsdl/TicketService/200 02/OrderTicketResponseAction, content);
```

XML Template

Template instanziieren

Schema-Validierung mit WCF

- Generische, parametrierbare Schema-Validierung
  - Message Schema als Konfigurationsparameter
- WCF Endpoint Behavior
- XmlReader & XmlReaderSettings

XsdValidationInspector

```
public object AfterReceiveRequest(ref Message request, …) {
  MessageBuffer buffer = request.CreateBufferedCopy(int.MaxValue);
  Message temp = buffer.CreateMessage();

  XmlDocument bodyDoc = new XmlDocument();
  bodyDoc.Load(temp.GetReaderAtBodyContents());
  XmlReader r = XmlReader.Create(new XmlNodeReader(bodyDoc),
    ReaderSettings);
  while (r.Read()) ; // nur validieren

  request = buffer.CreateMessage();
}
```


Und wo sind XSD & WSDL?

- Registory! (Registry/Repository)
  - Umdenken von einer isolierten IDE hin zu einer unternehmensweiten Umgebung
  - Microsoft? ☹
  - Open Source? ☹
  - Kommerzielle Registries
    - Systinet Registry (HP)
    - CentraSite (Software AG)

Fazit

- Entwickeln Sie Services, die den Service-orientierten Prinzipien genügen
- Setzen Sie sich mit XML, XSD, WSDL auseinander
- Setzen Sie auf den doc/schema-based Ansatz
- Schwächen von WCF lassen sich ausgleichen
  - WCF Extensions (Behaviors)

Danke!

Hartmut Wilms

innoQ Deutschland GmbH
Halskestraße 17
D-40880 Ratingen
Tel +49 2102 77 1620
Fax +49 2102 77 1601

http://www.innoq.com
http://www.innoq.com/blog/hw

Links

- Don Box, A Guide to Developing and Running Connected Systems with Indigo
  http://msdn.microsoft.com/msdnmag/issues/04/01/Indigo/?topics=/msdnmag/issues/04/01/Indigo
- Carloz Perez, Loosely Coupled Dimensions
  http://www.manageability.org/blog/stuff/loosely-coupled-dimensions
- Arjen Poutsma, XSLT that transforms from XSD to WSDL
  http://blog.springframework.com/arjen/archives/2006/07/27/xslt-that-transforms-from-xsd-to-wsdl/
- Aaron Skonnard, Contract-First Service Development
  http://msdn.microsoft.com/msdnmag/issues/05/05/ServiceStation/
  http://msdn.microsoft.com/msdnmag/issues/05/06/ServiceStation/
- Anne Thomas Manes, The "wrapped" document/literal convention
  http://atmanes.blogspot.com/2005/03/wrapped-documentliteral-convention.html
- AltovaXML
  http://www.altova.com/de/altovaxml.html