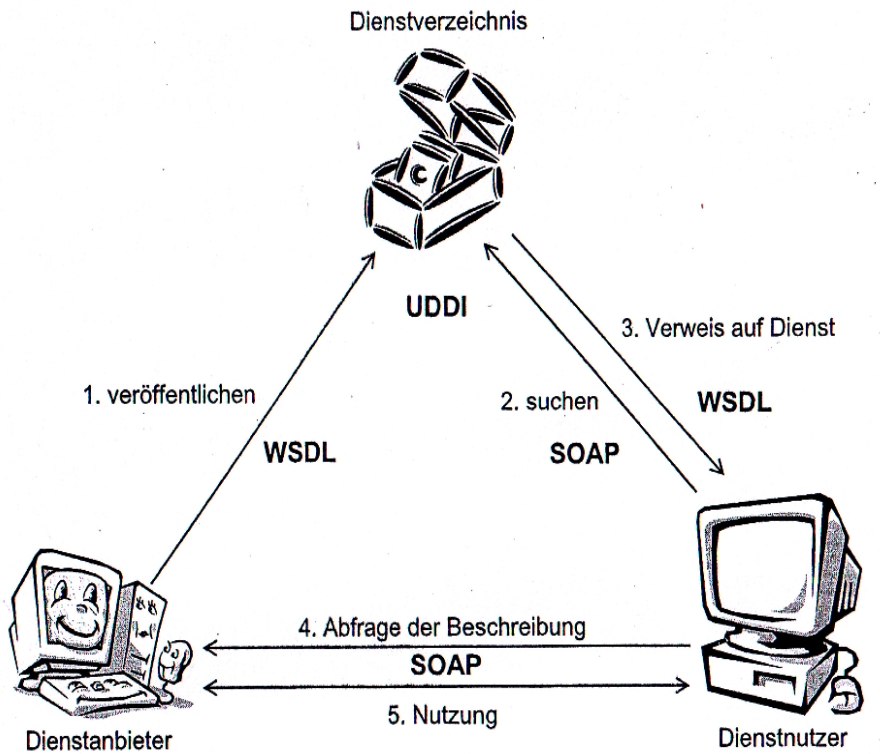
Informatik-Seminar WS 06/07

Service-orientierte Architektur (SOA)

Thema 2

WS-Policy, WS-Addressing, WS-ReliableMessaging: Zielsetzungen, Zusammenhang, Lösungen und Beispiele

Web Services Spezifikationen - Überblick und Aufbau

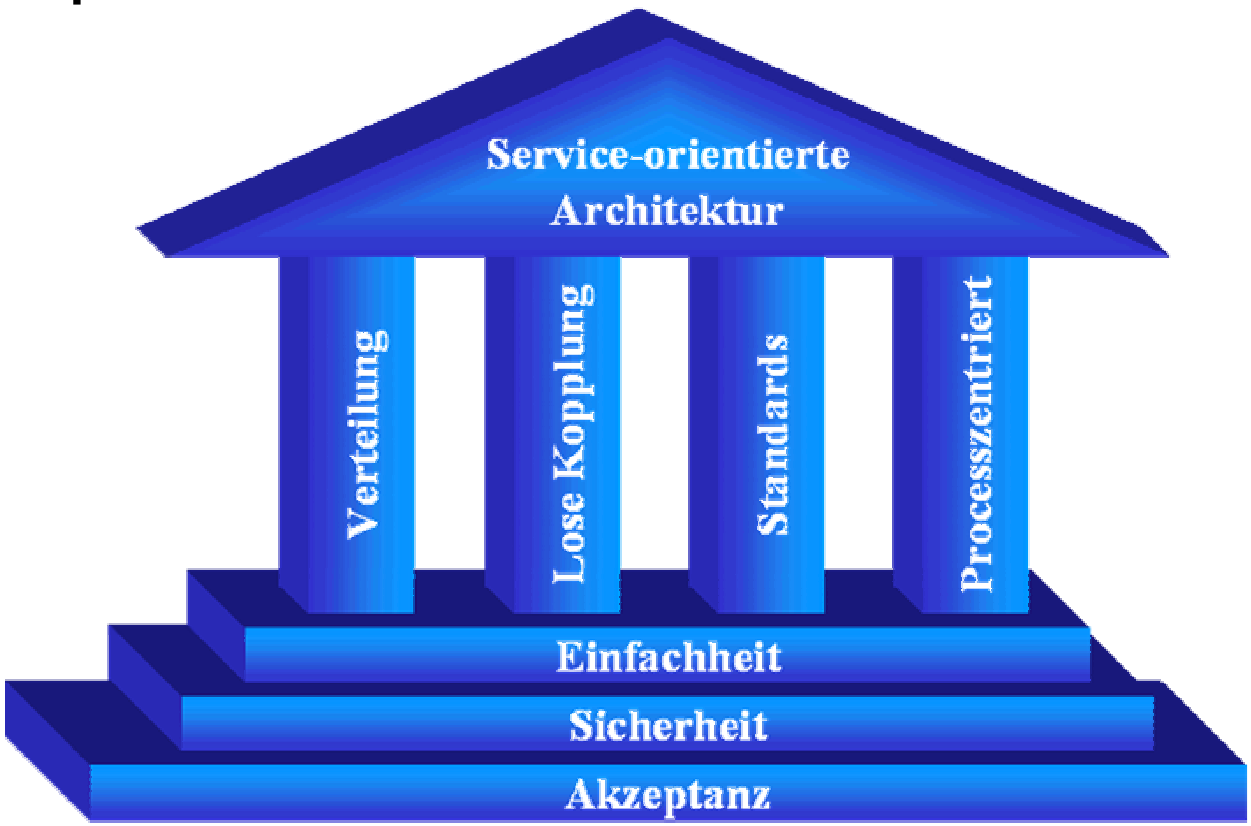


Quelle: Dostal

- WSDL,SOAP, UDII Basiskomponenten einer XML basierten SOA

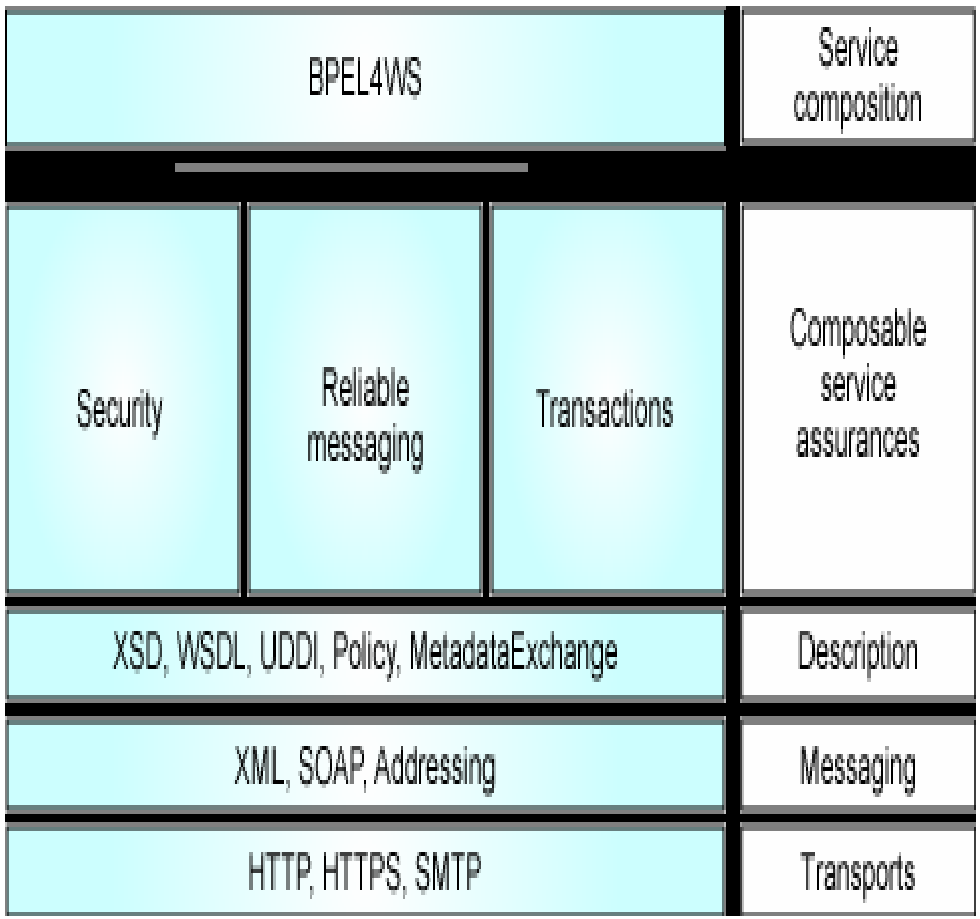
Anforderungen an eine vollwärtige SOA Implementierung

SOA - Tempel



Quelle:
Jeckle.de

WS-* Spezifikationen



- Ziele: Anforderungen einer SOA entsprechen
- Web Services sicherer u. zuverlässiger machen
- Transaktionen besser unterstützen

Quelle: ibm.com

Komponierbarkeit von WS-Spezifikationen

Quelle:Ibm.com

- SOAP unterstützt Komponierbarkeit durch Header-Block
- WSDL in Kombination mit WS-Policy unterstüzt Komponierbarkeit von WS-Beschreibungen

WS-Addressing – Problemstellung und Zielsetzung

```
POST /InStock HTTP/1.1
Host: www.stock.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn
SOAPAction: "www.stock.org/GetStockPrice"

<?xml version="1.0"?>
<soap:Envelope
        xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
        soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

        <soap:Body xmlns:m="http://www.stock.org/stock">
                <m:GetStockPrice>
                        <m:StockName>IBM</m:StockName>
                </m:GetStockPrice>
        </soap:Body>

</soap:Envelope>
```

- Addressierungs-informationen in Transportschicht
- Transport-protokoll schränkt Funktionsumfang ein

Problemstellungen durch HTTP-Binding

- Verwendung anderer Protokolle, wie SMPT wichtig. Keine Interaktionen mit unterschiedlichen Protokollen möglich
- Asynchroner Nachrichtenaustausch lässt sich nicht realisieren
- Prozessedauer oft lange und zustandabhängig – permanente Verbindung zu einen Service aber nicht zumutbar
- Die Informationen über Quelle und Ziel einer Nachricht können verloren gehen, z.B. durch Firewalls

Notwendige Adressinformationen

- Absender
- Empfänger
- Instanz einer Adresse
- Adresse, wenn ein Fehler passiert

WS-Addressing Architektur: Zwei Grundelegende Konzepte

- Endpoint Refernce (EPR)
  - Endpoint: Jede Quelle oder Ziel für Web Services Nachricht
  - EPR: Datenstruktur um Endpoint zur Laufzeit zu erreichen
- Message Information Headers
  - Benutzen EPRs
  - Adressierungs Informationen:
    - Absender
    - Empfänger
    - Adresse für Antwort
    - Adresse bei Fehler

Eingenschaften von Endpoint References

- Können dynamisch erstellt werden
- Unterstützen Adressierung von spzifischen Instanzen eines Services
- WSDL verfügt nicht über diese Eigenschaften

Elemente von Endpoint References

- **address**: URI des Endpoints
- **reference properties:** Werte, die verbunden sind mit einer Instanz des Webservices
- **reference parameters:** Parameter, die die Interaktion mit dem Endpoint/Instanz unterstützen können
- **selected port type**: Äquivalent zu WSDL Port Type/Interface, der eine Beschreibung der WS Aktionen enthält
- **service-port:** WSDL Service Element, das den Endpoint indentifiziert
- **policy:** Policy für den Endpoint

Beispiel EPR

- **<wsa:EndpointReference** xmlns:c="http://example.org/claims"
- xmlns:p="http://schemas.xmlsoap.org/ws/2002/12/policy"
- >
- <**wsa:Address>**http://claimserver/ins/p.asmx</wsa:Address>
- <**wsa:ReferenceProperties**>
- <c:PatientProfile>123456</c:PatientProfile>
- <c:CarrierID>987654</c:CarrierID>
- **</wsa:ReferenceProperties**>
- <**wsa:PortType**>c:ClaimsPortType</wsa:PortType>
- <**wsa:ServiceName** PortName="c:ClaimsSoapPort">c:ClaimsService
- </wsa:ServiceName>
- <**p:Policy**>
- ... <!-- policy statement omitted for brevity -->
- **</p:Policy**>
- </wsa:EndpointReference>

Elemente der Message Information Headers

- To: URI der Zieladresse
- Action:
  - semantische Bedeutung der Nachricht.
  - URI identifiziert output,input message eines WSDL PortTyps.
  - Action URI wird explizit ermittelt durch „wsa:action“Attribut inner eines WSDL Port Typs
  - oder implizit durch ableiten Targetnamspace von WSDL Dokument, Port Type Name und input/output Name durch verbinden der Namen

Optionale Elemente der MI Header

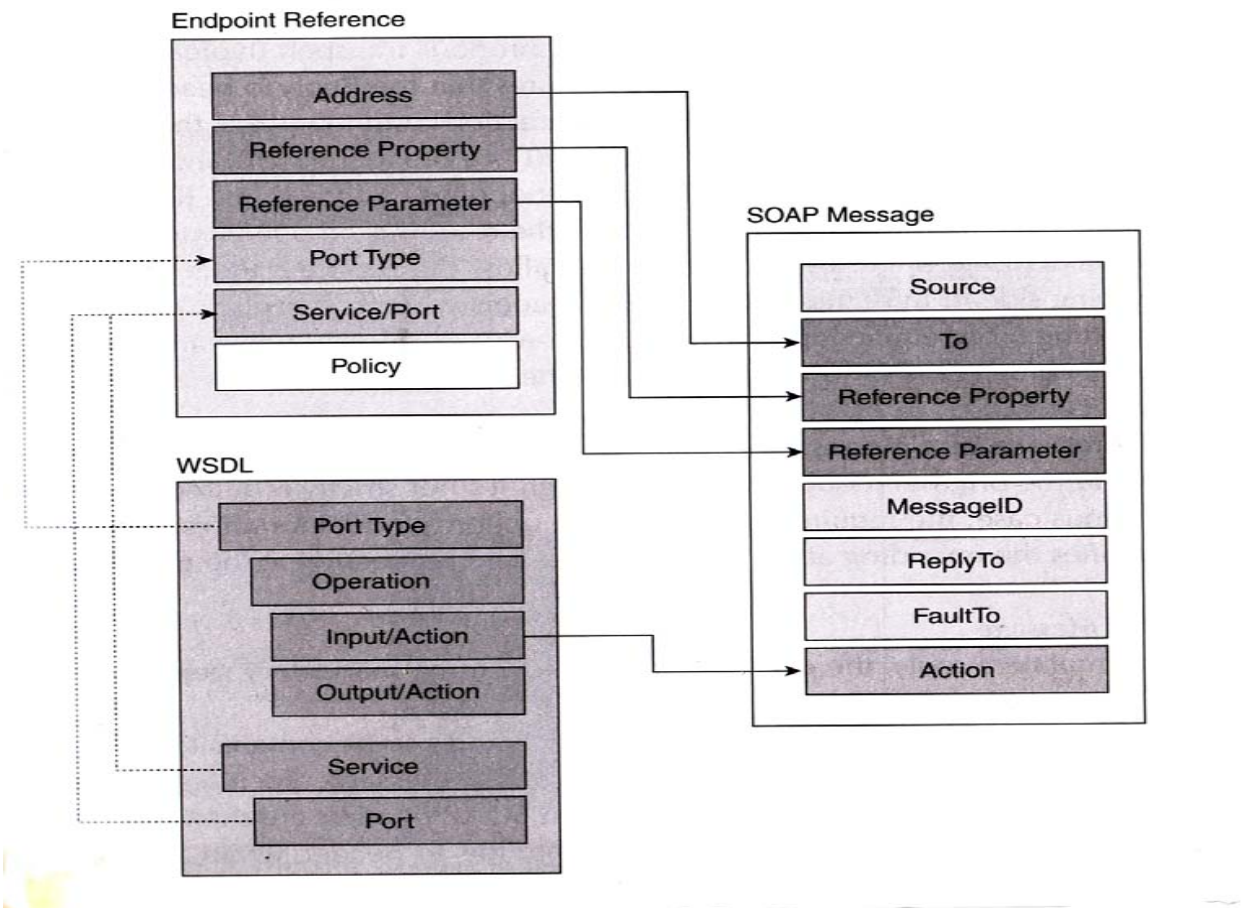
- **ReplyTo**:
  - EPR des Endpoints, wohin die Antworten geschickt werden sollen.
  - Erforderlich, wenn Antwort erwartet wird

- **From**: EPR der Ursprungsquelle
- **FaultTo**: EPR, wohin die fault Messages gesendet werden
- **messageID**:URI, die eine Nachricht eindeutig identifiziert
  - Erforderlich, wenn ReplyTo oder FaultTo Header verwendet wird
- **RelatesTo**: stellt Beziehung zwischen Nachrichten her
  - Gibt art der Beziehung durch Attribut an. Standard wsa:reply
  - Enhält messageID, von der Nachricht auf die sie sich bezieht
  - Erfoderlich, wenn es eine Antwort ist

Beispiel: SOPA Request Message

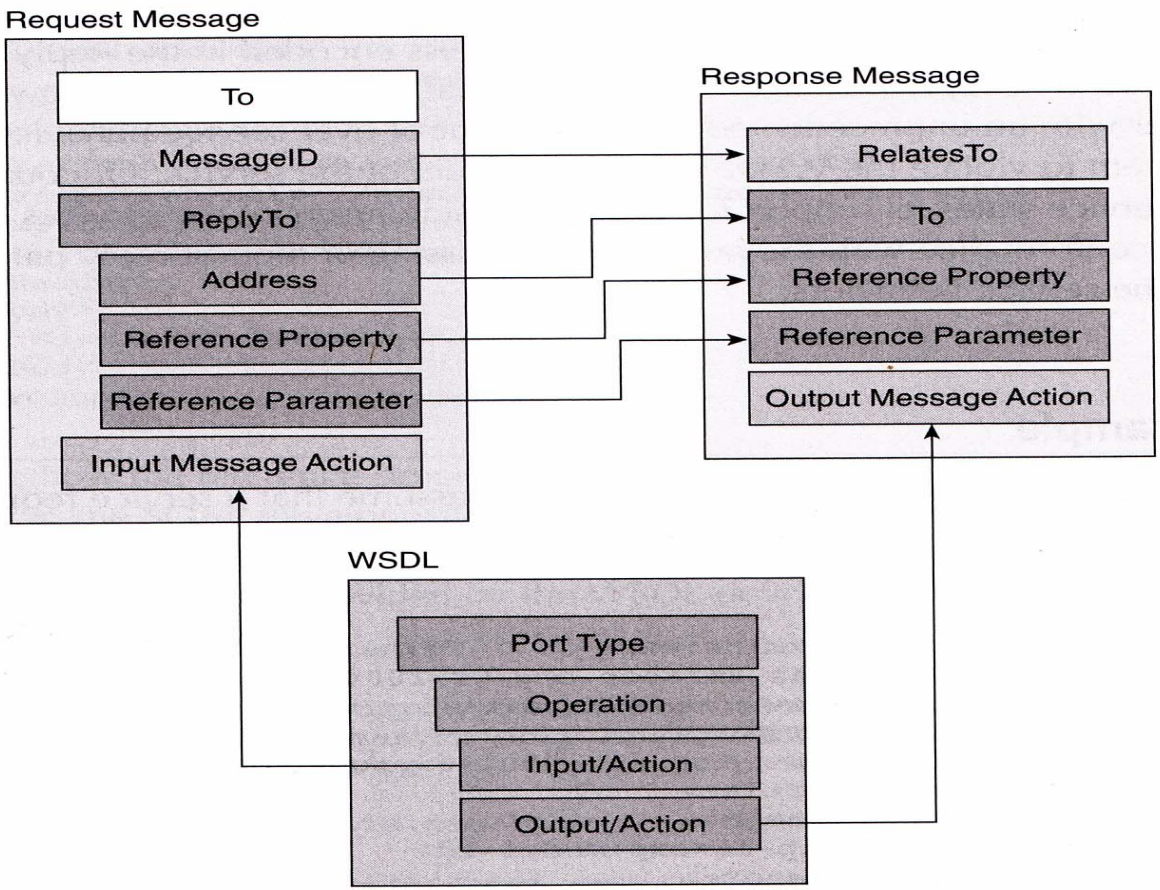
```
<s:Envelope xmlns:s="..." xmlns:wsa="...">
<s:Header>
  <was:MessageID>uuid:someid</wsa:MessageID>
  <wsa:Action>http://skonnard.com/SubmitClaim</wsa:Action>
  <wsa:To>http://skonnard.com/Claims/Submit.asmx</wsa:To>
  <wsa:From>
    <wsa:Address>http://skonnard.com/main/sub.asmx</wsa:Address>
     <wsa:ReferenceProperties>
       <c:PatientProfile>123456</c:PatientProfile>
       <c:CarrierID>987654</c:CarrierID>
     </wsa:ReferenceProperties>
   </wsa:From>
   <wsa:ReplyTo>
      <wsa:Address>http://skonnard.com/resp/resp.asmx</wsa:Address>
      <wsa:ReferenceProperties>
           <c:PatientProfile>123456</c:PatientProfile>
           <c:CarrierID>987654</c:CarrierID>
      </wsa:ReferenceProperties>
    </wsa:ReplyTo>
   <wsa:FaultTo>
       <wsa:Address>http://skonnard.com/fault/err.asmx</wsa:Address>
        <wsa:ReferenceProperties>
          <c:PatientProfile>123456</c:PatientProfile>
          <c:CarrierID>987654</c:CarrierID>
        </wsa:ReferenceProperties>
    </wsa:FaultTo>
</s:Header>
<s:Body xmlns:c="http://example.org/claims">
<c:SubmitClaim> ... </c:SubmitClaim>
```

Einbinden von EPRs in MI Headers



Quelle: Leymann u.a

MI-Headers in einer Response Message



Quelle:Leymann u.a

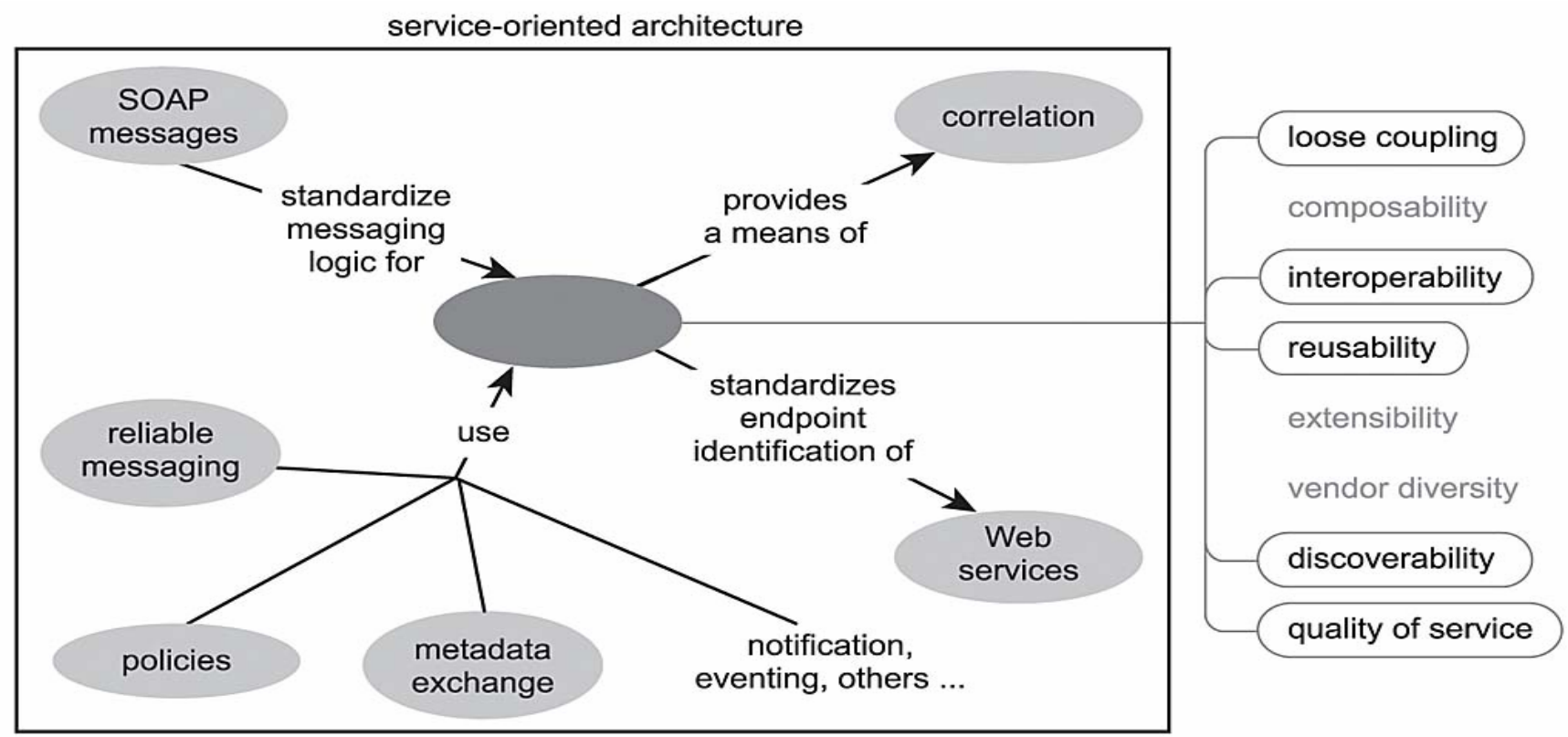
Reply Message für das Beispiel

```
....
<was:MessageID>uuid:someid</wsa:MessageID>
...
<wsa:ReplyTo>
      <wsa:Address>http://skonnard.com/resp/resp.asmx</wsa:Address>
      <wsa:ReferenceProperties>
         <c:PatientProfile>123456</c:PatientProfile>
         <c:CarrierID>987654</c:CarrierID>
      </wsa:ReferenceProperties>
</wsa:ReplyTo>
...
```

Die Reply Nachricht:

```
<s:Envelope xmlns:s="..." xmlns:wsa="..."
xmlns:c="http://example.org/claims">
  <s:Header>
    <wsa:RelatesTo RelationshipType="Reply">uuid:someid</wsa:RelatesTo>
    <wsa:To>http://skonnard.com/resp/resp.asmx</wsa:To>
    <c:PatientProfile>123456</c:PatientProfile>
    <c:CarrierID>987654</c:CarrierID>
    <wsa:Action>http://skonnard.com/SubmitClaimResponse</wsa:Action>
  </s:Header>
  <s:Body xmlns:c="http://example.org/claims">
    <c:SubmitClaim> ... </c:SubmitClaim>
  </s:Body>
</s:Envelope>
```

WS-Addresssing und SOA

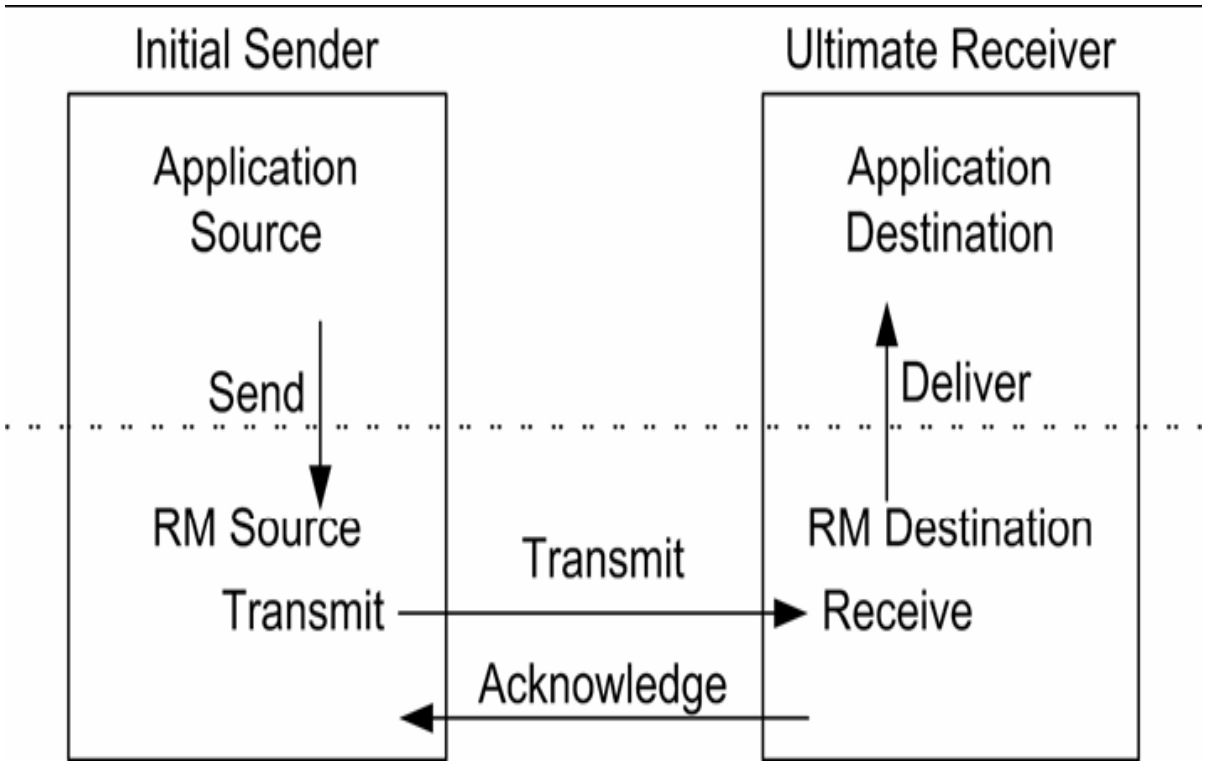


Quelle: T.Erl

WS-ReliableMessaging-Problemstellung und Zielsetzung

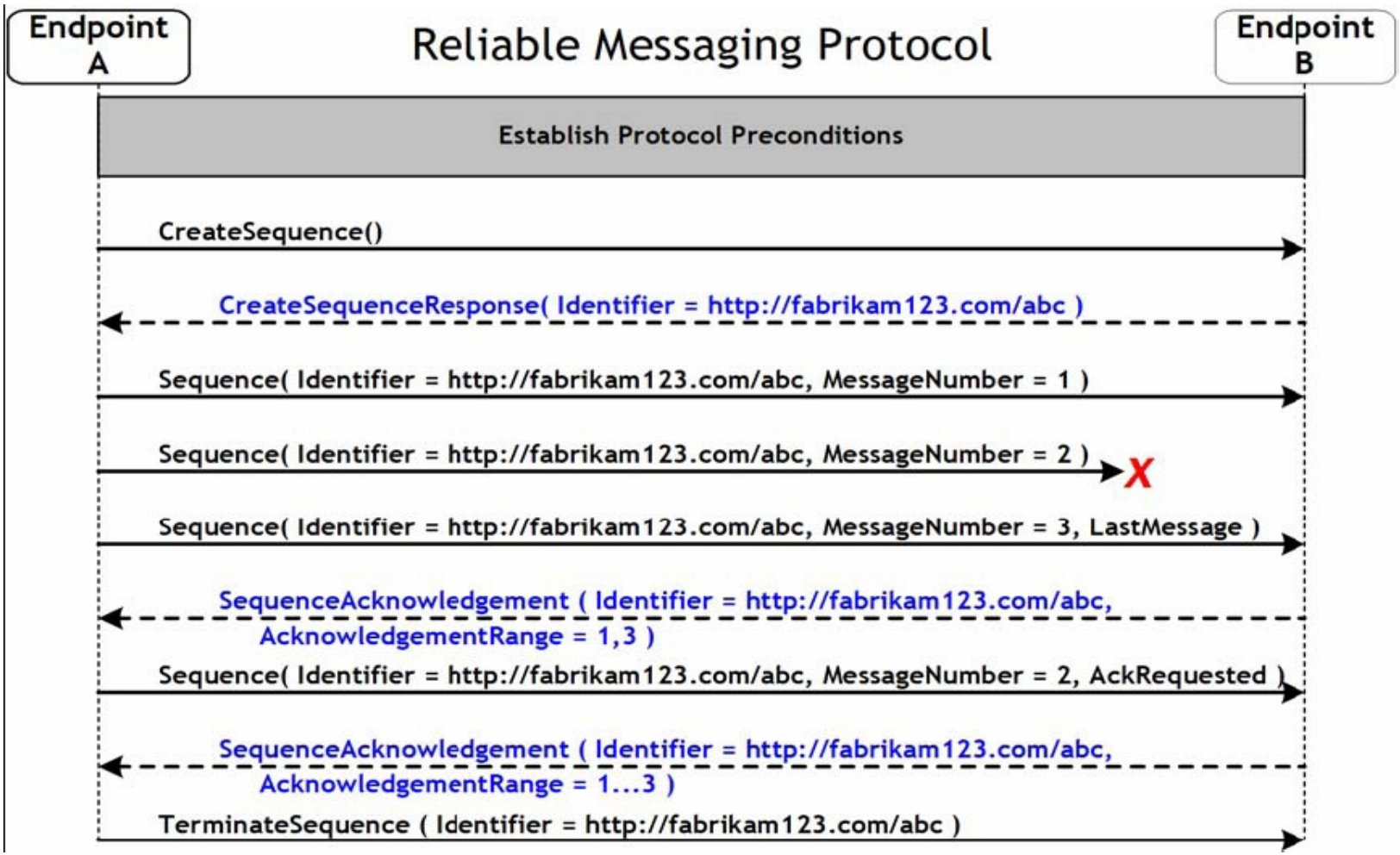
- Loose Kopplung von Services: Keine Kontrolle über aktuellen Kommunikationsprozess
- Web Services wissen nicht, ob gesendete Nachrichten empfangen,In der richigen Reihenfolge empfangen,oder Nachrichten dupliziert wurden
- Fehler sind extrem schwierig zu erkennen
- Systeme, die miteinander kommunizieren, haben unterschiedliche Infrastrukturen, Problem der Interoperabilität
- RM führt Protokoll zur zuverlässigen Nachrichtenübertragung ein
- Überbrückt unteschiedliche Infrastrukturen (Netzwerke etc), indem es eine logische Ende-zu-Ende Verbindung zw. Web Services herstellt

Reliable Messaging Modell



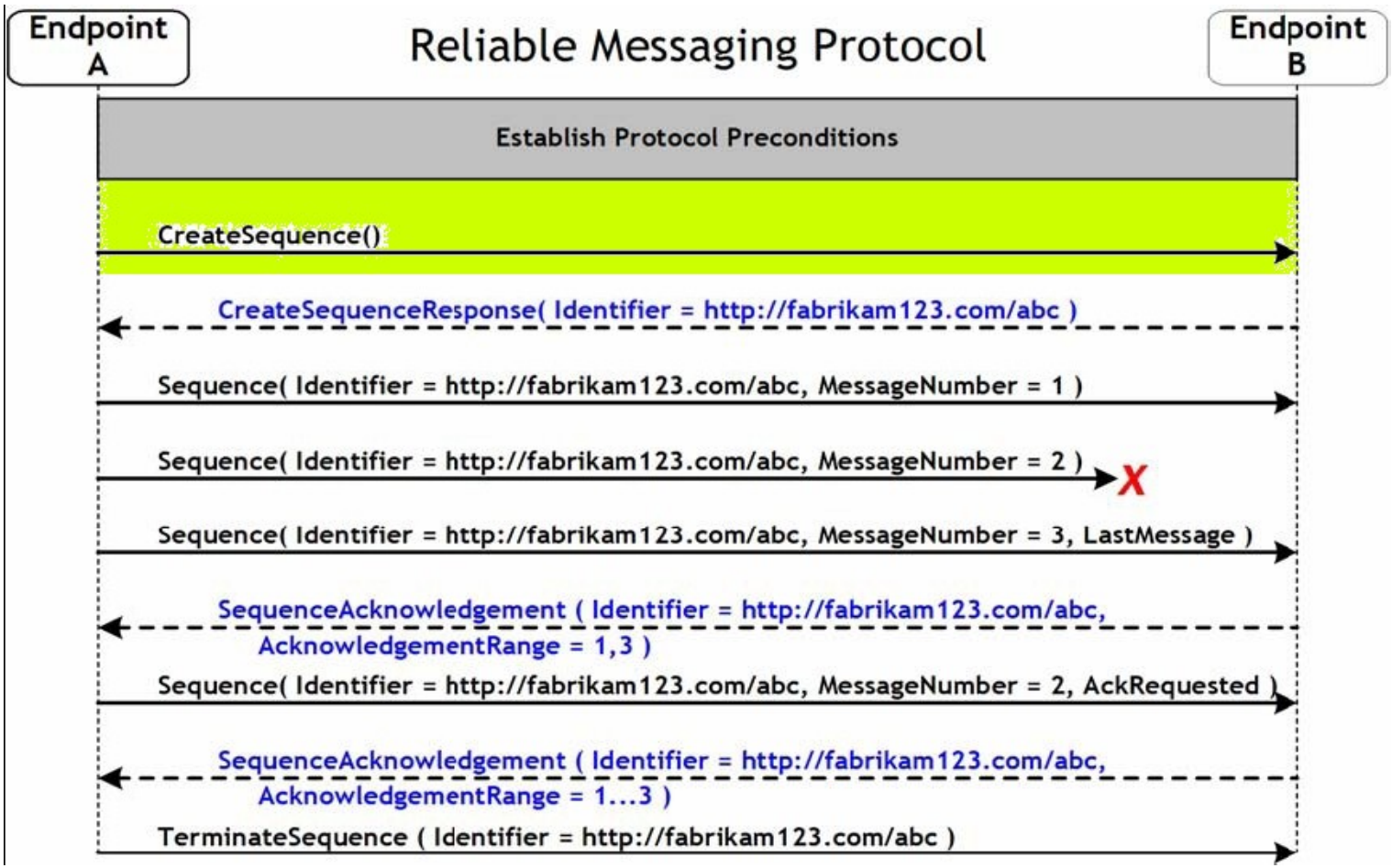
- 4 Arten von Zusicherungen:
  - **AtLeastOnce**
  - **AtMostOnce**
  - **ExactlyOnce**
  - **InOrder**

Ablauf des RM Protokolls



Quelle: WS-RM-Spezifikation

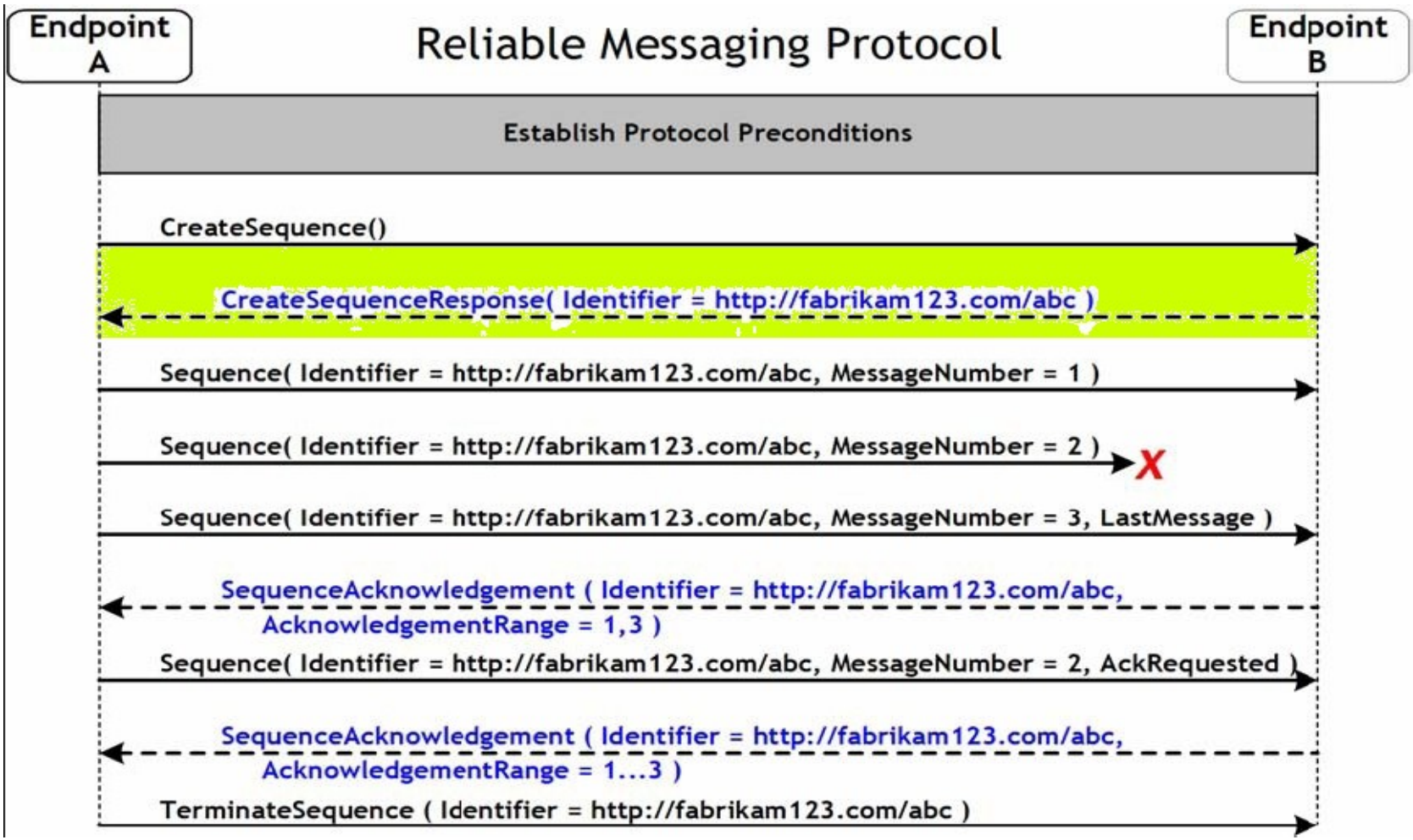
CreateSequence



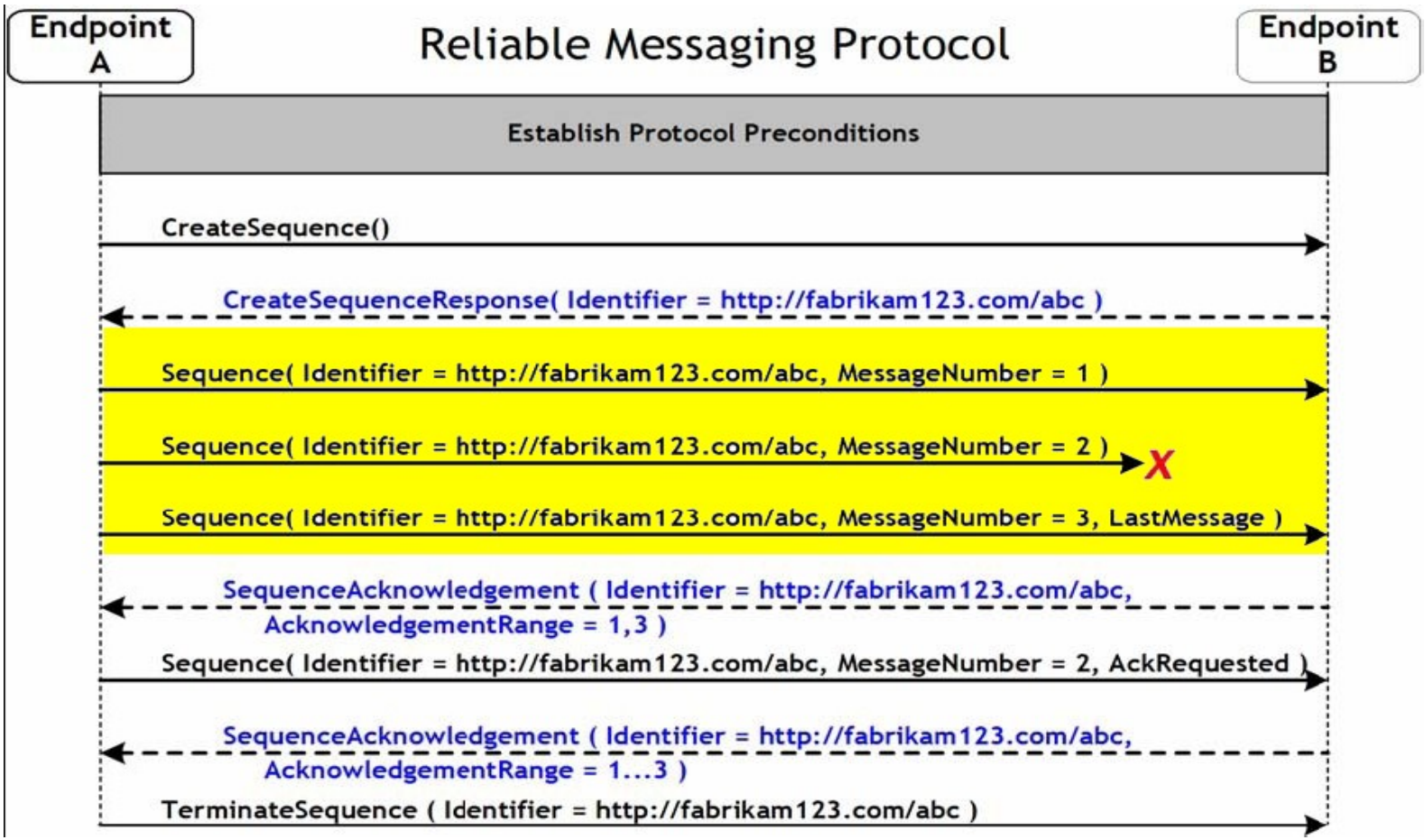
CreateSequence Message

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsrm="http://schemas.xmlsoap.org/ws/2005/02/rm"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
<S:Header>
  <wsa:MessageID>
   http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
  </wsa:MessageID>
  <wsa:To>http://fabrikam123.com/serviceB/123</wsa:To>
  <wsa:Action>http://schemas.xmlsoap.org/ws/2005/02/rm/CreateSequence
     </wsa:Action>
  <wsa:ReplyTo>
  <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
  </wsa:ReplyTo>
</S:Header>
<S:Body>
  <wsrm:CreateSequence>
   <wsrm:AcksTo>
     <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
   </wsrm:AcksTo>
</wsrm:CreateSequence>
</S:Body>
</S:Envelope>
```

CreateSequenceResponse



Drei Nachrichten werden über die Sequenz verschickt...



Nachricht 1

```
<S:Header>
  <wsa:MessageID>
   http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfcbc9e
  </wsa:MessageID>
  <wsa:To>http://fabrikam123.com/serviceB/123</wsa:To>
  <wsa:From>
  <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
  </wsa:From>
  <wsa:Action>http://fabrikam123.com/serviceB/123/request</wsa:Action>
  <wsrm:Sequence>
     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
     <wsrm:MessageNumber>1</wsrm:MessageNumber>
  </wsrm:Sequence>
</S:Header>
<S:Body>
<!-- Some Application Data -->
</S:Body>
```

Nachricht 2

...
<wsa:MessageID>
http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
</wsa:MessageID>
<wsrm:Sequence>
<wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
<wsrm:MessageNumber>2</wsrm:MessageNumber>
</wsrm:Sequence>
....

Nachricht 3

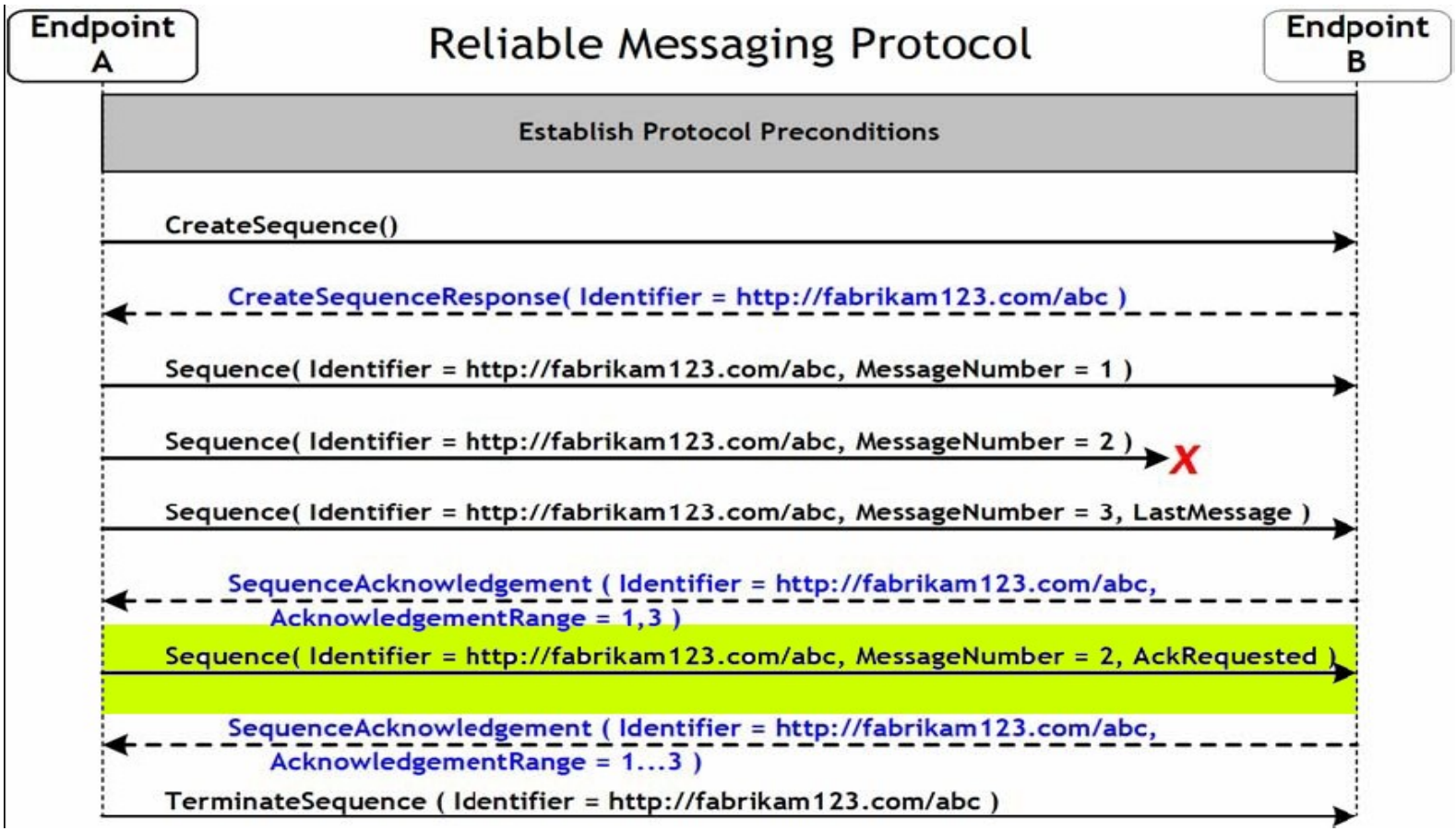
...
<wsa:MessageID>
http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546819
</wsa:MessageID>
<wsrm:Sequence>
<wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
<wsrm:MessageNumber>3</wsrm:MessageNumber>
<wsrm:LastMessage/>
</wsrm:Sequence>
....

Die 2 Nachricht ging verloren...

SequenceAcknowledgment

<S:Header>
<wsa:MessageID>
http://fabrikam123.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
</wsa:MessageID>
<wsa:To>http://Business456.com/serviceA/789</wsa:To>
<wsa:From>
<wsa:Address>http://fabrikam123.com/serviceB/123</wsa:Address>
</wsa:From>
<wsa:Action>
http://schemas.xmlsoap.org/ws/2005/02/rm/SequenceAcknowledgement
</wsa:Action>
<wsrm:SequenceAcknowledgement>
<wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
<wsrm:AcknowledgementRange Upper="1" Lower="1"/>
<wsrm:AcknowledgementRange Upper="3" Lower="3"/>
</wsrm:SequenceAcknowledgement>
</S:Header>
<S:Body/>

Die 2 Nachricht wird erneut gesendet und eine Bestätigung gefordert

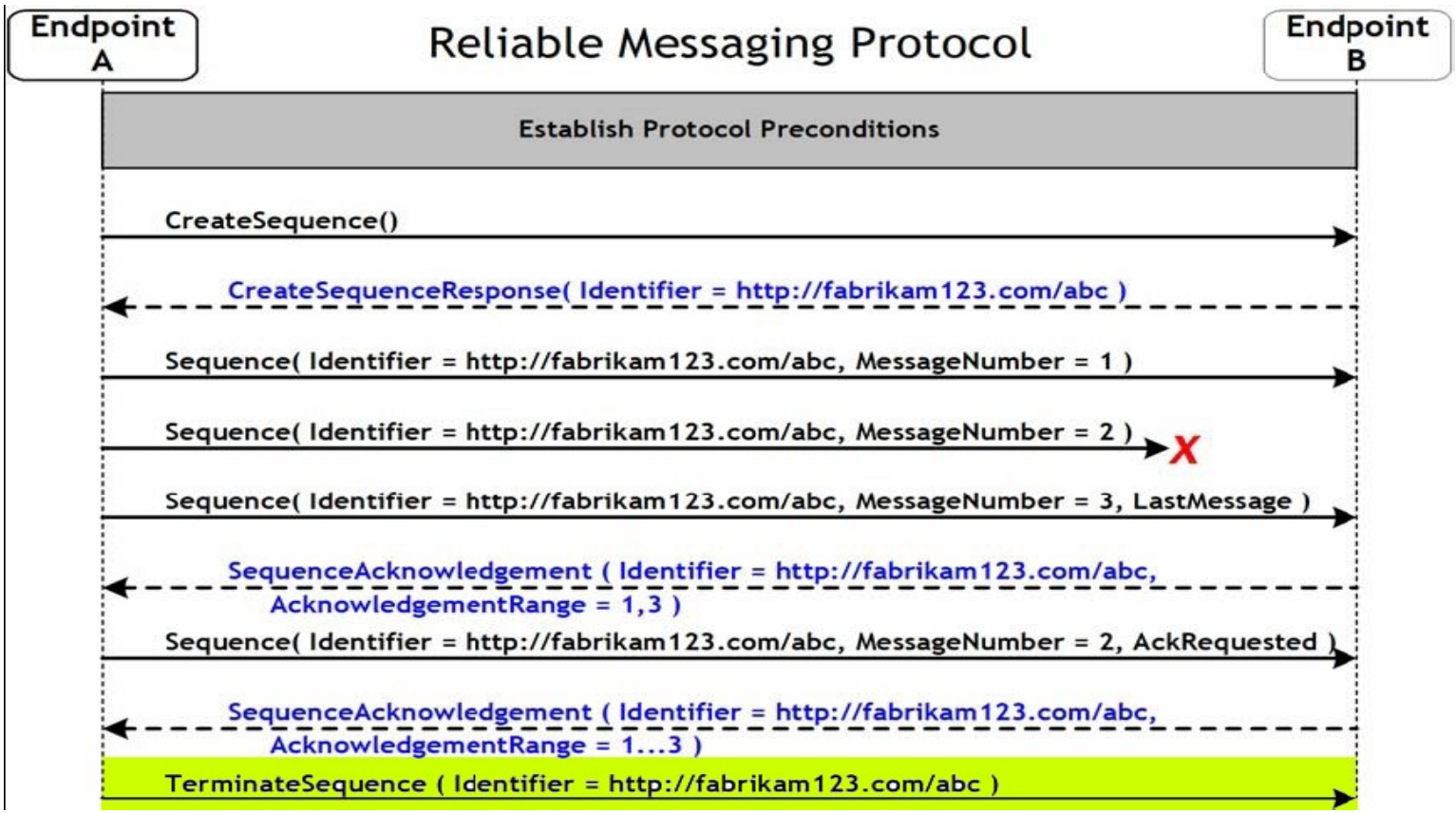


AckRequested

```
<S:Header>
  <wsa:MessageID>
     http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
   </wsa:MessageID>
   <wsa:To>http://fabrikam123.com/serviceB/123</wsa:To>
   <wsa:From>
     <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
   </wsa:From>
   <wsa:Action>http://fabrikam123.com/serviceB/123/request</wsa:Action>
   <wsrm:Sequence>
     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
     <wsrm:MessageNumber>2</wsrm:MessageNumber>
   </wsrm:Sequence>
   <wsrm:AckRequested>
      <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
   </wsrm:AckRequested>
</S:Header>
<S:Body>
<!-- Some Application Data -->
</S:Body>
</S:Envelope>
```

Empfang der Nachricht wird bestätigt...

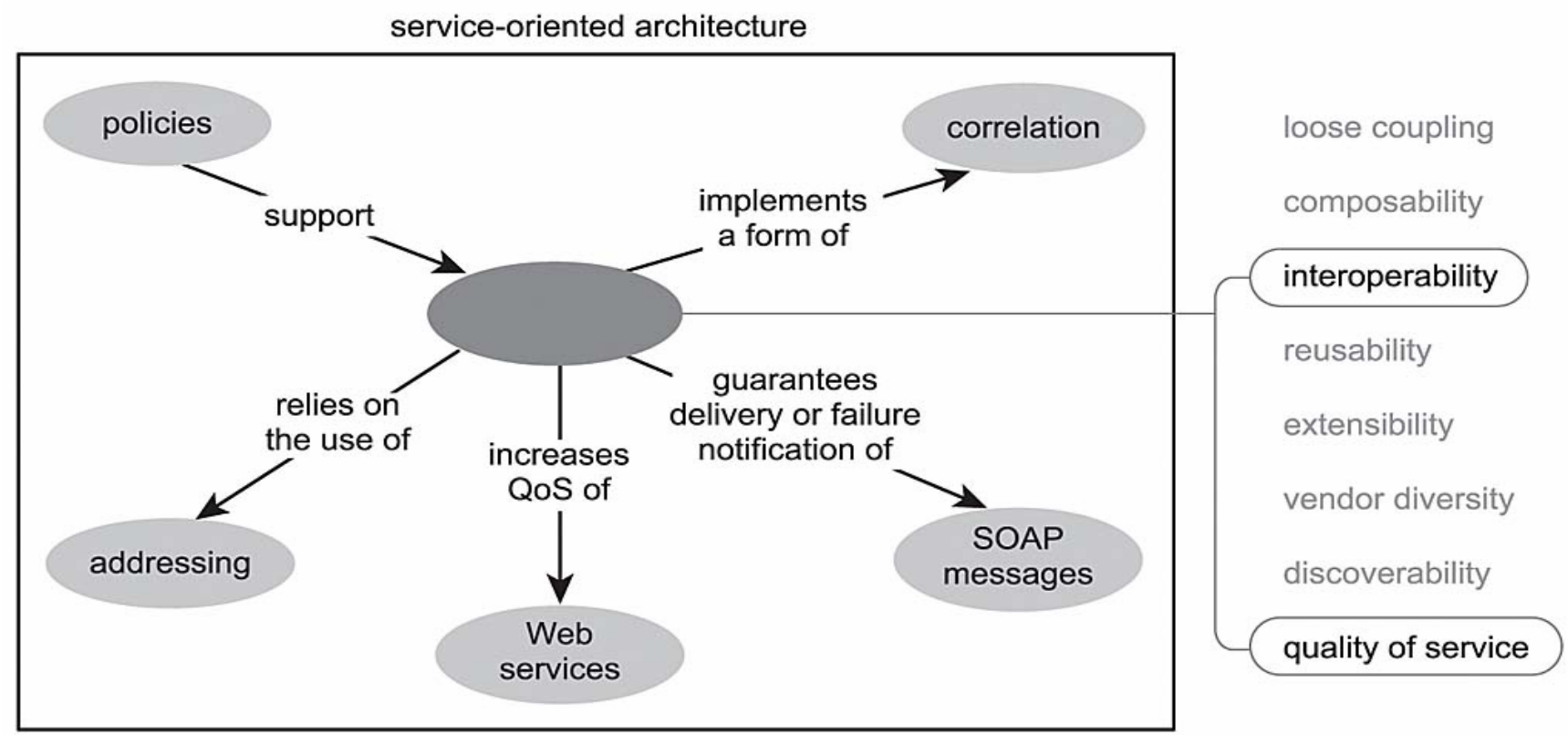
Die Sequenz wird beendet...



TerminateSequence

```
...
<wsa:Action>
    http://schemas.xmlsoap.org/ws/2005/02/rm/TerminateSequence
</wsa:Action>
</S:Header>
<S:Body>
  <wsrm:TerminateSequence>
     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
  </wsrm:TerminateSequence>
</S:Body>
```

WS-RM, WS-Addressing und WS-Policy



Quelle: T.Erl

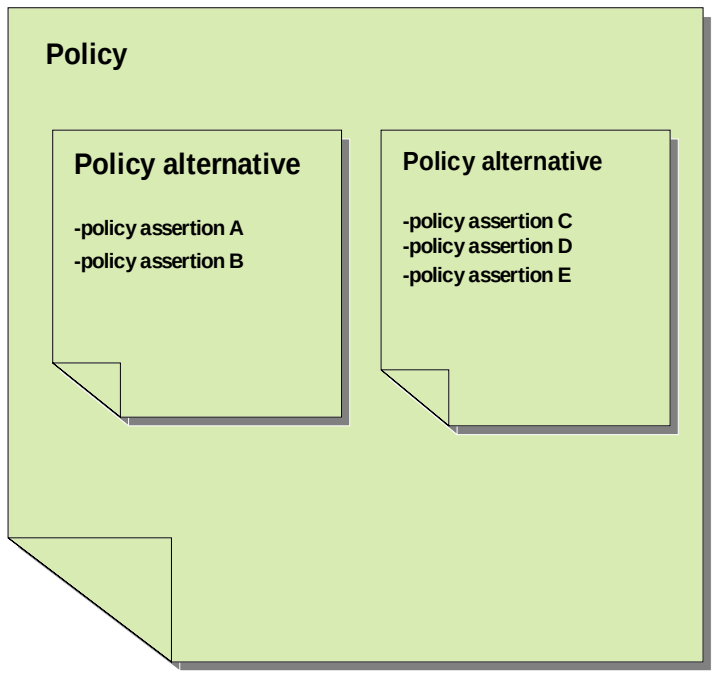
WS-Policy

- Policys sind Metadaten, die Anforderungen und Fähigkeiten einer WS Einheit beschreiben
- 3 wichtige Anwendungsszenarien:
  - Entwicklung und Anwendung interoperabler Web Services
  - Finden und Auswahl geeigneter Services
  - Dynamische Aktualisierung

WS-Policy Framework

- WS-Policy
  - Modell und Sprache um Policys zu beschreiben
- WS-PolicyAttatchment
  - Mechanismen um Policys an Web Services Einheiten zu binden

Policy Modell:



Policy Assertions

- Assertions sind Metadaten, die ein spezifisches Verhalten für einen speziellen Bereiche identifizieren
- Assertions werden von Autoren einer Domain definiert und veröffentlicht
- Für viele WS-*Spezifikationen sind Policy Assertions definiert z.B. WS-Security Policy, WS-RM Policy
- Der Assertion Typ wird durch Namespace Namen und lokalen Namen eindeutig identifiziert
- Assertions können Attribute, Parameter und selber Assertions besitzen, die Ihr Verhalten spezifizieren

WS RM-Policy

- **RMAssertion**-Element fordert den Einsatz des RM Protokolls
- Weitere Parameter spezifizieren das Protokollverhalten:
  - **Inactivity:** Ist ein Parameter, der einen Zeitraum angibt. Erhält ein Endpoint während dieses Zeitraums keine Nachricht, kann er annehmen, dass die Sequenz infolge von Inaktivität beendet wurde.
  - **BaseRetransmissionInterval:** Gibt einen Parameter für ein Zeitraum an. Erhält die RM Source keine Bestätigug innerhalb des Zeitraumes für eine verschickte Nachricht, wird sie erneut gesendet.
  - **ExponetialBackoff:** Durch Angabe dieses Parameters wird der BaseRetransmissionInterval Parameter angepasst durch den Backoff Algorithmus von Tanenbaum
  - **AckowledgmentInterval:** Gibt ein Zeitintervall an, in dem die RM Destination eine Acknowledgment Message an die RM Source sendet.
  -

Policy Expressions

- **Policy**: Container für Assertions und kombinierte Assertions
- **ALL/Policy**: Alle mit ALL kombinierten Assertions sind erforderlich

```
<wsp:Policy>
   xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
   xmlns:wsrm="http://schemas.xmlsoap.org/ws/2005/02/rm/policy"
   xmlns:wsap_“http://www.w3.org/2006/05/wsdl
   <wsrm:RMAssertion>
    <wsrm:InactivityTimeout Milliseconds="600000" />
    <wsrm:BaseRetransmissionInterval Milliseconds="3000" />
    <wsrm:ExponentialBackoff />
    <wsrm:AcknowledgementInterval Milliseconds="200" />
</wsrm:RMAssertion>
<wsap:UsingAdressing/>
<wsp:Policy>
```

Policy Expressions II

- **ExactlyOne**: Genau eine der Assertions ist erforderlich
- **Optional-**Attribut: Die Assertion ist optional

```
<wsp:Policy
   xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
   xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy" >
   xmlns:wsrm="http://schemas.xmlsoap.org/ws/2005/02/rm/policy
   <wsrm:RMAssertion wsp:optional=“true“/>
  <wsp:ExactlyOne>
     <sp:Basic256Rsa15 />
     <sp:TripleDesRsa15 />
  </wsp:ExactlyOne>
</wsp:Policy>
```

Policy Expressions III

- Policy Operatoren können beliebig miteinander verschachtelt werden
- Alle Policy Expression lassen sich in einer Normal-Form darstellen:

```
<wsp:Policy
xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy" >
 <wsp:ExactlyOne>
   <wsp:All>
     <sp:RequireDerivedKeys />
     <sp:WssUsernameToken10 />
   </wsp:All>
   <wsp:All>
     <sp:RequireDerivedKeys />
     <sp:WssUsernameToken11 />
   </wsp:All>
   <wsp:All>
     <sp:WssUsernameToken10 />
   </wsp:All>
   <wsp:All>
     <sp:WssUsernameToken11 />
   </wsp:All>
 </wsp:ExactlyOne>
</wsp:Policy>
```

Referenzierung von Policys

- Durch **wsp:id** oder **wsp:name**-Attribut lassen sich Policys eindeutig identifizieren
- Durch das **PolicyReference** Elementlassen sie sich referenzieren und wiederverwerten

Policy Intersection

- Mechanismus um die Kompatiblität von 2 Policys zu bestimmen
- Ermittelt die Schnittmenge von übereinstimmenden Policy Alternativen
- Übereinstimmung der Assertions wird nur über den Typ bestimmt, Parameter und Attribute können nicht verglichen werden

WS-PolicyAttatchment

- 2 allgemeine Mechanismen eine Policy an ein Policy Subjekt zu binden:
- Externes Anbinden durch **wsp:PolicyAttatchment**
- XML Attatchment an XML Elementen, die ein Subjekt beschreiben durch
  - **wsp:PolicyURI-Attribut** oder
  - **Wsp:PolicyReference** Element

XML Attatchment

```
<MyElement wsp:PolicyURIs="
  http://www.fabrikam123.com/policies#RmPolicy
  http://www.fabrikam123.com/policies#X509EndpointPolicy" />
```

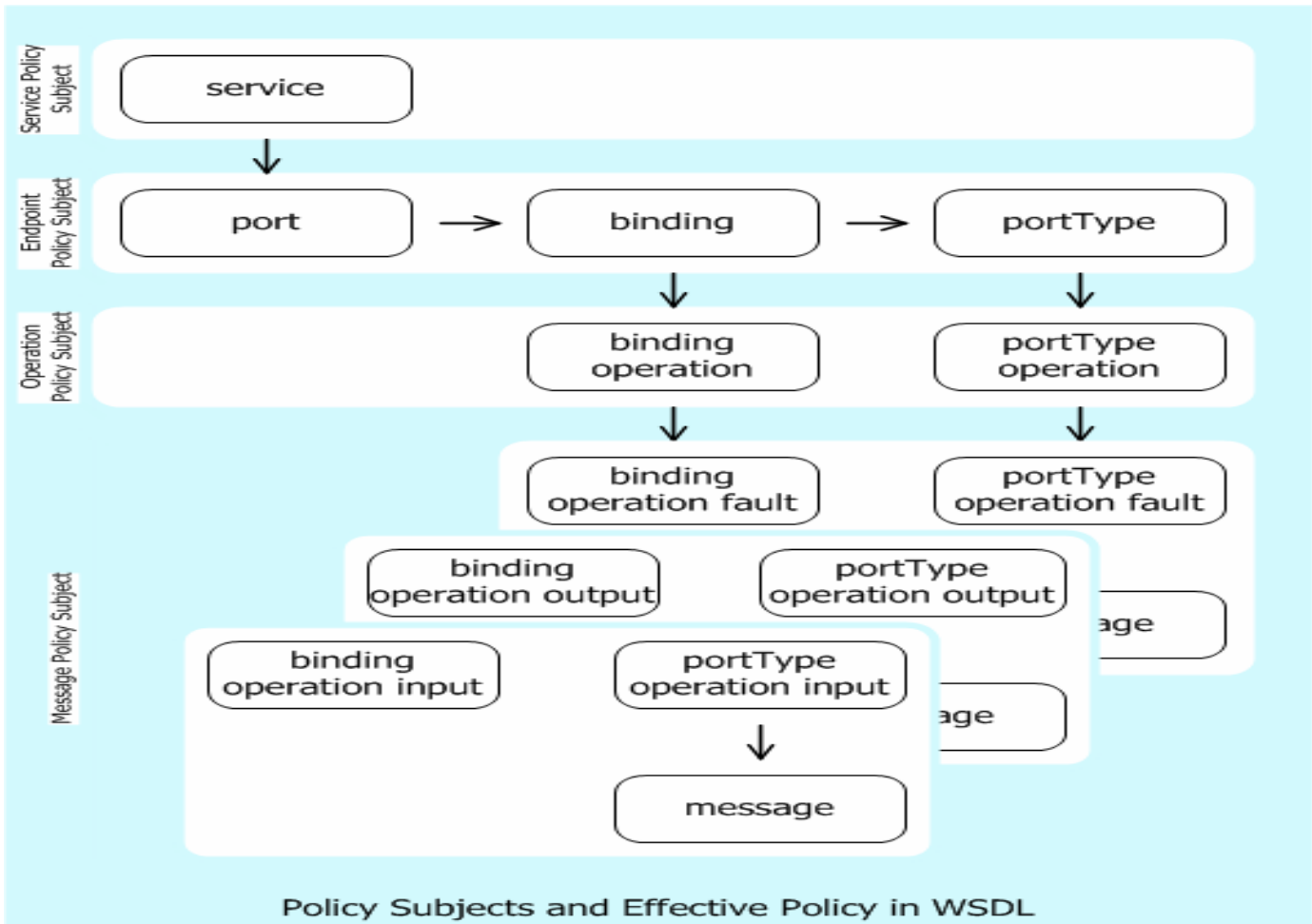
oder

```
<MyElement>
  <wsp:PolicyReference
    URI="http://www.fabrikam123.com/policies#RmPolicy" />
  <wsp:PolicyReference
    URI="http://www.fabrikam123.com/policies#X509EndpointPolicy" />
<MyElement/>
```

Die resultiernde Policy sieht jeweils so aus:

```
<wsp:Policy
xmlns:rmp="http://schemas.xmlsoap.org/ws/2005/02/rm/policy"
xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy" >
 <rmp:RMAssertion>
   <rmp:InactivityTimeout Milliseconds="600000" />
   <rmp:BaseRetransmissionInterval Milliseconds="3000" />
   <!-- Details omitted for readability -->
 </rmp:RMAssertion>
 <sp:AsymmetricBinding>
   <wsp:Policy>
     <!-- Details omitted for readability -->
     <sp:IncludeTimestamp />
     <sp:OnlySignEntireHeadersAndBody />
   </wsp:Policy>
 </sp:AsymmetricBinding>
</wsp:Policy>
```

Policy Attatchment bei WSDL



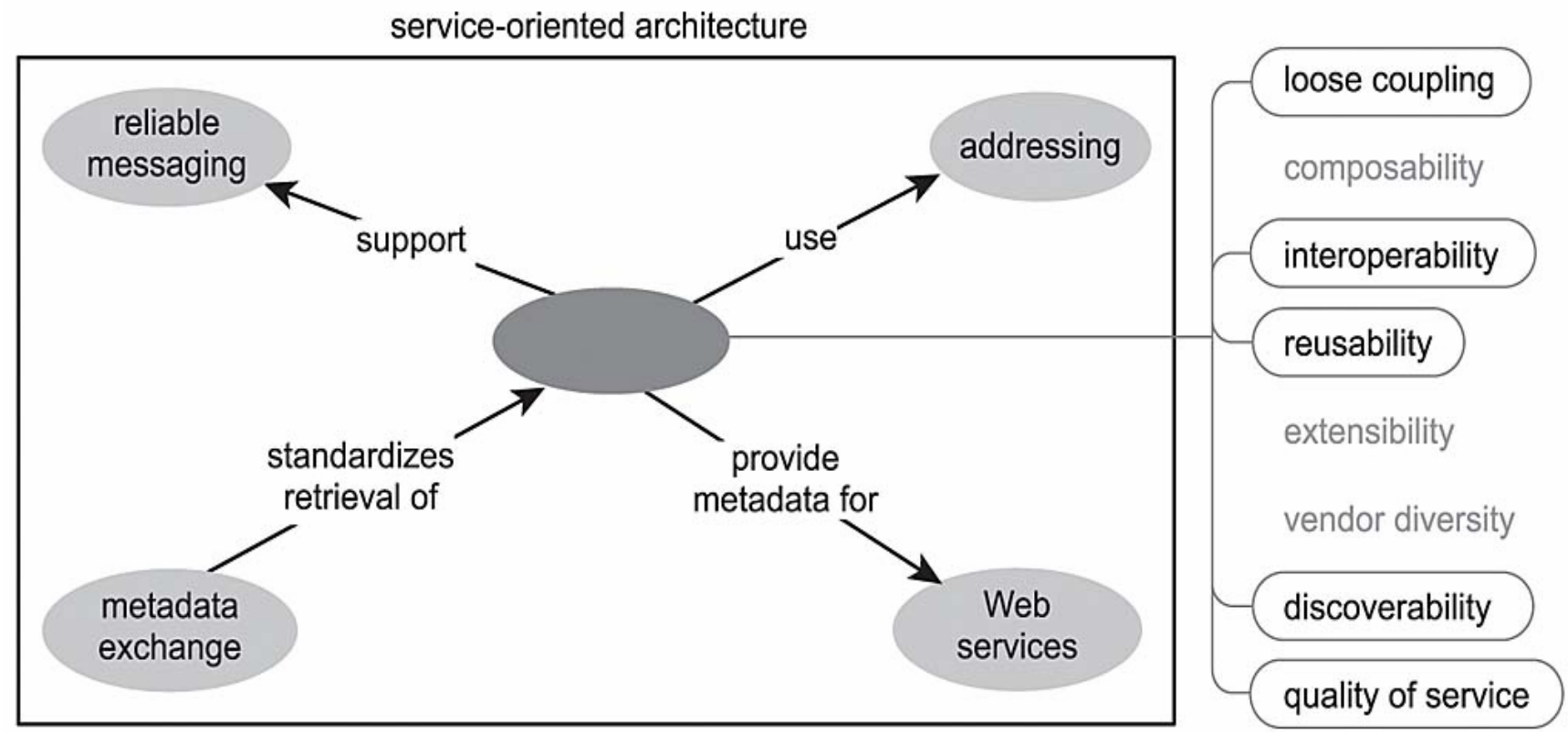
Quelle: Microsoft

Externes Attatchment durch das PolicyAttatchment Element

- PolicyAttatchment hat 3 Kindelemente
  - Wsp:AppliesTo:URI für Policy Scope
  - Wsp:PolicyReference/wsp:Policy für die Policy
  - Wsse:Security: Optionale angaben für die Sicherheit

```
<wsp:PolicyAttachment>
  <wsp:AppliesTo>
    <wsa:EndpointReference   xmlns:fabrikam="..." >
      <wsa:Address>http://www.fabrikam123.com/acct</wsa:Address>
      <wsa:PortType>fabrikam:InventoryPortType</wsa:PortType>
      <wsa:ServiceName>fabrikam:InventoryService</wsa:ServiceName>
    </wsa:EndpointReference>
  </wsp:AppliesTo>
  <wsp:PolicyReference URI="http://www.fabrikam123.com/policies#RmPolicy" />
</wsp:PolicyAttachment>
```

Zusammenhänge von WS-Policy,WS-Addressing und WS-RM



Quelle: T.Erl