WAM-XML: Maschinenlesbare Architekturbeschreibungen im Web

Johannes Meinecke, Frederic Majer, Martin Gaedke

Institut für Telematik
Universität Karlsruhe (TH)
Zirkel 2
76128 Karlsruhe
{meinecke, majer, gaedke}@tm.uni-karlsruhe.de

Die zunehmende Verflechtung von Geschäftsprozessen und Web Anwendungen über Unternehmensgrenzen hinweg begründet den Bedarf an systematischen Ansätzen für die Entwicklung und den Betrieb dieser Systeme. Aufgrund der zentralen Bedeutung der Evolution für Föderationen kommt der Modellierung der Architektur des Gesamtsystems eine entscheidende Rolle zu. Dieser Beitrag präsentiert das WebComposition Architecture Model (WAM) als einen Ansatz zur Modellierung der grundlegenden Aspekte von hochgradig verteilten und föderierten Systemen. Der Ansatz setzt sich aus einer graphischen Notation, einem Rahmenwerk für Erweiterungen, einer maschinenverarbeitbaren Sprache (WAM-XML) und der Unterstützung zur Laufzeit zusammen.

1 Einleitung

Der Trend hin zu Unternehmen, die ihre Dienste global und in Echtzeit anbieten sowie die damit einhergehende Verflechtung von Unternehmen und derer Geschäftsprozesse, stellen komplexe Herausforderungen an die zugrunde liegenden Technologien dar. Auch das Web wurde durch diesen Paradigmenwechsel beeinflusst was sich unter anderem auch im in diesem Zusammenhang geprägten Begriff Web 2.0 [Or05] äußert. Neben mehreren darunter subsumierten Konzepten liegt ein zentraler Aspekt in der Nutzung des Webs als Plattform für verteilte Anwendungen. Anstatt dem Benutzer lediglich Informationen zur Verfügung zu stellen, werden Anwendungen miteinander verschaltet und so genannte *Portale der 4. Generation* [GP03] ermöglicht. Diese zeichnen sich dadurch aus, dass sie nicht nur Informationen über einfache HTML verknüpfen, sondern darüber hinaus eine Kopplung zwischen zugrunde liegenden Portalkomponenten vorsehen, um beispielsweise Daten, Funktionalitäten oder Benutzerkonten mit anderen Portalen gemeinsam zu nutzen.

Als eine grundlegende Technologie kommen in diesen Beziehungsgeflechten oftmals Web Services zum Einsatz, die bestimmte Dienste in der Föderation zur Verfügung stellen und untereinander in Beziehung stehen. Fällt in solch einem Szenario ein Web Service aus, so kann dies unvorhersehbare Auswirkungen auf das Gesamtsystem haben.

Eine weitere Komplexität besteht in der Natur einer Föderation, da den Fragestellungen bezüglich der Zugriffskontrolle und anderer Sicherheitsaspekten begegnet werden muss [Ca05] und das gesamte System der ständigen Evolution unterliegt.

Zusammenfassend besteht der Bedarf an systematischen Ansätzen für Entwicklung und Betrieb von verteilten und Web-basierten Systemen. Einen zentralen Faktor stellen hierbei Modelle dar, die von den Systemen abstrahieren und nur die grundlegenden Aspekte wiedergeben, um diese Informationen beispielsweise Architekten oder Administratoren einfach vermitteln zu können. Zu diesem Zweck wird in diesem Beitrag das WebComposition Architecture Model (WAM) beschrieben. Im folgenden Abschnitt 2 zeigen wir die drei aus unserer Sicht zentralen Schlüsselanforderungen im Zusammenhang mit der Modellierung von Web-basierten, föderierten Systemen auf und untersuchen existierende Ansätze bezüglich dieser Anforderungen. Im Abschnitt 3 wir der WAM-Ansatz als Rahmenwerk zur Beschreibung der Architektur von Systemen vorgestellt. Der darauf folgende Abschnitt 4 beschreibt die auf XML basierende, maschinenverarbeitbare Notation von WAM, dessen praktischer Einsatz in Abschnitt 5 demonstriert wird.

2 Modellierung komplexer Systeme

Die Modellierung der Architektur eines Systems kann unter Berücksichtigung verschiedenster Aspekte und Gesichtspunkte durchgeführt werden. Aus Sicht des Software Engineerings definiert sich die Architektur als „die Struktur oder Strukturen eines Systems, die sich zusammensetzen aus Softwarekomponenten, deren öffentlicher Attribute und den Beziehungen zwischen ihnen“ [BCK98]. In dem vorliegenden Föderations- und Web-spezifischen Fall können Web Services und Web-Anwendungen als die Kernkomponenten des Systems aufgefasst werden. Anhand der im letzten Abschnitt herausgearbeiteten Charakteristika der vorliegenden Systeme, haben wir folgende drei Schlüsselanforderungen an einen Modellierungsansatz für die beschriebenen Systeme abgeleitet.

2.1 Anforderungen an die Modellierung komplexer Systeme

Implizite Berücksichtigung von Sicherheitsaspekten: Eine charakteristisches Merkmal der beschriebenen Systeme liegt in der Tatsache, dass Zugriffe auf einzelne Komponenten oftmals über die Unternehmensgrenzen hinweg geschehen. Aus diesem Grund ist der Begriff der Sicherheit auf die Kontrolle der Zugriffe auf die Ressourcen der Föderationspartner zu beziehen. In konventionellen Systemen wird die Sicherheit durch den Einsatz von Firewalls und implizit definierten Vertrauensbeziehungen innerhalb der „sicheren Zone“ hergestellt. In föderierten Szenarien besteht nun das Problem, dass eine Vielzahl an Zonen vorhanden ist und die Zugriffe in Abhängigkeit der definierten Vertrauensbeziehungen zwischen den Organisationseinheiten stattfinden. Da dieses Vertrauensgeflecht starken Einfluss auf die Struktur des Gesamtsystems hat, sollen die Sicherheitsaspekte zu einem frühen Zeitpunkt in das die Architektur beschreibende Modell einfließen.

Vermeidung unnötiger Komplexität: Modelle von föderierten Systemen müssen oft verschiedenste und hochkomplexe Sachverhalte widerspiegeln. Als Beispiel können Token-basierte Sicherheitsprotokolle genannt werden, bei denen man den Austausch der verschiedenen Nachrichten in Form eines UML Sequenzdiagramms modellieren könnte. Da die genaue Abfolge der Aufrufe aber in den meisten Fällen keinen Einfluss auf die eigentliche Modellierung der Architektur hat, sollte man sich bei der Modellierung nicht auf Details konzentrieren, sondern die relevanten Informationen festhalten, die zur Kommunikation der Architektur an andere nötig sind. Die resultierende Notation sollte es demnach ermöglichen, die wichtigen Fakten mit der Hand auf Papier festzuhalten und bei Bedarf (zu späteren Zeitpunkten) um Details erweiterbar sein.

Brückenschlag zwischen Modell und System: Aufgrund des immanenten Charakters der kontinuierlichen Evolution der betrachteten Systeme besteht die Gefahr, dass schnell eine Diskrepanz zwischen Modell und modelliertem System auftritt. Um dagegen jederzeit eine aktuelle Darstellung des Systems zu besitzen, bedarf es einer maschinenlesbaren Repräsentation des Modells, die von adäquaten Werkzeugen genutzt werden kann. Hierdurch lässt sich die automatische Aktualisierung des Modells (System zu Modell) und die Verarbeitung der Informationen aus dem Modell für den Betrieb (Modell zu System) realisieren. Um die Effizienz und die Wiederverwendbarkeit der Modellinformationen in möglichst vielen Werkzeugen zu gewährleisten, sollte die spezifizierte Notation auf existierenden Standards aufbauen.

2.2 Stand der Technik

Wie im vorherigen Kapitel beschrieben, benötigen Entwicklung und Betrieb von föderierten Systemen ein dediziertes Vorgehensmodell. Im Folgenden werden nun kurz einige existierende Ansätze vorgestellt und anhand der aufgestellten Anforderungen untersucht.

Eine erste Gruppe von Ansätzen zeichnen sich durch die Trennung der Architekturaspekte auf verschiedene Modellierungsschichten aus. Kirchner [Ki05] definiert Schichten für die Spezifikation der Geschäftsprozesse, der Software sowie der Hardware eines Systems und fokussiert dabei die Architektur eines einzelnen Unternehmens. Die geschieht in ähnlicher Weise in dem im Projekt ARCUS [He99] eingesetzten Modell, welches eine zusätzliche Schicht zur Definition von technischen Begriffen der Problemdomäne einführt. Darüber hinaus werden die Evolution des Systems und der damit verbundene Anforderung der (automatisierten) Transformation von System zu Modell und umgekehrt vernachlässigt.

Neuere Ansätze wie die Dynamic Systems Initiative (DSI), die Data Center Markup Language (DCML) und die Systems Modeling Language (SysML) versuchen die auftretende Diskrepanz zwischen Modell und System zu schließen. Die von Microsoft initiierte DSI wird von dem Gedanken angetrieben die Prozesse der Entwicklung und des Betriebs von verteilten Systemen zu vereinen [Mi03]. Die zentrale Komponente stellt hierbei das System Description Model (SDM) dar, das während der Entwicklung die Planung der Lösung auf verschiedenen Abstraktionsstufen (vergleichbar den oben erwähnten Schichten) unterstützt. Darüber hinaus fungiert es während des Betriebs als

Informationsquelle für Werkzeuge zur Wartung der Föderation. Da sich der Ansatz jedoch auf die Windows-Plattform beschränkt und wie die Data Center Markup Language [Oa04] und die Systems Modeling Language [Sy05] Sicherheitsaspekte oder föderierte Identitäten [Li03] nicht berücksichtigt, lassen sich komplexe Föderationen nur bedingt modellieren.

Abschließend bleiben Vorgehensmodelle aus dem Bereich des Web Engineerings [De02] zu untersuchen. Die meisten Vertreter verfügen über dedizierte Methodiken zur Entwicklung von Anwendungen unter Berücksichtigung Web-spezifischer Aspekte, wie etwa die Navigation oder Interaktion mit dem Benutzer. Beispielsweise fokussieren WebML, OOHDM und HERA [Ka03] die Komposition von Web-Anwendungen anhand von Seiten und navigierbaren Knoten, anstatt die Komposition von Diensten und Anwendungen zu betrachten.

3 Architekturmodellierung mit WAM

Das WebComposition Architecture Model (WAM) wurde im Hinblick auf die in den vorangegangenen Abschnitten erläuterten Anforderungen entwickelt. Der Ansatz umfasst ein Rahmenwerk sowie eine Modellierungsnotation zur Beschreibung von föderierten Systemarchitekturen, die im Folgenden kurz vorgestellt werden.

3.1 Rahmenwerk

Um der Anforderung der Darstellung einer Gesamtarchitektur unter Vermeidung unnötiger Komplexität bzw. Details gerecht zu werden, bietet sich die Trennung der verschiedenen Aspekte in mehrere Modelle (Schichten) an. In unserem Ansatz bildet das WebComposition Architecture Model das zentrale Basismodell zur Spezifikation der entscheidenden Charakteristika eines föderierten Systems. Wie aus Abbildung 1 ersichtlich, können im Basismodell unberücksichtigte Aspekte über Erweiterungen der Rahmenarchitektur um weitere Schichten aufgenommen und deren Elemente durch modellübergreifende Beziehungen mit Elementen anderer Schichten verknüpft werden.

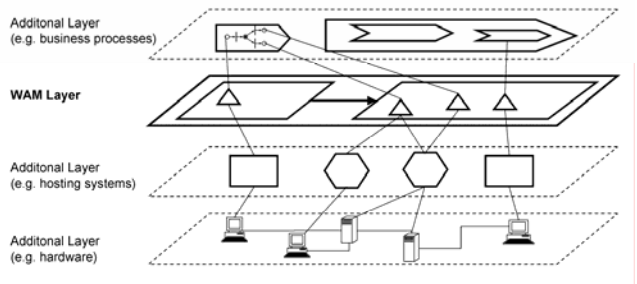

Abbildung 1: Schichtenmodell der Rahmenarchitektur

Um die Kompatibilität und die Möglichkeit der Komposition der Schichtenmodelle zu gewährleisten, müssen deren Definitionen einigen Regeln entsprechen. Unter anderem besteht eine Forderung darin, dass ein realer Systembestandteil nicht mehrfach im Modell abgebildet wird, um Inkonsistenzen zu vermeiden. Darüber hinaus existieren Vorgaben bezüglich der Struktur der Metamodelle für erweiternde Schichten.

3.2 Graphische Notation von WAM

Im folgenden Abschnitt wird erläutert, wie mittels WAM die Föderations-spezifischen Aspekte eines Web-basierten Systems beschrieben werden. Anstatt eine universelle Modellierungstechnik wie z.B. UML Stereotypen einzusetzen, wurde ein Domänen-spezifischer Ansatz mit einfach zu zeichnenden Elementen gewählt.

Abbildung 2 beinhaltet die zentralen Modellierungselemente von WAM. Der *Security Realm* sowie der *Identity Provider* stellen die Grundbausteine bei der Modellierung von Föderationen dar. Der Security Realm entspricht dem durch einen Föderationspartner verwalteten Bereich mit den sich darin enthaltenen Komponenten des Gesamtsystems. Der zu einem Security Realm zugehörige Identity Provider regelt die Authentifizierung von Benutzern der entsprechenden Organisationseinheit. Zum Aufbau von Föderationen bedient man sich der *Trust Relationship*, um die Vertrauensbeziehung zwischen verschiedenen Organisationseinheiten (Realms) zu spezifizieren. Die zentralen funktionalen Komponenten eines Systems können mittels der Elemente *Service* und *Application* in das Modell integriert werden. In der realen Welt stellen Dienste (SOAP Web Services) ihre Funktionalitäten über eine WSDL Schnittstelle zur Verfügung und

können selbst andere Dienste zur Durchführung ihrer Aufgaben aufrufen. Dagegen bilden Applikationen die zentralen Zugangspunkte für den Benutzer in Form von Web-Anwendungen und Portalen. Ein *Data Provider* stellt im Gegensatz zu einem Dienst eine Komponente dar, die nicht eigenständig Web-fähig ist. Ein Beispiel hierfür sind Legacysysteme, auf deren Daten innerhalb der Föderation nur über einen Wrapper Dienst zugegriffen werden kann. Sobald eine solche Komponente weiter reichende Funktionalitäten anbietet, wird diese im Modell als *Process Unit* repräsentiert. Als letztes Modellierungselement seien die *Invocation Links* erwähnt, die zur Spezifikation der Aufrufe von Applikationen und Dienste eingesetzt werden. Bei Bedarf können diese Aufrufe außerhalb der graphischen Notation detaillierter erfasst werden. Die Namen der hierdurch entstehenden Zugriffsprofile werden im Diagramm an den jeweiligen Aufruf annotiert und können an beliebigen Stellen wieder verwendet werden.

Um den Einsatz von WAM sowie die Beziehungen zwischen den verschiedenen Modellierungselementen abschließend kurz zusammenzufassen, sei auf Abbildung 2 verwiesen. Wie man dem Modell entnehmen kann, enthält es den Entwurf zweier föderierter Web-Portale (*Portal1*, *Portal2*). Beide Portale beziehen ihre Daten aus Diensten, wobei sich *Portal1* zusätzlich zum Dienst *WS1* auch der Funktionalität von Dienst *WS2* aus einem anderen Realm bedient. Diese Integration wird durch die unidirektionale Vertrauensbeziehung zwischen Realm *R1* und *R2* ermöglicht.

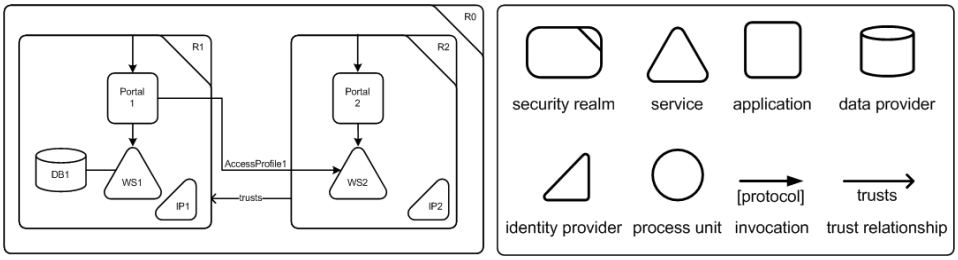

Abbildung 2: Beispielszenario von WAM

Da in diesem Unterkapitel WAM nur kurz skizziert und wichtige Aspekte wie die z.B. die Grundlagen der Token-basierte Zugriffssteuerung ausgeklammert wurden, sei auf die detaillierte Beschreibung in [MGN05] hingewiesen.

4 WAM-XML

Bei der Entwicklung der graphischen Notation von WAM bestand der Fokus darauf, einfache und verständliche Diagramme zur Abbildung der Architektur eines Gesamtsystems zur Verfügung zu stellen, um die Kommunikation zwischen verschiedenen an der Architektur beteiligten Personen, z.B. während des Entwicklungsprozesses eines Systems, zu erleichtern. Gleichwohl bleibt während des Betriebs eine Diskrepanz zwischen Modell und modellierten System aufgrund der Evolution unvermeidbar. Aus diesem Grund wurde eine XML-basierte Sprache (im Folgenden WAM-XML genannt) entwickelt, mit Hilfe deren sich die vorgestellten Konzepte geeignet beschreiben lassen. Sie stellt zugleich die Grundlage für die automatisierte Verarbeitung der Modellinformationen mittels Unterstützungswerkzeugen bzw. für die Code-Generierung dar. Neben der Möglichkeit die Standardelemente aus WAM zu spezifizieren, unterstützt WAM-XML die Einbindung von weiteren Modellierungsschichten und –elementen sowie die Aggregation von miteinander in Beziehung stehenden Elementen innerhalb eines Dokumentes, um eine übersichtliche Darstellung des Gesamtsystems zu ermöglichen.

Im Folgenden werden einige Auszüge aus dem die Sprache definierenden XML-Schema vorgestellt. Bei dem Entwurf der Schemadefinition wurde im Hinblick auf eine höchstmögliche Interoperabilität darauf geachtet, bestehende Standards zu berücksichtigen. Dies zeigt sich z.B. in dem Metadaten-Konzept der Sprache, welches auf dem Konzept der Dublin Core Metadata Initiative [An03] basiert und einen Standard bei der Definition von gängigen Metadaten darstellt. Darüber hinaus werden Beziehungen zwischen verschiedenen Elementen mittels XLink [DMO00], einem Standard zur Verknüpfung von Elementen in XML-Dokumenten, definiert. Um die verschiedenen Standards und die Modelle aus der Rahmenarchitektur von WAM besser unterscheiden zu können, werden im WAM-XML die Elemente verschiedenen Namensräumen zugewiesen. So finden sich separate Namensräume für das gesamt Rahmenwerk (*core*), für WAM (*wam*) sowie für integrierte Standards wie Dublin Core (*dc*) oder XLink (*xlink*). Außerdem können weitere Namensräume für Modellerweiterungen eingeführt werden.

Das Wurzelelement einer Architekturspezifikation *Model* setzt sich aus einem *Header* und einem *Body* zusammen (1). Hierbei beschreibt der Header das gesamte Architekturmodell sowie die einzelnen enthaltenen Modellebenen. Der Body beinhaltet die verschiedenen Elemente der einzelnen Ebenen sowie die Beziehungen zwischen den Elementen verschiedener Ebenen. Wie man dem Ausschnitt aus dem XML-Schema entnehmen kann, gibt es für Header und Body keine Einschränkungen bezüglich der inneren Struktur. Zwar schränkt dieser Freiheitsgrad die Verifizierbarkeit des resultierenden XML-Dokuments ein, aber es besteht der Vorteil, dass Erweiterungen an dem Modell keine Anpassungen an der Schemadefinition bzw. an den das XML verarbeitenden Werkzeugen nach sich ziehen.

```
<xs:complexType name="Model">                                    (1)
  <xs:sequence>
    <xs:element name="Header" type="core:Header" minOccurs="0"/>
    <xs:element name="Body" type="core:Body" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Header">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:any processContents="strict"/>
  </xs:choice>
</xs:complexType>

<xs:complexType name="Body">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:any processContents="strict"/>
  </xs:choice>
  <xs:attribute ref="xlink:type" fixed="extended"/>
</xs:complexType>
```

Aufgrund der Tatsache, dass die Modellierungselemente der einzelnen Modelle von der Typ *Element* (2) erben müssen, um der WAM-Spezifikation zu entsprechen, verfügen alle Elemente über eine Reihe an Standardattributen – den Dublin Core Attributen. Diese Konformität ermöglicht die einheitliche Verarbeitung der unterschiedlichsten Elementausprägungen, da jedes Element z.B. über einen Bezeichner (*Identifier*), einem Uniform Resource Identifier (URI) zur eindeutigen Identifizierung der Ressource oder über einen für den Menschen besser verständlichen Titel (*Title*) verfügt, welcher z.B. bei der Visualisierung des Modells Verwendung finden kann.

```
<xs:complexType name="Element">                                  (2)
  <xs:sequence>
    <xs:element ref="dc:Identifier"/>
    <xs:element ref="dc:Title" minOccurs="0"/>
    <xs:element ref="dc:Creator" minOccurs="0"/>
     ...
  </xs:sequence>
</xs:complexType>
```

Ein weiterer, von *Element* abgeleiteter, Modellierungstyp stellt *Relationship* dar, mit dessen Hilfe Beziehungen zwischen den verschiedenen Elementen ausgedrückt werden können (3). Entsprechend der XLink-Spezifikation werden die verschiedenen Ressourcen dabei nicht direkt über ihren Bezeichner adressiert, sondern den XLink-Attributen *from* und *to* werden die Namen von separat definierten Mengen an Ressourcen zugewiesen. Zur Definition dieser Mengen bedient man sich dem im WAM-XML beinhaltetem Typ Selector, welcher die Verknüpfung zwischen solch einem definierten Namen und dem URIs einer Ressource herstellt. Die URI stellt hierbei meist ein XPointer-Ausdruck dar und kann sich somit auch auf eine in einem anderen XML-Dokument spezifizierte Ressource beziehen. Insgesamt hat der Ansatz neben der Konformität zu XLink den Vorteil, dass durch die verwendete Indirektion mit der Spezifikation einer Relation eine Beziehung zwischen mehreren Quellen und Senken (n:m-Beziehung) hergestellt werden kann. So kann z.B. über eine Beziehung ausgedrückt werden, dass alle Realms eines Modells einem zentralen Realm vertrauen.

```
<xs:complexType name="Relationship">                                    (3)
  <xs:complexContent>
    <xs:extension base="core:Element">
      <xs:attribute ref="xlink:type" fixed="arc"/>
      <xs:attribute ref="xlink:from" use="required"/>
      <xs:attribute ref="xlink:to" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="Selector">
  <xs:attribute ref="xlink:type" fixed="resource"/>
  <xs:attribute ref="xlink:label" use="required"/>
  <xs:attribute ref="xlink:href" use="required"/>
</xs:complexType>
```

Der folgende Typ *Invocation* ist von Relationship abgeleitet und stellt ein Beispiel eines konkreten Modellierungselements von WAM dar (4). Da der Typ zur Spezifikation von Aufrufen einzelner Systemkomponenten dient, erweitert die Schemadefinition die geerbten Attribute der Obertypen um weitere schichtspezifische Attribute wie das zugrunde liegende Protokoll für den Transport der SOAP-Nachrichten sowie SOAP-spezifische Einschränkungen an die Anfrage- und Antwortnachrichten mittels dem *SOAPContext*.

```
<xs:complexType name="Invocation">                                      (4)
  <xs:complexContent>
    <xs:extension base="core:Relationship">
      <xs:sequence>
        <xs:element name="SOAPTransport" type="wam:SOAPTransportType"/>
        <xs:element name="Request" type="wam:SOAPContext" minOccurs="0"/>
        <xs:element name="Response" type="wam:SOAPContext" minOccurs="0"/>
          ...
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:simpleType name="SOAPTransportType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="http"/>
    <xs:enumeration value="https"/>
```

```
    <xs:enumeration value="smtp"/>
    <xs:enumeration value="tcp"/>
  </xs:restriction>
</xs:simpleType>
```

Abschließend soll nun anhand des bereits angesprochenen Beispielszenarios der Erweiterung der WAM-Rahmenarchitektur um eine Hosting-Schicht (*hst*) auszugsweise dargelegt werden, wie einzelne Schematypen instantiiert werden (5). In diesem speziellen Szenario befindet sich ein Web Service mit der Bezeichnung *WS1* auf einem Web Server *Server1*. Dieser Zustand wird mittels einer Relationship innerhalb des Namensraums der Hosting-Schicht spezifiziert. Wie in dem Abschnitt zur Definition von Beziehungen zwischen Modellierungselementen beschrieben, werden die beiden Ressourcen im WAM-XML nicht direkt verknüpft, sondern es wird eine Beziehung zwischen Platzhaltern für beliebig viele Elemente definiert. In diesem einfachen Fall stellen *LabelWS1* und *LabelS1* Platzhalter für jeweils eine Ressource dar und die konkrete Deklaration der Ressourcen erfolgt über einen XPointer-Ausdruck anhand ihrer Bezeichner. Im Falle der Nutzung des resultierenden WAM-XMLs durch eine Anwendung, welche keine Kenntnis von der Erweiterung der Rahmenarchitektur um die Hosting-Schicht hat, kann das Modell dennoch verarbeitet werden. Hierbei werden nur die bekannten Elemente interpretiert und unbekannte Elemente ignoriert.

```
<wam:Service>                                                    (5)
  <dc:Identifier>http://mwrg.tm.uka.de/ws1</dc:Identifier>
  <dc:Title>WS1</dc:Title>
  ...
</wam:Service>

<hst:System>
  <dc:Identifier>http://mwrg.tm.uka.de/server1</dc:Identifier>
  <dc:Title>Server1</dc:Title>
  <hst:Type>WebServer</hst:Type>
  ...
</hst:System>

<hst:HostingRelationship xlink:from="LabelS1" xlink:to="LabelWS1">
  <dc:Identifier>urn:sdfid:C8B906ED-...-4A8B6166399C</dc:Identifier>
</hst:HostingRelationship>

<core:Selector xlink:label="LabelWS1"
xlink:href="xpointer(/core:Model/core:Body/*[dc:Identifier='http:...'])"/>
<core:Selector xlink:label="LabelS1"
xlink:href="xpointer(/core:Model/core:Body/*[dc:Identifier='http:...'])"/>
```

Das in diesem Abschnitt vorgestellte WAM-XML eignet sich aufgrund seiner Komplexität nicht unbedingt zur manuellen Beschreibung der Architektur eines föderierten Systems, sondern stellt die Grundlage für eine automatisierte Verarbeitung der Modellinformationen während des Betriebs dieses Systems dar. Aufgrund der gleichen Ausdrucksmächtigkeit von WAM-XML und dem in Kapitel 3 beschrieben WAM ergibt sich die gewünschte Möglichkeit der Transformation eines Modells in das jeweilige andere Modell.

5 Anwendung von WAM-Konzepten

Nach der Einführung von WAM sowie dessen maschinenverständlichen Repräsentation in Form des WAM-XMLs, wird nun der Einsatz der Konzepte im Rahmen des universitätsweiten Enterprise Application Integration Projekt namens “Karlsruher Integriertes Informationsmanagement (KIM)” [Ju05] vorgestellt. Das Projektziel liegt insbesondere in der technologischen Umsetzung einer durchgängigen Integration der Geschäftsprozesse an der Universität Karlsruhe. Durch die damit verbundene Einbindung unterschiedlichster Softwaresysteme der zentralen Universitätseinrichtungen, der Fakultäten sowie weiterer Kooperationspartner entsteht eine im höchsten Maß verteilte Gesamtarchitektur mit stark föderativem Charakter, die es zu beherrschen gilt.

In diesem Zusammenhang ist die Existenz der maschinenverständlichen Repräsentation eines föderierten Systems ein erster Schritt zur Unterstützung des Entwicklung bzw. der Evolution der gesamten Architektur. In einem weiteren Schritt sollten die Informationen über das System und den darin beinhalteten Dienste frei zur Verfügung gestellt werden, damit sie von allen beteiligten Personengruppen genutzt und im Zuge der Evolution gepflegt werden können. Der hierfür implementierte Dienst (im Folgenden WAM Infrastruktur Service genannt) wurde, in Anlehnung an die vorhandenen funktionalen Komponenten der Föderation, als Web Service umgesetzt. Der Dienst implementiert eine standardisierte Schnittstelle namens CRUDS, welche das genormte Erstellen (**C**reate), Lesen (**R**ead), Aktualisieren (**U**pdate), Löschen (**D**elete) und Suchen (**S**earch) von XML-Objekten – den Modellierungselementen – ermöglicht. Über die Spezifikation von unterschiedlichen Ausgabeformaten, können die Elemente in verschiedenen Formaten und Detaillierungsstufen zur Verfügung gestellt werden. Beispielsweise könnte das Ergebnis einer Sucheanfrage als Liste von Elementen, aber auch als einzelne RSS-Objekte, zurückgeliefert werden. Durch ein genormtes und übergreifendes Adressierungskonzept wird sichergestellt, dass die Elementinformationen aus verschiedenen Datenquellen miteinander in Beziehung gesetzt werden können. So kann z.B. zu einer in WAM spezifizierten Web-Anwendung der Status anhand der gesammelten Log-Daten eines weiteren entwickelten Infrastrukturdienstes (Monitoring Service) ermittelt werden.

Basierend auf dem WAM Infrastruktur Service wurden zwei Anwendungen entwickelt, die die bereitgestellten Modellinformationen der Föderation nutzen. Die erste Anwendung unterstützt den Architekten bei der Beschreibung eines komplexen Systems anhand eines Modells. Hierfür wurde Microsoft Vision dahingehend angepasst, dass das Erstellen und Aktualisieren von WAM-Diagrammen möglich ist. Wie aus Abbildung 3 ersichtlich, kann die Positionierung der verschiedenen Modellierungselemente sowie deren Annotation über die gewohnte Drag & Drop Benutzeroberfläche vorgenommen werden. Über eine XSLT-basierte Transformationskomponente kann das aus Visio resultierende XML in WAM-XML überführt und in die durch den WAM Infrastruktur Service verwaltete Datenbank geschrieben werden.

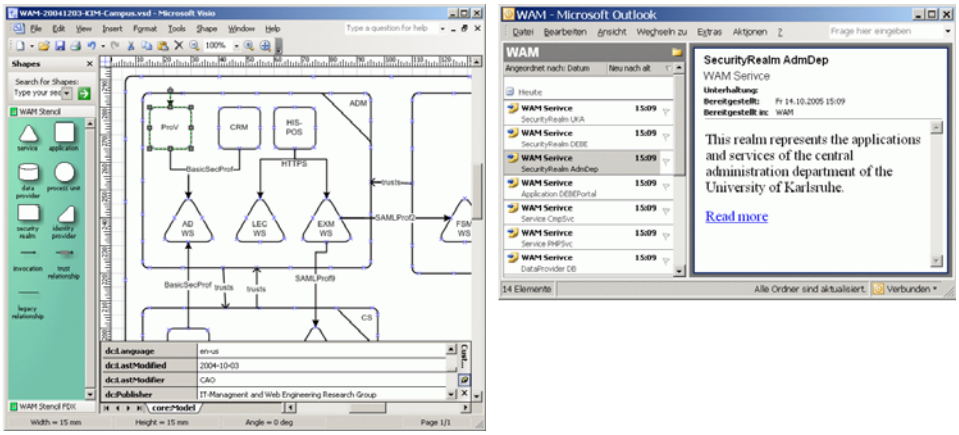

Abbildung 3: Anwendungsbeispiele von WAM (Microsoft Visio und RSS Reader)

Um auf die vielfältigen Möglichkeiten der Modelländerungen (manuell mittels Visio oder automatisiert über Unterstützungswerkzeuge) reagieren zu können, werden in realen Szenarien Benachrichtigungsmechanismen benötigt. Auf eine standardisierte Art und Weise kann dies über einen RSS Feed realisiert werden, der direkt durch den Benutzer über einen RSS Reader konsumiert (vgl.Abbildung 3) oder in aggregierter Form z.B. in ein Managementportal integriert werden kann.

In diesem Abschnitt wurde eine Methodik vorgestellt, wie Architekturinformationen eines föderativen Systems aus den modellierten WAM-Diagrammen über einen Dienst, welcher selbst einen Teil der Föderation darstellt, publiziert werden können. Die beschriebene Verarbeitung der Daten stellt eine Nutzungsalternative dar, wobei viele weitere Szenarien denkbar sind. Die Software zum Erstellen der WAM-Diagramme in Visio sowie verwandte Infrastrukturkomponenten stehen unter folgender Adresse zum Download zur Verfügung: http://mwrg.tm.uni-karlsruhe.de/downloadcenter/

6 Zusammenfassung und Ausblick

Mit der zunehmenden Nutzung des Webs als Plattform für unternehmensübergreifende, verteilte Applikationen steigt der Bedarf für dedizierte Ansätze für Entwicklung und Betrieb der resultierenden Systeme. Identifizierte Schlüsselanforderungen stellen hierbei die implizite Berücksichtigung der Sicherheitsaspekte, die Vermeidung unnötiger Komplexität sowie der Brückenschlag zwischen Modell und System dar. Die Analyse existierender Ansätze zur Modellierung von verteilten Architekturen ergab, dass kein Vertreter die Anforderungen vollständig erfüllt. Das präsentierte WebComposition Architecture Model begegnet diesem Mangel und stellt ein geeignetes Modell zur Beschreibung der grundlegenden Aspekte der Architektur von föderierten und Web-basierten Systemen in einem für Erweiterungen offenen Rahmenwerk zur Verfügung. Die auf XML basierende Repräsentation von WAM stellt die Grundlage für die automatisierte Verarbeitung der Modellinformationen dar. Über einen Infrastrukturdienst können sie der Föderation zur Nutzung in geeigneten Werkzeugen bereitgestellt werden.

Für die Zukunft ist die Entwicklung weiterer Unterstützungswerkzeuge zur Verarbeitung der zugänglich gemachten Modellinformationen vorgesehen. Denkbar ist beispielsweise die Nutzung der Daten zur automatischen Konfiguration von Systemkomponenten. Des Weiteren besteht der Bedarf einer umfassenderen Unterstützung bei der Verwaltung der gesamten Föderation, indem Aufgaben wie das Auffinden und Registrieren von bereits vorhandenen Systemkomponenten automatisch durchgeführt werden.

Literaturverzeichnis

[An03] Andresen, L.: Dublin Core Metadata Element Set, Version 1.1: Reference Description, Dublin Core Metadata Initiative (DCMI). http://dublincore.org/documents/dces/, 2003.

[BCK98] Bass, L.; Clements, P.Kazman, R.: Software Architectures in Practice; Addison-Wesley; Reading, USA, 1998.

[Ca05] Cameron, K.: The Laws of Identity. http://msdn.microsoft.com/library/en-us/dnwebsrv/html/lawsofidentity.asp, 2005.

[DMO00] DeRose, S.; Maler, E.;Orchard, D.; XML Linking Language (XLink) Version 1.0, World Wide Web Consortium, 2000.

[De02] Deshpande, Y. et al.: Web Engineering, Journal of Web Engineering, 1. Jg. (2002), S. 3-17.

[GP03] Gootzit, D.; Phifer, G.: Gen-4 Portal Functionality: From Unification to Federation. Stamford, CT, 2003.

[He99] Hermanns, J. et al.: Architekturmanagement im Großunternehmen, OBJEKTspektrum, Nr. 4/99, 1999.

[Ju05] Juling, W.: KIM Project Homepage, University of Karlsruhe. http://www.kim.uni-karlsruhe.de/, 2005.

[Ka03] Kappel, G. et al.: Web Engineering. John Wiley & Sons Ltd, England, 2003.

[Ki05] Kirchner, L.: Cost Oriented Modelling of IT-Landscapes: Generic Language Concepts of a Domain Specific Language. In Workshop on Enterprise Modelling and Information Systems Architectures (EMISA ´05), Klagenfurt, Austria, 2005; S. 166-179.

[Li03] Liberty Alliance Project: Business Benefits of Federated Identity, 2003.

[MGN05] Meinecke, J.; Gaedke, M.; Nussbaumer, M.: A Web Engineering Approach to Model the Architecture of Inter-Organizational Applications. In Conference on Component-Oriented Enterprise Applications, Erfurt 2005. Gesellschaft für Informatik; S. 125-137.

[Mi03] Microsoft: Dynamic Systems Initiative Roadmap. http://www.microsoft.com/dsi, 2003.

[Or05] O'Reilly, T.: What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software. http://www.oreillynet.com/pub/oreilly/tim/news/2005/09/30/what-is-web-20.html, 2005.

[Oa04] OASIS: Data Center Markup Language Framework Specification. http://www.dcml.org/technical_info/, 2004.

[Sy05] SysML Partners: Systems Modeling Language (SysML), Specification Version 0.9 Draft. http://www.sysml.org/, 2005.