

# ASAM-ODS Datentransfer

mit WeBServices und XML

Dipl.-Ing. Horst Fiedler  
(horst.fiedler@tiff.com)

## Hinweis

Dies ist die *TeX* - Version eines Beitrags zur ASAM-compact Konferenz. Formatierungen, Graphiken usw. entsprechen NICHT der WinWord - Variante desselben Artikels.

## Abstrakt

XML ist in der IT-Branche fest verankert. Gibt es noch Projekte, die ohne XML auskommen? Dieser Bericht schildert anhand eines Projekts, wie XML, kombiniert mit weiteren aktuellen Technologien (ECMA-Script, SOAP) und natürlich auch ASAM-ODS, in Projekten verwendet werden kann. Die eingesetzten Komponenten XML, XSL/T, ECMA Script und SOAP sind unter den Bedingungen von "public licenses" frei verfügbar.

Behandelt wird die Verwendung im Rahmen eines Datentransferprojekts wobei die Anwendung auf 2 (Quelle und Senke) Datendienste zugreift, sowie eine Interceptorkonfiguration derselben Basissoftware, bei der die Anwendung auf einen Datendienst zugreift und sich selbst als Datendienst präsentiert. Zuletzt wird die Verseifung (SOAP) des Datentransfers gezeigt, d.h. die Bereitstellung des Dienstes als eine über SOAP aktivierbare Komponente, und im Anhang wird auf Querbeziehungen zu in Arbeit befindlichen ASAM-ODS Standards eingegangen.

*XML became an approved IT standard and most projects include XML. This paper shows how XML, ASAM-ODS and related current technologies can be combined in projects.*

*The underlying example refers to a data transfer project (fetching data at one side, transferring them and importing them into another ASAM-ODS database). Additionally, the usage of the same software package to configure an ODS dataservice interceptor catching ODS-API requests and making background data service access (virtual server, model mapper) is shown. Last*

*example shows how SOAP can be applied to integrate client, transfer service and additional automation service to build an application out of these components. Final section mentions relationship to planned ASAM-ODS standards.*

## Über den Verfasser

Hr. Dipl.-Ing. H. Fiedler ist erreichbar per EMail [horst.fiedler@tiff.com](mailto:horst.fiedler@tiff.com). Firmenanschrift, Telefon/Fax, usw. siehe <http://www.tiff.com/tiff/>.

## Inhaltsverzeichnis

|  |           |
|--|-----------|
| <b>1 ASAM-ODS und XML</b>                        | <b>2</b>  |
| 1.1 XML . . . . .                                | 2         |
| 1.2 Datenschema . . . . .                        | 3         |
| 1.3 Datenfragment . . . . .                      | 4         |
| 1.4 Abbildung mit XSL/T . . . . .                | 5         |
| <b>2 Abbildungsfunktionen</b>                    | <b>6</b>  |
| 2.1 Abbildung mit Muster . . . . .               | 6         |
| 2.2 Konfigurierbarkeit . . . . .                 | 8         |
| <b>3 SOAP</b>                                    | <b>9</b>  |
| 3.1 Komponentenprotokoll . . . . .               | 9         |
| 3.2 Transferservice . . . . .                    | 10        |
| <b>4 Nachwort</b>                                | <b>13</b> |
| 4.1 Vergleich mit ASAM-ODS Vorschlägen . . . . . | 13        |
| 4.2 Ausblicke . . . . .                          | 14        |
| <b>5 Literaturverzeichnis</b>                    | <b>15</b> |

## 1 ASAM-ODS und XML

Das Ziel der unter dem Titel ASAM-ODS erarbeiteten Festlegungen war und ist es Vereinbarungen über die Strukturierung von im ASAM-Bereich anfallenden Daten (Basismodell) und den Zugriff darauf (API) zu treffen, um Daten und Softwaremodule austauschbar zu machen.

Leider gibt das Basismodell nur einen Rahmen vor, d.h. die konkrete Strukturierung einer Ablage hat Freiheitsgrade (Applikationsstruktur). Um Daten zwischen unterschiedlich strukturierten auszutauschen (inkl. Austausch mit nicht entsprechend ASAM-ODS strukturierten Ablagen, Fremdformaten) wird eine Abbildung nötig.

Abbildungen sind ihrerseits informationstechnisch fassbar, d.h. das Informationssystem kennt dann zusätzlich zu Datenstrukturelementen auch Abbildungselemente. Das nachfolgende beschriebene System versucht, mittels XML die komplexe Transformation (Zusammenspiel vieler Detailabbildungen) von Daten einer Struktur in Daten einer anderen Struktur handhabbar zu machen.

Zugleich wurde auch die Verwendung von XSL/T (Transformation) zur Abbildung von ODS - Daten untersucht.

## 1.1 XML

Ausgangspunkt des Projekts war die Anforderung, Daten aus der Ablage eines Datenerfassungssystem in eine entsprechend ASAM-ODS strukturierte Ablage zu transferieren. Dabei galt es, das Versprechen von ASAM-ODS den Austausch von Daten zu erleichtern, einzulösen. Bedingung dabei: XML soll inkludiert sein. Im Unterschied zu einem Vorläuferprojekt, sollte die Abbildung weitestgehend parametrisierbar sein. Auf ATF (den relevanten ODS Standard) wurde zugunsten des "offeneren" XML verzichtet. Die Grundstruktur der Anwendung entspricht dem üblichen Export-Transfer-Import Muster.

## 1.2 Datenschema

Die Forderung, den Transfer über eine XML-Schiene laufen zu lassen, erfordert die Festlegung eines Schemas, mithilfe dessen die zu transferierenden Daten strukturiert werden. Das Schema, in DTD - Syntax, wurde nur soweit ausformuliert, als zur Projektabwicklung nötig, d.h. eine Vollständigkeit betreffend Basiselemente, Basisnamen, Datentypen usw. war nicht beabsichtigt. Die Darstellungen hier sind auch etwas gekürzt, um für das Thema nicht relevante Teile auszublenden. Einen Vergleich zu anderen, aus ASAM-ODS Arbeitsgruppen stammenden Vorschlägen findet sich in Abschnitt 5.1.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- aoxs.dtd: cannot be used to verify a full ASAM-ODS application model for
ASAM-ODS conformity! It is used to validate export fragments created
by aoxs data binding. All typing information is informational only.
-->
<!ENTITY % basetypes '(ANY|ENV|TST|MEA|MEQ|DNA|DNG|DUN|DUG|DIM|JUT|JUP|SEQ|SEP|INS|INP) "ANY"' >
<!ENTITY % datatypes '(ANY|STRING|BYTE|SHORT|INT|FLOAT|DOUBLE|BYTESTR|LONG|BLOB|DATE) "ANY"' >

<!ELEMENT AOXs (AE+, (MI|MX)*)>
<!ELEMENT AE (AA+, AS*, AR*, IE*)>
<!ATTLIST AE ID ID #REQUIRED
NAME CDATA #REQUIRED
TYP %basetypes;
>
<!ELEMENT AA EMPTY>
<!ATTLIST AA ID ID #REQUIRED
NAME CDATA #REQUIRED
TYP CDATA #IMPLIED
```

```

    DTYP %datatypes;
>
<!ELEMENT AS EMPTY>
<!ATTLIST AS ID ID #REQUIRED
    NAME CDATA #REQUIRED
    TYP CDATA #IMPLIED
    REF IDREF #REQUIRED
>
<!ELEMENT AR EMPTY>
<!ATTLIST AR ID ID #REQUIRED
    NAME CDATA #REQUIRED
    TYP CDATA #IMPLIED
    REF IDREF #REQUIRED
>
<!ELEMENT IE (IV+, IS*, IR*)>
<!ATTLIST IE ID ID #IMPLIED
>
<!ELEMENT IV (#PCDATA)>

<!ELEMENT IS EMPTY>
<!ATTLIST IS REFS IDREFS #IMPLIED
>
<!ELEMENT IR EMPTY>
<!ATTLIST IR REF IDREF #IMPLIED
>
<!ELEMENT MI (MS+)>
<!ATTLIST MI REF IDREF #REQUIRED
>
<!ELEMENT MS (MC+)>

<!ELEMENT MC (MV*)>
<!ATTLIST MC REF IDREF #REQUIRED
    INDEP CDATA #IMPLIED
    IMPL CDATA #IMPLIED
>
<!ELEMENT MV (#PCDATA)>

<!ELEMENT MX EMPTY>
<!ATTLIST MX REF IDREF #REQUIRED
    HREF CDATA #REQUIRED
>

```

Wie erkennbar, ist das Schema einfach gehalten, Beziehungen sind immer paarweise vorhanden (jede SET-Beziehung hat auch eine INVERSE - Beziehung), der Bereich Messdaten kann in zwei unterschiedlichen Repräsentationen vorhanden sein: Explizit über Submatrix (MS), ValueSequence (MC) und Wert (MV), und als externe Container (MX), ca. entsprechend äquivalenten Konstrukten in der ATF Spezifikation.

### 1.3 Datenfragment

XML-Dokumente, die der DTD entsprechen, werden vom System erzeugt bzw. erkannt (XML Validierung). Ein Beispiel:

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE AOXS SYSTEM "aoxs.dtd">
<AOXS>
  <AE ID="A1" NAME="Umfeld" TYP="ENV">

```

```

<AA ID="A1.0" NAME="UmfeldID" TYP="ID" DTYP="INT"/>
<AA ID="A1.1" NAME="UmfeldInfo" TYP="ENVDESCR" DTYP="STRING"/>
<AA ID="A1.2" NAME="UmfeldName" TYP="NAME" DTYP="STRING"/>
<AS ID="S1.0" NAME="MotId(UmfeldID)" TYP="ENVID" REF="R2.0"/>
<IE ID="I1.1"><IV>1</IV><IV>Testdatenbank der Abteilung XT-EP315</IV>
  <IV>XT-EP315</IV><IS REFS="I2.1"/>
</IE>
</AE>
<AE ID="A2" NAME="MotId" TYP="TST">
  <AA ID="A2.0" NAME="VersuchsID" TYP="ID" DTYP="INT"/>
  <AA ID="A2.5" NAME="MotInfo" DTYP="STRING"/>
  <AA ID="A2.6" NAME="MotCreDate" DTYP="DATE"/>
  <AA ID="A2.7" NAME="d" DTYP="FLOAT"/>
  <AA ID="A2.8" NAME="Epsilon" DTYP="FLOAT"/>
  <AA ID="A2.9" NAME="Kreisprozess" DTYP="STRING"/>
  <AA ID="A2.10" NAME="MotBeginn" DTYP="DATE"/>
  <AA ID="A2.11" NAME="MotEnde" DTYP="DATE"/>
  <AA ID="A2.12" NAME="MOTID" TYP="NAME" DTYP="STRING"/>
  <AA ID="A2.13" NAME="Motorklasse" DTYP="STRING"/>
  <AA ID="A2.14" NAME="MTYP" DTYP="STRING"/>
  <AA ID="A2.22" NAME="MotEigner" DTYP="STRING"/>
  <AA ID="A2.23" NAME="MotIDHerkunf" DTYP="INT"/>
  <AA ID="A2.24" NAME="MotIDAendDat" DTYP="DATE"/>
  <AA ID="A2.25" NAME="Z" DTYP="INT"/>
  <AS ID="S2.5" NAME="Versuch(VersuchsID)" REF="R3.2"/>
  <AS ID="S2.9" NAME="Messung(VersuchsID)" REF="R4.3"/>
  <AS ID="S2.15" NAME="Messgroesse(VersuchsID)" REF="R5.4"/>
  <AR ID="R2.0" NAME="Umfeld(UmfeldID)" TYP="ENVID" REF="S1.0"/>
  <AR ID="R2.1" NAME="Users(UsersID)" REF="S15.1"/>
  <IE ID="I2.1">
    <IV>1</IV><IV>D3AZ523</IV><IV>20000828</IV><IV>130.0</IV><IV>0.0</IV>
    <IV></IV><IV></IV><IV></IV><IV>501SM3252_001</IV><IV></IV><IV>0M501LA</IV>
    <IV>LACHSTAEDTER</IV><IV>-999</IV><IV></IV><IV>6</IV>
    <IS REFS="I3.1"/><IS/><IS/><IR REF="I1.1"/><IR REF="I15.2"/>
  </IE>
  ...
</AE>
...

```

Enthaltene Redundanzen erlauben Zuordnungen (Abbildung) auf zwei unterschiedliche Arten. Die Metainformationen (Ax) tragen Namen (optional) und ID's. Die ID identifiziert Attribute im lokalen Datenmodell, d.h. solange sich das Datenbankdatenmodell nicht ändert, wird eine Abfrage an den lokalen Server `getSchema()` immer mit den gleichen ID-Werten antworten. Dieses Dokument entspricht einem Datenfragment, in dem keine Instanzen (IE, IV, MI, ...) enthalten sind und sollte nicht mit einem XML-Schema verwechselt werden. Da ein von einem anderen System einlangendes Datenfragment i.A. die Identifizierungen des lokalen Systems nicht kennt, und vermutlich auch unterschiedliche Benennungen verwendet, ist eine Transformation nötig. Diese Transformation kann nun anhand enthaltener Namen parametrisiert werden und wiederholt durchgeführt werden. Eine spezielle Option "smart" erlaubt eine nur auf Namen beruhende Assoziation beim Import.

Ähnliches gilt für das Verhältnis der XML-ID von Instanzelementen (IE) zum Wert des ASAM-ODS Basisattributs ID einer Instanz: Der Wert des

ASAM-ODS Basisattributs hat nur informativen Charakter, diese ID's werden beim Import neu vergeben. Anders die XML-ID: Mittels XML ID/IDREF wird die innere Struktur des transportierten Dokuments festgehalten.

#### 1.4 Abbildung mit XSL/T

In der ersten Phase des Projekts wurde vermutet, dass XSL/T zur Beschreibung (und ein zugehöriger Prozessor, z.B. xalan, zur Durchführung) ausreichend sein würde, d.h. es wurde versucht, folgenden Datenfluss zu implementieren:

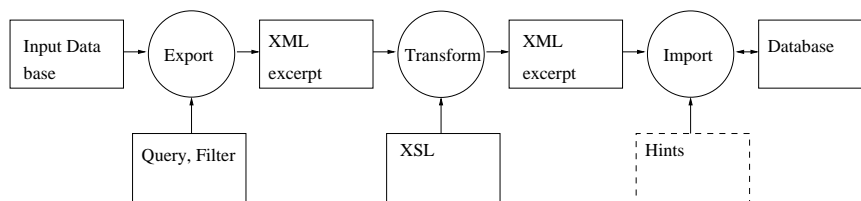


Abbildung 1: Transformation

Der Versuch scheiterte allerdings sehr rasch, da sich herausstellte, dass sich der Grossteil der auf Zielseite befindlichen Applikationsattribute in Messwerten der Quellseite versteckte, und obwohl XSLT durchaus erlaubt, prozedurale Erweiterungen (in eigenen Namespaces) auszuführen, war die Komplexität einer mittels Pattern-Matching definierten Abbildung im konkreten Fall unpraktikabel. Erkenntnis: Mit XSLT lässt sich gut filtern, auch kleinere Modelldifferenzen lassen sich sehr gut handhaben, ein XML-Dokument in ein anderes, das einer bestimmten DTD genügt, aber völlig anderen Inhalt hat, zu wandeln, geht über die Möglichkeiten von XSLT hinaus.

Daher wurde die Hauptaufgabe, eine Abbildung zu erstellen, modifiziert: Der Excerptgenerator füllt ein vorgegebenes Muster aus, d.h. von der Zielseite wird vorgegeben, welche Daten erwartet werden, und ein Templateprozessor erzeugt das XML-Dokument entsprechend dieser Vorgabe. Um dennoch eine externe (nicht kodierte) Parametrierung beim Füllen des Templates zu verwenden, wurde auf XML-Processing Instructions zurückgegriffen, d.h. ein Template entspricht einem am Zielsystem erzeugten Ausschnitt des Datenmodells (wieder obiger DTD folgend), angereichert mit den Elementen und Attributen (AE, AA, ...) zugeordneten PI's. Man könnte diese auch als eigenen Namespace definieren (äquivalent zu XSLT-Erweiterungen), da aber sonst keine Namespaces benötigt wurden und damit auch Parser verwendet werden können, die nicht namespace-fähig sind, unterblieb dies. Auch wurden PI's bereits in der XSLT - Variante (Abbildung oben) verwendet, um Hinweise (Storage-Hints) für den Import-Prozess bereits im XML Dokument zu hinterlegen. Diese Hinweise bestehen i.A. aus Bedingungen und

Aktionen, z.B. Bedingung: "Instanz dieses Namens ist bereits vorhanden",  
Aktion: "Mache Update" (statt "Insert", oder ev. auch "Beende").

## 2 Abbildungsfunktionen

Die Processing - Instruktionen, die zur Beschreibung der Abbildung verwendet werden, bestehen entsprechend W3C XML-Spezifikation aus Name und Inhalt. Der Name entspricht dem Adressaten, d.h. dem Stück Software, das in der Lage sein soll, den Inhalt der Processing Instruktion auszuführen. In der hier geschilderten Anwendung ist der Inhalt einer Transformations-PI eine in ECMAScript gehaltene Anweisung, die als Ergebnis den gewünschten Werte retournieren soll (Scriptlet).

### 2.1 Abbildung mit Muster

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE AOXS SYSTEM "aoxs.dtd">
<!-- This template is used for automated transfer.
      Testdata and measurement data are loaded using multiple process calls
-->
<AOXS>
<!-- never update (default) for all element: -->
<?c ifOne continue?>
  <AE ID="A1" NAME="Umfeld" TYP="ENV">
<!-- not even collecting data, but insert given instance (first time) -->
<?c eMode continue,ifNone insert?>
<?s setParent('I1.1');?>
<AA DTYP="INT" ID="A1.0" NAME="UmfeldID" TYP="ID"/>
<!-- upper level attributes are fine to hold "crosscontext" variables -->
<AA DTYP="STRING" ID="A1.1" NAME="UmfeldInfo" TYP="ENVDESCR"><?p1 {
  motid = aops.getMDV('!E','MOTID_PR',1);
  abteilung = aops.getMDV('!E','ABT',0);
  out(" MOTID", motid, " Abteilung", abteilung);
  return "Testdatenbank der Abteilung " + abteilung;
}?></AA>
<AA DTYP="STRING" ID="A1.2" NAME="UmfeldName" TYP="NAME"><?p1 {
  return abteilung;
}?></AA>
<AS ID="S1.0" NAME="MotId(UmfeldID)" REF="R2.0" TYP="ENVID"/>
  </AE>
  <AE ID="A2" NAME="MotId" TYP="TST">
<?c ifNone insert?><!-- but insert given instance (first time) -->
<AA DTYP="INT" ID="A2.0" NAME="VersuchsID" TYP="ID"/>
<AA DTYP="STRING" ID="A2.5" NAME="MotInfo"><?p1 {
  return aops.getMDV('!E','XDATSAEE',0);
}?></AA>
<AA DTYP="FLOAT" ID="A2.7" NAME="d"><?p1 {
  return aops.getMDV('!E','D',0);
}?></AA>
<AA DTYP="FLOAT" ID="A2.8" NAME="Epsilon"><?p1 {
  return aops.getMDV('!E','EPS',0);
}?></AA>
<AA DTYP="STRING" ID="A2.12" NAME="MOTID" TYP="NAME"><?p1 {
  return aops.getMDV('!E','MOTID_PR',0) + '_001';
}?><?s chain('A2', aox.getIEValue());?></AA>
<AA DTYP="STRING" ID="A2.13" NAME="Motorklasse"/>
```

```

<AA DTYP="STRING" ID="A2.14" NAME="MTYP"><?p1 {
  return aops.getMDV('!E', 'MTYP', 0);
}></AA>
<AA DTYP="STRING" ID="A2.22" NAME="MotEigner"><?p1 {
  x = aops.getMDV('!E', 'USER', 0);
  if (x == '') {
    return username;
  } else {
    return x
  };
}></AA>
...

```

In diesem Ausschnitt werden Instanzen (IE-Elemente) mittels Durchführung der p1 - Instruktionen erzeugt, wobei die Variable aops auf die Implementierung einer Session zum Quelldatenservice zeigt und eine Methode getMDV kennt, die als Argumente Angaben zur Identifizierung des Items im Qelldatenservice verwendet (Satzart, Spaltenname, Datensatznummer). Die Erzeugung von aops und aox - Objekten, d.h. die Festlegung des Transfer-Kontexts, und wie p1 aktiviert wird, wird im Abschnitt 4.2 weiter behandelt.

Die Scope - Eigenschaft von ECMAScript Variablen erlaubt lokalen und globalen Scope. Z.B. ist die Variable motid nicht nur innerhalb des Scriptlets (PI, in der sie verwendet wird) sichtbar, sondern in allen Scriptteilen. Variable können auch aus dem globalen Kontext genommen werden, z.B. die Variable username, die dann, falls in den Qelldaten kein User eingetragen ist, den Wert des Users, unter dem der Transfer exekutiert wird, erhält.

Damit ergibt sich eine Datentransportkette entsprechend

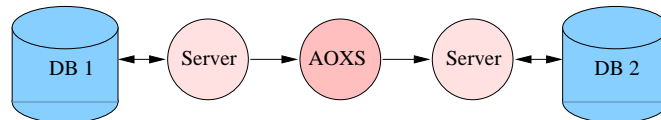


Abbildung 2: Konfiguration für Datentransfer

wobei, wenn Quelle und Ziel nicht von der Transferanwendung direkt erreichbar sind, die Transferanwendung auch unterteilt werden kann. Das Dokument wird dann nicht intern vom Templateprozessor an das Importmodul weitergereicht, sondern z.B. per EMail weitergereicht.

## 2.2 Konfigurierbarkeit

Das zuvor geschilderte Verfahren, Werte in einem XML-Dokument zur Laufzeit ermitteln zu lassen, indem Scriptlets (Processing Instructions) in einem XML Dokument, das eine Sicht repräsentiert, exekutiert werden, kann auch in einer völlig anderen Konfiguration eingesetzt werden, hier als Abbildung (online, beim ODS-API Request) bezeichnet.

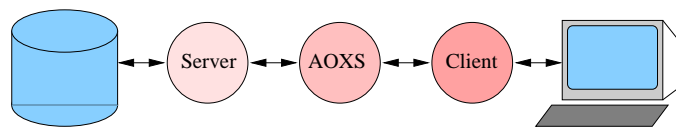


Abbildung 3: Konfiguration für Abbildung

ASAM-ODS beschäftigt sich unter dem Arbeitstitel “Model Mapping” mit diesem Thema. Ein Beispiel: Eine ASAM-ODS Datenauswertungsanwendung ist auf ein bestimmtes Applikationsmodell zugeschnitten, verwendet z.B. englische Attributnamen dort, wo keine Basisnamen mehr zur Verfügung stehen, oder setzt Attribute voraus, die erst durch Berechnung aus anderen Attributen ermittelt werden können, kann aber selbst die Berechnung nicht durchführen, erwartet Attribute, die sich de facto in anderen Entitäten befinden als Attribute einer bestimmten Entität, usw.

Anstatt eine weitere Variante derselben Auswertung zu erstellen, kann obiger Prozessor auch in den Pfad zwischen ASAM-ODS Server und Anwendung geschaltet werden. Ein virtueller ODS-Server verwendet das Template, beantwortet alle Metainformationsanfragen (nach Namen, Typen usw.) entsprechend dem Inhalt des Templates, und führt bei Anfragen nach Instanzelementen die in den Scriptlets (Processing-Instruktionen) enthaltenen Methoden durch, das Ergebnis wird über das ODS-API an die Anwendung retourniert. Dabei können auch Basisattribute “verschoben” werden, z.B. das Basisattribut NAME wird einer unterschiedlichen Spalte zugeordnet, z.B. der Spalte, in der der fremdsprachige Name steht, usw.

Sofern nicht zuviele Berechnungen, die Kaskaden von Zugriffen auf den Hintergrundserver auslösen, enthalten sind, ist die Performance einer solchen online-Abbildung überraschend gut. Das mag auch an der verwendeten Scripting Engine (rhino) liegen, die Scripts vorkompilieren (beim Start des virtuellen Servers) kann. Der Performance-Vorbehalt gilt übrigens auch für die Datentransferkonfiguration. Die Laufzeiten sind primär durch die Zugriffszeiten auf Quell- und Zielservers bestimmt, insbesondere häufige Lookups, z.B. zur Prüfung der Verfügbarkeit von Einheiten und Größen beeinflussen den Durchsatz.

### 3 SOAP

Die zuvor dargestellte Skript-Lösung erlaubt den Datentransfer im klassischen Konsolenmodus (auch wenn man z.B. den Filemanager zum Starten verwendet). Schön wäre es, die Anwendung als Dienst bereitzustellen. Das erlaubt in der Folge die Erweiterung um z.B. eine Automatisierung (Changemonitor), ohne den Transferdienst erweitern zu müssen. Die Automatisierung ihrerseits könnte ebenfalls wieder ein Dienst sein, der z.B. durch eine

RemoteAnwendung (GUI) bedient/parametrisiert wird, die in eine HTML-Seite eingebettet ist. Auf Anwendungen dieser Art zielt Microsofts .NET Strategie mit dem dazugehörige Objekt-Protokoll SOAP [4].

### 3.1 Komponentenprotokoll

Systeme zum Betrieb verteilter Komponenten benötigen eine Kommunikationsprotokoll und z.Z. sind die Systeme eng an die ihnen eigenen Protokolle gebunden.

DCOM und CORBA sind nur beschränkt für diese Aufgabe geeignet, da zu schwergewichtig bzw. plattformspezifisch. Es werden weitere TCP-Port's mit weiteren Protokollen (IIOP, DCE-RPC) belegt, damit wird die Firewall-Administration schwieriger, die Schnittstellen sind sehr starr, ..., und die Plattformunabhängigkeit ist nicht sehr ausgeprägt: DCOM gibt es fast nur auf Windows NT (wird auch nicht weiterentwickelt), CORBA in der Unix-Welt. RMI [6] ist zwar etwas leichtgewichtiger, hat seine Plattformunabhängigkeit allerdings nur durch die unmittelbare Bindung an Java und wird primär in EJB - Systemen eingesetzt.

Mit SOAP (Simple Object Access Protocol, [4], ursprünglicher Name: XML-RPC), tritt ein neues Protokoll in Erscheinung, das sehr gut zu Web-Anwendungen passt. SOAP fußt auf HTTP als weitestverbreitetes Internet Protokoll und damit ergibt sich ein weiterer Synergieeffekt: Anwendungen, die bisher als Thin-Client - Anwendungen erstellt wurden (z.B. alle jene, die mit HTML zur Präsentation im grossen und ganzen ihr Auslangen fanden), werden plötzlich einfacher erweiterbar, da sich Remote-Object-Strukturen auch ohne die doch einiges an Infrastruktur benötigenden schwergewichtigen EJB, CORBA, DCOM - Systeme aufbauen lassen.

SOAP ist allerdings noch im Entstehen begriffen, d.h. der Entwicklungsgrad hinkt hinter den anderen "großen" Protokollen nach. Nicht zu unterschätzende Vorteile sind aber seine Plattformunabhängigkeit, seine "XML-Zentrierung", wodurch keine neuen Sprachen für Deskriptoren und Definitionen eingeführt werden müssen, die Leichtgewichtigkeit, d.h. clientseitig sind die Systemanforderung gering, was Anwendungen gut downloadfähig macht, und nicht zuletzt der Druck durch den Desktop-OS-Marktführer Microsoft, dem sein DCOM abhanden gekommen ist, im Rahmen der .NET Initiative.

### 3.2 Transferservice

Die Technologie wurde im Projekt mittels SOAP von IBM (weiterentwickelt von apache.org) implementiert, und ermöglicht es, Teile der Anwendung verteilt zu exekutieren. Ein entsprechender Deployment Descriptor für den Transferdienst sieht dann wie folgt aus (verkürzt):

```
<?xml version="1.0"?>
```

```

<!-- This SOAP deploymentdescriptor offers a scriptable webservice at
      session level to execute a transfer.
-->
<isd:service xmlns:isd="http://xml.apache.org/xml-soap/deployment"
  id="urn:aoxs-transfer">
  <isd:provider type="script"
scope="Session"
methods="init transfer">
  <isd:script language="javascript"><![CDATA[
    // initialisation code to include processor and ASAM-ODS access objects
    Packages.org.mozilla.javascript.ScriptableObject.defineClass(
      this, Packages.com.tiff.js.Aox
    );
    Packages.org.mozilla.javascript.ScriptableObject.defineClass(
      this, Packages.com.tiff.js.Aop
    );

    // =====
    // global scope, parameters to run transfer,
    //
    var sourceAddr = "galois:975313576.3";
    var destAddr = "dirac:975313574";

    // no access info yet, provided by init call later
    var destUser = "";
    var destPasswd = "";

    // the work data (document)
    var aox;

    // the attached ODS services
    var aops;
    var aopt;

    // the return value client information (provided by template processing))
    var ret;

    // =====
    // May be used to set additional parameters, checks accessibility of
    // parametrized source and destination servers and
    // returns source server rpc address (the one to be monitored by
    // automation service))
    function init (user, passwd) {
    destUser = user;
    destPasswd = passwd;

    // open source and target services, may raise exceptions
    aops = new Aop(sourceAddr);
    aopt = new Aop(destAddr,"USER=" + user + "/" + passwd);
    return sourceAddr;
    }

    // =====
    // Transfer
    // @param test is name of test to be selected
    // @param template is name of template to be used
    function transfer (test, template) {

    // get source instance using testname only, missing info by constants:
    var ie = aops.getIEId("I1.1", "A2", test);
    if (ie == null)
      throw 'Unable to locate source data ' + test;

```

```

// read in template holding data model and PI's
aox = new Aox("file:/mydir/templates/" + template + ".xml");

    ret = '';

    // process the template, e.g. execute processing instructions p1
try {
    aox.process("p1", aops, ie);

    // relate actual document to target system (name lookups,
    // ID attributes are changed to target system's ID space)
    aox.relate(aopt, "s");

    // a preventive check (what will happen when attempting to store,
    // e.g. how many instance will be updated, inserted, ..., fail)
    var x = aox.status(aopt, "c");
    if (x[0] == 0){
        // no errors expected
        if (!aox.store(aopt))
            throw "Server failed to store data";
    } else {
        throw "No store attempt made due to " + x[0] + " expected errors";
    }
} catch (process) {
    // failure occurred, store current xml document for debugging
    aox.serialize("file:/mydir/logs/" + test + "_err.xml");
    // and return error
    throw process;
}
// return, ret shall hold test instance name or similar information
// about where result was stored (created by processing instructions)
return ret;
}
]]>
</isd:script>
</isd:provider>
<isd:faultListener>org.apache.soap.server.DOMFaultListener</isd:faultListener>
</isd:service>

```

Als ECMAScript-Paket wurde Rhino (mozilla.org) verwendet. Das aox-Objekt ist die Implementierung des XML-Schemas (XML - Java Data Binding), hier mittels Relax [8], d.h. das XML-Schema wurde kompiliert und von den erzeugten Klassen wurden Ableitungen gebildet. Das Scriptable aox ist der Manager von Objekten dieser Klassen und unabhängig von ODS-API's .

Die Objekte aops und aopt sind Implementierungen von ASAM-ODS Sessions (verwenden ASAM-ODS API 3.2 bzw. TRR, d.h. die ONC-RPC Varianten), d.h. die vom RPC Compiler erzeugten Stub-Klassen werden von der Scriptable-Klasse Aop für den Zugriff verwendet. Bei Verwendung von ODS 4.x müsste eine weitere Implementierung des internen Interfaces auf Basis der CORBA-Stubs erstellt werden. Das gilt auch für "Fremdformate", bei denen dann auch die Zugriffsmethoden für die Abbildung (z.B. getMDV() in Abschnitt 3.1) anzupassen sind.

Hier wird angenommen, dass der Transferdienst fest mit ASAM-ODS Diensten verdrahtet ist (RPC-Adressen). Die feste Verdrahtung erlaubt,

die Anzahl notwendiger Parameter gering zu halten. Der Dienst kann unter verschiedenen Namen (URN's) mit unterschiedlicher Parametereinstellung bzw. unterschiedlichen Skripts installiert (deployed) werden.

Der Dienst kann entweder direkt von einer Client-(end-)anwendung aktiviert werden (Einzeltransfer) oder auch von einem Automatisierungsdienst, der z.B. ein Changemonitoring am Quellserver durchführt. Der Automatisierungsdienst kann auch mit einer Desktopanwendung visualisiert und gesteuert werden (auch als Applet bzw. JNLP [7] gestartete Anwendung, da keine desktoplokalen Ressourcen benötigt werden):

| ID | TRR-Name   | Status         | Template | Datei/Datum | Versuch          | Fehler  |
|----|------------|----------------|----------|-------------|------------------|---|
| 1  | SZ2712_001 | transfer error | nfz_ti   | 14-APR-01   |                  | BSF Error: JavaScript Error: Process: MD value fetch fail |
| 3  | SM3252_008 | uptodate       | nfz_ti   | 12-JAN-00   | 501SM3252_001... |   |
| 2  | SM3252_009 | transfer error | nfz_ti   | 12-MAY-01   |                  | BSF Error: JavaScript Error: Process: MD value fetch fail |

Abbildung 4: Bedienung Automatisierung

Die Erweiterungen WDSL (zur Detailbeschreibung der Schnittstelle) und UDDI (Registry/Nameservice für WebServices) wurden im Rahmen des genannten Projekts nicht benötigt, da die Konfiguration (Festlegung der Adressen, ...) von Services und deren Clients statisch erfolgen kann. Für komplexere Anwendungen (B2B) müsste WDSL, z.B. zur autom. Erzeugung von Proxies, wenn mehrere Clients erstellt werden müssen, bzw. UDDI zur Registrierung von Diensten, falls Clients zur Laufzeit nicht über Parameter, sondern mittels Suche die passenden Dienste finden sollen, ebenfalls mitbetrachtet werden. Dies steht aber in keinem direkten Zusammenhang mit ASAM-ODS mehr, ist eher für ASAM-CCC relevant und soll hier nicht weiter betrachtet werden.

Ein ev. noch relevanter Aspekt von SOAP in Zusammenhang mit ASAM-ODS und XML: In Abschnitt 2 wurde ein XML-Element MX (external measured data) eingeführt. Wie im Fall ATF-Container ist dabei gedacht, aus diversen Gründen Objekte nicht komplett zu serialisieren, sondern als abgeschlossenes Objekt zu transferieren. Die SOAP-Spezifikationserweiterung "Messages with Attachements" stellt dafür den geeigneten Transportmechanismus bereit.

## 4 Nachwort

### 4.1 Vergleich mit ASAM-ODS Vorschlägen

Das hier behandelte Thema ist mit 2 innerhalb ASAM-ODS Arbeitsgruppen z.Z. in Bearbeitung befindlichen Themen verknüpft:

1. Model Mapping
2. XML-ATF

Beide Themen sind eng miteinander verknüpft. Nach Kenntnisstand des Autors entspricht die Aufgabenstellung des Modelmappings in etwas der in Abschnitt 3.2 behandelten Konfiguration, die Aufgabenstellung der XML-ATF Arbeitsgruppe der Festlegung eines Schemas (Abschnitt 2.1). Da dem Autor über Ergebnisse des Modelmapping Arbeitskreises keine Informationen vorliegen, beschränkt sich der Vergleich auf die XML-ATF Arbeiten, und auch hier wird nur auf die wesentlichsten Unterschiede eingegangen:

1. XML-ATF soll auf der W3C Spezifikation XML-Schema beruhen, d.h. das Schema soll “expressiver” sein als bei Verwendung von DTD möglich,
2. Das Schema soll möglichst gut ein Datenmodell wiedergeben, d.h. es sollen Datentypen und Inheritance-Struktur im Schema wiedergegeben werden.

Der erste Unterschied ist eine Folge der in Punkt 2 genannten Anforderungen, daher kann sich die Beurteilung auf diesen Punkt beschränken.

In diesem Projekt wurde für den Datenaustausch ein möglichst allgemeines Schema (repräsentiert in der DTD) gewählt, um die Struktur des transportierten Dokuments unabhängig von Quelle und Ziel zu halten, nur der Inhalt ist Nutzinformation, die DTD kann an zentraler Stelle (global eindeutig, z.B. auf [asam.org](http://asam.org)) hinterlegt werden und dient der Valididierung: Jedes Dokument, das dieser DTD entspricht, ist ein legales Datenfragment. Dies erlaubt ein statisches Binden von Implementierung an Schema (XML Data Binding), d.h. die Generierung vom Klassen, die die Elemente des Schemas repräsentieren. und erlaubt es, auch Formate (Exzerpte aus Ablagen), die nicht ASAM-ODS strukturiert sind, mit demselben Schema zu behandeln.

D.h. der Unterschied ergibt sich aus einer unterschiedlichen Zielsetzung: Ziel des hier verwendeten Schemas ist NICHT die möglichst exakte Wiedergabe des Datenmodells. Es ist nur soviel Metainformation beinhaltet, als aus praktischen Gründen zur Erstellung von Datenaustauschanwendungen notwendig ist. Die vorhandenen Transferendknoten (oft ist nur das Ziel des Transfers ein ASAM-ODS-System) kennen ihre Syntax, ein Austausch von Constraints und Kardinalitäten ist nicht sinnvoll, da die Modellierung beider Endknoten unabhängig voneinander erfolgte und feststeht. Wichtig

ist es aber, dass semantische Information (Attributnamen, ...) ausreichend vorhanden ist, um für die Parametrierung der Abbildung Bezugspunkte zu haben, was wiederum zum Thema Model Mapping gehört.

Die Aufgabe, die gesamte Metainformation möglichst exakt austauschen zu können, wird vom Autor als XMI-Thema, und nicht als ASAM-ODS - Datentransfer Thema gesehen.

## 4.2 Ausblicke

XML als ein großer IT-Standard ist ein Faktum. Ist die Standardisierung von ASAM-Schemas notwendig? Ja, zumindest für die Datenaustauschdomäne. Soll es ein einziges "wenig expressives" (wie hier dargestellt) Schema geben, oder ein "Metaschema", das festlegt, wie die expressiven spezifischen Schemas auszusehen haben?. Das wird hier nicht beantwortet, ev. beides. Betreffend Strenge von Spezifikationen: Auch hier ist XML hilfreich: Während z.B. bei IDL-basierten Spezifikationen (CORBA, RPC) die Nichteinhaltung zumeist als Showstopper wirkt, erlauben XML-basierte Spezifikation zumeist eine einfache Erstellung und Adaptieren, z.B. als XSLT Stylesheets. Dies kann speziell bei Versionskonflikten nützlich sein. In EDI-Anwendungen, und um diese zu ermöglichen, wurde ja ASAM-ODS in die Welt gesetzt, wird XML in dieser oder anderer Form sicher auch bei ASAM-ODS Eingang finden. Man darf sich aber auch die Frage stellen, ob neue, mit XML verbundene Technologien (z.B. SOAP) nicht auch in anderen ASAM-ODS Bereichen (neben ATF) berücksichtigt werden können oder sollten, z.B. auf API - Ebene: Die genannten Templates werden z.Z. durch Abfrage der Metainformationen des ODS-Servers erzeugt (Anzahl der Aufrufe ist relativ hoch, auch bei Verwendung des ODS 4.x Iterators), die Informationen könnten aber auch als ein Wert (literal XML als Datentyp) retourniert werden. Ebenso könnte man fragen, ob dort, wo Objekte "byValue" retourniert werden, statt CORBA Serialisierung nicht auch SOAP/XML verwendbar sein sollten. Und noch ein altes ASAM-ODS Thema bricht auf: Wenn eine Datenbank (z.B. Oracle, SQL-Server, Poet) XML-Objekte retournieren kann, warum diese Möglichkeiten nicht nutzbar machen?

Diesbzgl. erlaube ich mir eine Meinung zu äußern: ASAM-ODS wird auch in Zukunft mit Technologien experimentieren müssen, immer unter Berücksichtigung von dem, was in der IT-Industrie gerade als aktuelle Standards verbreitet wird. Die Anwendbarkeit, speziell von Kombinationen von Standards, wie hier XML - ECMAScript - SOAP, zu demonstrieren, ist noch immer ein aufwendiger Vorgang, der nicht auf Papier durchgeführt werden kann. Daher sollten vor dem Versuch, ein bestimmtes Verfahren oder eine bestimmte Schnittstelle zu einem Standard zu erklären, die Domaine, für die der Standard relevant sein soll, ausgelotet sein, am Besten anhand einer möglichst konkreten Aufgabenstellung. Diese "aktuellen" Standards sind übrigens bemerkenswert stabil, siehe XML (1.0 ist 4 Jahre alt). Und wenn

die W3C über SOAP wacht, kann man ähnliche Stabilität auch da erwarten. Praktisch alle ApplikationServer - Anbieter haben SOAP-Unterstützung zumindest angekündigt, auch die Java-Community integriert das Protokoll (JAXR, JAXM). Da SOAP für B2B Einsatz konzipiert ist und man ASAM-ODS Datenaustausch durchaus auch dieser Domäne zuordnen kann, kommen erste Erfahrungen in dieser Richtung vermutlich gerade recht. Oder anders herum: Wenn heute das ASAM-ODS Schema für Datenaustausch (ATF in XML) in Arbeitsgruppen definiert wird, sollte man bereits erkennen können, in welchem Kontext es verwendbar sein könnte, da dies die Entwicklungsrichtung beeinflussen könnte.

## 5 Literaturverzeichnis

1. Extensible Markup Language (XML) 1.0 Recommendation  
<http://www.w3.org/TR/REC-xml>
2. XSL Transformations (XSLT) W3C Recommendation  
<http://www.w3.org/TR/xslt>
3. ECMAScript Language Specification Ed. 3 24-Mar-2000  
ISO/IEC 16262, ECMA-262
4. Simple Object Access Protocol (SOAP) 1.1 W3C Note 08 May 2000  
<http://msdn.microsoft.com/library/en-us/dnsoapsp/html/soapspec.asp>
5. WDSL/UDDI XML-Magazin 5 2001 S 91 ff
6. Remote Method Invocation (RMI)  
<http://java.sun.com/j2se/1.3/docs/guide/rmi-iiop/>
7. Java Network Launching Protocol API JSR-56  
<http://jcp.org/jsr/detail/056.jsp>
8. Regular Language description for XML (Relax)  
ISO/IEC DTR 22250-1 <http://www.xml.gr.jp/relax/>