

Agile Geschäftsprozesse durch die Verwendung von XML bei Modellierung, Daten und Steuerlogik Workflow- basierter Systeme (in EAI, SOA und Portalen)

Marc Pellmann, Dr. Torsten Schmale

inubit AG

Lützowstr. 105-106
D-10785 Berlin
marc.pellmann@inubit.com
torsten.schmale@inubit.com

Abstract: Bei der Integration heterogener Systeme und Formate, wie sie von EAI [EAI01] Systemen durchgeführt werden, ist es wichtig, eine gemeinsame Welt zu finden, die innerhalb des Integrationsprozesses durchgängig erhalten bleibt. Hier bietet sich XML als Container der Daten und die damit verwandten Spezifikationen XSLT und XPath [XML01] zur Überführung und Abfrage der Strukturen an. Um alle Vorteile und Synergieeffekte nutzen zu können ist es wichtig, keine Brüche innerhalb der Daten und Kontrolllogik zu haben.

Einleitung

Die Abbildung automatisierter Geschäftsprozesse über Workflows unter Verwendung von XML Formaten für den Datenfluss und die Kontrollstrukturen, erlaubt eine einfache Integration von Systemen und Mitarbeitern. Sie sind ebenso das Mittel der Wahl für dynamische, agile Geschäftsprozesse.

In etablierten Architekturen erfolgt die Anbindung von Systemen und die Erstellung von Portalen über klassische Programmier Techniken. Dies erschwert ein schnelles und unkompliziertes Anpassen der Logik an agile Prozesse wie sie in heutigen Unternehmen und Geschäftsbeziehungen in der Regel vorkommen.

Durch eine SOA [SOA01] Architektur und die damit verbundene, lose Kopplung der Services über ein Workflow-System, kann schnell auf sich verändernde Businessprozesse und die darunter liegenden Systeme eingegangen werden.

Dies wird im Folgenden anhand einiger Beispiele im inubit IS [IS01] dargestellt. Es wird auch gezeigt, welche Vorteile sich in erweiterten Bereichen wie einem Workflow-getriebenen Web-Portal ergeben.

1 XML überall

Die Realisierung der Abläufe über Workflows, die in einem graphischen Diagrammeditor zusammengestellt werden, ist ein geeignetes Mittel auf schnell verändernde Prozesse und Szenarien reagieren zu können. Innerhalb der Workflows werden diverse Module miteinander verbunden. Die Daten, die von Modul zu Modul fließen, liegen entweder in XML vor oder werden innerhalb des Workflows direkt in XML umgewandelt.

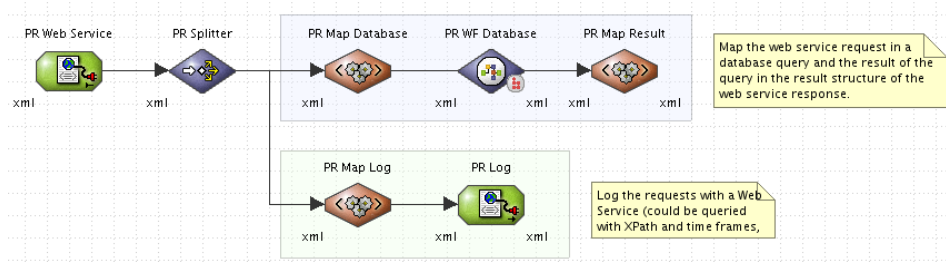


Abb.1: Bild eines Workflows mit einem WS-Connector als Anfang

Somit liegen die durch den Workflow fließenden Daten grundsätzlich in XML vor. Die Kontrolllogik innerhalb der Workflow arbeitet ebenfalls auf diesem XML (sowie auf erweiterten Kontrollvariablen, die mit dem XML in Interaktion stehen – vgl. Konformität zu [BPEL01]).

Diese Durchgängigkeit der Verwendung von XML und den damit verwandten Techniken wie XSLT/XPath zur Transformation und Abfrage findet sich innerhalb der Module wieder.

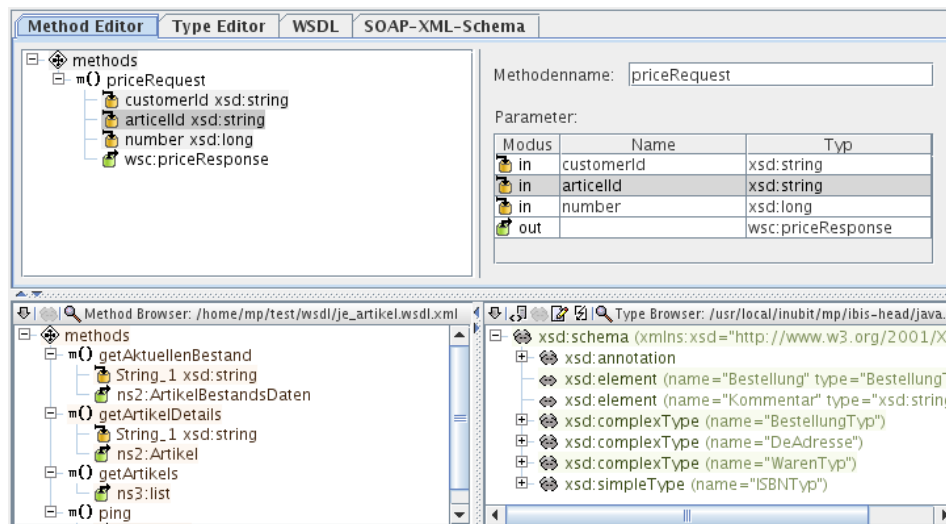


Abb.2: Graphische Erzeugung des WSDL

In der Benutzeroberfläche des Web Services Moduls (siehe Abb.2) wird die Struktur des erwarteten Aufrufes bzw. des Dokumentes graphisch definiert. Danach werden automatisch das WSDL und eine Mapping-Vorlage generiert. Wird dieses Web Services Modul nun am Anfang eines Workflows verwendet, kann dieser entsprechend der definierten Strukturen (bzw. entsprechend der generierten WSDL) aufgerufen werden. In der Ausführung auf dem Server reicht das Web Services Modul nur den Inhalt der Nachricht in XML an das nächste Modul im Workflow weiter (außerdem kann es natürlich eine Strukturüberprüfung durchführen, und behandelt die Fragen der Übertragung der Daten wie HTTPS, Client-Server Authentifizierung etc.).

Im Gegensatz zu vielen anderen Lösungen gibt es hier keine Brüche der verwendeten Technologien. Das WSDL wird nicht in Java [J01] oder .NET [MS01] Code umgewandelt, der dann für die weitere Verarbeitung notwendig ist und tiefgehende Programmierkenntnisse erfordert.

In der Regel wird das auf den Web Services Connector folgende Modul ein Steuermodul (hier ein Demultiplexer) sein, wenn verschiedene Methoden verarbeitet werden sollen. Es wird dann anhand einer XPath Abfrage entschieden, welche Zweige des Workflows weiter ausgeführt werden sollen.

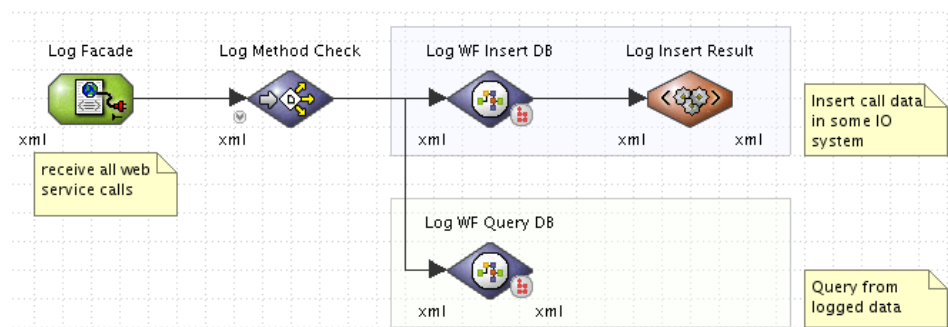


Abb.3: Demultiplexer zur Entscheidung über weitere Verarbeitung

Enthält die Beschreibung des Web Service nur eine einzelne Methode oder ist der Web Service dokumentenorientiert, kann direkt mit einem XSLT Converter Modul weitergemacht werden (in Abb.3 nicht direkt sichtbar, da in verlinktem Workflow „Log WF Insert DB“).

Dieses besteht innerhalb der Ausführung des Workflows auf dem Server aus einer XSLT Konvertierung, die verschiedenste XSLT-Prozessoren unterstützt. Die Konfiguration der Konvertierung innerhalb des Toolsets erfolgt über ein graphisches Mapping mit Test- und Debuggingssupport (siehe Abb. 4). Dies wird über ein einfaches drag-and-drop von Quellenachricht zur Zielnachricht erstellt. Auch hier werden XPath und XSLT verwendet. Erworbenes Know-how in den verschiedenen Standards um XML kann somit überall wieder verwendet werden, ohne dass imperative Programmiersprachen notwendig werden. In das XSLT kann auch jederzeit direkt eingegriffen werden, ohne dass die Möglichkeiten der graphischen Abbildung beeinträchtigt würden, d.h. volles Round-Trip Engineering ist gewährleistet.

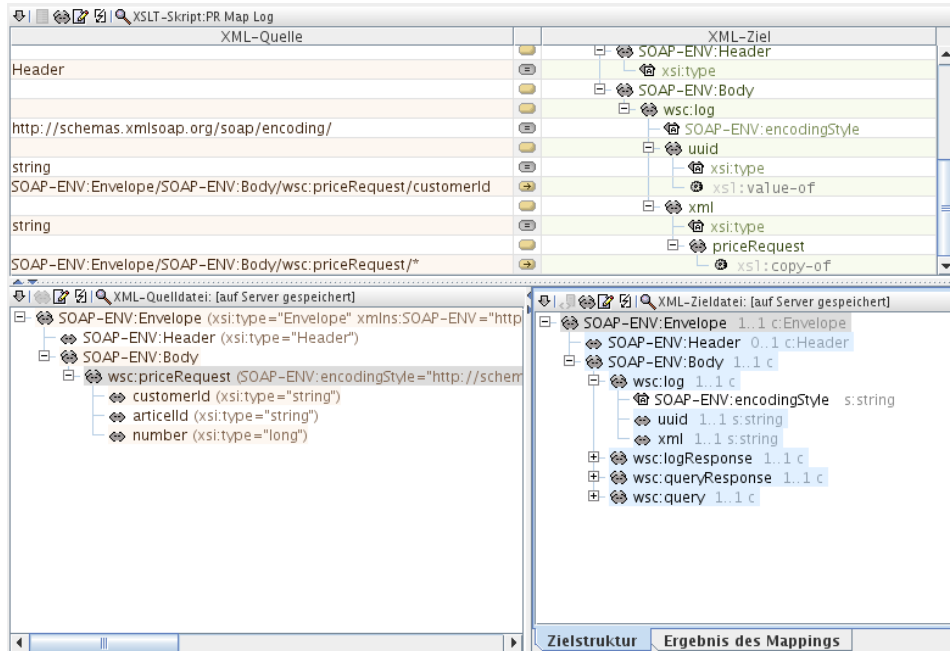


Abb.4: Ein Web Service wird auf einen anderen gemapped

Der Übergang in andere Systeme ist in den Modulen gekapselt – die Schnittstelle ist immer in XML gestaltet. Als Beispiel wird hier die über den Web Service eingegangene, kundenindividuelle Preisanfrage einmal über ein SAP System und einmal über eine betriebsinterne Datenbank aufgelöst. Im Fall von SAP sind die IDoc/BAPI-Aufrufe in einem einfachen XML Format zu füllen. Im Fall der Datenbank (Abb. 5) ist die SQL-Abfrage ebenfalls in eine XML Syntax verpackt. Somit wird zu jeder Zeit ein gleichförmiges Vorgehen über die Standards der XML Welt sowie ein bekanntes Werkzeug genutzt.

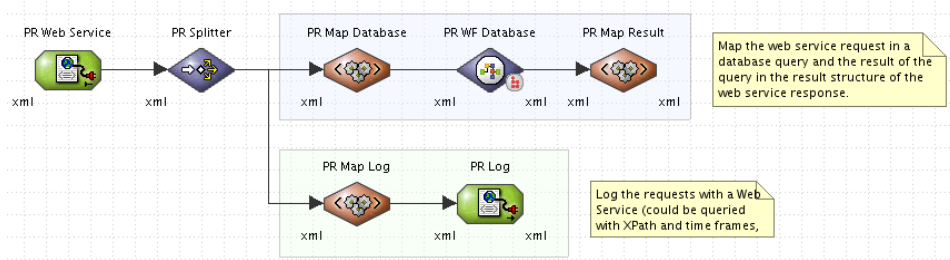


Abb.5: Preisanfrage geht an Datenbank und wird zusätzlich gelogged

2 Flexible Typisierung

Bleibt man - soweit möglich - in der XML-Welt, hat man den Vorteil, dass eine hohe Typsicherheit durch die Verwendung von XML Schemata möglich aber nicht zwingend ist. Somit ist, wo dies gewünscht ist, eine höhere Fehlertoleranz erzielbar. Ein einfaches Beispiel soll diesen Vorteil zeigen.

Abb. 1 vom Anfang dieses Textes zeigt einen Workflow der beliebige Web Service Aufrufe entgegennimmt und die übertragenen Daten in eine Datenbank speichert. Da es hier keine Typbindung und Auflösung in Java oder .NET Objekte gibt und die Validierung im Modul ausgeschaltet wurde, ist es kein Problem mit beliebigen SOAP Nachrichten umzugehen.

Dieser Workflow dient nun dazu, die eingehenden Web Services in einer Datenbank zu protokollieren, um diese später abfragen zu können. In einem anderen Workflow kann dieser Web Service also für das Mitschreiben sowohl eingehender wie auch ausgehender Web Services genutzt werden (Siehe Abb. 3). Es ist lediglich notwendig, den bestehenden eingehenden oder ausgehenden Aufruf noch um eine ID anzureichern.

Später können die geloggtten Services dann über die ID und eine Zeiteingrenzung abgefragt werden. Zusätzlich wird ein XPath angegeben, der auf die XML Struktur der Services angewendet wird. Der Service der Abfrage ist dabei, entgegen dem Service des Loggings, typsicher definiert.

Der Vorteil der sich hier ergibt liegt darin, dass beliebige Web Services über dieses Interface geloggt werden können. Es kann also jeder Web Service, der im Unternehmen Verwendung findet, mitgeschrieben werden. Es ist nicht notwendig intern zu loggen oder für jedes Logging einen neuen Web Service zu definieren.

3 Einbeziehung von Benutzerinteraktion

Vom Eingreifen des Benutzers in den Integrationsprozess bis zu Workflow-getriebenen Portalen ist auch die Benutzerinteraktion über die XML-Welt abbildbar. Die Beschreibung von graphischen Benutzeroberflächen in XML ist State-of-the-Art. Hier sprechen wir von der XML UI Language (XUL) [XUL01] gesprochen. Dies meint im Speziellen das Format, das die Mozilla Foundation für Mozilla/Netscape/Firefox entwickelt hat und im Allgemeinen die Beschreibung einer graphischen Benutzeroberfläche in XML. Auch das kommende Windows Longhorn wird auf einen solchen XML Dialekt basieren, der dort XAML heißt.

Der inubit IS bietet hier ein Format, das speziell auf die Anforderungen von Aufgabenlisten und Portalen innerhalb der Workflow-Ausführung zugeschnitten ist. Für die Darstellung kann zwischen HTML und Java/Swing gewählt werden.

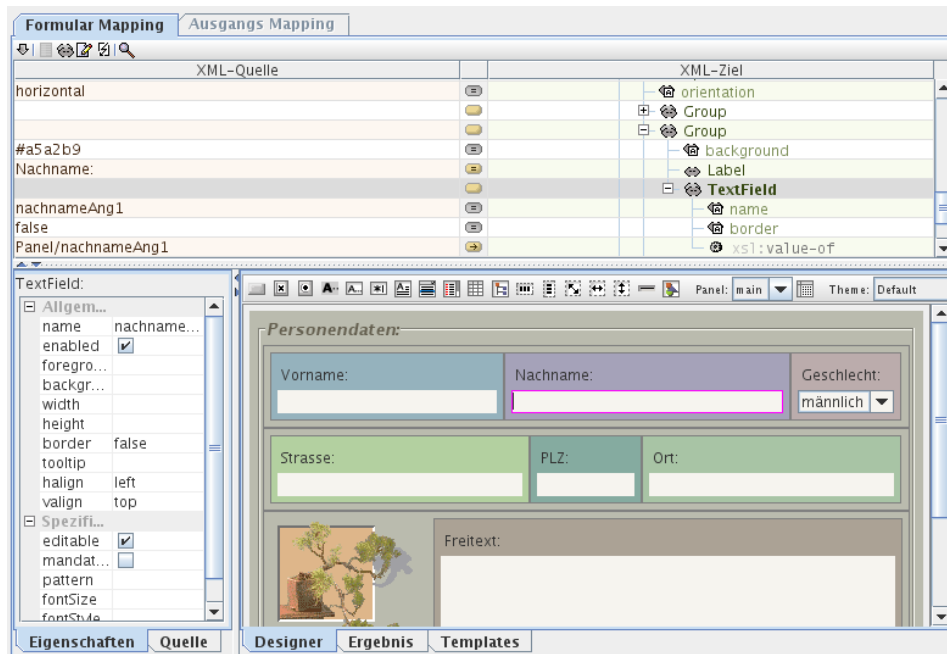


Abb. 6: Beispiel eines einfachen Formulars

Das Formular wird komfortabel über einen graphischen Formular Designer erstellt (Abb. 6). Die zur Verfügung stehenden Komponenten werden dabei über drag-and-drop in das Formular gezogen bzw. darin verschoben oder kopiert. Aus diesen Operationen entsteht ein vorgefülltes XSLT Skript, das nun nur noch in der gewohnten Art und Weise mit der in das Modul eingehenden Nachricht gemappt werden muss. Das Ergebnis des Formulars ist ebenfalls wieder XML.

Zusätzlich zu der bekannten Art aus dem Quelldokument in das Zieldokument zu mappen, können die Elemente aus dem Quelldokument hier auch direkt auf das graphische Formular – z.B. ein Textfeld gezogen werden. Die entsprechenden XSLT-Transformationen werden dann automatisch generiert.

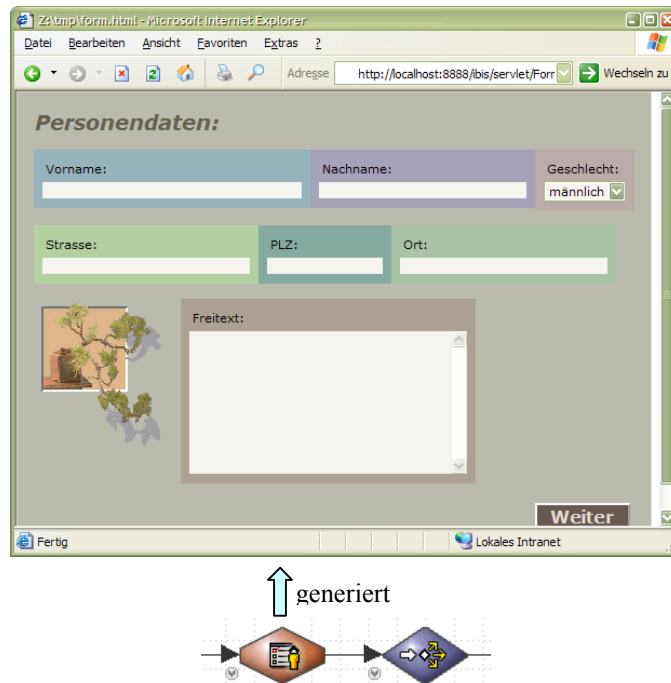


Abb. 7: Benutzerinteraktion im Workflow und die Anzeige im Browser

4 Ein Web Service für den Menschen

Der weiter oben dargestellte Web Service einer kundenindividuellen Preis Anfrage soll nun auch über eine Web-basierte Schnittstelle für einen Benutzer zur Verfügung gestellt werden (Abb.8).

An einer definierten URL wird dem Anwender eine Login-Seite zur Verfügung gestellt. Durch die Anmeldung wird der Workflow gestartet und ein erstes Formular für den Benutzer generiert, das ihm angezeigt wird.

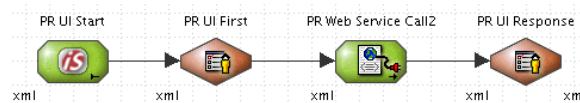


Abb. 8: Eingaben für Web Service und Response als Formulare

Er kann nun seine relevanten Daten wie gewohnt eingeben und das Formular abschicken. Die eingegebenen Formularfelder werden durch ein XSLT Mapping in den Web Service Call übersetzt. Das Ergebnis des Web Service wird dann wieder in ein folgendes Formular gemappt, und dem Benutzer werden seine kundenindividuellen Preise angezeigt.

Die komplette Webapplikation kann somit visuell durch einen Workflow beschrieben werden. Sämtliche Daten liegen dabei wieder in XML vor.

5 Eine Frage der Dynamik

Die Formulare müssen im obigen Beispiel für den Web Service im graphischen Formulardesigner von einem Anwender erstellt werden. Das ist sehr schnell erledigt und es kann somit auch auf Designfragen eingegangen werden. Außerdem kann hier aber auch auf dynamische Anteile der Web Service Beschreibung eingegangen werden, wie wir nun zeigen wollen.

Die Typbeschreibung des Service, die für das Mapping verwendet wird, zeigt die optionalen Typen oder Strukturen an. Auf diese kann nun beim Erstellen des Formulars eingegangen werden. Somit kann die volle Dynamik eines Web Services abgebildet werden. Diese könnte beispielsweise in der Übertragung von entweder einer Kundennummer oder einer kompletten Adresse liegen, die entsprechend der übertragenen Daten dynamisch abgefragt werden.

Diese Variabilität ist natürlich nicht nur auf die Formulare beschränkt. Sollte so eine Adresse beispielsweise auch gleich in eine Datenbank abgelegt werden, ist dieser Weg innerhalb des Workflows in Abhängigkeit der eingegangenen Daten ebenfalls leicht modellierbar.

6 Volle Dynamik

Im obigen Beispiel werden gewisse variable Teile eines Schemas (beziehungsweise des draus resultierenden Datenflusses) abgebildet. Der grundsätzliche Aufbau des Formulars ist aber fest und vom Benutzer entsprechend der Datenstruktur erstellt.

Es ist aber ebenfalls möglich, das komplette Formular dynamisch aus der Servicebeschreibung des Web Service zu generieren. Diese liegt in WSDL vor und enthält die Struktur in XML-Schema-formulierten Strukturbeschreibungen. In dem Schema liegt die volle syntaktische und ein wenig semantische Beschreibung der zu erwartenden Daten. Hieraus kann ein Formular generiert werden, das eine gute Abbildung erlaubt.

Es wird somit auf ein WSDL oder ein XML-Schema zugegriffen und daraus über XSLT ein Formular generiert. Mögliche Anpassungen – speziell im Design – sind nun noch durch den Benutzer im Formular Designer hinzufügbare.

Diese Transformation des WSDL/Schema zu einem generischer Client [GC01] kann einmalig erfolgen, es ist im inubit IS aber auch - wie im folgenden Beispiel eines dynamischen Antrags für das Versicherungswesen - möglich, das Interface bei jeder Ausführung des Workflows neu zu parsen und ein Formular zu erzeugen. Somit wird zu jeder Zeit dem aktuellen Interface entsprochen!

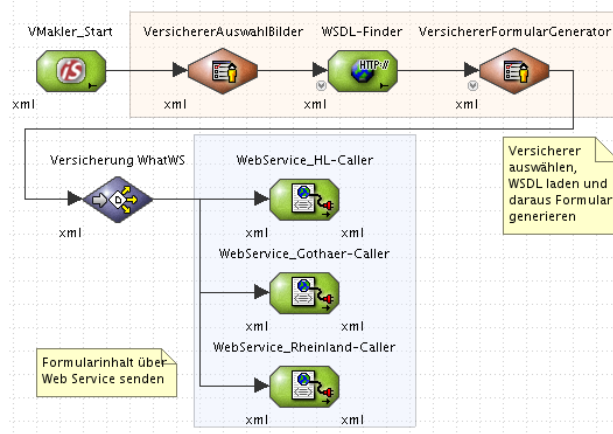


Abb. 9: Dynamisches Formular aus WSDL-Beschreibung

7 Fazit

Durch die Verwendung von XML innerhalb der Daten und Kontrolllogik der Workflows wird eine einheitliche Welt geboten, die sich auf weitere Standards dieses Umfelds wie XSLT und XPath stützt. Da sämtliche heterogene Systeme über eine XML Schnittstelle angesprochen werden, ist ein einheitliches und einfaches Handling über all diese Systeme hinweg möglich.

Ein weiterer Vorteil des durchgängigen Verbleibs in der XML-Welt liegt in der möglichen Fehlertoleranz, etwa bei der Verwendung von Web Services. Typprüfungen und Tests werden erleichtert und die Schnittstellen können dynamisiert werden.

Die Verwendung von Workflows zur Beschreibung der Business Prozesse erlaubt ein sehr schnelles Anpassen und Nachvollziehen von Änderungen dieser Prozesse hinein in die technische Welt der Integration und Portale.

Der innovative Ansatz des inubit IS erlaubt es somit, flexibel auf hoch dynamische Prozesse einzugehen und sehr effizient und homogen mit verschiedensten Systemen umzugehen.

8 Links

[IS01] inubit IS – <http://www.inubit.com>

[EAI01] EAI - <http://www.eaiforum.de>

[SOA01] SOA - http://de.wikipedia.org/wiki/Service_Oriented_Architecture

[BPEL01] Business Process Execution Language – <http://www.oasis-open.org>

[XUL01] Open XUL Alliance - <http://xul.sourceforge.net>

[GC01] Generischer Client

XML Application Infrastructure (University of California, Berkeley) -
<http://groups.sims.berkeley.edu/CDE/report/go-five-final-report.html>

IBM Alphaworks XML Forms Generator -
http://www.alphaworks.ibm.com/tech/xfp?open&S_TACT=105AGX59&S_CMP=GR&ca=dgr-eclpsw02awxfg

OASIS SOA Reference Model - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

[XML01] Spezifikationen zu XML, XSLT/XPath, XML-Schema, WSDL/SOAP -
<http://www.w3.org>

[J01] Java – <http://java.sun.com>

[MS01] .NET – <http://www.microsoft.com>