P. Freudenstein, F. Majer, A. Maurer; SOA in der Praxis - eine Referenzarchitektur; Journal Article in dot.net-magazin, Issue 11/2006, Pages 22-27, Software & Support Verlag, Frankfurt, Germany, ISSN: 1619-7933.

SOA in der Praxis

Eine Referenzarchitektur für erfolgreiche Enterprise-Application-Integration-Projekte mit neuesten Technologien und Plattformen

von Patrick Freudenstein, Frederic Majer und Axel Maurer

Viele reden von SOA, aber keiner macht es. Laut einer Studie von Pierre Audoin Consultants (PAC) werden in Deutschland nur 14 Prozent aller Investitionen in Integrationsprojekte für SOA aufgewendet, gerade einmal ein Prozent aller EAI-Projekte verfolgen einen umfassenden SOA-Ansatz.

Der Grund, warum SOA in der Praxis noch keine große Rolle spielt, mag zum einen etwas mit einem Mythos zu tun haben: SOA = SOAP und SOAP = langsam. Diese Fehleinschätzung wird alleine durch die neue *Windows Communication Foundation* (WCF) des kommenden .NET Framework 3.0 entkräftet. Zum anderen daran, dass der Mehrwert, den ein umfassender SOA-Ansatz bringt, nicht gleich im ersten Projekt sichtbar wird und zum dritten, dass zu SOA mehr gehört als nur ein paar Web Services in Betrieb zu setzen. Die Einführung einer SOA-basierten Lösung setzt häufig auch organisatorische Umstellungen voraus. SOA bietet die Möglichkeit, die Prozesse im Unternehmen flexibel zu gestalten und auf notwendige Änderungen in kürzester Zeit zu reagieren. Dazu ist es aber notwendig die Geschäftsprozesse zu kennen und sie auch zu leben. Das bedeutet beispielsweise, dass Änderungen explizit vorgenommen und dokumentiert werden und man nicht „einfach so mal eben nebenbei" alles ändert, ohne dass es dokumentiert wird und in den unterstützenden Systemen seinen Niederschlag findet.

kurz & bündig

Inhalt

Eine Referenzarchitektur für Integrationsvorhaben in Unternehmen unter Verwendung neuester Microsoft-Technologie am Beispiel des KIM-Projekts der Universität Karlsruhe (TH)

Zusammenfassung

Auch wenn ein SOA-Ansatz bei heutigen Projekten noch selten ist, bringt er viele Vorteile. In der Microsoft-Welt stehen mit WCF, BizTalk Server und SharePoint Server 2007 die erforderlichen Bausteine zur Verfügung, um eine leistungsfähige serviceorientierte Architektur zu implementieren

Das KIM Projekt der Universität Karlsruhe (TH)

Die *Universität Karlsruhe (TH)* hat einen umfassenden Ansatz zur Einführung einer SOA auf technischer und organisatorischer Ebene entwickelt und ist derzeit dabei, diesen in dem Projekt *Karlsruher Integriertes InformationsManagement (KIM)* für die komplette Informationsversorgung und -verarbeitung umzusetzen. In der ersten Phase werden die Geschäftsprozessunterstützung der Lehre und ein dienstebasiertes Identity Management implementiert. Basis der technischen Umsetzung – und nur um diese soll es hier gehen – ist die *integrierte SOA* (iSOA), die auf einem 4-Schichten-Modell beruht (Abbildung 1). Ausgehend von der Erfassung und Modellierung der Geschäftsprozesse werden Anwendungsdienste abgeleitet, die in Orchestrierungen von Basisdiensten und den zugehörigen

P. Freudenstein, F. Majer, A. Maurer; SOA in der Praxis - eine Referenzarchitektur; Journal Article in dot.net-magazin, Issue 11/2006, Pages 22-27, Software & Support Verlag, Frankfurt, Germany, ISSN: 1619-7933.

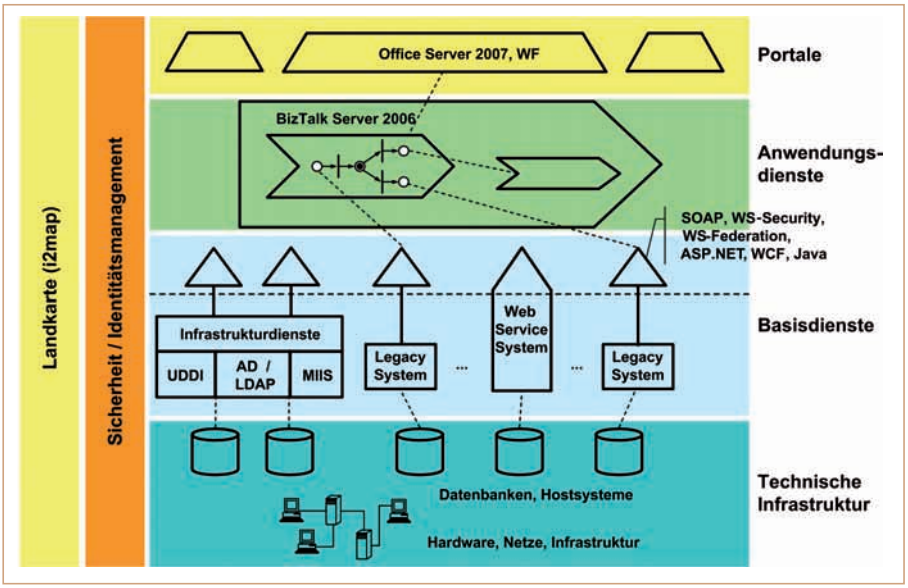

Abb. 1: Die integrierte serviceorientierte Architektur im Überblick

Benutzerinteraktionen abgebildet werden. Diese Orchestrierungen werden im BizTalk Server 2006 implementiert. Dadurch ist es möglich eine Pattform zur Verfügung zu stellen, die Interoperabilität zwischen den Systemen gewährleistet und gleichzeitig eine einheitliche Sicht auf die Geschäftsprozesse ermöglicht, auch dann, wenn sie über Einrichtungen, wie z.B. Fakultäten, Bibliothek und Verwaltung, hinweg reichen. Es wird also kein klassisches EAI-System eingesetzt, sondern der *Enterprise Service Bus* entsteht dadurch, dass alle vorhandenen Systeme mit Schnittstellen versehen werden, die sich auf Ebene der Basisdienste „iSOA-konform" verhalten.

Basisdienste – die Bausteine einer serviceorientierten Architektur

Die datenzentrierten *Basisdienste* in Form von Web Services stellen die grundlegenden Bausteine der iSOA dar. Sie kapseln eine jeweils klar definierte, semantisch stark zusammenhängende Menge an Geschäftsobjekten und bieten Operationen zu deren Erzeugung, Modifikation und Abfrage an. Um bestehende Systeme, sog. *Legacy-Systeme*, bzw. deren Daten in die iSOA zu integrieren, fungieren Basisdienste oft als sog. *Wrapper*. Darüber hinaus existieren auch Basisdienste, die Schnittstellen zu neu entwickelten Web-Service-basierten Systemen anbieten. Wrapper setzen auf einem oder mehreren Legacy-Systemen auf und bieten plattformunabhängige und wohldefinierte Schnittstellen für den Zugriff auf die Daten der darunter liegenden Systeme an. Abhängig von der Plattform und den angebotenen (oft plattformabhängigen) Schnittstellen des zu integrierenden Legacy-Systems bietet es sich an, den Wrapper-Web-Service auf der gleichen Plattform zu entwickeln. So wird in der Basisdienst-Schicht häufig eine Vielfalt verschiedener Technologien, von ASP.NET / Windows Communication Foundation (WCF) über Java bis hin zu PHP, eingesetzt. Oberhalb der Basisdienst-Schicht spielen die zur Implementierung der Basisdienste eingesetzten Technologien keine Rolle mehr, da Web Services aufgrund der zu Grunde liegenden Standards (SOAP, WSDL, XML) grundsätzlich plattformunabhängig sind – ein großer Vorteil in serviceorientierten Architekturen.

Bei der Entwicklung dieser Dienste gibt es allerdings große Unterschiede hinsichtlich der angebotenen Unterstützung durch die verwendete Entwicklungsplattform. Damit ist nicht nur der Komfort für den Entwickler durch umfangreiche und möglichst einfach zu verwendende Klassenbibliotheken gemeint, sondern insbesondere auch inwieweit Standards im Web-Service-Umfeld derzeit überhaupt bzw. standardkonform implementiert sind. Unter beiden Gesichtspunkten bieten ASP.NET bzw. die Windows Communication Foundation die besten Voraussetzungen, wohingegen beispielsweise bei PHP derzeit keine korrekte Implementierung von WS-Security verfügbar ist. Auch im Java-Umfeld stellt sich die Web-Service-Entwicklung deutlich aufwändiger als mit ASP.NET / WCF dar. Vor dem eigentlichen Entwicklungsbeginn fällt hier zunächst die recht aufwändige Evaluation und Zusammenstellung geeigneter standardkonformer Bibliotheken für die verschiedenen Aspekte der Web-Service-Entwicklung (XML-Serialisierung und Deserialisierung, Bibliotheken zur Unterstützung der verschiedenen WS-*-Standards etc.) an.

Aus Sicht der Theorie sind Web Services per se interoperabel und plattformunabhängig, da sie sich auf plattformunabhängige, XML-basierte Standards stützen. In der Praxis allerdings sind Interoperabilitätsprobleme zwischen Web Services, die auf verschiedenen Plattformen entwickelt wurden, nicht selten. Häufige Problembereiche sind plattformspezifische Datentypen, wie zum Beispiel eine .NET-Hashtable oder ein Java-Vektor, zu denen es keine entsprechenden Typen in der *XML Schema Language* gibt sowie unterschiedliche Auslegungen der teilweise unklaren Standard-Spezifikationen durch unterschiedliche Entwicklungsplattformen. Um dennoch Interoperabilität zu gewährleisten sind zwei Punkte zu beachten: Zum einen sollte bei der Web-Service-Entwicklung der WSDL-First-Ansatz gewählt werden. Zuerst sollte die WSDL-Beschreibung mitsamt aller Datentypen – basierend auf der *XML Schema Language* – erstellt und erst anschließend darauf basierend Client- *und* Service-Skelette generiert werden. Zum anderen sind die Testing Tools der Web-Services-Interoperability-Initiative (WS-I, www.ws-i.org) sehr zu empfehlen. Damit können sowohl die WSDL-Beschreibungen als auch der gesamte Nachrichtenverkehr auf Interoperabilitätsprobleme hin untersucht werden.

Aus Effizienzüberlegungen ist die Wiederverwendung eines der wichtigsten Prinzipien in serviceorientierten Architekturen. Der Schwerpunkt beim Entwurf und der Entwicklung von Basisdiensten liegt daher auf ihrer Wiederverwendbarkeit in möglichst vielen, auch zukünftigen, Anwendungsszenarien. Dabei sind insbesondere zwei Aspekte von zentraler Bedeutung: Die Granularität der Basisdienste sowie klar definierte und möglichst einheitliche Schnittstellen.

P. Freudenstein, F. Majer, A. Maurer; SOA in der Praxis - eine Referenzarchitektur; Journal Article in dot.net-magazin, Issue 11/2006, Pages 22-27, Software & Support Verlag, Frankfurt, Germany, ISSN: 1619-7933.

Hinsichtlich der Granularität bietet sich auf der Basisdienst-Schicht eine datenzentrierte Sichtweise an. Das bedeutet, je ein Basisdienst für eine semantisch stark zusammenhängende Menge an Geschäftsobjekten. Bei der Zuordnung hilft es, die Geschäftsobjekte nach dem „Was?“ (um was für Daten handelt es sich) und nicht nach dem „Wie?“ (wie werden die Daten verwendet) zu untersuchen. So werden Daten über den Studiumsverlauf und persönliche Daten zu Studierenden zwar in verschiedenen Kontexten („Wie?“) verwendet, im Grunde sind aber beides Daten über Studierende („Was?“) und werden von einem Basisdienst verwaltet.

Die CRUDS-Schnittstelle als Grundlage

Klar definierte und einheitliche Schnittstellen erhöhen die Wiederverwendbarkeit von Basisdiensten und schaffen die Grundlage für effiziente generische Konzepte zu deren Verwendung auf höheren Architekturebenen. Bei datenzentrierten Diensten bietet sich die so genannte CRUDS-Schnittstelle an (siehe Kasten "Die CRUDS-Schnittstelle"). So wird beispielsweise ein Basisdienst, der Rauminformationen verschiedenster Art verwaltet, in unterschiedlichsten Anwendungsszenarien von der Raumsuche über die Erstellung von Raumbelegungsplänen bis hin zur Steuerung der Klimatisierung wieder verwendet.

Anwendungsdienste

Die *Anwendungsdienste* realisieren diese Geschäftsprozesse, indem sie die wieder verwendbaren Bausteine der *iSOA*, die Basisdienste, zu höheren, prozessorientierten Diensten verknüpfen. Die Basisdienste werden mit dem Microsoft BizTalk Server zu einer Orchestrierung komponiert und als Web Service zur Verfügung gestellt, wodurch sie wiederum in Orchestrierungen anderer Geschäftsprozesse genutzt werden können. Zur Umsetzung eines Anwendungsdienstes wird in einer Vorstufe der gesamte Ablauf des Geschäftsprozesses mit all seinen Informationsflüssen über die verschiedenen IT-Systeme in Form von Petri-Netzen modelliert. Diese einfache grafische Aufbereitung der Prozesse sowie der Einsatz von geeigneten Werkzeugen zur Simulation der Ablaufmodelle stellen die Grundlage für die Analyse und Optimierung der Geschäftsprozesse dar. In diesem Zusammenhang hat sich die Nutzung von hierarchischen Petri-Netzen bewährt, da durch die mehrstufige Aufbereitung und die damit einhergehende klare Strukturierung der Prozesse in unterschiedlicher Granularität geeignete Sichten für einzelne Benutzergruppen erstellt werden können.

Wenn die Geschäftsprozesse ausreichend konkret erfasst sind, beinhaltet die vorletzte Detailstufe die Interaktionsschritte des Benutzers mit dem Anwendungsdienst. Diese Modelle werden bei der Entwicklung der Portalkomponenten, die die Schnittstelle zwischen dem Anwender und den Geschäftsprozessen darstellen, eingesetzt. Im höchsten Detaillierungsgrad – auf unterster Ebene in den hierarchischen Petri-Netzen – finden sich die konkreten, der Benutzerinteraktion zu Grunde liegenden Operationsaufrufe in den jeweiligen IT-Systemen und die hierbei ausgetauschten Geschäftsobjekte wieder. Diese Ablaufmodelle stellen die Grundlage für die Entwicklung der Anwendungsdienste nach dem Baukastenprinzip dar – wie in Abbildung 2 dargestellt wird. Es werden die BizTalk-Orchestrierungen durch die Verschaltung der benötigten Funktionen, die durch die standardisierten Operationen der Basisdienste erbracht werden, erstellt. Aufgrund des Einsatzes dieser lose gekoppelten Bausteine der Basisdienstschicht sowie der Dokumentation der Geschäftsprozesse in Form von Petri-Netzen können die Anwendungsdienste schnell und flexibel an sich verändernde Anforderungen der Geschäftsprozesse angepasst werden.

Die CRUDS-Schnittstelle

CRUDS steht für die Anfangsbuchstaben der in der Schnittstelle definierten Operationen:

```
public XmlElement Create(XmlElement prototype);
public XmlElement Read(XmlElement readContext);
public XmlElement Update(XmlElement updateContext, XmlElement businessObject);
public XmlCollection UpdateBulk(UpdateBulkCollection updateBulkContext);
public XmlElement Delete(XmlElement deleteContext);
public XmlCollection Search(XmlElement searchContext);
public XmlElement GetServiceCard(XmlElement serviceCardContext);
```

Die CRUDS-Methoden dienen zum Anlegen, Lesen, Ändern, Löschen und Suchen von Daten. Darüber hinaus dient die Methode *GetServiceCard* zur Abfrage statischer Metainformationen sowie dynamischer Laufzeitinformationen (die sog. *ServiceCard*). Um einerseits eine stabile Schnittstelle zu gewährleisten und andererseits dennoch sich ergebende Änderungen vornehmen zu können, basiert die Schnittstelle weitgehend auf dem generischen *XmlElement*-Typ von .NET, d.h. beliebigem XML. Die *XmlCollection* ist nichts anderes als ein *XmlElement*-Array. Dadurch wird Polymorphie auf Web-Service-Schnittstellenbasis ermöglicht, d.h. sowohl die Rückgabewerte (meist Geschäftsobjekte) als auch die Parameter können in mehreren Ausprägungen realisiert sein, wobei XML-Namespaces als eindeutige Schlüssel fungieren.
Ein sog. *Kontext* kapselt die Parameter einer Methode und gibt anhand eines Verweises auf einen Namespace an, in welcher Form (*OutputSchema*), d.h. basierend auf welchem XML-Schema, die Daten zurückgeliefert werden sollen:

```
<ReadContext xmlns="...">
 <ISID>
  <Identifier>79211</Identifier>
  <ServiceId>8CEA84E6-5AE0-4506-9288-E9292275B9C3</ServiceId>
 </ISID>
 <OutputSchema>http://schemas.kim.uni-karlsruhe.de/types/2005/07/EventType.xsd:EventDetailedType
                                                                    </OutputSchema>
</ReadContext>
```

Sind Änderungen an den Parametern notwendig, können neben den Standardkontexten zusätzliche speziellere Kontexte eingeführt werden. Die *ServiceCard* gibt Auskunft über die vom Basisdienst verarbeitbaren Kontexte und Datenschemas.

P. Freudenstein, F. Majer, A. Maurer; SOA in der Praxis - eine Referenzarchitektur; Journal Article in dot.net-magazin, Issue 11/2006, Pages 22-27, Software & Support Verlag, Frankfurt, Germany, ISSN: 1619-7933.

Das SharePoint-Portal – die Schnittstelle zu den Anwendern

Die Portalschicht ermöglicht über geeignete Benutzerschnittstellen den Anwendern den Zugriff auf die Basis- und Anwendungsdienste der darunter liegenden Schichten. Neben klassischen Portalen und Webanwendungen sind in dieser Schicht auch sonstige Client-Applikationen, z.B. Office-Produkte, denkbar. Im Folgenden wird die Portalentwicklung mit dem Microsoft Office SharePoint Server 2007 Beta 2 (MOSS) näher beleuchtet. Neben der Entwicklung des Layouts mithilfe von ASP.NET Master Pages, MOSS Layout Templates, selbst entwickelter Navigationskomponenten sowie dem Aufbau der Portalstruktur, ist in einer SOA besonders die Integration der Dienste ins Portal von Interesse.

In Anbetracht der Vielzahl an Diensten ist ein effizienter, möglichst generischer und wieder verwendbarer Lösungsansatz wünschenswert, um so auch zukünftig neue Dienste schnell ins Portal integrieren zu können. Das in MOSS enthaltene *Business-Data-Catalog*-Konzept (BDC) eignet sich nur für die simple Anzeige von Web-Service-Daten im Portal. Eine geeignete Integrationslösung muss jedoch die vielen Varianten an Integrationsszenarien abdecken. Diese können vom simplen „Web Service aufrufen – Daten anzeigen" bis hin zu komplexen Formular-, Datenpräsentations- und Web-Service-Aufruf-Abfolgen reichen. Als gemeinsamer Nenner bietet sich daher die Beschreibung der Integrationsszenarien anhand von Workflows mithilfe wieder verwendbarer Workflow-Bausteine an. Hier kommt die Windows Workflow Foundation (WF) aus dem kommenden .NET Framework 3.0 ins Spiel. Bei der WF handelt es sich um ein erweiterbares Framework, das ein Workflow-Modul, ein .NET-verwaltetes API, Laufzeitdienste und einen visuellen Designer und Debugger, integriert in Microsoft Visual Studio 2005, beinhaltet.

Abbildung 3 zeigt die Architektur der Workflow-basierten Integrationslösung im Überblick. Die unterste Schicht beinhaltet die zu integrierenden CRUDS-Web-Services. Auf der darüber liegenden Schicht sind Zustandsautomaten-Workflows angesiedelt, die das Integrationsszenario beschreiben und aus Workflow-Bausteinen zusammengesetzt werden.

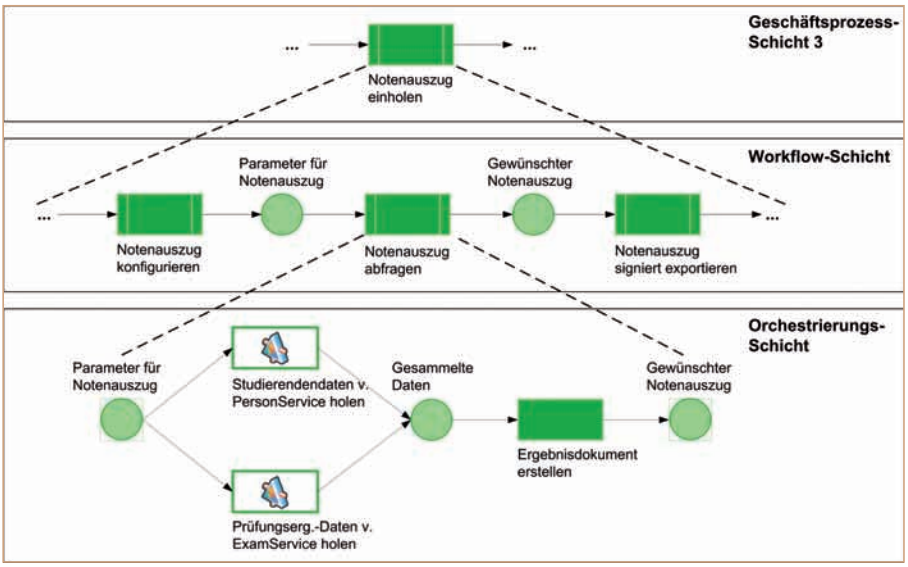

Abb. 2: Mehrstufige Geschäftsprozessmodellierung als Grundlage für Orchestrierungen

Die Zustände eines solchen Workflows entsprechen einzelnen Benutzerdialogen, zwischen denen der Benutzer anhand von Ereignissen (z.B. durch Anklicken eines Buttons oder Links) wechseln kann. Bei der Integration eines Veranstaltungsbasisdienstes in Form einer Veranstaltungssuche könnte der erste Zustand beispielsweise ein Suchformular, der zweite eine Trefferliste und der dritte Zustand Detailinformationen zu einer zuvor gewählten Veranstaltung repräsentieren. Beim Eintritt in einen Zustand wird eine Folge von Aktivitäten ausgeführt und anschließend auf ein neues Ereignis gewartet. Hierfür kann aus der Menge der mit der WF mitgelieferten oder selbst entwickelten Aktivitäten (sog. „Custom Acitivites") gewählt werden. Für die meisten Szenarien bietet sich die Entwicklung je einer Custom Activity für die Web-Service-Kommunikation (*InvokeCRUDS*), das Formular-Rendering (*ConstructForm*) und die Datenpräsentation (*RenderMarkup*) an.

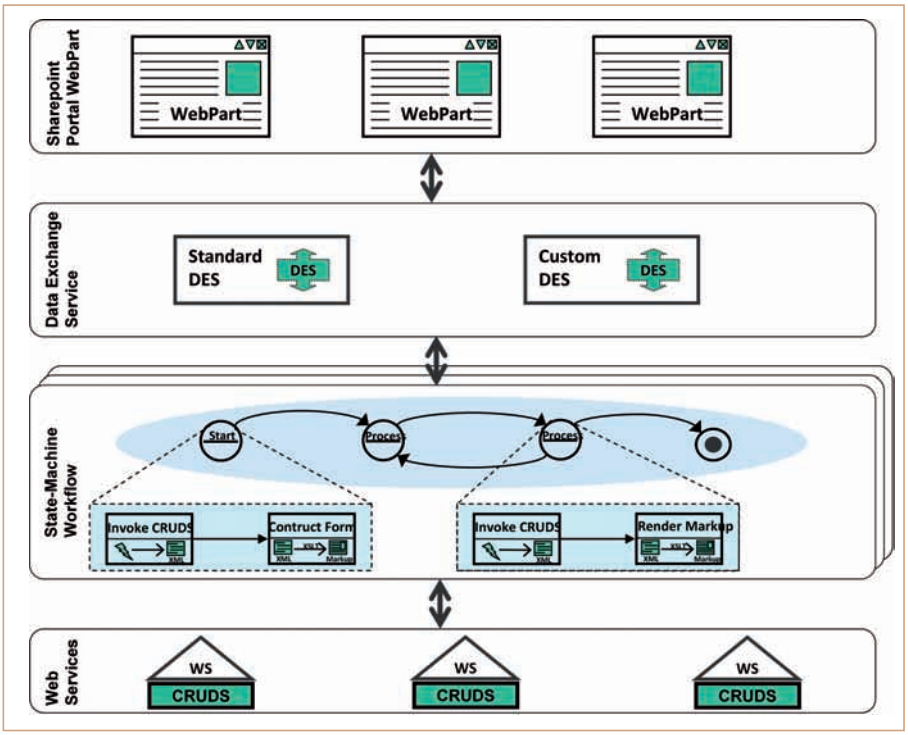

Abb. 3: Architektur der Workflow-basierten Dienstintegration in Portale

P. Freudenstein, F. Majer, A. Maurer; SOA in der Praxis - eine Referenzarchitektur; Journal Article in dot.net-magazin, Issue 11/2006, Pages 22-27, Software & Support Verlag, Frankfurt, Germany, ISSN: 1619-7933.

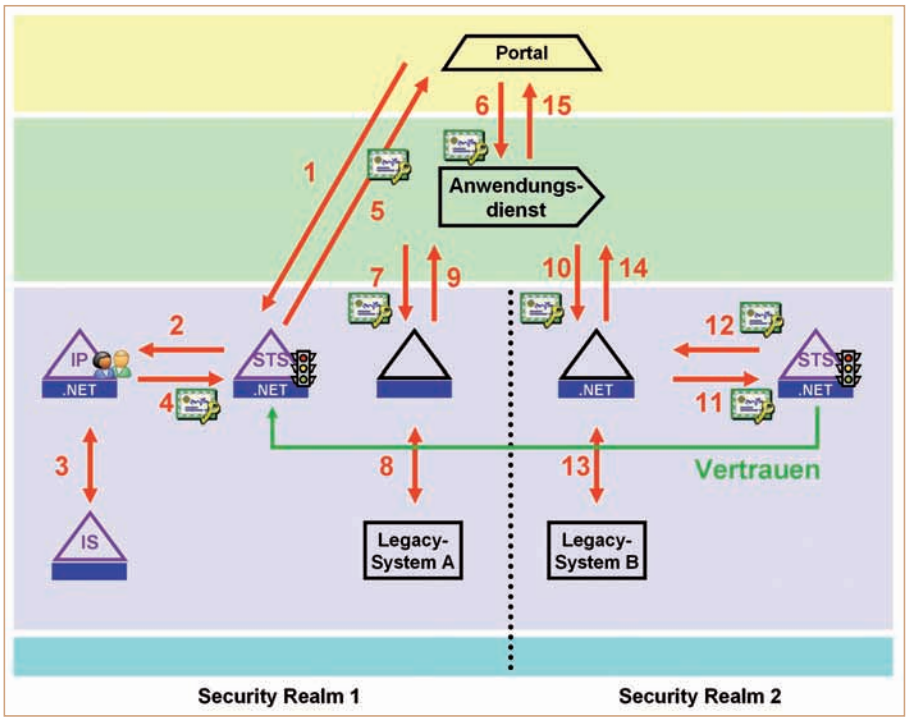

Abb. 4: Der Ablauf des föderativen Sicherheitskonzepts im Überblick

Für die Kommunkation mit CRUDS-Web-Services wird die *InvokeCRUDS*-Activity eingesetzt. Hierfür sind die URL, die aufzurufende Operation sowie die Aufrufparameter, d.h. der CRUDS-Kontext, konfigurierbar. Innerhalb des Kontexts kann auf Variablen aus dem HTTP-Request, d.h. beispielsweise Eingaben in Formularfeldern oder die Session zurückgegriffen werden. Das Ergebnis des Aufrufs, ein XML-Dokument, steht dann für andere Aktivitäten zur Weiterverarbeitung bereit. Die *InvokeCRUDS*-Activity lässt sich einfach auf andere Schnittstellentypen anpassen und beispielsweise durch Auswertung des WSDL-Dokuments eines Web Services auch auf beliebige Schnittstellentypen ausweiten.

Um qualitativ hochwertige Formulare zur Benutzerinteraktion zu gewährleisten, werden diese nicht manuell implementiert, sondern von der *ConstructForm*-Activity basierend auf einer XML-Beschreibung automatisch generiert. Dabei können auch Werte aus dem Ergebnisdokument eines vorangegangenen Web-Service-Aufrufs referenziert werden, um beispielsweise Dropdown-Listen zu befüllen. Durch die automatische Generierung wird insbesondere die konsequente Durchsetzung von Richtlinien bezüglich des Corporate Designs oder des barrierefreien Zugriffs für Menschen mit Behinderungen erreicht.

Die *RenderMarkup*-Activity wird zur Transformation von XML-Dokumenten mittels XSLT nach (X)HTML verwendet. Als Eingabeparameter sind das Ergebnis-XML-Dokument einer vorherigen *InvokeCRUDS*-Activity sowie der Pfad zu einem XSLT-Dokument konfigurierbar – das Ergebnis besteht in einem (X)HTML-String.

Die Architekturschicht, die darüber liegt, beinhaltet die so genannten *Data Exchange Services* (DES). Sie vermitteln die Kommunikation zwischen dem Workflow und dem ihn abwickelnden WebPart. Da zum Entwicklungszeitpunkt des Workflows dessen Kommunikationspartner unbekannt ist, sind die DES zur Entkopplung notwendig. In den meisten Fällen reicht ein Standard-DES aus, der eine Grundmenge an Methoden und Ereignissen zur bilateralen Kommunikation definiert. Der Workflow kann beim DES Methoden aufrufen, die dann über Ereignisse an Handler, in diesem Fall das WebPart, weitergeleitet werden. Umgekehrt kann das WebPart über den DES Ereignisse auslösen, die an den Workflow weitergereicht werden.

Zur Integration der Workflows ins Portal wurde ein universelles WebPart, das lediglich eine Referenz zur Workflow-Assembly im Global Assembly Cache als Parameter erhält und dann dessen Abwicklung übernimmt, entwickelt. Die Integration der Basis- und Anwendungsdienste funktioniert nun ganz einfach in zwei Schritten. Zuerst wird mithilfe der drei Custom Acitivities ein geeigneter Workflow zusammengebaut. Dann noch „schnell“ auf einer Portalseite das WebPart hinzufügen und den Verweis auf den abzuwickelnden Workflow konfigurieren – fertig ist die Integration.

Die Sicherheitsinfrastruktur

Universitäten, ebenso wie große Unternehmen, verfügen über eine doch stark dezentralisierte und verteilte Organisationsstruktur, woraus häufig eine technologisch heterogene Sicherheitsinfrastruktur resultiert. Um dennoch ein organisationsweites Single-Sign-On zu realisieren, ist ein föderatives Konzept für Authentifizierung und Autorisierung notwendig. Im SOA-Kontext existieren hierfür verschiedene Standards, unter anderem *WS-Federation* von IBM, Microsoft, BEA Systems, RSA Security und Verisign. Föderative Ansätze erfordern die Abbildung der Organisationsstruktur auf „Security Realms“ mit jeweils unterschiedlichen Sicherheitsanforderungen und -infrastruktur. Der Vorteil eines föderativen Ansatzes ist, dass jeder Nutzer nur einen einzigen Account in einem beliebigen Security Realm besitzen muss, um Dienste in allen Realms nutzen zu können.

WS-Federation sieht innerhalb eines Realms zwei Kernkomponenten in Form von Web Services vor (Abbildung 4): Den *Identity Provider (IP)* zur Authentifizierung sowie den *Security Token Service (STS)* zur Autorisierung. Der IP stellt nach erfolgreicher Authentifizierung ein Token aus, das beim STS vorgelegt werden kann, um ein um Realm-spezifische Rollen erweitertes Token zu erhalten. Zwischen den STS verschiedener Realms bestehen Vertrauensverhältnisse, sodass ein STS mittels konfigurierbarer Rollen-Mappings ein Token aus einem anderen Realm in ein für den eigenen Realm gültiges Token transformieren kann. Die Basis- und Anwendungsdienste erfordern beim Aufruf das Mitsenden eines solchen Tokens und prüfen anhand der dort spezifizierten Rollen und ggf. weiterer

P. Freudenstein, F. Majer, A. Maurer; SOA in der Praxis - eine Referenzarchitektur; Journal Article in dot.net-magazin, Issue 11/2006, Pages 22-27, Software & Support Verlag, Frankfurt, Germany, ISSN: 1619-7933.

Attribute die Berechtigungen auf Operations- und Datenebene. Die gesamte Kommunikation läuft gesichert auf Basis von WS-Security ab.

Die Windows Communication Foundation (WCF) vereinfacht die Entwicklung der beteiligten Dienste erheblich. Mit dem *WsFederationBinding* ist sogar eine explizite Unterstützung zur Realisierung von WS-Federation enthalten. Die *Active Directory Federation Services (ADFS)* in Windows Server 2003 R2 stellen bisher leider nur eine rudimentäre Implementierung von WS-Federation dar.

Landkarte: Unterstützung für Entwicklung, Betrieb und Wartung

Der Name lässt schon erahnen, worum es sich bei diesem Aspekt handelt. Die *Landkarte* bietet einen Überblick über alle Komponenten der *iSOA* sowie deren Beziehungen untereinander, um Entwicklung, Betrieb und Wartung der gesamten Systemlandschaft zu unterstützen. Dabei stellt sie insbesondere zur Laufzeit maschinenlesbare Systembeschreibungen sowie Ist-Daten des Systemzustandes zur Verfügung. Bedingt durch die Komplexität des stark verteilten Systems und dessen stetige Evolution ergeben sich für einzelne Benutzergruppen verschiedene Fragestellungen und Wünsche in Bezug auf die Landkarte. Beispielsweise benötigen Portaladministratoren einen Überblick welche Dienste neu entwickelt wurden bzw. welche Dienste auf welche Art und Weise in ein Portal eingebunden werden können. Für die Betreiber der *iSOA* sind im Bezug auf Systemausfälle und Wartungsarbeiten die Abhängigkeiten zwischen den Basis- und Anwendungsdiensten sowie der aktuelle Status der Komponenten relevant. Um die Neuentwicklung bzw. Evolution von Komponenten unter Nutzung bestehender Komponenten bestmöglich zu unterstützen, sind für die Entwickler Sichten auf föderations- und sicherheitsspezifische Aspekte der massiv verteilten Systemlandschaft von Interesse.

Aus den unterschiedlichen Beispielszenarien wird ersichtlich, dass Meta- und Statusinformationen über Komponenten aller vier Integrationsschichten benötigt werden. Basierend auf den spezifischen Charakteristika der einzelnen Schichten wurden hierfür dedizierte Modelle definiert, die unter anderem auch die Beschreibung von schichtenübergreifenden Beziehungen ermöglichen. Zusammenfassend wird die Menge aller Modelle als das *System Description Framwork (SDF)* bezeichnet und bildet die Grundlage für die Landkartenfunktionalitäten.

Zur Laufzeit werden – passend zu den jeweiligen Modellen – die relevanten Informationen durch mehrere Infrastrukturdienste zur Verfügung gestellt. Diese Basisdienste implementieren ebenfalls das CRUDS-Interface, wodurch sie eine standardisierte Schnittstelle für die Abfrage und Modifikation der verwalteten Daten bieten. Zumeist werden die auf die jeweiligen Problemfälle der Benutzergruppen zugeschnittenen Sichten auf die notwendigen Informationen durch die Verschaltung solcher Infrastrukturdienste erreicht. Die hieraus resultierenden Anwendungsdienste (sog. *Landkartendienste*) bieten die Informationen in aggregierter Form an und können in verschiedene Management-Portale eingebunden werden.

Beispielsweise wurde ein Landkartendienst entwickelt, der sich mit der Überwachung in Geschäftsprozessen eingebundener Basisdienste befasst. In einem ersten Schritt nutzt dieser Dienst einen Infrastrukturdienst, der die Abfrage des aktuellen Status eines Basisdienstes ermöglicht und diese Informationen verwaltet. Wenn ein ausgefallener Basisdienst identifiziert wurde, findet eine Anfrage an einem weiteren Infrastrukturdienst, der Daten zu den Geschäftsprozessen zur Verfügung stellt, statt. Die dort gehaltenen Informationen aus den modellierten Petri-Netzen geben Aufschluss über den Einsatz des ausgefallenen Basisdienstes in Prozessen, wodurch der Landkartendienst eine Liste aktuell beeinträchtigter Geschäftsprozesse liefert.

Erfolgsaussichten für SOA

Der SOA-Ansatz bietet den Vorteil innerhalb eines Projektes kurzfristige Erfolge zu erzielen, ohne den Blick auf das Gesamtbild zu verlieren.

SOA ist also auch in der Microsoft-Welt machbar. Wie in diesem Artikel am Beispiel des Hochschulprojektes *Karlsruher Integriertes InformationsManagement (KIM)* der Universität Karlsruhe (TH) gezeigt wurde, ist ein umfassender Ansatz auch dann möglich, wenn die zu Grunde liegenden Systeme noch nicht über die entsprechenden Schnittstellen verfügen. Natürlich ist der damit verbundene Aufwand deutlich höher. Daher wird bei zukünftigen Software-Ausschreibungen der Universität die Unterstützung von Web Services ein wesentliches Auswahlkriterium sein. Die Produktpalette von Microsoft bietet mit dem kommenden .NET Framework 3.0, dem BizTalk Server 2006 und dem ebenfalls in Kürze erhältlichen Microsoft Office SharePoint Server 2007 die besten Voraussetzungen eine durchgängige serviceorientierte Architektur zur Unterstützung von Geschäftsprozessen zu realisieren. Der SOA-Ansatz bietet den Vorteil innerhalb eines Projektes kurzfristige Erfolge zu erzielen, ohne den Blick auf das Gesamtbild zu verlieren. So wurde innerhalb kürzester Zeit die Anbindung des Veranstaltungsmanagements an die zentrale Klima- und Heizungstechnik über einen Basisdienst realisiert, der mittlerweile auch der Suche nach Veranstaltungen für die Studenten dient. Bei diesem Dienst werden die Raumbelegungen aus dem Veranstaltungskalender ausgelesen und zur Steuerung der Heizung und der Klimatisierung genutzt, was zu einer deutlich fünfstelligen Einsparung an Energiekosten geführt hat.

Sicherlich sind auch bei diesem Ansatz noch einige Fragen offen. So beispielsweise, wie das Ganze skaliert, wenn es den Pilotstatus verlässt und die zentrale Informations- und Kommunikationsplattform für über 30.000 Nutzer darstellen wird. Diese Frage zu beantworten wird aber erst dann möglich sein, wenn alle Produkte den Betastatus verlassen haben. Auch wenn die Universität sich durchaus gegenüber neuen Entwicklungen offen zeigt, sollte die Software für den produktiven Einsatz eine gesicherte Stabilität erreicht haben.

Patrick Freudenstein und **Frederic Majer** sind Doktoranden an der Universität Karlsruhe, **Axel Maurer** ist freiberuflicher Projektmanager. Gemeinsam arbeiten Sie im Projekt „Karlsruher Integriertes Informationsmanagement“. Sie erreichen sie unter nachname@kim.uni-karlsruhe.de.

Links & Literatur

[1] www.kim.uni-karlsruhe.de
[2] mwrg.tm.uni-karlsruhe.de
[3] www.netfx3.com
[4] blogs.msdn.com/sharepoint