Seminararbeit
zum Thema

Tipps zur Integration von XML & Web Services

des Seminars

Serviceorientierte Softwarearchitektur

im WS 05/06 an der FH Wedel

von:

Torben Deumert

Studienrichtung: Technische Informatik (II)
Matrikel-Nr.: ii5092
Semester: 7.

Anmerkung:

Die Ausarbeitung korrespondiert mit dem gleichnamigen Vortrag des Autors

Inhaltsverzeichnis

1 Einführung

Im Rahmen dieser Arbeit geht es um praxisorientierte Tipps zur Integration von XML und Web Services. Dabei handelt es sich nicht ausschließlich um Tipps zur eigentlichen Integration selbst. Vielmehr ist der Titel so zu verstehen, dass sich die Tipps auf unterschiedliche Phasen und Bereiche des gesamten Prozesses einer Integration beziehen und verschiedene Themen abdeckt, die z.T. erst im erweiterten Sinn einer Integration zugeordnet werden können.

Ziel dieser Arbeit ist nicht primär das Vorstellen von Tipps, die z.T. direkt für umzusetzende Integrations-Prozesse verwendet werden können, auch ist der Inhalt der Tipps ist nicht dogmatisch zu verstehen. Vielmehr soll hier die Integration von XML und Web Services aus vielen Blickwinkeln beleuchtet werden, um ein Bewusstsein für die Vielfältigkeit der Problemquellen aufzubauen und für dieses Thema zu sensibilisieren. Aufgrund des Umfangs und der Komplexität vermag diese Arbeit nur ein grobes Bild der Realität wiederzugeben, das an einigen Stellen jedoch exemplarisch vertieft wird.

Als inhaltiche Grundlage dieser Arbeit dienen die Kapitel 12 und 13 des Buches "***Service-Oriented Architecture*** - *A Field Guide to Integrating XML and Web Services*" von Thomas Erl.

2 Tipps zur Integration von XML

2.1 Verstehen, worauf man sich einlässt

Tipp:

> ***Bestimme Deinen Kenntnisstand relevanter XML-Technologien vor der Planung eines Projekts.***

Im Rahmen eines Projektes zur Integration von XML hat man es gewöhnlicherweise mit einer Vielzahl unterschiedlicher XML-Technologien zu tun. Sowohl das grundlegende Konzept, das hinter jeder dieser Technologien steht, als auch die Details, müssen dem zuständigen Projekt-Team bekannt sein, um einen erfolgreichen und damit reibungsarmen Projektverlauf gewährleisten zu können.

Der Grund dafür liegt auf der Hand:

Werden Technologien implementiert, die durch die Fähigkeiten des Teams nicht oder nur unzureichend abgedeckt werden, führt dies fast immer zu Problemen, da die entsprechenden Technologien - aufgrund der mangelnden Kenntnis - nicht so eingesetzt werden, wie dies zur vollen Entfaltung der Möglichkeiten notwendig wäre. So werden sie z.B. nicht an allen Stellen verwendet, wo sie nützlich sind, jedoch an einigen Stellen, wo eine andere Technologie oder eine andere als die gewählte Strategie deutlich mehr Vorteile gebracht hätten. Die mangelnde Sachkenntnis und fehlende Erfahrung des Teams im Umgang mit diesen Technologien führt so zu einer ebenso mangelhaften Integration.

Dem Team müssen zwei Punkte vor der Planung des Integrations-Projektes bewusst sein:

- wie und wo XML in die Applikation passt
  und
- welche Probleme gelöst werden, wenn XML integriert ist.

Der erste Punkt bezieht sich auf die grundsätzliche Analyse der möglichen Einsatzfelder relevanter XML-Technologien. Erst wenn ein grundlegendes Verständnis (v.a. der Konzepte) vorhanden ist, kann bestimmt werden, an welchen Stellen in der Applikation überhaupt XML eingesetzt werden kann. Nicht minder wichtig ist hierbei auch, dass festgestellt wird, wo XML nicht eingesetzt werden kann. Mögliche Einsatzgebiet in Frage kommender XML-Technologien müssen also in beide Richtungen begrenzt werden.

Der zweite Punkt erfordert nun eine weitergehende Analyse. Sind mögliche Einsatzgebiete bestimmt, muss geprüft werden, welche Probleme durch die für einsetzbar befundenen XML-Technologien adressiert und u.U. sogar gelöst werden können. Man muss sich auch hier in zwei Richtungen Gedanken machen, um zu einem vernünftigen Ergebnis zu kommen. Es ist also nicht nur notwendig sich darüber klar zu werden, welche Probleme durch den Einsatz der entsprechenden Technologien gelöst werden können, sondern auch, welche Probleme dadurch *nicht* gelöst werden können. Nur so erhält man ein realistisches Bild des zu erwartenden Ergebnisses.
Dabei ist anzumerken, dass XML selbst keine Probleme löst. Es ist vielmehr eine geschickt gewählte Strategie zur Integration von XML innerhalb der Applikation, die hilft, dass bestehende Probleme nach Abschluss der Integration abgeschwächt oder gelöst sind. XML selbst ist also lediglich als ein Werkzeug zu verstehen, das man sich zu Nutze machen kann, nicht jedoch als aktiver Helfer.

2.1.1 Übung zur eingehenden Analyse

Als Vorbereitung auf eine detaillierte Planung der Integration, ist eine Übung hilfreich, die aus den folgenden vier Schritten besteht:

1. Beschreiben, wie XML dabei helfen kann, Projekt-Aufgaben zu erfüllen.

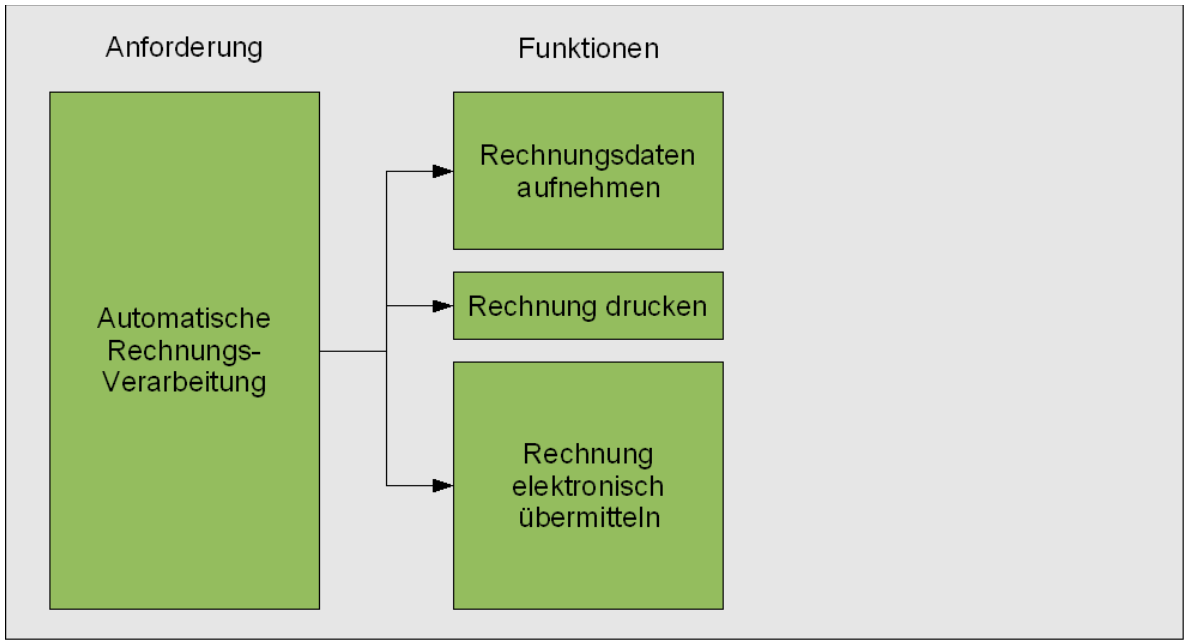
Zu Beginn dieser Übung stehen einem lediglich relativ allgemeine Anforderungen zur Verfügung, die im Rahmen der Integration erreicht werden sollen, wie z.B. eine "automatische Rechnungs-Verarbeitung".

Beispiel:

In diesem ersten Schritt sollen die Anforderungen konkretisiert werden. Dabei erfolgt die Durchführung in zwei Unterschritten:

a) Abbilden der Anforderungen auf Funktionen

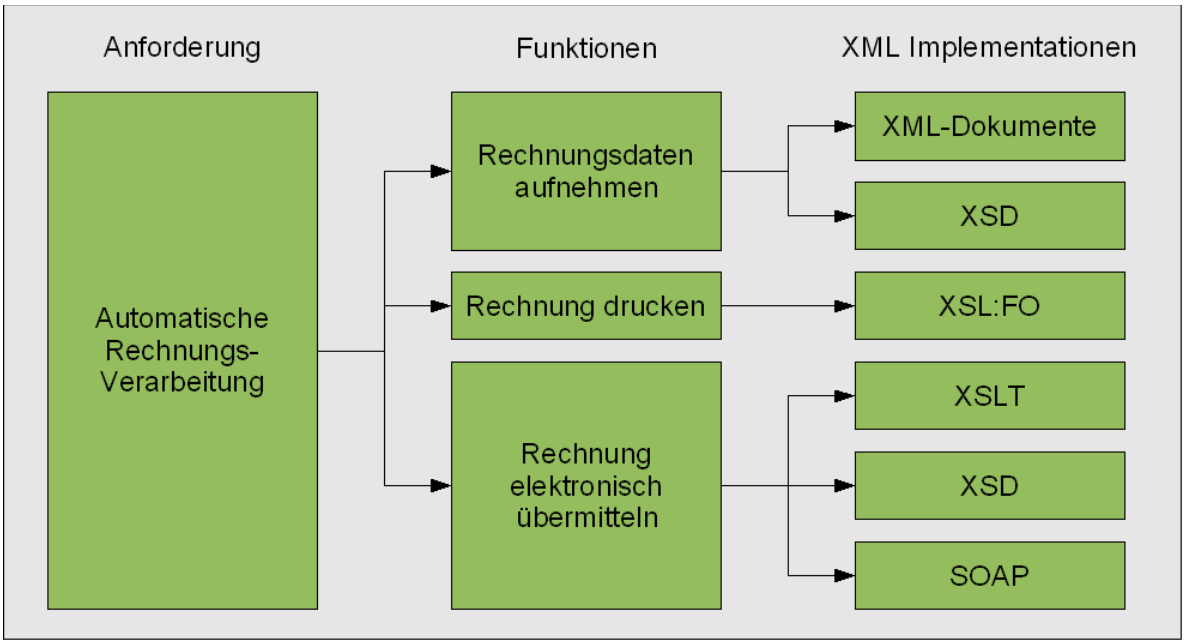
Zunächst werden alle allgemeinen Anforderungen auf konkrekte Funktionen abgebildet, die später in der Applikation zur Verfügung stehen sollen. Im Falle der automatischen Rechnungsverarbeitung erscheinen u.a. Funktionen wie "Rechnungsdaten aufnehmen", "Rechnung drucken" und "Rechnung elektronisch übermitteln" sinnvoll:



Diese drei Funktionen bilden selbstverständlich nur einen kleinen Ausschnitt der Menge aller sinnvollen bzw. sogar aller möglichen Funktionen.

b) Abbilden der Funktionen auf XML-Implementationen

In diesem Schritt wird nun jede Funktion auf entsprechende XML-Implementationen abgebildet. Hier erfolgt also eine konkrete Überlegung darüber, welche XML-Implementationen innerhalb dieser Funktion sinnvollerweise Verwendung finden sollten. In unserem Beispiele wäre etwa folgendes Ergebnis denkbar:



Welchen Zweck die jeweiligen Implementationen innerhalb der entsprechenden Funktionen erfüllen, sollte ebenfalls kurz notiert werden. Im Beispiel sieht das folgendermaßen aus:

XML-Dokumente werden zur Speicherung der Rechnungsdaten bei der Aufnahme verwendet, ein oder mehrere *XSD*-Schema-Dateien ermöglichen die Validierung dieser Dokumente. Beim Drucken einer Rechnung wird *XSL:FO* zur optischen Aufbereitung der Daten verwendet. Um eine Rechnung elektronisch zu übermitteln, lassen sich *XSLT* zur Transformation der Daten in ein für den Empfänger nutzbares Format, *XSD*-Schemas zur Validierung der transformierten Daten und *SOAP* zum eigentlichen Datentransport verwenden.

Am Ende des ersten Schrittes steht einem somit eine Übersicht darüber zur Verfügung, *wo welche* XML-Technologie *zu welchem Zweck* verwendet werden kann.

2. Auflisten der vorgeschlagenen XML-Technologien und beschreiben ihres allgemeinen Zwecks.

Zunächst werden alle Implementationen, die im ersten Schritt dieser Übungen ermittelt wurden, aufgelistet. Anschließend sollte zu jeder dieser Implementationen kurz ihr allgemeiner Zweck beschrieben werden. Mit Hilfe dieses Schrittes lässt sich überprüfen, wie gut das eigene grundsätzliche Verständnis der im ersten Schritt ermittelten XML-Technologien ist. Sinnvoll ist dieser Schritt aus zwei Gründen: Zunächst einmal ist ein Umgang mit diesen Technologien nicht sinnvoll, wenn jegliches Verständnis für sie fehlt. Andererseits ist es auch wichtig zu erkennen, dass selbst ein ausgeprägtes Detail-Wissen kein Ersatz für die Kenntnis des allgemeinen Zwecks einer Technologie ist, denn dieser ermöglicht einen guten Überblick und befähigt dazu eher zu einer sinnvollen allgemeinen Einsatz-Analyse als beispielsweise die genaue Kenntnis der Syntax.

3. Begründen, wie die Fähigkeiten das aktuelle Team für die Arbeit mit diesen Technologien qualifiziert.

Durch den Versuch eine entsprechende Begründung zu formulieren, ergibt sich die Antwort auf die Frage, ob das aktuelle Team überhaupt über ausreichende Qualifikationen verfügt. Nur wenn jede zu verwendende XML-Technologie durch die Kenntnisse des Teams ausreichend abgedeckt ist, kann auf weitere Maßnahmen verzichtet werden. Andererseits sind ggfs. Fortbildungsmaßnahmen, zusätzliche Team-Mitglieder, das Austauschen von Team-Mitgliedern oder das Heranziehen eines internen oder externen Experten sinnvoll. Die Auswahl geeigneter Maßnahmen erfolgt jedoch außerhalb dieser Übung.

4. Einholen einer zweiten Meinung von einem qualifizierten Fachmann

Als abschließenden Schritt sollte man die Ergebnisse jedes vorangegangenen Schritts von einem (unabhängigen) qualifizierten Fachmann prüfen lassen. Nur so lassen sich mögliche Fehler oder Unzulänglichkeiten erkennen und beheben, die die Kenntnisse und Erfahrungen eines Fachmannes erfordern. Selbst wenn diese Übung von einem Fachmann durchgeführt wurde, so ist eine zusätzliche Instanz zur Verifikation der Ergebnisse anzuraten.

2.1.2 Richtlinien zur Durchführung der Übung

Für die Durchführung dieser Übung gibt es einige Richtlinien, die diese vereinfachen und deren Effizienz steigern sollen:

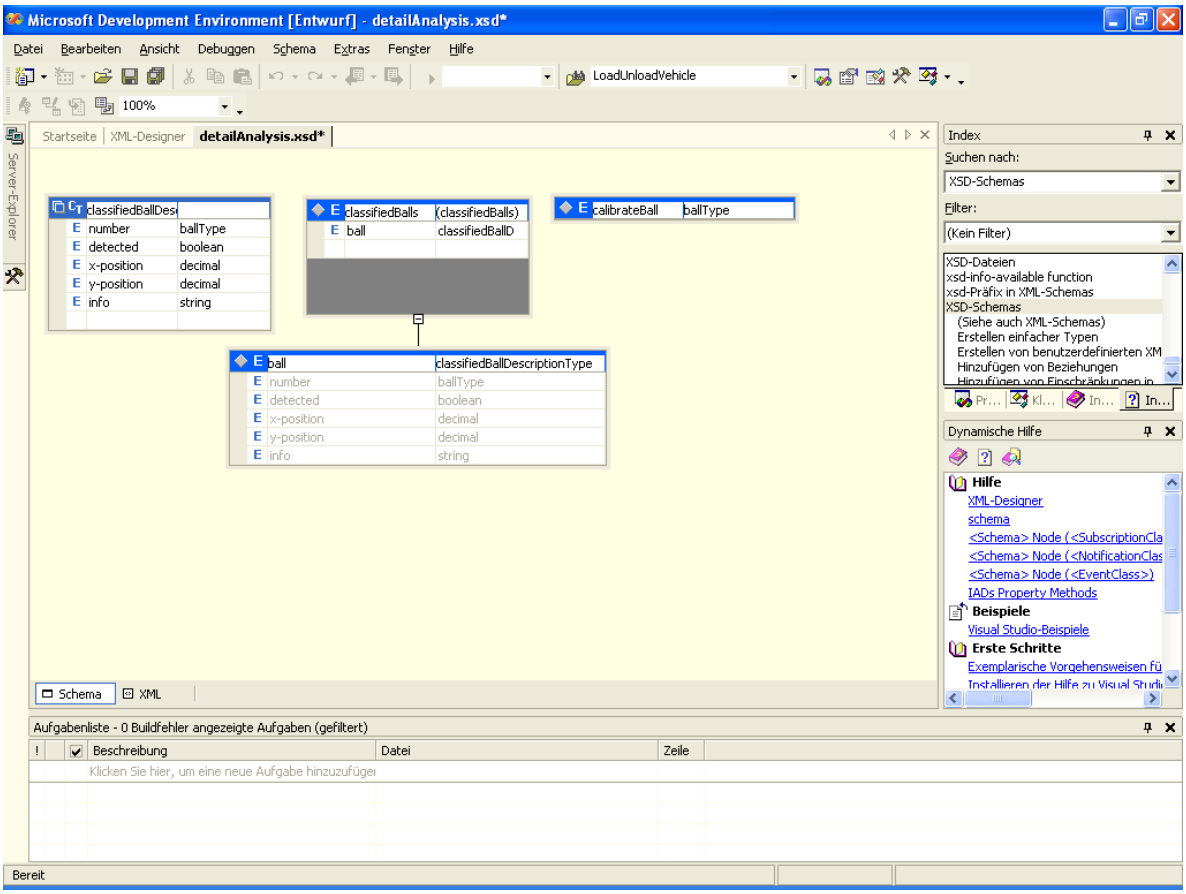
- **Funktionen teilen, wenn sie zu komplex sind.**
  Funktionen, deren Umfang zu groß ist, als dass erkennbar ist, welche XML-Technologien möglicherweise eingesetzt werden können, sollten in kleinere Funktionen zerlegt werden (dies entspricht dem "Prinzip der schrittweisen Verfeinerung"). Dieser Zerlegungsprozess ist rekursiv durchzuführen, bis eine handhabbare Komplexität der jeweiligen Funktionen vorliegt. Es ist darauf zu achten, dass die Rekursion rechtzeitig terminiert wird, da das Abbilden auf die XML-Implementationen zwar sonst sehr einfach wird, die Anzahl der Funktionen aber andererseits stark ansteigt. Die Komplexität aller Funktionen sollte am Ende einigermaßen gleichmäßig sein.

- **Nicht auf "Front-End"-Funktionalität konzentrieren.**
  Aufgaben des Front-Ends, also jegliche die Benutzer-Interaktion betreffenden Teile, sind verhältnismäßig unwichtig, da es bei dieser Übung primär um die eigentliche Geschäfts-Logik geht. Bei der Übung sollten diese daher eine untergeordnete Rolle spielen, wenn auch nicht ganz herausfallen. Anstatt jedoch z.B. den Ablauf zum Ausfüllen und Absenden eines Formulars in alle Einzelheiten zu zerlegen, lässt sich dieser lieber mit "Benutzer übermittelt Daten" zusammenfassen.

- **Beim ersten Durchführen der Übung nur Kern-Funktionen beachten.**
  Um den Umfang der Übung bei der ersten Durchführung zu begrenzen, ist es ratsam, sich zunächst auf die wichtigsten Funktionen zu beschränken. Diese zunächst transparenten Funktionen können dann in einer zweiten Durchführung mit eingezogen werden.

- **Bei der Team-Bewertung auch verfügbare Tools beachten, die u.U. verwendet werden.**
  Tools bieten oft die Möglichkeit Technologien auf vereinfachte Weise zu nutzen, indem sie den Entwickler von der Syntax der verwendeten Technologie distanzieren und die Technologie über ein vereinfachtes Interface zur Verfügung stellen. Ein Beispiel hierfür sind z.B. WYSIWYG-Editoren, die das Erstellen von WebSeiten ermöglichen, ohne dass der Benutzer die Syntax von HTML, CSS und JavaScript kennen muss. So lassen sich beispielsweise XSD-Schemas im Visual Studio von Microsoft grafisch entwickeln (siehe Screenshot); andere Hersteller bieten häufig vergleichbare Möglichkeiten.
  Dadurch werden natürlich die Anforderungen an das Team u.U. erheblich eingeschränkt, was im 3. Schritt der Übung zu berücksichtigen ist.

Screenshot:


Microsoft Development Environment [Entwurf] - detailAnalysis.xsd*
Datei Bearbeiten Ansicht Debuggen Schema Extras Fenster Hilfe
LoadUnloadVehicle
100%
Startseite XML-Designer detailAnalysis.xsd*
classifiedBallDes
number ballType
detected boolean
x-position decimal
y-position decimal
info string
classifiedBalls (classifiedBalls)
ball classifiedBallID
calibrateBall ballType
ball classifiedBallDescriptionType
number ballType
detected boolean
x-position decimal
y-position decimal
info string
Schema XML
Index
Suchen nach:
XSD-Schemas
Filter:
(Kein Filter)
XSD-Dateien
xsd-info-available function
xsd-Präfix in XML-Schemas
XSD-Schemas
(Siehe auch XML-Schemas)
Erstellen einfacher Typen
Erstellen von benutzerdefinierten XM
Hinzufügen von Beziehungen
Dynamische Hilfe
Hilfe
XML-Designer
schema
<Schema> Node (<SubscriptionCla
<Schema> Node (<NotificationClas
<Schema> Node (<EventClass>)
IADs Property Methods
Beispiele
Visual Studio-Beispiele
Erste Schritte
Exemplarische Vorgehensweisen fü
Aufgabenliste - 0 Buildfehler angezeigte Aufgaben (gefiltert)
Beschreibung Datei Zeile
Klicken Sie hier, um eine neue Aufgabe hinzuzufüge
Bereit

2.2 Standards erstellen und einbeziehen

Standards sind wie Ampeln.

Man stelle sich einmal folgende Situation vor: Man steht vor einer Ampel, es ist rot und man wartet auf grün, um endlich weiterfahren zu können. In einer solchen Situation empfindet man die Ampel als hinderlich, doch natürlich ist einem auch bewusst, dass sie dennoch notwendig ist. Ohne Ampeln mag man u.U. schneller ans Ziel kommen, aber die Wahrscheinlichkeit, dass man überhaupt das Ziel erreicht, ist deutlich geringer, denn die Fahrt zum Ziel wäre wesentlich gefährlicher.

Standards kommen eine ähnliche Bedeutung zu. Sie regeln Abläufe, erzeugen eine geordnete Struktur und verhindern Chaos.

Tipp:

Standardisiere immer die Integration von XML-Technologien.

Um auch bei der Integration von XML dem Chaos vorzubeugen, sollte man die Integration von XML-Technologien immer standardisieren. Dadurch erreicht man innerhalb des gesamten Projektes ein Maximum an Konsistenz und Ordnung. Verzichtet man hingegen auf Standards, so riskiert man, dass ein Re-Design früher erforderlich wird als notwendig und wesentlich früher als erhofft.

Häufig verwendete Standards, die im folgenden noch näher beschrieben werden sollen, im Rahmen von Projekten sind:

- Entwicklungs-Standards
- Anwendungs-Design-Standards
- Architektur-Standards
- Infrastruktur-Standards
- Wartungs-Standards
- Projekt-Standards
- Organisations-Standards
- Unternehmens-Standards

Entwicklungs-Standards umfassen

- *Namenskonventionen* (Benennung von Klassen, Eigenschaften, Methoden etc.),
- Regeln über das *Exception-Handling* innerhalb der Applikation,
- *Grammatik-Standards* (Richtlinien über die Verwendung bestimmter Sprachkonstrukte) und
- *Design-Standards* (allerdings ausschließlich bezogen auf Dokumentenstrukturen und die Aufteilung der Funktionen, wie beispielsweise bei der Modularisierung)

In **Anwendungs-Design-Standards** werden

- *Standards*,
- *Richtlinien* und
- *Muster*

definiert, die Informationen darüber enthalten, wie und wann jede der XML-Technologien benutzt werden soll. Hier sind ebenfalls Alternativen zu nennen (beispielsweise XSLT+XHTML+CSS statt XML:FO zur optischen Aufbereitung von XML-Daten)

Architektur-Standards enthalten konkretere Informationen als Anwendungs-Design-Standards, sind mit denen aber inhaltlich verwandt. Hier kann z.B. eine *Standard-Architektur* für zukünftige Entwicklungs-Projekte definiert werden, in der die verschiedenen *XML-Technologien konzeptionell und physikalisch positioniert* werden (korrespondiert mit den Standards, Richtlinien und Mustern über die Verwendung der XML-Technologien in den Anwendungs-Design-Standards). Weiterhin lassen sich innerhalb der Beispiel-Architektur *Beziehungen zwischen den XML-Technologien und Problemen* aus den Bereichen der Integration und der Interoperabilität definieren, so dass hier z.B. angegeben werden kann, welche Probleme durch den Einsatz welcher XML-Technologie adressiert werden sollen.

Die **Infrastruktur-Standards** befassen sich hauptsächlich mit der *Hosting-Umgebung* für Anwendungen. Hier geht es um

- *Server-Konfigurationen*,
- *Richtlinien für den Internet-Zugriff*,
- *Sicherheit*,
- *Konfigurationen* (neben der Server-Konfiguration) und
- *Skalierbarkeit*

Ggfs. geht es hier auch um die *Entwicklungs-Umgebung und entsprechende Tools*, wobei in den Infrastruktur-Standards keine Auswahl bestimmter Produkte, sondern vielmehr die Auswahl bestimmter Einstellungen der Produkte getroffen werden.

Wartungs-Standards beziehen sich auf

- *Administrations-Abläuf*e,
- die *Verwaltung von Benutzer-Konten*,
- *Skalierbarkeit*,
- die Sicherstellung der *Verfügbarkeit* und
- *Routine-Aufgaben der Wartung*.

In den **Projekt-Standards** werden

- *Qualifikations-Anforderungen*,
- *Prozesse* (z.B. innerhalb des Projekt-Teams) und
- *alle relevanten Standards* festgehalten.

Der letzte Punkte bezieht sich dabei auf die in diesem Abschnitt beschriebenen Standards. Im Rahmen der Projekt-Standards werden alle Standards ermittelt, die für das jeweilige Projekt von Bedeutung sind. Hier können auch Abweichungen zum bisherigen Anwendungs-Design und von bestehenden Entwicklungs-Konventionen notiert werden, die im Rahmen des Projektes notwendig sind.

Organisations-Standards können neue *Stellen oder sogar ganze Abteilungen* vorsehen, die für die Verwendung und Wartung der XML-Technologien verantwortlich sind. So lässt sich beispielsweise ein Mitarbeiter zu einem sog. "XSD Custodian" ernennen, der als einziger die Befugnis hat, XSD-Schemas zu erstellen und zu warten. Auf diese Weise fördert man die Bildung von Experten und erreicht ein hohes Maß an Konsistenz innerhalb des gesamten Unternehmens. Es wird ebenfalls festgesetzt, welche *Entwicklungs-Tools und Editoren* vom Projekt-Team verwendet werden dürfen. Ein weiterer Punkt ist - ähnlich wie bei den XML-Technologien - eine *Zuordnung der in diesem Abschnitt erwähnten Standards* zu bestimmen Personen oder Abteilungen.

Unternehmens-Standards entstehen aus sog. "*High-Level"-Anforderungen*, die sich aus der Unternehmens-Strategie ergeben. Verfolgt ein Unternehmen beispielsweise die recht allgemein gehaltene Idee, dass XML innerhalb der nächsten 9 Monate im Unternehmen zur "Schlüsseltechnologie" werden soll, so kann dies z.B. zu einem Unternehmens-Standard führen, der konkret vorgibt, dass XML innerhalb eines bestimmten, festen Bereiches des Systems innerhalb von 3 Monaten verwendet werden muss.

2.3 XML zur Standardisierung des Datenzugriffs

Tipp:

> ***Das Maximieren der Verwendung von XML im "Data Access Layer" erhöht die Kontrolle des Daten-Managements und standardisiert die Applikations-Architektur.***

Der eigentliche Zweck von XML ist die Repräsentation von Daten. Beim Design einer XML-Architektur wird dabei viel Aufwand auf die Vereinheitlichung heterogener Daten gelegt, was dazu führt, dass die Daten-Repräsentation zum zentralen Aspekt wird.

Das ist grundsätzlich sinnvoll, jedoch gibt es Gründe, sogar noch weiter zu gehen. So lässt sich XML auch zur Vereinheitlichung der Programm-Logik verwenden. Dies lässt sich z.B. dadurch realisieren, dass XQuery zum einzigen Einstiegspunkt für den Zugriff auf die Daten wird, was dadurch erreicht werden kann, dass XQuery z.B. SQL als Zugriffs-Sprache ablöst und nicht nur ergänzend eingesetzt wird.

Ein derartiger Ansatz bietet dabei folgende Vorteile:

- Anwendungen müssen sich nicht mehr um Datenbank-Anfragen *und* XML-Parsing kümmern, sondern nur noch um letzteres.

- Die Anforderungen für bestimmte Teile der Applikation, in denen dieser Ansatz verfolgt wird, werden verringert. Dies hat einen positiven Effekt auf die Anforderungen, die an die entsprechenden Projekt-Teams gestellt werden müssen, denn es sind z.B. keine SQL-Kenntnisse mehr notwendig.

2.4 Bewertung von Tools vor der Integration

Tipp:

> ***Bewerte die den Tools zu Grunde liegende Arbeitsweise vorsichtig. Vermeide jene mit proprietären Haken.***

Tools erlauben es Entwicklern XML zu editieren, zu verwalten und sogar zu generieren. Dabei ist jedoch - trotz allen Vorzügen - Vorsicht geboten, denn viele Tools verfügen über proprietäre Erweiterungen und Dateiformate, die dazu führen können, dass man an einen bestimmen Hersteller gebunden wird.

Bevor man daher ein Tool in einen Standard aufnimmt, sollte man die interne Arbeitsweise des entsprechenden Tools gründlich analysieren und bewerten und selbstverständlich sollte auch die Kompatibilität zur bisherigen Entwicklungsplattform getestet werden, um mögliche Probleme zu vermeiden.

2.5 Entwickeln eines Systems zur Wissens-Verbreitung

Tipp:

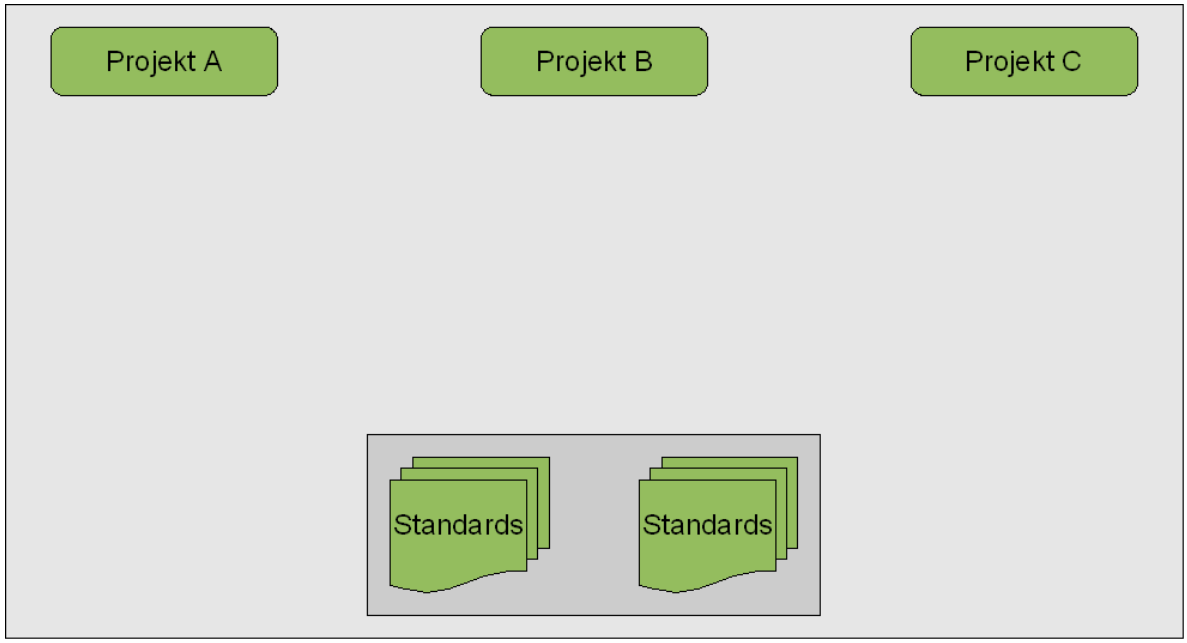
> ***Implementiere ein System zur Wissens-Verbreitung, um die Kommunikation zu verbessern und die Standards aktuell zu halten.***

Normalerweise geht jedes Projekt-Team innerhalb eines Unternehmens individuell mit auftretenden Problemen um. Wird ein Problem erkannt, erfolgt die Entwicklung einer Lösung und diese wird dann implementiert. Die Unabhängigkeit der Projekt-Teams ist hierbei ein großes Problem, denn da jedes Team eigenständig Lösungen entwickelt, sind Lösungen zu ähnlichen Problemen unterschiedlicher Teams nicht konsistent. Das Verwenden einer externen Quelle mit Hilfs-Informationen ist dabei nicht möglich, denn die verwendete Umgebung (Entwicklungs-Umgebung, verwendete Tools, etc.) ist einzigartig. Daher sind auch Probleme innerhalb des Unternehmens einzigartig und nicht unbedingt mit Problemen anderer Teams außerhalb des Unternehmens vergleichbar. Möglicherweise verfügbare Lösungsansätze sind so nicht immer zu gebrauchen.

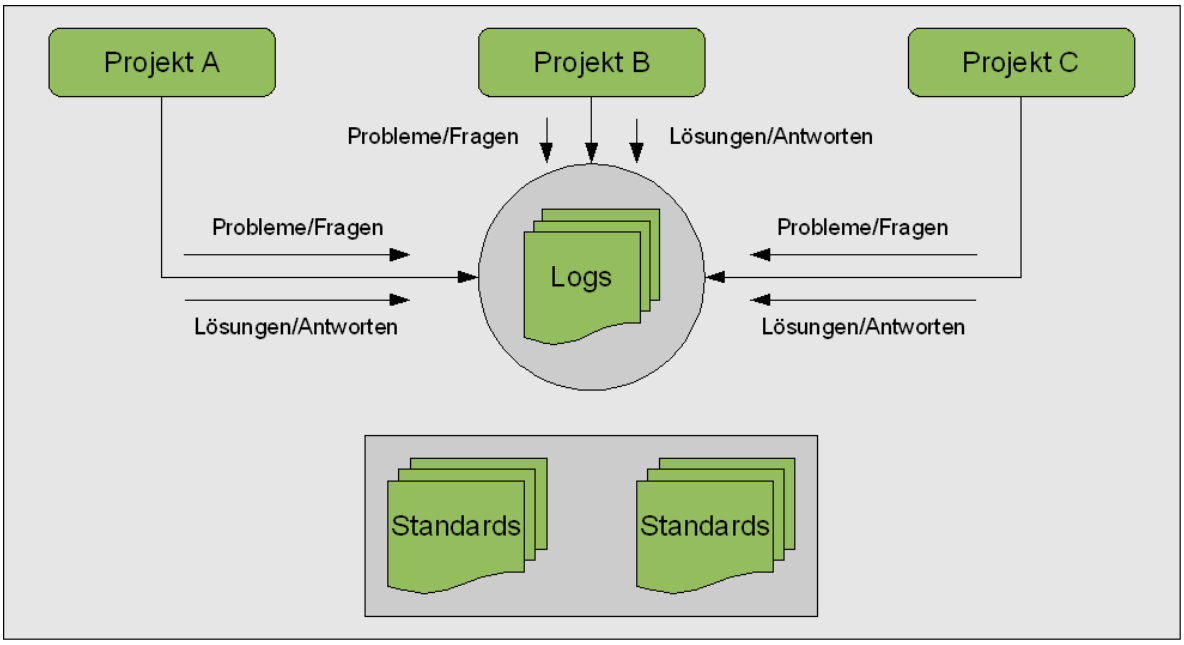
Da die Umgebung in den verschiedenen Projekt-Teams des Unternehmens jedoch u.U. gleich ist (siehe dazu 2.1. Standards erstellen und einbeziehen), kann ein System, dass es ermöglicht Informationen über Problemfälle und Lösungen innerhalb des Unternehmens zu speichern und zu teilen, extrem förderlich sein. Ein solches System kann folgendermaßen aussehen:

2.5.1 Ein Modell-System zur Wissens-Verbreitung

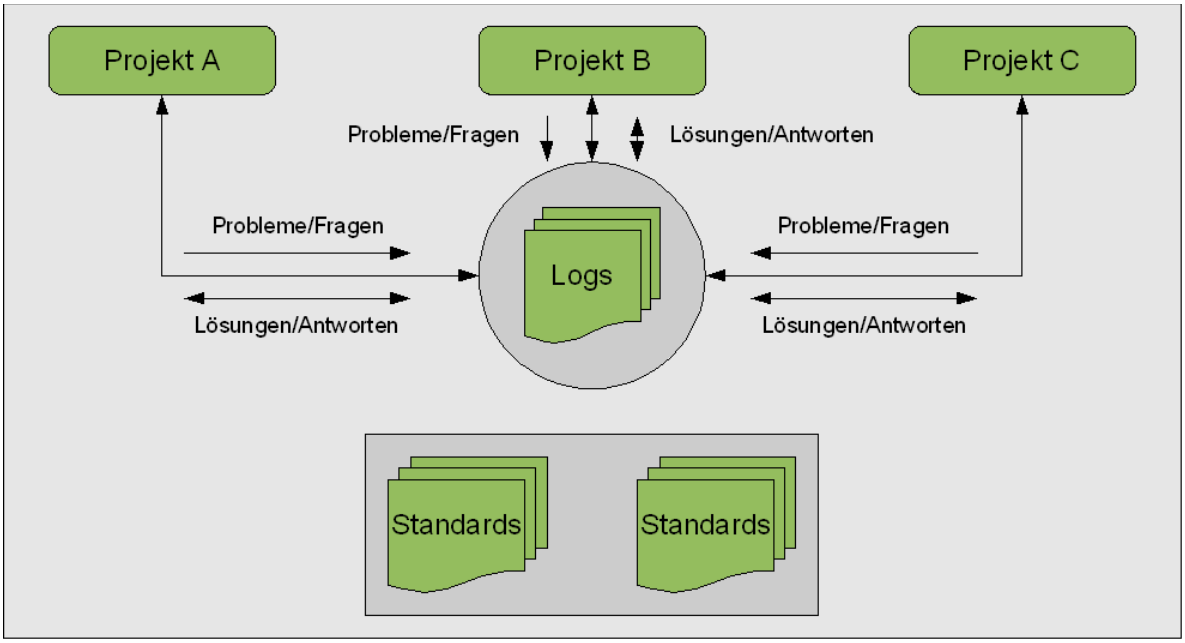
Innerhalb des Unternehmens gibt es eine Reihe von Projekt-Teams (hier exemplarisch "Projekt A", "Projekt B" und "Projekt C" genannt) und eine Reihe von Standards (siehe dazu 2.1. Standards erstellen und einbeziehen). Wir haben also vor der Einführung des Systems folgende Situation:



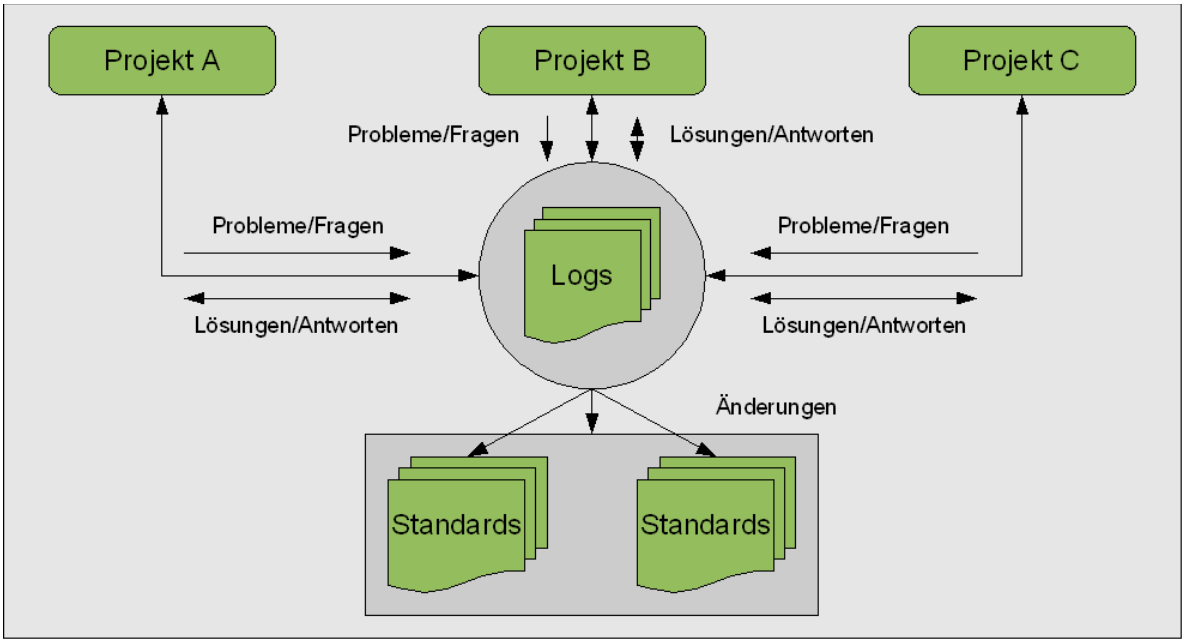
Das System sieht nun zunächst die Einführung eines zentralen Log-Systems vor. Die einzelnen Projekt-Teams erhalten die Möglichkeit auftretende Problemfälle und Fragestellen in dieses Log-System einzutragen. Zusätzlich können die Teams auch passende Lösungen und Antworten in das System eintragen:



Tritt nun innerhalb eines Projekt-Teams ein Problemfall ein oder ergibt sich eine wichtige Frage, so hat das Team nun die Möglichkeit nachzusehen, ob im Log-System bereits ein anderes Team einer ähnlichen Situation ausgesetzt war und ggfs. schon eine Lösung bzw. eine Antwort im Log-System gespeichert hat (siehe nächstes Bild). Ist dies der Fall, so hat das Projekt-Team die Möglichkeit die Erfahrungen des anderen Teams zu nutzen, indem es beispielsweise die Lösung zu dem Problem bzw. die Antwort auf die Frage aus dem Log-System verwendet anstatt sich selbst darum zu kümmern.



Weiterhin ist es auch möglich, die im Log-System gesammelten Informationen dazu zu verwenden, die vorhandenen Standards zu aktualisieren. So ist es beispielsweise möglich, dass bestimmte Problemfälle durch die vorliegenden Standards verursacht, begünstigt oder zumindest nicht gezielt verhindert werden. Durch ein Aktualisieren der Standards auf Basis der im Log-System enthaltenden Informationen ist es daher u.U. möglich, Präventiv-Maßnahmen für die Zukunft einzuleiten, um das Auftreten bestimmter Situationen auszuschließen:



Ein solches System bringt zwei entscheiden Vorteile mit sich. Zum einen kann die Einführung von XML-Technologien deutlich beschleunigt werden, da jedes Projekt-Team Zugriff auf die Erfahrungen der anderen Projekt-Teams hat und daher z.B. für Probleme u.U. keine eigene Lösung entwickelt werden muss. Zum anderen lassen sich mit Hilfe hoch-relevanter, im Log-System gespeicherter Informationen die den Teams zu Grunde liegenden Projekt-, Design- und Entwicklungs-Standards anpassen, wodurch diese immer aktuell bleiben.
Um zu erreichen, dass dieses System auch tatsächlich von allen Entwicklern verwendet wird und um sicherzustellen, dass dies auch in einem hohen Maße geschieht, ist es möglich, das System nicht nur zur Kommunikation zwischen einzelnen Teams einzusetzen, sondern auch zur Kommunikation innerhalb der Teams. Dadurch wird das System zu einer Notwendigkeit und die Vorteile können erschöpfend genutzt werden.

Ein solches System sollte frühzeitig realisiert werden!

3 Tipps zur Integration von Web Services

3.1 Wissen, WANN man Web Services verwenden sollte

Tipp:

> ***Definiere das Ausmaß, in dem Web Services verwendet werden sollen, bevor diese entwickelt werden.***

Nicht immer lässt sich zweifelsfrei klären, ob der Einsatz von Web Services sinnvoll ist. Es gibt jedoch einige Gründe, die für die Verwendung von Web Services sprechen:

- **früh Verständnis erlagen**
  Je früher Web Services integriert werden, desto früher können die Entwickler im Unternehmen Erfahrungen mit dieser neuen Plattform sammeln. Dies fördert die Bildung von Experten, die es dem Unternehmens u.a. auch ermöglichen, neueste Entwicklungen zu testen und zum eigenen Vorteil zu nutzen

- **keine neue Applikations-Architektur erforderlich**
  Um Web-Services verwenden zu können, ist es nicht notwendig, die bestehende Applikations-Architektur zu ersetzen. Dadurch ist der Aufwand für die Einführung einzelner, kleiner Web Services relativ gering.

- **lose Kopplung**
  Die lose Kopplung der Web Services untereinander bietet eine große Flexibilität und hohe Wartungsfreundlichkeit. Im Gegensatz zu "herkömmlichen" Applikationen wirken sich Änderungen an einem Web Service damit weniger stark auf die anderen Web Services aus, als dies bei einer festen Kopplung der Fall wäre.

- **service-orientiertes Design wird bereits verwendet**
  Wird im Unternehmen bereits ein service-orientiertes Design verwendet, so ist der Aufwand wesentlich geringer, das bestehende System auf Web Services umzustellen, als dies bei einem System ohne Service-Orientierung der Fall wäre. Dieser geringere Aufwand kann dazu führen, dass sich die Integration von Web Services lohnt, selbst wenn dies beim Umstieg von einem nicht service-orientierten System nicht der Fall wäre.

- **unterstützende Entwicklungs-Tools**
  Auch verfügbare Tools können den Aufwand für die Integration von Web Services weiter senken und dazu führen, dass sich der Umstieg für das Unternehmen lohnt.

3.2 Wissen, WIE man Web Services verwenden sollte

Tipp:

> ***Limitiere die Reichweite der Web Services auf die Reichweite Deines Kenntnis-Standes. Wenn Du keine Ahnung hast, sollte nichts service-orientiert sein, was wichtig ist.***

Steht innerhalb der Unternehmens kein ausreichendes Wissen über Web Services zur Verfügung, um das bestehende System komplett auf Web Services umzustellen, so sollte man die Ausmaße der Integration an den verfügbaren Kenntnisstand anpassen.

Falls innerhalb des Unternehmens kein Wissen über diese Plattform verfügbar ist, auf die Integration von Web Services jedoch nicht verzichtet werden soll, so bietet es sich an, zunächst mit risikoarmen Prototypen oder Pilotapplikationen erste Erfahrungen zu sammeln.

3.3 Wissen, WANN man Web Services NICHT verwenden sollte

Tipp:

> ***Auch wenn Web Services zum IT-Mainstream werden: Beziehe Web Services nur dort mit ein, wo sie Vorteile bringen.***

Auch wenn Web Services eine durchaus viel versprechende Technologie darstellen, sollte die Integration von Web Services sinnvoll begründet werden können, denn es gibt einige Gründe, die gegen die Verwendung von Web Services sprechen:

- **Anbieter-Unterstützung**
  Zwar sind die verfügbaren Standards robust und gut getestet, doch kann die Anbieter-Unterstützung der entsprechenden Standards erheblich schwanken. Dies ist insb. bei Standards für Web Services der zweiten Generation der Fall. Sollte die Integration von Web Services in der nahen Zukunft keine Notwendigkeit darstellen, so ist es u.U. durchaus ratsam zu warten, bis die Anbieter-Unterstützung der zu nutzenden Standards weiter gestiegen ist. Es ist in einem solchen Fall sicherlich sinnvoll, die Unterstützung seitens der Anbieter kontinuierlich zu analysieren, um sich regelmäßig ein Bild der aktuellen Situation machen zu können.

- **keine automatische Hilfe für Design-Optimierungen**
  Während Tools durchaus bei der Syntax verfügbarer Standards helfen, sind Optimierungen am Design nicht automatisch möglich. Daher ist es ratsam, auf Web Services zu verzichten, wenn kein Entwickler ausreichende Fähigkeiten im Bezug auf das Design von Web Services vorweisen kann.

- **proprietäre Erweiterungen**
  Die prorietären Erweiterungen der verfügbaren Tools sind nicht grundsätzlich schlecht, auch wenn das Wort proprietär von vielen ständig mit einem negativen Touch belegt wird. Es ist jedoch notwendig, dass man versteht, wie die proprietären Teile der Hersteller funktionieren. Man sollte sich außerdem bewusst sein, dass das Verwenden dieser Teile zu einer festen Bindung an einen bestimmten Hersteller führen kann, was nicht immer wünschenswert ist. Eine eingehende Analyse ist daher von Bedeutung. Fehlt jegliche Kenntnis über dieses Thema, so ist es u.U. ratsamer zunächst auf die Verwendung von Web Services zu verzichten.

- **Web Services u.U. gar nicht notwendig**
  So banal wie dieser Grund auch klingen mag, so wichtig ist er auch. Oft wird bei der Einführung neuer Technologien zu viel Energie darauf verwendet, diese zu nutzen. Insbesondere bei Technologien, deren Siegeszug von zahlreichen Unternehmen in einen regelrechten Hype verwandelt wird - wie dies bei Web Services schon fast der Fall ist - versuchen viele einfach "hinterher zu kommen". Dabei sollte vor der Einführung einer neuen Technolgie in einem Unternehmen zunächst die Frage gestellt werden, ob diese Technolgie für dieses Unternehmen überhaupt notwendig ist. Auch wenn die Vorteile von Web Services unbestritten sind, so bringen sie doch auch einige Nachteile mit sich (wie z.B. Kommunikations- und Verarbeitungs-Overhead), der in vielen Fällen von den Vorteilen nicht aufgewogen werden kann. Sollte eine eingehende Analyse ergeben, dass Web Services in einem besimmten Unternehmen keine Vorteile bringen würden (oder dass die Vorteile die Nachteile und den Integrations-Aufwand nicht aufwiegten), so sollte auf die Verwendung verzichtet werden.

3.4 Fortschreiten mit einer Übergangs-Architektur

Tipp:

> ***Ziehe eine Übergangs-Architektur in Betracht, die service-orientierte Konzepte einführt, nicht jedoch die entsprechende Technologie.***

In einigen Fällen kann es sinnvoll sein, innerhalb eines System zwar service-orientierte Konzepte einzuführen, auf die Verwendung der entsprechenden Technologien jedoch zu verzichten. Ein solcher Ansatz bietet einige Vorteile:

- **Ein Zurückkehren ist relativ problemlos möglich**
  Während die Integration bestimmter Technologien nur mit einem hohen Aufwand wieder rückgängig gemacht werden kann, ist dies nicht der Fall, wenn man lediglich die service-orientierten Konzepte berücksichtigt, denn die Veränderungen am bestehenden System sind minimal und der Einfluss auf bestehende Logik ist meist vernachlässigbar.

- **als Notfallplan einsetzbar**
  Sollte sich im Rahmen einer Risiko-Analyse herausstellen, dass der Aufwand für die Integration von Web-Services dem Nutzen nicht gerecht wird, so lässt sich die Überführung der bestehenden Architektur in eine Übergangs-Architektur, in der zumindest die service-orientierten Konzepte berücksichtigt werden, als Notfallplan einsetzen. Dadurch sinkt zukünftig der Aufwand, der für die Integration von Web-Services notwendig wäre.

- **Vorbereiten auf zukünftige SOA-Migration**
  Steht schon frühzeitig fest, dass mittel- oder langfristig eine Migration zu einer service-orientierten Architektur durchgeführt werden soll, so ermöglicht die kurzfristige Einführung einer Übergangs-Architektur den Aufwand der eigentlichen Migration zu verringern.

3.5 Altlasten zu seinem Vorteil nutzen

In vielen Fällen ist es sinnvoll Alt-Umgebungen zu ersetzen oder zu erneuern. Eine wichtige Alternative dazu ist jedoch eine sog. service-orientierte Integrations-Architektur, die die Verwendung von Alt-Logik in einer neuen, service-orientierten Architektur ermöglicht. Diese Alternative sollte vor dem Ersetzen/Erneuern in Betracht gezogen werden.

Tipp:

Erwäge immer die Wiederverwendung alter Logik, bevor Du sie ersetzt.

Als Basis einer solchen Integrations-Architektur können Adapter und/oder ein sog. "*Service Interface Layer*" verwendet werden, die zur funktionalen Abstraktion dienen und es so ermöglichen Alt-Logik in das neue System zu integrieren.

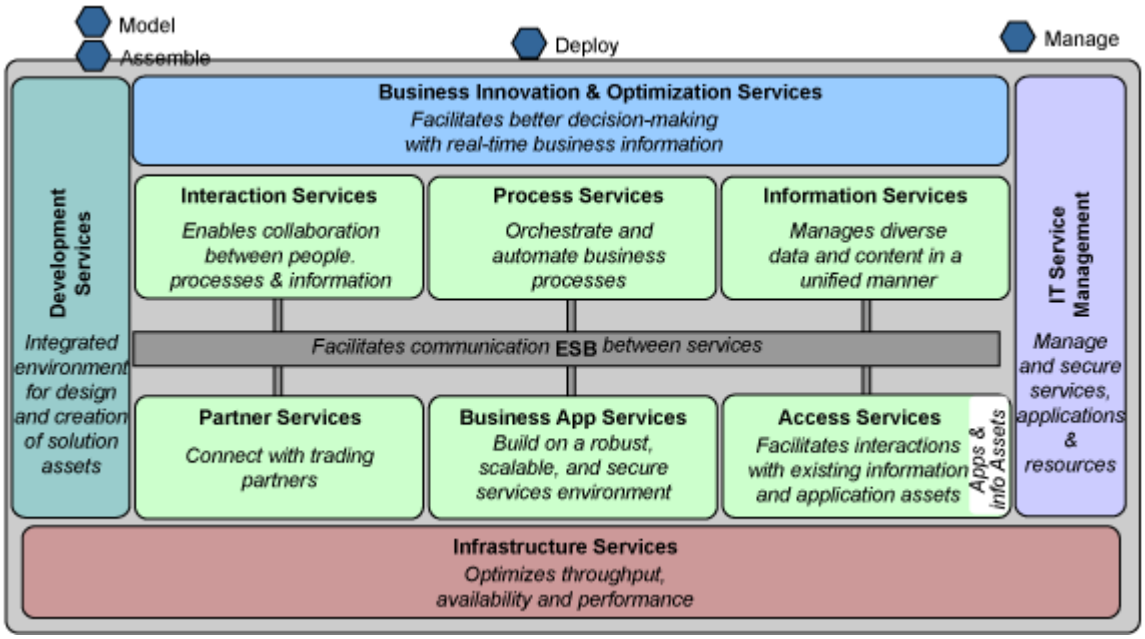
Der Vorteil dieses Ansatzes ist der u.U. deutlich gesunkene Aufwand, denn das Integrieren alter Logik ist in vielen Fällen schneller zu bewerkstelligen als eine komplette Neuentwicklung der betroffenen Teile. Bei der Integration ist jedoch Vorsicht geboten.

Tipp:

Definiere funktionale Grenzen um Alt-Anwendungen und integriere nicht über sie hinaus.

Während die Integration alter Logik nämlich dazu führt, dass das neue System schnell um Funktionalität wächst, so kann diese Strategie durchaus zu Problemen führen. Dies ist z.B. der Fall, wenn Anwendungen integriert werden, bei denen es sich ehemals um isolierte Applikationen gehandelt hat. Entsprechendes gilt für Systeme, die nicht dafür geeignet sind, von außen in ein anderes System integriert zu werden. Neben der Frage, ob eine Integration von Alt-Anwendungen sinnvoll ist, sollte auch immer geklärt werden, wie weit die Alt-Logik in das neue System integriert werden kann, ohne dass der Aufwand zu sehr steigt oder die Auswirkungen auf das neue System zu groß werden. Solange man diese Grenze kennt, handelt es sich jedoch um eine sinnvolle Strategie.

Gängige EAI-Lösungen (egal, ob service-orientiert oder nicht) basieren übrigens auf dem Prinzip der Verwendung von Adaptern zur Integration von Alt-Anwendungen. Ein Beispiel hierfür ist WebSphere von IBM, das alte Logik unter Verwendung von sog. "Business Integration Adapters" in WebSphere integriert. Die Kommunikation läuft dabei über den sog. Enterprise Service Bus (ESB), an den die Adapter angeschlossen werden können.



(siehe http://www-306.ibm.com/software/websphere/)

Bei der Integration von Alt-Logik mit Hilfe von Adaptern ist zu bedenken, dass die Verwendung intelligenter Adapter die Auswirkungen auf die Alt-Logik verringern können, wodurch diese ohne größere Änderungen integriert werden kann.

3.6 Um ein vernünftiges Sicherheits-Modell herumbauen

Tipp:

> ***Das Anwendungs-Design muss auf einem Sicherheits-Modell aufbauen, nicht andersrum.***

Ein "Sicherheits-System" kann vieles bedeuten. So enthält es u.a. Regeln, die die Kommunikation betreffen.

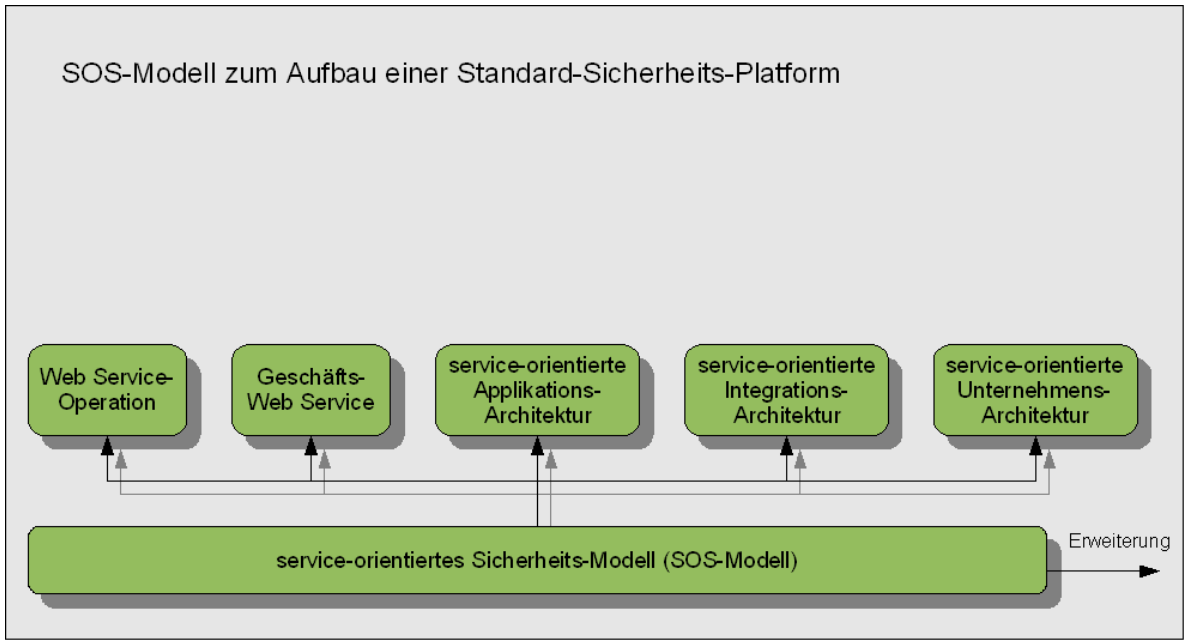
Sicherheits-Systeme für service-orientierte Architekturen sind einzigartig, komplex und multi-dimensional. Es gibt dabei viele Faktoren eines solchen Systems, die andere Teile des Anwendungs-Designs einschränken, was im folgenden noch erläutert wird.

Tipp:

> ***Ein wichtiger Teil eines standardisierten service-orientierten Systems ist ein service-orientiertes Sicherheits-Modell.***

3.6.1 Konzept

Konzeptionell bildet ein service-orientierten Sicherheits-Modells (SOS-Modell) die Basis für das gesamte System, wobei das SOS-Modell selbst erweiterbar sein sollte. Auf diese Basis werden dann alle weiteren Anwendungen aufgesetzt.



Dies ist auch der Grund dafür, dass das SOS-Modell das Anwendungs-Design anderer Teile einschränkt, denn alle kommenden Anwendungen müssen z.B. den Regeln des Sicherheits-Systems unterworfen werden. Die Berücksichtigung des unterliegenden Sicherheits-Modells führt dann beim Anwendungs-Design zu Einschränkungen in allen Punkten, die das Modell umfasst. Regeln bzgl. der Kommunikation zwischen Anwendungen sind ein Beispiel dafür.

3.6.2 Implementation

Die Implementation eines SOS-Modells kann in Form einer eigenen Schicht erfolgen. Diese Sicherheits-Schicht befindet sich dann zwischen je zwei miteinander kommunizierenden Web Services. Hier wird besonders deutlich, dass die Applikationen auf das SOS-Modell abgestimmt werden müssen und dass es sinnvoll ist das SOS-Modell vor allen anderen Teilen zu entwickeln.