Ein Komponenten-Framework für die situationsabhängige Adaption Web-Service-basierter Standardsoftware

Michael Amberg, Shota Okujava, Jens Wehrmann

Lehrstuhl für Wirtschaftsinformatik III, Friedrich-Alexander-Universität Erlangen-Nürnberg, Lange Gasse 20, 90403 Nürnberg, Deutschland, Tel.: +49(911)580796 - 40, Fax: - 42, amberg@wiso.uni-erlangen.de, shota.okujava@wiso.uni-erlangen.de, wehrmann@wiso.uni-erlangen.de, URL: http://www.wi3.uni-erlangen.de

Zusammenfassung: Die Intelligenz hinter einer Anwendung und das Verständnis für die Intensionen der Anwender wird für betrieblicher Standardsoftware immer wichtiger. Für die Neu- und Weiterentwicklung verteilter Standardsoftware werden Ansätze benötigt, die situationsabhängig eine selbstständige Anpassung der Software an die speziellen Bedürfnisse der Anwender und ihrer Tätigkeiten unterstützen.

In dieser Arbeit wird zunächst der Gesamtkontext adaptiver Software für betriebliche Standardsoftware diskutiert, bevor ein Komponenten-Framework für die situationsabhängige Adaption Web-Service-basierter Standardsoftware vorgestellt wird. Die wesentlichen Bestandteile des Komponenten-Frameworks sind die Grundkonzeption der situationsabhängigen Adaption mit Web-Services, die dazu erforderlichen Komponenten zur Protokollierung, die zentralen Komponenten des Adaptionsprozesses, sowie eine Gesamtarchitektur, die das Zusammenspiel der verwendeten Komponenten beschreibt.

Schlüsselworte: Web-Services, Situationsabhängigkeit, Situationsabhängige Protokollierung und Adaption, Adaptive Software, Komponenten-Framework, Standardsoftware, Pull- und Push-Services

1 Motivation

Die Entwicklung der Informationstechnologien ist rasant. Die technischen Möglichkeiten sind kaum noch ein limitierender Faktor in der Softwareentwicklung. Für zukünftige Softwareentwicklungen, insbesondere im Bereich betrieblicher Standardsoftware, wird die Intelligenz hinter einer Anwendung und das Verständnis für die Intensionen der Anwender ein immer bedeutenderer Wettbewerbsfaktor.

Eine Möglichkeit, adaptive bzw. intelligente Software zu realisieren, besteht in der Analyse des Nutzungsverhaltens und in der Speicherung bzw. selbständigen Verwendung von Nutzungsinformationen [OrGD1999]. Dies wird im folgenden als situationsabhängige Adaption bezeichnet. Nutzungsinformationen werden beispielsweise als Nutzungsprofile und Protokolldaten gespeichert. Diese können teilweise sehr detaillierte Informationen über das Verhalten der Anwender bei der Nutzung einer Software beinhalten. Mit Hilfe derartiger Informationen ist es möglich, Situationen zu erkennen und aus den erkannten Situationen

Informationen abzuleiten, die einen Mehrwert sowohl für den Anwender als auch für den Anbieter einer Standardsoftware generieren.

Die Verwendung von Nutzungsinformationen bei Internetanwendungen und mobilen Anwendungen kann bereits als gebräuchlich bezeichnet werden, jedoch fehlen unmittelbar anwendbare Standards sowie standardisierte Softwarekomponenten, die eine effiziente Entwicklung entsprechend adaptiver Software unterstützen. In dieser Arbeit soll am Beispiel Web-Service-basierter Standardsoftware ein Beitrag zur Beantwortung folgender Fragestellungen geleistet werden:

- Welche Arten von situationsabhängiger Adaption lassen sich bei Web-Service-basierter Standardsoftware im Wesentlichen unterscheiden? Welche Vorteile und Auswirkungen sind damit verknüpft?
- Wie sieht ein Komponenten-Framework aus, welches die situationsabhängige Protokollierung und Adaption von Web-Service-basierter Standardsoftware unterstützt? Welche Arten von situationsabhängiger Adaption lassen sich mit dem aufgezeigten Komponenten-Framework wie unterstützen?

Der hier vorgestellte Beitrag wurde im Umfeld eines durch die Bayerische Staatsregierung geförderten Projektes erarbeitet. Im Projektkontext plant ein KMU-Unternehmen als Softwarehersteller die Neuentwicklung einer Controllingsoftware als betriebliche Standardsoftware unter Nutzung von Web-Services. Die Software soll beispielsweise Energieversorgungsunternehmen angeboten werden, wo ca. zehn bis hundert Mitarbeiter des Unternehmens die Software in unterschiedlichen Rollen nutzen werden. Um sich von Wettbewerbern abzuheben, soll eine situationsabhängige Adaption in der Art integriert werden, dass unter Zuhilfenahme von Komponenten-Framework und Standardkomponenten die Adaption möglichst flexibel und in Bezug zur Kernsoftware weitgehend lose gekoppelt ist. An dieser Stelle sei angemerkt, dass die Ausführungen im Grundsatz auch auf die Entwicklung von Individualsoftware mit Multi-Tier-Architektur übertragen werden können.

2 Situationsabhängige Protokollierung und Adaption in Web-Service-basierter Standardsoftware

Im Folgenden wird zunächst der Gesamtkontext aufgezeigt und darauf eingegangen, was unter situationsabhängiger Adaption zu verstehen ist. In Kapitel 2.1 wird skizziert, welche Ziele bei der Einführung situationsabhängiger Adaption in betrieblicher Standardsoftware verfolgt werden. Kapitel 2.2 setzt sich mit der Entwicklung Web-Service-basierter Standardsoftware auseinander.

Der Gesamtanwendungskontext der situationsabhängigen Adaption betrieblicher Standardsoftware ist in Bild 1 vereinfacht dargestellt. Ein Softwarehersteller stellt einem Unternehmen eine vorgefertigte betriebliche Standardsoftware bereit und führt die Weiterentwicklung und Pflege der Software durch. Das Unternehmen (auch als Software-Lizenznehmer bezeichnet) ist u.a. für die unternehmensspezifische Anpassung, die Systemeinführung und den Systembetrieb verantwortlich. Die Software wird im Allgemeinen von mehreren Mitarbeitern und sonstigen Anwendern (z.B. Kunden) verwendet, wobei verschiedene Anwender in der Regel unterschiedliche Softwarefunktionalitäten nutzen. Als Interessensgruppen lassen sich auch unabhängig von der konkreten Software die Softwarehersteller bzw. Entwickler, sowie (End-) Anwender, Administratoren und das (Unternehmens-) Management unterscheiden.

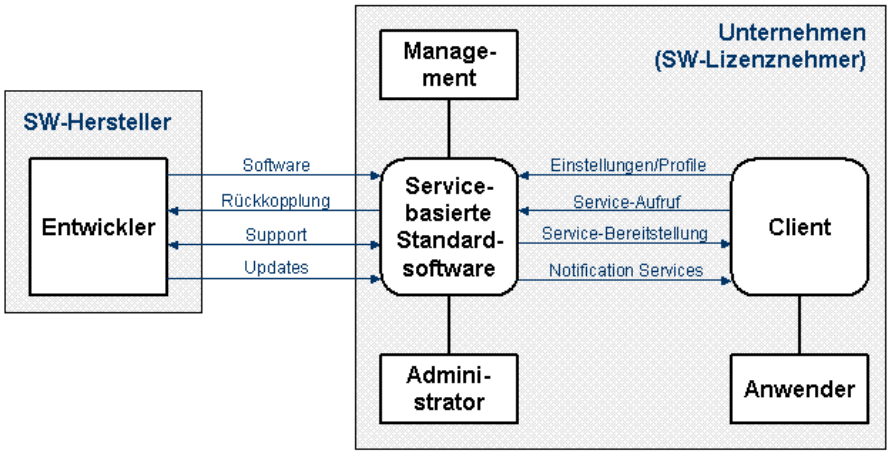

Bild 1 - Gesamtanwendungskontext der situationsabhängigen Adaption betrieblicher Standardsoftware

In diesem Beitrag wird von der Annahme ausgegangen, dass die Anpassung der Funktionalität einer betrieblichen Standardsoftware an die aktuelle Nutzungssituation eines Anwenders einen konkreten Mehrwert darstellt. Dabei wird unterstellt, dass die Problemstellung des Anwenders mit dieser Nutzungssituation korrespondiert. Eine Software, die Informationen über die Situation nutzt, kann sehr viel besser zur Problemlösung beitragen als dies ohne Situationsinformationen möglich wäre. Dabei werden Informationen verarbeitet, die auch für ergänzende Aufgaben und weitere Interessensgruppen hilfreich sind. Art und Umfang einer situationsabhängigen Adaption sind maßgeblich davon abhängig, wie der Situationsbegriff systematisiert und konkretisiert wird.

Ursprünglich entstammt die Diskussion über den Situationsbegriff aus dem Umfeld der mobilen Dienste. In der Literatur finden sich unterschiedliche Ansätze, die den Situationsbegriff bearbeiten und jeweils eigenständig interpretieren (z.B. [Gase2001}, [May2001], [Zobe2001], [AmFW2002] und [AmWe2002]). Hitz et al. [HiKG2002] betrachten in Ihrer Arbeit darüber hinaus explizit die Modellierung von ubiquitären Web-Anwendungen. Sie nehmen die Unterscheidung zwischen *natürlichem* (Ort, Zeit), *technischem* (Endgerät, Browser, Netzwerk, Status) und *sozialem Kontext* (Benutzerprofil und -verhalten) vor. Diese Differenzierung ist auch für betriebliche Standardsoftware geeignet.

Unter dem Begriff der situationsabhängigen Adaption wird die Anpassung einer Softwarefunktionalität an die aktuelle Nutzungssituation verstanden. Dabei sollen die Softwarekomponenten im Sinne einer intelligenten Software in der Lage sein, sich selbstständig an die speziellen Bedürfnisse der Anwender anzupassen und die Anwender in ihren Tätigkeiten aktiv zu unterstützen.

2.1 Ziele der situationsabhängigen Protokollierung und Adaption

Das übergeordnete Ziel für die Einführung der situationsabhängigen Adaption ist es, durch Ausrichtung bzw. Anpassung der Softwarefunktionalität an die spezifischen Bedürfnisse der Anwender einen erkennbaren Mehrwert zu schaffen, die Nutzungshäufigkeit der Software zu steuern sowie die Effektivität der Nutzung zu steigern. Hierzu muss die Software das Nutzungsverhalten protokollieren, wiederholt analysieren und Nutzungsinformationen speichern bzw. Nutzungsprofile anpassen und verwalten.

Über die **inhaltliche Anpassung der Softwarefunktionalität** auf die Bedürfnisse der Anwender hinaus bietet der Umgang mit derartigen Nutzungsinformationen in betrieblicher Standardsoftware die Gelegenheit, eine oder mehrere der folgenden Zielsetzungen zu realisieren:

Gestaltung adaptiver Nutzerschnittstellen
Mit Gestaltung adaptiver Nutzerschnittstellen wird hier einerseits die Bereitstellung der Softwarefunktionalität in einer für den Anwender intuitiven Art und Weise sowie andererseits das situationsabhängige Anbieten von Hilfestellungen und Zusatzinformationen bezeichnet. Ersteres umfasst beispielsweise Anpassungen der Menüs, eine Reduktion auf anwendbare Funktionen oder die Bereitstellung individualisierter Zusatzfunktionen. Letzteres umfasst beispielsweise sogenannte Notification-Services, um Anwender Hinweise auf anstehende Tätigkeiten bzw. geänderte Datenbestände zu geben und Verbesserungsvorschläge für ein effektives Arbeiten zu unterbreiten. Die benötigten Funktionalitäten sollen so effizienter zur Verfügung gestellt werden und ein intuitiveres Benutzen der Software ermöglichen.

Nutzungsabhängige Leistungsverteilung und Performanzmanagement
Die Protokollierung des Nutzungsverhaltens ermöglicht die Messung und Analyse der nutzungsabhängigen Systemlast. Entsprechende Ergebnisse erlauben es der Software, nutzungsspezifisch (Zwischen-) Ergebnisse gegebenenfalls im Vorfeld einer Anfrage zu berechnen und zwischen zu speichern. Dies kann Engpasssituationen vermeiden und helfen, das Systemverhalten zu verbessern.

Nutzungsabhängige interne Leistungsverrechnung
Die Protokollierung des Nutzungsverhaltens kann weiterhin Grundlage einer internen Leistungsverrechnung werden. Es kann festgestellt werden, welcher Mitarbeiter was, wann, wie oft und insbesondere in welcher Art und Weise einsetzt. Dies kann sowohl für die nutzungsabhängige Aufteilung auf innerbetriebliche Kostenstellen als auch für die Auswertung zu Controllingzwecken verwendet werden. Derartige Informationen helfen weiterhin, die Aktualität der Datenbestände zu sichern sowie den Bedarf für Schulungsmaßnahmen und Anreizsysteme zur Steigerung der Softwarenutzung zu identifizieren.

Nutzungsabhängige Softwareanalyse, –wartung und –pflege
Die Nutzungsinformationen einer Software stellen für den Hersteller der Standardsoftware eine wertvolle Informationsquelle dar. So erlauben entsprechende Analysen (Welche Funktionen der Software werden wie und wie häufig genutzt?) eine Bewertung der Nützlichkeit von Softwarefunktionen und liefern Hinweise für zu ergänzende oder zu überarbeitende Funktionen. Eine Schnittstelle für den regelmäßigen Austausch von Nutzungsinformationen zwischen Anbieter und Nutzer schafft ein stetiges Verbesserungspotenzial, damit die Software permanent weiterentwickelt und ohne lange Versionszyklen verbessert werden kann. Dies ist insbesondere bei der Erstentwicklung einer Standardsoftware und bei Pilotanwendungen hilfreich.

Erweitertes Application Service Providing (ASP)
Im Zusammenhang mit der Bereitstellung der Software als Application Service können Nutzungsinformationen hilfreich sein, um leistungsabhängige Lizenzierungsverfahren zu realisieren. Diese werden nicht wie herkömmliche Lizenzen einmalig oder periodisch sondern nach dem Grad der Softwarenutzung abgerechnet. Zudem kann die nutzungsabhängige Leistungsverteilung und Performanzmanagement berücksichtigt und die situationsabhängige Betreuung der Anwender verbessert werden.

Diese Aufzählung verdeutlicht, dass die situationsabhängige Protokollierung und Adaption bei betrieblicher Standardsoftware einen erkennbaren Mehrwert für die beteiligten Interessensgruppen bieten. In **Bild 2** wird der Zusammenhang zwischen Interessensgruppen und Zielsetzungen der situationsabhängigen Protokollierung und Adaption dargestellt.

Kategorien / Inter-essengruppen	Adaptive Nutzerschnitt-stellen	Leistungsvert. & Performance-management	Interne (Leistungs-) Verrechnung	Analyse, Wartung & Pflege	Erweitertes ASP
Software-Anbieter				X	X
Management			X		X
Administrator		X		X	
Endanwender	X	X			

Bild 2 – Übersicht der Interessensgruppen einer situationsabhängigen Protokollierung und Adaption

Kernproblem der situationsabhängigen Adaption ist der Umgang mit den Nutzungs- und Personendaten. Es sind die Aspekte des Datenschutzes zu beachten und eine Überwachung der Mitarbeiter zu verhindern. Gegebenenfalls kann durch Verschlüsselung und Verwendung von Pseudonymen sichergestellt werden, dass derartige Informationen nur in einer Art und Weise an Personen weitergegeben werden, dass diese nicht auf konkrete Anwender rückschließen können. Auf diese Problematik sowie mögliche Lösungsansätze soll in diesem Beitrag nicht weiter eingegangen werden.

Die Datenschutzproblematik wird hier dadurch abgeschwächt, dass im Folgenden die Kategorien von situationsabhängiger Adaption weiter betrachtet werden, die sich auf die Nutzergruppe *Endanwender* fokussieren und insbesondere automatisiert verarbeitet werden. Bei einer automatisierten Verarbeitung können Informationen nicht ohne weiteres von Dritten eingesehen werden. Für die automatisierte Verarbeitung in Web-Service-basierter Standardsoftware sind insbesondere die folgenden Subkategorien von Interesse:

- ***Adaptive Pull-Services:*** Ein Serviceaufruf eines Anwenders wird interpretiert und das Ergebnis entsprechend der Nutzungsinformationen modifiziert. Der Anwender erhält eine auf seinen Nutzungskontext angepasste Softwarefunktionalität.
- ***Adaptive Push-Services:*** Ein Service wird von sich aus aktiv und informiert den Anwender geeignet. Es können Hinweise auf anstehende Tätigkeiten, über geänderte Datenbestände oder Verbesserungsvorschläge für ein effizienteres Arbeiten gegeben werden. Diese Form von Services wird im Folgenden als Notification-Service bezeichnet.
- ***Adaptive Nutzeroberfläche (GUI):*** Die Ergebnisse eines Serviceaufrufes werden hinsichtlich der individuellen Bedürfnisse (individuelle Vorlieben, technische Möglichkeiten des Endgerätes) in der Darstellung angepasst. Daraus können auch Anpassungen der Menüführung resultieren (Reduktion auf anwendbare Funktionen oder die Bereitstellung individualisierter Zusatzfunktionen).

Im Folgenden soll näher darauf eingegangen werden, wie die situationsabhängige Adaption bei der Entwicklung Web-Service-basierter Standardsoftware berücksichtigt werden kann.

2.2 Entwicklungsaspekte Web-Service-basierter Standardsoftware

Insbesondere bei der Neu- und Weiterentwicklung betrieblicher Standardsoftware besitzt die komponentenbasierte Softwareentwicklung gegenüber der konventionellen Entwicklung zahlreiche Vorteile, wie beispielsweise geringere Kosten und die Möglichkeit, die Software adäquat und wirtschaftlich an die individuellen Bedürfnisse der Unternehmen anzupassen [OrHE1996]. Darüber hinaus stellt gerade die Wiederverwendung von Komponenten für die Hersteller von Standardsoftware eine Möglichkeit dar, die zunehmende Komplexität ihrer Anwendungssysteme zu beherrschen [Turo1999].

Web-Service-basierte Software wird als eine besondere Klasse komponentenbasierter Software aufgefasst. Web-Services unterteilen eine Software in grob granulare Softwarekomponenten, die eher lose miteinander gekoppelt sind. Softwarehersteller erwarten bei der Entwicklung verteilter Anwendungen Vorteile durch die anwendungsnahe Spezifikation von Softwarekomponenten, die sich an den Geschäftsprozessen orientieren. Die Funktionen eines Geschäftsprozesses werden eins zu eins auf Web-Services abgebildet und können über mehrere Rechner verteilt werden. Durch die Unabhängigkeit von Plattformen, Betriebssystemen und Programmiersprachen und die Nutzung von etablierten Web-Standards kann eine Software auch über die Unternehmensgrenzen hinaus verwendet werden.

<table>
<tr><th>Kategorien / Granularität</th><th>Adaptive Pull-Services</th><th>Adaptive Push-Services</th><th>Adaptive Nutzer-oberfläche</th><th>Leistungsvert. & Performance-management</th><th>Interne (Leistungs-) Verrechnung</th><th>Analyse, Wartung & Pflege</th><th>Erweitertes ASP</th></tr>
<tr><td>Web-Service Request</td><td>Anpassung bezügl. Person, Zeit und Ort</td><td rowspan="4">Versenden einer Information (Notification Service), falls ein Wert einen definierten Bereich verlässt</td><td>Berücksichtigung von Userprofilen</td><td>Vorberechnung aller möglicher Anfragen</td><td>Separate Abrechnung beim Abruf recheninten-siver Berichte</td><td rowspan="2">Fehlermeldung, Exeption Management, Feedback bei Fehlern, Optimierung des Services im Zeitverlauf</td><td rowspan="3">Nutzungs-abhängige Abrechnung</td></tr>
<tr><td>Web-Service</td><td rowspan="2">Wertehistory, Rezensionen, Empfehlungen, Notizen, Tips, Kommentare</td><td rowspan="2">Anpassung der Menüs, letzte Funktionen, Favoriten (clientseitig)</td><td rowspan="2">Individuelle kontextab-hängige Vor-berechnung (mittels Muster-erkennung) oder Caching von lokalen Informationen</td><td>Interne Vergabe von Rechten für die Benutzung spezifischer Dienste</td></tr>
<tr><td>Teil-Applikation</td><td rowspan="2">Abrechnung nach Benutzer-gruppen (die auf spezifische Komponenten zugreifen) und Verteilung der Kosten auf entsprechende Kostenstellen</td><td rowspan="2">Benutzungs-statistiken, Prioritätenma-nagement für die Weiter-entwicklung</td></tr>
<tr><td>Gesamt-Applikation</td><td>Integration von Inhalten, Daten und Funktionen</td><td>Anpassung der Default-GUI-Werte, Favoriten (serverseitig)</td><td>Lastverteilung, Verdichtungs-operationen</td><td>Wahl des optimalen Abrechnungs-modells</td></tr>
</table>

Bild 3 - Beispiele situationsabhängiger Adaption in Web-Service-basierter Standardsoftware

Aus Anwendungssicht stellt eine Web-Service-basierte (Standard-) Software die aufrufbare Softwarefunktionalität in Form von Web-Services bereit. Inhaltlich zusammenhängende Web-Services werden gegebenenfalls mehrstufig zu größeren Softwarekomponenten zusammengefasst. Die sich ergebende Granularität von Softwarekomponenten kann für die

Identifikation und Systematisierung der unterschiedlichen Arten einer situationsabhängigen Adaption hilfreich sein.

In **Bild 3** werden Beispiele für unterschiedliche Arten einer situationsabhängigen Adaption in Web-Service-basierter Standardsoftware aufgezeigt. Zur Strukturierung werden einerseits die in Kapitel 2.1 aufgezeigten Kategorien und andererseits Granularitätsstufen verwendet. Es werden hier vereinfacht vier Granularitätsstufen unterschieden: Web-Serviceaufruf (Web-Service Request), Web-Service, Softwarekomponente (Teil-Applikation) und Gesamtsoftware (Gesamt-Applikation). Die Abbildung zeigt am Beispiel Controllingsoftware auf, dass für eine situationsabhängige Adaption ein sehr breites Anwendungsspektrum existiert.

3 Ein Komponenten-Framework für die situationsabhängige Adaption

In diesem Kapitel wird ein Komponenten-Framework vorgestellt, welches auf eine umfassende Unterstützung der situationsabhängigen Adaption in Web-Service-basierter Standardsoftware ausgerichtet ist. Das Komponenten-Framework soll folgenden Anforderungen gerecht werden:

- **Lose Kopplung zwischen Adaption und Kernsoftware:** Adaption und Kernsoftware sollen möglichst unabhängig voneinander und nur lose miteinander gekoppelt sein. Inhalt und Umfang der Adaption sollen sich zur Laufzeit flexibel an veränderte Anforderungen anpassen können.
- **Flexibilität hinsichtlich Adaption:** Inhalt und Umfang der situationsabhängigen Adaption soll ausschließlich durch die Anforderungen der Kernsoftware und nicht durch die Implementierung des Komponenten-Frameworks festgelegt werden.
- **Gewährleistung hoher Performanz:** Einbußen in der Performanz durch die Adaption sollen weitestgehend minimiert bzw. vermieden werden. Eine direkte Kommunikation sowie eine gegebenenfalls parallelisierte Bearbeitung werden bevorzugt.
- **Realisierung mit Standardkomponenten:** Der Adaptionsprozess soll weitgehend automatisiert und standardisiert werden. Der Aufwand für die softwareindividuelle Adaption ist weitgehend zu reduzieren.
- **Orientierung an offenen Standards:** Aufbauend auf der Entwicklungsplattform Java 2 Enterprise Edition (J2EE) und Web-Services sollen möglichst offene Standards (z.B. XML, XSLT) und Open-Source-Komponenten (z.B. JAXP) eingesetzt werden.

Im Folgenden werden wesentliche Bestandteile des Komponenten-Framework vorgestellt. Zunächst wird die Grundkonzeption der situationsabhängigen Adaption mit Web-Services aufgezeigt (Kapitel 4). Anschließend werden in Kapitel 5 zentrale Komponenten des Adaptionsprozesses eingeführt. Die Erweiterungen werden in eine Gesamtarchitektur integriert und das Zusammenspiel der verwendeten Komponenten beschrieben (Kapitel 6).

4 Konzeption der situationsabhängigen Protokollierung und Adaption mit Web-Services

Die Grundkonzeption der situationsabhängigen Adaption mit Web-Services zielt auf eine lose Kopplung zwischen Adaptions- und sonstiger Kernsoftwarefunktionalität ab, um so eine nachträgliche Berücksichtigung bzw. Erweiterung der situationsabhängigen Protokollierung und Adaption zu unterstützen.

Die Grundkonzeption ist in **Bild 4** dargestellt. Es wird in diesem Ansatz davon ausgegangen, dass eine Client-Komponente bei einem Serviceaufruf keine Kenntnisse über Art und Umfang der Adaption hat bzw. erhält. Die aufgerufene Server-Komponente benötigt derartige Informationen lediglich, wenn eine inhaltliche Anpassung der Softwarefunktionalität erfolgen soll. Jede andere Form der Adaption erfolgt durch die Analyse, Interpretation und gegebenenfalls Modifikation der Kommunikationsnachrichten durch eine Standardkomponente für die situationsabhängige Adaption (SDA-Komponente). Die SDA-Komponente besteht aus einer XSLT-Engine für die Transformation von XML-Nachrichten mit Stylesheets und einem Stylesheet-Generator, der die Stylesheets abhängig vom Kontext und Situation generiert.

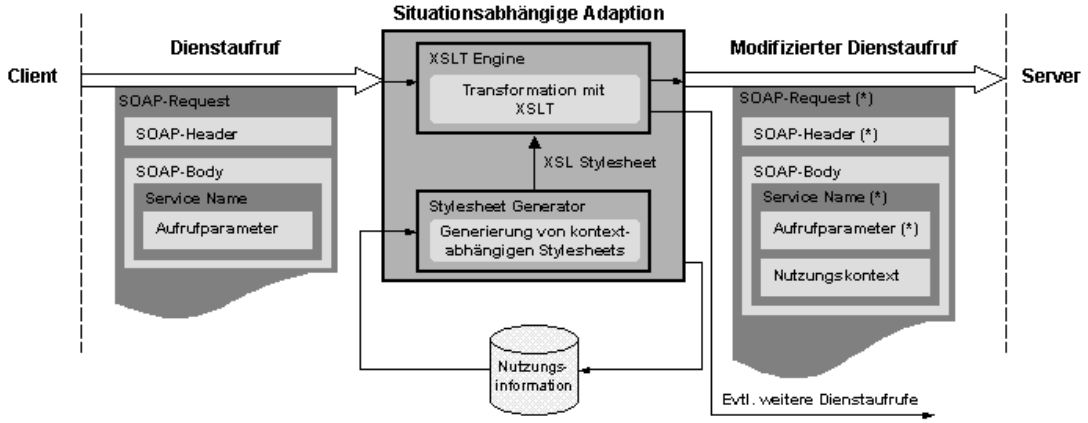

Bild 4 - Grundkonzept der situationsabhängigen Protokollierung und Adaption mit Web-Services

Der grundsätzliche Ablauf lässt sich wie folgt beschreiben: Ein Serviceaufruf (SOAP-Request) wird von der SDA-Komponente entgegengenommen, mit Hilfe einer Standard-XML-Transformation modifiziert und gegebenenfalls um Informationen zum relevanten Nutzungskontext ergänzt. Im modifizierten Serviceaufruf können sowohl der Header (z.B. Ergänzung von Metadaten für den Adaptionsprozess), der Servicename (z.B. Nutzung eines anderen Services) als auch die Aufrufparameter (z.B. Erweiterung des Aufrufs um den Nutzungskontext) verändert worden sein. Das für den Transformationsprozess notwendige Stylesheet wird unter Nutzung eines Stylesheet-Generators gegebenenfalls dynamisch aus gespeicherten Nutzungsinformationen generiert. Der Transformationsprozess wird zudem verwendet, um Informationen über das Nutzungsverhalten abzuleiten bzw. weitere Serviceaufrufe für weiterführende bzw. ergänzende Adaptionen zu generieren.

Die gleichen Adaptionsschritte können auch auf Ebene der Aufrufergebnisse (SOAP-Response) durchgeführt werden. Die Ergebnisse eines Aufrufs werden entsprechend des Kontextes und der Situation modifiziert bzw. erweitert. Der beschriebene Adaptionsprozess kann neben der Transformation genutzt werden, um das Nutzungsverhalten zu protokollieren oder sonstige Aktionen zu initiieren. So ist beispielsweise beim Eintreten eines vorgegebenen Nutzungsereignisses eine Benachrichtigung von Endanwendern über die Nutzeroberfläche, per Email oder SMS möglich.

5 Komponenten des situationsabhängigen Adaptionsprozesses

Für den situationsabhängigen Adaptionsprozess lassen sich folgende Gruppen von Komponenten unterscheiden:

- **Komponenten der Kernsoftware:** Umfasst die Komponenten der Kernsoftware unterteilt in Client-Komponenten und Server-Komponenten inklusive Web-Service-Interfaces.
- **Softwareindividuelle Adaptionskomponenten:** Umfasst die Komponenten, die eine situationsabhängige Adaption softwareindividuell realisieren. Sie können in serverseitige sowie clientseitige Adaptionskomponenten gegliedert werden.
- **Standard-Adaptionskomponenten:** Umfasst die Standardkomponenten zur Adaption, die unmittelbar auf den situationsabhängigen Adaptionsprozess ausgerichtet sind und unabhängig von einer konkreten Software eingesetzt werden können.

Im folgenden wird insbesondere auf die dritte Komponentengruppe fokussiert und das Zusammenspiel der Komponenten aus dem Blickwinkel der Standardkomponenten betrachtet. Die zentralen Designcharakteristika für das Zusammenspiel der benötigten Komponenten sind:

- **Mehrstufiger Adaptionsprozess:** Für die umfassende Unterstützung der situationsabhängigen Adaption eines Serviceaufrufes wird ein mehrstufiger Bearbeitungsprozess zugrundegelegt. Die Bearbeitungsschritte können beispielsweise nach der Zielsetzung der Adaption oder dem Bearbeitungsaufwand differenziert werden. Die Bearbeitung erfolgt zunächst auf dem Client (z.B. für die Ergänzung des Nutzungskontextes und für eine clientseitige Adaption), anschließend durch Standardkomponenten, die allgemeine Adaptionsfunktionalität bereitstellen (z.B. für eine serviceunabhängige Adaption) und abschließend auf dem Server (z.B. für eine serverseitige Adaption).
- **Performanz und Lastverteilung:** Um der Entstehung von Systemengpässen vorzubeugen, sollen die Anzahl der Bearbeitungskomponenten sowie die Ablauffolge variabel gehandhabt werden. Abhängig vom konkreten Serviceaufruf und allgemeinen Adaptionseinstellungen sollen lediglich die benötigten Bearbeitungskomponenten angesprochen werden (Variable Kommunikationsrouten). Wenn möglich, werden voneinander unabhängige Bearbeitungskomponenten parallel aufgerufen (Autonome Bearbeitung). Informationen (z.B. Protokollierungsinformationen), die für den Echtzeitbetrieb nicht zwingend erforderlich sind, werden asynchron mit Messages übertragen. Hierzu werden Komponenten für die Administration benötigt.

Das resultierende Komponentendesign ist in Bild 5 dargestellt. Es basiert auf den Standards Java 2 Enterprise Edition (J2EE) und Web-Services und erweitert die Architektur für Web-Service-basierte Software um Komponenten für die situationsabhängige Adaption.

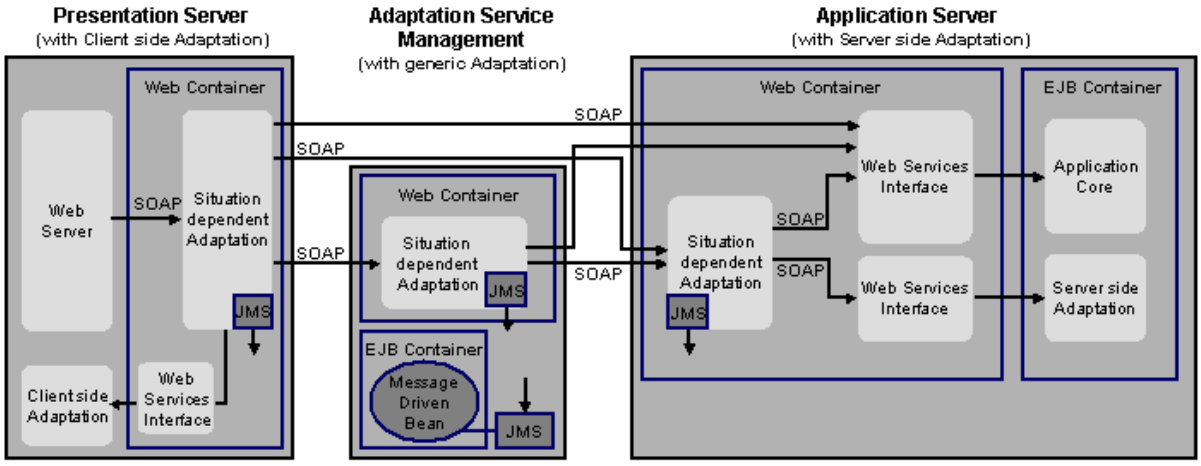

Bild 5 - Um Komponenten für die situationsabhängige Adaption erweiterte Multi-Tier-Architektur

Eine Web-Service-basierte Standardsoftware ist in der Regel als Multi-Tier-Architektur realisiert, die im Allgemeinen vier grundlegende Komponenten unterscheidet: (Thin-) Client, Presentation-Server, Application-Server und Datenbank-Server. Für die situationsabhängige Protokollierung und Adaption sind insbesondere die in der Abbildung dargestellten Komponenten von Interesse, die um einen weiteren Server ergänzt werden.

Der **Präsentations-(Web-)Server** gibt einem oder mehreren (Thin-) Clients die Möglichkeit, Web-Services eines Applikations-Servers zu nutzen. Dieser ist als Webserver konzipiert und um eine Clientseitige Adaptionskomponente (CSA) ergänzt. Der Web-Server nimmt die Anfrage eines Client entgegen, wandelt diese in einen Web-Service-Aufruf (SOAP-Request) um und leitet den Aufruf an eine situationsabhängige Adaptionskomponente weiter. Diese Adaptionskomponente erweitert den Web-Service-Aufruf um spezifische Informationen über den Nutzungskontext und legt die weitere Bearbeitungsfolge fest. Ein Web-Service eines Applikations-Servers kann sowohl direkt, als auch indirekt über eine speziellen Adaptions-Server (Adaptation Service Management) aufgerufen werden. Ein Anwendungsbeispiel für eine clientseitige Adaption ist die Hervorhebung der Werte eines Berichtes, die sich seit der letzten Anzeige des Berichtes verändert haben.

Der **Applikations-Server** stellt die funktionalen Dienste der Software als Web-Services zentral zur Verfügung. Er besteht aus einem Web Container und einem Enterprise Java Beans (EJB)-Container. Der Web Container beinhaltet eine Komponente zur situationsabhängigen Adaption sowie Web-Service-Schnittstellen zum EJB Container. Dieser beinhaltet die Kernfunktionalität der Software und wird um eine Serverseitige Adaptionskomponente (SSA) erweitert. Ein Anwendungsbeispiel hierfür ist die Vorverdichtung von zeit- und ressourcenintensiven Berechnungen, die mittels einer Mustererkennung zeitlich vorgelagert ausgeführt werden kann.

Für die situationsabhängige Adaption wird weiterhin ein spezieller **Adaptions-Server** (Adaptation Service Management) benötigt. Dieser besteht aus einen Web-Container für situationsabhängige Adaptionskomponenten und einem EJB-Container. Ersteres ist für rechenintensive oder serviceunabhängige Adaptionen hilfreich. Als Anwendungsbeispiel kann das Modifizieren eines Nutzerprofils genannt werden. Letzteres dient der Bearbeitung nicht zeitkritischer Adaptionsaufgaben, z.B. das Protokollieren der Web-Service-Aufrufe für die interne Leistungsverrechnung. Hierzu nimmt eine Java Messaging Service (JMS)-

Komponente entsprechende Anfragen aller Komponenten als Messages entgegen. Message-Driven-Beans im EJB-Container führen die entsprechenden Adaptionsoperationen durch.

6 Gesamtarchitektur des Komponenten-Frameworks

Die Gesamtarchitektur des Komponenten-Frameworks geht insbesondere auf die für das Management der Adaptionsprozesse benötigten Komponenten ein und zeigt das Zusammenspiel mit weiteren administrativen Komponenten verteilter Software auf. Es werden die folgenden Designkriterien zugrundegelegt:

- **Standardisiertes Kommunikations- und Informationsmanagement:** Die notwendigen Informationen zur Durchführung eines Adaptionsprozesses sind in der Regel auf den betroffenen Komponenten gespeichert. Lediglich bei der ersten Durchführung und bei Veränderung der Adaption ist eine Synchronisation der Komponenten notwendig. Über einen Synchronisationsbus werden die entsprechenden Komponenten mit den benötigten Informationen einheitlich versorgt.
- **Erweiterbarkeit auf mehrere Softwarekomponenten:** Die Architektur soll eine einheitliche Plattform darstellen, um die situationsabhängige Adaption bei mehreren Komponenten gegebenenfalls unterschiedlicher Softwaresysteme zu unterstützen. Der Entwicklungsaufwand für die Unterstützung neue Softwarekomponenten soll reduziert werden.

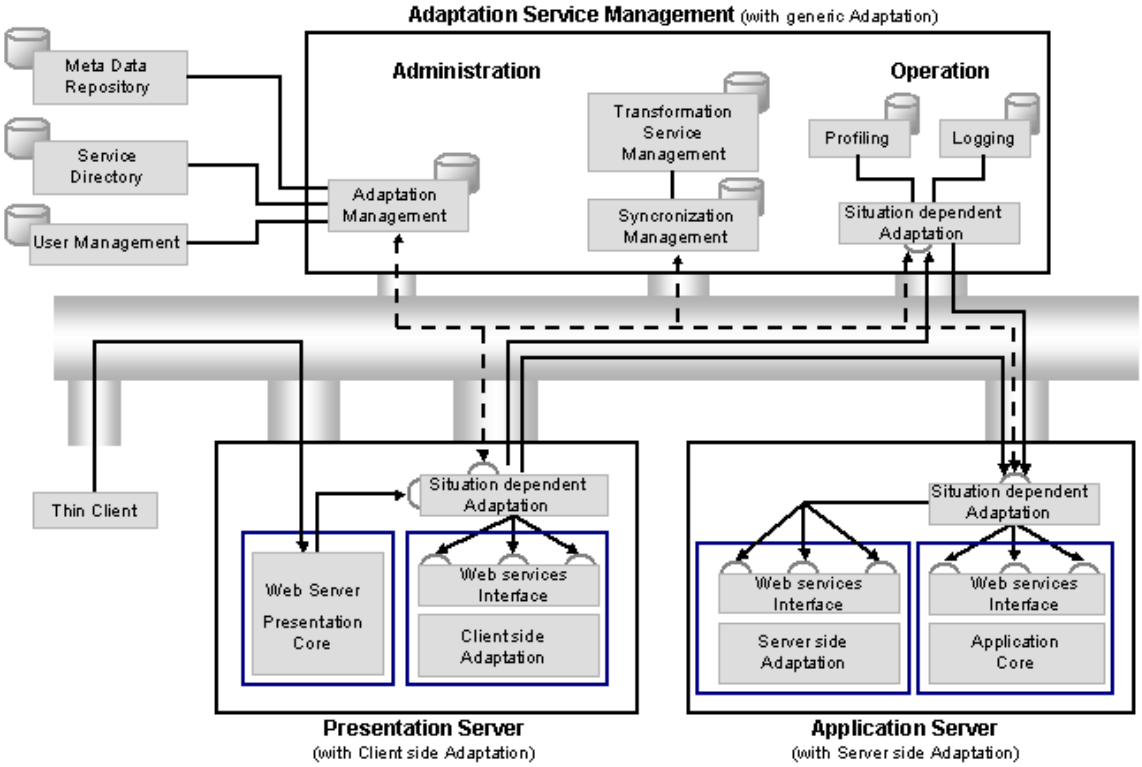

Bild *6* - Gesamtarchitektur des Komponenten-Frameworks

In der Gesamtarchitektur werden Komponenten zum Management des Gesamtsystems ergänzt (vgl. Bild 6). Neutrale Komponenten der Administration in verteilter Software sind Komponenten für die Benutzerverwaltung (User-Management), zum Verwalten von Services

(Service Directory) sowie zur Beschreibung der Services (Meta Data Repository). Diese werden auch für die situationsabhängige Adaption benötigt. Für das Adaptionsmanagement (Adaptation Service Management) werden mehrere Komponenten benötigt. Diese werden in administrative und operationale Komponenten unterteilt. Als **administrative Komponenten** werden hier die Komponenten für das übergeordnete Adaptionsmanagement (Adaptation Management), für das Synchronisationsmanagement der SDA-Komponenten (Synchronization Management) und zur Verwaltung der Transformationsvorschriften (Transformation Service Management) bezeichnet. Die wesentlichen Interaktionsbeziehungen sind ebenfalls in Bild 6 dargestellt. Die gestrichelten Kanten zeigen die Synchronisation im Gesamtsystem. Die Synchronisations-Management-Komponente aktualisiert die Daten (z.B. Profile) und synchronisiert andere Komponenten.

Als **operationale Komponenten** werden im Wesentlichen Komponenten zur Protokollierung von Transaktionsdaten, Benutzerverhalten, Antwortzeiten und sonstigen Nutzungsinformationen (Logging) sowie zur Mustererkennung in Nutzungsinformationen und für die automatischen Profilgenerierung (Profiling) benötigt. Die generierten Nutzungssituationen und Profile werden von den administrativen Komponenten verwendet. Weiterhin werden generische Adaptionskomponenten hinzugezählt.

Die Vorteile dieser Architektur liegen neben der losen Kopplung zur Kernsoftware insbesondere in der Performanz durch Reduktion des Verwaltungsaufwandes sowie in der Flexibilität hinsichtlich der Adaptionsprozesse. Durch die Komponentenorientierung wird das Gesamtsystem skalierbar und leichter erweiterbar. Die Aufteilung in Präsentations-Server und Applikations-Server bietet die nötige Flexibilität und die leichte Erweiterbarkeit für unterschiedliche Clients. Sogenannte Thin Clients (Web Browser, PDA-Browser,...) können über den Präsentations-Server flexibel eingebunden werden.

7 Zusammenfassung und Ausblick

Unter dem Begriff der situationsabhängigen Adaption wird die Anpassung einer Softwarefunktionalität an die aktuelle Nutzungssituation der Anwender verstanden. Eine situationsabhängige Software soll in der Lage sein, sich selbstständig an die speziellen Bedürfnisse der Anwender anzupassen und in ihren Tätigkeiten aktiv zu unterstützen. Aufgrund der besonderen Vorteile komponentenbasierter Software wurde im Rahmen dieser Arbeit die Entwicklung situationsabhängiger Adaption in Web-Service-basierter Standardsoftware näher untersucht. Web-Services als besondere Form der komponentenbasierten Software sind bei einem anwendungsnahen Design besonders gut für die Entwicklung adaptiver Software geeignet.

Zunächst wurde der Gesamtkontext der situationsabhängigen Adaption betrieblicher Standardsoftware beschrieben und es wurden die unterschiedlichen Teilziele einer Anwendung aufgezeigt. Der Kern der Arbeit besteht aus dem systematischen Aufbau eines Komponenten-Frameworks für die situationsabhängige Protokollierung und Adaption Web-Service-basierter Standardsoftware. Die Grundkonzeption zielt auf eine lose Kopplung zwischen Kernsoftware- und Adaptionsfunktionalität ab, um so eine nachträgliche Berücksichtigung bzw. Erweiterung der situationsabhängigen Adaption zu unterstützen. Für eine mehrstufige und performante Adaption sind spezielle Komponenten erforderlich. Diese werden in die Architektur des Gesamtsystems integriert. Sie stellt einen, gegebenenfalls auf mehrere Anwendungssysteme erweiterbaren standardisierten Rahmen für situationsabhängige Web-Service-basierte Standardsoftware bereit.

Die hier vorgestellten Konzepte für eine situationsabhängige Protokollierung und Adaption in Web-Service-basierter Standardsoftware werden bereits in einem Projekt angewendet und befinden sich in der Implementierungsphase. Nach Projektabschluss werden konkrete Erfahrungen über Aufwand und Nutzen des Komponenten-Frameworks vorliegen. Weiterhin ist noch zu klären, inwieweit der hier vorgestellte Ansatz auf andere Arten von Software übertragen werden kann.

8 Literatur

[AmFW2002] Amberg, M.; Figge, S.; Wehrmann, J. (2002): Compass – Ein Kooperationsmodell für situationsabhängige mobile Dienste: in Hampe, J. F.; Schwabe, G. (Hrsg.), Proceedings zur Teilkonferenz Mobile and Collaborative Business der Multikonferenz Wirtschaftsinformatik (MKWI 2002), Nürnberg, Deutschland.

[AmWe2002] Amberg, M.; Wehrmann, J. (2002): A Framework for the Classification of Situation Dependent Services: Eighth Americas Conference on Information Systems Proceedings (AMCIS 2002), pp. 1838-1843, Dallas, USA.

[BlBI2000] Blair, G.; Blair, L.; Issarny, V.; Tuma, P.; Zarras, A. (2000): The Role of Software Architecture in Containing Adaptation in Component-based Middleware Plattforms. Proceedings of Middleware 2000, IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing, Hudson River Valley (NY), USA.

[BaFK2000] Badrinath, B.; Fox, A.; Kleinrock, L.; Popek, G. (2000): A Conceptual Framework for Network and Client Adaptation. ACM MONET Journal, Vol. 5, No. 4, pp. 221-231.

[Gase2001] Gasenzer, R. (2001): Positionsbasierte Leistungsangebote für den mobilen Handel. In: Heilmann, H. (Hrsg.): *HMD - Praxis der Wirtschaftsinformatik*, Heft 220, S. 37-51, Heidelberg, Deutschland.

[GeJe2001] Gessler, S.; Jesse, K. (2001): Advanced Location Modeling to enable sophisticated LBS Provisioning in 3G networks. In: Beigl, M.; Gray, P.; Salber, D. (Hrsg.): *Proceedings of the Workshop on Location Modeling for Ubiquitous Computing*. Atlanta, USA.

[HiKR2002] Hitz, M.; Kappel, G.; Retschitzegger, W.; Schwinger, W. (2002): Ein UML-basiertes Framework zur Modellierung ubiquitärer Web-Anwendungen. In Wirtschaftsinformatik 44 (3/2002), S.225-235, Wiesbaden, Deutschland.

[KeBC2002] Ketfi, A.; Belkhatir, N.; Cunin, P.-Y. (2002): Automatic Adaptation of Component-based Software. PDPTA'02, Las Vegas, USA.

[May2001] May, P. (2001): *Mobile Commerce – Opportunities, Applications, and Technologies of Wireless Business*. Cambridge University Press, Cambridge, UK.

[OrGT1999] Oreizy, P.; Gorlick, M.M.; Taylor, R.N.; Heimbigner, D.; Johnson, G.; Medvidovic, N.; Quilici, A.; Rosenblum, D.S.; Wolf, A.L. (1999): An Architecture-Based Approach to Self-Adaptive Software in: IEEE Intelligent Systems, May/June 1999 (Vol. 14, No. 3), pp 54-62.

[OrHE1996] Orfali, R.; Harkey, D.; Edwards, J. (1996): The Essential Distributed Objects Survival Guide. John Wiley & Sons, New York, USA.

[Turo1999] Turowski, K. (1999): Ordnungsrahmen für komponentenbasierte betriebliche Anwendungssysteme. In: K. Turowski (Hrsg.): Tagungsband des 1. Workshops Komponentenorientierte betriebliche Anwendungssysteme (WKBA 1), S. 3-14, Magdeburg, Deutschland.

[WaSc2000] Wang, X.; Schulzrinne, H. (2000): An Integrated Resource Negotiation, Pricing, and QoS Adaptation Framework for Multimedia Applications. IEEE Journal on Selected Areas in Communications (JSAC), Vol. 18, No. 12, pp. 2514-2529.

[Zobe2001] Zobel, J. (2001): *Mobile Business und M-Commerce*. München, Deutschland.