Endbenutzergerechte Anpassung von serviceorientierten Softwaresystemen

Markus Hofmann, Benedikt Ley, Christian Dörner

Institut für Wirtschaftsinformatik und Neue Medien
Universität Siegen
Hölderlinstraße 3
57076 Siegen
{markus.hofmann, benedikt.ley, christian.doerner}@uni-siegen.de

Abstract: Anpassbarkeit ist ein wichtiges Qualitätskriterium moderner Softwaresysteme. Die Internationalisierung der Absatzmärkte verlangt eine hohe Flexibilität von Unternehmen und folglich auch von deren IT. Serviceorientierte Architekturen (SOA) eignen sich in besonderer Weise zur Implementierung von anpassbarer Software durch Endbenutzer, weil sie die Geschäftsprozesse eines Unternehmens in den Mittelpunkt des Systementwurfs stellen. Auf diese Weise können Fachanwender aus den einzelnen Unternehmensbereichen leichter in den Entwicklungs- und Anpassungsprozess der IT-Systeme eingebunden werden. Sie beteiligen sich an dieser Entwicklung und Anpassung durch die Orchestrierung von Geschäftsprozessen. Diese Anpassungsmöglichkeiten sind aus Sicht des End User Development bisher wenig betrachtet worden und sollen daher in diesem Aufsatz untersucht werden. Das Auffinden und Verstehen von Services sowie die endbenutzergerechte Abbildung von Geschäftsprozessen stehen im Mittelpunkt der Betrachtung. Die im Rahmen der Untersuchung festgestellten Unzulänglichkeiten stellen die Grundlage für die Entwicklung der End User Service Orchestration Platform (EUSOP) dar, die in diesem Aufsatz vorgestellt wird.

1 Einleitung

Die stetig zunehmende Internationalisierung der Märkte und die damit verbundene Verschärfung des Wettbewerbs fordern von Unternehmen ein hohes Maß an Flexibilität, um schnell auf sich verändernde Kundenwünsche und Handlungen der Wettbewerber reagieren zu können [Wu94]. Die dadurch entstandenen Veränderungen der Organisationsstrukturen wirken sich auch auf die IT der Unternehmen aus. Die Anpassbarkeit an veränderte Geschäftsprozesse zur Laufzeit der Systeme, bei gleichzeitig niedrigem Zeit- und Kostenaufwand, ist somit eine wichtige Anforderung an moderne Softwaresysteme. Anpassungen sollen nicht nur professionellen Entwicklern, sondern, durch angemessene Methoden des End User Development (EUD), auch den Endbenutzern[1] ermöglicht

[1] Unter Endbenutzern verstehen wir gemäß der Einteilung von Benutzergruppen nach Nardi und Miller [NM91] einerseits technisch nicht versierte *Non-Programmer*, die sich auf ihre fachliche Arbeit konzentrieren, sowie technisch versiertere *Local-Developer*, die über fortgeschrittene IT-Kenntnisse in ihrem jeweiligen Arbeitskontext verfügen.

werden. Das heißt, dass Endbenutzer beispielsweise durch Parametrisierung oder Customizing ihre Softwareumgebung an veränderte Anforderungen anpassen können [WSW06], [KTV07].

In diesem Kontext sind serviceorientierte Architekturen als Architekturparadigma von Informationssystemen von besonderem Interesse, weil diese, durch die lose Kopplung von Diensten, die Entwicklung von Anwendungen mit einem hohen Maß an Flexibilität ermöglichen. Derzeit sind Web Services die am weitesten verbreiteten Implementierungen von serviceorientierten Architekturen und die Anzahl der verfügbaren Web Services steigt stetig ([Ab06], [CLM03]). In einer serviceorientierten Architektur können Geschäftsprozesse explizit abgebildet und durch Sprachen wie beispielsweise die BUSINESS PROCESS EXECUTION LANGUAGE (BPEL)[2] beschrieben werden. Die einzelnen Aktivitäten innerhalb von Geschäftsprozessen werden durch Web Services repräsentiert. Mit BPEL lässt sich die kontrollierte Abfolge von Serviceaufrufen realisieren, wobei sowohl sequentielle, parallele als auch bedingte Kontrollflüsse möglich sind [AKC07]. Die Kombination mehrerer Web Services zu einem Geschäftsprozess wird Orchestrierung[3] genannt und durch Werkzeuge wie den ACTIVEBPEL DESIGNER[4] oder den WEBSPHERE BUSINESS MODELER[5] unterstützt. Die existierenden Orchestrierungswerkzeuge wurden für technisch versierte Programmierer entwickelt und sind zu komplex, als dass sie von Endbenutzern beherrscht werden können [LHM07]. Um diesem Problem entgegenzuwirken und Endbenutzern ein Werkzeug zum Entwurf von Geschäftsprozessen an die Hand zu geben, wurde die END USER SERVICE ORCHESTRATION PLATFORM (EUSOP) entwickelt, die in diesem Aufsatz vorgestellt wird.

Zu Beginn wird in Kapitel 2 der aktuelle Stand der Forschung beschrieben. Anschließend werden die Anforderungen an eine Orchestrierungsplattform aus der Perspektive des End User Development skizziert. In Kapitel 4 werden die Architektur und die Funktionalitäten der Plattform vorgestellt. Abschließend werden im letzten Kapitel die Ergebnisse zusammenfassend dargestellt und ein Ausblick auf Weiterentwicklungen der EUSOP gegeben.

2 Stand der Forschung

Zunächst sollen die grundlegenden Aspekte des Forschungsbereichs End User Development sowie die Basiskonzepte von serviceorientierten Architekturen dargestellt werden. Darauf folgend wird diskutiert, wie End User Development Ansätze in serviceorientierten Architekturen integriert werden könnten. Am Ende des Abschnitts werden anpassbare Architekturen und Plattformen vorgestellt, die ähnliche Funktionalitäten wie die EUSOP bieten.

---

[2] http://www.ibm.com/developerworks/library/specification/ws-bpel/
[3] In der Literatur sind im Zusammenhang mit dem Begriff Orchestrierung ebenfalls die Begriffe Komposition, Konversation oder Choreographie zu finden.
[4] http://www.active-endpoints.com/active-bpel-designer.htm
[5] http://www-111.ibm.com/ecatalog/Detail.wss?locale=de_DE&synkey=L106045O85235S77

2.1 Theoretische Grundlagen

Unter dem Begriff End User Development versteht man Tools, Methoden und Techniken, die es Endbenutzern ohne Programmierkenntnisse erlauben, ihre Softwareumgebung selbständig an ihre Bedürfnisse und Anforderungen anzupassen [Li06]. Endbenutzer können ihre Softwareumgebung zur Laufzeit gestalten, verändern und ergänzen. Im Rahmen der End User Development Forschung werden Methoden entwickelt, Software flexibler zu gestalten, so dass der Endbenutzer seine Software nicht nur leicht anwenden, sondern auch leicht an seine individuellen Bedürfnisse anpassen kann und sogar weiterentwickeln kann. Dazu muss er in der Lage sein, auf einfache Weise die Anpassungsmöglichkeiten einer Software erlernen und verstehen zu können. Im Allgemeinen lassen sich aufgrund der breiten Definition des Begriffs EUD verschiedene Komplexitätsstufen unterscheiden. Auf niedriger Komplexitätsstufe kann der Endbenutzer Möglichkeiten der Parametrisierung und des Customizings einsetzen. Solche Mechanismen werden bereits von vielen Softwaresystemen angeboten, und finden sich beispielsweise in MICROSOFT WORD in Form von anpassbaren Symbolleisten und Optionseinstellungen wieder. Derartige Systeme können auf verschiedene Arten vom Endbenutzer eingerichtet und konfiguriert werden. Die vorhandenen Auswahlmöglichkeiten müssen allerdings bereits während des Entwurfs der Software von den Entwicklern berücksichtigt werden. Im Gegensatz dazu umfassen EUD Methoden auf hoher Komplexitätsstufe die Erstellung und Modifikation von Software. Endbenutzer können beispielsweise Makros programmieren, um Tabellen in einem Kalkulationsprogramm – wie MICROSOFT EXCEL – automatisch umzuordnen oder sie können visuelle Programmiersprachen nutzen, um Anwendungen anzupassen [Li06]. Durch EUD-Ansätze wird es Endbenutzern möglich, auf veränderte Anforderungen im Arbeitskontext flexibel zu reagieren. Eine gute Übersicht über aktuelle EUD-Ansätze bietet das Buch END-USER DEVELOPMENT [LPW06].

Vor dem Ziel den Entwicklungsprozess von Softwaresystemen effizienter zu gestalten, wurden verschiedene Paradigmen wie beispielsweise objektorientierte Programmierung, komponentenbasierte Entwicklung oder serviceorientierte Architekturen entwickelt, die die Wiederverwendbarkeit von Quellcode verbessern [BBM96]. Das Architekturparadigma der serviceorientierten Architektur unterscheidet sich von anderen Architekturparadigmen, wie dem Client-Server Paradigma, in erster Linie darin, dass nicht mehr die Applikation an sich, sondern die Geschäftsprozesse des Unternehmens im Mittelpunkt der Betrachtung stehen. Im Folgenden werden die wichtigsten Merkmale und Eigenschaften von serviceorientierten Architekturen beschrieben. Als konkrete Implementierung gehen wir dabei immer von Web Services aus.

Applikationen auf Basis einer SOA werden aus einer Menge von Services zusammengesetzt, die sich auf unterschiedlichen Systemen befinden können und lose miteinander gekoppelt, d. h. logisch vollkommen voneinander getrennt sind. Durch dynamisches Binden besteht die Möglichkeit, zur Laufzeit der Anwendung neue Services zu suchen, deren Funktionalität zu interpretieren und diese in die Applikation zu integrieren. Die Plattform- und Programmiersprachenunabhängigkeit und dadurch die Interoperabilität von Services wird durch die Verwendung offener Standards gewährleistet. Die Kommunikation zwischen verschiedenen Services wird in der Regel durch das auf XML

basierende SIMPLE OBJECT ACCESS PROTOCOL (SOAP)[6] realisiert. Eine Alternative zu SOAP stellt die REPRESENTATIONAL STATE TRANSFER (REST)[7] Architektur dar, die beispielsweise in Mash-up Anwendungen, wie YAHOO! PIPES[8], eingesetzt wird. Der Aufbau einer serviceorientierten Architektur, bestehend aus Serviceanbieter, Servicenutzer und Serviceverzeichnis, ist in Abbildung 1 dargestellt.

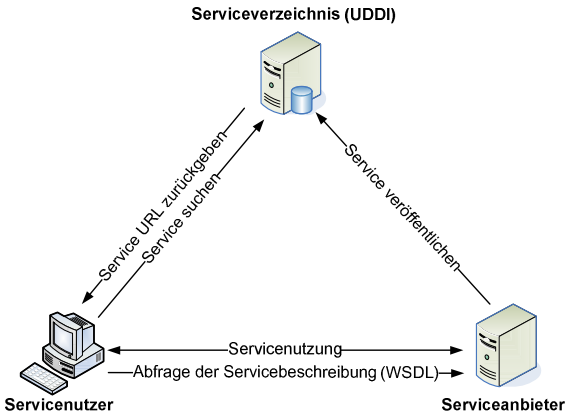

Abbildung 1: Aufbau einer serviceorientierten Architektur

UNIVERSAL DESCRIPTION, DISCOVERY AND INTEGRATION (UDDI)[9] Dienste realisieren ein Serviceverzeichnis, das das Auffinden von Services erleichtert. Sie sind allerdings kein zwingend notwendiger Bestandteil einer SOA. Serviceanbieter können ihre Dienste im UDDI registrieren und Servicenutzer können an das Serviceverzeichnis Suchanfragen stellen. Als Antwort erhält der Servicenutzer den Service Endpoint (eine URL) des gesuchten Dienstes, über die er die Beschreibung des Services beim Serviceanbieter abrufen kann. Die Informationen über die Funktionen des Services und dessen Schnittstelle werden in so genannten WSDL (WEB SERVICE DESCRIPTION LANGUAGE)[10] Dateien beschrieben. Mittels der Informationen aus der WSDL Datei kann der Servicenutzer anschließend den Service mittels eines SOAP-Calls mit den erforderlichen Parametern aufrufen.

BPEL ist eine auf XML basierende Sprache, die der Standard zur Beschreibung von Geschäftsprozessen auf Basis von Web Services ist [Si07]. Die einzelnen Aktivitäten eines Geschäftsprozesses werden durch Web Services implementiert. Mittels verschiedener Kontrollstrukturen werden die Aufrufe der einzelnen Services aufeinander abgestimmt. Die Kombination mehrerer Web Services wird Orchestrierung genannt. Orchestrierte BPEL Prozesse können wiederum als Web Services aufgerufen werden, so

[6] http://www.w3.org/TR/soap/
[7] http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
[8] http://pipes.yahoo.com/pipes/
[9] http://uddi.org/pubs/uddi-v3.0.2-20041019.htm
[10] http://www.w3.org/TR/wsdl20/

dass sich atomare Services zu komplexen Services zusammensetzen lassen. Die im Juni 2007 verabschiedete Erweiterung BPEL4People[11] ermöglicht zusätzlich die Integration von menschlichen Aktivitäten in den Geschäftsprozess.

2.2 End User Development in serviceorientierten Architekturen

Die Flexibilität von serviceorientierten Architekturen verändert nicht nur den Entwicklungsprozess von Informationssystemen, sondern bietet auch gute Möglichkeiten für die Integration von End User Development Ansätzen. Durch die Trennung von Fachkonzept und Schnittstellenbeschreibung eines Web Services kommt der Endbenutzer nicht direkt mit der technischen Implementierung in Kontakt. Endbenutzer orientieren sich in der Praxis an Geschäftsprozessen, um die Aufgaben ihres Arbeitsalltages zu erledigen (z. B. der Wareneingang im ERP-System, wobei die Folge der benötigten Masken dem Anwender bekannt ist). Die Prozesse sind daher nahe an der Denkwelt des Fachanwenders, aber nicht zwangsweise nahe an der Fachlogik des genutzten Systems.

In einer serviceorientierten Architektur stehen dagegen, wie oben erläutert, die Geschäftsprozesse im Mittelpunkt der gesamten Softwareumgebung. Da die Endbenutzer über das implizite Prozesswissen verfügen, sollte eine explizite Abbildung der Geschäftsprozesse unter Beteiligung der Endbenutzer erfolgen. Softwarelandschaften, die auf serviceorientierten Architekturen aufbauen, bestehen aus implementierten Geschäftsprozessen. Durch Änderungen an diesen Prozessen kann die Softwareumgebung folglich unmittelbar an neue Anforderungen angepasst werden, ohne dass ein hoher Transformationsaufwand innerhalb der Softwarearchitektur notwendig ist. Die explizite Speicherung der Geschäftsprozesse ermöglicht zudem, dass einmal orchestrierte Prozesse kooperativ genutzt und weiterentwickelt werden können.

Die Umsetzung des zuvor beschriebenen Ansatzes zur Anpassung von Softwaresystemen stellt sich zunächst als schwierig heraus, weil Web Services ursprünglich für die Intermaschinenkommunikation entwickelt wurden. Die auf XML basierenden Dateien und Protokolle, wie BPEL, WSDL und SOAP sind, da sie als Klartext formatiert sind, für den Menschen prinzipiell lesbar, können allerdings nur mit Fachkenntnissen der Syntax und Semantik der entsprechenden Sprachen verstanden werden. Das Aneignen dieser Fachkenntnisse kann von einem Endbenutzer nicht erwartet werden [LHM07]. Selbst für professionelle Softwareentwickler sind komplexe Services durch den Umfang der Schnittstellenbeschreibungen ohne Hilfsmittel nur schwer verständlich. Hinzu kommt, dass Geschäftsprozesse nur dann anpasst werden können, wenn die benötigten Services zuvor gefunden und ausgewählt wurden. Die Qualität der Suche nach Services ist allerdings von den Informationen abhängig, die der Serviceanbieter im UDDI einträgt. Diese Informationen sind häufig unpräzise oder veraltet. Zudem werden Quality of Service (QoS) Merkmale wie Servicelaufzeiten, Ausfallsicherheit, Ergebnisqualität oder Robustheit, sowie weitere für den Endbenutzer relevante, nicht-funktionale Informationen vom Serviceprovider häufig nicht angegeben [Ab06].

[11] http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/

Die technischen Beschreibungen innerhalb der WSDL Datei, beispielsweise Methoden- und Attributbezeichner, sind oft nicht selbsterklärend und daher für Endbenutzer ohne zusätzliche nicht-funktionale Beschreibungen selten verständlich. WSDL bietet die Möglichkeit, durch so genannte Documentation-Tags, Beschreibungen zu Elementen, wie beispielsweise Methoden oder Attributen, hinzuzufügen. Diese Möglichkeiten werden jedoch nur selten durch den Serviceanbieter genutzt und zielen in der Regel auf Softwareentwickler ab [KHD06]. Für die semantische Beschreibung von Web Services gibt es verschiedene Ansätze wie OSRR (ONTOLOGY-ENHANCED SEMANTIC REQUEST AND RESPONSE) [Sh06] oder WSMX (WEB SERVICE MODELLING EXECUTION ENVIRONMENT) [Mo06], die sich mit dieser Problematik beschäftigen. Viele dieser Ansätze basieren auf der OWL (WEB ONTOLOGY LANGUAGE), einem W3C Standard [DS04] und zielen meist darauf ab, Services automatisch zur Laufzeit zu orchestrieren [Mo06, SPH04].

2.3 Verwandte Arbeiten

FREEVOLVE [WSW06] ist eine Open Source Plattform, die aus der EVOLVE Plattform von Stiemerling entstanden ist [St00]. Sie ermöglicht es den Nutzern, Anpassungen der Software zur Laufzeit des Systems vorzunehmen. Die Plattform basiert auf dem FLEXIBEANS Modell, das eine Erweiterung des JavaBeans Komponentenmodells ist und Änderungen von Systemkomponenten zur Laufzeit zulässt. FREEVOLVE bietet verschiedene graphische Anpassungsumgebungen, mit denen der Benutzer das System verändern kann. Er kann zur Laufzeit in die Anpassungsumgebung wechseln, Veränderungen am System vornehmen und wieder zurück in die laufende Anwendung wechseln und sofort die zuvor gemachten Anpassungen nutzen. Komponenten können hinzugefügt, verschoben, verbunden oder gelöscht werden. Alle Informationen über gemachte Veränderungen werden in Konfigurationsdateien gespeichert, die mit anderen Nutzern über ein zentrales Verzeichnis ausgetauscht werden können.

Die TAILORBPEL-PLATTFORM [AKC07] ermöglicht es Endbenutzern, orchestrierte Web Services mittels eines graphischen Editors zur Laufzeit zu verändern. Endbenutzer können Services, die mit BPEL orchestriert wurden, durch Änderungen an der Zusammenstellung und Anordnung der BPEL Elemente anpassen. Diese Änderungen an den Prozessen werden personalisiert abgespeichert, können allerdings auch mit anderen Benutzern geteilt werden. Die Plattform umfasst außerdem eine Komponente, die das Auffinden von relevanten Services durch Endbenutzer erleichtern soll.

Die ADAPTIVE SERVICES GRID (ASG) Plattform [Ab06] basiert auf der Idee, dass gute und aktuelle Beschreibungen von Services benötigt werden, um sie in der Praxis einsetzen zu können. Die Beschreibungen der Serviceanbieter sind dazu nicht ausreichend, da sie nur statisch sind und nicht-funktionale Aspekte nur sehr selten abgedeckt werden. Daher werden diese Beschreibungen um Benutzer-Feedback und Monitoringdaten ergänzt. Endbenutzer werden von der Plattform bei der Suche nach geeigneten Services, jedoch nicht bei deren Orchestrierung unterstützt.

Die SERVICE COMPONENT ARCHITECTURE (SCA)[12] ermöglicht die Zusammensetzung von Komponenten auf verschiedenen Abstraktionsebenen, die beispielsweise Java-Klassen, Web Services oder durch BPEL orchestrierte Services integrieren. Darauf aufbauend werden lauffähige Applikationen aus Komponenten zusammengesetzt und spezifiziert, wie die Interaktion zwischen den einzelnen Modulen abläuft. Die Spezifikationen erlauben herstellerspezifische Ergänzungen. Es gibt unterschiedliche Implementierungen von SCA, wie APACHE TUSCANY[13] oder ROGUE WAVE HYDRASCA[14], die sich auf verschiedene Aspekte konzentrieren. Während BPEL die Geschäftslogik eines Geschäftsprozesses als eine Sequenz von Services abbildet, konzentriert sich SCA darauf, die Struktur einer Applikation zu beschreiben. Daher können BPEL und SCA einander ergänzend eingesetzt werden [ZD06].

Allen Plattformen ist gemein, dass sie entweder EUD Methoden implementieren oder auf Basis einer SOA implementiert sind. Die EUSOP vereint beide Ansätze und bietet Endbenutzern eine beherrschbare Anpassungsumgebung.

3 Anforderungen an die EUSOP

Wie bereits in Kapitel 2.2 erwähnt, gibt es zahlreiche Probleme, Endbenutzern service-orientierte Architekturen zugänglich zu machen. Um diesen Problemen entgegen zu wirken, wurde die END USER SERVICE ORCHESTRATION PLATFORM entwickelt, die es Endbenutzern als Experten ihrer jeweiligen Domäne ermöglichen soll, Anpassungen an Geschäftsprozessen vorzunehmen. Die Endbenutzer müssen dabei beim Auffinden und Verstehen von relevanten Services, bei deren Nutzung und bei der Komposition verschiedener Services unterstützt werden. Letzteres sollte kooperativ in einer Nutzergemeinschaft erfolgen können, da Softwaresysteme, die in Organisationen eingesetzt werden, kollaborative Aktivitäten der Endbenutzer unterstützen sollten [PK06]. Ein in diesem Kontext interessantes und beispielhaftes Anwendungsszenario stellt Brahes und Schmidts empirische Studie der Arbeitspraxis einer Bank dar, in der kooperativ auf Basis einer serviceorientierten Architektur ein zentraler Geschäftsprozess modelliert wurde [BS07].

Im Rahmen einer Vorstudie wurde in fünf Unternehmen analysiert, welche Probleme Endbenutzer mit ihrer Softwareumgebung haben und wie sie versuchen, diese Probleme zu lösen [DHP07]. Darauf aufbauend wurde mit drei Endbenutzern aus einem der an der Studie beteiligten Unternehmen ein Participatory Design Workshop durchgeführt. Der Workshop brachte Erkenntnisse darüber, wie Endbenutzer für ein konkretes Szenario aus ihrer Arbeitspraxis einen Geschäftsprozess mit Hilfe von Web Services modellieren würden. Auf Basis der Ergebnisse beider Untersuchungen, sowie des aktuellen Standes der Forschung, wurden Anforderungen ausgearbeitet, die die Grundlage für die Entwicklung der EUSOP bilden:

---

[12] http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications
[13] http://incubator.apache.org/tuscany/
[14] http://www.roguewave.com/hydra/hydrasca.cfm

- **Speicherung zusätzlicher UDDI-Metadaten:** Die Metadaten, die zu einem Service oder einem Serviceprovider im UDDI gespeichert sind, sollen um beliebige Metainformationen ergänzt werden können. Solche Metadaten können zum Beispiel Kommentare, Bewertungen, Erfahrungsberichte oder Schlüsselwörter sein. Durch zusätzliche Metainformationen soll das Auffinden und die Selektion geeigneter Services erleichtert werden. Metadaten sollen domänenspezifisch frei definiert werden können.

- **Erweiterbare Dokumentation in WSDL-Dateien:** In WSDL-Dateien können jegliche Elemente, wie beispielsweise Operationen oder Attribute, mit einer Dokumentation versehen werden. Diese Beschreibungen dienen der besseren Verständlichkeit und haben keinen Einfluss auf die Funktionalität des Services. Da die Möglichkeit dieser Dokumentationen vom Serviceanbieter oft nur schlecht oder überhaupt nicht genutzt wird, sollen Endbenutzer Beschreibungen selbst ergänzen können, die danach auch anderen Benutzern zur Verfügung stehen.

- **Schnittstelle für die graphische Komposition von Services:** Um Endbenutzern die Orchestrierung von Services zu ermöglichen, brauchen diese eine benutzerfreundliche graphische Oberfläche. Es sollen daher Schnittstellen entwickelt werden, an die eine graphische Oberfläche angebunden werden kann. Der Entwickler einer solchen Oberfläche kommt durch diese Kapselung nicht mehr mit den tiefer liegenden technischen Details sowie den XML-Dateien und -Protokollen in Kontakt.

- **Benutzer- und Rollenkonzept:** Es soll möglich sein, verschiedene Rollen zu definieren, die unterschiedliche Sichten auf die Plattform realisieren. Die Sichten unterscheiden sich im Umfang der zur Verfügung stehenden Funktionen. Dadurch lassen sich beispielsweise Endanwender und Geschäftsprozessdesigner besser unterscheiden.

4 EUSOP System

In diesem Kapitel wird die Architektur der EUSOP vorgestellt. Im ersten Schritt der Entwicklung wurde zunächst die Fachkonzeptschicht der EUSOP implementiert. Zur Implementierung von EUSOP wurden bestehende Open Source Lösungen verwendet. Die Plattform wurde in JAVA entwickelt und nutzt APACHE TOMCAT[15] als Applikationsserver. Abbildung 2 gibt einen Überblick über den Aufbau der Plattform und zeigt, wie die verschiedenen Komponenten zueinander in Beziehung stehen.

[15] http://tomcat.apache.org/

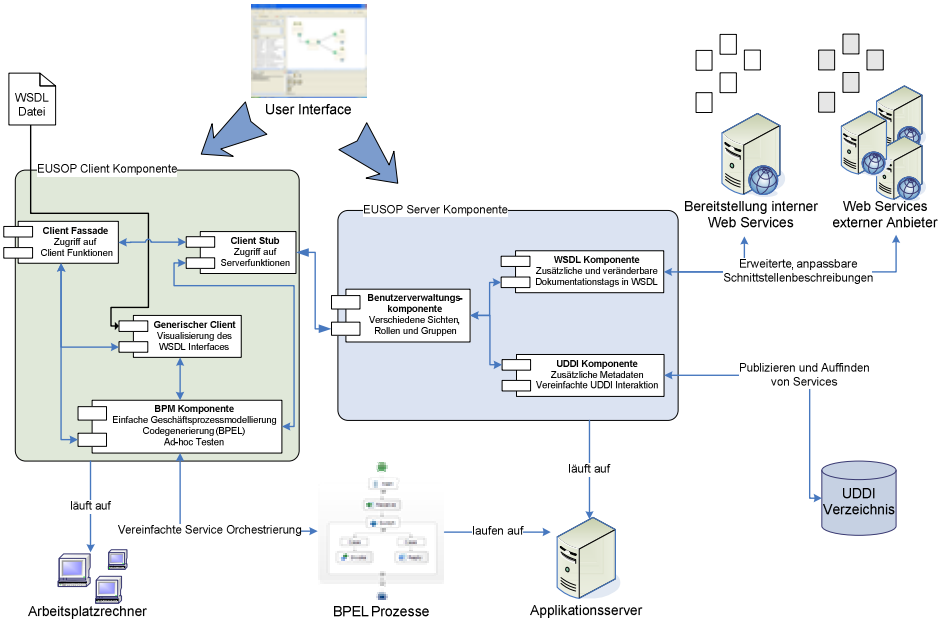

Abbildung 2: EUSOP - End User Service Orchestration Platform

Die Plattform besteht aus einer Client und einer Server Komponente. Die Client Komponente umfasst die *Client Fassade*, den *generischen Client*, die *Business Process Modeling (BPM) Komponente* sowie den *Client Stub*. Die Server Komponente besteht aus der *Benutzerverwaltung*, der *WSDL Komponente* und der *UDDI Komponente*.

Die *Client Fassade* ermöglicht den Zugriff auf sämtliche Funktionen der Client Komponente. Über den *Client Stub* wird die Verbindung zwischen Client und Server Komponente hergestellt. Um einen Web Service ansprechen, testen und nutzen zu können, benötigt man in der Regel einen speziell entwickelten Client, der die Schnittstelle des Services darstellt. Die EUSOP verfügt hingegen über einen *generischen Client*, der dynamisch die Elemente der WSDL-Datei eines Services interpretiert. Dazu erzeugt der *generische Client* innerhalb der Java Laufzeitumgebung Stellvertreterobjekte für die jeweiligen Web Services.

Die *BPM Komponente* baut auf einem universellen Komponenten- und Containermodell auf, das an die META OBJECT FACILITY (MOF)[16] angelehnt ist und die sprachunabhängige Modellierung von Geschäftsprozessen ermöglicht. Eine graphische Oberfläche, die derzeit auf Basis der EUSOP entwickelt wird, nutzt die Box-and-Wire-Metapher zur vereinfachten Darstellung der Serviceorchestrierung. Hierbei werden die Methoden eines Services als rechteckige Container (Boxes) dargestellt, deren Ein- und Ausgänge über Linien (Wires) miteinander verbunden werden. Die EUSOP *BPM Komponente* prüft dabei, ob die Datentypen der zu verbindenden Services syntaktisch zueinander passen.

[16] http://www.omg.org/mof/

Aus dem fertigen Modell lässt sich anschließend BPEL Code generieren, der ad-hoc getestet und auf der Server Komponente veröffentlicht werden kann.

Die serverseitige *Benutzerverwaltung* bietet ein Rollen- und Sichtenkonzept, bei dem zwischen Servicenutzern (Non-Programmer), Serviceentwicklern (Local-Developer) und Administratoren unterschieden wird. In Abhängigkeit der jeweiligen Rolle werden unterschiedliche Sichten und Funktionalitäten zur Verfügung gestellt. Servicenutzer können Services suchen, ausführen sowie Services beschreiben, bewerten und kommentieren. Serviceentwickler haben zusätzlich die Möglichkeit, Services zu orchestrieren und diese Orchestrierungen als Abbildungen von Geschäftsprozessen der Community zur Verfügung zu stellen. Außerdem können sie existierende Geschäftsprozesse verändern. Administratoren können zudem Einstellungen an der Systemkonfiguration vornehmen.

Die *WSDL Komponente* bietet die Möglichkeit, zusätzliche Beschreibungen innerhalb der WSDL Dateien hinzuzufügen oder diese zu verändern. So können beispielsweise einzelne Operationen oder Attribute annotiert werden. Die zusätzlichen Informationen werden in den Documentation-Tags gespeichert, um konform zum WSDL Standard zu bleiben. Da die WSDL Datei beim Serviceanbieter liegt, ist es nicht möglich, diese Datei direkt zu verändern. Deshalb werden die Beschreibungen dynamisch über einen EXTENSIBLE STYLESHEET LANGUAGE TRANSFORMATION (XSLT) [Cl99] Prozess beim Abruf der WSDL Datei ergänzt. Die nötigen Transformationsregeln werden auf dem EUSOP-Server abgespeichert.

Die *UDDI Komponente* basiert auf einem modifizierten APACHE JUDDI[17] Server. Die API-Aufrufe des UDDI-Dienstes wurden vereinfacht. Es können beliebige Metadaten, in Form von Schlüssel-Wert-Paaren, zu Services oder Serviceanbietern, die im UDDI gespeichert sind, ergänzt oder verändert werden. Das Verändern oder Ergänzen von Metadaten erleichtert Endbenutzern die Suche nach geeigneten Services, da diese nicht mehr ausschließlich von den Serviceanbietern, sondern auch von der Nutzergemeinschaft zur Verfügung gestellt werden.

5 Ausblick

Eine serviceorientierte Architektur stellt die Geschäftsprozesse einer Organisation in den Vordergrund und ist folglich nahe am Arbeitskontext der Endbenutzer. Da serviceorientierte Architekturen nicht für die Anpassung durch Endbenutzer entwickelt wurden, ist eine Realisierung von End User Development Ansätzen in solchen Softwareumgebungen derzeit mit zahlreichen Problemen behaftet. Die in diesem Aufsatz vorgestellte EUSOP stellt einen ersten Ansatz dar, um diese Probleme zu lösen und Endbenutzer zu befähigen, Anpassungen an einer solchen Softwareumgebung selbst vorzunehmen. Um Endbenutzern die Suche nach Web Services und das Verständnis dieser Services zu erleichtern, wurden zusätzliche Beschreibungsmöglichkeiten (in der WSDL Datei sowie im UDDI) von Web Services in der EUSOP realisiert. Das

[17] http://ws.apache.org/juddi/

sprachunabhängige Datenmodell der Orchestrierung ermöglicht eine Zusammenstellung von Services ohne Kenntnisse der unterliegenden Technologien und Implementierungen. Derzeit kann dieses Modell von der BPM-Komponente in BPEL-Code transformiert werden. Geschäftsprozesse, die auf diese Weise orchestriert wurden, können mittels der Plattform veröffentlicht und somit von anderen Endbenutzern verwendet und verändert werden. Auf Basis der bereits implementierten Fachkonzeptschicht wird zeitnah eine graphische Oberfläche entwickelt, die auf der Box-and-Wire-Metapher aufbaut und die endbenutzergerechte Modellierung von Geschäftsprozessen ermöglichen soll.

In nachfolgenden Arbeiten soll unter anderem ein Vorschlagssystem entwickelt werden, das Endbenutzern geeignete Web Services für ihre Orchestrierung vorschlägt. Ergänzend zur realisierten modellgetriebenen Generierung von BPEL-Code soll eine Anbindung an die SERVICE COMPONENT ARCHITECTURE entwickelt werden. Nach Abschluss dieser Arbeiten ist die Evaluierung des Gesamtkonzeptes geplant, die im aktuellen Stadium der Entwicklung noch nicht umgesetzt werden konnte. Bevor die Evaluierung erfolgen kann ist es notwendig, mit Anwendern aus einer Organisation eine Basis von atomaren und orchestrierten Services zu entwickeln, um auf dieser Grundlage ein evolutionäres Wachstum der Plattform zu ermöglichen. In regelmäßigen Abständen soll die gewachsene Landschaft untersucht und gegebenenfalls wieder geordnet und formalisiert werden [Fi94]. Weiterhin ist es interessant, über die Integration von BPEL4PEOPLE in die Plattform nachzudenken, um die Interaktion zwischen Menschen und Services zu unterstützen, da derzeitig keine menschlichen Interaktionen als Teil der Geschäftsprozesse modelliert werden können.

6 Danksagungen

Die Forschungsarbeit wurde durch das Bundesministerium für Bildung und Forschung (BMBF) im Rahmen der Forschungsoffensive "Software Engineering 2006" (Förderkennzeichen: 01 IS E03 A), sowie durch die Deutsche Forschungsgemeinschaft (DFG) innerhalb des SFB/FK 615 „Medienumbrüche“ gefördert. Wir danken Moritz Weber für seine Ideen bei der Entwicklung der EUSOP sowie für die angestoßenen kritischen Diskussionen. Außerdem danken wir Mirko Heinbuch und Min Wang für ihren Einsatz bei der Entwicklung der Plattform.

Literaturverzeichnis

[Ab06] Abramowicz, W. et al.: Architecture for Service Profiling. In: IEEE Services Computing Workshops (SCW'06), 2006, S. 121-130.

[AKC07] Alda, S.; Kuck,J.; Cremers, A. B.: Tailorability of personalized BPEL-based Workflow Compositions. In: Proceedings of 1st International Workshop on Web Service Composition and Adaptation (WSCA-2007), IEEE Publishing. Salt Lake City, 2007.

[BBM96] Basili, V. R.; Briand, L. C.; Melo, W. L.: How reuse influences productivity in object-oriented systems. In: Communications of the ACM, Vol. 39, Nr. 10, ACM Press, 1996, S. 104-116.

[BS07] Brahe, S.; Schmidt, K.: The Story of a Working Workflow Management System. In:

Proceedings of the 2007 international ACM conference on Conference on supporting group work, Sanibel Island, Florida, 2007, S. 249-258.

[Cl99] Clark, J. Hrsg.: XSL Transformations (XSLT) Version 1.0, http://www.w3.org/TR/xslt, Abruf am 26.09.2007.

[CLM03] Chung, J.-Y.; Lin, K.-J.; Mathieu, R. G.: Web Services Computing: Advancing Software Interoperability. In: Computer Magazine, Vol. 36, Nr. 10, IEEE Computer Society, 2003, S. 35-37.

[DHP07] Dörner, C.; Heß, J.; Pipek, V.: Improving Information Systems By End User Development: A Case Study. In: Proceedings of the ECIS2007, St. Gallen, 2007, S. 783-794.

[DS04] Dean, M; Schreiber, G. Hrsg.: OWL Web Ontology Language – Reference, http://www.w3.org/TR/owl-ref/ Abruf am 25.09.2007.

[Fi94] Fischer, G. et al: Seeding, Evolutionary Growth and Reseeding: Supporting the Incremental Development of Design Environments. In: HumanFactors in Computing Systems, CHI'94 Conference Proceedings, 1994, S. 292-298.

[KHD06] Kokash, N; van den Heuvel, W.-J.; D'Anderea, V.: Leveraging Web Services Discovery with Customizable Hybrid Matching. In (Dan, A.; Lamersdorf, W., Hrsg.): Service-Oriented Computing – ICSOC 2006, Springer, 2006, S. 522-528.

[KTV07] Kilian-Kehr, R.; Terzidis, O.; Voelz, D.: Industrialisation of the Software Sector. In: Wirtschaftsinformatik 49, Sonderheft 2007, S.62-71.

[Li06] Lieberman, H. et al.: End-User Development: An Emerging Paradigm. In (Lieberman, H. et al. Hrsg.): End-User Development. Springer, Dordrecht, 2006, S. 1-8.

[LPW06] Liebermann, H; Paterno, F.; Wulf, V. Hrsg.: End-User Development, Springer, Dordrecht, 2006.

[LHM07] Liu, X.; Huang, G.; Mei, H.: Towards End User Service Composition, 31st Annual International Computer Software and Applications Conference Vol. 1 (COMPSAC 2007), 2007, S. 676-678.

[Mo06] Mocan, A. et al.: Filling the Gap – Extending Service Oriented Architectures with Semantics. In: e-Business Engineering. ICEBE '06. IEEE International Conference, 2006, S. 594-601.

[NM91] Nardi, B. A.; Miller, J. R.: Twinkling lights and nested loops: distributed problem solving and spreadsheet development. In: International Journal of Man-Machine Studies, Vol 34, Nr. 2, Academic Press Ltd., London, 1991, S. 161-184.

[PK06] Pipek, V.; Kahler, H.: Supportung Collaborative Tailoring – Issues and Approaches. In (Lieberman, H. et al. Hrsg.): End-User Development. Springer, Dordrecht, 2006, S. 315-345.

[Sh06] Shi, X.: Sharing Service Semantics using SOAP-Based and REST Web Services. In: IT Professional, Vol.8, Nr. 2, Mär/Apr 2006, S.18-24.

[Si07] Siedersleben, J.: SOA revisited: Komponentenorientierung bei Systemlandschaften. In: Wirtschaftsinformatik 49, Sonderheft 2007, S. 110-117.

[SPH04] Sirin, E.; Parsia, B.; Hendler, J.: Filtering and Selecting Semantic Web Services with Interactive Composition Techniques. In: Intelligent Systems, IEEE 19, 2004, S. 42-49.

[St00] Stiemerling, O.: Component-Based Tailorability. Ph.D Thesis, Universität Bonn, 05/2000.

[WSW06] Won, M.; Stimmerling, O.; Wulf, V.: Component-Based Approaches to Tailorable Systems. In (Lieberman, H. et al. Hrsg.): End-User Development. Springer, Dordrecht, 2006, S. 115-141.

[Wu94] Wulf, V.: Anpaßbarkeit im Prozeß evolutionärer Systementwicklung. In: GMD-Spiegel, 24. Jg., 3/1994, S. 41-46.

[ZD06] Zou, Z.; Duan, Z.: Building Business Processes or Assembling Service Components: Reuse Services with BPEL4WS and SCA, European Conference on Web Services (ECOWS'06), 2006, S.138-147.