Universität Siegen
Fachbereich 5 – Wirtschaftswissenschaften
Institut für Wirtschaftsinformatik und Neue Medien
Prof. Dr. Volker Wulf

UNIVERSITÄT SIEGEN

Diplomarbeit

Vereinfachte Modifikation serviceorientierter Software durch Abstraktion und Transformation am Beispiel von Web Services

Autor: Frank Paczynski
Betreuer: Dipl-Wirt.-Inf. Christian Dörner
Erstprüfer: Prof. Dr. Volkmar Pipek
Zweitprüfer: Prof. Dr. Volker Wulf
Eingereicht: 10.06.2008

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, insbesondere keine anderen als die angegebenen Informationen aus dem Internet. Diejenigen Paragraphen der für mich gültigen Prüfungsordnung, welche etwaige Betrugsversuche betreffen, habe ich zur Kenntnis genommen. Die Speicherung meiner Diplomarbeit zum Zweck der Plagiatsprüfung stimme ich zu. Ich versichere, dass die elektronische Version mit der gedruckten Version inhaltlich übereinstimmt.

Siegen, den 10.06.2008

_____

Frank Paczynski

Inhaltsangabe

Durch die Dynamik der globalisierten Märkte ist Flexibilität zu einer Grundvoraussetzung für Unternehmen geworden, welche ebenfalls auf der Ebene der Informationssysteme umgesetzt werden muss. Eine ökonomische und zeitnahe Realisierung kann jedoch lediglich durch die Einbeziehung der Fähigkeiten und des Fachwissens von Endbenutzern[1] aus den jeweiligen Fachabteilungen erfolgen. Das sich stark ausbreitende Paradigma serviceorientierter Architekturen (SOA) ermöglicht in Form von Web Services und der hierauf aufbauenden Geschäftsprozessmodellierungssprache BPEL (Business Process Engineering Language) flexibel anpassbare Informationssysteme. Entsprechende Editoren setzen jedoch ein umfangreiches technisches Wissen voraus, was deren Verwendung im Endbenutzerkontext nahezu ausschließt. Das in der vorliegenden Arbeit entwickelte Konzept beschreibt einen grafischen BPEL-Editor, welcher unter anderem durch die Berücksichtigung domänenspezifischer, kollaborativer und kognitiver Faktoren für Endbenutzer mit geringem Lern- und Nutzungsaufwand einsetzbar ist. Dabei wird durch die Verfügbarkeit beliebiger per WSDL (Web Service Description Language) beschriebener Funktionalitäten und die Erzeugung spezifikationskonformer BPEL Prozesse eine allgemeine Flexibilität und Applikationsunabhängigkeit erreicht. Die Grundausrichtung des Konzeptes basiert hierbei auf einer aus Nutzersicht bewerteten Aufwandsminimierung bei gleichzeitiger Realisierung einer ähnlichen Ausdrucksstärke, Mächtigkeit und Flexibilität, wie sie durch den Einsatz der für technische Anwender konzipierten BPEL Editoren möglich ist.

Inhaltsverzeichnis

1 Einleitung

Nicht nur durch den zunehmenden Grad des herrschenden globalen Wettbewerbs erfordern die Gegebenheiten des Marktes von Unternehmen eine zunehmend höhere Flexibilität [Wul'94], die auch von deren Informationssystemen erfüllt werden muss [Sti'07]. Da zukünftiger Änderungsbedarf sich einer Antizipierbarkeit entzieht [OH'97], ist die einst starre Abfolge von Design, Konzeption und Realisation von Software einer evolutionären Entwicklung während der gesamten Phase der Nutzung gewichen. Die notwendigen Anpassungen ergeben sich hierbei primär auf operativer Ebene und damit bei den informationstechnisch abgebildeten Geschäftsprozessen eines Unternehmens. Durch den gegenwärtigen technologischen Wandel hin zu serviceorientierten Architekturen im Allgemeinen und Web Services [DJM'05] im Speziellen ist eine derart geforderte Flexibilität technisch verfügbar. Diese lässt sich im unternehmerischen Alltag jedoch nur durch permanente und eigenverantwortliche Einbindung der Endbenutzer in den Anpassungsprozess verwirklichen. Die Kombination der Endbenutzerkompetenzen mit flexiblen Entwicklungstechnologien ist ökonomisch durch qualitativ höherwertige und eine kostengünstigere [WJ'04] Anpassung motiviert. Das steigende Qualitätsniveau resultiert auf der direkten Anpassbarkeit, welche somit das Problem der kommunikativen Diskrepanz zwischen Software-Designer und Endbenutzer [CFM'06] umgeht. Dies führt dazu, dass in hohem Umfang Fachkenntnisse einfließen und auch das vormals nur sehr schwer nutzbare unterbewusste Praxiswissen integriert werden kann. Zeitgleich ergibt sich hieraus eine effektivere und kostengünstigere Anpassung, da deren Durchführung nicht nur beschleunigt wird [LPW'06], sondern auch frei gewordene Kapazitäten des technischen Personals für andere Aufgaben herangezogen oder freigesetzt werden können. Im Gegenzug muss jedoch auch der Nachteil eines erhöhten Arbeitspotentials für den Endbenutzer beachtet werden. Die genannten Vorteile lassen sich jedoch nur durch Endbenutzer mit den erforderlichen Charakteristika der Domänenzugehörigkeit, technischem Interesse (vgl. *local developer*, [GN'92]) und einer natürlichen Experimentierfreudigkeit (vgl. *tinkerer*, [MCL'90]) effektiv realisieren. Zur besseren Abgrenzung mit dem in der Literatur nicht konsistent verwendeten Begriff des Endbenutzers wird die im Kontext dieser Arbeit adressierte Personengruppe in Kapitel 3.1.2 definiert.

Für die Entwicklung flexibler Informationssysteme durch Endbenutzer bieten serviceorientierte Architekturen eine aussichtsreiche technische Plattform [GMS'06], deren Flexibilität auf folgenden von Dostal et al. [DJM'05] kompakt definierten Kerncharakteristika beruht:

> *„Unter einer SOA versteht man eine Systemarchitektur, die vielfältig, verschiedene und eventuell inkompatible Methoden oder Applikationen als wieder verwendbare und offen zugreifbare Dienste repräsentiert und dadurch eine plattform- und sprachen–unabhängige Nutzung und Wiederverwendung ermöglicht.“*

Auf den Unternehmenskontext bezogen erlauben es beispielsweise Sprachen, wie die Business Process Execution Language (BPEL), plattform-, applikations- und unternehmensübergreifend Web-Service-Funktionalitäten zu (teil-)automatisierten Geschäftsprozessen zusammenzufügen, welches als Orchestrierung bezeichnet wird. Die Orchestrierung von Web-Service-Funktionalitäten in Form von BPEL Prozessen wird bislang fast ausschließlich durch Werkzeuge für technische

Anwendergruppen, wie beispielsweise dem *ActiveBPEL Designer* von Active Endpoints, unterstützt. Diese setzen für deren Nutzung ein tieferes Verständnis der beteiligten Standards voraus, welches somit die Anwendung durch Nicht-Techniker ausschließt. Bislang fehlt es an einem funktional mächtigen und zugleich beherrschbaren Konzept, welches diese technische Flexibilität einer SOA auch auf der speziellen Ebene der Anpassung durch Endbenutzer zugänglich macht.

Doch nicht erst seit Auftreten des Paradigmas serviceorientierter Architekturen besteht der Bedarf an Konzepten, welche Endbenutzern eine eigenständige Softwareanpassung und -entwicklung ermöglichen. In der Vergangenheit wurden diverse Flexibilisierungsansätze mit unterschiedlichen Schwerpunkten im Forschungsbereich des End-User Development (EUD) erforscht und entwickelt, welcher auch unter dem Begriff End-User Computing bekannt ist. Gemeinsames Ziel ist und war die Befähigung von Endbenutzern zur flexiblen Gestaltung von Softwareartefakten durch Anpassung, Veränderung oder funktionaler Erweiterung [LPK'06]. Beginnend mit *Buttons* [MCL'90] entwickelten sich vielfältigste Tailoring-Plattformen, deren Fokus auf einer umfassenden Anpassung zur Laufzeit bereits bestehender Software liegt. Einer der wichtigsten Vertreter ist in diesem Zusammenhang *FreEvolve*, welcher auf dem Softwarekomponentenmodell *FlexiBeans* beruht und eine kooperative Nutzung und Erstellung verteilter Prozesse innerhalb eines Groupware Systems erlaubt [WPW'08].

Darüber hinaus existieren Anwendungsszenarien, in denen Endbenutzer bereits erfolgreich eigenständig servicebasierte Funktionalitäten kombinieren können. Diese fokussieren primär Datenintegration oder Datenrepräsentation und produzieren so genannte Mashups, welche als browserbasierte Mini-Applikationen verstanden werden können, allerdings einen meist sehr begrenzten Grad an Funktionsmächtigkeit und Flexibilität besitzen. Im Bereich der Orchestrierung mit BPEL wird unter anderem angestrebt, eine ausreichende Komplexitätsreduktion durch Schaffung einer vereinfachten Endbenutzer-Programmiersprache zu erreichen [GMS'06]. Hingegen versucht das sich in der Entwicklung befindliche System *TailorBPEL* von Alda et al. [AKC'07] direkt auf BPEL aufzubauen und eine grafische Orchestrierung von Prozessen zu ermöglichen. Aktuell konzentrieren sich diese und andere Ansätze zur endbenutzergeeigneten Serviceorchestrierung jedoch lediglich auf begrenzte Teilaspekte und versäumen eine grundlegende Ausrichtung auf Endbenutzerbedürfnisse.

Obwohl serviceorientierte Architekturen eine aussichtsreiche technische Basis für EUD im Allgemeinen und Tailoring im Speziellen bieten, fehlt es aktuell an geeigneten Konzepten, welche eine annähernde Mächtigkeit und Flexibilität von BPEL erreichen und zeitgleich mit geringem Lernaufwand bedienbar sind. Aktuelle Servicekompositionswerkzeuge setzen häufig das zur Nutzung vorausgesetzte technische Wissen mit der möglichen Mächtigkeit des Kompositionsproduktes in einen direkten Zusammenhang. Es wird ein übergreifendes Konzept benötigt, welches die künstlichen Grenzen der beiden bisher isoliert wahrgenommenen Denkwelten aufbricht. Die vorliegende Arbeit bemüht sich um eine erste Überwindung dieser Grenzen, um Tailoring von serviceorientierten Architekturen auf Geschäftsprozessebene zu ermöglichen. Durch diesen Schritt wurde eine Möglichkeit geschaffen, konsequent das volle ökonomische Potential dieser neuen Technologie

abteilungsübergreifend im gesamten Unternehmen auszunutzen und aktuelle Entwicklungen im Rahmen dieser neuen Technologie diesbezüglich zu sensibilisieren.

Die Arbeit gliedert sich in die drei großen Teile von Analyse, Konzeption und Realisierung. An die Einleitung schließt das Grundlagenkapitel an, welches kompakt den aktuellen Stand der relevanten Technologie- und Forschungsbereiche beschreibt und durch eine kritische Analyse sowie anschließende Zielformulierung abgeschlossen wird. Diese wird in Kapitel 3 mit Rückgriff auf zusätzlich relevante Grundlagen weiter ausgearbeitet und konkretisiert. Auf dieser Basis wird in Kapitel 4 ein Grundkonzept entworfen, welches die Grundlage einer nutzerbeteiligten Vorstudie in Form eines Participatory Design Workshops darstellt und die notwendige Informationsbasis vervollständigt. Der verbleibende Inhalt des Kapitels befasst sich mit der vollständigen Konzipierung des grafischen Editors, welcher sich an einer aus Nutzersicht durchgeführten Gegenüberstellung von Aufwand und Nutzen orientiert. Die prototypische Umsetzung der grundlegenden Funktionalitäten wird in Kapitel 5 beschrieben, welches ebenso die Skizzierung der verwendeten Technologien und Rahmenwerke beinhaltet. Eine Zusammenfassung, die kritische Analyse der Ergebnisse sowie ein Ausblick möglicher Ansatzpunkte für die Fortführung dieses Projektes folgt in Kapitel 6, welches diese Arbeit abschließt.

2 Grundlagen

Der Inhalt dieses Kapitels vermittelt die grundlegenden Informationen der relevanten Themenbereiche, welche für das weitere Verständnis der Arbeit notwendig sind. Begonnen wird mit der Vorstellung des Paradigmas der serviceorientierten Architektur und deren technischer Realisation in Form der Web-Service-Technologie. Daran anschließen wird die Analyse einer Auswahl verschiedener SOA-Plattformen sowie ein Überblick unterschiedlichster Editoren zur Servicekomposition. Ein weiterer Schwerpunkt dieses Kapitels bildet die Forschungsarbeit auf dem Gebiet des End-User Development, wobei der Fokus auf dem Gebiet des Tailoring liegt. Im Anschluss an die Skizzierung der jeweiligen Ausgangssituationen erfolgt eine kritische Analyse bisheriger Projekte, die Endbenutzern eine Serviceorchestrierung ermöglichen sollen. Das Kapitel abschließen wird eine Analyse der Stärken, Schwächen und des Potentials bereit realisierter sowie zukünftig realisierbarer Verknüpfungsversuche dieser beiden Denkwelten, welches der Formulierung einer hieraus abgeleiteten Forschungsfrage dieser Arbeit resultiert.

2.1 Serviceorientierte Architektur

Die bisherige Entwicklung der Informationstechnologie fassen Sing und Huhns in drei Generationen zusammen: Monolitische, Client-Server und Peer-to-Peer Systeme. Diese Entwicklung in Richtung Verteilung und Entkopplung hat zu einem darauf aufbauenden Trend in der Vernetzung geführt. Folglich entwickelte sich ebenfalls der Bedarf an einer Architektur für eine systemübergreifende und offene Kommunikation [SH'05], welcher durch das Paradigma der serviceorientierten Architektur bedient wird. Im Gegensatz zu den vormals genutzten starren und hart codierten Architekturformen erlaubt sie eine flexible Verknüpfung verteilter Funktionalitäten zu Prozessen und Anwendungen. In Kombination mit den hierbei verfolgten Prinzipien der Technologie-Unabhängigkeit, der Kapselung von Geschäftsfunktionalität und der Entkopplung der Schnittstelle wird eine bisher nicht erreichte Flexibilität erreicht.

2.1.1 Kernkonzepte

Web Services stellen die gebräuchlichste Technologie zur Realisierung der isoliert oder kombiniert verwendbaren Funktionalitäten [JMS'06] dar. Ein Web Service wird in Anlehnung an Dostal [DJM'05] und Singh [SH'05] als eine über Netzwerk erreichbare Funktionalität definiert und bildet das funktionale Kernelement einer derart realisierten SOA. Technisch bauen die in diesem Kontext verwendeten Standards auf XML (Extensible Markup Language) auf. XML bietet einen Satz von Regeln, um neben Nutzdaten auch deren Datenstruktur in einem offenen, erweiterbaren, plattform- und programmiersprachenunabhängigen Format zu speichern, welches somit einen Datenaustausch zwischen Web Services unterschiedlichster Technologien ermöglicht. In Abbildung 1 werden die Funktion der im Folgenden erläuterten Standards kompakt skizziert und mit den Rollenkonzepten einer SOA - Service Vermittler, Service Anbieter und Service Konsument - in Beziehung gebracht. Entfernte Funktionsaufrufe (Remote Procedure Calls) und der Datenaustausch zwischen Konsument und Anbieter eines Web Services wird durch SOAP (anfänglich für Simple Object Access Protocol) [SOAP] ermöglicht, welches hierfür verschiedene Transportprotokolle, wie etwa

HTTP nutzen kann. Weiterhin elementar ist WSDL (Web Services Description Language) [WSDL], welche die konkreten Schnittstellenbeschreibungen und Anforderungen an die Nachrichtenstromformate enthält.

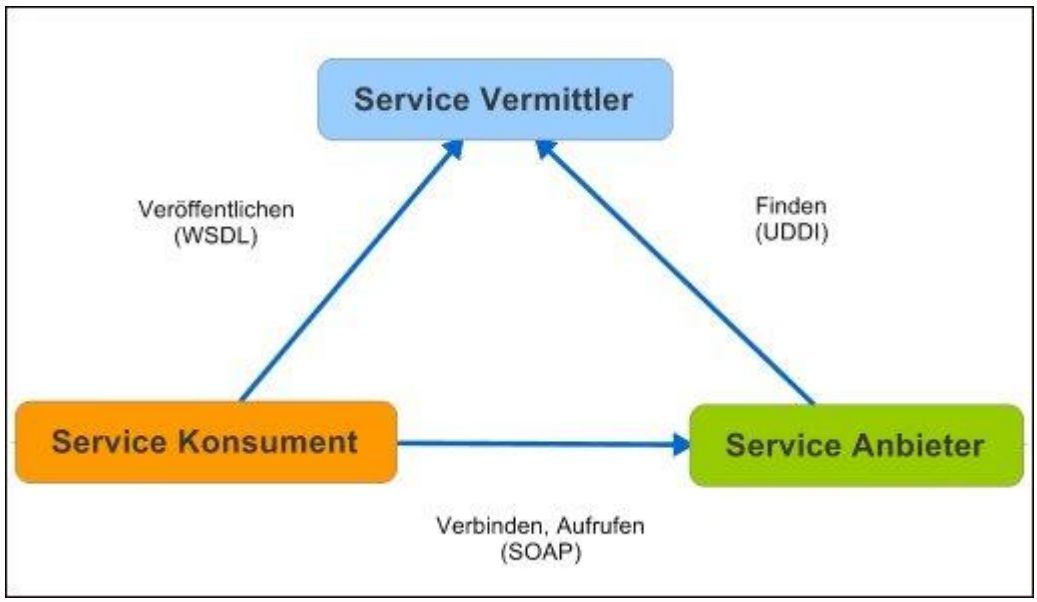

Abbildung 1: Das so genannte „SOA-Dreieck“

Eine WSDL Datei kann darüber hinaus optional zusätzliche Informationen enthalten, wie etwa den Aufbau des Web Service oder eine Schnittstellendokumentation. Als zentraler Ort für diese WSDL Dateien dienen häufig UDDI (*Universal Description, Discovery and Integration*) [UDDI] Server, die so ein gezieltes Auffinden erlauben, für die eigentliche Kommunikation aber nicht zwingend erforderlich sind. Der Aufbau von UDDI gliedert sich in drei Haupttabellen. Die White Pages enthalten Informationen über die Anbieterunternehmen, die Yellow Pages ermöglichen eine gruppierte Sicht auf die Anbieter je Geschäftsfeld, und die Green Pages stellen eine Sicht auf die Beschreibungen der registrierten Dienste zur Verfügung. In einer maschinenlesbaren Form wird diese Tabelle *Service Type Registration* genannt.

Komposition

Neben einer rein isolierten Nutzung einzelner Operationen eines Web Services, welche in der WSDL als *PortType* definiert sind, können diese ebenfalls durch Verknüpfung zu automatisierten Prozessen verknüpft werden. Dabei ist eine solche als Komposition bezeichnete Verknüpfung wiederum als Web Service nutzbar und ermöglicht durch aufbauende Verschachtelung die Realisierung einer beliebig komplexen Struktur.

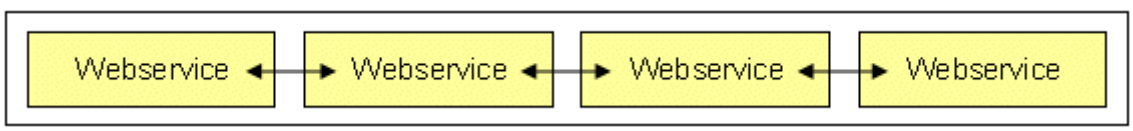

Abbildung 2: Zusammenarbeit in einer Choreographie

Neben dem Oberbegriff Komposition fallen in diesem Zusammenhang auch die Begrifflichkeiten der dezentralen Choreographie/Koordination und der zentralen Orchestrierung [RS'04]. Ergebnis der Choreographie ist eine Sammlung von Regeln, welche die „eigenständige“ Kommunikation der

Web Services untereinander festlegt [MS'07] und durch Abbildung 2 verdeutlicht wird. Die Orchestrierung (siehe Abbildung 3) steuert die Webserviceaufrufe hingegen zentral und bietet hierdurch den flexibleren Ansatz, da beispielsweise die Einbindung neuer Web Services oder fehlermeldungsabhängiger Alternativszenarien vereinfacht wird [JMS'06]. Aufgrund dieser Charakteristika und deren konzeptionell größeren Verwandtschaft zum grundlegenden Aspekt automatisierter Geschäftsprozesse werden sich die weiteren Inhalte dieser Arbeit auf diese Form der Komposition beziehen.

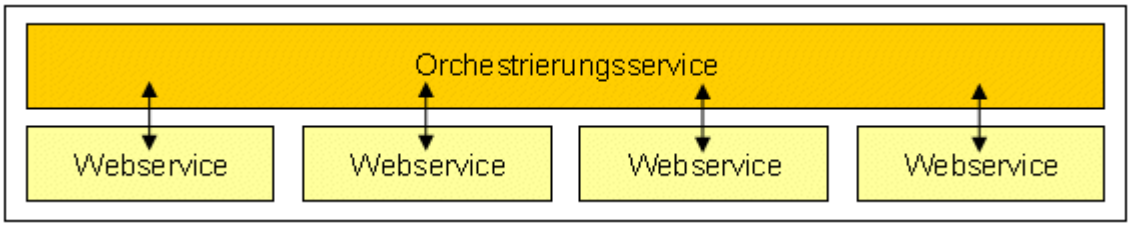

Abbildung 3: Steuerung durch Orchestrierungsservice

Eine solche Orchestrierung wird unter anderem durch Nutzung der frei verwendbaren Business Process Execution Language (BPEL) [BPEL] ermöglicht. Ursprünglich durch BEA, IBM und Microsoft entwickelt, hat sich BPEL mittlerweile zum einen durch OASIS (Organization for the Advancement of Structured Information Standards) zertifizierten und die Praxis dominierenden Standard entwickelt. Aufbauend auf WSDL nutzt BPEL die Standards XPath [XP], XMLSchema [XSc] und WS-Adressing [WSA] und bietet darüber hinaus unter anderem einen Mechanismus zur Fehlerbehandlung. BPEL unterstützt die Anwendung einer Vielzahl von Grundfunktionen klassischer Programmiersprachen, wie beispielsweise Schleifenkonstrukte, Verzweigungen oder Variablen. Durch den Fokus auf die Anwendung als Prozess-Definitionssprache adressiert BPEL ausschließlich technische Anwender, ist jedoch weit weniger komplex als gängige Programmiersprachen. Darüber hinaus können in den Prozess sowohl synchrone als auch asynchrone Aufrufe eingebunden werden. Nähere Details der vorgestellten Standards werden durch Margolis [MS'07] kompakt vermittelt.

Der kombinierte Einsatz dieser Standards ergibt Vorteile, die über Wiederverwendung und Informationskapselung der bisher als flexibel geltenden komponentenbasierten Systeme hinausgehen. Die Kommunikation zwischen Web Services, unabhängig von Plattform, Programmiersprache und Clientsoftware, ermöglicht eine bisher nicht erreichte technologische Flexibilität, welche unter anderem eine einfache unternehmensübergreifende Verknüpfung von Systemfunktionalitäten erlaubt. Diese, als lose Kopplung bezeichnete Unabhängigkeit der einzelnen Web Services bietet zusätzlich die Möglichkeit, Web Services während der Laufzeit auszutauschen. Darüber hinaus ist die in diesem Kontext stark fokussierte inkrementelle Softwareentwicklung eine mögliche Lösung für Brooks [Bro'87] Problembeschreibung der essentiellen Komplexität (essential complexity), welche während der Erstellung konzeptueller Konstrukte vorherrscht. Laut Brooks ist diese, im Gegensatz zur repräsentationsabhängigen nebensächlichen Komplexität (accidental complexity), durch inkrementelle Softwareentwicklung maßgeblich zu reduzieren.

Komplementäre und substitutive Standards

Neben den beschriebenen Kernstandards gibt es eine Vielzahl komplementärer und substitutiver Standards [HL'04]. Im Folgenden wird eine Auswahl der wichtigsten Vertreter grundlegend skizziert. Im Hinblick auf BPEL gab es lange Zeit keine Möglichkeit, die Kommunikation mit Menschen in einen Prozess zu integrieren. Nach einem grundlegendem Whitepaper in 2005 [KKL'05], welches die Anforderungen für die Eingliederung menschlicher Interaktion in BPEL Prozesse analysierte, erfolgte Mitte 2007 die Veröffentlichung von WS-BPEL4People [B4P] und WS-HumanTask [WSHT]. Die von Active Endpoints, Adobe, BEA, IBM, Oracle und SAP veröffentlichten Spezifikationen erlauben es, dass Services als *Task*, d.h. als menschliche Aufgabe „implementiert“ werden können. Durch den Aufruf eines dieser Services innerhalb eines Prozesses wird die Erzeugung einer solchen *Task* angestoßen, welche dann einer vorher definierten Person per Gruppenkonstrukt, logischem Ausdruck oder direkt zugewiesen wird. Den Mangel einer fehlenden grafischen Repräsentation von BPEL kann durch die Nutzung der Spezifikationssprache Business Process Modeling Notation [BPMN] begegnet werden, welche Symbole zur Modellierung von Geschäftsprozessen zur Verfügung stellt. Die von White [Whi'04] beschriebenen Elemente eines solchen Prozesses können hierbei als Basis eines maschinenlesbaren und damit ausführbaren BPEL Prozess dienen, wobei diese Abbildungs-Beziehungen in der BPMN Spezifikation definiert sind [Whi'05]. Wie in den folgenden Kapiteln zusätzlich ersichtlich wird, stellt BPMN daher eine häufig verwendete Notation für eine graphische Erstellung von BPEL Prozessen dar. Eine Alternative zu dem zuvor beschriebenen SOAP-Protokoll ist beispielsweise das weniger komplexe XML-RPC (Extensible Markup Language Remote Procedure Call), sowie der hier nicht intensiver betrachtete REST (Representational State Transfer) Architekturstil [DJM'05]. Der Einsatzbereich des client-server-basierten REST und dessen auf sechs Befehle begrenzten Vokabulars ist durch dessen minimalistischen Ansatz primär bei einfachen Anwendungen zu finden [Min'05]. Darüber hinaus existieren neben BPEL alternative Orchestrierungssprachen, welche aktuell jedoch lediglich einen geringen Verbreitungsgrad besitzen. BPML (Business Process Markup Language) baut beispielsweise auf den gleichen Grundlagen wie BPEL auf, erlaubt jedoch durch eine erweiterte Syntax komplexere Geschäftsprozesse und ist somit laut Juric et al. [JMS'06] als Obermenge von BPEL zu verstehen. Eine vergleichbare Funktionalität wäre alternativ auch durch die Erweiterung von BPEL mittels BPXL (Business Process eXtension Layers) erreichbar. Auf Grundlage der Ähnlichkeiten halten Juric et al. eine zukünftige Annäherung der beiden Sprachen für möglich. Näheres über verwandte, aber weniger relevante Standards wie WSCI (Web Service Choreopraphy Interface), WS-CDL (Web Services Choreography Description Language) und ebXML BPSS (Electronic Business XML, Business Process Specification Schema) werden ebenso bei Juric et al. beschrieben [JMS'06]. Im Zuge der Vollständigkeit muss erwähnt werden, dass eine SOA grundsätzlich als technologie-unabhängiges Architekturmodell verstanden werden muss und auch mit anderen Technologien, wie z.B. CORBA (Common Object Request Broker Architecture) [HL'04] realisiert werden kann. Jedoch stellt die hier beschriebene Umsetzung in Form von Web Services die weitaus gängigste Nutzung dar, obwohl sie, beispielsweise gegenüber Architekturen auf Basis proprietärer Technolgien, Performanznachteile besitzt [MS'07].

2.1.2 SOA-Plattformen

Die große Menge der Realisationen serviceorientierter Architekturen, welche mit Hilfe verschiedener Technologien und auf unterschiedliche Arten umgesetzt werden kann, zwingt zu einer stark fokussierten Betrachtung. Die im Folgenden betrachtete *Service Component Architecture* stellt durch breite Unterstützung führender Softwarehersteller (u.a. BEA, IBM, SAP und Sun) die relevanteste offene Plattform des Marktes dar, auch wenn diese sich bisher lediglich in einem frühen Entwicklungsstadium befindet. Enterprise SOA (ESOA) von SAP, als Vertreter einer proprietären Plattform, qualifiziert sich durch die grundlegende Unternehmensausrichtung und einer zumindest partiellen Endbenutzeradressierung in der Geschäftsprozessmodellierung. Darüber hinaus ist das Kernkonstrukt der ESOA, der SAP NetWeaver, vergleichbaren Technologien, wie Microsoft .NET oder IBM WebSphere laut Fuchs [Fuc'07] in diversen Punkten überlegen und stellt somit *„[…] die derzeit ausgereifteste Integrations- und Anwendungsplattform für Unternehmenssoftware […]“* dar. Ergänzt werden diese Analysen durch die Untersuchungen der aus der E-Government Bereich stammenden Umsetzung des Online Services Computer Interface (OSCI) sowie einer Forschungsarbeit zur Entwicklung einer End User Service Orchestration Plattform (EUSOP).

Service Component Architecturce

Open SOA stellt eine unternehmensübergreifende Initiative führender Softwarehersteller, wie beispielsweise IBM, BEA, SAP und Sun, dar. Unter der Prämisse einer Hersteller- und Technologieunabhängigkeit, verfolgt sie die Konzipierung eines Modells für eine einfache Nutzung der SOA im Kontext der Unternehmenssoftwareentwicklung. Die offene Spezifikationssammlung SCA (Service Component Architecture) beschreibt ein programmiersprachenunabhängiges Modell zur Erstellung von Applikationen, welche eine serviceorientierte Architektur nutzen [BBB'05]. Zum Zeitpunkt des Verfassens dieser Arbeit lagen die Spezifikationen in der Version 1.0 vor und unterliegen seither der Verantwortung von OASIS. Exemplarisch sei hier auf die Assembly Modell Spezifikation [SCA] verwiesen. Parallel wird von der Initiative die Entwicklung der zu SCA komplementären Service Data Objects vorangetrieben. Die Zielsetzung von SDO besteht darin, einen einheitlichen Zugriff auf heterogene Daten, welche in unterschiedlichen Standorten und Formaten gespeichert sind, zu ermöglichen. Die Version 2.1 von SDO [SDO], beispielsweise für Java und C++ wurde Ende 2006 fertiggestellt. Allgemein ist das wichtigste Prinzip der SCA eine einheitliche Abstraktion zu verschiedenen Service Typen, wie beispielsweise Web Services, Java und BPEL. Diese Trennung von Geschäftslogik und der Logik der Infrastruktur stellt den Kern des Konzeptes dar (vgl. Abbildung 4) und wird mit Hilfe dem SCA Komponentenmodell (*Component*) erreicht [MS'07]. Dabei kann eine Komponente durch verschiedene Sprachen realisiert werden, wie Java, PHP, C++, COBOL, XML (BPEL, XSLT), SQL oder XQuery. Ebenso ist die Verwendung verschiedener Programmierstile (asynchron, synchron, nachrichtenorientiert, call-and-return) und vieler Nachrichtenprotokolle (Web Services, Messaging Systems, CORBA IIOP) möglich. Wie mit einer bestimmten Technologie eine Komponente erstellt werden kann, wird dabei in Implementierungsspezifikationen beschrieben. Ein Containerkonstrukt (*Composites*) ermöglicht die Erstellung von regulären oder verschachtelten SCA-Applikationen, indem die anbietenden und konsumierenden Komponenten miteinander verknüpft werden. Unabhängig von der verwendeten Programmiersprache besteht eine Komponente aus einer gewissen Menge von elementaren Abstraktionen, welche

die Interaktion mit dieser genau spezifiziert. Konkret sind dies Services, Referenzen, Properties und Bindings, welche in der mit SCDL (Service Component Definition Language) verfassten SCA Konfigurationsdatei spezifiziert werden. Die von SCA bereitgestellten strukturellen Beziehungen zwischen Komponenten werden durch die Nutzung von BPEL mit einer Prozesslogik versehen [Edw'07].

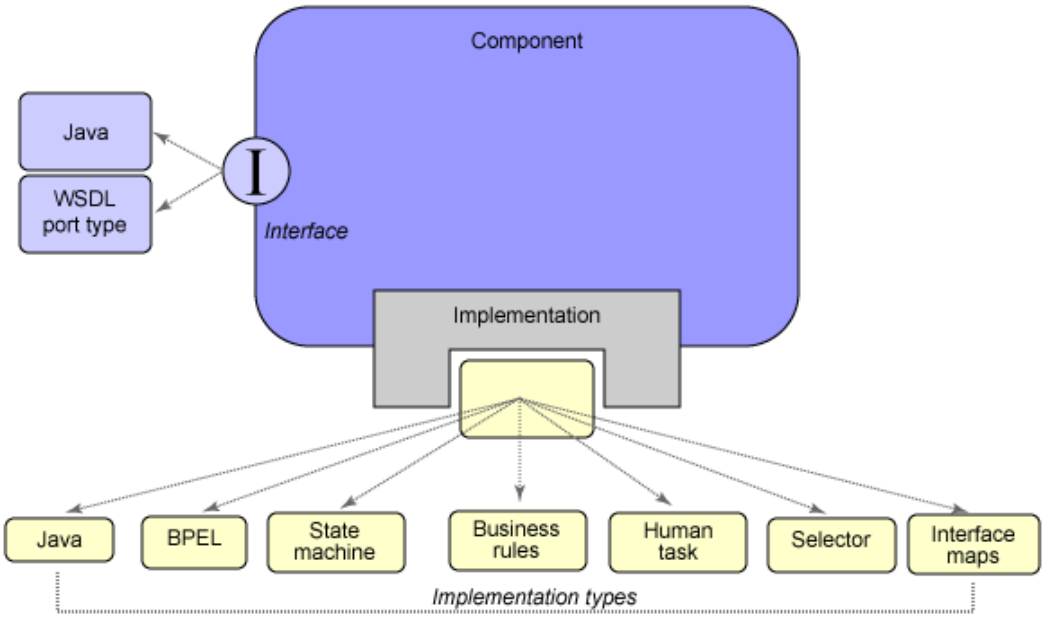

Abbildung 4: Komponente in SCA [BB'05]

Zusätzlich bietet SCA einige Erweiterungen für BPEL, welche verwendet werden können, um SCA-Funktionalitäten zu nutzen, die in BPEL nicht unterstützt werden. SCA und BPEL stellen somit komplementäre Standards für die verschachtelte Orchestrierung von Komponenten dar. Die frühen Konzepte softwaretechnischer Unterstützung basieren dementsprechend ebenfalls auf BPEL und BPMN und unterscheiden sich kaum von Realisationen der gängigen grafischen BPEL Editoren (siehe Kapitel 2.1.3). SCA befindet sich aktuell noch in einer frühen Phase der Entwicklung. Dennoch beinhalten bereits aktuelle Versionen der Spezifikationen relativ umfassendes Material über die Grundkonzepte, welches unter anderem eine Komposition von Web Services beschreibt. Aktiv unterstützt wird die Initiative von bedeutenden Vertretern des Softwaremarktes, wie IBM, Oracle, BEA, SAP und Sun. Durch die Betrachtung von Open SOA lässt sich eine mögliche zukünftige Richtung dieser Softwareprodukte erkennen, da trotz ausbleibender Unterstützung von Microsoft von einer größeren Relevanz für den gesamten Softwaresektor auszugehen ist. Zum aktuellen Zeitpunkt kann jedoch keine gesicherte Prognose über die zukünftigen Entwicklungen und den praktischen Stellenwert gemacht werden.

Enterprise SOA

Nach eigenen Aussagen stellt Enterprise SOA eine von SAP an die Anforderungen einer Verwendung in der Geschäftspraxis von Unternehmen angepasste Variante der serviceorientierten Architektur dar, welche sich an Anforderungen von Geschäftsprozessen orientiert. Technisch realisiert wird die Enterprise SOA mit Hilfe der Integrationsplattform *NetWeaver* [EKK'07], welche Funktionalitäten von SAP oder SAP-fremden Anbietern als Enterprise Services zur Verfügung stellt und zusätzlich als Plattform für die aufbauende Modellierungsumgebung *NetWeaver Composition*

Environment dient. Enterprise Services sind ebenfalls über WSDL nutzbar, weisen jedoch eine weniger feine Granularität als Web Services auf und sind mit einem konkreten Geschäftsobjekt verbunden, welches Repräsentant einer Entität oder eines Dokumentes sein kann. Diese sind wiederum einer Prozess-Komponente zugewiesen, welche zu logischen Gruppen zusammengeführt eine bestimmte Geschäftsfunktion, beispielsweise die einer Abteilung, informationstechnisch unterstützen. Das *NetWeaver Composition Environment* stellt eine integrierte Entwicklungs-, Modellier- und Laufzeitumgebung dar, in der diese Funktionalitäten und Daten in Form von *Composite Applications* zu Geschäftsprozessen kombiniert werden. Composite Applications können darüber hinaus kollaborative Elemente, individuell zusammengestellte Benutzeroberflächen und eine eigene Geschäftslogik enthalten [RS'07], wobei einzelne Prozessschritte interaktiv oder automatisiert ablaufen können. Zu Beginn wird ein Geschäftsobjektmodell spezifiziert sowie eine darauf aufbauende Geschäftslogik implementiert. Diese Datenschnittstellen werden im Anschluss mit Oberflächenkonstrukten verknüpft, welches bei weniger komplexen Benutzerschnittstellen mit Hilfe des endbenutzerkompatiblen *Visual Composers* möglich ist, der in Kapitel 2.1.3 näher beschrieben wird. Aufwendigeren Benutzerschnittstelle müssen jedoch mit Hilfe der Programmierumgebung *WebDynpro* [RS'07] erstellt werden. Hierauf baut im Anschluss die Prozessschicht auf, welche Services und Oberflächen in Form von *Guided Procedures* zu Abläufen verknüpft und für kollaborative Schritte Rollen spezifiziert [Sti'07]. Als Ausgabeformat dient das in die Portalanwendungen einfach integrierbare Flash Format. ESOP kennzeichnet ein großer Anteil proprietärer Formate der verschiedenen Schichten und der Fokus auf die Kompatibilität zu SAP Produkten, wie SAP Portal. Insgesamt wird hierdurch ein technologieübergreifender Austausch von orchestrierten Prozessen verhindert beziehungsweise erst durch aufwendige Konvertierungen ermöglicht, welcher auch zum Verlust einzelner Datenteile führen könnte.

Online Services Computer Interface

OSCI (Online Services Computer Interface) ist eine Sammlung bestehender Protokollstandards, welche sich gemäß dem Signaturgesetz primär auf rechtsverbindliche Transaktionen im Rahmen des E-Government konzentrieren. Den Kern bildet OSCI-Transport [OT], ein auf SOAP, XML-Signature [XSi] und XML-Encryption [XE] basierender Kommunikationsprotokollstandard zur sicheren Übertragung von Daten. Darauf aufbauend existieren diverse Datenformatstandards auf XML-Basis zum elektronischen Austausch fachbezogener Daten (z.B. XMeld, XJustiz, etc.) in der öffentlichen Verwaltung, welche anhand des XÖV Frameworks [OXÖV'06] entwickelt wurden und sich in Deutschland in diesem Bereich als de facto Standard durchgesetzt haben. Nachrichten im OSCI-Transport erfüllen die Anforderungen der rechtsverbindlichen Transaktion durch die Verwendung einer PKI (Public Key Infrastructure), wobei das Sicherheitsniveau frei anpassbar ist. Die einzelnen Mechanismen werden durch spezielle Webeservices (Cryptography/TransportCryptography, Directory, Persistence, Timestamp und Transport) bereitgestellt und bei Bedarf in den Geschäftsprozess eingebunden, welcher die Erzeugung der Nachrichten steuert. Das Protokoll unterstützt sowohl synchrone als auch asynchrone Nachrichten, welche mit einem nahezu beliebigen technischen Transportprotokoll übertragen werden können. Hierbei kommunizieren jedoch Kommunikationspartner nie direkt miteinander, sondern ausschließlich über die Rolle eines Intermediärs. Dieser erfüllt neben der Koordination der Kommunikation zusätzlich klassische Providerfunktionen wie Zertifikatsprüfung, Zeitstempel, Store & Forward, Regelüberwachung und Quittungen. Die Inhalts-

daten sind im Gegensatz zu den speziell für den Intermediär chiffrierten Nutzdaten Ende-zu-Ende verschlüsselt [OH'05]. Die Realisation unter LGPL (GNU Lesser General Public License), unter anderem in Form einer in Java verfassten [URL,a] und frei zugänglichen Bibliothek, ermöglicht die Komposition und Dekomposition valider Nachrichten gemäß der Spezifikation, inklusive einer syntaktischen Überprüfung. Alle benötigten Objekte der Nachricht mit Ausnahme der Inhaltsdaten werden durch die Bibliothek bereitgestellt, welche darüber hinaus die Möglichkeit zur Steuerung und Überwachung des Kommunikationsweges ermöglicht. Weitere Bestandteile, wie den technischen Versand, Visualisierung oder die Bereitstellung kryptographischer Funktionen sind nicht enthalten und müssen eigenständig realisiert und integriert werden [Med'04]. Die Komposition des Workflows in einer OSCI-kompatiblen Form erfolgt durch Techniker der sogenannten OSCI Leitstelle, und nicht durch direkt von am Workflow beteiligte Personen. Es liegt also eine klassische Modellierung von außen vor, was zum einen an der relativ hohen Langlebigkeit der behördlichen Workflows und zum anderen an der deutlich technologieorientierten Umsetzung der gesamten OSCI-Spezifikation liegen kann. Aus diesem Grund liegen keine näheren Informationen über den Orchestrierungsprozess und dessen softwaretechnische Unterstützung vor. Im Fokus der Spezifikation steht die Datensicherheit übermittelter Nutzdaten, welche laut einer Untersuchung des Bundesamtes für Sicherheit in der Informationstechnik [BSI'02] *„[…] angemessen hoch […]*“ ist. Die äußerst rudimentären Informationen über den Vorgang der Erstellung und Anpassung von Geschäftsprozessen machen eine Einschätzung der Interoperabilität schwierig. Orchestrierbare Web Services müssten durch spezielle Adapter angesprochen werden, um miteinander über den OSCI-Transport und damit über den Intermediär kommunizieren zu können, welches die Menge unmittelbar nutzbarer Web Services stark reduzieren würde. Hinzu kommt ein zusätzlicher Performanzverlust durch den Verschlüsselungsvorgang und die Rolle des Intermediärs.

End User Service Orchestration Plattform

Die technischen Anforderungen einer Serviceorchestrierung durch Endbenutzer leiten die Konzipierung und Realisierung der EUSOP Plattform [HLD'08]. Ausgangspunkt der Entwicklung bildeten die mangelnde Verfügbarkeit notwendiger Daten auf Ebene von UDDI und WSDL, welche für den Endbenutzer erforderlich sind, um Services zu suchen, zu verstehen und zu orchestrieren. Des Weiteren standen kollaborative und rollenspezifische Aspekte im Fokus der Entwickler. Unterteilt in eine Client- und Serverkomponente entstand die in Abbildung 5 skizzierte Plattform. Diese stellt den funktionalen Unterbau für die Modellierung servicebasierter Geschäftsprozesse bereit, bei der der UDDI spezifikationskonform um beliebige Metadaten erweitert werden kann sowie die Integration von Beschreibungen in die WSDL möglich ist. Kernelement bildet ein generischer Client, welcher *„[…] dynamisch die Elemente der WSDL-Datei eines Services interpretiert. Dazu erzeugt der generische Client innerhalb der Java Laufzeitumgebung Stellvertreterobjekte für die jeweiligen Web Services.“* [HLD'08]. Darüber hinaus ermöglicht das universelle Komponenten- und Containermodell, die so genannte *BPM Komponente*, eine sprachunabhängige Modellierung, integriert eine syntaktische Überprüfung verbundener Services und ermöglicht eine Erzeugung sowie Ausführung von BPEL Quellecode. In der zukünftigen Entwicklung wird eine technische Verbreiterung der Plattform durch die Anbindung von SCA (vgl. dieses Kapitel) und BPEL4People (vgl. Kapitel 2.1.1) angestrebt.

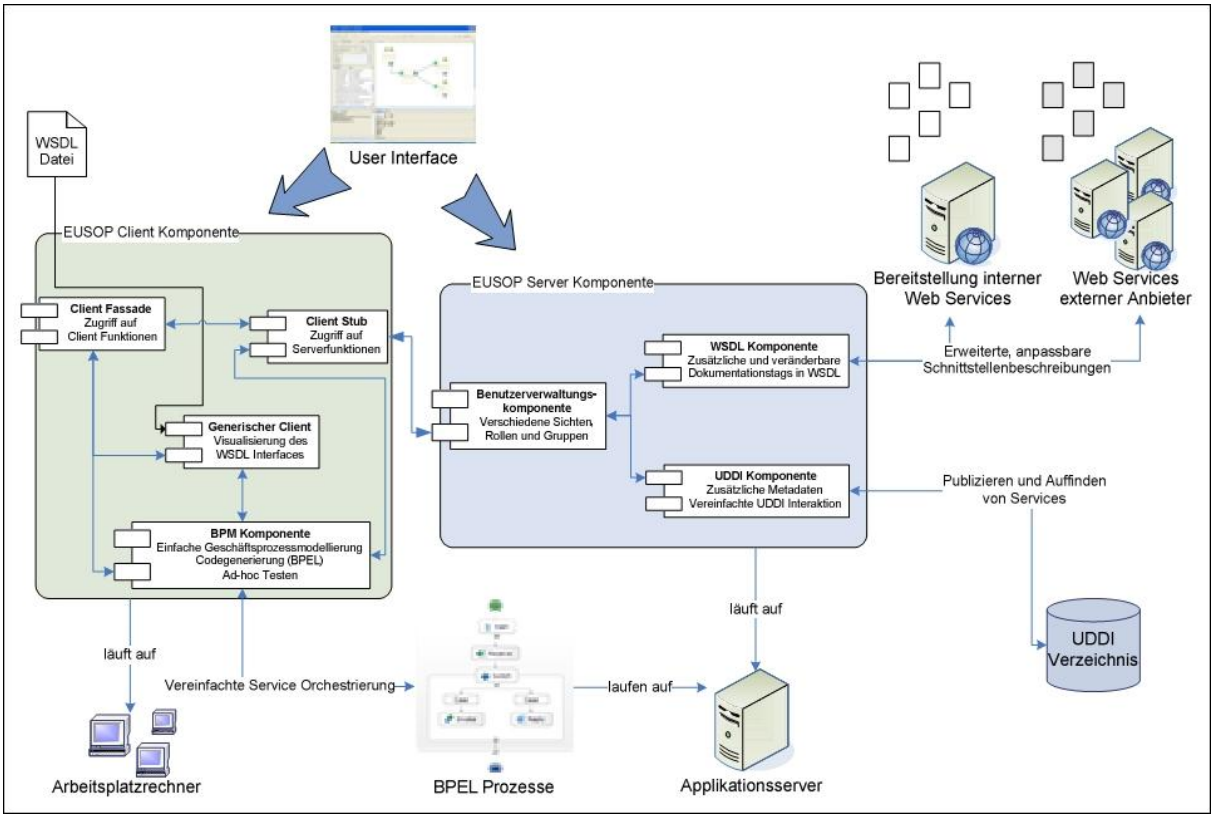

Abbildung 5: Aufbau der EUSOP Plattform [HLD'08]

2.1.3 Serviceorientierte Modellierungssoftware

Im Bereich der Editoren zur Service-Komposition existiert eine Vielzahl unterschiedlicher Schwerpunkte, Reifegrade und adressierter Anwendergruppen. Aufgrund dieser nahezu unüberschaubaren Menge an Realisationen kann die folgende Analyse nicht den Anspruch der Vollständigkeit erfüllen, enthält jedoch Vertreter der wichtigsten Kernkonzepte.

Grafische BPEL Editoren

Die Gruppe der BPEL Editoren umfasst eine große Auswahl kostenloser sowie kommerzieller Realisation zur grafischen Erstellung von Prozessen. Hierbei adressieren nahezu alle Konzepte ausschließlich technische Anwender, da zur Modellierung ein tieferes Verständnis der Webservicetechnologie und der Standards WSDL und BPEL vorausgesetzt wird. Im Regelfall wird ein Prozess durch die Auswahl und Kombination grafisch repräsentierter BPEL Befehle erstellt, wobei für die Verknüpfung zusätzlich umfangreiche Koordinationsparameter konfiguriert werden müssen. Ein großer Anteil dieser grafischen Editoren nutzt als Notation BPMN (vgl. 2.1.1) oder verwendet Varianten mit einer vergleichbaren Komplexität. Aus dieser Gruppe von BPEL Applikationen sind hier exemplarisch *BPEL Maestro* von Parasoft, der *Service Orchestrator* von OpenStorm und das der in Abbildung 6 gezeigte *BPEL Project* der Eclipse BPEL Teams genannt. Diese drei Editoren, welche auf der Eclipse Plattform (siehe Kapitel 5.1.1) aufbauen, können als repräsentativ für die breite Masse der Editoren mit einem eher rudimentären Funktionsumfang und Unterstützungspotential angesehen werden.

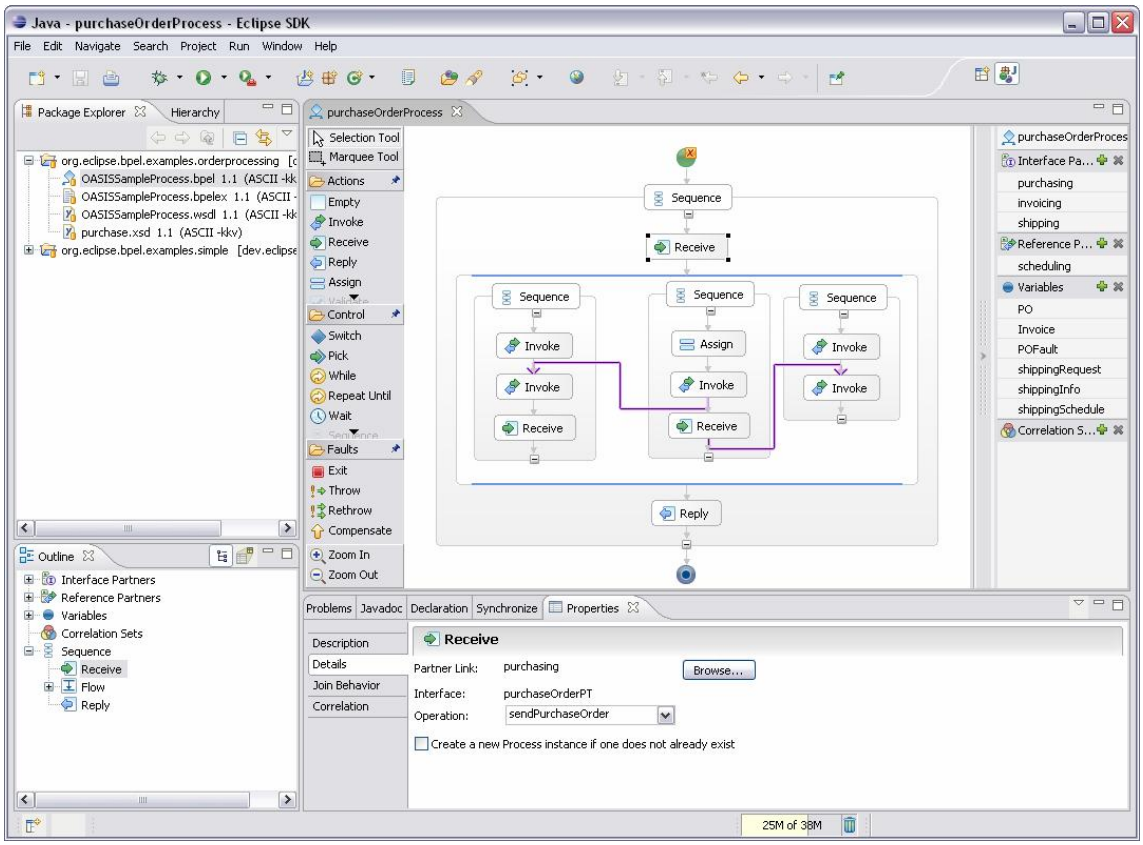

Abbildung 6: Eclipse BPEL Project

Aktuelle Business Process Management Systeme beinhalten zum Teil ebenfalls Komponenten mit einer begrenzten BPEL-Funktionalität. Während der in *jBoss* enthaltene *jBPM Designer* lediglich den Import eines BPEL-Prozesses ermöglicht, erlaubt der in Abbildung 7 gezeigte *BPMS Designer* von intalio eine visuelle Zuordnung einzelner Parameter.

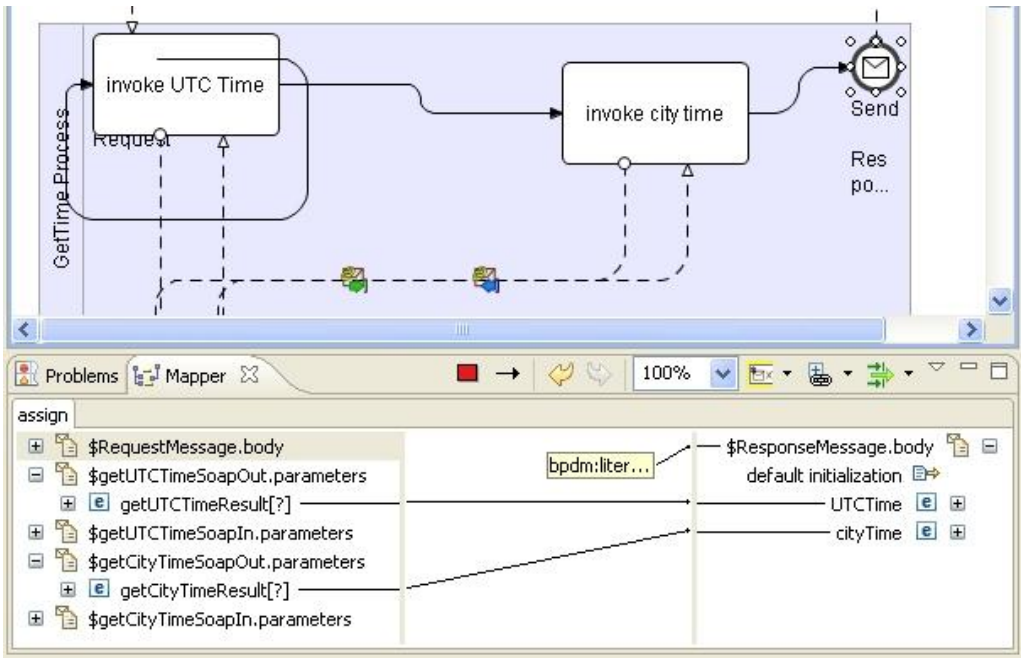

Abbildung 7: Prozess und Mapping des BPMN Designers

Der Funktionsumfang der verschiedenen Editoren ist meist auf eine übersichtlichere Darstellung der XML Dokumentinhalte und der Anzeige von Syntaxfehlern begrenzt, aber es sind auch simulierte Durchläufe, wie bei dem in Abbildung 8 gezeigten *ActiveBPEL Designer* von ActiveEndpoints möglich.

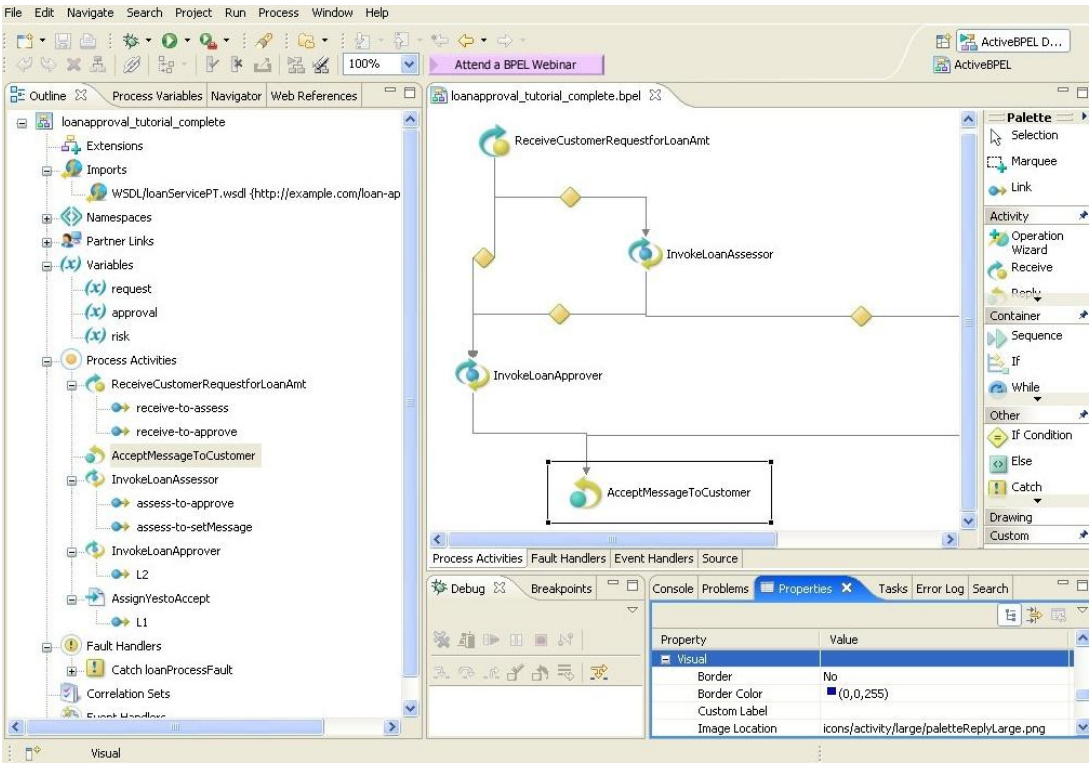

Abbildung 8: ActiveBPEL Designer

Darüber hinaus bietet dieser die Funktion, Teile eines BPEL Prozesses als sogenannte *BPELets* zu speichern und diese in anderen Prozessen als Modellierungsbaustein wieder zu verwenden. Gemein ist allen Realisationen, dass sie im Bestfall lediglich durch die Speicherung der Modelle in einem zentralen Repository, wie bei dem auf Microsoft Visio aufbauenden *Process Modeler 5* von itp-commerce, kollaborative Aspekte berücksichtigen. Die verschiedenen offenen und kommerziellen BPEL Editoren unterscheiden sich insgesamt primär im Bedienkomfort für die modellierenden Techniker, ohne eine maßgebliche Komplexitätsreduktion, Abstraktion oder Unterstützung zu bieten und somit für Endbenutzer nutzbar zu sein.

Mashup Editoren

Eine andere Art von service-kombinierender Anwendungen stellen Mashup Editoren dar. Diese meist browserbasierten Editoren erreichen durch Nutzung von AJAX (Asynchronous JavaScript and XML) oder anderer proprietärer Technologien annähernd den Bedienkomfort von Desktop Anwendungen. Das Paradebeispiel und einer der ersten Vertreter von Mashups stellt HousingMaps.com dar, welches auf Craigslist.com gelistete und zum Verkauf stehende Häuser anhand der Adressdaten mit dem Service von Google Maps parallel auf einer Landkarte anzeigt [Jhi'06]. Im Gegensatz zu der Prozessorientierung bei BPEL liegt der Fokus jedoch primär auf der gezielten Bereitstellung von Informationen, welche durch Kombination vorhandener, aber eigentlich getrennter Datenquellen und Services zu neuen Applikationen erfolgt. Aus diesem Grund können Mas-

hups primär als ”[…] *user-driven micro-integration of Web-accessible data* […]“ [Jac'07] verstanden werden. Gewöhnlich setzt die Erstellung derartiger Mashups einen gewissen Umfang an Erfahrung in der Webprogrammierung sowie anderer relevanter Webtechnologien voraus. Die oben beschriebenen Editoren erlauben es jedoch teilweise auch Anwendern, welche nicht über diese Kenntnisse verfügen, Mashups zu erstellen. Diese sind zwar vergleichsweise limitiert, erlauben jedoch die Herstellung situierter Applikationen, welche spontan erstellt, einen gerade auftretenden Bedarf bedienen oder ein Problem lösen [Jhi'06]. Konkret werden innerhalb dieser Editoren grafische Repräsentationen von Datenservices mit teilweise komplexen Oberflächenelementen, sogenannten Widgets kombiniert. Diese werden häufig in Form einer WYSIWYG (What You See Is What You Get) Darstellung angeordnet und mit Datenquellen mit meist webbasiertem Inhalt verknüpft. Die verknüpfbaren Komponenten basieren jedoch nicht auf allgemeinen Standards und können demnach nicht übergreifend zwischen verschiedenen Mashup-Editoren eingesetzt werden, welches die direkte Einbindung eines bisher nicht zur Verfügung stehenden Servicefunktionalitäten durch den Anwender erschwert oder sogar verhindert. Eine Gruppe dieser Editoren fokussiert den Integrationsaspekt von Daten aus verschiedensten Quellen und bietet nur bedingt einbindbare Oberflächenelemente. In dieser Gruppe erreicht das in Abbildung 9 dargestellte *Yahoo Pipes* wohl den größten Reifegrad, welches auch von Anwendern ohne technisches Hintergrundwissen verwendet werden kann. Vorgefertigte Funktionskomponenten werden zur Datenintegration, -manipulation und –ausgabe parametrisiert sowie kombiniert und ergeben die so genannte *Pipe*. Die Abbildung 9 zeigt die Modellierungsumgebung und die Repräsentation der einzelnen Komponenten sowie das am unteren Bildschirmrand positionierte Fenster zur Anzeige von Zwischen- und Endergebnissen.

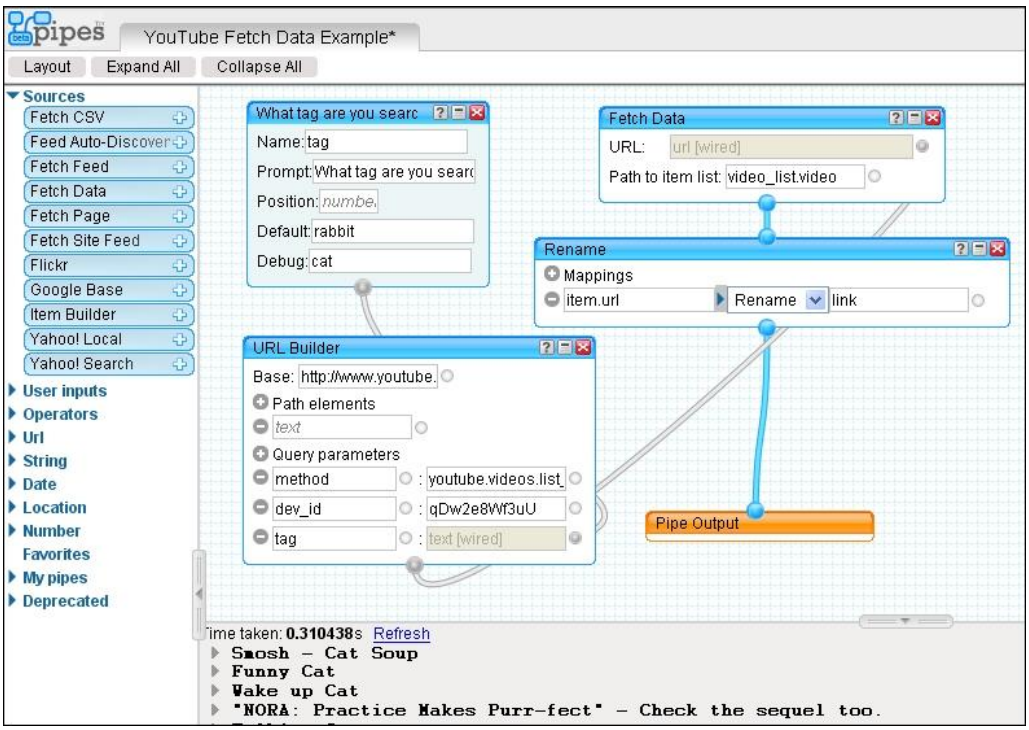

Abbildung 9: Yahoo Pipes Orchestrierung

Als Ausgabeformat unterstützt *Yahoo Pipes* beispielsweise RSS, JSON, Widgets oder die Darstellung auf interaktiven digitalen Landkarten, wie etwa *Google Maps*. Um die Flexibilität einzelner

Pipes zu erhöhen, können gesetzte Parameter alternativ als Variable deklariert werden, deren Werte vor Ausführungsbeginn mit Hilfe einer automatisch generierten Eingabeoberfläche anpassbar sind. Es ist voraussichtlich nur eine Frage der Zeit, bis die individuelle Einbindung von bereit gestellten Oberflächenelementen möglich ist. Darüber hinaus bietet Yahoo Pipes noch eine rudimentäre Berücksichtigung kollaborativer Aspekte, da erstellte Pipes veröffentlicht und damit anderen Nutzern zur Ausführung oder als Basis einer neu erstellten Pipe zur Verfügung stehen. Das von Microsoft entwickelte *Popfly* konzentriert sich hingegen auf optisch ansprechende Editor- und Mashup-Oberflächen und bietet im aktuellen Entwicklungsstand lediglich einen begrenzten Funktionsumfang. Kernelement ist dessen optisch herausstechendes 3D-Orchestrierkonzept, welches vermutlich besonders Anwender ohne technisches Hintergrundwissen ansprechen soll. Eine andere Gruppe von Editoren konzentriert sich auf Anforderungen von Mashups, welche im Kontext von Unternehmen eingesetzt werden können. Häufig unter dem Begriff Enterprise Mashups geführt, greifen diese neben externen auch auf interne Datenquellen, wie beispielsweise die Datenbank eines ERP-Systems (Enterprise Resource Planning) zu. Motivation ist es dabei, betriebswirtschaftlich relevante Daten in der gewünschten Form schnell zugreifbar zu machen. In diesem Bereich existieren ebenfalls Konzepte, welche die Datenintegration fokussieren, wie etwa *Apatar* (siehe Abbildung 10) oder *JackBe Wires* der JackBe Corporation, während Applikationen wie der von SAP entwickelte *Visual Composer* eine umfangreiche Oberflächengestaltung erlaubt und zusätzlich auch kollaborative Funktionalitäten bietet. Editoren für hauptsächlich nicht technische Anwender zielen dabei häufig auf die Einbindung der erstellten Mashups in bestimmte Portalanwendungen und damit in individuelle Portalseiten, wie es auch beispielsweise bei dem Entwickler adressierenden Editor *Data Mashups* von Applibase der Fall ist.

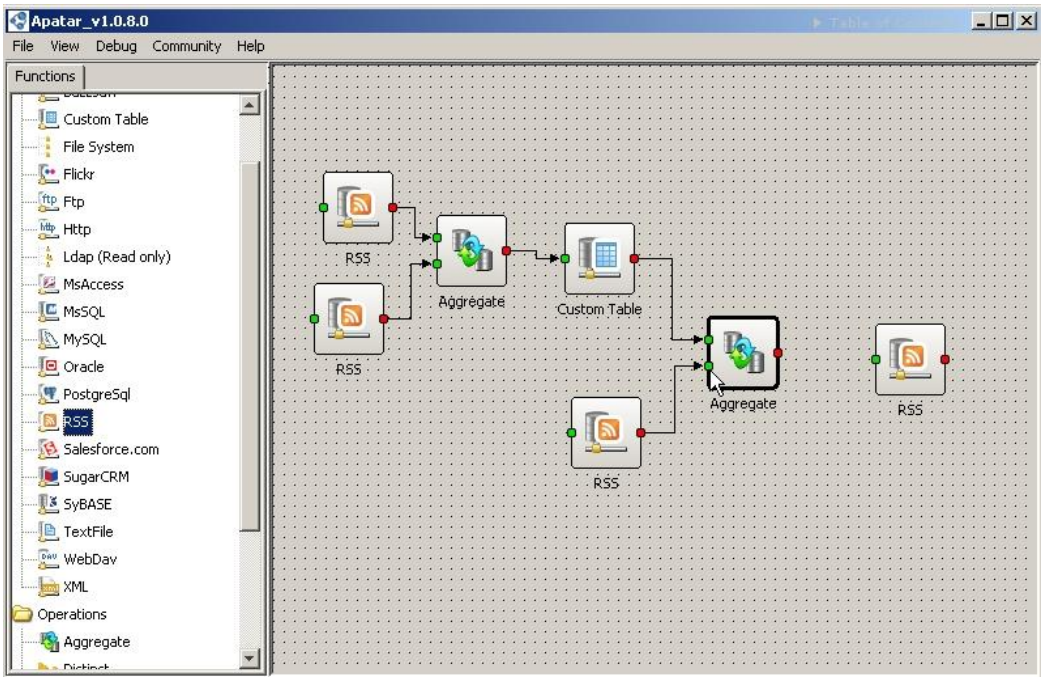

Abbildung 10: Orchestrierung mit Apatar

Zu dieser Gruppe gehört auch der in Abbildung 11 dargestellt und bereits erwähnte *Visual Composer* von SAP, welcher, aufbauend auf *SAP NetWeaver*, die Verknüpfung von Datenquellen, Services und Oberflächenelemente anhand einer Form der weit verbreiteten *Box and Wire* Metapher

ermöglicht. Zur Modellierung wird die proprietäre Modelsprache GML (GUI Modeling Language) verwendet, welche im Normalfall im Anschluss in Flash umgewandelt wird, um gemäß der Rollenspezifikation dort für die berechtigten Nutzer direkt im Portal zugreifbar zu sein. Während der Modellierung besteht die Möglichkeit, erstellte Mashups zu testen und das Layout der Oberflächenelemente in Form von WYSIWYG zu bearbeiten.

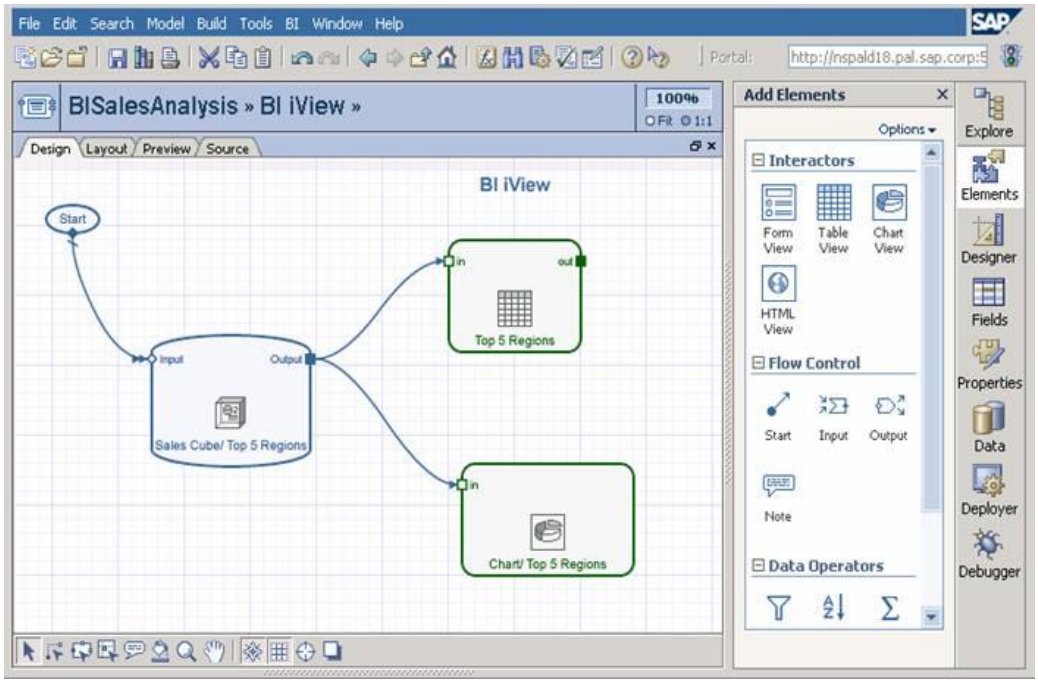

Abbildung 11: SAP Visual Composer

Ebenfalls auf den unternehmerischen Kontext ausgerichtet ist das von IBM entwickelte *QEDWiki* (Quick and Easily Done), welches im aktuellen Entwicklungsstand Endbenutzern die Erstellung von Mashups durch die Kombination von Widgets erlaubt. Diese stellen eigenständige Funktionsblöcke auf Basis von REST-Web-Services dar und beinhalten teilweise entsprechende Oberflächenelemente. Die Modellierung in einer ausschließlich WYSIWYG-basierten Form und die dem Wiki-Prinzip folgende Berücksichtigung kollaborativer Aspekte erleichtert die Handhabung durch nicht technische Anwender. Jedoch gestalten sich die Integration von REST-Web-Services und die Datenverknüpfungen der verschiedenen Widgets noch recht schwierig. Herausstechend ist die teilweise Automatisierung von Datenverknüpfungen durch das Erkennen bestimmter Datenmuster, welches in einem demonstrierten Beispiel [URL,b] anhand der automatischen Erkennung von Mobilfunknummern und der selbständigen Verknüpfung mit einem Service zum Versenden von SMS-Nachrichten gezeigt wurde.

2.2 End User Development

In der Geschichte der Softwareentwicklung nahmen zu Beginn der Mensch und dessen Bedürfnisse nur eine sehr untergeordnete Rolle ein. Dieser Faktor gewann jedoch zunehmend an Bedeutung, so dass aktuelle Entwicklungsprinzipien dem Menschen die zentrale Rolle zuweisen. Eine auf dessen Anforderungen optimal angepasste Software muss jedoch die jeweilige Arbeitpraxis und das konkrete Individuum berücksichtigen. Entwicklern können aufgrund eines stark vereinfachten Nutzermodells und des rudimentärem Wissens über die Domäne daher nur suboptimale Er-

gebnisse erzielen. Wie es in der Einleitung (vgl. Kapitel 1) dargelegt wurde, verfügt nur der Endbenutzer selber über das notwendige Wissen einer optimalen informationstechnischen Unterstützung der eigenen Arbeit. Dieser besitzt jedoch im Regelfall keine ausreichenden Programmierfähigkeiten zur Realisation derselbigen. Daher wurden und werden Konzepte erforscht, welche Endbenutzer eine eigenständigen Entwicklung oder umfassende Anpassung ermöglichen sollen. Dieses Kapitel skizziert die grundsätzliche Motivation sowie der bisherigen Entwicklungen, wobei der Fokus auf dem Bereich des Tailoring liegt, welches die Anpassung während der Nutzung von Software adressiert.

2.2.1 Motivation und Historie

Fischer und Girgensohn [FG'90] klassifizierten 1990 zwei grundlegende Gruppen von Computer Systemen. Systeme der ersten Gruppe können äußerst flexibel auf nahezu jedes Problem angewendet werden, erfordern jedoch Programmierfähigkeiten der Nutzer, wie etwa bei der Verwendung von *Unix Shell*. Die zweite Gruppe enthält einfach zu bedienende Systeme, welche jedoch lediglich einen relativ starren Funktionsumfang von Möglichkeiten zur Problembewältigung bieten [FG'90], wie es beispielsweise bei der Textverarbeitung von *Open Office* der Fall ist. Softwareentwickler konzipieren und implementieren diese spezialisierten Systeme für Anwender ohne Programmierkenntnisse mit dem Ziel, die konkrete Arbeitspraxis möglichst optimal zu unterstützen, welches in der Realität meist nur unzureichend realisiert werden kann. Grund hierfür ist fehlendes Wissen der Entwickler über die Anwendungsdomäne und den damit verbundenen Anforderungen der zu unterstützenden Tätigkeit. Dabei ist ein umfassender Wissensaustausch oder ein gemeinsam erstelltes Anforderungsprofil durch die herrschende kommunikative Diskrepanz zwischen Designer und Endbenutzer [CFM'06] sowie der schwierigen Verbalisierung unterbewussten Wissens nicht ohne Weiteres möglich. Zur Lösung dieses Problems wurden Konzepte entwickelt, welche Techniken zur Überbrückung kommunikationserschwerender Faktoren und eine stärkere Endbenutzerbeteiligung beinhalten, deren Durchführung jedoch äußerst zeitintensiv sind. Die Methode des Participatory Design (PD) (vgl. Kapitel 3.2.1) stellt einen der bedeutendsten Vertreter dar, ist jedoch in der Anwendung auf die Designphase begrenzt und kann somit nicht die sich während der Softwarenutzung verändernden Anforderungen adressieren. Diese Veränderungen können unter anderem durch die Einführung der Applikation [OH'97], durch die sich permanent wandelnden Anforderungen der globalen Märkte [Wul'94] oder durch nutzerindividuelle Präferenzen entstehen. Dieses Problem adressiert das Vorgehensmodell Seeding, Evolutionary Growth and Reseeding (SER) [Fis'96; FGM'01], welches Systeme als offenen Entwicklungsumgebung realisiert und deren nutzungsbegleitende Evolution aufgrund einer gemeinsamen Wissens- und Anforderungsbasis von Designer und Anwender gesteuert wird [FGY'04]. Dieses allgemein durch Fischer als Meta-Design [FG'06] bezeichnete Prinzip zur nutzerbeteiligten Entwicklung dieser Umgebungen, kann jedoch auch prinzipiell durch andere evolutionäre Vorgehensmodelle, wie beispielsweise STEPS (Softwaretechnik für evolutionäre und partizipative Systemgestaltung) [FRS'89], erfolgen.

Unter zeitlichen und finanziellen Aspekten [WJ'04] kann eine effektive permanente evolutionäre Entwicklung von Software trotz potentieller Risiken der Informationssicherheit [SB'06] nur durch die

aktive und direkte Einbindung der Endbenutzer erfolgen. Verschiedene, aktuell unter der Bezeichnung End-User Development zusammengefasste, Forschungsbereiche verfolgen aus den genannten Gründen das Ziel, entsprechende Umgebungen zur eigenständige Anpassung oder Entwicklung von Software durch Endbenutzer zu realisieren. Im Kontext der weiteren Ausführungen wird eine vereinfachte Definition des Endbenutzers verwendet. Sie wird als Person verstanden, deren Hauptaufgabe nicht die Entwicklung, Veränderung oder Administration von Applikationen ist, sondern diese „nur“ zur Lösung von Problemen der Arbeitsdomäne nutzt und dementsprechend im Regelfall keine Programmierfähigkeiten besitzt. An dieser Stelle ist zu beachten, dass diese derart eingegrenzte Gruppe jedoch eine sehr große Diversität aufweist, wobei sich Nutzer bereits durch Alter, Kultur, Sprache, IT- und Domänenkenntnissen unterscheiden [SB'06], so dass in der folgenden Übersicht immer von einem stark vereinfachten Nutzerprofil ausgegangen wird, welches in der Realität in dieser Form nicht existiert. Dieses vereinfachte Bild war jedoch auch eine der Grundlagen zu Beginn des bislang 30jährigen Forschungsverlaufes, welcher durch zunehmende technische Möglichkeiten und relevanter Forschungserkenntnisse anderer Gebiete beeinflusst wurde und durch wechselnde Schwerpunkte und Begrifflichkeiten gekennzeichnet war. Ausgehend von dem Forschungsbereich des Human-Computer Interaction (HCI) wurden in den 70iger Jahren erste Arbeiten dieses Themenkomplexes veröffentlicht, welcher jedoch erst später, unter anderem mit der Arbeit von McLean [Mcl'79], breitere Beachtung fand und in Folge unter dem Begriff des End-User Computing geführt wurde. Die EUC-Forschung konzentrierte sich bis Anfang der 90er Jahre hauptsächlich auf die selektive Untersuchung und das Verständnis der grundlegenden Komponenten und Faktoren, in denen hauptsächlich eine informationstechnisch-orientierte Sicht eingenommen wurde [BB'93] und somit auch ein zentrales Management der Nutzaktivitäten eine wichtige Rolle spielte. Dennoch gab es bereits seit Ende der 70er Jahren verschiedene Methoden und Techniken zur Anpassung und Entwicklung von Systemen durch Endbenutzer, wie beispielsweise Programming by Demonstration bzw. Programming by Example, welche unter dem Begriff des End-User Programmings geführt wurden. Seit Anfang dieses Jahrtausends werden die Anstrebungen allgemein unter dem Begriff des End-User Development zusammengefasst. Der Grund für die verschiedenartigen Konzepte liegt in der Komplexität des gesetzten Zieles, welches von einer großen Anzahl bekannter und unbekannter Faktoren abhängt. Erst eine interdisziplinäre Zusammenarbeit von auf den ersten Blick unabhängigen Forschungsbereichen ermöglicht eine Annäherung an das gesetzte Ziel. Beispielsweise erforschen Teilbereiche der Psychologie durch Analyse grundlegender Lern-, Verständnis- und Problemlösungsprozesse eine optimale Zugänglichkeit und Nutzbarkeit [Bla'06a]. Aus dem Forschungsbereich der Groupware werden Unterstützungskonzepte der natürlichen kollaborativen Problemlösungspraxis [GN'92] während der Anwendung und Entwicklung adaptiert, um diesen zentralen Aspekt der Arbeitspraxis im System zu berücksichtigen. Künstliche Intelligenz soll helfen, eine an die Nutzerfähigkeit, Problemstellungsanforderung und Situation angepasste Oberfläche und Funktionalität bereitzustellen [RI'06]. Diese und andere Erkenntnisse sind Grundlagen der im Folgenden vorgestellten und im Kontext dieser Arbeit wichtigten Bereiche des EUD. Von besonderer Bedeutung ist das übergreifend relevante Konzept des *gentle slope of complexity* von MacLean et al. [MCL'90]. Dieses zeichnet sich durch die Abwesenheit von Komplexitätssprüngen aus, so dass ein relativ geringer Lernvorschuss zu einer nahezu direkten Erweiterung der eigenen Anpassungsfähigkeiten und der damit erreichten Mächtigkeit führt. Übergreifend wird ebenfalls zur Reduzierung des Lernaufwandes vorhandenes Nutzerwissen

eingesetzt, so dass beispielsweise domänenrelevante Begrifflichkeiten oder Metaphern [WPW'08] Verwendung finden.

2.2.2 End User Programming

Der Bereich des End-User Programming (EUP) zielt auf die Erschaffung eines für den Endbenutzer einfacher erlernbaren textuellen oder grafischen Programmierparadigmas und nutzt hierfür verschiedene Konzepte, von denen eines Programming by Demonstration (PBD) ist, welches ebenso unter dem Begriff Programming by Example (PBE) bekannt ist. Im Vergleich zu der häufig in Büroanwendungen zu findenden Makroprogrammierung ist es jedoch nicht auf eine reine Wiederholung von Nutzeraktionen und Eingaben begrenzt, sondern kann von aufgezeigten Beispielen abstrahieren. Beispielsweise nutzt das Autorenwerkzeug *AgentSheets* und die darauf aufbauende regelbasierte Sprache *Visual AgenTalk* teilweise diese Technik zur visuellen Erstellung von Simulationen durch programmierbare Agenten [RIZ'00]. Eine Übersicht weiterer Systeme, welche eine Programmierung durch Demonstration erlauben, bietet das Buch *Watch What I Do: Programming by Demonstration* von Cypher [Cyp'93]. Das Konzept der *Natürlichen Programmierung* hingegen versucht allgemeine menschliche Handlungsweisen während Programmierhandlungen zu identifizieren, um ein derart ausgerichtetes Programmierparadigma schaffen zu können, welches auch Einfluss auf die allgemeine Form der Programmierung hat. Grundlage hierfür bilden Untersuchungen mit Programmierern und Nicht-Programmierern, wie sie beispielsweise von Myers und Pane durchgeführt wurden [PM'06]. Diese empirische Untersuchung stellte Nicht-Programmierer vor die Aufgabe, mit Hilfe von selbst gewählten Formalismen ein Spiel zu „Programmieren“, welches unter anderem eine starke Orientierung an ereignisbasierten Beschreibungen zum Ergebnis hatte. Eine der ersten Realisationen dieser Art bildete die Entwicklungsumgebung HANDS (Human-centered Advances for Novice Development of Software) [PMM'02], welche Kindern ohne Programmierkenntnisse die Entwicklung von einfachen Programmen ermöglicht [MPK'04].

2.2.3 Tailoring

Durch die Zielsetzung in BPEL formulierte Geschäftsprozesse ad hoc zu Erstellen oder zu Verändern, ist für diese Arbeit besonders der von Henderson und Kyng [HK'91] geprägte Begriff des Tailoring relevant, da dieser alle Aktivitäten beschreibt, um eine Applikation im Kontext ihrer Nutzung zu verändern. Erstmals erlaubte der von MacLean et al. [MCL'90] entwickelte Prototyp *Buttons* eine solche umfangreiche Anpassbarkeit eines bereits bestehenden Informationssystems durch Endbenutzer, wobei einzelne Aspekte des Tailorings bereits früher (u.a. [Sta'81]) realisiert wurden. Das durch die Nutzer durchgeführte Tailoring, welches implizit nach dem Designprozess stattfindet, modifiziert oder erweitert das System und generiert somit neue Funktionalität. Dabei lässt sich ein Zusammenhang zwischen Tailoring und der Organisationsstruktur identifizieren, welcher von Pipek [Pi'05] wie folgt dargestellt wird:

> *"Tailoring activities take place as part of technology appropriation processes that represent the technological level of ongoing organisational change."*

Durch die Erschaffung neuer Funktionalitäten lässt sich Tailoring damit klar gegenüber Anpassbarkeit und Adaptivität abgrenzen, welche als Auswahl zwischen bereits bestehendem Programmverhalten (z.B. Parametrisierung) beziehungsweise automatische Systemanpassung aufgrund des Nutzerverhaltens verstanden werden. Aufgrund der höchst heterogenen Endbenutzerfähigkeiten sollte Tailoring auf verschiedenen Komplexitätsstufen [CFL'03] ermöglicht werden, deren einfachste Form die Modifizierung eines vorhandenen Tailoring-Dokumentes sein kann. Darüber hinaus empfiehlt sich eine Berücksichtigung und Unterstützung kollaborativer Aspekte, da dieser elementare Bestandteil der Arbeitspraxis ebenfalls während des Tailoring präsent ist [Pi'05] und sich beispielsweise durch den Austausch [Mac'90] oder der gemeinsamen Bearbeitung [NM'91] von Tailoring-Dokumenten äußert. Meist dient ein Mitglied der Domäne, welches über einen höheren Grad an Tailoringwissen und Experimentierfreudigkeit verfügt, jedoch kein professioneller Programmierer ist, als Hilfesteller. Gantt und Nardi [GN'92] bezeichnen diesen Typus als *local developer*, während MacLean et al. in einer ähnlichen Definition den Begriff *tinkerer* [MCL'90]) verwendet. Um jedoch unabhängig von dessen Unterstützung Tailoring für alle Endbenutzertypen zu ermöglichen, ist der bereits erwähnte *gentle slope of complexity* elementar. Zusätzlich ist durch die Integration vorhandenen Nutzerwissens, beispielsweise in Form von Metaphern, der notwendige Lernaufwand zu reduzieren. Die in diesem Abschnitt lediglich auszugsweise skizzierten Anforderungen, welche eine Tailoringplattform allgemein erfüllen sollte, werden im Kapitel der Konzeptentwicklung (Kapitel 4) entsprechend der Zielsetzung dieser Arbeit aufgegriffen, erweitert und konkretisiert.

2.2.4 Endusergeeignete Modellierungssoftware

Die folgende Übersicht relevanter EUD-Modellierungskonzepte wird sich aufgrund des technologischen sowie konzeptuellen Kontextes dieser Arbeit auf komponentenbasierte Tailoringplattformen konzentrieren. Das aus der Verknüpfung elektrischer oder mechanischer Bausteine abgeleitete Prinzip realisiert Softwarekomponenten als zustandslose und voneinander unabhängige Funktionseinheiten, welche über eine wohl definierte Schnittstelle angesprochen und zu beliebig komplexen Applikationen kombiniert werden können. Im Regelfall lediglich während der Implementierungsphase von monolithischen Systemen verwendet, ermöglicht der Einsatz unter Gesichtspunkten des Tailoring eine flexible Veränder-, Austausch- und Erweiterbarkeit der Funktionseinheiten.

FreEvolve und Flexibeans

Eine der bedeutendsten Entwicklungen in dieser Hinsicht stellt die client-server-basierte Tailoring-Plattform *FreEvolve* [Hin'99] dar, welche auf dem in Java implementierten Softwarekomponentenmodell *FlexiBeans* basiert [Sti'00; SC'00]. FreEvolve erlaubt eine kooperative Nutzung, Anpassung und Erstellung von Applikationen innerhalb eines verteilten Groupware Systems [SHC'99], wobei die Menge der von den Anpassungen betroffenen Nutzer individuell festgelegt werden kann. Die im Modell vorhandenen abstrakten Komponenten ermöglichen eine beliebig komplexe hierarchische Verschachtlung, welche durch eine zunehmende Abstraktion von technischen Details ein Tailoring auf unterschiedlichen Komplexitätsstufen [MSW'04; WSW'06] erlaubt. Beispielsweise könnte eine einfache Form durch Parametrisierung eines Tailoringdokumentes, welches eine größere Nähe zur Semantik der Anwendungsdomäne besitzt; erreicht werden. Darüber hinaus ist

auf allen logischen Hierarchiestufen und Abstraktionsebenen eine Anpassung durch Ersetzen, Hinzufügen oder Entfernen von Komponenten möglich. Deren Verständnis, Handhabung und Suche soll durch von Endbenutzern erstellte Metadaten und den Einsatz von Metaphern [Ste'02] zusätzlich vereinfacht werden. Der wichtige Stelltenwert der Kollaboration wird durch einfachen Austausch von abstrakten Komponenten und anderen Tailoring-Dokumenten zwischen Endbenutzern untermauert [WSW'06]. Im verwendeten Modell besitzt jede Komponente einen Namen und typisierte Ports, welche entweder Ein- oder Ausgabeschnittstellen repräsentieren. Im Verlauf der Forschung wurde ein grafischer Editor und eine entsprechende visuelle Tailoring Sprache entwickelt.

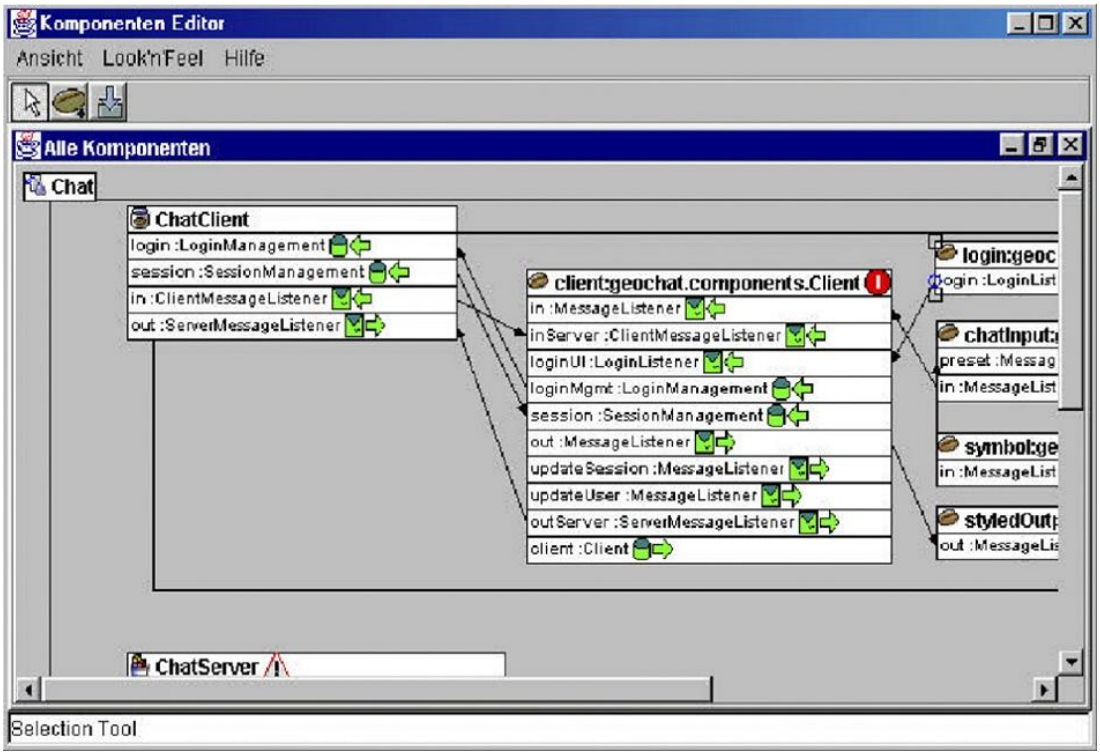

Abbildung 12: Eine der beiden architektur-orientierten Sichten [WPW'08]

Die 2D-Benutzerschnittstelle verfügt über drei Sichten, von denen eine auf visuell darstellbare Komponenten begrenzt ist (ähnlich WYSIWYG), während die beiden anderen unterschiedliche Darstellungen auf Architekturebene erlauben (vgl. Abbildung 12) [WPW'08]. Die datenflussähnliche Verknüpfung dieser Ports erfolgt auf Grundlage der Kompositionssprache CAT (Component Architecture for Tailorability) [Sti'97]. Eine zwischenzeitlich entwickelte 3D-Benutzerschnittstelle zur Verbesserung der Darstellung nicht visueller Komponenten und der Anzeige des Speicherortes der Komponenten wurde aufgrund von festgestellten Usabillity-Nachteilen nicht weiter verfolgt und deren Zielsetzung durch zusätzliche Sichten in die 2D-Benutzerschnittstelle integriert [WSW'06]. Der Einsatz dieser Sichten bietet zwar aufgabespezifische Vorteile, erhöht jedoch durch diese Wahlmöglichkeit zeitgleich auch die grundsätzliche Komplexität des Tailoring. Nicht nur dieses Kriterium motiviert die in der Applikation umgesetzte Darstellung der Komposition, welche in den verschiedenen Nutzungsphasen und Sichten möglichst identisch ist um eine notwendige Neuorientierung zu minimieren. Weitere Entwicklungen erweiterten FreEvolve um eine auf Ereignissen und Nebenbedingungen beruhende Integritätsprüfung [Won'04], eine direkte Aktivierbarkeit [WG'01] des Tailoringmodus [MSW'04; WSW'06] sowie einer Explorationsumgebung zur Simulation erstell-

ter Kompositionen [WSW'06; Wul'00]. Darüber hinaus wurden im Rahmen einer Fallstudie Konzepte der erweiterten Zugriffskontrolle erforscht [Ste'02]. Aktuell bestehen Bestrebungen, das Client-Server-Modell in eine Peer-To-Peer Architektur zu überführen [Ald'06; AC'04].

2.3 SOA im Kontext von EUD

Die in Kapitel 2.1 und 2.2 erzielten Erkenntnisse werden im Folgenden unter dem Gesichtspunkt einer Zusammenführung von SOA und EUD, jeweils kompakt um Aspekte der bisher nicht beachteten Perspektive ergänzt. Den Schwerpunkt dieses Kapitels bildet der anschließende Überblick bisher erfolgter Arbeiten und Realisationen auf dem Gebiet des serviceorientierten EUD. Den Hintergrund bildet die Frage, welche besonderen Anforderungen für die Gestaltung eines endbenutzergerechtes BPEL Editors bestehen und in welchem Umfang diese durch die aktuellen Umsetzungen berücksichtigt werden.

2.3.1 Zusammenfassung

Die in der Praxis gängigsten Umsetzungen einer SOA basieren auf dem Konzept der Web Services. Die für eine Orchestrierung notwendigen Kernstandards SOAP, WSDL sowie BPEL erlauben eine technisch flexible Kombination von Funktionalitäten zu servicebasierten Prozessen. Für eine Nutzung der realisierten BPEL Editoren zur Anpassung und Erstellung servicebasierter Software ist jedoch ein umfassendes Wissen der beteiligten Standards vorausgesetzt. Eine maßgebliche Abstraktion von technischen Details, um anderen Zielgruppen die Anwendung zu ermöglichen, existiert bislang nicht. Wie eine solche Abstraktion aussehen könnte, verdeutlichen einige Vertreter von Editoren, welche die grafische Anpassung und Erstellung service-konsumierender Mashups ermöglichen. Diese meist browserbasierten Mini-Applikationen dienen hauptsächlich der Integration und Bereitstellung unterschiedlicher Daten und unterliegen bei der Verknüpfung allgemein einer sehr begrenzten Reichweite in Bezug auf Mächtigkeit und Flexibilität. Mit Ausnahme der sich in einem sehr frühen Entwicklungsstadium befindlichen EUSOP Plattform existiert weder auf Architektur- noch Applikationsebene eine geeignete Infrastruktur zur Bereitstellung eines für Endbenutzer handhabbaren Konzeptes für die Anpassung und Erstellung servicebasierter Geschäftsprozesse. Nutzerdiversität und die Vielzahl bekannter und unbekannter Faktoren des menschlichen Handelns und Denkens verhindern bislang die Identifikation einer optimalen Form des End-User Development, auch wenn gewisse Grundprinzipien erkennbar sind. Erkenntnisse können meist nur über langwierige Prozesse der Konzeption und Evaluation gesammelt werden, was zu einer breiten Masse von Projekten mit unterschiedlichen Schwerpunkten führte. Der im Hintergrund dieser Arbeit relevanteste Bereich ist der des Tailoring, welcher sich ausschließlich auf eine Anpassung und Erweiterung bestehender Systeme konzentriert. Die komponentenbasierte Tailoringplattform FreEvolve teilt dem SOA Paradigma verwandte Grundprinzipien, kann jedoch nicht deren Grad der Flexibilität erreichen, wie die folgenden Ausführungen verdeutlichen werden.

2.3.2 Bewertung

Zu Beginn stellt sich die Frage, welche Faktoren ein serviceorientiertes Tailoring notwendig machen, wenn komponentenbasierte Systeme, wie die in Kapitel 2.2 betrachtete Tailoringplattform

FreEvolve, eine auf den ersten Blick ausreichende Flexibilität bieten. Auf technischer Ebene ermöglicht zwar das client-server-basierte FreEvolve eine Nutzung und Ausführung von verteilten Komponenten, welche jedoch von den Eigenentwicklungen *FlexiBeans* und *CAT* (Component Architecture for Tailoring) beziehungsweise *DCAT* (Distributed CAT) abhängig sind [WSW'06]. Das SOA Paradigma basiert hingegen auf offenen und allgemein akzeptierten Standards, welches in der komponentenbasierten Welt, besonders im Hinblick auf Tailoring Funktionen nicht der Fall ist. Diese Standards und der allgemeine Trend, Funktionalitäten von Neuentwicklungen und Altsysteme über Web Services verfügbar zu machen, stellt kurz- bis mittelfristig eine nahezu unerschöpfliche Quelle an orchestrierbaren Web Services bereit. Hinzu kommt, dass SOA zusätzlich eine Prozessperspektive bietet, während komponentenbasierte Systeme primär applikationsorientiert aufgebaut sind [WPW'08]. Zusammenfassend kann das SOA Paradigma als konsequente Weiterentwicklung der grundlegenden Prinzipien komponentenbasierter Architekturen angesehen werden. Das Architektur Paradigma ist jedoch erst in Form konkreter technischer Realisationen praktisch nutzbar, wobei sich die in Kapitel 2.1.2 betrachteten Plattformen deutlich in den Punkten Offenheit, Flexibilität und Endbenutzerorientierung unterscheiden. SCA strebt eine größtmögliche Flexibilität und Offenheit an, ohne dass es bisher Hinweise auf die Berücksichtigung einer Orchestrierung durch Endanwender gibt. Dieser Faktor spielt wiederum in der von SAP realisierten ESOA-Plattform *NetWeaver* zwar nicht durchgängig, aber zumindest bis zu einem bestimmten Mächtigkeitsgrad eine Rolle. Eine SAP übergreifende Nutzung und Bearbeitung erstellter Servicekompositionen wird jedoch durch die Verwendung proprietärer Formate stark eingeschränkt. Die E-Government Infrastruktur um *OSCI-Transport* bietet als Einziges der untersuchten Systeme und Konzepte eine fundierte Methodik der Informationssicherheit, welches jedoch zeitgleich die Einbindung beliebiger Web Services erschwert. Darüber hinaus erfolgen Orchestrierungen ausschließlich durch technisches Personal, so dass ebenfalls keine Tailoringaspekte für Endbenutzer berücksichtigt wurden. Lediglich EUSOP bietet eine Nutzung und kompatible Erweiterung der allgemeinen Standards unter EUD Gesichtspunkten, befindet sich aktuell jedoch noch in einer frühen Entwicklungsphase und ist auf die Basisfunktionalitäten begrenzt.

Unabhängig von der auf Architekturebene bereitgestellten Infrastruktur, ist für die Konzeption einer End-User Development Umgebung das Design der Benutzerschnittstelle von entscheidender Bedeutung. Die in dieser Arbeit untersuchten service-komponierenden Editoren erlauben zwar durch visuelle Programmierung ohne manuelle Codeingaben eine komfortablere Handhabung durch technische Anwender, versäumen jedoch diese Konzepte in Richtung einer nicht technischen Anwendergruppe auszudehnen. Beispielsweise fehlt es an grundlegenden Rahmenkonzepten, wie Komplexitätsreduktion und Informationserweiterung sowie der Unterstützung von Kollaboration. Die bisher betrachteten Mashup-Editoren können unter gewissen Gesichtspunkten dem Bereich des EUD zugeordnet werden, sind durch ihre Limitierung auf Datenintegration und Datendarstellung jedoch nicht als funktional vollwertige Konzepte anzusehen. Dessen ungeachtet bieten diese meist browserbasierten Systeme eine Benutzbarkeit ohne tieferes technisches Wissen, welches in den Konzepten jedoch zu Lasten der Mächtigkeit und Flexibilität der realisierbaren Kompositionen geht. Die fehlende Fortführung der SOA Grundprinzipien durch Verwendung individueller, teils proprietärer, Serviceformen und Kompositionssprachen versperrt die Einbindung

externer Services und verhindert einen applikationsübergreifenden Austausch von Mashup-Dokumenten.

2.3.3 Serviceorientiertes End-User Development

Aus diesen und vergleichbaren Gründen wurden, ausgehend von einer EUD Perspektive, Servicekompositionskonzepte für Endbenutzer entwickelt. Zu dieser Gruppe gehört der grafische Mashup Editor *Marmite* [WH'07], dessen datenflussorientierte Verknüpfung von Datenquellen, Datenverarbeiteten Operatoren und Datenausgabeformen an *Yahoo Pipes* erinnert. Eine permanent eingeblendete Sicht stellt abhängig vom selektierten Operator Zwischenergebnisse in einer Tabellenform dar. Von den bisher vorgestellten Konzepten unterscheidet sich das System primär durch ein automatisches Vorschlagsystem „nützlicher“ Operatoren und die geplante Erweiterbarkeit der Operatorenpalette anhand der Angabe von WSDL Dateien [WH'07]. Im Bereich servicekonsumierender Prozesse orientieren sich die meisten Konzepte an BPEL, wie beispielsweise die vereinfachte textuelle Orchestrierungssprache Simple Service Composition Language (SSCL) [GMS'06]. Obwohl eine bidirektionale Transformationsbeziehung zu BPEL besteht, verfügt SSCL bislang über keine Möglichkeit der erweiterten Datenabwicklung oder Variablenmanipulation. Eine nähere Untersuchung ist aufgrund der bisher nicht fertig gestellten browserbasierten Entwicklungsumgebung und der lückenhaften BPEL Unterstützung nicht möglich. Direkt auf BPEL baut das Konzept TailoringBPEL [AKC'07] auf, welches die Entwicklung einer browserbasierten Tailoringplattform auf Basis eines grafischen Editors anstrebt. Obwohl kollaborative Elemente berücksichtigt werden und die Suche nach geeigneten Web Services unterstützt wird, sind in dieser frühen Entwicklungsphase die komplexen technischen Aspekte von Web Services und BPEL bislang nicht ausreichend konsequent vereinfacht.

Zwei technisch vielseitigere Systeme stellen *Triana* [MTS'04] (Abbildung 13) sowie das offene und erweiterbare *JOpera* [Pau'04; Pau'05] (Abbildung 14) dar, welche eine grafische Prozessorchestrierung technologieübergreifender Funktionalitäten erlauben.

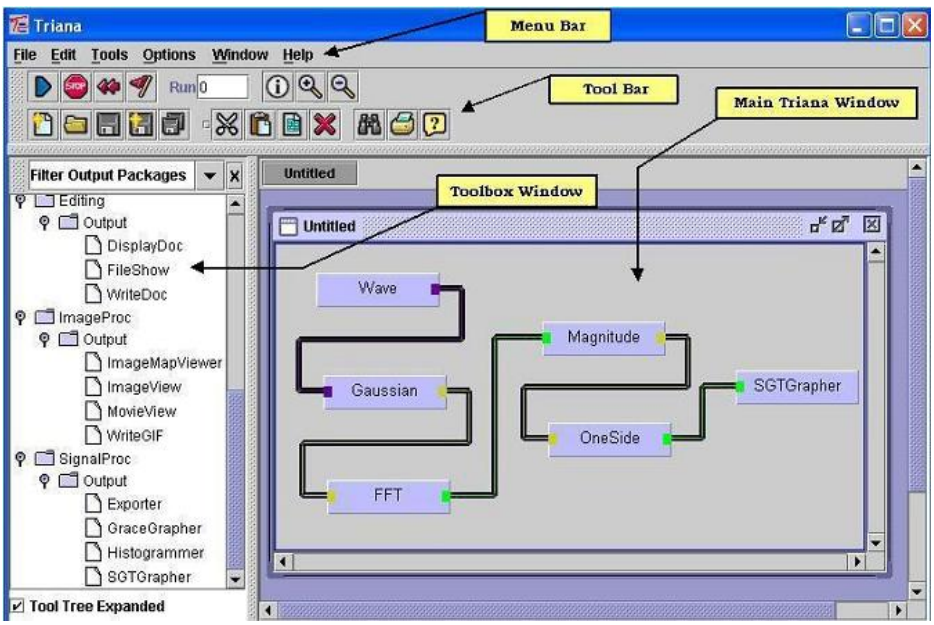

Abbildung 13: Triana Benutzerschnittstelle [Tri]

Triana nutzt hierfür das GAP Interface (Grid Application Prototype) in Kombination mit Bindings zu Web Services, Grid-Services und P2PS-Services (Peer-to-Peer Simplified) [Tri], während *JOpera* mit entsprechenden Adaptern zu Servicetypen, wie beispielsweise Web Services, Java Snippets oder menschlichen Aktivitäten arbeitet. Die Orchestrierung erfolgt durch die Verknüpfung grafischer Repräsentanten nach dem Prinzip der *Box und Wire* Metapher, wobei *JOpera* neben der Datenperspektive noch eine Steuerungsperspektive mit gleicher Notation bereitstellt. Durch Angabe von WSDL Schnittstellenbeschreibungen kann die Menge orchestrierbarer Services in beiden Systemen gezielt erweitert werden. Fertiggestellte Orchestrierungen sind nach Wunsch im Anschluss, durch Veröffentlichung einer WSDL Schnittstellenbeschreibung extern aufrufbar. Von Nachteil ist, dass die Orchestrierungen nur auf dem jeweiligen System lauffähig sind, da die - zumindest in *Triana* - geplante Speicherung im BPEL Format [MTS'04] nach den verfügbaren Informationen [URL,c; Tri] bislang noch nicht vorliegt.Im Kern unterschieden sich die beiden Systeme darin, dass der Fokus von *Triana* auf einer möglichst einfachen Orchestrierbarkeit und dem reibungslosen Zusammenspiel mit UDDI bei Suche und Veröffentlichung liegt, welche durch eine intuitive Notation und Benutzeroberfläche unterstützt wird.

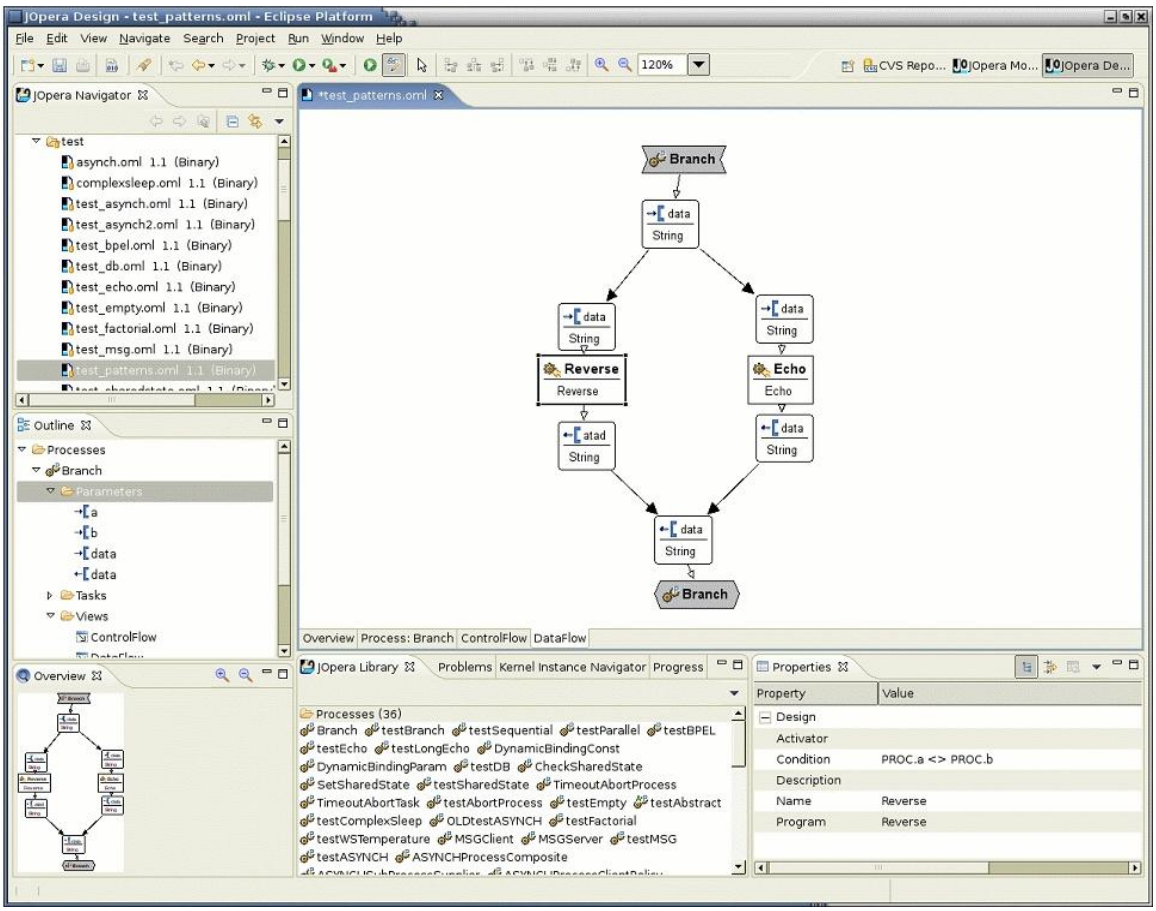

Abbildung 14: *JOpera* Benutzerschnittstelle [Bur'05]

JOpera hingegen versucht eine möglichst breite und umfangreiche funktionale Abdeckung zu erreichen, welche durch diverse technische Möglichkeiten, wie etwa durch späte Bindung [PA'05] und Refaktorisierung [Pau'05] deutlich wird. Im Vergleich zu *Triana* nutzt *JOpera* neben der Datenflusssicht eine zusätzliche Kontrollflusssicht, welche unter anderem die Spezifikation von Schleifenkonstrukten und Fehlerbehandlung ermöglicht [Pau'04]. Darüber hinaus besitzt das System

eine weitreichende Möglichkeit der Überwachung und Steuerung laufender Prozesse, welche mit Hilfe der aus der Modellierung bekannten grafischen Notation erfolgt. Obwohl in beiden Systemen eine deutliche Komplexitätsreduzierung gelungen ist, fehlen Schritte zur weiteren Annäherung der Benutzerschnittstelle an die Bedürfnisse der Endbenutzer, wie es beispielsweise in Form von domänenspezifischen Begrifflichkeiten, Metaphern und kollaborativer Aspekten möglich und notwendig ist. Der deutlichste Mangel ist jedoch die Systemabhängigkeit der Orchestrierungen, welche einen applikationsübergreifenden Austausch ausschließt.

2.3.4 Forschungsfrage

Die fehlende oder lediglich rudimentäre Berücksichtigung endbenutzer-spezifischer Anforderungen der untersuchten service-komponierenden Werkzeuge und Editoren verdeutlichen die Lücke zwischen vorhandener technischer Flexibilität und einer geeigneten Zugänglichkeit für Endbenutzer. Eine der elementarsten Anforderungen stellt das für eine Bedienung notwendige technische Wissen dar, welches im Idealfall möglichst gering sein sollte. Dieses wird jedoch in aktuellen Systemen in eine direkte Beziehung zur erreichbaren Mächtigkeit des Kompositionsproduktes gesetzt, wie es in Abbildung 15 skizziert wird.

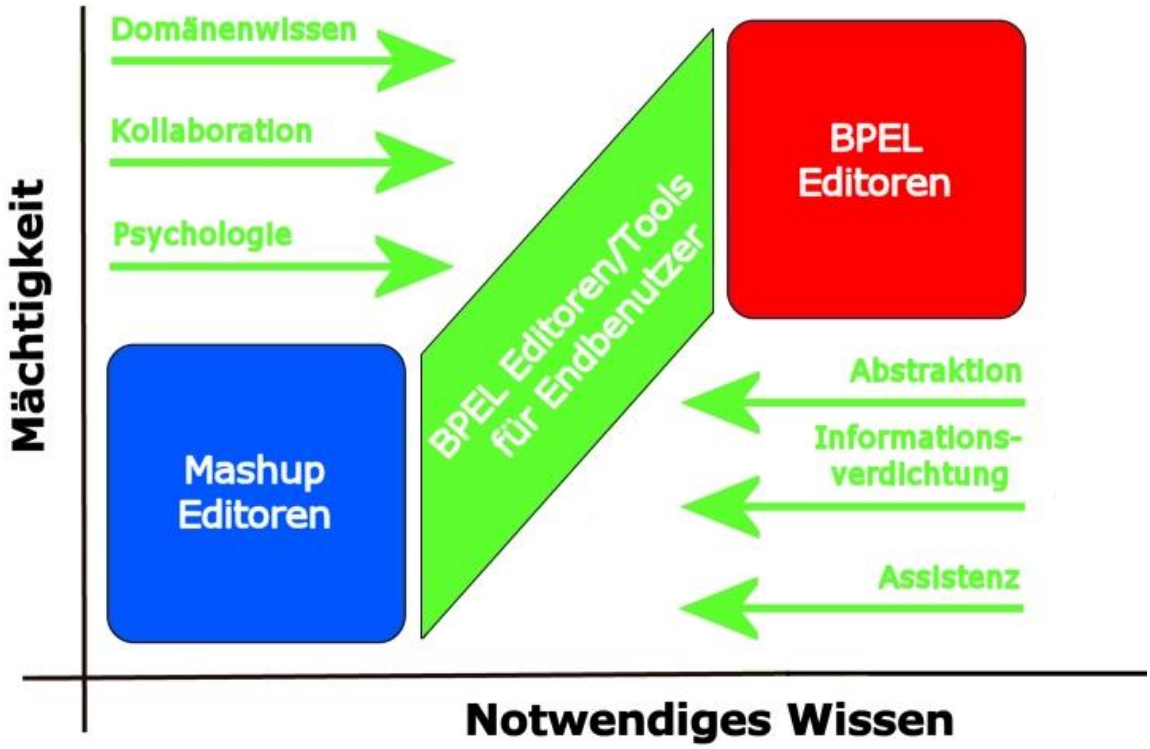

Abbildung 15: Servicekonsumierende Editoren

Mit der Absicht, eine service-orchestrierende Tailoringmöglichkeit für Endbenutzer auf Ebene der informationstechnischen Geschäftsprozessunterstützung zu realisieren, ist das Ziel dieser Arbeit, die speziellen Anforderungen einer Orchestrierung von BPEL Prozessen durch Endbenutzer zu ermitteln und umzusetzen. Konkret soll auf Grundlage dieser Erkenntnisse ein Konzept entwickelt werden, welches auch den technischen Ansprüchen von Mächtigkeit, Offenheit und Flexibilität genügt. Dieses Konzept soll im Anschluss prototypisch implementiert werden. Ein Hauptanspruch des gesetzten Zieles ist daher die Realisation einer möglichst weitreichenden Orchestrierungs-

mächtigkeit, die mit Hilfe eines möglichst geringen Wissensstands nutzbar ist. Grundsätzlich muss hierfür die Web-Service-Technologie im Allgemeinen und die Prozessanpassung im Speziellen durch Abstraktion, Informationsbündelung, nutzergenerierten Inhalten (User Generated Content) und Assistenzfunktionalitäten in ihrer für den Anwender wahrgenommenen Komplexität reduziert werden. Daneben muss bereits vorhandenes Wissen der Endbenutzer integriert werden, um den zur Bedienung erforderlichen Lernaufwand so gering wie möglich zu halten. Dies beinhaltet ebenfalls Aspekte bestehender kollaborativer Arbeitspraxis, wie beispielsweise spezielle Begrifflichkeiten oder Normen der Domäne sowie die Berücksichtigung allgemeiner psychologischer Rahmenbedingungen (vgl. Abbildung 15). Darüber hinaus existiert eine Vielzahl weiterer relevanter Faktoren, deren positive sowie negativen Einflüsse auf das zu entwickelnde Konzept analysiert und berücksichtigt werden müssen.

3 Vorüberlegungen

Als Basis für die folgende Konzeption werden die grundlegenden Rahmenbedingungen festgelegt, welche die Eingrenzung von Anwendungsfeld, Nutzergruppe und Schwerpunkten sowie die Festlegung der im weiteren Verlauf der Arbeit verwendeten Methodik umfassen. Hierauf aufbauend erfolgt für die nachfolgenden Schritte eine Beschreibung relevanter Forschungsbereiche, Konzepte und Werkzeuge.

3.1 Rahmenbedingungen

Die Rahmenbedingungen umfassen die grundlegende Zielsetzung sowie die hierzu notwendige Eingrenzung auf technischer und konzeptioneller Ebene.

3.1.1 Erkenntnisse der Analysephase

Durch die Analyse der bisher betrachteten Konzepte, Applikationen und Forschungserkenntnisse konnten diverse Ursachen identifiziert werden, die ein service-basiertes Tailoring durch Endbenutzer erschweren oder ausschließen. Die gewichtigste Einschränkung stellt das zur grundlegenden Bedienung notwendige Wissen dar. In den vielen Fällen einer nahezu identischen Abbildung der Orchestrierungskomplexität und der vorausgesetzten technischen Sachkenntnis kann unter realistischen Gesichtspunkten ein Endbenutzer diesen geforderten Wissensstand nicht erreichen. Hinzu kommt eine mangelnde Flexibilität auf technischer Ebene, welche aus einer stark eingeschränkten Funktionsmächtigkeit des Ergebnisses, der fehlenden Orchestrierbarkeit beliebiger Web Services und der damit eng verbundenen nicht durchgängigen Nutzung offener und allgemein akzeptierter Standards resultiert. Im Kontext möglichst leicht zugänglicher und aufwandsarmer Editoren fehlt es darüber hinaus an einer umfangreicheren Nutzerorientierung. Besonders deutlich wird dies durch die fehlende Berücksichtigung der Domäne und die unzureichende Unterstützung des allgemeinen Aspektes der Arbeitspraxis – der Kollaboration.

3.1.2 Konkretisierung der Zielsetzung

Für die Konzeptentwicklung bedarf es einer klaren Abgrenzung des zukünftigen Einsatzszenarios und die Festlegung grundlegender Anforderungsschwerpunkte, welche unter anderem auch die oben genannten Kritikpunkte adressieren. Das Konzept soll eine Orchestrierung von Web Services zu informationstechnisch unterstützten Geschäftsprozessen erlauben. Hierbei sollen ausschließlich die Standards SOAP, WSDL, UDDI sowie BPEL Verwendung finden und bei Bedarf lediglich spezifikationskonform erweitert werden, welches eine grundlegende Kompatibilität zu externen Diensten und Prozessen garantiert und eine editorunabhängige Nutzung erstellter Prozesse erlaubt. Der Schwerpunkt liegt auf einer möglichst einfachen, schnell zu erlernenden und funktional mächtigen Orchestrierung, welche die speziellen Anforderungen von Endbenutzern berücksichtigt. Für die genauere Bestimmung der Benutzergruppe muss der in der Literatur nicht konsistent verwendete Begriff des Endbenutzers spezifiziert werden. Die drei von Gantt und Nardi [GN'92] definierten Gruppen des Endbenutzerkontinuums – *non-programmers*, *local developers*, *programmers* – ermöglichen eine erste Abgrenzung. Der im Kontext dieser Arbeit verwendete Endbenutzerbegriff

entspricht der grundsätzlichen Definition des *local developer*, der von Gantt und Nardi wie folgt definiert wird: *„Within a group of users, there is at least one local expert who provides support for other users. We call this person a local developer. The local developer is a fellow domain expert, not a professional programmer, outside technical consultant or MIS staff member."* Diese Definition lässt sich weiter detaillieren und um das Merkmal der natürlichen Experimentierfreudigkeit des von MacLean et al. [MCL'90] spezifizierten Typus des „*tinkerer*" erweitern. Zusammengefasst zeichnet sich im Rahmen dieser Arbeit ein Endbenutzer durch seine Experimentierfreudigkeit im Hinblick auf Anwendung und Anpassung der von ihm verwendeten Informationstechnologie aus. Ferner wird nicht davon ausgegangen, dass der Endbenutzer über Programmierkenntnisse verfügt, sondern ausschließlich über das von ihm erworbene Praxiswissen der eigenen Domäne, wie beispielsweise einer Unternehmensabteilung. Aus diesen Gründen soll die Orchestrierung dem Nutzer in einer primär grafischen Form zugänglich gemacht werden, wobei diese die Konzeption einer geeigneten Repräsentation und Notation beinhaltet, welche mit einem möglichst geringen Lernaufwand nutzbar ist. Darüber hinaus soll eine Anpassung des Editors auf Ebene des Individuums und der Domäne möglich sein sowie die bestehende kollaborative Arbeitspraxis unterstützt werden.

3.1.3 Methodik

Gegen ein klassisches Vorgehen in Form von Analyse, Design und Evaluation und für eine frühzeitige aktive Beteiligung von potentiellen Nutzern in Form einer Vorstudie sprechen vor allem zwei Gründe. Zum einen neigen Nutzer dazu, sich beispielsweise gedanklich nur schlecht von externen Darstellungen [Whi'96] oder allgemeinen Programmiersprachencharakteristiken [NC'01] zu lösen. Die Rückmeldungen von Endbenutzern in einer Evaluation sind daher ungewollt stark vom untersuchten Konzept beeinflusst, welches somit zu fokussierten und weniger kreativen Ergebnissen führt. Zum anderen besteht durch die Neuartigkeit des gesetzten Zieles und den damit einhergehenden Anforderungen keine ausreichende empirische Basis, um die elementaren Designentscheidungen früher Entwicklungsphasen auf Grundlage einer fundierten Informationsbasis treffen zu können. Aus diesen Gründen wird im Folgenden ein elementares Konzept entwickelt, welches die Basis einer nutzerzentrierten Designstudie bildete.

3.2 Grundlagen der Designstudie

Neben der Betrachtung des zur Vorstudie eingesetzten Instrumentes des Participatory Designs, wird auf die Kernelemente der visueller Programmierparadigmen auf relevante psychologische Rahmenbedingungen eingegangen, welche unter anderem durch den Einsatz das *Cognitive Dimension Framework* und eine vereinfachte Form der *Activity Theory* erfolgen.

3.2.1 Participatory Design

Die Entwicklung des Participatory Designs hat ihren Ursprung in der frühen Kooperationen zwischen skandinavischen Gewerkschaften und Softwaredesignern, welche in den 70er Jahren zu den Grundlagen einer nutzerbeteiligten Softwareentwicklung führten. Das NJMF-Projekt (Norwegian Iron and Metal Workers Union) [EK'87] und DEMOS (Democratic Planning and Control of Wor-

king Life) [Ehn'90] stellten in diesem Zusammenhang die ersten praktischen Ansätze eines kooperativen und gleichberechtigten Designs dar. Die Anfang der 80er Jahre durch die Gewerkschaften durchgesetzte Regelung des gesetzlich festgelegten Mitbestimmungsrechts der Arbeitnehmer bei der Gestaltung von Arbeitsmitteln führte zu einer Vielzahl von Konzepten zur Nutzerbeteiligung im Softwareentwicklungsprozess. Aufbauend auf den praktischen Erfahrungen wurden Konzepte des Participatory Designs entwickelt [Bød'96], auch wenn diese aktuell zunehmend an Beachtung zu verlieren scheinen [Bec'02]. Grundsätzlich sollen diese Konzepte die aktive Benutzerbeteiligung während der Designphase sicherstellen und ihnen den Einfluss auf das Endprodukt ermöglichen, wobei in der konkreten praktischen Anwendung dessen Umfang stark variiert. Die Beteiligung findet unter anderem im Rahmen von Participatory Design Workshops statt, welche in einer zeitlich begrenzten Phase ablaufen können oder den gesamten Entwicklungsprozess begleiten. Zur Vorbereitung eines Workshops, wie ihn beispielsweise Kyng beschreibt [Kyn'94], können Analysewerkzeuge wie Interviews oder Beobachtungen zum Einsatz kommen, um die bestehende Arbeitspraxis im Kontext der Organisation zu erfassen [KB'98]. Ein hierbei durch Audio- und Videomedien aufgezeichnetes Material gibt die Möglichkeit, auch nicht artikulierbare Faktoren zu erfassen und auszuwerten. Alternativ oder ergänzend werden in der ersten Phase eines Workshops Einheiten zur wechselseitigen Kommunikation und Erläuterung der jeweiligen Arbeitspraxis von Anwender und Designer eingesetzt, um die praktische Arbeit der Teilnehmer transparenter zu machen. Für den eigentlichen kooperativen Designprozess werden vor allem Prototypen aus Papier, sogenannte Mock-ups [Kyn'88] verwendet. Mock-ups wurden im Rahmen eines der einflussreichsten PD-Projekte dem zwischen 1981 und 1985 durchgeführten Utopia Projekt konzipiert [BEK'87] und werden seitdem konzeptionell für verschiede Forschungssituationen angepasst und weiterentwickelt. Diese und andere Techniken, wie beispielsweise Szenarios bilden eine Möglichkeit, die Zusammenhänge von Arbeit und Technologie aufzuzeigen und zu reflektieren [KB'98]. Mit Hilfe von Mock-ups können Nutzer in Nutzungssituationen versetzt werden und somit auch unterbewusstes Wissen durch diesen Simulations- und Experimentierprozess freisetzen [BGK'93]. Erst sie ermöglichen die Arbeit von Nutzern und Designern auf gleicher Ebene, da komplexe technische Aspekte ausgeblendet werden und auch Nutzer eigenständig Veränderungen sowie Erweiterungen vornehmen können. Diesen Vorteilen und den sehr geringen Herstellungskosten der Papierprototypen stehen die Nachteile einer fehlenden Begrenzung durch die technische Realisierbarkeit und die aufwändigen Änderungen bestimmter physischer Aspekte einzelner Mock-ups gegenüber [BGK'93]. Von besonderem Interesse sind in diesem umfassenden Bereich die Arbeiten rund um PICTIVE (Plastic Interface for Collaborative Technology Initiative through Video Exploration) [Mul'91]. Das Konzept sieht beispielsweise vor, dass Designer auf Grundlage ihrer grundlegenden Wissensbasis über Arbeitskontext und Arbeitsprozesse der Endbenutzer im Vorfeld Mock-ups entwickeln [Mul'92], welche somit als erste Diskussionsbasis dienen können. Neben PICTIVE existieren weiterer Ansätze mit unterschiedlichen Schwerpunkten bei der Umsetzung von Nutzerbeteiligung (u.a. [KB'98; Kyn'88; Sal'94]). Trotz dieses breiten Vielfalt sind die grundsätzlichen Vorteile eine Nutzerbeteiligung für die Softwareentwicklung größtenteils akzeptiert, weisen jedoch eine Kontextabhängigkeit und eine mangelnde empirische Fundierung auf [BF'97; LS'00].

3.2.2 Visuelle Programmierung

Seit die Ressourcen von Computern neben der textuellen auch visuelle Darstellungen erlauben und diese als vorteilhafter angesehen wurden, entwickelte die Forschung Konzepte, welche das Programmieren anhand grafischer Symbole ermöglichen sollen [Bur'99]. Motiviert waren diese unter anderem durch den Aspekt, dass klassische textuelle Programmiersprachen hauptsächlich der Erleichterung der Eingabe, nicht aber dem Verständnis dienten. Ein höheres Verständnis, welches die Grundlage für die Erleichterung des Lern- und Anwendungsprozesses der Programmierung bildet, sollte durch die Nutzung grafischer Notationen erreicht werden [GP'92]. Neben diesem Ziel wurde vor allem die Zugänglichkeit der Programmierung für bestimmte Nutzergruppen und eine Verbesserung der Korrektheit sowie Geschwindigkeit des Programmierprozesses angestrebt [Bur'99]. Die anfängliche Euphorie wandelte sich jedoch zunehmend in ein zwiespältiges Verhältnis, da die scheinbar offensichtlichen Vorteile grafischer Notationen zu Beginn den Blick auf die deutlichen Nachteile dieser Darstellungsform versperrten. Diese frühe Entwicklung ist laut Whitley [Whi'96] auch durch das Versäumnis herbeigeführt, unbestätigte Annahmen nicht durch empirische Untersuchungen zu stützen. Denn, wie Green erkannte, gibt es keine universell beste Darstellungsform, da immer ein Informationsaspekt auf Kosten eines anderen hervorgehoben wird [GP'92]. Aktuell ist die visuelle Programmierung hauptsächlich im professionellen Umfeld erfolgreich. Bis auf wenige Ausnahmen, wie etwa das datenflussorientierte Programm *LabVIEW* (Laboratory Virtual Instruments Engineering Workbench) von National Instruments zur Programmierung im Bereich elektronischer Steuerungs- und Messtechnik, sind diese jedoch nicht für Endbenutzer konzipiert. Primär werden Teilbereiche der Programmierung adressiert, wie beispielsweise Werkzeuge des Computer-Aided Software Engineering (CASE) und der Model Driven Architecture (MDA) sowie Generatoren für grafische Oberflächen. Im Vergleich zur textuellen Programmierung ist eine ausschließlich auf grafischen Notationen basierende Form der klassischen Programmierung noch zu ineffizient [Sch'98]. In der Vergangenheit fehlte es an begrifflich abgeschlossenen Definitionen, welches zu unpräzisen und teilweise falschen Verwendungen des Begriffs führte. Aus diesem Grund erarbeitete Schiffer unter anderem die folgende Definition: *„Visuelle Programmierung ist die Erstellung von Software mit visuellen Programmiersprachen und visuellen Softwarebeschreibungssprachen. Eine Menge integrierter Werkzeuge zur visuellen Programmierung heißt VP-System.“* [Sch'98]. Abhängig vom jeweiligen Einsatzszenario existieren verschiedene Thesen und empirische Ergebnisse über Vor- und Nachteile von visueller im Vergleich zur klassischen textuellen Programmierung. Unter anderem stellen laut Berti et al. [BPS'06] Symboliken und Metaphern die primären Komponenten kreativen Denkens dar. An dieser Stelle wird jedoch nur auf die wichtigsten Punkte eingegangen und für umfassendere Betrachtungen auf Arbeiten von Schiffer verwiesen [Sch'96; Sch'98], welcher die grundsätzlichen Stärken visueller Programmierung bei der Vermittlung von Ideen, einer guten Darstellbarkeit - auch komplexer Zusammenhänge - und der effektiven kognitiven Aufnahme visueller Eindrücke sieht. Dies wird auch von Green teilweise bestätigt, indem er bei Modellen auf Grundlage grafischer Notationen Vorteile bei schnellem Überblicken und Unterscheiden einräumt [GP'92], so dass allgemein von einer effektiveren Informationsaufnahme gesprochen werden kann. Der Anwender erkennt Unterschiede und Abweichungen direkt und kann auf eine Übersichtlichkeit zurückgreifen, die textuelle Darstellungen in der Regel nicht bieten können. Diese Aussage ist jedoch bei weitem nicht für alle visuelle Darstellungsformen

gültig, da sie unterschiedlichste Strukturen aufweisen, die teilweise auch Elemente enthalten können, welche die Aufnahme erschweren können [GP'92]. Hinzu kommt, dass sich dieser Vorteil durch den hohen Platzverbrauch der Darstellung ab einer gewissen Komplexität negiert. Für eine einfachere Nutzbarkeit sind hingegen die direkte Manipulierbarkeit und das hieraus unmittelbar resultierende visuelle Feedback von Vorteil. Außerdem können syntaktische Regeln implizit in das VP-System integriert werden, so dass beispielsweise Verknüpfungen nur zwischen kompatiblen Objekten möglich sind. Ein positiver Einfluss auf die Effektivität der Problemlösung durch eine visuelle Darstellung ist denkbar, aber nicht bestätigt, da Menschen laut Whitley dazu neigen, externe Repräsentationen kognitiv zu übernehmen. Dieses mangelnde Abstraktionsvermögen führt zu einem starken Einfluss der Darstellungsform auf die Konstruktion interner Modelle, welche wiederum einen wichtigen Faktor für eine effektive Problemlösung darstellt [Whi'96]. Im Bereich der negativen Aspekte dominiert die schlechte Skalierbarkeit des benötigten Aufwandes, da zwar meist eine schnelle Lösung einfacher Aufgabenstellungen ermöglicht wird, aber komplexe Probleme eine unverhältnismäßig hohe Anstrengung erfordern. Ebenso sind nachträgliche Änderungen an Programmen mit einem überproportional hohen Aufwand verbunden [PM'96]. Eine Klassifikation lässt sich auf Grundlage des ACM CRS (Association for Computing Machinery, Computing Review System) von Burnett und Baker [BB'94] sowie der Gruppierung in drei Hauptformen [Sch'98] vornehmen, in der das Kriterium der verwendeten visuellen Sprache und deren konzeptioneller Orientierung dominiert. Diese Hauptgruppen sind konkret VP-Systeme, welche auf verbalen bzw. visuellen Sprachen basieren sowie VP-Systeme für spezielle Sprachen. Aufgrund der größeren konzeptionellen Nähe zur Orchestrierung konzentriert sich die weitere Betrachtung jedoch lediglich auf die Gruppe der VP-Systeme, welche sich an verbalen Programmiersprachen orientieren. In dieser Gruppe dominieren die datenfluss- und steuerflussorientieren VP-Systeme, welche durch die Datenverfügbarkeit beziehungsweise den Eintritt von Ereignissen gesteuert werden.

Navarro-Prieto und Cañas [NC'01] untersuchten den Zusammenhang von Verständnis, mentaler Repräsentation und visueller sowie textbasierter prozeduraler Programmiersprachen. Sie formulierten die Hypothese, dass Programmierer visueller Sprachen schneller eine datenflussbasierte mentale Repräsentation entwickeln können, als Programmierer prozeduraler Sprachen. Hintergrund hierfür bildeten die Forschungsaussagen, dass Programmverständnis primär auf einer datenflussbasierten mentalen Repräsentation basiert und eine visuelle Darstellung die schnellere Aufnahme aussagekräftiger Informationen ermöglicht. Eine experimentelle Überprüfung mit C-Programmieren und Tabellenkalkulations-Anwendern unterstützte diese Hypothese. Im Kontext dieser Arbeit spricht damit für eine hauptsächlich datenflussbasierte Orientierung des Editors, dass hierdurch die Entwicklung der entsprechenden mentalen Repräsentation begünstigt wird, welche für das Programmverständnis primär ausschlaggebend ist. Darüber hinaus stellten Navarro-Prieto und Cañas bei Tabellenkalkulations-Programmierern, welche im unternehmerischen Endbenutzerkontext eine stärkere Verbreitung besitzen, die große Nähe zu dieser Steuerungsform fest. Abschließend spricht der größere Verbreitungsgrad visueller Datenflusssprachen im Bereich endbenutzeradressierender Software (vgl. Kapitel 2.2 und 2.3) und die damit verbundene Vertrautheit für einen Teil der Endbenutzer für diese Ausrichtung. Aus diesen Gründen wird der weitere Fokus dieser Arbeit primär auf datenflussorientierten VP-Systemen liegen.

Ein solches System besteht aus verschiedenen Komponenten, die visuell verknüpft werden, um den Datenfluss und damit die sequentiellen Abhängigkeiten zu spezifizieren. Dabei besitzen die in Abbildung 16 skizzierten Grundtypen (Datenquelle, Datensenke, Operationen) jeweils eine gewisse Anzahl von Dateneingängen (nicht bei Datenquellen) und Datenausgängen (nicht bei Datensenken). Für die Abbildung einer Sequenz ohne Datenabhängigkeit verfügen VP-Systeme meist zusätzlich über spezielle Mechanismen.

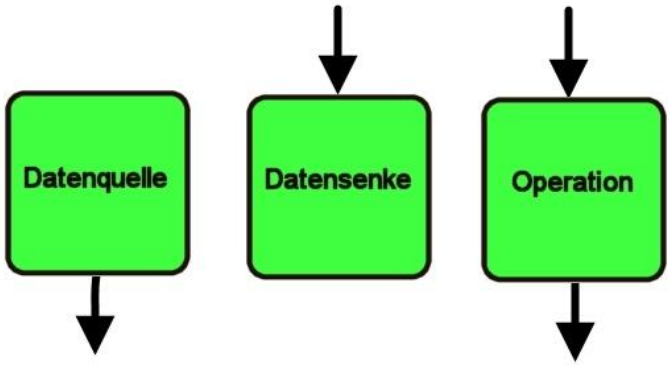

Abbildung 16: Grundtypen innerhalb eines Datenflusses

Das sowohl am Markt erfolgreichste und in der Forschung am häufigsten als Grundlage verwendete datenflussorientierte VP-System [BH'94; BP'99; GP'96] stellt das bereits zuvor erwähnte *LabVIEW* dar, welches die Programmierung virtueller Instrumente im Bereich der Mess- und Automatisierungstechnik unter Ausnutzung der Schaltkreismetapher erlaubt. Mittlerweile umfasst das ursprünglich primär auf die Messdatenverarbeitung ausgerichtete Programm ein Anwendungsspektrum von Datenerfassung, Instrumentensteuerung und Instrumentenautomation. Als Sprache verwendet *LabVIEW* die visuelle Datenflusssprache *G*, welche das klassische Konzept dieser Sprachart um Sequenzen und lokale/globale Variablen erweitert. Zusätzlich gibt es Konstrukte, welche Schleifenkonzepte in die visuelle Gestaltung integrieren können. *LabVIEW* verfügt zusätzlich über Hilfsmittel, wie beispielsweise die Möglichkeit, den animierten Programmablauf schrittweise auszuführen, Haltepunkte zu setzen oder sogenannte Sonden einzurichten, welche Veränderungen einer Variablen anzeigen.

3.2.3 Notation & Metapher

Eng verknüpft mit der visuellen Programmierung ist die Notationen der verwendeten Elemente, welche zusammen mit der Umgebung das System darstellen. Allgemein legt eine Notation die formale Struktur zur Formulierung eines Befehles in der verwendeten Programmiersprache fest. Bei einer grafischen Notation setzt sich diese Struktur und Formulierung aus grafischen Objekten zusammen, wie es am Beispiel von BPEL häufig durch die Nutzung der BPMN Symbolen geschieht. Somit legt die Notation die Modellbasis fest, welche den Ansprüchen der verschiedenen Nutzungsphasen von Verstehen, Erstellen, Anpassen, Ausführen sowie dem Beseitigen von Fehlern im Verlauf des Lebenszyklus eines nutzergenerierten- oder angepassten Modells genügen muss [Gre'96]. Indirekt werden so durch die Notation Faktoren festgelegt, welche den Lernaufwand und die Nutzung erleichtern oder erschweren, wie etwa eine implizite Integration von Syntaxregeln. Eine „gute“ Notation ist jedoch von der zu erfüllenden Aufgabe abhängig [PM'96] und kann

meist nur einen bestimmten Informationsaspekt zu Lasten eines anderen hervorheben, so dass eine allgemein optimale Notation nicht existiert [GP'92]. Trotz dieser Argumente sollte die Notation, zumindest in großen Teilen, darstellungs- und phasenübergreifend beibehalten werden, um eine ansonsten häufig notwendige Neuorientierung der Nutzer (z.B. beim Wechseln der Ansicht) zu vermeiden [LPK'06]. Eine Neuorientierung kann zusätzlich dadurch reduziert werden, indem sich das Notationsmodell an den Charakteristiken der mentalen Modellen von Menschen orientiert, welches zusätzlich den kognitiven Aufwand zur Übertragung und Transformation verringert [BPS'06]. Nicht nur im Bereich der Notation und Repräsentation wird häufig auf vorhandenes Nutzerwissen in Form von Metaphern zurückgegriffen, um ein leichteres Verständnis abstrakter informationstechnischer Aspekte zu ermöglichen. Hierbei ist jedoch grundsätzlich zu beachten, dass nicht zwingend eine passende Metapher existieren muss [SQK'06]. Die Übertragung von einem realen Objekt, beispielsweise einem Kopiergerät auf eine abstrakte Funktionalität, wie etwa des informationstechnischen Kopierens, birgt zusätzlich ein gewisses Fehlerpotential [PM'96]. Zum einen können beim Nutzer Fehlannahmen entstehen, indem gewisse Eigenschaften des realen Objektes übertragen werden [Sch'98], wie im obigen Beispiel durch die Vermutung, dass eine Kopie automatisch mit einem Qualitätsverlust behaftet ist. Zum anderen bieten Metaphern immer einen gewissen Interpretationsspielraum, so dass besonders Faktoren wie persönliche Erfahrungen oder das Aufwachsen in anderen Kulturkreisen zu gravierenden Missinterpretationen und damit zu Orientierungsproblemen oder unerwartetem Systemverhalten führen können. Auf Ebene der von Blackwell und Green untersuchten visuellen Notation, können diese Fehlannahmen und Interpretationsprobleme ebenfalls auftreten, da laut ihren Untersuchungen Endbenutzer dazu neigen eigene Erklärungsmetaphern zu erstellen, falls keine Metapher zur Verfügung gestellt wird. Obwohl auf dieser Ebene ein genereller positiver Effekt auf Lerngeschwindigkeit und Usability empirisch nicht bestätigt werden konnte [BG'99], erreichte beispielsweise eine abstraktere Form der *Box und Wire* Metapher, welche auf der Verknüpfung elektronischer Bauteile basiert, eine breite Akzeptanz (vgl. Kapitel 2.2 und 2.3).

3.2.4 Kontext & Kollaboration

Nicht nur bei der Auswahl geeigneter Metaphern ist die Berücksichtigung des Kontextes der Nutzung ein wichtiger Faktor [PM'96]. Durch die Integration domänenspezifischer Aspekte, wie etwa Begrifflichkeiten, Normen oder Metaphern, können, neben der Möglichkeit den Lernaufwand zu minimieren, ebenfalls motivationsfördernde und nutzungshürdenabbauende Effekte erzielt werden. Um das System in unterschiedlichen Domänen nutzen zu können, gilt es einen Ansatz zu finden, der einen einfachen, umfassenden und flexibeln Austausch domänenspezifische Systemelemente ermöglicht. Diese Berücksichtigung auf Gruppenebene kann jedoch nicht die grundsätzliche Heterogenität der individuellen Nutzer adressieren, welche nur durch eine möglichst umfassende individuelle Anpassbarkeit der Tailoringumgebung ermöglicht werden kann. Einen elementaren Aspekt der allgemeinen Arbeitspraxis stellt Kollaboration dar, welche im zunehmenden Maße auch über Software erfolgt [PK'06]. Die hierdurch entstehende Beeinflussung der Kollaborationspraxis, beispielsweise durch Zugänglichkeit eines neuen E-Mail-Kommunikationskanals oder die Vorgabe einer gewissen Ausführungsreihenfolge stellt sich bei genauerer Betrachtung als bidirektionale Beziehung heraus. Bereits bei der reinen Softwarennutzung wird diese von kollaborativen Aspekten beeinflusst, wie die Untersuchungen von Nardi und Miller anhand von Tabellenkalkulationen

feststellen konnten [NM'91]. Beim Tailoring ist dieser Einfluss ebenso präsent, wie es beispielsweise durch den Austausch von Tailoringdokumenten zwischen Kollegen deutlich wird [Mac'90]. Darüber eignen sich Endbenutzer das Wissen, welches beim Tailoring zur Anwendung kommt, häufig durch soziale Interaktion an [Pi'05]. Tailoring stellt also nicht das Ergebnis eines isoliert handelnden Individuums dar und muss diesen kollaborativen Einfluss in der Konzeption berücksichtigen und unterstützen, wie es im oben dargestellten Beispiel durch die Schaffung eines zentral zugreifbaren „Lagerortes“ für Anpassungsdateien denkbar wäre. Eine detaillierte Betrachtung der kollaborativen Aspekte des Tailorings und dessen angrenzender Bereiche wird durch die Arbeit von Pipek [Pi'05] vermittelt, welcher die Aktivitäten von Tailoringnutzern untersucht und entsprechende Unterstützungsmöglichkeiten aufzeigt.

3.2.5 Psychologische Faktoren

Über die bisher genannten grundlegenden Punkte hinaus sind zusätzliche psychologische Faktoren relevant, um eine möglichst einfache Erlernbarkeit, eine anhaltende Motivation und die Vermeidung von Nutzungshürden zu realisieren. Grundsätzlich können die Erkenntnisse des Forschungsbereiches der natürlichen Programmierung wertvolle Hinweise zur Gestaltung geben, wie beispielsweise die von Programmierlaien gewählten Formen der Programmierung oder häufig auftretende Fehlertypen [KM'05]. Unabhängig von der Form sollte Tailoring grundsätzlich auf verschiedenen Stufen von Mächtigkeit und damit auch auf verschiedenen Stufen der Komplexität [MCL'90; WPW'08] möglich sein, um je nach Wissensstand einen adäquaten Anpassungszugang zu bieten. Die Lernkurve, welche die Beziehung der Dimensionen der Nutzerfähigkeiten und Tailoringmächtigkeit abbildet, sollte parallel hierzu möglichst flach ansteigen und keinerlei Sprünge enthalten, damit Nutzer durch kleine Lernschritte direkt neue Nutzungsmöglichkeiten erschließen können.

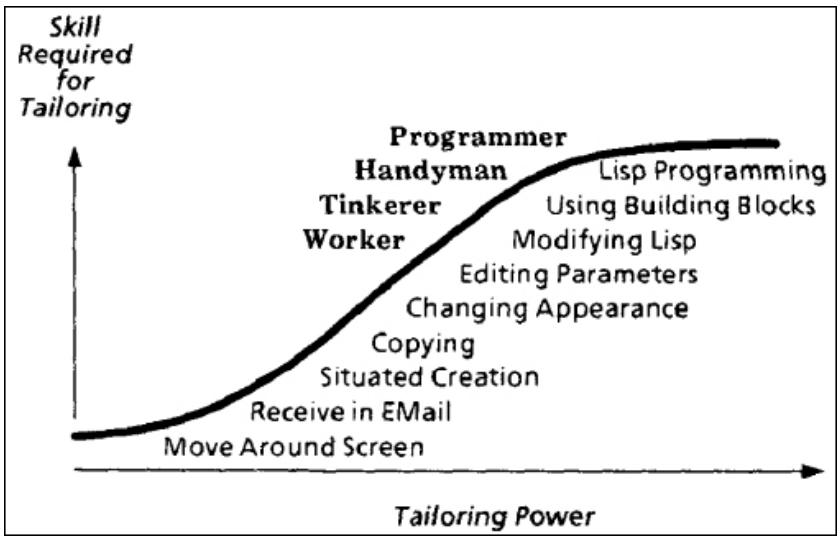

Abbildung 17: Flache Lernkurve [MCL'90] am Beispiel von *Buttons*

Die aus der Arbeit von MacLean et al. entnommene Abbildung 17 zeigt eine solche flach ansteigende Lernkurve am Beispiel der Tailoringmöglichkeiten von *Buttons* [MCL'90]. Der hierdurch erreichte positive Effekt auf die Lernmotivation kann besonders im Rahmen der erstmaligen Auseinandersetzung mit der Tailoringsoftware durch Aufgreifen bekannter Elemente der Arbeitsdomäne

oder der realen Welt verstärkt werden. Die größte Nutzungshürde in dieser ersten Phase ist die Angst, durch das eigene Handeln Fehler oder Schäden im System zu produzieren, welcher durch Einsatz einer Simulationsumgebung entgegengewirkt werden kann. Im Hinblick auf die grundlegende Motivation stellen Modellierungsfehler einen kritischen Faktor dar, welcher durch Eingrenzung möglicher Fehlerquellen, frühzeitige Hinweise oder einer breiten Informationsbasis zu Fehlerursache und Fehlerbehebung vorgebeugt werden kann.

3.2.6 Cognitive Dimension Framework

Des weiteren wird hauptsächlich während der konkreten Konzeption der Notation und daran angrenzender Faktoren das von Green [Gre'89] entwickelten Framework der „cognitive dimensions of notations“ [GP'96] verwendet, welches allgemein während der Entwicklung sowie Evaluation von Modellierungs- und Programmiersprachen zum Einsatz kommt. Es ordnet bekannte aber in der Literatur nicht homogen verwendete Aspekte in eine Struktur ein, etabliert ein einheitliches Vokabular und zeigt Kompromissbeziehungen, d.h. Abhängigkeiten zwischen den dreizehn definierten Dimensionen auf. Konkret besteht das Framework aus den Dimensionen Abstraction Gradient, Closeness of Mapping, Consistency, Diffuseness/Terseness, Error-proneness, Hard Mental Operations, Hidden Dependencies, Premature Commitment, Progressive Evaluation, Role-expressiveness, Secondary Notation/Escape from Formalism, Viscosity und Visibility/Juxtaposability. Den Hintergrund des Frameworks bildet die Fragestellung, ob die vom Nutzer angestrebte Aktivitäten, welche auf den kognitiven Strukturen und deren Einschränkungen basieren, adäquat von der Struktur des Informationsartefaktes unterstützt werden [Gre'00]. Die Anwendung erfolgt auf der Festlegung der von dem Informationsartefakt unterstützten generischen Aktivitäten, wie beispielsweise Suchen oder Modifizieren, und wie deren einzelne Elemente die jeweils relevanten Dimensionen unterstützen. Im Kontext dieser Arbeit dienen die definierten Dimensionen während der Konzeption als Orientierungshilfe, auf welche in den entsprechenden Stellen des Kapitels 4 verwiesen wird.

3.2.7 Aktivitätstheorie

Die Aktivitätstheorie (Activity Theory) ist ein Zweig der marxistischen Psychologie, welche in der früheren Sowjetunion entwickelt wurde. Dessen ursprünglicher Fokus auf die menschliche Entwicklung wurde jedoch in den letzten Jahrzehnten auf viele andere Gebiete ausgeweitet. Basierend auf der Dissertation von Bødker [Bød'91], welche die Aktivitätstheorie im Rahmen der HCI auf die Gestaltung der Benutzeroberfläche erweitert, wird im Folgenden ein vereinfachtes Modell beschrieben, welches primär als grundlegende Orientierungshilfe während der Konzeption der Umgebung dienen soll. Ziel ist es neben der Einbeziehung kognitiver Aspekte bei Notation und Lernmotivation, den Editor ebenfalls im Hinblick auf allgemeines menschliche Handeln und Werkzeugnutzung zu gestalten. In der Darstellung durch Bødker [Bød'91] werden menschliche Handlungen durch ein Bedürfnis oder einen Wunsch motiviert, welches beim Subjekt zur Ausführung einer Aktivität (Activity) führt. Diese setzt sich aus zielgerichteten und bewusst durchgeführten Aktionen (Actions) zusammen und stellt somit einen Prozess zur Befriedigung des Bedürfnisses dar. Aktionen wiederum setzen sich aus Operationen (Operations) zusammen, die durch herrschende Rahmenbedingungen ausgelöst und hauptsächlich unterbewusst ausgeführt werden. Die

hierarchische Beziehung dieser Elemente menschlichen Handelns wird durch Abbildung 18 verdeutlicht.

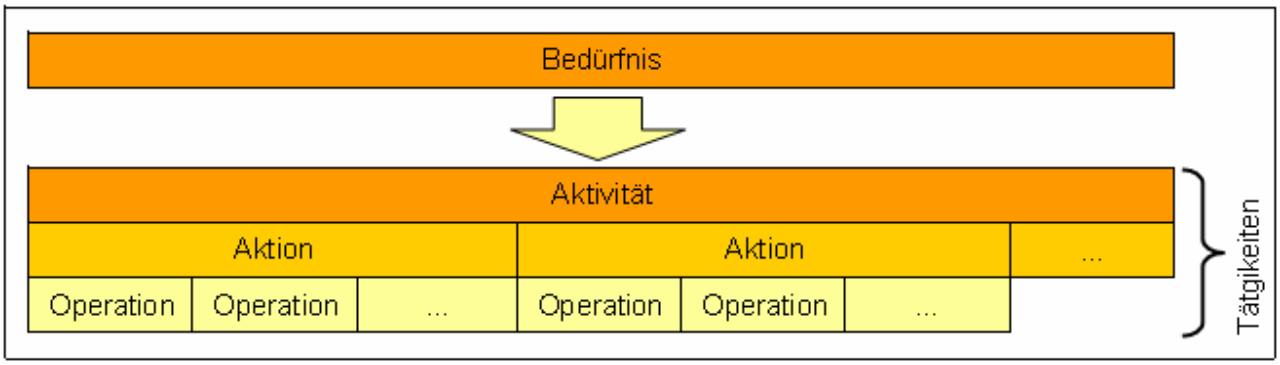

Abbildung 18: Hierarchie menschlichen Verhaltens

Die Tätigkeiten auf diesen drei Ebenen zielen hierbei stets auf die Veränderung eines Subjekts oder eines physischen oder nicht-physischen Objekts. Analoge Beispiele ohne Berücksichtigung einer einheitlichen Tätigkeitsebene sind das Vermitteln einer bestimmten Information, das Öffnen einer Flasche oder das Programmieren eines Algorithmus. Dabei wird das Subjekt oder Objekt meist nicht direkt, sondern durch die Nutzung von Werkzeugen oder Hilfsmitteln verändert. Konform zu den obigen Beispielen wären dies Sprache, ein Flaschenöffner oder ein Computerartefakt. Dabei werden die jeweiligen Tätigkeiten durch die Eigenschaften der Ziele und Objekte sowie durch die genutzten Werkzeuge begrenzt und strukturiert. Allgemein sind also die Auswahl und Ausführung der Tätigkeiten durch die herrschenden bzw. wahrgenommen situativen Faktoren beeinflusst, werden jedoch von diesen nicht determiniert. Grundsätzlich können Aktionen mit zunehmender Wiederholung operationalisiert werden, so dass sie in das Repertoire der automatisiert nutzbaren Operationen übergehen, wie es beispielsweise beim Benutzen der Gangschaltung eines Fahrzeuges der Fall ist. Der entgegengesetzte Weg, das Konzeptualisieren, ist ebenso möglich und kann auf ungewohnte Situationen (Nutzung der Gangschaltung in einem Fahrzeug für Linksverkehr) oder beispielsweise einer beabsichtigten Reflektion (jemanden die Nutzung der Gangschaltung erklären) beruhen. Im Kontext der Konzipierung liegt der Fokus auf der Unterstützung von Operationalisierung und der Vermeidung designbedingter Konzeptualisierung. Da Tätigkeiten auf Subjekte und Objekte größtenteils durch Werkzeuge vermittelt werden, stehen die Designentscheidung im Hinblick auf die Editorumgebung diesbezüglich im Fokus. Hierbei ist es allgemein wichtig, dass in der späteren Nutzung das Ziel der jeweiligen Tätigkeiten auf das zu bearbeitenden Objekt oder Subjekt zielen können, ohne diese gezwungenermaßen auf das Werkzeuge richten zu müssen. Eine solche Fokusverschiebung des Anwenders wir beispielsweise durch Popup-Fenster, welche gelesen und geschlossen werden müssen, ausgelöst. Im Kern sollte daher die Editorumgebung eine der Bedeutung der Nachricht angepasste Form der Kommunikation mit dem Nutzer verwenden sowie zeitgleich diese und weitere Aspekte derart gestalten, dass Operationalisierung unterstützt und ermöglich wird. Darüber hinaus muss sich beispielsweise die Umgebung in gleichen Situationen konsistent Verhalten und dem Nutzer die aktuell herrschenden situativen Faktoren, welche hier als die Materialbedingungen verstanden werden können, transparent machen.

Zusammenfassend kann das Ziel dieser Diplomarbeit nur durch die Reduzierung des benötigten Wissens, welches zur Nutzung der Tailoringplattform vorausgesetzt wird, erreicht werden. Hierfür ist es essentiell, von der Komplexität der technischen Basis zu abstrahieren, ohne die funktionale Mächtigkeit und Flexibilität einzuschränken und parallel domänenspezifische, kollaborative und allgemeine psychologische Gesichtspunkte zu berücksichtigen. Während der Konzipierung dienen darüber hinaus kognitiv- und lernpsychologische Forschungsergebnisse als Orientierung, welche durch Ergebnisse der Durchführung eines Participatory Design Workshops ergänzt werden sollen.

4 Konzept

Die Grundlage des im Weiteren vorgestellten Konzeptes bildet die in Kapitel 4.2 beschriebe Durchführung einer in Form eines Participatory Design Workshops durchgeführten nutzerbeteiligten Vorstudie. Ausgehend von den hierbei gewonnenen Design-Erkenntnissen wurde unter zu Hilfenahme der Literatur das im Verlauf dieses Kapitel dargestellte Konzept entwickelt, welches nach dem Prinzip einer Gegenüberstellung von investiertem Aufwand und realisiertem Nutzen strukturiert ist. Das Konzept grenzt sich von den in Kapiteln 2.2 und 2.3 vorgestellten Realisationen primär durch seine mächtige und zu gleich möglichst einfache Bearbeitung serviceorientierter Prozesse in einem spezifischen unternehmerischen Anwendungskontext ab. Diese Abgrenzung erlaubt die Bereitstellung eines auf die Anforderungen und die Arbeitspraxis der Domäne abgestimmten Editors. Die relativ geschlossene Nutzergemeinschaft mit ähnlichem Arbeitskontext begünstigt den Einsatz kollaborativer Aspekte, welche sich beispielsweise durch gegenseitigen Hilfestellungen, dem Austausch von Prozessdokumenten und dem Aufbau funktionalitätsspezifischer Informationsquellen ausdrücken. Besonders deutlich unterscheidet sich das Konzept von betrachteten Realisationen durch die technologische Flexibilität, welche sich durch die Erweiterbarkeit der orchestrierbaren Funktionalitäten per Import der WSDL-Schnittstellenbeschreibung und die Ausgabe spezifikationskonformer BPEL-Prozesse ausdrückt. Ausgehend von dieser Abgrenzung wurde für das Konzept die Bezeichnung SiSeOr (Simple Service Orchestration) gewählt, welches im weiteren Verlauf auch synonym mit dem zu entwickelnden Softwareprototyp verwendet wird.

4.1 Grundlegendes Konzept

Basierend auf den ermittelten Schwerpunkten und den gesammelten Erkenntnissen bildet das im Folgenden entwickelte Grundkonzept die Ausgangsbasis des später beschriebenen Participatory Design Workshops. Die Orchestrierung von BPEL Prozessen lässt sich im Kern auf eine Auswahl von Web-Service-Operationen reduzieren, deren Aufrufreihenfolge von der Verknüpfung der Datenaus- und Dateneingänge bzw. der Datenverfügbarkeit derselbigen abhängt. Zur Vereinfachung für den Nutzer bilden Web-Service-Operationen im Konzept eigenständige Konstrukte und werden nicht innerhalb der entsprechenden Web Services geschachtelt. Aufbauend auf dieser Abstraktion und Komplexitätsreduzierung wird eine Orchestrierung mit Hilfe einer datenflussorientierten Anordnung der essentiellen Elemente, respektive deren grafischer Repräsentanten, erfolgen. Die Entscheidung für eine visuelle Programmierung basiert auf einer Neubewertung beider Programmierformen (textuell, visuell) im Kontext dieser Arbeit, wobei hierbei zwei Faktoren ausschlaggebend sind. Zum einen ist dies ein geringer Lernaufwand bei gleichzeitig hoher Mächtigkeit, wie es Fischer et. al bei ihrer Arbeit an einem EUD Meta-Design Konzept aufführen [FGY'04]. Zum anderen die Nähe zur mentalen Modellierung, wie es Blackwell et. al unter anderem als Möglichkeit einer natürlicheren modell-basierten Entwicklungsumgebung für Endbenutzer nennt [BPS'06].

Auf Grundlage der der *Box und Wire* Metapher werden Web-Service-Operationen in SiSeOr durch Rechtecke (Box) und der Datenfluss durch gerichtete Verbindungslinien (Wires) repräsentiert. Als Anknüpfungspunkte dienen die jeweiligen Datenein- bzw. -ausgänge der Web-Service-

Operationen, welche durch entsprechende Felder innerhalb der Rechtecke dargestellt und im weiteren Verlauf als Ports bezeichnet werden. Die visuelle Differenzierung der Ein- und Ausgangsports kann hierbei unter anderem durch Formgebung, Farbgestaltung, Symboliken und die Positionierung erreicht werden. Besondere Repräsentationsformen erhalten Web-Service-Operationen, welche geschachtelten BPEL Prozessen entsprechen, um dem Nutzer die Erreichbarkeit dieser logischen Ebenen anzuzeigen. Konkrete Überlegungen in diese Richtung finden jedoch erst nach Durchführung und Analyse der Vorstudie statt. Basierend auf diesen grundlegenden Konzeptentscheidungen ist der Nutzer in der Lage, Web-Service-Operationen durch die visuelle Verknüpfung der entsprechenden Datenports zu einem BPEL Prozess zu orchestrieren, ohne dass er die darunter liegenden technischen Strukturen kennen muss. Für eine einfache Verwendung durch Endbenutzer fehlt es jedoch an geeigneten Informationen, auf deren Basis ein Verstehen, Auffinden und Auswählen geeigneter Web-Service-Operationen oder Prozesse erfolgen kann. Konform zu den WSDL Spezifikationen werden daher den Schnittstellenbeschreibungen zusätzliche Metadatenfelder hinzugefügt, in denen diese Informationen unter anderem in natürlicher Sprache angegeben und von den Nutzern verändert werden können. Die Bereitstellung von Diensten und Prozessen wird in Form eines zentralen Speicherortes erfolgen, dessen Menge orchestrierbarer Funktionalitäten durch Import neuer WSDL-Schnittstellenbeschreibung beliebig erweitert werden kann. Der Zugriff ist durch „Stöbern“ oder eine differenzierte Suchfunktion möglich, wobei hierfür die Quantität und Qualität verfügbarer Informationen der zusätzlichen Metadaten entscheidend ist. Diese können beispielsweise eine in natürlicher Sprache verfasste Beschreibung der Funktionsweise oder kollaborativ produzierte Bewertungen und Schlagwörter beinhalten.

In der Phase der Anpassung oder Erstellung eines Prozesses sind Modelländerungen direkt durch die Manipulation der entsprechenden Verbindungen zwischen den Datenports möglich. Durch eine automatische Syntaxüberprüfung werden potentielle Fehlerquellen reduziert und das Erlernen dieser Regeln erleichtert. Da Modellierung in evolutionären Schritten meist nach einem Versuch und Irrtum Prinzip durchgeführt wird, nehmen eine möglichst direkte und einfache Änderbarkeit sowie die Rückgängigmachung von Nutzeraktionen zentrale Punkte im Konzept ein. Basierend auf Untersuchungen im Bereich mentaler Modelle und natürlicher Programmierung, beispielsweise durch Blackwell und Peters [BP'99] sowie Myers und Ko [MK'05] wird es möglich sein, auch unvollständige Orchestrierungen in Teilen auszuführen. Des Weiteren können Änderungen am Modell ebenfalls auf Gruppen von Objekten angewendet werden, um somit möglichst nah an die Flexibilität kognitiver Modelle zu gelangen. Der allgemein herrschenden Besorgnis der Endbenutzer, durch das Ausführen von Prozessen Schäden am System zu produzieren, wird durch die Bereitstellung eines Simulationsmodus begegnet. In diesem kann mit Hilfe von Testdaten das Verhalten des Prozesses überprüft werden, ohne das Produktivsystems zu beeinflussen. In Folge der Fertigstellung einer solchen Anpassung oder Neu-Orchestrierung wird die Möglichkeit bestehen, die Orchestrierung anderen Personengruppen oder einzelnen Individuen zugänglich zu machen.

4.2 Nutzerbeteiligte Vorstudie

Als Ausgangsbasis der weiteren Konzeption beziehungsweise der Evaluation des Grundkonzeptes diente ein Participatory Design Workshop mit Mitgliedern der potentiellen Anwendergruppe von Endbenutzern. Angelehnt an PICTIVE [Mur'04] von Muller wurde ein Workshop Konzept entwickelt, bei dessen Durchführung die Teilnehmer möglichst unbeeinflusst Papier Prototypen von Notation, Modell, Repräsentation und Umgebung entwickeln sollten. Im Anschluss wurden in einer Analysephase implizite und explizite Informationen aus den Aufzeichnungen des Workshops extrahiert und konzeptionell in das SiSeOr Grundkonzept integriert. Das zuvor erläuterte SiSeOr Grundkonzept wurde während des Workshops genutzt, um im Bedarfsfall selektiv Anregungen und Beispiele einfließen zu lassen sowie gegenüber den im Workshop entwickelten Mock-ups als Vergleichsmöglichkeit zu dienen.

4.2.1 Grundlagen des Participatory Design Workshops

Das Hauptziel des Workshops war die Bildung einer ausreichenden Informationsbasis, um nach einer anschließenden Analyse das SiSeOr Konzept in Folge der Ergebnisse anzupassen, zu erweitern und zu verfeinern. Ein Teilziel war daher die Untersuchung der allgemeinen Verständlichkeit von Funktionskomponenten und deren Verwendung in Form einer visuellen Orchestrierung. Dies sollte unter anderem durch die gemeinsame Entwicklung einer passenden Begrifflichkeit für den inhaltlich abstrakt vermittelten Fachbegriff der Web Services bzw. der Web-Service-Operationen erfolgen. Ein weiteres Teilziel bildete die Ermittlung, wie sich Endbenutzer in einer möglichst unbeeinflussten Orchestrierungssituation verhalten, welche Bedürfnisse entstehen und in welcher Form diese zur Anpassung der Papierprototypen führen. Ferner sollte nach dieser Phase durch Vergleich und Diskussion von Teilnehmerkonzept und SiSeOr Grundkonzept eine Evaluation spezieller Aspekte erfolgen. Basierend auf den empirischen Untersuchungen von Lin und Shao über den Zusammenhang von PD und dem Erfolg des entwickelten Systems [LS'00] sollte während des Workshops eine grundlegende Einbeziehung der Arbeitspraxis sichergestellt sein, um so praxisnahe und verwertbare Ergebnisse zu liefern. Essentiell war eine Förderung intergruppaler Verständigungsmechanismen, um den nötigen Wissens- und Perspektivenaustausch herbei zu führen. Ein Punkt von besonderer Bedeutung war ein offenes und gleichberechtigtes Verhältnis zwischen Nutzern und Entwicklern, welches auch die möglichst geringe Beeinflussung der Nutzer mit einschloss. Zusätzlich muss der Ablauf des Workshops eine gewisse Struktur bieten, zeitgleich aber auch flexibel auf ad hoc entstandene Anforderungen anpassbar sein. Bezüglich der Atmosphäre sollte ein Mittelweg zwischen einer Arbeits- und einer Spielsituation angestrebt werden, um Produktivität und zeitgleich Motivation zu begünstigen. Hinzu kommen die Bedingungen, dass den Endbenutzern die Mock-up Materialien vertraut sein müssen und das Situationen, in denen der Nutzer verunsichert über den folgenden Bedienschritt ist (Breakdowns), unmittelbar zur Diskussion und Anpassung führen [Kyn'88]. Aufbauend auf den Erfahrungen der in Kapitel 3.2.1 genannten PD-Projekte konnten viele grundlegende Fehlerquellen bereits im Vorfeld weitestgehend ausgeschlossen werden. Um für einen möglichst reibungslosen Workshopverlauf zu sorgen, wurde zusätzlich im Vorfeld ein Probedurchlauf durchgeführt, in denen Studenten aus unterschiedlichen Fachdisziplinen die Rolle der Nutzer des späteren Workshops einnahmen. Wie erhofft konnten durch den Probelauf diverse Komplikationen oder Fehlannahmen, sowie eine

Vielzahl kleiner Mängel festgestellt werden, welche zwar weitestgehend ad hoc gelöst werden konnten, jedoch den Ablauf des Workshops wiederholt unterbrachen.

Konzept

Angelehnt an PICTIVE [Mul'92] wurde auf Grundlage der gesammelten Erkenntnisse und bestehenden Rahmenbedingungen das folgende Konzept entwickelt, dessen Ablaufstrukturierung größtenteils auf Erfahrungen des zuvor erwähnten Probedurchlaufs beruhen. Im Vergleich zu anderen Konzepten wurden hier ausschließlich die direkt betroffenen Personengruppen der Designer und Endbenutzer adressiert, welche während der Durchführung durch neutrale Moderatoren und Protokollanten unterstützt werden. Der Workshopeinstieg erfolgte über eine kurze verbale Beschreibung der aktuellen organisatorischen und technologischen Arbeitspraxis der Endbenutzer, gefolgt von einer gemeinschaftlichen Erarbeitung der akuten Mängel der entsprechenden informationstechnischen Unterstützung. Dies entspricht der Reflektion und Analyse der Nutzungssituation (use situation), welche von Kyng in *Users and computers: A contextual approach to design of computer artifacts* [Kyn'95] als idealer Ausgangspunkt eines Designprozess genannt wird. Nach einer beispielhaften Einführung abstrakt vermittelter Funktionskomponenten unter der Bezeichnung Web Service erfolgte die Erarbeitung und Begriffsbildung des gemeinsamen Verständnisses. Diese Definition stellte beispielsweise einen der in der Workshopsituation herrschenden Aspekt des *third space* Ansatzes von Muller [Mul'03] dar, bei dem der Kontext eines PD Workshops erst zwischen den beiden heterogenen Gruppen von Endanwendern und Designern ausgehandelt werden muss. Im Anschluss folgte die gemeinsame Entwicklung und schriftliche Fixierung von Szenarien, welche auf der bestehenden Arbeitspraxis der Endbenutzer beruhten. Diese dienten als Hilfestellung in der anschließend selbständig durchgeführten Erstellung entsprechender Funktions-Mock-ups, welche während der Orchestrierung mit Hilfe von Stiften, Notizzetteln und Scheren, an die aus der Situation entstehenden Bedürfnisse der Endbenutzer angepasst werden konnten. Diese den Endbenutzern vertrauten Materialien und Werkzeuge [Mul'91] dienten darüber hinaus der Schaffung und Anpassung einer entsprechenden Notation und Orchestrierungsumgebung. Die Designer hatten in dieser Phase eine auf die Technologie fokussierte Aufgabe, welche die Beantwortung auftauchender Fragen und die Kommunikation notwendiger technischer Grenzen umfasste. Um die hohe Informationsdichte dieses Workshops erfassbar zu machen, wurden nach vorheriger Absprache mit den Teilnehmern soziale Interaktionen sowie die Arbeit mit den Mock-ups per Audio- und Videoaufzeichnungsgeräten dokumentiert. Den Abschluss bildete ein von Muller konzipierter Fragebogen [Mul'92], der in diesem Fall eine Betrachtung der Workshopergebnisse aus Teilnehmerperspektive ermöglicht und dem Anhang dieser Arbeit beigelegt ist.

Materialien

Neben der wiederbeschreibbaren Arbeitsfläche, welche in Abbildung 19 skizziert ist, stand den Workshopteilnehmern eine Schultafel und eine interaktives Whiteboard (Smartboard) zur Verfügung, welche primär als kollaborativ nutzbare Medien zur schriftlichen Fixierung von Fragen, Begriffen und Daten diente. Zusätzlich lagen für alle Teilnehmer in ausreichendem Umfang Papierbögen bereit, um auch am Sitzplatz Notizen anfertigen zu können. Ein Videoprojektor (Beamer) diente zur Anzeige der workshopbegleitenden Folienpräsentation, welche den Benutzern ebenfalls in Form eines Informationsblattes zugänglich war. Aufgrund der Anforderung, dass die Materialien

der Mock-up Gestaltung den Nutzern vertraut sein müssen und eine flexible und umgehende Änderung zulassen, beschränkte sich die Auswahl auf nicht-permanente Filzstifte und selbstklebende Notizzettel. Alle Arbeitsmaterialen wurden bewusst in verschiedenen Farben bereitgestellt und konnten beliebig genutzt werden. Damit entsprachen die Materialen einer kontextspezifischen angepassten und erweiterten Variante der durch Muller beschriebenen Materialbedingungen von PICTIVE [Mul'93].

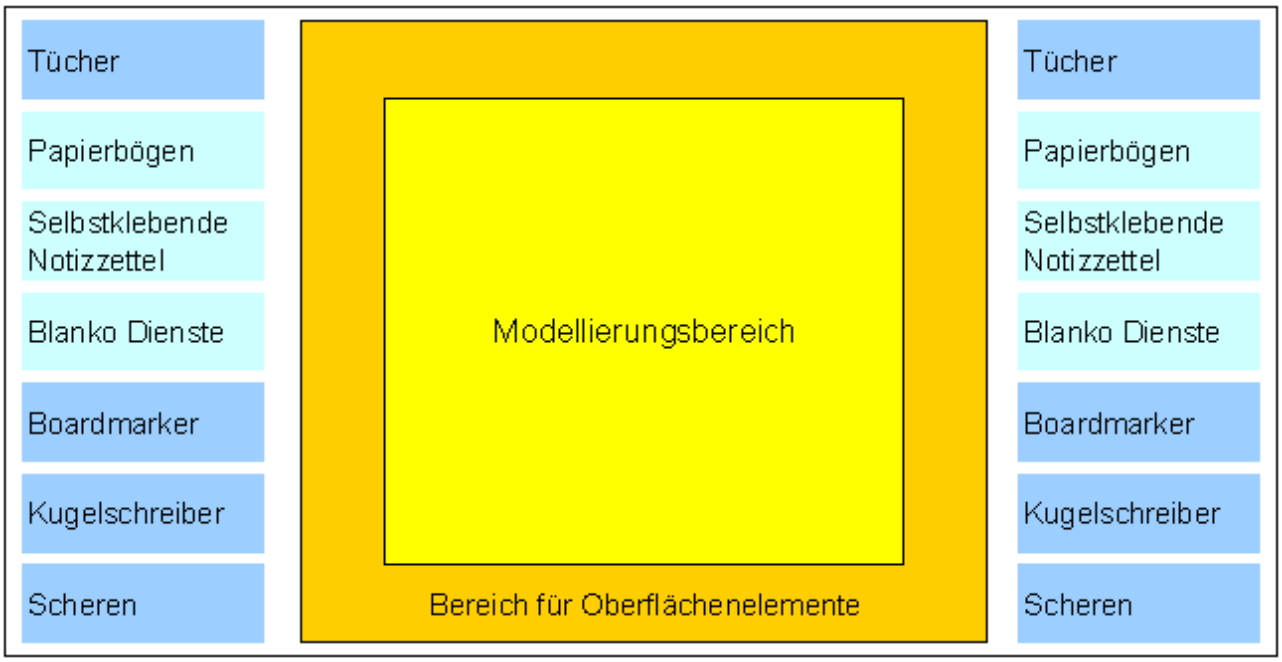

Abbildung 19: Aufbau der Arbeitsfläche

Durch diese Zusammenstellung sollte unter anderem in Kombination mit der freien Struktur innerhalb dieser Phase eine Atmosphäre zwischen Arbeit und Spiel entstehen. Hinzu kamen Papierbögen und Scheren, um eigene Darstellungen für Web Services und Oberflächenelemente zu entwerfen bzw. anzupassen. Die zentrale Arbeitsfläche bestand aus einer 1,00 m x 1,40 m messenden, widerbeschreibbaren Zeichenfläche, welche für die Darstellung des Orchestrierungsbereichs und der grafischen Benutzeroberfläche diente. Der gesamte Aufbau ist in Abbildung 19 skizziert, wobei der entsprechende Raum für Oberflächenelemente im Vergleich zum „Bildschirm“ nicht speziell hervorgehoben war.

4.2.2 Szenerie und Durchführung

Zeitlich war der Workshop auf einen Tag begrenzt und fand in einem Seminarraum der Universität Siegen statt, welcher über die zuvor angesprochene Schultafel, das interaktive Whiteboard und einen Videoprojektor verfügte. Das Zentrum des Raumes bildete ein kreisrunder Tisch, an dem alle Teilnehmer Platz fanden und an welchem alle Phasen des Workshops absolviert werden konnten. Die Abbildung 20 zeigt die Arbeitssituation der Endbenutzer in der Phase der Mock-up Bearbeitung. Die Teilnehmergruppe bestand aus einem Moderator und einem Co-Moderator, welche den Ablauf koordinierten, als neutrale Ansprechperson dienten sowie schriftliche Fixierungen an Tafel und interaktivem Whiteboard vornahmen. Eine wissenschaftliche Mitarbeiterin und ein wissenschaftlicher Mitarbeiter übernahmen die Aufgabe der Dokumentation, welche die Bedienung der

technischen Geräte (Kamera, Audiorekorder) und das stichwortartige Protokollieren wichtiger Kernpunkte des Workshops umfasste. Zwei Techniker dienten als Informationsquelle für technologische Fragen und gaben auf Nachfrage der Nutzer allgemeine oder spezielle Anregungen, die sich anhand des SiSeOr Grundkonzepte orientierten.

Abbildung 20: Arbeitsfläche und -material

Die Nutzergruppe setzte sich aus zwei Nutzerinnen (Person A und B) aus der Verwaltung und dem Leiter der IT-Abteilung (Person C) zusammen, welche Teil der Belegschaft eines mittelständischen Unternehmens mit ca. 150 Mitarbeitern sind. Hierbei entsprechen diese drei Personen dem zuvor in Kapitel 3.1.2 definierten Begriff des Endbenutzers. Zum Zeitpunkt des Workshops arbeitete Person A als Assistentin der Verkaufsleitung und erstellte mit Hilfe von Tabellenkalkulationen spezifische Analysen, welche kurzfristige Informationsbedürfnisse der Verkaufleitung adressierten. Person B war Abteilungsleiterin des Verkaufs und nutzt Tabellenkalkulationen für die Planung und Kontrolle der Einkaufswirtschaft. Person C erstellte unter anderem auf besonderen Wunsch von Mitarbeitern spezielle Sichten auf die Daten des ERP Systems, um die notwendigen Informationen für die Aufgabenbewältigung durch Tabellenkalkulationen zu erleichtern bzw. zu ermöglichen. Der Workshoptag, dessen Agenda dem Anhang beigelegt ist, war mit fünf Stunden angesetzt, musste jedoch einvernehmlich um ca. eine Stunde verlängert werden, da die verfügbare Zeit andernfalls nicht ausgereicht hätte. Den Auftakt bildete eine Vorstellungsrunde, welcher sich eine kurze Einführung in Ursprung, Vorgehen und Motivation eines PD Workshops anschloss. Um die Endbenutzer bereits in dieser Phase aktiv einzubeziehen, konnte jede Person Erwartungen und Ziele an den Workshop verbal skizzieren, auch um einen Vergleich für die Reflektion am Ende des Workshops zu besitzen. In Anlehnung des Vorgehens von Bødker, Grønbaek und Kyng [BGK'93] während des Utopia Projektes folgte die bereits erwähnte gemeinsame Szenarioerstellung. Dies sollte zur Erfassung, Analyse und kritischen Betrachtung der aktuellen Arbeitssituation dienen und allen Teilnehmern transparent machen. Dieses gesammelte Material stellte zusätzlich die Ausgangsbasis der anschließenden Einführung der neuen, auf funktionalen Komponenten basierenden Technologie dar. Diese Erschließung eines stark vereinfachten und abstrakten Web-Service-Konzeptes

bildete den Auftakt für eine kollaborative Verständnisbildung und Namensfindung. Anschließend wurde das Szenario in seine einzelnen funktionalen Komponenten aufgeteilt und so für eine Transformation in eine orchestrierbare Web-Service-Form vorbereitet. Diese Transformation sollten die Nutzer möglichst eigenständig durchführen, wurden jedoch auf Anfrage an einigen Stellen durch die Techniker unterstützt. Dieser Entwicklung schloss sich die Orchestrierung der zuvor erstellten Web-Service-Bausteine durch die Nutzer an, dessen Ergebnis in Abbildung 22 auszugsweise zu sehen ist.

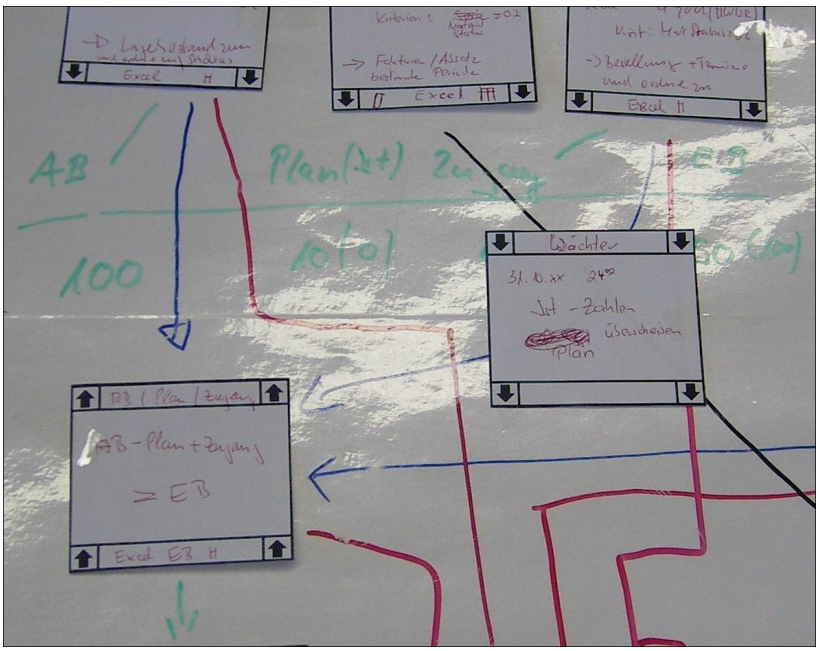

Abbildung 21: Orchestrierung des gewählten Szenarios

In dieser Phase wurden neue Funktionalitäten und eigene Notationsformen entworfen, welche jedoch durch die notwendigerweise vorgegebene Grundform eines Dienstes nicht vollständig unbeeinflusst war. Auf eine detaillierte Besprechung der produzierten Orchestrierung und der durch die Nutzer angepassten Mock-ups sowie dem Vergleich mit dem SiSeOr Grundkonzept, konnte aus Zeitgründen nicht mehr eingegangen werden, da zu diesem Zeitpunkt bereits das ad hoc verlängerte Zeitfenster überschritten wurde. Dennoch konnte eine kurze Diskussion bestehender Oberflächen-Mock-ups der Orchestrierungsumgebung in Hinblick auf ihre potentielle Zugänglichkeit und Eignung für die Teilnehmer durchgeführt werden.

4.2.3 Analyse

Die Auswertung stützt sich auf einen fünf Punkte umfassenden Fragebogen, Notizen, Protokolle, den Arbeitsbereich (Design-Space), Audiomitschnitte und Videomaterial. In Anbetracht der zeitlich sehr eingeschränkten Anwendung sind die hier durch Analyse gewonnenen impliziten und expliziten Erkenntnisse lediglich als unscharfe Orientierung zu verstehen, welche als sehr informativ, aber bei weitem nicht als repräsentativ angesehen werden können. Die im Folgenden vorgestellten Erkenntnisse gliedern sich nach ihrem chronologischen Auftreten im Workshop und werden an den entsprechenden Stellen durch zusätzliche Anmerkungen erläutert. Bereits während der Vorstellungsrunde und dem kurzen Statement der jeweiligen Arbeitspraxis wurde deutlich, dass die mangelnde Unterstützung des Datenaustausches zwischen Applikationen ein großes Hindernis dar-

stellt. Das eigentliche Problem bildete jedoch die mangelnde Flexibilität des ERP Systems, die durch eine individuell entwickelte von Tabellenkalkulationen ausgeglichen wurde. Von besonderer Bedeutung war in diesem Zusammenhang die aufwendige, unzureichende und inflexible Abfrage der Daten sowie deren nicht anforderungsgerechte Darstellung. Während der Erarbeitung eines dem Forschungsteam bereits aus einer älteren Studie bekannten Szenarios und der von den Teilnehmern vorgeschlagener Szenarien wurde die Komplexität und Vielschichtigkeit der Arbeitspraxis deutlich. Das im Vorfeld analysierte Informationsmaterial (Tabellenkalkulations-Dokumente, etc.) konnte nur sehr ungenügend die Anforderungen der Arbeitspraxis vermitteln, so dass hier noch einmal der Stellenwert des gegenseitigen Dialogs und des Verständnisses der Arbeitspraxis hervorgehoben werden muss. Die kooperative Analyse der Szenarien untermauerte den starken, bereits in der Anfangsphase identifizierten Bedarf der anwendungsübergreifenden Datenverfügbarkeit während der Durchführung eins Geschäftsprozesses. Die damalige Praxis musste aufgrund diesen Mangels, zu einem Grossteil auf eine manuelle Übertragung von Daten zurückgreifen. Der hierfür geschätzte Zeitbedarf verdeutlicht den ökonomisch suboptimalen Zustand und die nicht optimalen Arbeitsbedingungen.

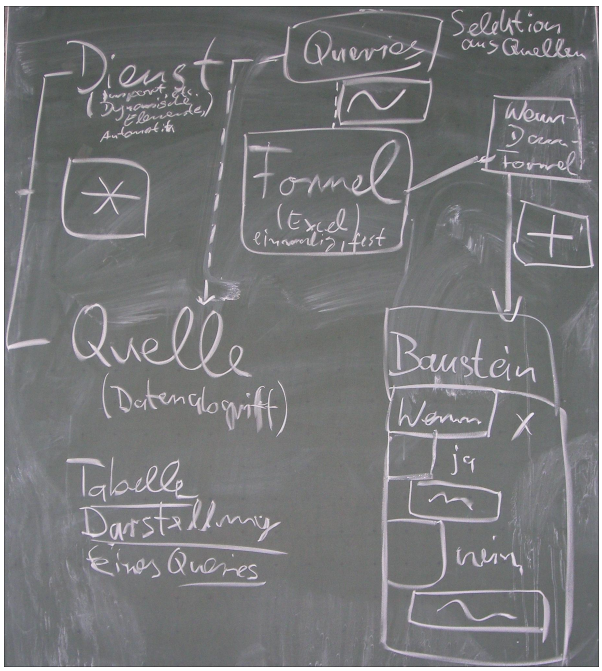

Abbildung 22: Erarbeitung des Technologie-Begriffs des *Dienstes*

Die zur Kommunikation als Boxen dargestellte Web-Service-Operationen wurden von den Benutzern umgehend verinnerlicht. Im kooperativen Definitionsprozess der Verständnisses eines Web Services bzw. dessen Operationen wurden an der Tafel die individuellen Erklärungsmodelle aller Teilnehmer (Dienst, Quelle, Baustein, Queries, Formel, Wenn-Dann etc.) in einen logischen und visuell verknüpften Zusammenhang gebracht (vgl. Abbildung 22). Im Hinblick auf die Zugänglich- und Verständlichkeit der allgemeinen Prinzipien der Web-Service-Technologie waren jedoch einige wenige Einschränkungen festzustellen. Da während des Workshops bewusst nur wenige Beispiele Verwendung gegeben wurden, griffen die Nutzer während der Transformation des Szenarios in funktionale Einheiten häufig auf bekannte Konzepte aus dem jeweiligen Softwarehorizont zurück. In diesem Zusammenhang setzte eine sich bereits in der vorherigen Phase des Workshops angedeutete Unterteilung der Funktionalitäten nach Datenquelle, Verarbeitung und Ausgabe fort, wel-

che den Nutzern allgemein die Handhabung zu vereinfachen schien. Unabhängig davon hatten jedoch weiterhin die funktionalen Grenzen von ERP- und Tabellenkalkulationssystemen Bestand, da lediglich der verbesserte Datenimport in die Tabellenkalkulation das Hauptmotiv zu sein schien. Im praktischen Teil der Orchestrierung wurde deutlich, dass ein durch Linien verknüpfter visualisierter Datenkanal zweifelsfrei verstanden wurde. Die bereits erwähnte logische Typisierung durch die Endbenutzer drückte sich visuell durch die Verwendung unterschiedlicher Farben bei der Verknüpfung der Mock-ups aus. Die Endbenutzer neigten zu Beginn dazu, einzelne Web-Service-Operationen funktional sehr umfangreich zu gestalten, da bis zu diesem Zeitpunkt die Möglichkeit der Wiederverwendbarkeit von Bausteinen noch nicht kommuniziert werden konnte, aber nach entsprechenden Hinweisen im Anschluss Berücksichtigung fand. Während der Orchestrierung diente eine Skizze des gewünschten Ergebnisses (in Form einer Tabelle) dazu, den Überblick in diesem relativ komplexen Szenario zu behalten. Im Hinblick auf die weiterführende Konzeption von SiSeOr können die folgenden Ergebnisse festgehalten werden.

4.2.4 Designerkenntnisse

Die schnelle Einarbeitung in Notation und Repräsentation der Web-Service-Operationen bzw. deren Orchestrierung bestätigt die Wahl dieser Darstellungsform und unterstützt deren Nutzung und Weiterentwicklung. Die Analyse zeigt deutlich den häufigen Rückgriff auf vorhandene Handlungsstrategien und Konzepte, welcher sich primär durch den Anwendungscharakter der Orchestrierung und eine starke Prägung der Tabellenkalkulations-Gedankenwelt erkennen lässt. Dies erfordert eine geeignete Berücksichtigung in SiSeOr, darf jedoch nicht den zentralen Charakter der Web-Service-Technologie und deren Vorteile verzerren. Vielmehr gilt es eine Hilfestellung zu entwickeln, welche, beispielsweise durch die Bereitstellung von aussagekräftigen Beispielen und Tutorialen oder durch Assistenzfunktionalität, diese grundlegenden Informationen möglichst frühzeitig vermitteln. Zeitgleich könnten hierdurch die im Workshop nur bedingt untersuchbaren Aspekte der Wiederverwendung adressiert werden. Zur Erleichterung der Orchestrierung sollte die beobachtbare Typisierung der Web-Service-Operationen und Nummerierung der Datenkanäle mit in die Konzeptionsüberlegungen einbezogen werden. Die individuelle Diversität der Nutzer wurde vor allen durch die unterschiedlichen Präferenzen im Hinblick auf das Vorgehen während der Orchestrierung deutlich. Das System muss die Individualität der Anwender durch umfangreiche Anpassungsmöglichkeiten an die verschiedenen Nutzerbedürfnisse berücksichtigen, welches sich auch in einer möglichst freien Reihenfolge der Orchestrierungsschritte niederschlägt. Hierbei könnte die während der Orchestrierung genutzte Übersichtskizze einen Hinweis darauf geben, dass ein Vorgehen ausgehend vom erwünschten Endprodukt einen besondern Stellenwert einnehmen könnte. Darauf aufbauend sollte der aktuelle Status der Orchestrierung bzw. dessen Ausgabe in einer Form präsentiert werden können, welche der Nutzer möglichst einfach mit dem angestrebten Ergebnis vergleichen kann. Weiterhin wurde die von Pane im Rahmen natürlicher Programmiersprachen und -umgebungen [PM'06] bereits festgestellte Tendenz zu zeit- und damit ereignisabhängigen Konstrukten deutlich, welche im Konzept ebenso berücksichtigt werden muss. Im Hinblick auf die präsentierten Oberflächenelemente war deren Gestaltung grundsätzlich verständlich und auftretende Fragen schienen hauptsächlich aus mangelnder Vertrautheit mit der Technologie und der Orchestrierung zu resultierten. Grundsätzlich bezogen sich die Endbenutzer innerhalb der

Diskussion über einzelne Elemente der Orchestrierungsumgebung samt deren Funktionalität häufig auf Aspekte bereits bekannter Software, so dass hier bei der Konzeption des Orchestrierungswerkzeuges angeknüpft werden könnte.

4.3 SiSeOr: Konzeptrahmen

Allgemein wird menschliches Handeln durch die Verfolgung von Zielen geleitet, zu deren Erreichung unter anderem Softwarewerkzeuge eingesetzt werden. Dabei entscheidet auch eine durchaus subjektive Gegenüberstellung von Aufwand und Nutzen über die individuelle Motivation zum Gebrauch oder Nicht-Gebrauch und damit zwangsläufig auch über gute und schlechte Software. Dieser Zusammenhang von Aufwand und Nutzen kann in verschiedenen Forschungsarbeiten, in unterschiedlicher Form und Ausprägung wiedergefunden werden, wie beispielsweise bei Sutcliffe et. al [SLM'03] oder Wulf und Golombek [WG'01]. Hierauf aufbauend stellen die Minimierung beziehungsweise Maximierung dieser beiden Faktoren die prinzipiellen Grundanforderungen dar, welche jedoch lediglich unter Berücksichtigung relevanter Aspekte der individuellen, allgemeinen und arbeitspraktischen Ausgangslage der Anwender erfolgen kann. Abbildung 23 zeigt die im weiteren Verlauf der Konzeptbeschreibung adressierten Faktoren, welche durch eine entsprechende Gestaltung ein mindestens ausgeglichenes Verhältnis von Aufwand und Nutzen herstellen sollen.

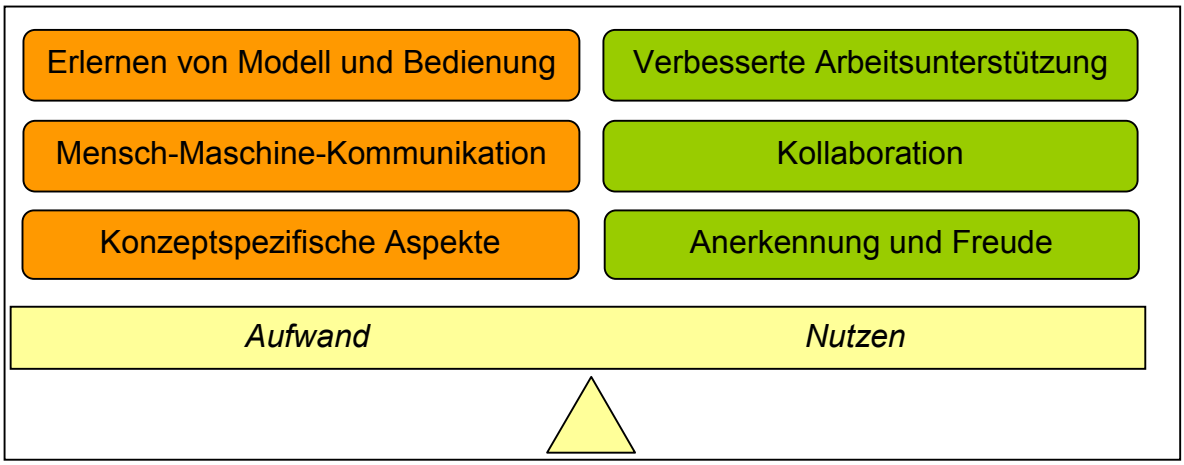

Abbildung 23: Faktoren zur Aufwandsreduzierung und Nutzenerweiterung

Basierend auf der motivierten ökonomischen Notwendigkeit des Tailoring ist die konkrete Zielsetzung von SiSeOr die Befähigung von Endbenutzern zur Anpassung und Erstellung in BPEL informationstechnisch abgebildeter Geschäftsprozesse. Anders ausgedrückt soll ein Bindeglied zwischen Fähigkeiten, Wissen und Lernpotential der Endbenutzer, dem Paradigma der serviceorientierten Architektur und der technisch geprägten textuellen Orchestrierungssprache BPEL geschaffen werden. Im Kontext der Softwareevolution [MEC'99], hier in Form des Tailoring, sind diese Faktoren besonders durch die Leichtigkeit der Anpassung und der hierdurch erreichbaren verbesserten Unterstützung der jeweiligen Arbeitspraxis geprägt. Demzufolge ist die Kernanforderung die Aufwandsminimierung, welche jedoch nicht maßgeblich zu Lasten der Mächtigkeit und Flexibilität der Orchestrierungen gehen darf, wie es beispielsweise in Kapitel 2.1.3 betrachteten Mashup-Editoren noch der Fall ist. Dabei muss ein neues, weniger technisches und komplexes Verständnis für Web Services innerhalb des Editors kommuniziert werden, um konzeptionsbeding-

te Verständnisprobleme während des Einstiegs und der ersten Nutzung zu beseitigen. Allgemein muss SiSeOr grundsätzliche Usability Aspekte berücksichtigen, welches durch einer Orientierung an Nielsens Grundregeln zur Erzielung einer ausreichenden Usability [Nie'93] erfolgt. Neben einer einfachen Erlernbarkeit und effizienten Nutzbarkeit wird dessen Ausprägung in Nielsens Definition unter anderem durch die Möglichkeit eines einfachen Erinnerns, der Minimierung der Fehlerquellen und eine nutzerindividuelle gefällige Erscheinung bestimmt. Ergänzend hierzu wird an relevanten Stellen der nun folgenden Konzeption selektiv auf die entsprechenden Dimensionen des in Kapitel 3.2.6 beschriebenen Frameworks der „cognitive dimensions of notations“ und der Aspekte der von Bødker erweiterten Aktivitätstheorie (vgl. Kapitel 3.2.7) verwiesen. Hintergrund der Nutzung dieser verschiedenen Orientierungshilfen bildet die Anforderung, dass besonders im Bereich des EUD nicht die Erfahrungen der Designer als alleiniger Maßstab für Usability von Programmiersprache und Entwicklungsumgebung verwendet werden dürfen [Bla'06a] und bereits vor der eigentlich maßgeblich Evaluation grundlegende Faktoren berücksichtigt werden müssen.

4.4 SiSeOr: Aufwandsminimierung

Der Aufwand, welcher als Summe der Investition von Zeit und Mühe verstanden werden kann, um eine Tailoringvorhaben zu realisieren, setzt sich primär aus der Aneignung von notwendigem Bedienungswissen und der wechselseitigen Kommunikation zwischen Anwender und System zusammen [SLM'03]. Hierbei ist der erforderliche Lernaufwand für eine grundlegende Nutzung von dem zur Erarbeitung aufbauender Nutzungsfacetten zu unterscheiden, welche beispielsweise den eigenen Handlungsumfang, die Flexibilität und die Mächtigkeit der Anpassung erweitern. Die wechselseitige Kommunikation zwischen Endbenutzer und System setzt sich dagegen aus der Leistung zusammen, ein mentales Modell kognitiv in die vom Tailoringsystem vorgegebene Form zu transformieren sowie alle dazu notwendigen Schritte, dieses in das System einzugeben, es dort zu testen, von Fehlern zu befreien und zu nutzen. Zusammengefasst kann man an dieser Stelle vom Ziel einer möglichst einfachen oder aufwandsarmen Nutzbarkeit sprechen.

4.4.1 Aneignung der Bediengrundlagen

Im Rahmen des Tailoring stellt der Lernaufwand für den Endbenutzer eine Vorleistung dar, welche durch den später zu erwartenden Nutzen motiviert ist [SLM'03] und auf die von Nardi und Miller definierten Endbenutzertypen [NM'91] als Nutzungshürde einen unterschiedlich starken negativen Einfluss [GN'92] hat. Um eine bestehende Grundmotivation zu nutzen und aufrecht zu erhalten, muss die Komplexität und damit das erforderliche Einstiegswissen möglichst gering gehalten werden, um dem Nutzer schnellstmöglich einen konkreten, wenn auch funktional begrenzten Nutzen zu bieten. Beispielsweise bieten Tabellenkalkulationsprogramme durch ihr flaches Einstiegswissen diese von Nardi geforderte Einfachheit der Nutzung [Nar'93]. Darüber hinaus kann die von de Ruyter und van den Sluis im Rahmen von EUD in intelligenten Umgebungen getätigte Aussage [DV'06], dass jede Erhöhung der Anpassungskomplexität die Anzahl potentieller Nutzer reduziert, auf diesen Kontext übertragen werden. In SiSeOr ist der Lernaufwand zur Aneignung dieses Einstiegswissens besonders vom Verständnis des grundlegenden Modells einschließlich der Notation und Repräsentation eines vorhandenen Prozesses und dessen Elementen abhängig. Eine maß-

gebliche Reduzierung des Lernaufwandes muss daher an diesem Punkt ansetzen, welcher als Grundstein für spätere Anpassungen dient.

Grundmodell

Die Basis hierzu bildet eine maßgebliche Abstraktion und Komplexitätsreduzierung technischer Gesichtspunkte. Wie im grundlegenden Konzept beschrieben, werden in SiSeOr dem Endbenutzer diesbezüglich Web-Service-Operationen als eigenständige Funktionalitäten präsentiert, welche über datentypenabhängige Verknüpfungspunkte visuell orchestriert werden können. Die lediglich unter technischen Aspekten relevante Zwischenebene der Web Services bleibt dem Nutzer somit verborgen. Die aufgrund der Ergebnisse des Workshops (vgl. Kapitel 4.2) als *Dienst* bezeichneten Konstrukte können primär in Form eines Datenflusses zu einem *Prozess* zusammengeschlossen werden, welcher einen bestehenden Geschäftsprozess informationstechnisch abbildet und partiell oder vollständig automatisiert ablaufen kann. Ein solcher mit Hilfe von BPEL formulierter Prozess ist wiederum als Dienst aufrufbar und ermöglicht somit eine beliebige komplexe Verschachtelung. Dienste und Prozesse stehen dem Nutzer in Form eines UDDI Repository mit der Bezeichnung *Zentralarchiv* zur Verfügung, welches durch den Import von WSDL Dateien, bezeichnet als *Dienstbeschreibung*, erweitert werden kann. Die Begrenzung auf diese elementaren Bestandteile bietet neben dem leichteren Verständnis eine deutliche Vereinfachung der Orchestrierung, da Ausführungsreihenfolge und damit der Effekt eines Prozesses ausschließlich durch das Verbinden von Verknüpfungspunkten ausgewählter Dienste festgelegt werden. Darüber hinaus kann die von Brooks [Bro'87] spezifizierte essentielle Komplexität durch die inkrementelle Entwicklung und die mögliche Verschachtelung (vgl. Abstraction Gradient [GP'96]) reduziert werden, da die Modellierung auf verschiedenen Detailstufen beliebig fokussiert durchgeführt werden kann.

Programmierparadigma

Zusätzlich entscheidend für den Lernaufwand ist die Darstellungsform der Dienste und Prozesse. Basierend auf den Erkenntnissen des aktuellen Forschungsstandes und den Ergebnissen der Vorstudie, löst sich SiSeOr von textuellen Darstellungen der Sprachen BPEL oder SSCL [GMS'06] und ermöglicht eine visuelle Orchestrierung auf Grundlage einer auf der *Box und Wire* Metapher basierenden Notation und Repräsentation. Diese häufige verwendete Metapher (vgl. Kapitel 2.1.2, 2.2.4) nimmt eine ausgeglichene Position zwischen Konkretem und Abstraktem ein und gewährleistet in Kombination mit dem Gedankenbild des Datenflusses ein, wie es die Ergebnisse des PD Workshops gezeigt haben (vgl. Kapitel 4.2.3), insgesamt verständliches grafisches Grundmodell. Weiterhin ausschlaggebend für diese Entscheidung ist die in einer Textform nur schwer erfassbare logische Struktur einer Orchestrierung sowie die Vorteile der besseren Übersichtlichkeit und der effektiven Aufnahme visueller Informationen durch das Gehirn, wie beispielsweise bestimmter Schlüsselinformationen [NC'01]. Darüber hinaus sprechen weitere Punkte für diese grafische Form, welche in Kapitel 3.2.2 ausgeführt wurden.

Dienst- und prozessspezifische Informationen

Auf Grundlage dieses Einstiegswissens sind Endbenutzer prinzipiell in der Lage, die Zusammensetzung von Prozessen und das Zusammenwirken der einzelnen Dienste zu verstehen. Jedoch

sind hierfür aussagekräftige und verständliche Informationen über die Dienstfunktionalität erforderlich, welche über die reinen Operationsnamen und die Typen der Datenports hinausgehen (vgl. Escape from Formalism [GP'96]). Grundsätzlich beinhalten Schnittstellenbeschreibungen Dokumentationsmöglichkeiten, welche Entwickler jedoch nur lückenhaft nutzen und diese meist ausschließlich technische Aspekte beinhalten. Darüber hinaus fehlen Informationen, welche eine gezielte Suche innerhalb einer Menge von Diensten, beispielsweise im *Zentralarchiv* ermöglichen. Konform zum WSDL Standard wird dieser Informationsbedarf durch eine Erweiterung der Schnittstellenbeschreibungen um zusätzliche Metadaten erreicht, welche einer Bearbeitung durch Endbenutzer offenstehen. Neben einer nicht-technischen Beschreibung der Dienstfunktionalität, in deren Rahmen auch konkret auf etwaige Anforderungen der Datenports eingegangen werden kann, existieren zusätzlich automatisch generierte Inhalte, wie beispielsweise der Zeitpunkt der letzten Ausführung. Weiterhin können zur Unterstützung der Verdeutlichung abstrakter Funktionalitäten und Vorgänge dieser Beschreibung, dem Endbenutzer bekannte symbolhafte oder textuelle Metaphern hinzugefügt werden, wie sie beispielsweise Nielsen [Nie'93] aus allgemeinen Usability Gründen vorschlägt. Durch die Gefahr von fehlerhaften Interpretationen und falschen Beziehungsannahmen zwischen realem Objekt und der durch die Metapher dargestellten Funktionen (vgl. Cobol Effekt [Bla'06b]), dienen diese jedoch nur als Unterstützung einer ausführlicheren textuellen Beschreibung. Zeitgleich wird sowohl im Modell als auch in der Umgebung auf den Einsatz arbiträrer Icons (vgl. [Nie'93]), welche auf reinen Konventionen beruhen, verzichtet. Ausgehend von der Bedeutung und dem Einfluss kollaborativer Faktoren in der allgemeinen und computerunterstützten Arbeitspraxis werden zusätzliche Typen von Metadaten bereit gestellt, welche, unter anderem durch Bewertungsmöglichkeiten und Schlagwörtern, die spezifische Suche und Auswahl von Diensten unterstützen. Anzumerken ist, dass die hier gemachten Aussagen aufgrund der Verschachtelungsstruktur ebenso für Prozesse und deren Schnittstellenbeschreibung zutreffen. Dabei wird der Stellenwert einer möglichst optimal unterstützten Suche dadurch unterstrichen, dass Endbenutzer während der Einarbeitung häufig auf vorhandene Tailoringdokumente anderer Nutzer zugreifen, um diese zu nutzen oder gegebenenfalls anzupassen, wie es beispielsweise bei Gantt und Nardi festgestellt wurde [GN'92].

Bedienungswissen

Im Hinblick auf die eigene Handlungsfähigkeit und die Orientierung innerhalb der Umgebung können generalisierte Erfahrungen mit anderen Softwareprodukten genutzt werden, um bereits bestehendes Bedienungswissen zu übertragen. Generelle Annahmen über Erfahrungen der Nutzer können zwar nur in sehr begrenzten Umfang getätigt werden, dennoch werden bestimmte verbreitete Konventionen, sowohl auf der funktionalen Ebene als auch beim Design der Benutzeroberfläche im System aufgegriffen. Nach dem in Kapitel 3.2.7 formulierten Handlungsmodell, welches auf der Arbeit von Bødker beruht, begünstigt SiSeOr die Nutzung vorhandener Operationen aus anderen Softwarekontexten. Die unterbewusst und damit automatisiert ausgeführten Operationen werden durch die subjektiv wahrgenommenen Rahmenbedingungen ausgelöst, so dass diese situativen Faktoren in SiSeOr eine möglichst hohe Ähnlichkeit aufweisen. Beispielsweise werden wichtige Funktionen über eine Symbolleiste verfügbar sein oder die Möglichkeit bestehen, Bedienschritte rückgängig zu machen, so dass eine rudimentäre Orientierung für den Nutzer ermöglicht wird. Darüber hinaus wird die Bedienung eine möglichst einfache Bildung von neuen Operationen

erlauben. Auf Grundlage dieses grundlegenden Bedienungswissens und einer anleitenden Hilfestellung (vgl. den Abschnitt *Aneignungsformen des Bedienwissens* innerhalb dieses Kapitel) durch das System ist der Endbenutzer prinzipiell in der Lage, einen Dienst oder einen Prozess zu suchen, auszuwählen, zu verstehen und auszuführen. Für die Anpassung oder Erstellung ist jedoch ein zusätzlicher Lernaufwand erforderlich, welcher sich primär aus der Komplexität und dem Umfang der Anpassung ergibt. Die hier realisierte niedrige Grundkomplexität der Orchestrierung resultiert aus dem Anspruch, im EUD Kontext eine möglichst einfache und komplexitätsarme Form der Programmierung zu ermöglichen, um eine möglichst große Anzahl potentieller Nutzer zu adressieren und die negativen Effekte auf die Grundmotivation zu vermeiden.

Komplexitätsstufen

Zur Vereinfachung der Orchestrierung unterstützt SiSeOr das Tailoring auf verschiedenen Komplexitätsniveaus, welches in der EUD Forschung in verschiedenen Formen propagiert wird und beispielsweise *FreEvolve* (vgl. Kapitel 2.2.4) genutzt wird. Im Kontext von SiSeOr würde die einfachste Anpassungsform auf Grundlage eines Prozesses erfolgen, dessen Funktionen und eingebundenen Dienste inhaltlich nah an der Domäne des Nutzers positioniert sind, beispielsweise durch das Entfernen eines Dienstes, welcher das Ergebnis einer Berechnung an eine Emailadresse versendet. Abhängig vom Umfang und der Zielsetzung des Anpassungsproblems sowie der Nutzung eher technisch orientierter Dienstfunktionalitäten kann diese Komplexität beliebig zunehmen. Für den Nutzer steht daher die Möglichkeit offen, Dienste und Prozesse unterhalb eines bestimmten technischen Grades auszublenden. Dieser drückt die logische Nähe einer Funktionalität zur technischen Ebene bzw. die Distanz zu Domänenebene aus und wird auf Grundlage einer Einteilung (in der Benutzeroberfläche durch einen Schiebregler dargestellt) durch die Nutzergemeinschaft festgelegt. Hierdurch können beispielsweise unerfahrene Nutzer eine begrenzte Menge praxisnaher Komponenten nutzen, um erste Erfahrungen mit SiSeOr zu sammeln, ohne durch die Komplexität technisch orientierter Funktionalitäten überfordert und demotiviert zu werden.

Lernmotivation

Um die zu Beginn der Einarbeitung vorhandene Lernmotivation längerfristig aufrecht zu erhalten, werden motivationshemmende Aspekte, wie beispielsweise Komplexitätssprünge auf der Lernkurve vermieden und zeitgleich motivationsfördernde Maßnahmen integriert. Hierfür maßgeblich ist die Form und die Granularität, in welcher das notwendige Wissen hinzugewonnen werden kann und wie dieser Lernprozess vom System geleitet und unterstützt wird. Basierend auf dem Ansatz des *gentle slope of complexity* [MCL'90] steigt die Lernkurve, also das Verhältnis von Fähigkeiten und der hiermit realisierbaren Tailoringmächtigkeit möglichst stufenlos und ohne Komplexitätssprünge an. Durch feingranulare Lerneinheiten können auch kleine Lernschritte zu einer direkten Vergrößerung des eigenen Handlungsspielraums und damit zu einer Ausweitung der Nutzenstiftung des Editors führen. Durch dieses Prinzip des nahezu direkten realisierbaren Gegenwertes eines investierten Lernaufwandes kann zeitgleich die Lernmotivation positiv beeinflusst werden. Für diese muss jedoch nach Repenning und Ioannidou [RI'06] ebenfalls ein ausgeglichenes Verhältnis von Fähigkeit und Herausforderung eingehalten werden, so dass die Lernkurve zwar möglichst flach ansteigt, jedoch beim Nutzer keine Unter- oder Überforderung eintritt.

Aneignungsformen des Bedienwissens

Zur Wissensvermittlung werden allgemein klassische Dokumentationsinhalte verwendet, welche die Programmnutzung kommunizieren und kurzfristige Hilfestellung bei einem konkreten Bedienproblem geben sollen. Im Hinblick auf die Vermittlung des Bedienungswissens stellt sich diese Form jedoch als wenig sinnvoll heraus, da Nutzer laut Nielsen [Nie'93] häufig das Lesen solcher klassischen gedruckten oder digitalen Handbücher vermeiden. Kernmethodik zur Vermittlung des Einstiegswissens und aufbauender Nutzungsmöglichkeiten bilden im Konzept von SiSeOr primär Tutoriale, welche anhand domänenspezifischer Szenarien und Prozesse interaktiv die Bedienung des Editors darlegen. Diese Methodik unterstützend wird zusätzlich eine breite Auswahl gut dokumentierter Beispielprozesse (u.a. von empfohlen von Mackay [Mac'90]) und Bildschirmvideos von grundlegenden Bedienschritten bereitgestellt, welche hiermit auch ein individuell gestaltbares Lernen ermöglichen. Im Hinblick auf Granularität der Lernunterstützung kann die jeweilige Herausforderung individuell festgelegt werden, indem jederzeit zwischen abgestuften Schwierigkeits- oder besser Komplexitätsstufen gewählt werden kann, welche beispielsweise in kompakt, ausgewogen und ausführlich unterteilt werden könnten. Für Endbenutzer, die eine eher traditionelle Dokumentations- und Lernform bevorzugen, liegen weiterhin wiki-basierte Hilfesysteme vor, welche ebenfalls durch die Nutzergemeinschaft editier- und erweiterbar sein sollten. In Kombination mit einer möglichen automatischen Ermittlung der aktuellen Nutzungssituation könnten somit Breakdown Situationen von Nutzern dokumentiert werden. Diese Breakdowns sind charakterisiert durch eine für den Anwender unerwarteten Zustand während der Nutzung eines Programms und erfordern ein kognitiv bewusstes Nachsinnen über die weiteren Handlungsschritte. Eine derartige Kombination von automatischer Zustandsbestimmung und durch die Nutzer beeinflussbaren Hilfestellung, würde zu einer situationsspezifischen und direkten Hilfe führen. Zur sporadischen Erweiterung des Nutzungshorizontes werden darüber hinaus bestimmte Systemaspekte nach dem „Kennen Sie schon...“-Prinzip vermittelt, welche jedoch entgegen der klassischen Verwendung nicht Form eines Popups erfolgt und somit keine Fokusverschiebung beim Nutzer erzwingt (vgl. Kapitel 3.2.7). Ergänzend hierzu wäre eine menschliche Hilfestellung über die Bereitstellung eines für die Domäne passenden Kommunikationskanals, wie etwa beispielsweise von Chaträumen sinnvoll. Diese muss jedoch dem Anspruch genügen, die jeweilige Nutzungssituation mit wenig Aufwand zwischen den Kommunikationspartnern übermitteln zu können. Durch die genannten Maßnahmen werden einige der grundlegenden Ideen der durch Pipek [Pi'05] aufgezeigten Unterstützungsmöglichkeiten adressiert, welche sich aus dem Aneignungsaktivitäten der Tailoringnutzer ergeben. Neben dem eigentlichen Tailoring (Teil der „usage modification“) umfassen diese Aktivitäten die zeitlich vorgelagerte Kommunikation („usage communication“) und Wahrnehmung („usage observation/visualisation“) neuer Werkzeugnutzungsformen.

4.4.2 Mensch-Maschine-Kommunikation

Die Anstrengung im Verlauf der Mensch-Maschine-Kommunikation bildet den zweiten ausschlaggebenden Faktor, welcher den Aufwand zur Nutzung der Tailoringumgebung maßgeblich beeinflusst und sich unter anderem aus dem kognitiven Transformationsaufwand zur wechselseitigen semantischen Umwandlung von Systemmodell und dem mentalen Modell zusammensetzt. Pane und Myers [PM'06] folgend, soll SiSeOr durch eine größtmögliche Gleichschaltung von Program-

miersprache und der Art wie Menschen über Aufgaben nachdenken eine einfachere Lern- sowie Nutzbarkeit realisieren. Für diesen Zweck greift SiSeOr unter anderem auf Forschungserkenntnisse der Bereiche Psychologie und Natürlicher Programmierung zurück, um Entwicklungsumgebung und Modell allgemeinen menschlichen Charakteristiken und den hieraus resultierenden Anforderungen anzugleichen. Darüber hinaus werden diese unter anderem durch die von Poswig [Pos'96] herausgearbeiteten Gestaltungsempfehlungen im Kontext visueller Empfehlung ergänzt und erweitert.

Gestaltung von Repräsentation und Notation

Für den Aufwand der Übertragung ist primär die mentale Repräsentation und Notation ausschlaggebend, da laut Pane und Myers [PM'06] die Distanz zu der im System verwendeten Form der Programmierung maßgeblich den kognitiven Transformationsaufwand beeinflusst. Hierbei kann kein detailliertes und pauschal gültiges Modell formuliert werden, da der individuelle Entwicklungsprozess einer mentale Notation und deren Repräsentation von Erfahrung, Ziel der Aufgabe, Länge des Programms und den Charakteristiken der Programmiersprache abhängig sind [NC'01]. Laut Berti et al. stellen Symboliken und Metaphern die primären Komponenten kreativen Denkens dar [BPS'06] und sind somit als Grundelemente der Problemlösung einer aufgabenspezifischen Notation und Repräsentation zu betrachten. SiSeOr versucht daher eine Annäherung von Problem- und der Programmierdomäne zu erreichen, um einen positiven Effekt bei Verständnis und Lösungsprozess zu realisieren (vgl. Closeness of Mapping, [GP'96]). Auch Nardi [Nar'93] und Costabile et al. [CFM'06] kennzeichnen eine Nähe von Programmiersprache und Sprache der Praxis als wichtigen Faktor in EUD Systemen, welche zu einer Reduzierung des Lernaufwandes und einer Erhöhung der Motivation führen können. Die genannte Anforderung der Ausrichtung an der zu lösenden Aufgabe oder Aspekten der Arbeitspraxis kann unter dem Begriff der Domänenorientierung zusammengefasst werden. Aus diesen Gründen verfügt SiSeOr über eine flexible und umfangreiche Konfigurationsmöglichkeit, welche eine domänenspezifische Anpassung im Vorfeld der Einführung, und im Bedarfsfall während des Einsatzes des Systems, erlaubt. Maßgeblich ist hierfür in erster Linie die Verwendung semiotischer Faktoren zur punktuellen Anpassung der visuellen Programmiersprache, wie es beispielsweise durch die Nutzung eines bestimmten Fachvokabulars oder entsprechender Metaphern möglich ist. Darüber hinaus kann die allgemeine Metapher des Datenflusses samt deren Repräsentation in eine funktionell analoge aber domänenspezifischere Form überführt werden. Beispielsweise könnten beim Einsatz in einem Logistikunternehmen über Analogien kommuniziert werden, indem Web-Service-Operationen als Fabriken mit bestimmten Produkten dargestellt werden, deren Abarbeitungsreihenfolgen über die Routen zu Wareneingängen weiterverarbeitender Fabriken gesteuert werden.

Aufgrund der starken Diversität von Endbenutzern kann jedoch eine Anpassung auf Ebene der Domäne nicht die personenbezogenen Unterschiede berücksichtigen. Diesem Punkt wird durch eine möglichst umfassende individuelle Anpassbarkeit an die eigenen Bedürfnisse begegnet. In diesem Zusammenhang stellt sich Frage, ob und zu welchen Anteilen dieselbige automatisiert, also adaptiv erfolgen soll. Gegen eine allgemeine Adaptivität spricht die Aussage von Kyng [Kyn'95]: “*Computers systems are tools, and need to be designed to be under the control of the people using them.*” Lieberman et al. [LPK'06] teilen diese Ansicht, indem sie vorbringen, dass die

endgültige Entscheidungsgewalt beim Nutzer liegen soll. Dennoch verweisen sie auf den möglichen positiven Einfluss eines umsichtigen Einsatzes von Adaptivität. Klann et al. [KEO'03] integriert diese beiden Aspekte, da in deren adaptiven System ein Nutzer in letzter Instanz die Entscheidungsgewalt besitzt und durch eine stärkere Verzahnung von Adaptivität und Anpassbarkeit eine beidseitige positive Beeinflussung erreicht wird. Eine Berücksichtigung dieser Aspekte in SiSeOr erscheint reizvoll, kann jedoch innerhalb dieser ersten Konzeption noch nicht erfolgen, da hierfür zumindest grundlegende Erfahrungen über den praktischen Einsatz von SiSeOr vorliegen müssen. Neben dieser Anpassbarkeit umfasst SiSeOr diverse Möglichkeiten, um bestimmte Editorfunktionen auf unterschiedliche Art auszuführen, beispielsweise durch bestimmte Tastenkombinationen, die erfahrenen Nutzern eine schnellere Bedienung erlauben [Nie'93] und somit auch die Operationalisierung (vgl. Aktivitätstheorie in Kapitel 3.2.7) erleichtert.

Charakteristiken kognitiver Modelle

Neben den zuvor erläuterten Maßnahmen versucht SiSeOr zusätzlich die allgemeine Flexibilität und Vielfältigkeit kognitiver Modelle in das Systemmodell zu übertragen. Die nach eigenen Aussagen nicht repräsentativen Untersuchungen von Blackwell und Petre [BP'99] über kognitive Modelle ergaben beispielsweise das Merkmal, die Dynamik eines laufenden Modells beliebig anhalten zu können. Konkret wird diesem Aspekte in SiSeOr durch die von Repenning und Ioannidou [RI'06] vorgeschlagene schrittweise Ausführung von Prozessen begegnet. Weiterhin wurden im Kontext der Vorstudie (vgl. Kapitel 4.2) die Erkenntnisse [MK'05; PM'06] bestätigt, dass unbeeinflusste Nicht-Programmierer häufiger auf ereignisbasierte Steuerungsmechanismen zurückgreifen, so dass die primäre Datenflussorientierung diese nicht kategorisch diese ausschließen darf. Eine Annäherung wird im System durch die in Kapitel 4.5.1 beschriebenen Web-Service-Operationen erreicht. Das des Weiteren festgestellte Charakteristikum einer anpassbaren Granularität der Abstraktion [BP'99] wird durch eine möglichst unkomplizierte Navigierbarkeit innerhalb der verschiedenen logischen Ebenen eines verschachtelten Prozesses umgesetzt. Dies schließt bei einem konkreten Dienst oder Prozess auch die Informationen der vorhandenen „besteht aus“ und „wird genutzt in“ Beziehungen ein, um Auswirkungen eine Änderung transparent zu machen und ein tieferes Verständnis über die funktionalen Teilkomponenten gewinnen zu können. (vgl. Abstraction Gradient und Hidden Dependencies [GP'96]). Obwohl keine Notation in allen Nutzungssituationen die optimale Darstellungsform bieten kann [GP'92], verzichtet SiSeOr auf eine Auswahl von unterschiedlichen Bearbeitungs-Sichten mit verschiedenen Informationsschwerpunkten, wie sie beispielsweise in FreEvolve realisiert wurden [WSW'06]. Durch diese Designentscheidung soll der Aufwand der Auswahl einer situationsabhängig passenden Sicht und der notwendigen kognitiven Neuorientierung (vgl. Aktivitätstheorie in Kapitel 3.2.7) beim Wechsel der Sichten vermieden werden. Lediglich zum Zweck der Prozessausführung und dessen Visualisierung verfügt SiSeOr über eine separate Ablaufumgebung und eine entsprechenden Sicht. Diese verwendet jedoch eine nahezu identische Notation und Peripherie, um der unter anderem von Lieberman et. al [LPK'06] formulierten Anforderung eines minimalen Orientierungsaufwandes für den Endbenutzer zu entsprechen. Die Umgebung bietet ebenfalls die Möglichkeit eines schrittweise durchgeführten Ausführungsmodus, um durch die Anzeige von Zwischenergebnissen die Fehlersuche zu vereinfachen. Grundsätzlich erfolgt die Entwicklung eines mentalen Modells evolutionär - die einzelnen Entwicklungsschritte finden in keiner festen Reihenfolge statt [RI'06], so dass nicht zusammen-

hängende Teile des Modells entstehen können, deren Detailgrad und Granularität zusätzlich stark voneinander abweichen können. Trotz dieser von Blackwell und Petre in deutscher Sprache als *vorläufige Unvollständigkeit* beschreibbaren Sachverhaltes (vgl. incompleteness and provisionality [BP'99]) muss eine Überprüfung der Korrektheit des Modells möglich sein, welche lediglich über eine informationstechnische beziehungsweise mentale Ausführung möglich ist. Daraus resultierend herrschen in SiSeOr keine Vorgaben über die Reihenfolge der Orchestrierungsschritte. Es besteht eine nachträgliche Änderbarkeit aller Modellierungsentscheidungen (vgl. Progressive Evaluation und Premature Commitment [GP'96]) und auch die Ausführung von Teil-Prozessen wird ermöglicht (vgl. Anforderungen von EUD im Kontext intelligenter Umgebungen [DV'06]). Hierdurch können auch Änderungen an unvollständigen Prozessen die von Myers und Pane [PM'96] vorgeschlagene direkte Rückmeldung bereit stellen. Darüber hinaus kann das Konstrukt eines universellen Dienst-Platzhalters eingesetzt werden, welcher zwischenzeitlich einen noch nicht gefundenen Dienst repräsentiert und dessen Funktionalität beispielsweise durch einen festen Ausgabewert simuliert. Merkmale wie fokussierende Aufmerksamkeit, die Existenz vieler Dimensionen und die parallele Betrachtung vielfältiger Alternativen [BP'99] können hingegen nur schwer geeignet unterstützt werden.

Direkte Manipulierbarkeit

Neben einer möglichst geringern kognitiven Transformationsleistung und der umfangreichen Bereitstellung relevanter Informationen, sind für den Aufwand zusätzlich die notwendigen Nutzeraktionen relevant, um das Model entsprechend der mentalen Repräsentation zu verändern. Zur Erhöhung der Änderungseffizienz dominiert in den Forschungsergebnissen eine direkte Manipulierbarkeit (vgl. Viscosity [GP'96]), welche eine Änderbarkeit mit einem Minimum notwendiger Aktionen beinhaltet. SiSeOr setzt diesen Anspruch im besonderen Maße bei den häufig im Rahmen der Orchestrierung auftreten Vorgängen um und kann hierbei zusätzlich die in diesem Kapitel bereits beschriebene Steuerung per Tastatureingabe anbieten. Darüber werden dem Endbenutzer eine breite Anzahl potentieller Möglichkeiten bereitgestellt, um die Effizienz der Nutzung zu erhöhen. Beispielsweise wird dies durch das Speichern der Suchbegriffhistorie realisiert, welches eine schnelle Wiederholung eine Suchanfrage ermöglicht und zeitgleich den notwendigen Gedächtnisaufwand reduziert, deren Minimierung auch einer von Nielsen Grundregeln entspricht [Nie'93]. In Folge des Stellenwertes der Metadaten als grundlegende Informationsbasis ist deren Bearbeitungsmöglichkeit besonders einfach zugänglich und nutzbar, da der durch die Nutzer investierte Mehraufwand zu deren Pflege im Prinzip ohne direkte Gegenleistung erfolgt.

Bedarf und Bereitstellung notwendiger Informationen

Für die eigentliche Orchestrierung müssen dem Nutzer diese und weitere Informationen, wie beispielsweise Systemfeedback in unterschiedlichen textuellen oder grafischen Formen zur Verfügung gestellt werden. Dabei muss die Bereitstellung zum richtigen Zeitpunkt, in passender Form und angebrachtem Umfang erfolgen. Dabei soll der kognitive Aufwand des Abrufens (vgl. Visibility und Juxtaposability [GP'96]), Aufnehmens [NC'01] und Merkens [Nie'93] der Informationen minimiert werden. Systemfeedback wird hierbei als automatische visuelle Aktualisierung der direkten Auswirkungen der Nutzerinteraktion auf den Zustand der Anwendung, d.h. der Umgebung oder das Modells verstanden. Hingegen entsprechen Dokumentationsinhalte beispielsweise den bereits

erwähnten und manuell aktivierbaren Hilfetexten. Konzeptuell von zentralem Interesse ist die Bereitstellung von Detailinformationen von Prozessen und Diensten. Diese setzen sich aus den untereinander bestehenden Beziehungen und den entsprechenden Metadaten zusammen, bei dem die letztgenannten sowohl automatisch als auch manuell erzeugt werden. Bei der kognitiven Effektivität der Informationsaufnahme wurden in Kapitel 3.2.2 bereits einige Argumente für visuelle Darstellungsformen deutlich gemacht. Im Hinblick auf die in einer Programmiersituation erforderlichen Informationen sprechen Navarro-Prieto und Cañas von einer verbesserten Zugänglichkeit derselbigen, wenn eine bildhafte Darstellungsform gewählt wird [NC'01]. Hiervon ausgehend nutzt SiSeOr in größerem Umfang visuelle Kommunikationskanäle, in der dem Nutzer bekannte oder bekannt gemachte Symboliken verwendet werden. Hierbei wird jedoch darauf geachtet, keine zu große Anzahl von Informationen anzuzeigen, da Nutzer laut Pane und Myers [PM'96] dazu neigen, alle Symbole eines visuellen Programmiersystems als relevant anzusehen und sicht somit die Vorteilhaftigkeit ab einem gewissen Symbolumfang negiert. Hervorzuheben ist, dass Farbe in den weitern Ausführungen nicht als vollständiger Informationskanal angesehen wird, sondern lediglich unterstützend wirkt, um nicht Nutzer mit Farbenfehlsichtigkeit oder -blindheit [Nie'93] auszuschließen. Beispielsweise werden bei Selektion eine Datenports automatisch syntaktisch kompatible Verknüpfungspunkte durch eine visuelle Form- und unterstützende Farbänderung hervorgehoben. Zusätzlich kann ein Teil der kollaborativ zusammengestellte Metadaten visuell dargestellt werden, wie beispielsweise durch Anzeige der durchschnittlichen Bewertung eines Dienstes in Form einer Thermometer-Metapher. Bei den primär textuellen Metadaten hingegen scheidet die visuelle Darstellungsform aus. Deren Inhalte können jedoch auf Nutzerwunsch, abhängig vom aktuell selektierten Dienst, in einem speziellen Teil der Orchestrierungsumgebung automatisch angezeigt werden und stehen somit unmittelbar bereit. Weitere Informationen, wie etwa die Verfügbarkeit logisch tiefer liegender Prozessebenen, können zwar visuell angedeutet werden, erfordern jedoch eine eigene Darstellungsform, welche im SiSeOr standardmäßig baum-basiert erfolgt. Einen Sonderstatus nehmen die Informationen innerhalb des orchestrierten Prozesses ein, welche durch visuelle Konstrukte mit Textinhalt repräsentiert werden und Kommentaren in traditionellen Programmiersprachen entsprechen. Innerhalb des Modells könnten Elemente oder Elementgruppen mit diesen Kommentaren versehen werden, um diese für das eigene Verständnis oder das Verständnis anderer Nutzer zu erhöhen. Zusätzlich ermöglicht die absolute Positionierung der Dienste eine Möglichkeit für sekundäre Notationen [RI'06], welche durch die Positionierung oder Gruppierung schnell erfassbare Informationen beinhalten können (vgl. Secondary Notation und Escape from Formalism [GP'96]). Nicht nur mit diesen zusätzlichen Informationen dienen Prozesse den Nutzern häufig als eine Vorlage für eine neue Orchestrierung. SiSeOr erlaubt daher, dass in der Entwicklungsumgebung mehr als ein Prozess parallel angezeigt werden kann (vgl. Juxtaposability [GP'96]). Durch die hier skizzierte Integration einzelner Informationsaspekte in die Entwicklungsumgebung soll vor allen Dingen eine Fokusverschiebung während der Orchestrierung vermieden werden (vgl. Aktivitätstheorie in Kapitel 3.2.7) und zeitgleich die Inanspruchnahme kognitiver Ressourcen der menschlichen Merkfähigkeit reduziert werden [Nie'93].

4.4.3 Konzeptspezifische Aspekte

Neben einer möglichst einfachen und schnell erlernbaren Bedienbarkeit sowie der angestrebten Ähnlichkeit zwischen informationstechnischem und kognitivem Modell beinhaltet SiSeOr darüber hinausgehende Maßnahmen, um die Usability konzeptspezifischer Aspekte zu verbessern. Dies beinhaltet die Relativierung einiger Nachteile des visuellen Programmierparadigmas, einer Ratgeberfunktionalität und Maßnahmen zur Reduzierung von Fehlerquellen.

Charakteristiken visueller Programmierung

Die im Folgenden skizzierten Maßnahmen adressieren einige der in Kapitel 3.2.2 genannten Nachteile visueller Programmierung, wobei der erhöhte Platzbedarf und die Änderungsviskosität bestimmter Aspekte dominierend sind. Um notwendige Symbole und deren Platzbedarf gering zu halten, werden in SiSeOr genutzte Dienste innerhalb der Modellierungsumgebung durch eine optisch reduzierte Repräsentation dargestellt, welche jedoch individuell durch Einblendung gewisser zusätzlicher Informationen angepasst werden kann, welches einer der allgemeinen Empfehlungen von Pane und Myers zu entsprechen versucht [PM'96] (vgl. Diffuseness und Terseness [GP'96]). Ein Beispiel hierfür sind Symbole, welche die Anzahl der tiefer liegenden logischen Schichten vermitteln. Die Problematik des in der Literatur häufig genannten Raumbedarfs und der aus diesem Grund erschwerten Übersichtlichkeit tritt maßgeblich bei größeren Prozessen auf [Sch'96]. Dieser kann daher durch sinnvolles Zusammenfassen bestimmter Dienste zu eigenständigen Prozessen begegnet werden. SiSeOr unterstützt durch eine Assistenzfunktionalität sowohl diesen Aufteilungsprozess, als auch eine automatische Anordnung der grafischen Repräsentation des Prozesses. Des Weiteren kann durch eine beliebige Verkleiner- oder Vergrößerung der Modellrepräsentation und ein bereitgestelltes Übersichtsfenster ein bessere Überblick und eine schnellere Navigation geboten werden.

Ratgeberfunktionalitäten

Einen weiteren Schwerpunkt des Konzeptes bildet eine Ratgeberfunktonalität, welche allgemeine oder situationsspezifische Empfehlungen bereitstellt. Eine Form stellt der spezielle Startbildschirm bereit, welcher sich aus individuell zusammenstellbaren Modulen zusammensetzt und neben allgemeinen Ratgeberfunktonalitäten auch den Zugriff auch die wichtigsten Kernfunktionalitäten wie Suche, Hilfe und Bearbeitungshistorie bietet. Die Ratgebermodule zeigen beispielsweise Anwender mit ähnlichem Nutzungsverhalten oder die von diesen aktuell genutzten oder anpassten Dienste und Prozesse. Darüber hinaus existieren Module, welche interessante Vorgänge im Zentralarchiv aufzeigen, die somit den aus den aus der Groupware Forschung stammenden Aspekten der kollaborativen Bewusstseins (Collaborative Awareness) aufgreifen. Eine umfassende Übersicht der Module wird in Kapitel 4.6.3 erfolgen. Eine weitere Ratgeberfunktionalität betrifft die von Myers und Pane [PM'96] im Rahmen der natürlichen Programmierung berichteten Schwierigkeit der Nutzer, die notwendigen Schritten des Programmiervorhabens zu planen und umzusetzen. Aus diesem Grund kann das System Vorschläge für einsetzbare Dienste generieren, auf vergleichbare Prozesse verweisen oder grundlegende Ratschläge für das allgemeine Vorgehen unterbreiten. Die hierfür notwendige Informationsbasis wird auf Grundlage der Beantwortung einiger weniger Fragen gebildet, welche das Ziel des Orchestrierungsvorhabens eingrenzen und so eine Zuweisung auf

einen definierten Grundtyp erlauben. Ein solcher im Vorfeld definierter Grundtyp könnte die spezielle Darstellung oder Formatierung einer gewissen Datenmenge sein, beispielsweise die Ausgabe eines Datenbankeintrages in Form eines PDF Dokumentes. Unabhängig von derartigen Informationen können aufgrund von Laufzeitanalysen oder durch Nutzerbewertungen Empfehlungen für alternative Dienste und Prozesse gegeben werden. Deren funktionelle Verwandtschaft kann zu einem gewissen Teil anhand der Datenports und Schlagwörter erfolgen, erfordert jedoch aufgrund der nicht maschinenlesbaren Semantik zusätzlich weiteres manuell erstelltes Datenmaterial. Neben einer zeitlichen Optimierung ermöglichen die Daten der Laufzeitanalyse dem Nutzer darüber hinaus eine begrenzte Form der Evaluation des orchestrierten Prozesses, welche durch die Arbeit von Mackay [Mac'90] motiviert ist und die Effektivität von Orchestrierungen transparenter machen soll.

Implizite Syntax und Explorationsumgebung

Während der Programmierung wirken sich Fehler, wie sie beispielsweise durch Ko und Myers zusammengetragen wurden [KM'05], und die hieraus resultierenden nicht erwartenden Effekte negativ auf die Nutzermotivation aus [SLM'03]. Diese Fehleranfälligkeit wird jedoch durch die Nutzung des visuellen Programmierparadigmas und einer stark vereinfachten Notation reduziert (vgl error-proneness [GP'96]). Darüber hinaus gilt es jedoch auch die im Vorfeld nicht vermeidbaren Fehler, wie es Nielsen empfiehlt [Nie'93] durch eine geeignete Fehlermeldung zu begegnen, welche das Problem verständlich beschreibt und wenn möglich, konkrete Lösungsvorschläge anbietet. Grundsätzlich ist der Einfluss von Fehlern besonders in frühen Nutzungsphasen maßgeblich und wird zusätzlich durch die Befürchtung beeinflusst, durch diese Fehler irreparable Defekte am Informationssystem oder Dateien zu verursachen [Mac'90]. Eine Maßnahme, diesem zu begegnen, bildet die Möglichkeit einer Syntaxüberprüfung, wie sie beispielsweise von Won für die FreEvolve Plattform durch einen regelbasierten Integritätscheck entwickelt wurde [Won'03; Won'04]. Demgegenüber nutzt SiSeOr die Methodik einer impliziten Syntax, wie sie auch in der Arbeit von Repenning und Ioannidou zu finden ist [RI'06]. Diese erlaubt ausschließlich die visuelle Verknüpfung kompatibler Datenports und schließt somit syntaktische Fehler aus. Zur Vereinfachung werden bei der Selektion eines Datenkanals alle kompatiblen Verknüpfungspunkte der bisher im Prozess befindlichen Dienste visuelle hervorgehoben oder eine gezielte Suche nach Diensten der entsprechenden syntaktischen Vorgabe ermöglicht. Kurz vor der Ausführung eines Prozesses findet darüber hinaus eine abschließende Überprüfung statt, welche den Nutzer kompakt über nicht verbundene und damit ungenutzte Datenports informiert. Die Überprüfung der semantischen Korrektheit eines Prozesses kann hingegen informationstechnisch nicht befriedigend realisiert werden und wird indirekt durch die Bereitstellung von anpassbaren und erweiterbaren Metadaten sowie automatisch generierten Informationen zu einem Dienst oder Prozesse adressiert. Die Angst vor aus dem eigenen Handeln resultierenden Schäden wird mit Hilfe einer Explorationsumgebung entgegengewirkt, durch die ein Prozess risikolos getestet werden kann. Hierfür kann auf verschiedene Datenquellen ausschließlich lesend zugegriffen werden oder es können über den bereits erwähnten Dienst-Platzhalter konkrete Werte an den entsprechenden Datenports angelegt werden. Ausgehend von Burnett et. al [BRC'06] ist es aus Gründen der besseren Übersichtlichkeit bei größeren Prozessen möglich, einzelne Dienste mit einer Markierung zu versehen, welche anzeigt, ob diese bereits im Prozesse getestet wurden oder dies noch aussteht.

Der Stellenwert einer solchen Umgebung wird deutlich, wenn man die Relevanz der experimentellen Auseinandersetzung mit der Orchestrierung für den Lernfortschritt betrachtet [RI'06; Won'03]. In diesem Zusammenhang ist es darüber hinaus essentiell, dass der Nutzer permanent in der Lage ist, bis zu einem gewissen Grad Aktionen rückgängig zu machen, um ein schnelles und effektives Experimentieren zu ermöglichen.

4.5 SiSeOr: Nutzenmaximierung

Tailoringaktivitäten sind nicht Teil der grundsätzlichen Arbeitsaufgabe der Endbenutzer und werden daher ausschließlich durch die mögliche Verbesserung der durch das System geleisteten Arbeitsunterstützung motiviert [SLM'03], welches eine frühzeitige Kommunikation der Tailoringmöglichkeit und der realisierbaren Ergebnisse notwendig macht. Neben dieser rein effizienzorientierten Sichtweise wird zusätzlich versucht, bestimmte Formen sekundären Nutzens zu schaffen, wie etwa die Freude am Tailoring (vgl. beispielsweise [DV'06] und [Nar'93]), welches auch als sekundärer Effekt durch die Einbeziehung kollaborativer Faktoren ermöglicht wird. Gefolgert aus der Arbeit von Wulf und Golombek [WG'01] hat im Hinblick auf die Förderung und Aufrechterhaltung einer allgemeinen Nutzermotivation darüber hinaus das Design der Benutzeroberfläche einen starken Einfluss und muss entsprechend gestaltet werden.

4.5.1 Mächtigkeit und Flexibilität

Aus technischer Sicht soll durch Orchestrierung mit Hilfe des Editors eine ähnlich hohe Flexibilität, wie durch einen direkten Einsatz von BPEL, erreicht werden, welches sich sowohl in der Mächtigkeit der Orchestrierung als auch in der spezifikationskonformen Verwendung und Erweiterung ausdrückt. Durch die primär datenflussbasierte Darstellung fehlen ereignisbasierte Steuerungskonstrukte, welche jedoch in der Forschung des *Natural Programming* einen besondern Stellenwert einnehmen und darüber hinaus tendenziell besser mit dem Prinzip von Prozessen harmonieren. Aus Gründen der Vereinfachung wird jedoch in dieser Grundkonzeption auf eine rein datenflussbasierte Darstellung und Steuerung gesetzt, welche jedoch um spezielle Web Services erweitert wird, welche einzelne ereignisbasierte Prinzipien in einer Datenflussform zur Verfügung stellen. In dieser Hinsicht sind Dienste denkbar, welche zwischen bestimmten Datenports eingesetzt werden können, um beispielsweise eine gewisse zeitabhängige Verfügbarkeit der Daten zu realisieren. Für eine nähere Konzeption ist jedoch eine praktische Evaluation unerlässlich, um die allgemeinen Effekte dieser Herangehensweise im Hinblick auf Handhabbarkeit und Flexibilität bewerten zu können. Daneben sollen weitere Konstrukte zur Verfügung stehen, welche die Flexibilität der Nutzung innerhalb der durch die BPEL Spezifikation gesetzten Grenzen von BPEL erweitern, in dem bestimmte Basisdienste zur Verfügung gestellt werden. Beispielsweise sei hier auf Dienste verwiesen, welche einen Rückgabewert in Form eines komplexen Datentyps in die datentypspezifischen Bestandteile zerlegen oder abhängig von benötigten und ausgegebenen Datenwerten Eingabemasken und Ausgabeformen generieren. Die Konzeption einer individuell zusammenstellbaren Ausführungsoberfläche eines spezifischen Prozesses steht noch aus, sollte jedoch entsprechend den Ergebnissen von Myers und Ko [MK'05] in einer WYSIWYG Form möglich sein.

4.5.2 Transparenz und Erreichbarkeit des Tailorings

Der Fokus des Konzeptes in dieser Ausarbeitung liegt primär auf einer Tailoring- und weniger auf eine Ausführungsumgebung. Die reine Ausführung stellt dabei die komplexitätsärmste Nutzungsmöglichkeit dar und dient im Regelfall als Einstieg in die Programmnutzung. In frühen Nutzungsphasen der Software muss daher der Ausführungsaspekt stärker hervorgehoben werden, ohne jedoch die Anpassungsmöglichkeiten zu stark in den Hintergrund treten zu lassen, da laut Mackay [Mac'90] besonders bei Einführung von neuer Software die höchste Tailoringmotivation herrscht. Während dieser Nutzungsphase muss daher durch die Benutzeroberfläche nachhaltig die Anpassungsmöglichkeit kommuniziert werden, welches zeitgleich durch eine offensichtliche sowie direkte Zugänglichkeit derselbigen unterstützt wird. Diese Handlungsalternative gilt es besonders in Situationen aufzuzeigen, in der Nutzern gewöhnlich bewusst wird, dass die Ausführung nicht angepasster Dienste oder Prozesse nur eine suboptimale Nutzenstiftung hervorbringt. Im Kontext von SiSeOr soll hierzu das von Wulf und Golombek [WG'01] erarbeitete Konzept der *Direct Activation* als Orientierungsmöglichkeit dienen.

4.5.3 Kollaborative Aspekte

Die bisher im Konzept beschriebenen Maßnahmen auf technischer Ebene bieten eine Infrastruktur, um auf einfache Weise erstellte oder angepasste Prozesse anderen Nutzern zugänglich zu machen. Die im folgenden beschriebenen Aspekte, welche sich unter anderem an *Patterns of sharing customizable software* von Mackay [Mac'90] und an *From tailoring to appropriation support: Negotiating groupware usage* von Pipek [Pi'05] orientieren, ergänzen und erweitern diese Infrastruktur durch die Berücksichtigung kollaborativer Faktoren. Das Konzept bietet beispielsweise versierteren Nutzern technischer Prozesse und Dienste die Möglichkeit, der Nutzergemeinschaft diese in einer domänen-näheren Form bereitzustellen, in dem diese verschachtelt und/oder um entsprechenden Metadaten ergänzt werden. So kann eine vergleichbare Form der Arbeitsteilung entstehen, wie es Nardi und Miller [NM'91] bei der Nutzung von Spreadsheets festgestellt haben. Bestimmte Nutzertypen beschränken sich im Kontext von Tailoring jedoch Mangels Zeit und Motivation ausschließlich auf die reine Nutzung von Anpassungsdateien, ohne eigenständig solche zu verändern oder zu erstellen [Mac'90]. Besonders im durch den letztgenannten Fall ist der erreichbare Nutzen von SiSeOr massiv von der Bereitschaft anderer Nutzer abhängig, eigene Prozesse im Zentralarchiv zugänglich zu machen, obwohl dies mit einem zusätzlichen Aufwand verbunden ist. Um diese Bereitschaft positiv zu beeinflussen ermöglicht SiSeOr neben der regulären, personenbezogenen Bereitstellung ebenfalls eine anonyme Form der Veröffentlichung. Dies soll potentiellen Bedenken der Nutzer begegnen, welche beispielsweise durch suboptimale oder fehlerhafte Prozesse negative Effekte für die eigene Reputation befürchten. Die transparente Verknüpfung von Prozessen und Nutzerprofilen - wobei dieses nicht zwingend auf die reale Person schließen lassen muss - ermöglicht unter anderem ein umfangreiches Vorschlagssystem, welches Gegenstand von Kapitel 4.6.3 ist. Basierend auf der Arbeit von Gantt und Nardi [GN'92] werden im Hinblick auf eine Motivationssteigerung durch SiSeOr individuelle Profilseiten bereitgestellt, wie sie häufig bei losen Nutzergemeinschaften des Internets eingesetzt werden. Ziel ist es, die Leistungen für die Gemeinschaft transparent zu machen, um somit eine gewisse Gegenleistung für den Aufwand der Veröffentlichung zu bieten. Dieser kann sich beispielsweise durch positive Bewertungen der Prozesse oder

durch Anerkennung und einen gewissen Status in der Gemeinschaft ausdrücken. Zusätzlich ermöglicht eine Bereitstellung von synchronen oder asynchronen Kommunikationskanälen, welche einen direkten oder indirekten Austausch gestatten, die Schaffung einer Diskussionsbasis, welche zu einer sukzessiven Verbesserung einzelner Prozesse führen kann. Daneben bietet sich die Möglichkeit, Nutzer mit ähnlicher Arbeitspraxis und deren genutzte, angepasste oder erstellte Prozesse zu durchsuchen, welche zusätzlich durch das in Kapitel 4.6.3 vorgestellte Informationsportal unterstützt wird. Konkret kann der Informationsumfang und Informationsreichweite der eigenen Profilseite flexibel festgelegt werden, um nur die vom Nutzer gewünschten Daten anderen Nutzern zugänglich zu machen. Um den kommunikativen Aspekt hervorzuheben, wird zum einen ein systeminterner Kommunikationskanal zur Verfügung gestellt, welcher unter anderem über die Profilseite erreichbar ist. Zudem ist es möglich, weitere Kontaktdaten anzugeben, das Profil mit weiteren persönlichen Informationen sowie einem Foto zu erweitern. Die Profilseite ermöglicht es darüber hinaus, die jeweils veröffentlichten und genutzten Dienste und Prozesse zu durchstöbern, so weit die öffentliche Verfügbarkeit dieser Information vom Nutzer im Vorfeld nicht eingeschränkt wurde. Automatisch wird vom System je nach Umfang veröffentlichter Prozesse und hinzugefügter Metadaten ein aussagekräftiger Status verliehen, welcher durch visuelle Symboliken hervorgehoben wird.

4.6 SiSeOr: Konzeptdetails

Diese in den vergangenen beiden Kapiteln skizzierten Konzeptaspekte werden aufgrund des Umfangs der Arbeit in diesem Kapitel lediglich auszugsweise im Detail ausgearbeitet.

4.6.1 Repräsentation

Die visuelle Repräsentation eines Dienstes erfolgt in der Grundausführung als eine Quadratische Box, an deren Außenseite in Form von Kreisen die Repräsentanten der Datenports angehängt sind. Die Verbindung derselbigen wird durch einfache Linien dargestellt, welche durch einen Pfeil die Datenflussrichtung anzeigen. Die grundlegende Darstellung eines sich in der Orchestrierung befindlichen Prozesses wird in Abbildung 24 verdeutlicht.

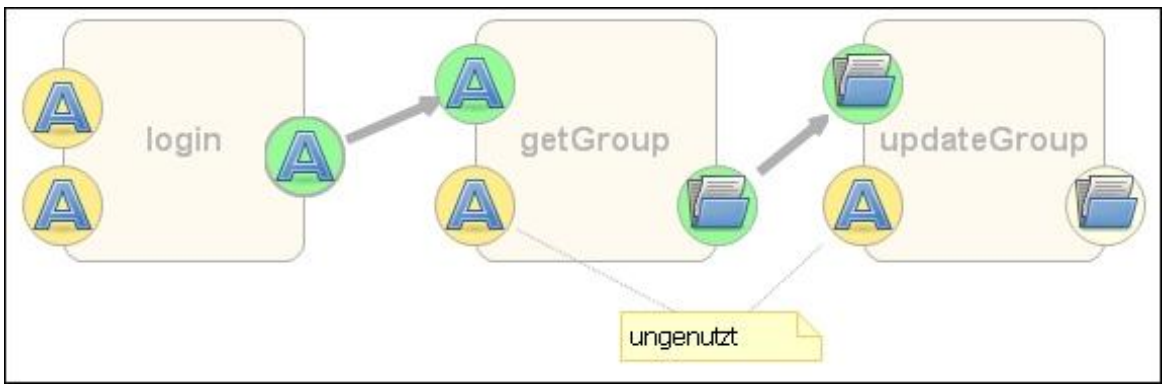

Abbildung 24: Repräsentation sowie Notation eines Prozesses und seiner Dienste

Je nach Domäne können diese grundlegenden Repräsentationen in gewissem Umfang durch Konfiguration in Form und Farbe angepasst werden. Innerhalb der Box kann ein als Referenz in den Metadaten hinterlegbares Icon angezeigt werden, welches beispielsweise eine schnell visuell

erfassbare Einordnung in eine gewisse Typisierung erlaubt (beispielsweise Datenquelle, Datenverarbeitung, Datenausgabe) oder als Metapher die Funktionsweise veranschaulicht.Die Repräsentation der Datenports unterliegt einer besonderen Flexibilität, da ihre Position am Rand der Box beliebig verschoben werden kann, um beispielsweise einen Prozess in einer horizontalen oder vertikalen Orientierung abbilden zu können. Darüber hinaus werden anhand der Datentypen die Datenports mit einem passenden Icon versehen, um ein schnelles Überblicken der Kompatibilitäten zu ermöglichen. Zur besseren Veranschaulichung werden verknüpfte Datenports und die entsprechenden Verbindungen durch einen breiten Rand und eine zusätzliche farbliche Hervorhebung markiert, wie es ebenfalls in Abbildung 24 zu sehen ist.

4.6.2 Metadaten-Grundtypen

Auf technischer Ebene bildet die Erweiterung der WSDL-Schnittstellenbeschreibung den zentralen Aspekt, welche die Nutzung im EUD Kontext ermöglicht und die Integration kollaborativer Elemente der Arbeitspraxis erlaubt. Neben der Möglichkeit, individuelle Typen von Metadaten zu spezifizieren und zu erstellen, beinhaltet SiSeOr eine gewisse Anzahl von Grundtypen. Essentiell ist hierbei die in natürlicher Sprache verfasste Beschreibung der Funktionsweise und Anforderungen des Dienstes, welche durch ein zugewiesenes Symbol visuell unterstützt werden kann. Zusätzlich sind zur deutlicheren Abgrenzung wichtige Unterscheidungsmerkmale durch Typisierung (Datenquelle, -verarbeitung und –ausgabe), frei definierbare Schlagwörter und die Einteilung auf einer Orientierungsskala der Domänen- und Technologienähe vorhanden. Darüber hinaus wird neben der Dokumentation von Ein- und Ausgabewerten auch die Angabe eines konkreten Datenbeispiels ermöglicht, um missverständliche Beschreibungen entgegenzuwirken. Neben diesen rein funktionellen Aspekten ermöglichen Bewertungen und Kommentare auch die Angabe qualitativer Faktoren, welche durch automatisch generierte Daten ergänzt werden. Hierbei werden unter anderem die Verwendungshäufigkeit und übliche Verknüpfungspartner gespeichert und Verweise auf beispielhafte Verwendungen in anderen Prozessen generiert. Zusätzlich beinhalten diese Metadaten statistische Informationen, wie den letzten Zugriffszeitpunkt oder das Datum der Erstellung beziehungsweise der Importierung in das UDDI. Hinzu kommt die Speicherung des Autors eines Dienstes oder Prozesses, welches entweder auf einen externen Hersteller oder ein Nutzerprofil verweist, vorausgesetzt der Anwender hat diese Information freigegeben. Eine Besonderheit bildet die Speicherung der über das System an den Autor oder die Nutzergemeinschaft gestellten Fragen und die dazugehörigen Antworten, welches den Aufbau einer dienstspezifischen Hilfe ermöglicht. Allgemein könnte ein Assistent die Eingabe der manuell erzeugten Metadaten in gewisser Weise leiten, indem in absteigender Priorität entsprechende Daten in Frageform eingefordert werden, wobei konkrete Schritte vom Nutzer jederzeit übersprungen werden können oder in eine direkte Eingabemöglichkeit gewechselt werden kann.

4.6.3 Informationsportal

Endbenutzer werden über die Startseite der Anwendung mit relevanten Informationen über inhaltliche Veränderungen des Zentralarchivs und Empfehlungen aufgrund von Profilanalysen versorgt. Diese Seite orientiert sich im Aufbau an der Metapher des Armaturenbrettes und ist durch Entfernen und Heinzelner Module individuell anpassbar. Dabei wird die Privatsphäre der Nutzer in der

Art und Weise berücksichtigt, dass als Informationsbasis dieser Analyse nur Daten genutzt werden, welche die betrachteten Nutzer im Vorfeld als öffentlich zugänglich klassifiziert haben. Die Module, welche durch eine beispielhaften Zusammenstellung in Abbildung 25 skizziert sind, unterteilen sich in die thematischen Blöcke von Suche, Nutzer und Dienste.

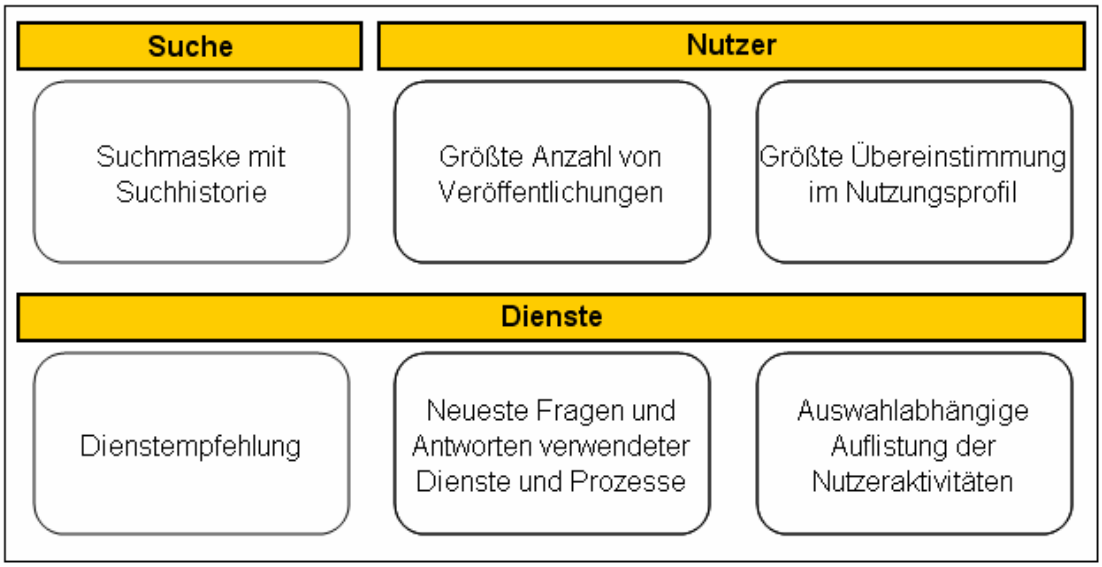

Abbildung 25: Skizzierte Zusammenstellung eines spezifischen Informationsportals

Der Suchblock setzt sich aus einer Suchmaske und einer Liste der am häufigsten verwendeten Schlagwörter zusammen. Im Personen-Block werden die Nutzer aufgelistet, welche in einer gewissen Zeitspanne die meisten Prozesse öffentlich zugänglich gemacht haben. Dieser ermöglicht zum einen die Identifikation potentieller Tailoring-Experten und dient zum anderen Steigerung der Motivation, Prozesse zu veröffentlichen. Um Personen mit einer ähnlichen Aufgabenstruktur zu finden, werden in diesem Block profildatenabhängig andere Nutzer mit ähnlichen Strukturen verwendeter Prozesse und Dienste angezeigt. Dabei sind die Profile der im Rahmen dieses Blocks aufgelisteten Nutzer mit deren Namen verknüpft, so dass diese direkt erreichbar sind. Ebenfalls im Dienstblock werden anhand des Nutzungsprofils die neuesten Fragen, Antworten und Kommentare der verwendeten Dienste und Prozesse angezeigt. Darüber hinaus wird innerhalb dieses Blockes auf einen ausgewählten Dienst oder Prozess verwiesen, welcher anhand vorher spezifizierter Kriterien, wie beispielsweise der Bewertung, ausgewählt wird und somit zu einer punktuellen Erweiterung der eigenen Nutzungshorizontes führen soll. Ebenso werden im Rahmen der Suche automatisch generierte Vorschläge unterbreitet, welche Services umfassen, die im derzeitigen Arbeitskontext besonders geeignet erscheinen oder von Kollegen in ähnlichen Situationen bereits verwendet wurden.

4.6.4 Kernbereiche der Entwicklungsumgebung

Im Folgenden werden kurz die Kernelemente der Entwicklungsumgebung und deren individuelle Charakteristiken vorgestellt.

Suche

Basierend auf der Erfahrungen des komponentenbasierten Tailoring Systems *FreEvolve* [WSW'06] liegt ein besonderes Augenmerk auf der optimalen Unterstützung der Suche von Web Services und der Navigation innerhalb des *Zentralarchivs*, welches beispielsweise laut Baroth und Hartsough [BH'94] in den früheren Umsetzungen von *LabVIEW* und *VEE* (Visual Engineering Environment) lediglich suboptimal unterstützt wurde. Dabei ist eine Suche nach nahezu allen Informationen der Metadaten möglich, wobei ebenfalls kombinierte Suchparameter verwendet werden können. Beispielsweise kann eine Suchanfrage formuliert werden, welche alle Prozesse anzeigt, die von einem bestimmten Nutzer erstellt wurden und deren durchschnittliche Bewertungen einen bestimmten Mindestwert erreichen. Zusätzlich werden wie im Konzept bereits erwähnt die letzten Suchanfragen in einer Historie gespeichert und können somit unmittelbar wiederholt werden, welches zu einer Reduzierung des notwenigen Merkaufwandes des Nutzers führt [Nie'93]. Dabei können die Ergebnisse in einer Listenform mit frei definierbaren Spalten oder in Form einer Miniaturansicht dargestellt werden, bei der im letztgenannten Fall visuell die Anzahl der Eingabe- und Ausgabedatenports vermittelt wird.

Metadaten

In allen Nutzungsphasen werden je selektiertem Dienst oder Prozess die Metadaten in einem speziellen Bereich der Entwicklungsumgebung angezeigt, welcher zeitgleich eine direkte Bearbeitung ermöglicht. Diese unmittelbare Verfügbarkeit soll den Informationsbedarf bedienen, die Merkfähigkeit der Nutzer entlasten und zeitgleich den vermeidbaren Aufwand zur Pflege von Metadaten reduzieren.

Hierarchie

Die Hierarchie der zusammengesetzten Prozesse wird in der Grundversion über eine baumartige Struktur navigierbar sein, die so eine möglichst einfache Vermittlung der bestehenden Beziehung zwischen den Funktionskomponenten vermittelt. Diese ist im Grundzustand von SiSeOr ebenso wie der Bereich zur Anzeige und Bearbeitung der Metadaten ein permanent sichtbarer Bestandteil der Entwicklungsumgebung.

Palette und Favoriten

Die in einer konkreten Orchestrierungssituation verfügbaren Dienste und Prozesse sind in einer sogenannten Palette gesammelt, welches als Teil der Entwicklungsumgebung ebenfalls die Werkzeuge für grundlegende Befehle enthält, wie etwa die Verbindung zweier Datenports. Die in einer symbol- oder listenbasierten Darstellungsform dargestellten Dienste können sich dabei in zwei verschiedenen Bereichen befinden. Ein allgemeiner Bereich dient der Sammlung von primär in der aktuellen Situation und damit einmalig verwendeten Diensten, während ein zweiter Bereich so genannte Dienstfavoriten beinhaltet, deren Status vom Nutzer individuell gesetzt werden kann und einen schnellen Zugriff auf häufig verwendete Funktionalitäten erlaubt. Die Aufteilung und Inhalte der Palette werden durch die Skizze in Abbildung 26 deutlich gemacht.

Abbildung 26: Skizze der Bereiche und Inhalte der Palette

Für größere Mengen an Favoriten kann darüber hinaus das Spektrum der Suchfunktion von SiSeOr auf die Menge der eigenen Favoriten beschränkt werden und eine eigene Ordnungsstruktur angelegt werden.

4.7 Zusammenfassung

Basierend auf der Analyse von Literatur, Konzeptrealisierungen und der durchgeführten Vorstudie, wurde in diesem Kapitel ein Konzept entwickelt, welches Endbenutzern eine Softwareanpassung in Form von frei orchestrierbaren BPEL-Prozessen erlaubt. Hierbei diente eine aus Endbenutzerperspektive betrachtete Gegenüberstellung von Aufwand und Nutzenstiftung als grundsätzliche Orientierung. Das Zentrum des als SiSeOr bezeichneten Konzeptes bildet die visuelle Orchestrierung eines datenflussbasierten Geschäftsprozesses auf Grundlage der *Box und Wire* Metapher. Auf Grundlage verschiedener Komplexitätsstufen ermöglicht SiSeOr einer differenzierte Nutzbarkeit, welches zusätzlich durch ein geringes Einstiegswissen ergänzt wird. Dieses wird durch die Bereitstellung zusätzlicher Metadaten, der Abstraktion elementarer technischer Faktoren sowie motivationsfördernder und differenzierter Lernmöglichkeiten erreicht. Weiterhin bildet die umfassende Orientierung an die Charakteristiken mentaler Modelle die Unterstützung durch Ratgeberfunktionalitäten und die Reduzierung potentieller Fehlerquellen einen weiteren Kernaspekt. Im Hinblick auf die erreichbare Nutzenstiftung ist neben der realisierbaren Mächtigkeit der Prozesse die direkte Verfügbarkeit relevanter Daten zentral, welche sich aus automatisch generierten und durch die Nutzergemeinschaft kollaborativ erstellten Informationen zusammensetzen. Die im folgenden Kapitel beschriebene Implementierung realisiert die essentiellen Punkte dieses Konzeptes, welches die Bereitstellung einer Umgebung zur Orchestration umfasst, die auf Grundlage einer geeigneten visuellen Repräsentation und Notation die Erstellung eines BPEL Prozesses erlaubt.

5 Implementierung

Im Rahmen der Diplomarbeit erfolgte auf Grundlage der Eclipse Plattform eine prototypische Umsetzung des Konzeptes, welches zum einen auf einer angepassten Variante von EUSOP und zum anderen auf dem Grafical Modeling Framework (GMF) von Eclipse aufsetzt. Die Realisation beinhaltet die essentiellen Merkmale des Konzeptes, wobei der Fokus der Realisation auf der Unterstützung der grundlegenden Nutzungsform liegt. Diese beinhaltet die grafische Modellierung beliebiger Web Services und dessen Operationen zu einem BPEL-Prozess, der sich aus einer beliebigen Anzahl unterschiedlicher Web Services zusammensetzen kann.

5.1 Technische Basis

Im Folgenden werden die grundlegenden Informationen der verwendeten Technologien vermittelt, welche als Verständnisgrundlage der in den anschließenden Unterkapiteln beschriebenen Schritte der Modellierung und Realisierung dienen. Dabei bildet Eclipse sowohl Entwicklungsumgebung als auch das technische Basis-Framework des Prototyps, welcher als Eclipse Plug-in umgesetzt wurde.

5.1.1 Eclipse

Eclipse ist eine unter der Eclipse Public License (EPL) veröffentlichte und in Java implementierte open source Rich-Client Platform, welche auf einem flexibel erweiterbaren Komponentenmodell basiert. Traditionell wird Eclipse als universelles Entwicklungswerkzeug verstanden und beinhaltet in der Ausführung als Software Development Kit (SDK) eine integrierte Entwicklungsumgebung (IDE), welche durch die *Java Development Tools (JDT)* und die *Plug-in Development Environment* (PDE) zu einer Entwicklungsumgebung für Java bzw. Eclipse Komponenten erweitert wird (vgl. Abbildung 27).

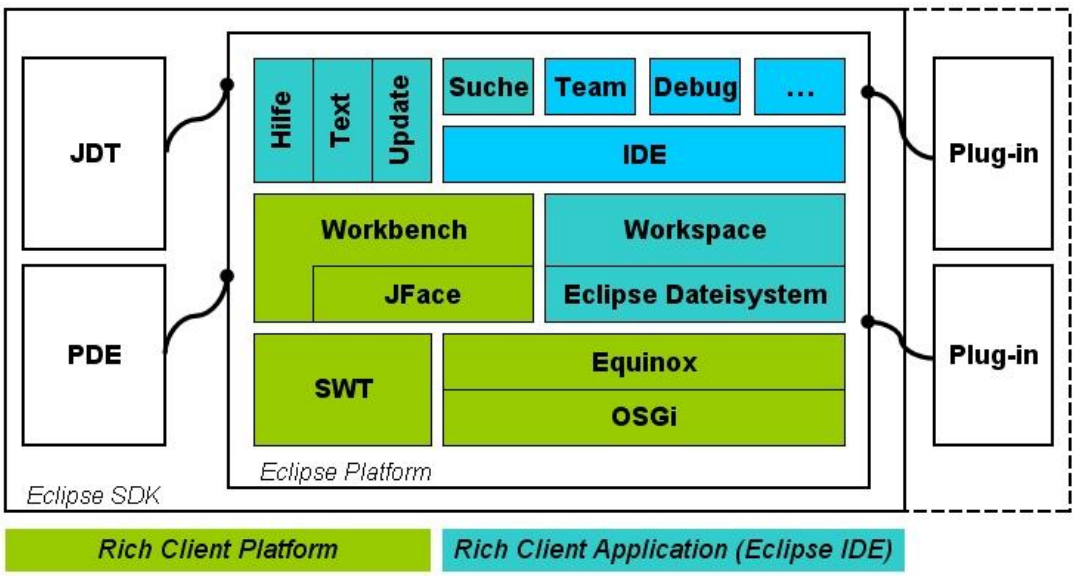

Abbildung 27: Eclipse Software Development Kit

Neben dieser Nutzungsform dient die RCP zusätzlich als technische Basis eigenständiger Softwareprodukte, so genannter Rich-Client Applications, wie sie auch die Eclipse IDE darstellt. Folglich wird Eclipse als universelles Werkzeug zur Softwareentwicklung genutzt, dient der Realisierung aufbauender Komponenten (plug-ins) oder bildet die Grundlage autonomer Rich-Client-Applications. Aus Kompatibilitätsaspekten wird seit 2006 jährlich eine offizielle Distribution der Eclipse Plattform veröffentlicht, welche bereits die wichtigsten Open-Source-Projekte im Rahmen der Softwareentwicklung beinhaltet. Die aktuell auf Eclipse 3.3 basierende Distribution mit der Bezeichnung *Europa* [URL,d] wurde im Juni 2007 veröffentlicht.

Rich Client Platform

Als RCP wird die minimale Menge an elementaren Eclipse Komponenten bezeichnet, welche zur Entwicklung eigenständiger Rich Client Applications vorhanden sein muss. Essentiell ist hierbei der Laufzeitkern [GB'03], welcher auf Grundlage des *Equinox* Komponentenmodells den Lebenszyklus (Auffinden, Laden, Ausführen), der auch als *plug-in* oder *bundels* bezeichneten Komponenten innerhalb der Anwendung steuert und überwacht. *Equinox* stellt dabei eine spezielle Implementation der OSGi (früher Open Services Gateway initiative) R4 Kernspezifikation dar, welche unter anderem zur Laufzeit ein dynamisches Installieren und Deinstallieren (hot plugging) ermöglicht. Auf diesem Fundament bauen alle weiteren Komponenten der RCP und darüber hinausgehender Funktionalitäten auf. Neben dem Laufzeitkern beinhaltet die RCP die grundlegenden Komponenten für die Oberflächen, die sich aus dem Standard Widget Toolkit (SWT) und JFace zusammensetzen, welche die Basisfunktionalitäten der Benutzeroberfläche zur Verfügung stellen und hierbei auf die nativen GUI-Bibliotheken des Betriebssystems zurückgreifen. Abgeschlossen wird RCP durch eine generische Workbench, welche die grundlegenden und teilweise hochkomplexen Benutzeroberflächenmodule, wie beispielsweise Editoren, Wizards und Sichten, enthält [Dau'07]. Dieser minimalen Rich Client Platform können jedoch nach Bedarf zusätzlich optionalen Komponenten, wie beispielsweise das von der Eclipse IDE unabhängige Hilfesystem oder die Updatefunktionalität, hinzugefügt werden.

Erweiterbarkeit

Die Flexibilität von Eclipse beruht auf dem Prinzip lose gekoppelter Komponenten, welche über Schnittstellen funktional erweitert werden können. Diese sogenannten *Extension Points* bestehen aus einer Beschreibung der Art und Weise wie diese konkret durch *Extensions* erweitert werden müssen, um beispielsweise das Standardverhalten einer Komponente zu überschreiben. Die Beschreibung von *Extension Points* und konkreter *Extensions* sowie spezifizierte Sichtbarkeiten, Abhängigkeiten und Eigenschaften, sind innerhalb der in XML formulierten Metadaten (Manifest.mf, plug-in.xml) einer Komponente hinterlegt. Typischerweise beinhaltet eine solche Komponente neben den beiden genannten Dateien den kompilierten Java-Quelltext, benötigte Bibliotheken und zusätzlich erforderliche Ressourcen. Häufig wird dessen Lebenszyklus durch eine Unterklasse der Klasse *plug-in gesteue*rt, welche nach Aktivierung des plug-ins instanziiert (Singleton) wird. Neben den plug-ins, welche die kleinste sinnvolle Einheit von Funktionalität darstellt, existieren weiterhin Features und Fragments. Fragments kann als eine von einem plug-in abhängige Ergänzung angesehen werden, welche zur Ausführungszeit mit diesem gemischt wird. Fea-

tures, wie beispielsweise die JDT, bündeln hingegen eine gewisse Menge von plug-ins, Fragments und weiterer Features zu Applikationseinheiten.

Rich Client Applications

Die Entwicklung einer eigenständigen Rich-Client-Application (auch bezeichnet als RCP-Distribution) sowie die Erweiterung eines bestehenden Eclipse Systems erfolgt daher durch die Einbindung zuvor entwickelter oder bereits bestehender plug-ins. In Fall einer RCA werden daher neben den eigenentwickelten Funktionalitäten ausschließlich notwendige plug-ins integriert. Dies ermöglicht eine äußerlich als Einheit präsentierte Anwendung, welche als eigenständiges Produkt kommerziell nutzbar ist. Beispiele für solche RC-Anwendungen sind *IBM WebSphere Studio Homepage Builder* und der *Borland JBuilder 2007*.

Application Frameworks

Die Mehrzahl von Funktionen sind jedoch nicht als eigenständige Anwendungen sondern weiterhin in Form von ergänzenden Projekten und Plug-ins für die Eclipse Plattform vorhanden. Neben JDT und PDE sind hier vor allen Dingen verschiedene Entwicklungsumgebungen für eine breite Anzahl von Programmiersprachen wie C++ und PHP zu nennen. Eine andere Gruppe bilden die Application Frameworks, welche durch Bereitstellung funktionaler Baublöcke den Entwicklungsprozess maßgeblich reduzieren können. Die im weiteren Verlauf betrachteten Frameworks EMF, GEF und besonders GMF stellten solche Frameworks dar.

5.1.2 Graphical Modeling Framework

Das Graphical Modeling Framework (GMF) ermöglicht eine modellgetriebene Softwareentwicklung grafischer Eclipse Editor-Plug-ins, welches durch eine integrierte Nutzung und generative Verknüpfung der beiden getrennten Frameworks EMF (Eclipse Modeling Framework) und GEF (Graphical Editing Framework) ermöglicht wird. Das Eclipse Modeling Framework erlaubt auf Grundlage eines Metamodells im Ecore Format (XMI) die Generierung von Java-basierter Eclipse Plug-ins, welche neben dem Modell auch die Funktionen zur Erstellung, Manipulation und Serialisierung von Modellinstanzen umfassen [BSM'03]. Optional kann ein Editor Plug-in generiert werden, welches zur Laufzeit einen baum-basierten Editor zum Verändern dieser Instanzen zur Verfügung stellt. Das Grafic Modeling Framework hingegen bietet Basisklassen zur Entwicklung eines grafischen Editors, welcher ein existierendes Datenmodell mit einer editierbaren grafischen Sicht verknüpft, ohne jedoch eine metamodellbasierte Codegenerierung zu unterstützen. GEF Editoren liegt hierbei die Model-View-Controller Architektur zugrunde, wobei die Repräsentationselemente durch die Renderingfunktionalität des Draw2D Toolkit bereitgestellt werden, welches ebenso die Aufgaben Layouting und Skalierung erfüllt. GEF übernimmt hierbei zu Laufzeit Controllerfunktion und erweitert die Workbench um spezifische Konstrukte, wie beispielsweise eine spezielle Interaktionsschicht für die grafische Manipulation durch den Nutzer. Als Voraussetzung für den Einsatz von GEF muss das Datenmodell über Modellmanipulationsfunktionalität sowie Mechanismen zu Persistenz, Wiederherstellung und Benachrichtigung verfügen [HS'05]. Durch die generische Überwindung der Trennung zwischen EMF und GEF ermöglicht der Einsatz von GMF eine schnelle Entwicklung modellgetriebener Editoren, deren grafische Notation als domänenspezifische Sprache (DSL) konzipiert

werden kann. Im Gegensatz zur traditionellen Kopplung von EMF und GEF beinhaltet die Nutzung von GMF über die Generierungsfunktionalität hinausgehende Vorteile, wie beispielsweise eine umfassende Erweiterbarkeit, wieder verwendbare Komponenten für grafische Editoren und die Serialisierung der konkreten Modellrepräsentation. Darüber hinaus trennt GMF sowohl bei der Generierung zur Laufzeit und bei der Serialisierung klar zwischen Model und Repräsentation.

Generische Komponente

Der durch eine Assistenzfunktionalität geregelte iterative Prozess der Erstellung und Nutzung dieser Modellspezifikation bzw. Generierungsdateien wird in Abbildung 28 skizziert und im Weiteren verdeutlicht. Die Serialisierung dieses und alle anderen Modellspezifikationsdateien erfolgt im leicht abgewandelten [Ala'06] XMI 2.0 (XML Metadata Interchange) Format, so dass diese prinzipiell auch per Texteditor erstellt und verändert werden können.

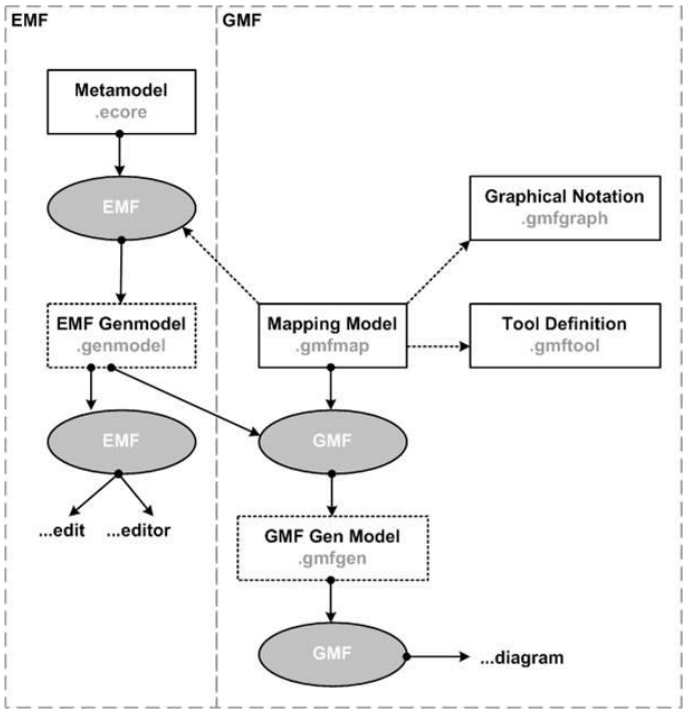

Abbildung 28: Prozessschritte und involvierte Modelldateien [VKE'06]

Die Basis der Codegenerierung bildet das grundlegende Meta-Modell, welches in einer Beschreibung konform zum Ecore-Klassendiagramm formuliert sein muss. Ecore ist prinzipiell mit dem Essential Meta-Object Facility (EMOF) vergleichbar und ermöglicht die Beschreibung von Meta-Modellen, wie beispielsweise UML (Unified Modeling Language) – es stellt folglich ein Meta-Meta-Modell dar. Neben einer direkten Modellierung durch Nutzung eines entsprechend verfügbaren baumbasierten (EMF) oder visuellen (GEF) Editors besteht darüber hinaus eine Importmöglichkeit aus unterstützen Formaten, wie beispielsweise annotiertem Java Code oder UML Klassendiagrammen. Das grundlegende Metamodell (.ecore), auch als Domänen-Meta-Modell bezeichnet, wird im Anschluss in den Java Quellcode eines oder mehrerer Eclipse plug-ins transformiert, wobei dieser Schritt durch eine spezielle Generierungsdatei (.genmodel) und auf Grundlage dessen anpassbarer Parameter durchgeführt wird. Hierauf folgt die durch GMF bereitgestellte Spezifikationsmöglichkeit der grafischen Repräsentationen von Knoten (Klassen) und Verbindungen (Assoziationen) in der Grafical Notation (*.gmfgraph*), wobei auch Kinderknoten und Verschachtelungen

unterstützt werden. Im Gegensatz hierzu legt die Tool Definiton (*.gmftool*) unter anderem die Elemente der grafischen Palette oder Menüinhalte fest. Das Mapping Modell (*.gmfmap*) umfasst die Zusammenführung von Metamodell, Grafical Notation und Tool Definition und spezifiziert die jeweiligen Beziehungen zwischen den Modell- und Repräsentationselementen. Zusätzlich besteht an dieser Stelle die Möglichkeit, bestimmte Nebenbedingungen zu setzen, welche eine permanente oder gezielt initiierte Validierung ermöglichen und unter anderem mit Hilfe von OCL (Object Constraint Language) formuliert werden können. Die bis zu diesem Punkt herrschende Trennung zwischen den Spezifikationsdateien ermöglicht eine unabhängige Wiederverwendung von Model und grafischer Notation. Abschließend wird wiederum ein GMF Gen Model (*.gmfgen*) erstellt, welches ebenfalls auf Grundlage veränderbarer Parameter die Codeerzeugung des grafischen Editors steuert. Neben den Einflussmöglichkeiten während der Generierung können manuelle Änderungen im generierten Quellcode durch spezielle Annotationen gekennzeichnet werden, so dass diese bei einer Anpassung der Spezifikationsdateien und einer anschließenden Neugenerierung erhalten bleiben. Als Ergebnis der generischen Komponente von GMF werden im allgemeinen Fall vier verschiedene plug-ins erstellt, welche Modell (.model), Modifikationsfunktionalitäten (.edit), einen baumbasierten Editor (.editor) sowie die Klassen der grafischen Notation und deren Laufzeitfunktionalitäten (.diagram) enthalten.

MVC Architektur und Laufzeitkomponente

Zur Laufzeit dient die GEF MVC-Infrastruktur die Konsistenzerhaltung zwischen Modell und der Präsentation. Diese wird durch Draw2D in Form von Figuren bereitgestellt und ist im Kontext von GMF als Pendant des EMF Modells zu betrachten (siehe Abbildung 29). Neben dieser Funktion stellt GEF die eigentliche Interaktionschicht bereit, welche Maus- und Tastatureingaben verarbeitet, interpretiert und dementsprechend Änderungen am Modell durchführt. Zusätzlich erweitert GEF die Eclipse Workbench beispielsweise um eine Werkzeugpalette, sowie weitere Komponenten und Funktionalitäten.

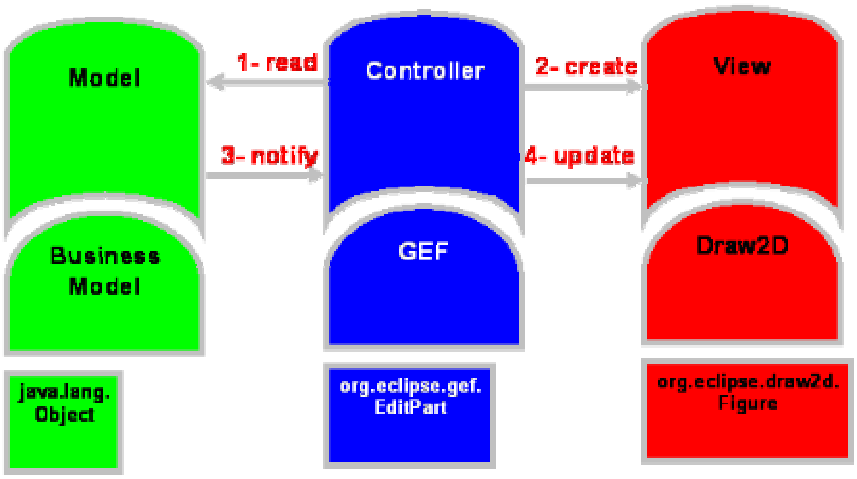

Abbildung 29: MVC Struktur [URL,e]

Wie bereits erwähnt erfolgt das Layouting der in der Modellspezifikation festgelegten Symbole durch das direkt auf SWT aufsetzende Draw2D Toolkit. Symbole werden in Form von Figuren bereitgestellt, welche auch nicht-rechteckige Formen annehmen können, verschachtelbar sind und

durch Kombination geometrischer Grundformen auch die Erstellung individueller Figuren erlauben. Darüber hinaus unterstützt das Toolkit Maus und Tastatur Events, verfügt über ein flexibles Koordinatensystem und beinhaltet unter anderem verschiedene Layoutstile sowie Routing-Algorithmen. Zentrales Element der Steuerungsinfrastruktur sind spezielle EditParts, welche die Modellinstanzen verwalten, Änderungen am Modell durchführen und die Repräsentation aktualisieren. Hierbei existiert für jede Klasse bzw. Assoziation des Modells eine eigene EditPart Klasse, welche die Interaktionsverarbeitung und Konsistenzerhaltung übernimmt. Durch das System ausgelöste oder nutzerinitiierte Interaktionen erfolgen über Requestobjekte, welche sich hauptsächlich in CreateRequests, GroupRequests und LocationRequests unterteilen. Deren Bearbeitung erfolgt jedoch nicht direkt durch EditParts. Stattdessen werden diese Requests an durch Rollen kategorisierte EditPolicies delegiert, welche die durch das Editieren entstehen Aufgaben übernehmen. Ein EditPart kann je Rolle maximal eine dieser wieder verwendbaren EditPolicies beinhalten, deren Zusammensetzung unter anderem die Interaktionsmöglichkeiten eines EditParts spezifiziert, das Feedback festlegt und die Weiterleitung von Requests bestimmt. Abhängig von der Implementierung der EditPolicy erzeugt diese auf Grundlage eines Requests ein entsprechendes Command, welches eine Arbeitseinheit der Laufzeitumgebung darstellt, die ausgeführt, wiederholt und rückgängig gemacht werden kann. Exemplarisch kann eine Request zum Löschen eines Elementes mit dem entsprechenden Command zur Manipulation des Modells beantwortet werden und zeitgleich das Request an die EditParts der Kinderelemente weiterleiten.

5.1.3 End-User Service Orchestration Platform

Ausgehend von Ergebnissen der untersuchten SOA-Plattformen, bildet das in Kapitel 2.1.1 beschrieben EUSOP das SOA-BackEnd von SiSeOr. Ausgehend von der grundsätzlichen Ausrichtung auf End-User Development, der flexiblen Einsatzmöglichkeit des *Generischen Clients* sowie der spezifikationskonformen Erweiterbarkeit von Metadaten stellt diese Plattform alle notwendigen Funktionalitäten bereit. Die dort realisierte Modellierung in Form eines universellen Geschäftsprozessmodells sowie die technologische Flexibilität des Generischen Clients bieten darüber hinaus die Möglichkeit, die Nutzung von SiSeOr auf andere Servicetechnologien auszudehnen. Für eine Nutzung des als Java Projekt realisierten EUSOP im Kontext von SiSeOr musste diese in die Form eines Eclipse Plug-ins überführt werden. Eine klassische Einbindung durch abgeschlossene JAR Archive wurde durch die Anforderungen der EUSOP dynamisch zu Laufzeit durchgeführte Erstellung von Web-Service-Stellvertreterobjekten verhindert. Die Überführung erfolgte unter der Voraussetzung, die Unabhängigkeit von Eclipse aufrecht zu erhalten, so dass notwendige Veränderungen am Quellcode als additive Nutzungsvarianten realisiert wurden, um somit auch weiterhin die ursprünglich Verwendung von EUSOP zu erhalten. Hierbei wurden primär laufzeitspezifische Funktionalitäten zur dynamischen Bestimmung und Verwendung notwendiger Pfadangaben sowie zur Kompilierung der Stub-Klassen ergänzt. Die genannten Anpassungen begrenzten sich auf die clientseitige Komponente EUSOP und dort konkret auf den *Generischen Client*, welcher durch die Bereitstellung dieser Klassen einen Web-Service-Zugriff per Java ermöglicht. Hingegen unverändert wird die *BPM* Komponente verwendet, welche eine sprachunabhängige Geschäftsprozessmodellierung erlaubt. Das BPM Metamodell orientiert sich dabei an dem Meta Object Facility (MOF) Standard und unterstützt in der vorliegenden Version die Transformation in lauffähigen BPEL Quellcode. Darüber hinausgehende serverseitige Komponenten, wie *UDDI*, *WSDL* und

Benutzerverwaltung wurden ebenfalls vollständig übernommen, konnten jedoch im Rahmen der Diplomarbeit bislang noch nicht an SiSeOr angebunden werden.

5.2 Architektur

Die Funktionalität des auf GMF aufbauenden SiSeOr unterteil sich in die beiden Eclipse Plug-ins *siseor.model* und *siseor.diagram*, welche die Modellebene beziehungsweise die Steuerungs- und Repräsentationsebene enthalten. Die Web Service und BPEL Technologie (vgl. Kapitel 2.1.1) wird durch EUSOP (vgl. Kapitel 2.1.3) bereitgestellt, deren Nutzung jedoch die beschriebe Überführung in Form eines Eclipse Plug-in (Kapitel 5.1.3, *eusop.plugin*) erforderlich machte. Einen Überblick über diese Grundarchitektur von SiSeOr gibt Abbildung 30, wobei die unvollständige Darstellung von Eclipse aus Übersichtlichkeitsgründen lediglich die Komponenten der RCP (Kapitel 5.1.1) beinhaltet. Dabei ist SiSeOr in der aktuellen Phase der Entwicklung als Feature, bestehend aus den drei genannten Plug-ins *siseor.model, siseor.diagram* und *eusop.plugin* realisiert

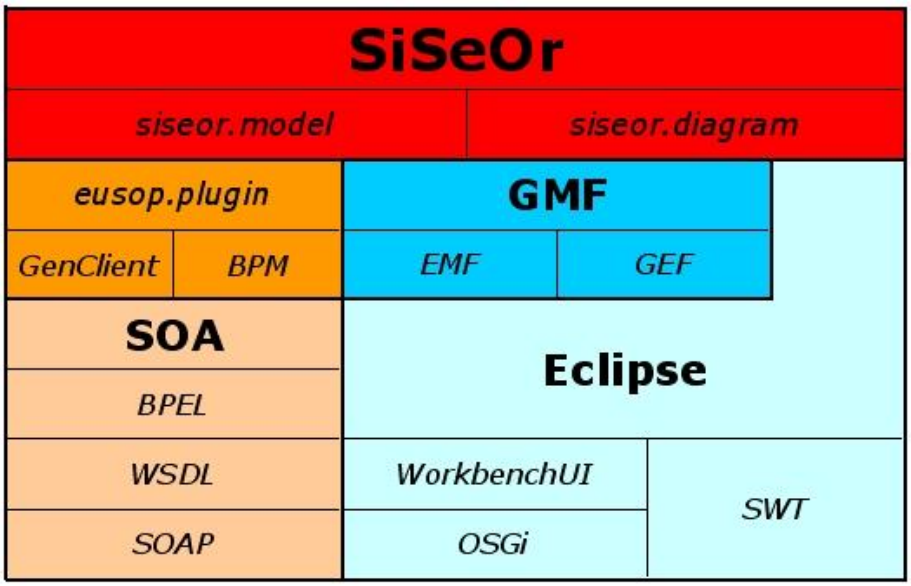

Abbildung 30: Architektur

5.2.1 Komponenten

SiSeOr setzt sich aus den Eclipse Plug-ins *siseor.model* und *siseor.diagram* zusammen, welche die angepassten und erweiterten Ergebnisse der generischen Komponente des GMF Frameworks darstellen. Das Plug-in *siseor.model* umfasst den durch EMF generierten Quellcode des Metamodells (vgl. Kapitel 5.1.2), sowie die Funktionalitäten zur Manipulation und automatisierten Serialisierung/Deserialisierung konkreter Modellinstanzen. Eine maßgebliche Anpassung stellt die gekoppelte Erzeugung des logischen Konstruktes der Web-Service-Operation dar, welche aus den essentiellen Modellelementen Rumpf, Port und WSDL zusammensetzt. Die hierfür notwendigen Informationen der WSDL-Schnittstellenbeschreibung sind durch den generischen Client von EUSOP zugreifbar. Des Weiteren beinhaltet das Plug-in die Realisierung des Visitor-Entwurfmusters in Form eines BPEL Visitors, welcher durch Modell-Traversierung die notwendigen Informationen für die Konstruktion und die anschließenden Ausgabe des Geschäftsprozesses über

die BPM-Komponente von EUSOP ermöglicht. Den größten funktionellen Anteil von SiSeOr beinhaltet das *siseor.diagram* Plug-in, welches die Steuerungsinfrastruktur und grafische Repräsentation und den größten Umfang notwendiger Modifikation enthält. Neben der Anpassung der reinen MVC Infrastruktur wurden der Entwicklungsumgebung neue Fenster (in Eclipse als Sichten bezeichnet) und deren funktionaler Unterbau hinzugefügt sowie vorhandene Fenster angepasst. Daneben wurden diverse Oberflächenelemente eingebunden, welche die direkte Nutzung gewisser Funktionalitäten von EUSOP ermöglichen oder deren Ergebnisse darstellen können. Darüber hinaus wurden verschiedene EditPolicies, welche allgemein das Editorverhalten als auch visuelles Feedback während der der Orchestrierung betreffen, angepasst oder neu erstellt.

5.2.2 Datenmodell

Das in Form des .ecore Meta-Modells verfasste grundlegende Datenmodell unterlag während der Entwicklung einem stetigen Wandel. In Abbildung 31 wird der im Vorfeld der Implementierungsphase entwickelte Entwurf desselbigen dargestellt. Hierbei bildet *Process* das Wurzelelement, welches für die Verwaltung der verwendeten *Operations* verantwortlich ist und somit die im Konzept geforderte logische Unabhängigkeit von den Web Services realisiert. Eine Operation bildet zugleich das zentrale Element der Modellierung und referenziert direkt oder indirekt *Ports* und deren Verbindungen. Zusätzlich wird hierüber eine Menge beliebiger *Descriptions* verwaltet, welche beispielsweise die WSDL Schnittstellenbeschreibung umfassen. Der Datentyp eines Ports wird hierbei durch Referenz auf eine zentrale Datentyp-Klasse beschrieben.

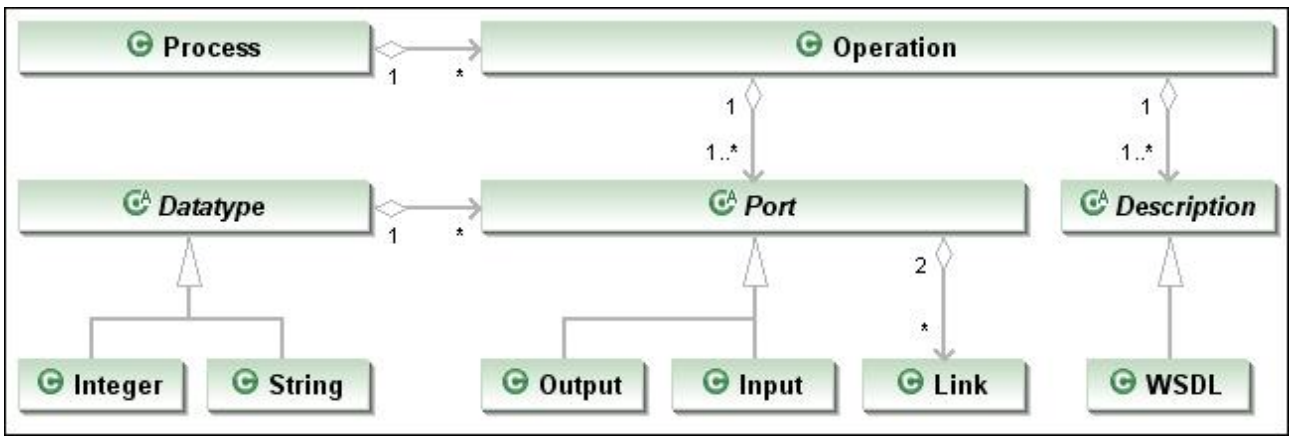

Abbildung 31: Datenmodell Entwurfsphase

Im Verlauf der Modellierung, Generierung und Implementierung wandelte sich dieses Datenmodell und wurde durch Reduzierung der verwendeten Klassen und Referenzen deutlich vereinfacht. Hierdurch konnten eine strukturell weniger umfangreiche und komplexe Bearbeitung der Modellspezifikationendateien sowie Anpassungen der Implementierung vorgenommen werden. Wesentliche Unterschiede bilden die eingeschränkte Generalisierung zwischen Eingabe- und Ausgabeport, sowie die vereinfachte Speicherung des Datentyps anhand eines String Attributes. Neu hinzugekommen sind notwendige Klassen des Visitor Entwurfsmusters, um die Traversierung des Modells zu ermöglichen. Dieses Metamodell wird voraussichtlich nicht den Ansprüchen zukünftigen Entwicklungsphasen genügen können. Mit der Zielsetzung in dem begrenzten Zeitraum ein größtmöglichen Umfang an Grundfunktionalitäten zu realisieren und durch die von der generischen GMF

Komponente unterstützte nachträgliche Anpassung von Metamodell/Repräsentation/Mapping, wurde hierdurch jedoch ein sinnvoller Kompromiss realisiert.

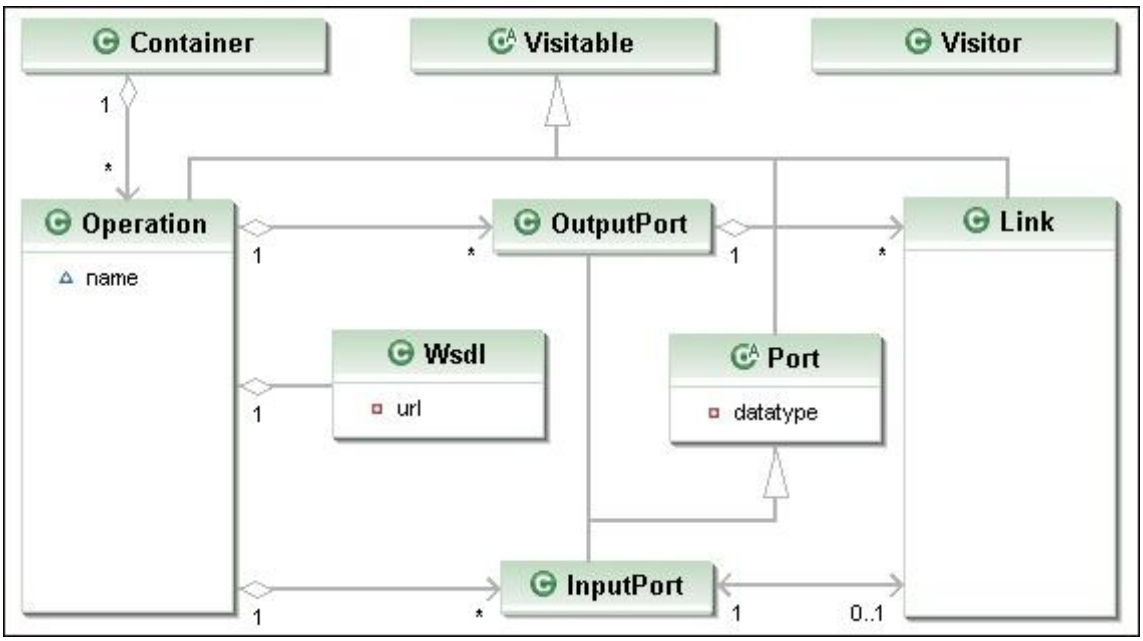

Abbildung 32: Datenmodell Implementierungsphase

5.3 Intendierte Nutzung und Anwendung

Die intendierte Nutzung des Prototyps ist die Erstellung oder Änderungen eines Prozesses durch das Entfernen oder Hinzufügen von Operationen. Hierbei können alle per WSDL-Schnittstellenbeschreibung spezifizierten Web Services verwendet und deren im Rahmen der Kommunikation mit dem Nutzer als Dienst bezeichneten Operationen datentypkonform beliebig kombiniert werden. Im Anschluss an die Fertigstellung kann der Prozess interpretiert und daraus mit Hilfe der BPM Komponenten von EUSOP der entsprechende BPEL Code erzeugt werden, welcher applikationsübergreifend eingesetzt werden kann. SiSeOr ist in der jetzigen Form noch nicht als eigenständige Rich Client Applications realisiert und kann daher in jedes beliebige Eclipse System integriert werden, welches ebenfalls die für dessen Ausführung notwendige Komponenten, wie beispielsweise GMF, enthält. Der Fokus der Darstellung liegt hierbei auf der Entwicklungsumgebung und der Repräsentation sowie den entsprechend veränderten und ergänzten Funktionalitäten. Für das Verständnis der grundsätzlichen Funktionsweise eines auf GMF basierten grafischen Editors bieten die Arbeiten von Budinsky et al. [BSM'03] und Moore et al. [MDG'04] sowie die im Internet erreichbare GMF-spezifische Eclipse Dokumentation [URL,f] eine breite Informationsbasis. Die nun anschließende Erläuterung der Oberfläche und deren Funktionalitäten orientiert sich dabei an dem chronologischen Ablaufs einer typischen Anwendung.

5.3.1 Start

Innerhalb der jeweiligen Eclipse Plattform kann mit Hilfe eines entsprechenden Symbols oder Menüeintrages die SiSeOr Perspektive ausgewählt werden, welche die entsprechenden Fenster

(Sichten) der SiSeOr Entwicklungsumgebung einblendet und in Abbildung 33 dargestellt ist. Neben dem im Weiteren beschriebenen Suchfenster, der Werkzeugpalette und dem eigentlichen grafischen Editor wird zusätzlich ein Konsolenfenster eingeblendet, welches dem Nutzer textuelles Feedback über die im Hintergrund ablaufenden und nur schwerlich visuell darstellbaren Operationen gibt. Funktionell stellt dies lediglich eine Umleitung der Ausgabe des *System.out.print* Befehls dar, hat im Gegensatz zu einem Popup jedoch den Vorteil, die Arbeit des Nutzers nicht durch eine erzwungene Fokusverschiebung bei der eigentlichen Bearbeitung eines Prozesses zu unterbrechen. Darüber hinaus bietet das im Rahmen der Generierung erzeugte *Outline* Fenster eine wahlweise grafische und baumartige Übersicht des aktuellen Status des Prozessmodells. Des Weiteren steht der durch die Eclipse Umgebung bereitgestellte und unverändert genutzte *Projekt Browser* zur Verfügung, welcher die Navigation innerhalb der als Container genutzten Projektkonstrukte und deren beinhalteten Dateien erlaubt. Bei der Absicht, einen neuen Prozess zu erstellen, umfasst der erste Bedienschritt die Erstellung eines solchen Projektes und das anschließende Anlegen einer neuen Diagram Datei (*.sdf*, SiSeOr Diagram File). Das Öffnen derselbigen initiiert den Start des grafischen Editors und befüllt gleichzeitig die bis zu diesem Zeitpunkt leere Werkzeugpalette mit den Symbolen der entsprechenden Grundfunktionen.

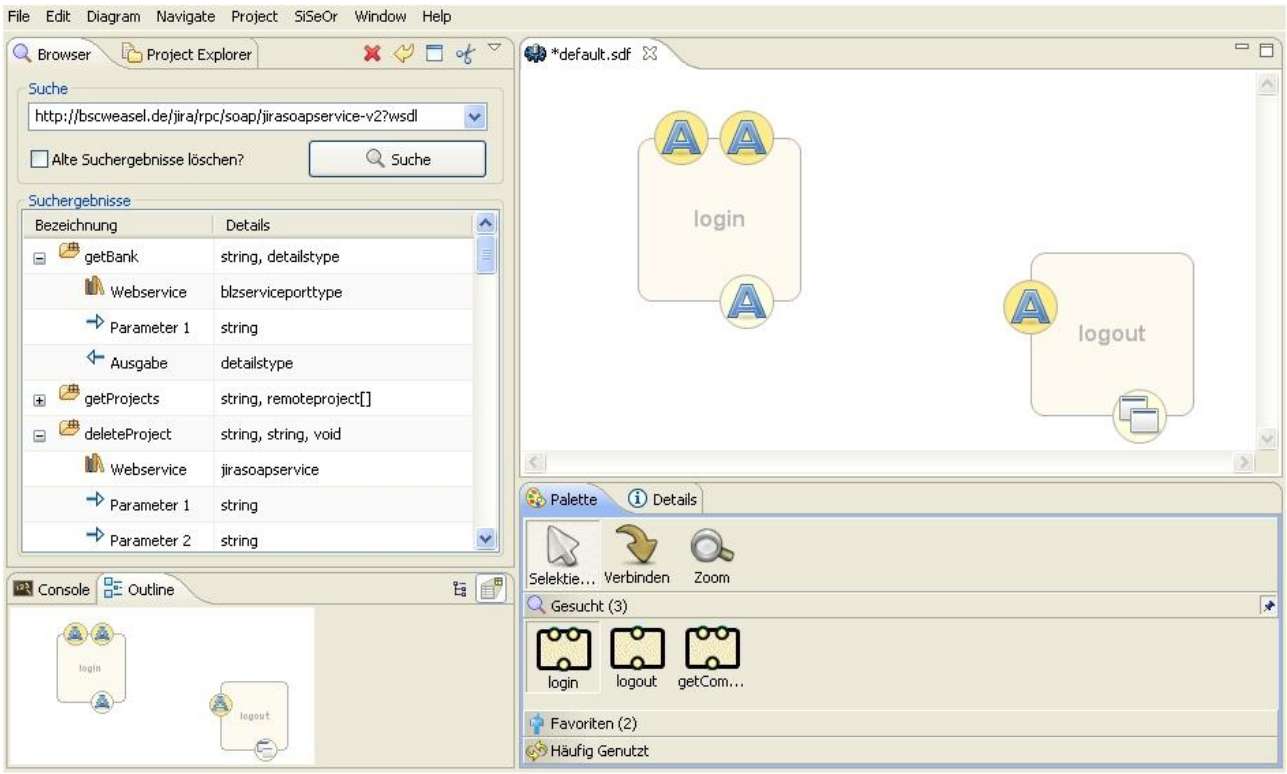

Abbildung 33: SiSeOr Entwicklungsumgebung

5.3.2 Suche

Das sich am linken Bildschirmrand befindende Suchfenster (siehe Abbildung 33) ist durch die bislang nicht realisierte Anbindung an den jUDDI Server von EUSOP auf die manuelle Angabe einer lokalen oder im Internet publizierten WSDL Schnittstellenschreibung abhängig. Nach betätigen der Suchtaste werden mit Hilfe des generischen Clients von EUSOP im Hintergrund die entsprechenden Stellvertreterklassen und –objekte erzeugt und die relevanten Information in einer

durch Icons visuell unterstützen baumartigen Darstellung angezeigt. Dabei beschränken sich die noch nicht um zusätzliche Metadaten erweiterte Schnittstellenbeschreibungen auf die verfügbaren und für Nutzer relevanten Informationen. Lediglich der Name der Dienstes, den Ein- und Ausgabeports samt deren Datentypen sowie die Angabe des Web Services werden angezeigt. Dem Nutzer steht nun die Möglichkeit offen, aus dieser Liste unerwünschte Dienste zu entfernen, um beispielsweise eine verbesserte Übersichtlichkeit bei Web Services mit einer großen Anzahl von Operatione zu ermöglichen. Darüber hinaus kann diese Liste durch Angabe weiterer WSDL Dokumenten erweitert werden. Aus dieser nach den Wünschen des Nutzers veränderten Auflistung können jederzeit das jeweilige Werkzeug zur Erstellung desselbigen der Palette hinzugefügt werden, um somit während der Orchestrierung zur Verfügung stehen. Die derart übernommenen Elemente werden zeitgleich aus der Ergebnisliste entfernt, um den Nutzer durch die Redundanz nicht zu verwirren und eine bessere Übersichtlichkeit zu schaffen.

5.3.3 Werkzeugpalette

Die durch die generische Komponente von GMF erzeugte Factory-Klasse zur Erzeugung einer Werkzeugpalette wurde grundlegend überarbeitet. Maßgeblich ist die Möglichkeit, dieser während der Laufzeit dynamisch neue Werkzeuge hinzufügen zu können, welche den bearbeiten Prozess um ein konkretes Dienstelement erweitern. Diese sind zur besseren Abgrenzung mit einem passenden Icon versehen, welches die jeweilige Anzahl der Ports skizziert. Die Palette (vgl. Abbildung 33) verfügt neben einem statischen Bereich mit den Grundbefehlen Selektieren, Verbinden und Zoomen über drei dynamische Bereiche. Innerhalb der Palette können die Werkzeuge zur Erzeugung von Dienstelementen einem oder mehreren dieser dynamischen Bereiche zugeordnet werden, wobei aufgrund der Suchergebnisse neu hinzugefügt Dienstwerkzeuge dem Bereich *Gesucht* zugewiesen werden. Darüber hinaus existiert ein Bereich für individuell zusammenstellbare *Favoriten* und für die noch nicht realisierte Funktionalität situationsabhängiger *Empfehlung* konkreter Dienste bzw. dessen Werkzeugen. Die Ausführung eines Grundbefehls oder die Erzeugung eines Dienstelementes wird nach vorheriger Selektion des entsprechenden Werkzeuges durch einen Klick innerhalb des Modellierungsbereichs erreicht, wobei ein Symbol die positionsabhängige Gültigkeit dieser Funktion anzeigt.

5.3.4 Editor und Repräsentation

Die Repräsentation entspricht im Allgemeinen dem in Kapitel 4.6.1 beschriebenen Konzept und wird durch den beispielhaften Teil-Prozess in Abbildung 34 verdeutlicht. Zur besseren Orientierung werden die jeweiligen Datentypen der Ports durch Icons symbolisiert und sind bislang in unverknüpften Zustand ausschließlich farblich zu unterscheiden. Durch die fehlenden UDDI Anbindung fehlt bislang die im Konzept beschriebene Möglichkeit eines dienstindividuell verknüpfbaren Icons. Die erfolgreiche Verbindung zwischen zwei Ports wird in der Repräsentation durch eine grüne Hinterlegung derselbigen angezeigt, wobei im Fall von Ausgabeports zusätzlich der Rand abhängig von der Anzahl der Verbindungen im Umfang zunimmt. Eine Verbindung kann jedoch ausschließlich zwischen einem Output- und einem Inputport geschlossen werden, wenn diese zusätzlich zu unterschiedlichen Diensten gehören, wobei die Information über die Gültigkeit einer

Verbindung während des konkreten Verbindungsversuchs durch ein entsprechendes Symbol angezeigt wird.

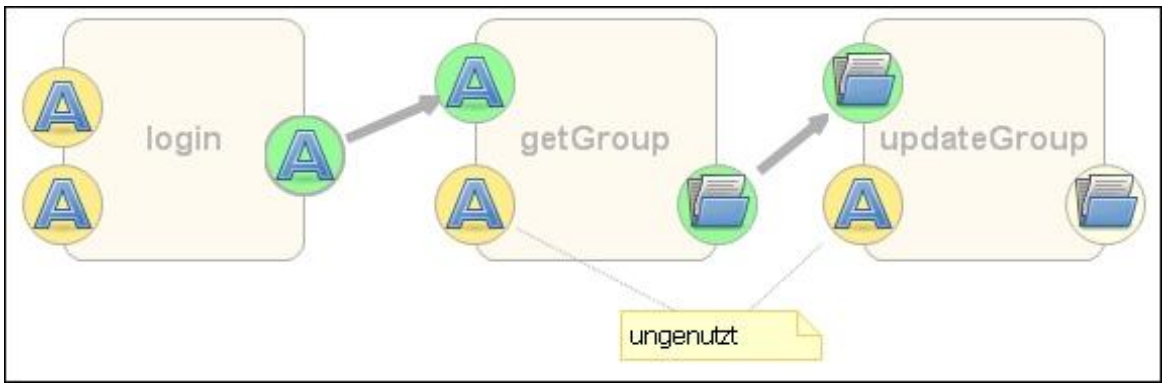

Abbildung 34: Repräsentation und Notation

Weitere in Form von EditPolicies hinterlegte Rahmenbedingungen sind beispielsweise die farbliche Hervorhebung der aktuell mit dem Mauszeiger überfahrenen Dienst-Repräsentation und die Unterbindung, dass einzelne Ports eines Dienstelementes separat gelöscht werden können. Daneben erlaubt eine freie Verschiebbarkeit einzelner Datenports auf der Randmarkierung der jeweiligen Dienstrepräsentation eine beliebe Anordnung, welche beispielsweise horizontal oder vertikal orientiert sein kann. Zu den relevanten aber nicht angepassten Funktionalitäten, welche durch das GMF Framework generiert wurde, zählt die Möglichkeit, Annotationskonstrukte mit individuellem Textinhalt mit einer beliebigen Anzahl von Elementen zu verbinden (vgl. Abbildung 34). Darüber hinaus wird ein automatisches Routing für die visuell dargestellten Verbindungen genutzt, welches jedoch durch das zusammengesetzte Konstrukt des Dienstes nur suboptimale Ergebnisse hervorbringt. Die gleiche Problematik liegt der ebenfalls generierten Funktionalität der automatischen Ausrichtung der Prozessrepräsentation zugrunde, welche daher aktuell nur bedingt sinnvoll einsetzbar ist.

5.3.5 BPEL-Prozessbeschreibung

Nach Abschluss des Orchestrierungsprozesses kann der Nutzer im letzten Schritt die Generierung der entsprechenden BPEL-Prozessbeschreibung auslösen. Der im Vorfeld bereits erwähnte Visitor durchläuft hierbei das Modell, wobei in diesem Entwicklungstand von der Annahme ausgegangen wird, dass ein Dienst ohne verknüpfte Eingangsports den jeweiligen Startdienst darstellt. Aufgrund dieser Daten wird mit Hilfe der BPM Komponente und dessen Metamodell der Prozess abgebildet, welches im Anschluss in Form eines Popups ausgegeben wird.

6 Zusammenfassung und Ausblick

Dieses abschließende Kapitel fasst den Inhalt und das Vorgehen diese Arbeit kompakt zusammen und gibt ein abschließendes Fazit. Hieran schließt ein Ausblick über anschließende Arbeiten an, welche zum einen die Realisierung weiterer Aspekte der Konzeption sowie deren punktuelle Erweiterung umfassen. Den Abschluss bildet eine kritische Reflektion über dieser Arbeit zugrunde liegenden Methodik.

6.1 Zusammenfassung

Die aus Wettbewerbsgründen notwendige Flexibilität von Unternehmen kann nur durch eine aktive Einbeziehung der Endbenutzer in eine evolutionäre Entwicklung der informationstechnisch unterstützen Geschäftsprozesse erfolgen. Serviceorientierte Architekturen bieten hierfür die notwendige technische Anpassungsfähigkeit, deren funktionale Gestaltung und Anpassung in Form von BPEL Prozessen jedoch durch das umfangreich vorausgesetzte technische Wissen für Endbenutzer nicht zugänglich ist. Allgemeiner formuliert fehlt es EUD-Anwendungen anderer Technologieformen an technischer Flexibilität und den Produkten servicebasierter Editoren an Einfachheit bzw. Kompatibilität und Mächtigkeit. Diese Ausgangsbasis bildete die Motivation der Fragestellung, wie eine servicebasierte Tailoringmöglichkeit für Endbenutzer auf Ebene der informationstechnischen Geschäftsprozessunterstützung zu realisieren ist. Das entwickelte Konzept wurde folglich aus einer endbenutzer-zentrierten Gegenüberstellung des zu investierenden Aufwandes und der hieraus erreichbaren Nutzenstiftung entwickelt. Aufgrund der für Endbenutzer absolut neuartigen Technologie und Nutzungsform und der relativ starken Beeinflussung von Nutzern durch bestehende Konzepte wurde eine praktische Vorstudie in Form eines Participatory Design Workshops durchgeführt. Um möglichst unbeeinflusste Ergebnisse zu bekommen, wurden hierbei nur die absolut elementaren Informationen eines vorher entwickelten Grundkonzeptes verwendet. Die Analyse der geleiteten und eigenständig durchgeführten Abschnitte des Workshops, konnte allgemeine Zugänglichkeit der Orchestrierung servicebasierter Funktionskomponenten bestätigen und wertvolle Hinweise für die weiterführende Konzeption liefern.

Diese Konzeption basiert auf einer grafischen Orchestrierung, welche das Prinzip der *Box and Wire* Metapher verwendet. Diese Form der visuellen Programmierung ermöglicht durch zusätzliche Abstraktion, Vereinfachung und der Integration impliziter Syntaxregeln die Reduzierung der technischen Komplexität in nicht unerheblichem Maße. Die logische Entkopplung von Operationen und den jeweiligen Web Services setzt diese Reduzierung auf die essentiellen Elemente einer Orchestrierung weiter fort. Diese nur schwerlich weiter reduzierbare Essenz muss jedoch auf der anderen Seite zusätzliche Informationen in einer endbenutzergerechten Form beinhalten, um eine Verwendung einzelnen Operationen für die Anwender überhaupt erst möglich zu machen. Eine spezifikationskonforme Erweiterung der durch die WSDL Schnittstellenbeschreibung bereit gestellten Metadaten, welche durch Nutzer beliebig ergänzt und erweitert werden können, bildet diese fehlende Informationsbasis ab. Diese unterstützend werden neben der durch die Nutzergemeinschaft veränder- und zugreifbaren Informationen zusätzlich weitere kollaborative Aspekte unterstützt. Wesentlich ist hierbei die Möglichkeit, durch ein zentral zugreifbares UDDI den Austausch von or-

chestrierten Prozessen zu erlauben. Die kollaborativ erstellten und automatisch generierten Informationen der erweiterten Schnittstellenbeschreibungen dienen darüber hinaus als Basis einer Empfehlungsfunktionalität, welche beispielsweise abhängig von Bewertung und Nutzungshäufigkeit dem Anwender Dienste oder Prozesse vorschlägt. Um das Verständnis, den Lernaufwand, die Bedienbarkeit und nicht zuletzt die Motivation zusätzlich positiv zu beeinflussen, wurde das Konzept um eine domänenspezifische Konfigurierbarkeit erweitert, welche das Aufgreifen von Begrifflichkeiten und Metaphern der Arbeitspraxis ermöglicht. Neben dieser Konfiguration auf Gruppenebene verfügt SiSeOr zusätzlich über eine breite Anpassungsmöglichkeit, um auch individuelle Präferenzen in der Bedienung und Darstellung umsetzen zu können. Weiterhin dienten Konzepte des Natural Programming und diverser anderer psychologischer Bereiche als Informationsquelle, um sowohl funktionelle als auch visuelle Elemente möglichst intuitiv und zweckdienlich zu gestalten. Ergebnis dieser Maßnahmen waren unter anderem eine differenzierte und leicht zugängliche Lernunterstützung und die Bereitstellung verschiedener Ebenen der Nutzungskomplexität. Eine im Anschluss prototypisch realisierte Umsetzung konzentrierte sich auf die Implementierung der Grundfunktionalitäten und baut technisch auf der *End User Service Orchestration Platform* sowie auf dem Eclipse *Grafical Modeling Framework* auf. Die von SiSeOr ausschließlich spezifikationskonforme Verwendung von Schnittstellen- und Prozessbeschreibungen erlauben hierbei eine umfangreiche Flexibilität bei der Auswahl der orchestrierten Dienste und Nutzung der generierten BPEL Prozessbeschreibungen.

Das immense Potential eines in der Praxis einsetzbaren Tailoringsystems, besonders im Hinblick auf die positiven ökonomischen Effekte, verdeutlicht den Stellenwert dieser Forschungsrichtung. Allgemein mangelt es trotz immensen Forschungsleistungen aktuellen Umsetzungen im Rahmen des EUD hauptsächlich an einer Verknüpfung der Aspekte Einfachheit, Flexibilität und Mächtigkeit. Da im Rahmen dieser Diplomarbeit methodisch auf eine abschließende Evaluation zu Gunsten einer nutzerbeteiligten Vorstudie verzichtet wurde, kann das praktische Resultat dieser Arbeit nur im begrenzten Umfang bewertet werden. Offene Fragen bestehen aktuell primär in der Praxistauglichkeit des von der Mitarbeit der Nutzer abhängigen Prinzips der Pflege und Erweiterung der Metadaten sowie die geeignete Navigation und Darstellung einer logischen Hierarchie von Prozessen und deren funktionalen Bestandteilen. Darüber hinaus muss beispielsweise untersucht werden, inwieweit die fast ausschließliche Nutzung datenflussgesteuerter Konstrukte eine ausreichende Flexibilität und Mächtigkeit orchestrierter Prozesse erlauben oder wie geeignet Eingabe- und Ausgabeelemente erzeugt und angebunden werden können.

Die Leistung dieser Arbeit kann aufgrund der zeitlichen Begrenzung weniger an der prototypischen Umsetzung der grundlegenden Funktionalitäten gemessen werden. Vielmehr drückt sich diese einerseits in der strukturierten Darstellung und Betrachtung der im Kontext einer servicebasierten Geschäftsprozessmodellierung für Endbenutzer relevanten Faktoren aus. Die eigentliche Essenz dieser Arbeit bildet jedoch die Entwicklung eines auf diesen Erkenntnissen aufbauenden Konzeptes, in der technische sowie menschliche Anforderungen gleichberechtigt und konfliktfrei berücksichtigt wurden. Neben der Abstraktion technischen Gesichtspunkte bildet hauptsächlich die Orientierung an kognitiven Aspekten die Grundlage des positiven Einflusses auf Lernaufwand, Bedieneffizienz und die Gestaltung der Mensch-Maschine-Kommunikation. Darüber hinaus bilden

die Charakteristiken kognitiver Modelle die Vorlage, um die Orchestrierung optimal an die natürliche Kreativitäts- und Problemlösungsprozesse von Endbenutzer anzupassen und diese zeitgleich durch das System zu unterstützen. Darüber hinaus hat die Berücksichtigung des Anwendungskontexts, der sich aus jeweiligen Domäne und den allgemeinen sowie speziellen Formen der Kollaboration zusammensetzt, einen großen Einfluss. Zusammenfassend bietet SiSeOr somit erstmals Endbenutzern einen innovativen und aufwandsminimalen Zugang zur serviceorientierter Technologien und deren Flexibilität, welcher bislang nur durch die Aneignung umfangreichen technischen Wissens möglich war.

6.2 Ausblick

Neben den in der Zusammenfassung genannten Fragen, welche die Weiterentwicklung des Konzeptes betreffen, steht in erster Linie eine Fortführung der prototypischen Realisierung und dessen Evaluation. Der nächste Schritt stellt dementsprechend die Anbindung der EUSOP Komponenten WSDL und UDDI dar. Damit einher geht die Umsetzung einer Suchfunktion sowie entsprechender Oberflächenelemente zur Anzeige und Bearbeitung der Metadaten. Der nächste Schritt umfasst die Realisierung der genannten Möglichkeiten zur Unterstützung des Lernprozesses, sowie die Spezifizierung verschiedener Komplexitätsstufen, welche unter anderem auch von der logischen Nähe einzelner Funktionalitäten zur Domäne des Nutzers abhängen. Parallel hierzu muss die beschriebene Möglichkeit zur Fehlersuche und einer entsprechenden Explorationsumgebung realisiert werden, um so die grundsätzliche Basis eines offenen Lernprozesses für den Anwender zu bieten. Darüber hinausgehende Maßnahmen betreffen unter anderem eine umfassende Einbindung der in Form der Metadaten gesammelten Informationen in Entwicklungsumgebung und Modell, sowie die Konfigurierbarkeit domänenspezifischer Gesichtspunkte. Über diese unvollständige Aufzählung hinaus kann der Nutzen und die Weiterentwicklung von SiSeOr durch die Anbindung softwareunabhängiger Projekten weiter optimiert werden. In diesem Zusammenhang ist zum einen CHiC (Community Help in Context) [SW'06; Wie'06] zu nennen, welches ein in der Konzeption beschriebenes wiki-basierters Hilfesystem zur Verfügung stellt. Daneben würde eine Verbindung mit dem als Eclipse Plug-in verfügbare PaDU (Participatory Design in Use) [Dra'07] die evolutionäre Weiterentwicklung von SiSeOr vereinfachen, da es eine in die Arbeitsoberfläche integriertes Werkzeug zur nutzerbeteiligten Weiterentwicklung im Kontext der Nutzung bereit stellt. Neben demn im Konzept dargestellten Ideen birgt die Möglichkeit der gemeinsamen Bearbeitung eines Prozess zusätzliches Potential. Dies ist zum einen durch den großen Stellenwert der Kollaboration innerhalb der allgemeinen Arbeitspraxis motiviert, so dass in Folge trotz einer etwaigen räumlichen Trennung dieses Arbeitsprinzip auch während der Modellierung ermöglicht würde. Zum anderen sind in die Ausführung eines Geschäftsprozessen häufig mehrere Personen involviert, so dass hier auch eine individuelle Einflussnahme möglich sein sollte. Basierend auf der grundsätzlichen Motivation von EUD (vgl. Kapitel 1) ist längerfristig die Umwandlung der Editorfunktionalität in eine orchestrierbare Form erstrebenswert, die in Anlehnung an *Squeak*, mittels der eigenen Orchestrierungsfunktionen selbst verändert werden könnte.

6.3 Reflektion: Methodik und Technologie

Die Durchführung der aufwandsintensiven Vorstudie in Form eines PD Workshops kann nach Abschluss der Arbeit als grundlegend richtige Entscheidung bewertet werden. Erst die direkte Auseinandersetzung mit Nutzern, die gemeinsame Entwicklung der Definition eines Dienstes und die Analyse der eigenständigen Arbeit mit der Mock-ups komplettierte die notwendige Wissensbasis der weiterführenden Konzeption. Neben Ergebnissen im Hinblick auf die Editorkonzeption konnte das Werkzeug PD unter erschwerten Rahmenbedingungen auf seine Praktikabilität hin untersucht werden. Diese Rahmenbedingungen setzten sich aus der Begrenzung auf einen Workshoptag und der Kommunikation einer für die Teilnehmer völlig neuen Technologie zusammen. Die Erwartungen an die praktische Durchführung des Konzepts wurden grundsätzlich erfüllt, wobei sich besonders die gemeinsame Erarbeitung von Arbeitsszenarien und Technologie-Definition als wichtiges Instrument zur Reduzierung potentieller Kommunikationsschwierigkeiten zwischen Nutzern und Technikern herausstelle. Die deutlichste Schwäche des Konzeptes bildete die praktisch nicht realisierbare Zeitplanung. Trotz einer Verlängerung auf insgesamt sechs Stunden reiner „Arbeitszeit“ konnte keine befriedigende Möglichkeit zur eigenständigen Gestaltung der eigentlichen Editor-Umgebung durch die Teilnehmer erfolgen, so dass die Konzeption in dieser Hinsicht angepasst werden muss. Die Dokumentation des Workshops durch Audio und Videomaterial konnten die relevanten Informationen für die Analyse in ausreichender Qualität bereitstellen, wobei diese jedoch zusätzlich durch selektiv aufgenommene Fotografien von Tafelbild und Mock-up Arbeitsplatz ergänzt wurden. Abschließend sei noch bemerkt, dass situationsabhängige Abweichungen von den passiveren Teilnehmerrollen innerhalb des Workshops den stattgefundenen Diskussionen wichtige Impulse lieferten. Neben der Reflektion der Methodik kann Eclipse als Entwicklungsumgebung und Plattform grundsätzlich als empfehlenswert bezeichnet werden und bestätigt somit dessen breite Akzeptanz und Einsatz in der Praxis. Das seit 2006 verfügbare GMF Framework bietet durch die generative Überbrückung von EMF und GEF und die „out-of-the-box“ generierten Grundkomponenten und -funktionalitäten eines grafischen Editors eine deutliche Reduzierung der Entwicklungszeit, erfordert jedoch eine teilweise aufwendige Einarbeitung in die Erstellung der Modellspezifikationen sowie der grundlegenden GEF Infrastruktur. Dies wird durch das Fehlen einer zentral verfügbaren und umfassenden Dokumentation, welche bei EMF [BSM'03] und GEF [MDG'04] beispielsweise bereits in Buchform vorliegt, erschwert. Eine Schwierigkeit bei der Spezifizierung der verschiedenen Modellspezifikationsdateien stellt der Informationsgehalt der auftretenden Fehlermeldungen dar, welcher die Behebung deutlich erschwert. Hingegen kann die teilweise aufwendige Erstellung und Veränderung des im Ecore formulierten Datenmodells durch die Nutzung der von IBM entwickelten Emfatic Sprache umgangen werden. Diese ermöglicht eine Repräsentation des Ecore Modells in einer Java-ähnlichen Syntax, deren bi-direktionale Beziehung und die schnelle Anpassung innerhalb der Emfatic Sprache eine deutliche Vereinfachung der Erstellung und Anpassung ermöglicht.

Danksagung und Nachwort

Zunächst möchte ich meiner Familie, meinen Freunden und im Besonderen meiner Ehefrau Mira danken, ohne deren Unterstützung die Realisierung dieser Arbeit nicht möglich gewesen wäre. Darüber hinaus möchte ich ganz besonders Prof. Dr. Pipek und Prof. Dr. Wulf des Lehrstuhls Wirtschaftsinformatik und Neue Medien der Universität Siegen danken, welche mir diese Arbeit ermöglicht haben. Für die Hilfe, die wertvolle Zusammenarbeit und das Aufzeigen alternativer Blickwinkel gebührt meinem Betreuer Dipl-Wirt.-Inf. Christian Dörner ein ganz spezieller Dank. Ebenfalls an dieser Stelle möchte ich Michael Veith, Sebastian Draxler, Claudia Müller, Michel Claussmann und wiederum Christian Dörner erwähnt wissen, ohne deren Einsatz und Mühe die Durchführung des Participatory Design Workshops nicht möglich gewesen wäre. Abschließend möchte Moritz Weber meinen Dank aussprechen, welcher mir als „Vater des Generischen Clients“ während der Überführung der EUSOP Funktionalität wertvolle Hinweise gab und tiefe Einblicke in dessen Funktionsweise gewährte.

Allgemein stellt Verbesserung und Vereinfachung die eigentliche Triebfeder meines Interesses für die Informationstechnologie und mein Studium dar. Diese innere Bestrebung, Optimierungspotentiale zu nutzen, führte auch im Rahmen meiner Diplomarbeit zu Ideen, die ich dem geneigten Leser nicht vorenthalten möchte. Im Zuge der Literaturrecherche wurde die schwierige Überschaubarkeit und Transparenz relevanten Arbeiten zur Entwicklung eine EUD Systems deutlich, welches wohl auch durch die große Anzahl für die EUD wichtiger Forschungsbereiche resultiert. Daher stellen Arbeiten, die den jeweils aktuellen Forschungsstand sammeln, strukturieren und bewerten, nicht zu unterschätzende Zwischenergebnisse des teilweise chaotisch ablaufenden Forschungsprozesses dar. Es fehlen Arbeiten wie sie Schiffer [Sch'98] und Green [GP'96] in ihren jeweiligen Gebieten angefertigt haben, um ausgehend von eine klaren Standortbestimmung und Wissenskondensation die aktuellen Herausforderungen deutlicher erkennen zu können. Darüber hinaus sollten gerade in der Erforschung einer Verbesserung der Mensch-Maschine-Kommunikation und der damit direkt verbundenen Ausweitung des Unterstützungspotential des „Universal Werkzeugs“ Computer, die Möglichkeiten einer computerunterstützten Zugänglichkeit und Strukturierung schriftlich fixierten Forschungswissens berücksichtigt werden. Besonders das Potential kollaborativer Aspekte und nutzergenerierter Inhalte nach dem Vorbild webbasierter Nutzergemeinschaften könnten wertvolle Impulse für eine größer Transparenz und damit einer zielgerichteteren Forschung schaffen.

Abkürzungsverzeichnis

ACM CRS	Association for Computing Machinery Computing Review System
Ajax	Asynchronous JavaScript and XML
BPEL	Business Process Engineering Language
BPML	Business Process Markup Language
BPXL	Business Process eXtension Layers
CAT	Component Architecture for Tailorability
CORBA	Common Object Request Broker Architecture
DCAT	Distributed Component Architecture for Tailorability
DEMOS	Democratic Planning and Ccontrol of Working Life
ebXML BPSS	Electronic Business XML, Business Process Specification Schema
EMF	Eclipse Modeling Framework
ERP	Enterprise Ressource Planning
ESOA	Enterprise SOA
EUC	End-User Computing (wird häufig synonym zu EUD verwendet)
EUD	End-User Development
EUSOP	End User Service Orchestration Plattform
GEF	Grafical Editing Framwork
GMF	Grafical Modeling Framework
GML	GUI Modeling Language
GUI	Grafical User Interface
HANDS	Human-centered Advances for Novice Development of Software
HCI	Human-Computer Interaction
HTTP	Hypertext Transfer Protocol
LGPL	GNU Lesser General Public License
OASIS	Organization for the Advancement of Structured Information Standards
OSCI	Online Services Computer Interface
PBD	Programming by Demonstration
PBE	Programming by Example
PICTIVE	Plastic Interface for Collaborative Technology Initiative through Video Exploration
PKI	Public Key Infrastructure
NJMF	Norwegian Iron and Metal Workers Union
QEDWiki	Quick and Easily Done Wiki
SCA	Service Component Architecture
SCDL	Service Component Definition Language
SDO	Service Data Objects
SED	Seeding, Evolutionary Growth and Reseeding
SiSeOr	Editorbezeichnung, Akronym für *Simple Service Orchestration*
SOA	Serviceorientierte Architektur (engl. Service Oriented Architecture)
SOAP	Eigenwort, anfänglich Akronym für *Simple Object Access Protocol*
SSCL	Simple Service Composition Language
UDDI	Universal Description, Discovery and Integration

WC3	World Wide Web Consortium
WSDL	Web Service Description Language
WYSIWYG	What You See Is What You Get
XML	Extensible Markup Language
XML-RPC	Extensible Markup Language Remote Procedure Call
WS-CDL	Web Services Choreography Description Language
WSCI	Web Service Choreopraphy Interface

Abbildungsverzeichnis

Literaturverzeichnis

[AC'04]
Alda, S. and Cremers, A.B. (2004) "Towards Composition Management for Component-based Peer-to-Peer Architectures".

[AKC'07]
Alda, S.; Kuck, J. and Cremers, A.B. (2007) "Tailorability of personalized BPEL-based Workflow Compositions"
IEEE Congress on Services, Salt Lake City, USA, 2007.

[Ala'06]
Alanen, M. (2006) "EMF/ECORE Integration in the Coral Modeling Framework"
Proceedings of the 4th Nordic Workshop on UML and Software Modelling, Grimstad, Norwegen, 2006.

[Ald'06]
Alda, S. (2006) "Component-based Adaptation Methods for Service-Oriented Peer-to-Peer Software Architectures"
Dissertation: Rheinische Friedrich-Wilhelms-Universität Bonn, Deutschland.

[B4P]
Spezifikation: WS-BPEL Extension for People (1.0), u.a. IBM, Oracle, SAP
(http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/, Juni 2008).

[BB'93]
Brancheau, J.C. and Brown, C.V. (1993) "The Management of End-User Computing: Status and Directions"
in ACM Computing Surveys (CSUR), Jahrgang 25, Ausgabe 4.

[BB'94]
Burnett, M.M. and Baker, M.J. (1994) "A Classification System for Visual Programming Languages"
in Journal of Visual Languages and Computing, Jahrgang 5, Ausgabe 3.

[BB'05]
Barcia, R. and Brent, J. (2005) "Building SOA solutions with the Service Component Architecture"
(http://www-128.ibm.com/developerworks/websphere/techjournal/0510_brent/0510_brent.html, Juni 2008).

[BBB'05]
Beisiegel, M.; Blohm, H.; Booz, D. and Al., E. (2005) "Service Component Architecture - Building Systems using a Service Oriented Architecture".

[Bec'02]
Beck, E.E. (2002) "P for Political - Participation is Not Enough"
in Scandinavian Journal of Information Systems, Jahrgang 2002, Ausgabe 14.

[BEK'87]
Bødker, S.; Ehn, P.; Kammersgaard, J.; Kyng, M. and Sundblad, Y. (1987) "A Utopian Experience"
in Bjerknes, G.; Ehn, P. and M., K. (1987) "Computers and democracy: A Scandinavian challenge", Avebury.

[BF'97]
Butler, T. and Fitzgerald, B. (1997) "A case study of user participation in the information systems development process"
Proceedings of the eighteenth international conference on Information systems, Atlanta, USA, 1997.

[BG'99]
Blackwell, A.F. and Green, T.R.G. (1999) "Does Metaphor Increase Visual Language Usability?"
Proceedings of the IEEE Symposium on Visual Languages, 1999.

[BGK'93]
Bødker, S.; Grønbaek, K. and Kyng, M. (1993) "Cooperative Design: Techniques and Experiences From the Scandinavian Scene"
in Douglas, S. and Aki, N. (1993) "Participatory Design: Principles and Practices", Lawrence Erlbaum Associates Incorporated.

[BH'94]
Baroth, E. and Hartsough, C. (1994) "Experience Report: Visual Programming in the Real World".

[Bla'06a]
Blackwell, A.F. (2006a) "Psychological Issues in End-User Programming"
in Lieberman, H.; Paternò, F.; Klann, M. and Wulf, V. (2006a) "End User Development", Springer.

[Bla'06b]
Blackwell, A.F. (2006b) "The Reification of Metaphor as a Design Tool"
in ACM Transactions on Computer-Human Interaction (TOCHI), Jahrgang 13, Ausgabe 4.

[Bød'91]
Bødker, S. (1991) "Through the Interface: A Human Activity Approach to User Interface Design", Lawrence Erlbaum Associates Incorporated.

[Bød'96]
Bødker, S. (1996) "Creating conditions for participation: Conflicts and resources in systems design"
in Human Computer Interaction, Jahrgang 11, Ausgabe 3.

[BP'99]
Blackwell, A.F. and Petre, M. (1999) "Mental Imagery in Program Design and Visual Programming"
in International Journal of Human-Computer Studies, Jahrgang 51, Ausgabe 1.

[BPEL]
Spezifikation: BPEL (1.1), OASIS
(http://www.oasis-open.org/committees/wsbpel, Juni 2008).

[BPMN]
Spezifikation: BPMN (1.0), OMG
(http://www.bpmn.org, Juni 2008).

[BPS'06]
Berti, S.; Paternò, F. and Santoro, C. (2006) "Natural Development of Nomadic Interfaces Based on Conceptual Descriptions"
in Lieberman, H.; Paternò, F.; Klann, M. and Wulf, V. (2006) "End User Development", Springer.

[BRC'06]
Burnett, M.; Rothermel, G. and Cook, C. (2006) "An Integrated Software Engineering Approach for End-User Programmers"
in Lieberman, H.; Paternò, F.; Klann, M. and Wulf, V. (2006) "End User Development", Springer.

[Bro'87]
Brooks, F.P.J. (1987) "No silver bullet: essence and accidents of software engineering"
in Computer, Jahrgang 20, Ausgabe 4.

[BSI'02]
Bundesamt für Sicherheit in der Informationstechnik (BSI) (2002) "Sicherheitsbewertung zur Spezifikation OSCI – Transport 1.2".

[BSM'03]
Budinsky, F.; Steinberg, F.; Merks, E.; Ellersick, R. and Grose, T.J. (2003) "Eclipse Modeling Framework", Addison-Wesley Professional.

[Bur'99]
Burnett, M.M. (1999) "Visual Programming"
in Webster , J.G. (1999) "Wiley Encyclopedia of Electrical and Electronics Engineering", Wiley-Interscience.

[Bur'05]
Bur, A. (2005) "JOpera: A Flexible System for Visual Service Composition"
(http://www.jopera.org/node/63, Juni 2008).

[CFL'03]
Costabile, M.F.; Fogli, D.; Letondal, C.; Mussio, P. and Piccinno, A. (2003) "Domain-Expert Users and their Needs of Software Development"
10th International Conference on Human–Computer Interaction (HCII), Iraklio, Griechenland, 2003.

[CFM'06]
Costabile, M.F.; Fogli, D.; Mussio, P. and Piccinno, A. (2006) "End-User Development : The Software Shaping Workshop Approach"
in Lieberman, H.; Paternò, F.; Klann, M. and Wulf, V. (2006) "End User Development", Springer.

[Cyp'93]
Cypher, A.E. (1993) "Watch What I Do: Programming by Demonstration", The MIT Press.

[Dau'07]
Daum, B. (2007) "Rich-Client Entwicklung mit Eclipse 3.2 - Anwendungen entwickeln mit der Rich Client Platform", Dpunkt Verlag.

[DJM'05]
Dostal, W.; Jeckle, M. and Melzer, I. (2005) "Service-orientierte Architekturen mit Web Services : Konzepte - Standards - Praxis", Spektrum Akademischer Verlag.

[Dra'07]
Draxler, S. (2007) "Participatory Design in Use: Integration einer Infrastruktur zur beteiligungsorientierten Softwareentwicklung in den Nutzungskontext"
Diplomarbeit: Universität Siegen, Deutschland.

[DV'06]
De Ruyter, B. and Van Den Sluis, R. (2006) "Challenges for End-User Development in Intelligent Environments"
in Lieberman, H.; Paternò, F.; Klann, M. and Wulf, V. (2006) "End User Development", Springer.

[Edw'07]
Edwards, M. (2007) "Relationship between SCA and BPEL"
(http://www.osoa.org/display/Main/Relationship+between+SCA+and+BPEL, Juni 2008).

[Ehn'90]
Ehn, P. (1990) "From DEMOS and UTOPIA to Work-Oriented Design"
in Ehn, P. (1990) "Work-Oriented Design of Computer Artifacts", Lawrence Erlbaum Associates Incorporated.

[EK'87]
Ehn, P. and Kyng, M. (1987) "The Collective Resource Approach to Systems Design"
in Bjerknes, G.; Ehn, P. and M., K. (1987) "Computers and democracy: A Scandinavian challenge", Avebury.

[EKK'07]
Eisele, M.; Kolb, R.; Kraus, E. and Von Ehrenstein, C. (2007) "SAP NetWeaver: Slicing the fridge"
in Informatik Spektrum, Jahrgang 2007, Ausgabe 06.

[FG'90]
Fischer, G. and Girgensohn, A. (1990) "End-User Modifiability in Design Environments"
Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people, Seattle, USA, 1990.

[FG'06]
Fischer, G. and Giaccardi, E. (2006) "Meta-Design: A Framework for the Future of End-User Development"
in Lieberman, H.; Paternò, F.; Klann, M. and Wulf, V. (2006) "End User Development", Springer.

[FGM'01]
Fischer, G.; Grudin, J.; Mccall, R.; Ostwald, J.; Redmiles, D.; Reeves, D. and Shipman, F. (2001) "Seeding, Evolutionary Growth and Reseeding: The Incremental Development of Collaborative Design Environments"
in Olson, G.; Malone, T. and Smith, J. (2001) "Coordination Theory and Collaboration Technology", Lawrence Erlbaum Associates Incorporated.

[FGY'04]
Fischer, G.; Giaccardi, E.; Ye, Y.; Sutcliffe, A.G. and Mehandjiev, N. (2004) "Meta-Design: A Manifesto for End-User Development"
in Communications of the ACM, Jahrgang 47, Ausgabe 9.

[Fis'96]
Fischer, G. (1996) "Seeding, Evolutionary Growth and Reseeding: Constructing, Capturing, and Evolving Knowledge in Domain-Oriented Design Environments"
Domain Knowledge for Interactive System Design: Proceedings of the TC8/WG8.2 Conference on Domain Knowledge in Interactive System Design, Genf, Schweiz, 1996.

[FRS'89]
Floyd, C.; Reisin, F.-M. and Schmidt, G. (1989) "STEPS to Software Development with Users"
2nd European Software Engineering Conference (ESEC), Warwick, Großbritannien, 1989.

[Fuc'07]
Fuchs, S.K. (2007) "SAP NetWeaver in der Praxis – Wie gut bewährt sich der Technologie-Stack in der praktischen Arbeit?"
in Informatik Spektrum, Jahrgang 2007, Ausgabe 06.

[GB'03]
Gamma, E. and Beck, K. (2003) "Contributing to Eclipse: Principles, Patterns, and Plug-Ins", Addison-Wesley Professional.

[GMS'06]
Gavran, I.; Milanovi, A. and Srbkji, S. (2006) "End-User Programming Language for Service-Oriented Integration".

[GN'92]
Gantt, M. and Nardi, B.A. (1992) "Gardeners and gurus: patterns of cooperation among CAD users"
Proceedings of the SIGCHI conference on Human factors in computing systems, Monterey, USA, 1992.

[GP'92]
Green, T.R.G. and Petre, M. (1992) "When Visual Programs are Harder to Read than Textual Programs"
Proceedings of the 6th European Conference on Cognitive Ergonomics (ECCE-6): "Human-Computer Interaction: Tasks and Organization", Plattensee, Ungarn, 1992.

[GP'96]
Green, T.R.G. and Petre, M. (1996) "Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework"
in Journal of Visual Languages and Computing, Jahrgang 7, Ausgabe 2.

[Gre'89]
Green, T.R.G. (1989) "Cognitive dimensions of notations"
in Sutcliffe, A. and Macaulay, L. (1989) "People and Computers V", Cambridge University Press.

[Gre'96]
Green, T.R.G. (1996) "An Introduction to the Cognitive Dimensions Framework"
Extended abstract of invited talk at MIRA workshop, Monselice, Italien, 1996.

[Gre'00]
Green, T.R.G. (2000) "Instructions and Descriptions: some cognitive aspects of programming and similar activities"
Proceedings of Working Conference on Advanced Visual Interfaces, Palermo, Italien, 2000.

[Hin'99]
Hinken, R. (1999) "Verteilte komponentenbasierter Anpassbarkeite für Groupware - Eine Laufzeit- und Anpassungsumgebung"
Diplomarbeit: Rheinische Friedrich-Wilhelms-Universität Bonn, Deutschland.

[HK'91]
Henderson, A. and Kyng, M. (1991) "There's no place like home: continuing design in use"
in Greenbaum, J. and Kyng, M. (1991) "Design at work: cooperative design of computer systems", Lawrence Erlbaum Associates Incorporated.

[HL'04]
Hauser, T. and Löwer, U.M. (2004) "Web-Services: die Standards", Galileo Press.

[HLD'08]
Hofmann, M.; Ley, B. and Dörner, C. (2008) "Endbenutzergerechte Anpassung von serviceorientierten Softwaresystemen"
Multikonferenz Wirtschaftsinformatik 2008, München, Deutschland, 2008.

[HS'05]
Hudson, R. and Shah, P. (2005) "Tutorial #23, GEF In Depth", Präentationsfolien der EclipseCon 2005 (www.eclipse.org/gef/reference/, Juni 2008).

[Jac'07]
JackBe Corporation (2007) "A Business Guide to Enterprise Mashups".

[Jhi'06]
Jhingran, A. (2006) "Enterprise Information Mashups: Integrating Information, Simply"
Proceedings of the 32nd international conference on Very large data bases, Seoul, Korea, 2006.

[JMS'06]
Juric, M.B.; Mathew, B. and Sarang, P. (2006) "Business Process Execution Language for Web Services", Packt Publishing Limited.

[KB'98]
Kensing, F. and Blomberg, J. (1998) "Participatory Design: Issues and Concerns"
in Computer Supported Cooperative Work, Jahrgang 7, Ausgabe 3-4.

[KEO'03]
Klann, M.; Eisenhauer, M.; Oppermann, R. and Wulf, V. (2003) "Shared initiative: Cross-fertilisation between system adaptivity and adaptability"
10th International Conference on Human–Computer Interaction (HCII), Iraklio, Griechenland, 2003.

[KKL'05]
Kloppmann, M.; Koenig, D.; Leymann, F.; Pfau, G.; Rickayzen, A.; Von Riegen, C.; Schmidt, P. and Trickovic, I. (2005) "WS-BPEL Extension for People – BPEL4People".

[KM'05]
Ko, A.J. and Myers, B.A. (2005) "Human factors affecting dependability in end-user programming"
Proceedings of the first workshop on End-user software engineering, St. Louis, USA, 2005.

[Kyn'88]
Kyng, M. (1988) "Designing for a dollar a day"
Proceedings of the 1988 ACM conference on Computer-supported cooperative work, Portland, USA, 1988.

[Kyn'94]
Kyng, M. (1994) "Scandinavian design: users in product development"
Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating inter-dependence, Boston, USA, 1994.

[Kyn'95]
Kyng, M. (1995) "Users and computers: a contextual approach to design of computer artifacts".

[LPK'06]
Lieberman, H.; Paternò, F.; Klann, M. and Wulf, V. (2006) "End-User Development: An Emerging Paradigm"
in Lieberman, H.; Paternò, F.; Klann, M. and Wulf, V. (2006) "End User Development", Springer.

[LPW'06]
Lieberman, H.; Paternò, F. and Wulf, V. (2006) "End User Development", Springer.

[LS'00]
Lin, W.T. and Shao, B.B.M. (2000) "The relationship between user participation and system success: a simultaneous contingency approach"
in Information and Management, Jahrgang 37, Ausgabe 6.

[Mac'90]
Mackay, W.E. (1990) "Patterns of sharing customizable software"
Proceedings of the 1990 ACM conference on Computer-supported cooperative work, Los Angeles, USA, 1990.

[Mcl'79]
Mclean, E.R. (1979) "End Users as Application Developers"
in MIS Quarterly, Jahrgang 3, Ausgabe 4.

[MCL'90]
Maclean, A.; Carter, K.; Lennart, L. and Moran, K. (1990) "User-tailorable systems: pressing the issues with buttons"
Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people, Seattle, USA, 1990.

[MDG'04]
Moore, B.; Dean, D.; Gerber, A.; Wagenknecht, G. and Vanderheyden, P. (2004) "Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework", IBM.

[MEC'99]
Millea, T.; Edwards, J. and Coutts, I. (1999) "The Engineer, the Architect and the Flower Arranger: A Model for Software Systems Evolution"
Proceedings of the International Workshop on the Principals of Software Evolution (IWPSE), Fukuoka, Japan, 1999.

[Med'04]
MediaKomm (2004) "OSCI-Transport Bibliothek - Funktionsbeschreibung".

[Min'05]
Mintert, S. (2005) "Implementierung von Webservices - REST vs. SOAP?"
in Wirtschaftsinformatik, Jahrgang 2005, Ausgabe 1.

[MK'05]
Myers, B.A. and Ko, A. (2005) "More Natural and Open User Interface Tools"
Proposal to attend the ACM CHI 2005 Workshop on the Future of User Interface, Portland, USA, 2005.

[MPK'04]
Myers, B.A.; Pane, J.F. and Ko, A. (2004) "Natural programming languages and environments"
in Communications of the ACM, Jahrgang 47, Ausgabe 9.

[MS'07]
Margolis, B. and Sharpe, J. (2007) "SOA for the Business Developer: Concepts, BPEL, and SCA", Mc Press.

[MSW'04]
Mørch, A.I.; Stevens, G.; Won, M.; Klann, M.; Dittrich, Y. and Wulf, V. (2004) "Component-Based Technologies for End-User Development"
in Communications of the ACM, Jahrgang 47, Ausgabe 9.

[MTS'04]
Majithia, S.; Taylor, I.; Shields, M. and Wang, I. (2004) "Triana as a Graphical Web Services Composition Toolkit"
Proceedings of UK e-Science All Hands Meeting, Nottingham, Großbritannien, 2004.

[Mul'91]
Muller, M.J. (1991) "PICTIVE - an exploration in participatory design"
Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology, New Orleans, USA, 1991.

[Mul'92]
Muller, M.J. (1992) "Retrospective on a year of participatory design using the PICTIVE technique"
Proceedings of the SIGCHI conference on Human factors in computing systems, Monterey, USA, 1992.

[Mul'93]
Muller, M.J. (1993) "PICTIVE: Democratizing the Dynamics of the Design Session"
in Douglas, S. and Aki, N. (1993) "Participatory Design: Principles and Practices", Lawrence Erlbaum Associates Incorporated.

[Mul'03]
Muller, M.J. (2003) "Participatory design: the third space in HCI"
in Jacko, J.A. and Sears, A. (2003) "The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications", Lawrence Erlbaum Associates Incorporated.

[Mur'04]
Murray, M. (2004) "A closer look at PICTIVE: A participatory design technique".

[Nar'93]
Nardi, B.A. (1993) "A small matter of programming : perspectives on end user computing", The MIT Press.

[NC'01]
Navarro-Prieto, R. and Cañas, J.J. (2001) "Are visual programming languages better? The role of imagery in program comprehension"
in International Journal of Human-Computer Studies, Jahrgang 54, Ausgabe 6.

[Nie'93]
Nielsen, J. (1993) "Usability engineering", B&T.

[NM'91]
Nardi, B.A. and Miller, J.R. (1991) "Twinkling lights and nested loops: distributed problem solving and spreadsheet development"
in International Journal of Man-Machine Studies, Jahrgang 34, Ausgabe 2.

[OH'97]
Orlikowski, W.J. and Hofman, J.D. (1997) "An Improvisational Model of Change Management: The Case of Groupware Technologies"
in Sloan Management Review, Jahrgang 38, Ausgabe 2.

[OH'05]
Mediakomm (2005) "OSCI Handbuch"
(http://osci.mediakomm.esslingen.de/technik/docs.htm, Juni 2008).

[OT]
Spezifikation: OSCI-Transport (1.2), MediaKomm
(http://www1.osci.de/sixcms/detail.php?gsid=bremen02.c.1403.de, Juni 2008).

[OXÖV'06]
MediaKomm (2006) "XÖV-Framework V1.0, Leitlinien für die XÖV-Standardisierung ".

[PA'05]
Pautasso, C. and Alonso, G. (2005) "Flexible Binding for Reusable Composition of Web Services"
Proceedings of the 4th Workshop on Software Composition, Edinburgh, Schottland, 2005.

[Pau'04]
Pautasso, C. (2004) "A Flexible System for Visual Service Composition"
Dissertation: Eidgenössische Technische Hochschule Zürich, Schweiz.

[Pau'05]
Pautasso, C. (2005) "JOpera: an Agile Environment for Web Service Composition with Visual Unit Testing and Refactoring"
Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing, Dallas, USA, 2005.

[Pi'05]
Pipek, V. (2005) "From tailoring to appropriation support: Negotiating groupware usage"
Dissertation: University of Oulu, Finnland.

[PK'06]
Pipek, V. and Kahler, H. (2006) "Supporting Collaborative Tailoring"
in Lieberman, H.; Paternò, F.; Klann, M. and Wulf, V. (2006) "End User Development", Springer.

[PM'96]
Pane, J.F. and Myers, B.A. (1996) "Usability Issues in the Design of Novice Programming Systems"
Carnegie Mellon University, School of Computer Science, Technical Report CMU-CS-96-132, 1996.

[PM'06]
Pane, J.F. and Myers, B.A. (2006) "More Natural Programming Languages and Environments"
in Lieberman, H.; Paternò, F.; Klann, M. and Wulf, V. (2006) "End User Development", Springer.

[PMM'02]
Pane, J.F.; Myers, B.A. and Miller, L.B. (2002) "Using HCI Techniques to Design a More Usable Programming System"
Proceedings of IEEE 2002 Symposia on Human Centric Computing Languages and Environments, Arlington, USA, 2002.

[Pos'96]
Poswig, J. (1996) "Visuelle Programmierung: Computerprogramme auf graphischem Wege erstellen", Hanser Fachbuchverlag.

[RI'06]
Repenning, A. and Ioannidou, A. (2006) "What Makes End-User Development Tick? 13 Guidelines"
in Lieberman, H.; Paternò, F.; Klann, M. and Wulf, V. (2006) "End User Development", Springer.

[RIZ'00]
Repenning, A.; Ioannidou, A. and Zola, J. (2000) "AgentSheets: End-User Programmable Simulations"
in Journal of Artificial Societies and Social Simulation Jahrgang 3, Ausgabe 3.

[RS'04]
Reichert, M. and Stoll, D. (2004) "Komposition, Choreographie und Orchestrierung von Web Services – Ein Überblick"
in EMISA Forum, Jahrgang 24, Ausgabe 2.

[RS'07]
Rauscher, J. and Stiehl, V. (2007) "Programmierhandbuch SAP NetWeaver Composition Environment", Galileo Pres.

[Sal'94]
Saleem, N. (1994) "Alternative Perspectives of User Participation: Practical Implications"
in ACM SIGCPR Computer Personnel, Jahrgang 15, Ausgabe 2.

[SB'06]
Sauerzapf, G. and Behrmann, T. (2006) "End-User Development von Groupware"
Teil des Seminars 1915 der Fernuniversität Hagen: Anwenderorientierte Groupware, 2006.

[SC'00]
Stiemerling, O. and Cremers, A.B. (2000) "The EVOLVE Project: Component-Based Tailorability for CSCW Applications"
in AI & Society, Jahrgang 14, Ausgabe 1.

[SCA]
Spezifikation: SCA Assembly Model (1.0), Open SOA Collaboration bzw. OASIS
(http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications, Juni 2008).

[Sch'96]
Schiffer, S. (1996) "Visuelle Programmierung - Potential und Grenzen"
Tagungsband der Informatik '96: Beherrschung von Informationssystemen, Klagenfurt, Österreich, 1996.

[Sch'98]
Schiffer, S. (1998) "Visuelle Programmierung - Grundlagen und Einsatzmöglichkeiten", Addison Wesley Verlag.

[SDO]
Spezifikation: SDO (2.1), Open SOA Collaboration
(http://www.osoa.org/display/Main/Service+Data+Objects+Specifications, Juni 2008).

[SH'05]
Singh, M.P. and Huhns, M.N. (2005) "Service-oriented computing: semantics, processes, agents", Wiley.

[SHC'99]
Stiemerling, O.; Hinken, R. and Cremers, A.B. (1999) "Distributed Component-Based Tailorability for CSCW Applications"
International Symposium on Autonomous Decentralized Systems, Tokio, Japan, 1999.

[SLM'03]
Sutcliffe, A.; Lee, D. and Mehandjiev, N. (2003) "Contributions, Costs and Prospects for End-User Development"
Proceedings of the 10th International Conference on Human-Computer Interaction, Iraklio, Griechenland, 2003.

[SOAP]
Spezifikation: SOAP (1.2), W3C
(http://www.w3.org/TR/soap/, Juni 2008).

[SQK'06]
Stevens, G.; Quaisser, G. and Klann, M. (2006) "Breaking it up: An industrial case study of component-based tailorable software design"
in Lieberman, H.; Paternò, F.; Klann, M. and Wulf, V. (2006) "End User Development", Springer.

[Sta'81]
Stallman, R.M. (1981) "EMACS the extensible, customizable self-documenting display editor"
in ACM SIGPLAN Notices, Jahrgang 16, Ausgabe 6.

[Ste'02]
Stevens, G. (2002) "Komponentenbasierte Anpassbarkeit - FlexiBeans zur Realisierung einer erweiterten Zugriffskontrolle"
Diplomarbeit: Rheinische Friedrich-Wilhelms-Universität Bonn, Deutschland.

[Sti'97]
Stiemerling, O. (1997) "CAT: Component Architecture for Tailorability".

[Sti'00]
Stiemerling, O. (2000) "Component-Based Tailoring"
Dissertation: Rheinische Friedrich-Wilhelms-Universität Bonn, Deutschland.

[Sti'07]
Stiehl, V. (2007) "Composite Applications: Neue Verfahren für flexible Geschäftsprozesse"
in Informatik Spektrum, Jahrgang 30, Ausgabe 6.

[SW'06]
Stevens, G. and Wiedenhöfer, T. (2006) "CHIC - a pluggable solution for community help in context"
Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles, Oslo, Norwegen, 2006.

[Tri]
The Triana Team, "Triana User Guide".

[UDDI]
Spezifikation: UDDI (3.0.2), OASIS
(http://uddi.org/pubs/uddi-v3.0.2-20041019.htm, Juni 2008).

[URL,a]
"OSCI-Transport Bibliothek - Java"
(http://www1.osci.de/de/detail.php?gsid=bremen02.c.1403.de, Juni 2008).

[URL,b]
IBM "Intro to QEDWiki"
(http://www.youtube.com/watch?v=63qIq9t9Gqs, Juni 2008).

[URL,c]
The Triana Team "Triana and BPEL"
(http://forge.nesc.ac.uk/pipermail/triana-user/2006-May/000120.html, Juni 2008).

[URL,d]
Eclipse "Europa Simultaneous Release"
(http://wiki.eclipse.org/index.php/Europa_Simultaneous_Release, Juni 2008).

[URL,e]
IBM "Developer Guide to the GMF Runtime Framework"
(http://help.eclipse.org/help32/index.jsp?topic=/org.eclipse.gmf.doc/prog-guide/runtime/Developer%20Guide%20to%20Diagram%20Runtime.html, Juni 2008).

[URL,f]
Eclipse "GMF Developer Guide"
(http://help.eclipse.org/help32/index.jsp?topic=/org.eclipse.gmf.doc/prog-guide/runtime/Developer%20Guide%20to%20Diagram%20Runtime.html, Juni 2008).

[VKE'06]
Voelter, M.; Kolb, B.; Efftinge, S. and Haase, A. (2006) "From Front End To Code - MDSD in Practice"
(http://www.eclipse.org/articles/Article-FromFrontendToCode-MDSDInPractice/article.html, Juni 2008).

[WG'01]
Wulf, V. and Golombek, B. (2001) "Direct Activation: A Concept to Encourage Tailoring Activities"
in Behaviour and information technology, Jahrgang 20, Ausgabe 4.

[WH'07]
Wong, J. and Hong, J.I. (2007) "Making Mashups with Marmite: Towards End-User Programming for the Web"
Proceedings of the SIGCHI conference on Human factors in computing systems, San Jose, USA, 2007.

[Whi'96]
Whitley, K.N. (1996) "Visual Programming Languages and the Empirical Evidence For and Against"
in Journal of Visual Languages and Computing, Jahrgang 8, Ausgabe 1.

[Whi'04]
White, S.A. (2004) "Introduction to BPMN"
(http://www.bpmn.org, Juni 2008).

[Whi'05]
White, S.A. (2005) "Mapping BPMN to BPEL Example"
(http://www.bpmn.org, Juni 2008).

[Wie'06]
Wiedenhöfer, T. (2006) "Help in Context - Konzeption und Umsetzung eines communityunterstützten Hilfesystems"
Diplomarbeit: Universität Siegen, Deutschland.

[WJ'04]
Wulf, V. and Jarke, M. (2004) "The economics of end-user development"
in Communications of the ACM, Jahrgang 47, Ausgabe 9.

[Won'03]
Won, M. (2003) "Supporting End-User Development of Component-Based Applications by Checking Semantic Integrity"
Workshop in Perspectives on End-User Development in Conjunction with CHI 2003, Fort Lauderdale, USA, 2003.

[Won'04]
Won, M. (2004) "Interaktive Integritätsprüfung für komponentenbasierte Architekturen"
Dissertation: Rheinische Friedrich-Wilhelms-Universität Bonn, Deutschland.

[WPW'08]
Wulf, V.; Pipek, V. and Won, M. (2008) "Component-Based Tailorability: Enabling highly flexible software applications"
in International Journal of Human-Computer Studies, Jahrgang 66, Ausgabe 1.

[WSA]
Spezifikation: WS-Adressing W3C
(http://www.w3.org/Submission/ws-addressing/, Juni 2008).

[WSDL]
Spezifikation: WSDL (2.0), W3C
(http://www.w3.org/TR/wsdl, Juni 2008).

[WSHT]
Spezifikation: WS-HumanTask (1.0), u.a. IBM, Oracle, SAP
(http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel4people, Juni 2008).

[WSW'06]
Won, M.; Stiemerling, O. and Wulf, V. (2006) "Component-based Approaches to tailorable Systems"
in Lieberman, H.; Paternò, F.; Klann, M. and Wulf, V. (2006) "End User Development", Springer.

[Wul'94]
Wulf, V. (1994) "Anpaßbarbarkeit im Prozeß evolutionärer Systementwicklung"
in GMD-Spiegel, Jahrgang 24, Ausgabe 3.

[Wul'00]
Wulf, V. (2000) "Exploration Environments: Supporting Users to Learn Groupware Functions"
in Interacting with Computers, Jahrgang 13, Ausgabe 2.

[XE]
Spezifikation: XML-Encryption W3C
(http://www.w3.org/TR/xmlenc-core/, Juni 2008).

[XP]
Spezifikation: XPath (1.0), W3C
(http://www.w3.org/TR/xpath, Juni 2008).

[XSc]
Spezifikation: XML Schema (2004-03-18), W3C
(http://www.w3.org/XML/Schema, Juni 2008).

[XSi]
Spezifikation: XML Signature (2002-02-12), W3C
(http://www.w3.org/TR/xmldsig-core/, Juni 2008).

Anhang: Participatory Design Workshop

Agenda

10:00 Einführung

10:15 Motivation

10:30 Kaffeepause

10:40 Kontext und Rahmen

11:00 Scenario Building

12:30 Mittagspause

13:30 Szenarioumsetzung (Orchestrierung), Reflexion und Evaluation

15:00 Abschluss

Fragebogen

	Stimme voll zu				Stimme gar nicht zu
Die Techniker haben mich verstanden					
Ich konnte mich völlig frei äußern					
Ich fühlte mich gleichberechtigt					
Ich bin zufrieden mit Hilfsmitteln, die mir zur Verfügung standen					
Der Workshop hat mir Spaß gemacht					
Der Workshop war interessant					
Der Workshop war nützlich					
Ich habe es gehasst, auf Video aufgenommen zu werden					