Thema:

Nutzerorientierte Anwendungsdienstentwicklung unter Verwendung der Web Service Technologie für die Institutionen der öffentlichen Verwaltung

Diplomarbeit

Arbeitsgruppe Wirtschaftsinformatik

Themensteller: Prof. Dr. rer. pol. habil. Hans-Knud Arndt
Betreuer: Prof. Dr. rer. pol. habil. Hans-Knud Arndt

vorgelegt von: Heiko Dolgener

Abgabetermin: 01. Oktober 2008

Inhaltsverzeichnis

Verzeichnis der Abkürzungen und Akronyme

ACID	Atomicity, Consistency, Isolation, Durability
BPEL	Business Process Execution Language
DBMS	Datenbankmanagementsystem
FK^N	Funktionskomplex auf der Nutzerebene
FK^K	Funktionskomplex auf der konzeptuellen Ebene
IE	interne Ebene
IM	identifizierende Merkmalsmenge
KE	konzeptuelle Ebene
NE	Nutzerebene
UDDI	Universal Description, Discovery and Integration
RPC	Remote Procedure Call
SAGA	Standards und Architekturen für E-Government-Anwendungen
SLA	Service Level Agreement
SOA	Service orientierte Architektur
SOAP	Simple Object Access Protocol
SSL	Secret Socket Layer
TKS	Telekooperationssystem
TRS	Teilrechtssystem
TSL	Transport Socket Layer
WSDM	Web Services Distributed Management
XML	Extensible Markup Language

Symbolverzeichnis

aD	abstrakter Dienst
DBS_x	Datenbeschreibungsstruktur
DM_x	Datenmodell
DEP_{ij}	Dienstentwicklungsprozess
DES_{ij}	Dienstentwicklungssystem
DG_{ij}	Dienstgang mit den Zuständen i und j
DP_{ij}	Dienstprozess mit den Zuständen i und j
DS_{ij}	Dienstsystem mit den Zustandsbeschreibungen i und j
DTP_{ij}	Dienstteilprozess
DVf_{ij}	Dienstverfahren mit den Punkten i und j
$(V)P_{ij}$	(Verwaltungs-) Prozess mit den Zuständen i und j
$(V)PG_{ij}^{k}$	k-ter (Verwaltungs-) Prozessgang mit den Zuständen i und j
$(V)S_{ij}$	(Verwaltungs-) System mit den Zustandsbeschreibungen i und j
$(V)Vf_{ij}$	(Verwaltungs-) Verfahren mit den Punkten i und j
$a_k(VS_{ij})$	k-te Ausführung des Verwaltungssystems
kD	konkreter Dienst
l_i, l_j	Ort i, Ort j
NK_{kl}	Nutzerklasse
t_i, t_j	Zeit i, Zeit j
TPij	Transportprozess mit den Zuständen i und j
VG_{ij}^{k}	k-ter Verwaltungsgang mit den Zuständen i und j
VV_{ij}	Verwaltungsverhalten
VVg_{ij}	Verwaltungsvorgang mit den Zuständen i und j
VA_{ij}	Verwaltungsaufgabe
VAA_{ij}	Verwaltungsaufgabenausprägung
VAB_{ij}	Verwaltungsaufgabebeschreibung
VAt_{ij}	Verwaltungsauftrag
VO_{ij}	Verwaltungsobjekt
VOK_{ij}	Verwaltungsobjektkomponente
$VOKB_{ij}$	Verwaltungsobjektkomponentenbeschreibung
$VOKA_{ij}$	Verwaltungsobjektkomponentenausprägung
W_i, W_j	Wissen i, Wissen j
WEP_{ij}	Werkzeugentwicklungsprozess
WES_{ij}	Werkzeugentwicklungssystem
Z_i, Z_j	Zustand i, Zustand j
ZA_i, ZA_j	Zustandsausprägung i, Zustandsausprägung j
ZB_i, ZB_j	Zustandsbeschreibung i, Zustandsbeschreibung j

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

E-Government ist eine Entwicklungsrichtung der Institutionen der öffentlichen Verwaltung, welche sich mit der „[…] Abwicklung geschäftlicher Prozesse im Zusammenhang mit Regieren und Verwalten (Government) mit Hilfe von Informations- und Kommunikationstechniken über elektronische Medien“ (Lucke, Reinermann (2000), S.1) befasst. Dabei geht es um neue Formen der Information, Kommunikation und Transaktion der Institutionen mit verschiedenen Nutzerklassen (u. a. Angestellte eines Unternehmens, verwaltungserfahrene Bürger, interneterfahrene Bürger).

Ein modernes E-Government bietet zum Einen starke Einsparungs- und Rationalisierungspotentiale, welche im Rahmen von leeren Staatskassen und einer hohen Staatsverschuldung nicht zu verachten sind. Diese Potentiale können alleine schon durch die Reduzierung des herkömmlichen Schriftverkehrs und damit dem Wegfall von Transportzeiten und -kosten verdeutlicht werden. Zum Anderen bietet es eine stärkere Ausrichtung an den tatsächlichen Nutzer der Dienstleistungen der Institutionen der öffentlichen Verwaltung und damit mehr Transparenz, schnellere Bearbeitungswege und eine 24 Stunden Verfügbarkeit der Onlinedienstleistungen.

Bei der Softwareentwicklung ist ein starker Trend zur Nutzung der Web Service Technologie zur Realisierung serviceorientierter Architektur erkennbar. Dieser Trend wird auch in den Institutionen der öffentlichen Verwaltung deutlich. Wie später noch zu sehen ist, bietet dies gerade in komplexen und großen System- und Softwarelandschaften mehrere Vorteile.

Diese Arbeit hat zum Ziel, die nutzerorientierte Systementwicklung (nach Lüttich) mit Hilfe der aktuellen Web Service Technologien zu realisieren. Dabei wird speziell eine Werkzeugumgebung geschaffen, die es einem Nutzer ermöglicht verwaltungsnahe Dienstleistungen als einen Anwendungsdienst ohne eigene Programmierkenntnisse zu erstellen. Dazu ist diese Arbeit wie folgt gegliedert:

Zunächst werden im zweiten Kapitel die Grundlagen des nutzerorientierten Systementwicklungsansatzes von Lüttich ausführlich erläutert. Im dritten Kapitel folgen dann die Grundlagen der Web Service Technologie. Anschließend erfolgt die Definition und Überführung eines Dienstes in einen erweiterten Anwendungsdienst. Dazu wird die Realisierbarkeit des nutzerorientierten Systementwicklungsansatzes mittels der Web Service Technologie geprüft. Das fünfte Kapitel erläutert die Anforderungen an das zu konzipierende System, um dann im sechsten Kapitel zu beschreiben, wie die Anwendungsdienstentwicklung aus Nutzersicht realisiert werden kann. Im letzten Kapitel erfolgen schließlich eine Zusammenfassung und ein Ausblick.

2 Nutzerorientierte Systementwicklung

In Anlehnung an das Drei-Schichtenmodell nach der ANSI-SPARC-Architektur (siehe u. a. Rautenstrauch, Schulze (2003), S. 124 f.) des Datenbankmanagementsystems, welches aus der Benutzer- bzw. externen Ebene, der konzeptuellen bzw. logischen Ebene und der internen bzw. physischen Ebene besteht, kann auch die Softwaresystementwicklung in drei Ebenen unterteilt werden. Diese sind: die nutzerorientierte Ebene (*NE*), die konzeptuelle Ebene (*KE*), welche oft auch synonym als konzeptionelle Ebene bezeichnet wird (vgl. Manthey (2002), S.2) und die interne Ebene (*IE*).

In der bisherigen Systementwicklung wird von einer Vielzahl an konzeptuellen Ebenen ausgegangen. Diese sind durch eine Vielzahl von unterschiedlichen Programmiersprachen und verschiedenen Datenbank-(betriebs-)Systemen notwendig. So muss ein Anwendungssystem, insofern es auf einer anderen Plattform lauffähig sein soll, oftmals komplett umgeschrieben werden (Ausnahmen bilden hier plattformunabhängige Programmiersprachen, z. B. Java). Es wird davon ausgegangen, dass nur eine homogene Menge an Nutzern existiert, welche auf das Anwendungssystem zugreift. Aus diesem Grund wird überwiegend nur eine Nutzerebene betrachtet.

In der nutzerorientierten Systementwicklung wird der Ansatz verfolgt, dass nicht nur eine homogene Menge an Nutzern existiert. Aus diesem Grund werden multiple Nutzerebenen eingeführt. Damit dennoch die Systementwicklung ermöglicht werden kann, wird von der Existenz einer kompetenten Nutzerklasse (*KNE*, siehe Kapitel 2.3) ausgegangen. Diese hat umfassende Kenntnisse über den Ablauf des Gesamtprozesses und die beteiligten Ressourcen, muss dabei allerdings keine Softwareentwicklungskenntnisse besitzen. Die konzeptuelle Ebene untergliedert sich nach diesem Modell in zwei Ebenen: eine sprachunabhängige und eine sprachabhängige. Für die Überführung des entworfenen Systems aus Nutzersicht kommen entsprechende Softwarewerkzeuge (siehe Kapitel 2.5) zum Einsatz. Dabei müssen die aus Nutzersicht entwickelten Systeme nicht für jede Programmiersprache in der sprachabhängigen konzeptuellen Ebene vollständig neu entwickelt werden, sondern der Code wird durch Generierungsverfahren erzeugt.

Die interne Ebene betrachtet die systemnahe Realisierung eines Anwendungssystems bzw. Dienstes. Diese Ebene wird in der Regel für die Entwickler durch die Verwendung von höheren Programmiersprachen in der konzeptuellen Ebene verdeckt. In Abb. 2.1 ist dieser Zusammenhang einmal grafisch dargestellt.

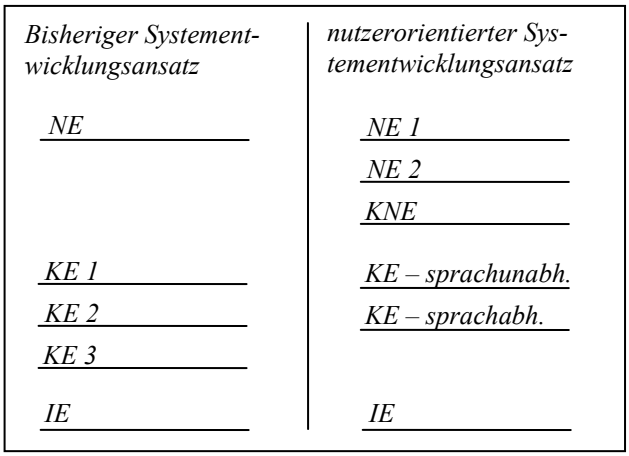


Abb. 2.1: unterschiedliche Struktur der Systementwicklungsansätze

Die nutzerorientierte Systementwicklung nach Lüttich bietet aufbauend auf einem erweiterten Prozessmodell einen Ansatz zur Modellierung eines Softwaresystems, bei dem 80-90% des späteren Quelltextes generiert werden können und nur die eigentlichen Teilverfahren aus Nutzersicht eigenständig programmiert werden müssen.

Aufgrund der Themenstellung dieser Arbeit im Aufgabengebiet der Verwaltungsinformatik wird im Folgenden die nutzerorientierte Systementwicklung speziell aus Sicht der Institutionen der öffentlichen Verwaltung vorgestellt. Die besondere Herausforderung der auf diesem Gebiet tätigen Verwaltungsinformatik besteht darin, dass eine Vielzahl an multiplen Nutzerklassen (siehe Kapitel 2.3) auf ein Anwendungs- oder auch ein reales System zugreifen müssen, um entsprechende Dienstleistungen der Institutionen der öffentlichen Verwaltung nutzen bzw. Aufgaben erledigen zu können. Gleichwohl lässt sich dieser Systementwicklungsansatz auch auf alle anderen Gebiete, wie z. B. die der Wirtschaftsinformatik, übertragen. Hier wird allerdings von einer weitestgehend homogenen Nutzerklasse ausgegangen, welche sich höchstens durch unterschiedliche Rollen[1] innerhalb eines Anwendungssystems, unterscheidet.

Es sei darauf hingewiesen, dass diese Ausführungen nur einen Überblick darstellen. Für weitere Informationen zur Softwareentwicklung aus Nutzersicht sei auf die Materialien zu den Vorlesungen „Nutzerorientierte Systementwicklung“, „Software Werkzeuge“, „E-Government“ und „Telekooperation in den öffentlichen Verwaltungen“, sowie auf die Veröffentlichung in Lüttich, Rautenstrauch (2000), S. 458-471 von LÜTTICH verwiesen.

[1] Rollen sind „[…] eine Menge von Privilegien, die einer Benutzergruppe zugeordnet werden kann.“ (Rautenstrauch, Schulze 2003, S. 99)

2.1 Erweitertes Prozessmodell

Es gibt unterschiedliche Ansätze für die Definition eines Prozesses. Daher muss zunächst ein einheitliches Verständnis geschaffen werden. Ein Prozess stellt nach ROSEMANN „[…] die inhaltlich abgeschlossene, zeitliche und sachlogische Abfolge der Funktionen dar, die zur Bearbeitung eines betriebswirtschaftlichen Prozessobjekts ausgeführt werden“ (Rosemann (1995), S. 9). Damit kann ein Prozess nach Rosemann wie in Abb. 2.2 dargestellt werden. Es ist erkennbar, dass im zeitlichen Verlauf nacheinander die einzelnen Funktionen eins, zwei und drei ausgeführt werden, um das Prozessobjekt zu bearbeiten.

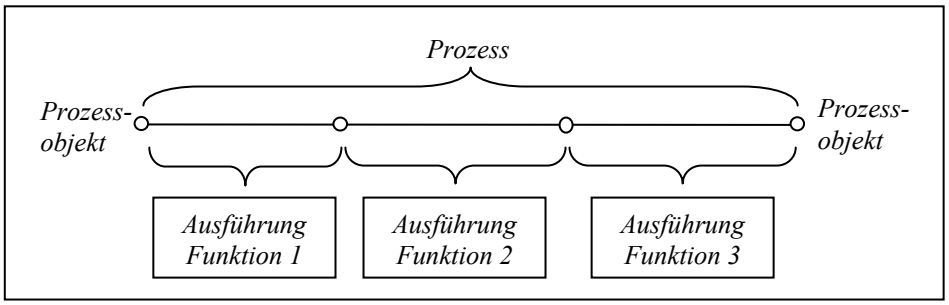


Abb. 2.2: Prozessdefinition nach Rosemann

Nach RAUTENSTRAUCH wird der Prozess im Gebiet der (Wirtschafts-) Informatik, aufbauend auf der Definition von Rosemann, wie folgt definiert: „Als Prozess wird eine zeitliche Abfolge von Aktivitäten bezeichnet, die jeweils Zustandsänderungen nach sich ziehen. Zustände werden durch die Belegung von Variablen dokumentiert“ (Rautenstrauch, Schulze (2003), S. 95). Wenn auch nicht explizit formuliert, wird hier die Ausführung der Aktivitäten verlangt.

Im erweiterten Prozessmodell von LÜTTICH wird ein Prozess (P_{ij}) als ein spezieller Informationsprozess definiert (vgl. Lüttich, Turowski (2000), S. 458). Ein Prozess ergibt sich aus der Vereinigung aller möglichen Prozessgänge. Ein Prozessgang (PG_{ij}^{k}) ist als eine einmalige Ausführung eines Verwaltungssystems (VS_{ij}) definiert. Daraus resultiert:

$$P_{ij} = \bigcup_{k} PG_{ij}^{k} \text{ über } VS_{ij}.$$

Die verwendeten Indizes sind Ausdruck dafür, dass es sich um unterschiedliche Komponenten handelt. Sie können beliebig verwendet werden. In dieser Arbeit wird überwiegend der Index i für die Anfangs- und der Index j für die Endkomponenten eines Verwaltungsprozesses genutzt.

Wichtig ist hier zu erwähnen, dass nach Lüttich (vgl. Lüttich (2007a)) die Gesamtheit der tatsächlichen Ausführungen der Funktionen eines Systems einen Prozess darstellt.

Werden nur die Funktionen und nicht deren Ausführungen betrachtet, muss von einem System gesprochen werden. Erfolgt nur eine einmalige Ausführung eines Systems mit einem entsprechenden Auftrag, wird dies als ein Prozessgang bezeichnet. Diese klare Trennung zwischen Prozess, Prozessgang und System wird in der Literatur teilweise vernachlässigt. Daher stellen die Definitionen eines Prozesses von Rosemann und von Rautenstrauch, nach der Auffassung von Lüttich, lediglich einen Prozessgang dar.

In der Systementwicklung aus Nutzersicht wird das Prozessobjekt als Nutzerobjekt bezeichnet. Es stellt dabei ein spezielles Prozessobjekt dar und kann aus mehreren Objektkomponenten bestehen. Diese können verschiedene Objekte der konzeptuellen Ebene darstellen (für weitere Erläuterungen, siehe Abschnitt: Verwaltungsobjekt). In der Verwaltungsinformatik wird von einem Verwaltungsobjekt gesprochen. Dieses stellt ein spezielles Nutzerobjekt unter Anwendung eines Rechtsrahmens dar.

Als Beispiel sei die Bearbeitung eines Wohnortwechsels genannt. Der Bürger einer Stadt, welcher einer bestimmten Nutzerklasse angehört (z. B. verwaltungs- und interneterfahren), setzt mittels eines Antrages (mündlich im Bürgerbüro, schriftlich per Brief oder online per Eingabeformular), welcher in einen Verwaltungsauftrag (VAt_{ij}) überführt wird, einen Verwaltungsvorgang (VVg_{ij}) des Verwaltungsprozesses (VP_{ij}) in Gange. In diesem Fall wird das Verwaltungsobjekt „Meldeadresse Bürger x“ bearbeitet.

Das erweiterte Prozessmodel gliedert sich in drei Ebenen: der Prozessebene, der Systemebene und der Verfahrensebene. Im Folgenden werden diese Ebenen sowie deren einzelne Bestandteile erläutert.

2.1.1 Prozessebene - Verwaltungsprozess (VP)

Die Prozessebene stellt die Ausführung eines Verwaltungssystems ($a_k(VS_{ij})$) dar. Dabei vereint die Prozessdarstellung alle Ausführungen der möglichen Verwaltungsgänge auf dem darunterliegenden Verwaltungssystem (VS_{ij}).

Alle Komponenten dieser Ebene bestehen aus einer Komponentenbeschreibung und einer -ausprägung. Somit besteht ein Verwaltungsprozess (VP_{ij}) aus einem Anfangs- und einem Endzustand (Z_i bzw. Z_j) sowie der Relation zwischen diesen beiden. Gleichzeitig bestehen diese wiederum aus einer Zustandsbeschreibung und einer Zustandsausprägung ($Z_i = (ZB_i, ZA_i)$ bzw. $Z_j = (ZB_j, ZA_j)$). Während die Zustandsbeschreibung die einzelnen Attribute benennt, enthalten die Zustandsausprägungen die jeweiligen konkreten Werte zum Zeitpunkt der Ausführung des Prozesses. Bei Netzen oder Ketten von Verwaltungsprozessen existieren demzufolge entsprechend mehrere Zwischenzu-

stände. Im Fünf-Komponenten-Modell, welches auch erweitert werden kann, treten die folgenden Komponenten des Anfangszustandes auf:

- die Verwaltungsaufgabe (VA_{ij}),
- das Verwaltungsobjekt (VO_i),
- das Wissen (W_i) und
- die speziellen Beobachtungsfunktionen für die Zeit (t_i) und den Ort (l_i).

Zusätzlich besteht wiederum jede Komponente der Zustände (Z_i, Z_j) auf der Prozessebene aus einer Komponentenbeschreibung und einer -ausprägung. Der Endzustand Z_j beinhaltet nahezu die selben Komponenten wie der Anfangszustand Z_i. Hier entfällt allerdings die Verwaltungsaufgabe, da diese bereits ausgeführt wurde. Dafür wird die Komponente des Verwaltungsverhaltens (VV_{ij}) eingeführt.

Nachdem alle Bestandteile eines Verwaltungsprozesses benannt wurden, kann dieser wie in Abb. 2.3 dargestellt werden.

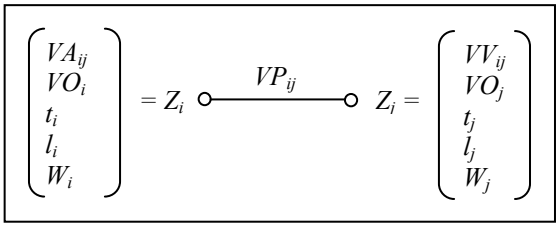


Abb. 2.3: Prozessdarstellung

Ein Verwaltungsprozess stellt, wie bereits definiert, die Vereinigung der einzelnen möglichen Verwaltungsprozessgänge dar (siehe Abb. 2.4). Dabei besitzt jeder Verwaltungsprozessgang (VPG_{ij}^{k}) anstelle einer Verwaltungsaufgabe (VA_{ij}) einen Verwaltungsauftrag (VAt_{ij}). Dieser ruft nur bestimmte Teilaufgaben einer Verwaltungsaufgabe auf. Damit werden nur Teilbereiche des darunter liegenden Verwaltungssystems (VS_{ij}) genutzt. Die restlichen Komponenten entsprechen denen eines Verwaltungsprozesses (VP_{ij}).

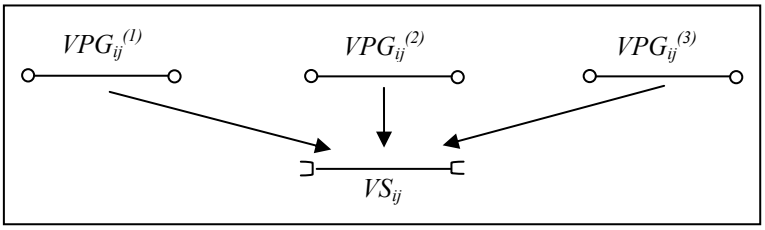


Abb. 2.4: Zusammenhang Verwaltungsprozessgang - Verwaltungssystem

Aus der Definition des Verwaltungsprozesses: $VP_{ij} = \bigcup_k VPG_{ij}^k$ ergibt sich,
dass auch die Zustände Z_i und Z_j aus der Vereinigung der Zustände der einzelnen Verwaltungsprozessgänge gebildet werden:

$$Z_i = \bigcup_k Z_i^k \ , \ Z_j = \bigcup_k Z_j^k \ .$$

Gleiches gilt für die Verwaltungsaufgabe (VA_{ij}), die sich aus der Vereinigung der möglichen Aufträge der Prozessgänge ergibt:

$$VA_{ij} = \bigcup_k VAt_{ij}^k \ .$$

Die Relation ($rel(Z_i, Z_j)$) zwischen dem Anfangs- und Endzustand (Z_i und Z_j) des Verwaltungsprozesses und die Vereinigung der Relationen zwischen den Anfangs- und Endzuständen (Z_i^k, Z_j^k) der Verwaltungsgänge, wird per Definition gleich gesetzt mit dem Verwaltungssystem:

$$rel\left(Z_i, Z_j\right) \overset{!}{=} \bigcup_k rel(Z_i^k, Z_j^k) \overset{!}{=} VS_{ij}$$

Ein beliebiger Verwaltungsprozessgang (VPG_{ij}^k) kann mittels den Anfangs- und Endzuständen (Z_i^k, Z_j^k) sowie der Relation zwischen den Zuständen beschrieben werden. Zusätzlich wird festgelegt, dass die Relation $rel(Z_i^k, Z_j^k)$ per Definition mit dem dabei benötigten Teil des Verwaltungssystems (VS_{ij}^k) gleichgesetzt werden kann. Dies wiederum bedingt, dass ein Verwaltungsprozessgang VPG_{ij}^k aus der k-ten Ausführung (a_k) des Verwaltungssystems ($a_k(VS_{ij})$) und damit ein Verwaltungsprozess (VP_{ij}) aus der Vereinigung aller Verwaltungsgänge ($\bigcup_k VPG_{ij}^k$), resultiert.

$$VPG_{ij}^k = (Z_i^k, Z_j^k, rel(Z_i^k, Z_j^k)) = (Z_i^k, Z_j^k, VS_{ij}^k) = a_k(VS_{ij}) \, ,$$

$$VP_{ij} = \bigcup_k a_k(VS_{ij}) = \bigcup_k VPG_{ij}^k \ .$$

Der Index k steht für die Anzahl der auszuführenden Prozessgänge. Es existiert der mögliche Sonderfall k=1. In diesem Fall muss nur ein Verwaltungsprozessgang (VPG_{ij}^k) ausgeführt werden. Im Folgenden werden die einzelnen Komponenten der Zustände Z_i und Z_j weiter erläutert:

Die **Verwaltungsaufgabe (VA_{ij})** besteht aus einer Verwaltungsaufgabenbeschreibung (VAB_{ij}) und einer Verwaltungsaufgabenausprägung (VAA_{ij}) oder kurz $VA_{ij} = (VAB_{ij}, VAA_{ij})$. Sie beinhaltet alle möglichen Aufgaben eines Verwaltungsprozesses. Ein Beispiel ist in Tab. 2.1 dargestellt. Hier ist anzumerken, dass eine vollständige Auflistung

aller möglichen ausführbaren Verwaltungsteilaufgaben als Verwaltungsaufgabe bezeichnet wird. Erfolgt nur eine Auflistung der für einen Verwaltungsgang notwendigen Teilaufträge, handelt es sich um einen Auftrag. Dabei aktiviert die Verwaltungsaufgabe die einzelnen Teilprozesse über den Teilsystemen und steuert damit den Ablauf des kompletten Verwaltungsprozesses.

Tab. 2.1: Beispiel für eine Verwaltungsaufgabe

VAB_{ij}	Anweisungsnummer	Operator	Operanden
VAA_{ij}	1	Invoke	Zulassung_PKW
	2	Invoke	Bestellung_Kennzeichen
	3	Reply	Anmeldung erfolgreich
	…	…	…

Wenn ein Nutzer eine Aufgabe an das Verwaltungssystem stellt, wird ein Auftrag an dieses übergeben. Dieser Auftrag enthält die Reihenfolge und die notwendigen Attribute (Merkmalsausprägungen) der auszuführenden Anweisungen und wird als Verwaltungsprozessgang ausgeführt. Dabei bildet der Auftrag eine Teilmenge der Verwaltungsaufgabe des Verwaltungsprozesses.

Die Nutzersprache der betrachteten Verwaltungsprozesse bildet nach Lüttich die Gesamtheit der Operatoren und Operanden der Verwaltungsaufgabe. Sie muss entsprechend definiert werden. In dieser Arbeit dient die Beschreibungssprache „Web Service - Business Execution Language“ (WS-BPEL) als Grundlage, auf welche in Kapitel 3.2 näher eingegangen wird.

Das **Verwaltungsobjekt (VO_i bzw. VO_j)** (auf der Prozessebene auch als konkretes Verwaltungsobjekt bezeichnet) stellt ein globales Objekt[2] dar und besteht aus einer Verwaltungsobjektbeschreibung und einer -ausprägung: $VO_i = (VOB_i, VOA_i)$. Die VOB_i und dementsprechend auch die VOA_i können wiederum aus mehreren Verwaltungsobjektkomponentenbeschreibungen bzw. -ausprägungen ($VOKB_i$ bzw. $VOKA_i$) bzw. Merkmalsmengenbeschreibungen (*MMB*) bestehen. In diesen Komponenten sind die einzelnen Eingangs- und Ausgangsinformationen aus Nutzersicht für einen Verwaltungsprozess abgelegt. Sie stellen fachbezogene Mengen von Merkmalsbeschreibungen dar, die nach verschiedenen Gesichtspunkten klassifiziert werden. Als Beispiel sei hier die Klassifizierung nach der Merkmalsherkunft genannt:

- *MMB0*: identifizierende Merkmale,
- *MMB1*: Informationen des Bürgerbüros,

[2] Dies bedeutet, dass ein Verwaltungsobjekt in mehreren Verwaltungsprozessen verwendet werden kann.

- *MMB2*: Informationen des Umweltamtes.

Die einzelnen Merkmalsmengen können dabei den folgenden Aspekten genügen:

- dem datenorientierten Aspekt (zwingend notwendig),
- dem funktionalen bzw. aufgabenbezogenen Aspekt und
- dem verhaltensorientierten Aspekt.

Das Vorhandensein des datenorientierten Aspekts ist die minimale Anforderung an ein Verwaltungsobjekt. Ist dieser Aspekt erfüllt, können die Informationen in einer Datenbank gespeichert, bearbeitet und abgerufen werden. Aufgrund dieses Aspekts besteht das Verwaltungsobjekt aus einer identifizierenden und mindestens einer qualitativen oder quantitativen Merkmalsmenge. Würden die identifizierenden Merkmale fehlen, könnte keine Zuordnung der abgelegten Informationen erfolgen. Die Merkmalsmengenbeschreibung Null (*MMB0*) beinhaltet dabei die identifizierenden Merkmale (*IM*) einer Objektbeschreibung. Die Merkmalsausprägungen, die ein Nutzerobjekt enthält, können aus verschiedenen Informationsquellen stammen und dennoch als ein Objekt genutzt werden. So kann beispielsweise das Objekt Hauseigentümer für das Bürgerbüro aus den Merkmalsmengen des Grundbuchamtes aber auch aus denen der Versicherung sowie des Finanzamtes zusammengesetzt sein. Diese flexible Nutzung und Zuordnung der verschiedenen Informationsquellen ermöglicht ein umfassendes Wissen in der aktuell zu bearbeitenden Aufgabe bzw. deren Teilaufträge. Dadurch wird Mehr- und Doppelarbeit in den Institutionen der öffentlichen Verwaltung verhindert und es ermöglicht bei der Sachbearbeitung ein möglichst umfassendes Bild über den Antragsteller und dessen Anliegen zu erhalten. Somit wird eine schnelle und sachliche Entscheidungsfindung möglich.

Mit Hilfe des funktionalen bzw. aufgabenbezogenen Aspekts wird die Möglichkeit der Bearbeitung der einzelnen Verwaltungsobjekte gegeben. Diese beinhalten die jeweils ausführbaren Funktionen bzw. die Gesamtheit der möglichen Aufgaben eines Verwaltungsobjektes.

Der verhaltensorientierte Aspekt beinhaltet die doppelte Protokollierung des Ablaufs und spiegelt die Komponente des Verwaltungsverhaltens wieder. Zum einen erfolgt die Erfassung des Ablaufs des Verwaltungsprozesses und zum anderen die Erfassung von aufgetretenen Fehlern und Nachrichten.

Die Abb. 2.5 stellt beispielhaft eine Verwaltungsobjektbeschreibungsstruktur[3] dar. Diese beinhaltet alle drei zuvor erläuterten Aspekte inklusive einer angenommenen Merkmalsbeschreibung der jeweiligen Merkmalsmengenbeschreibungen.

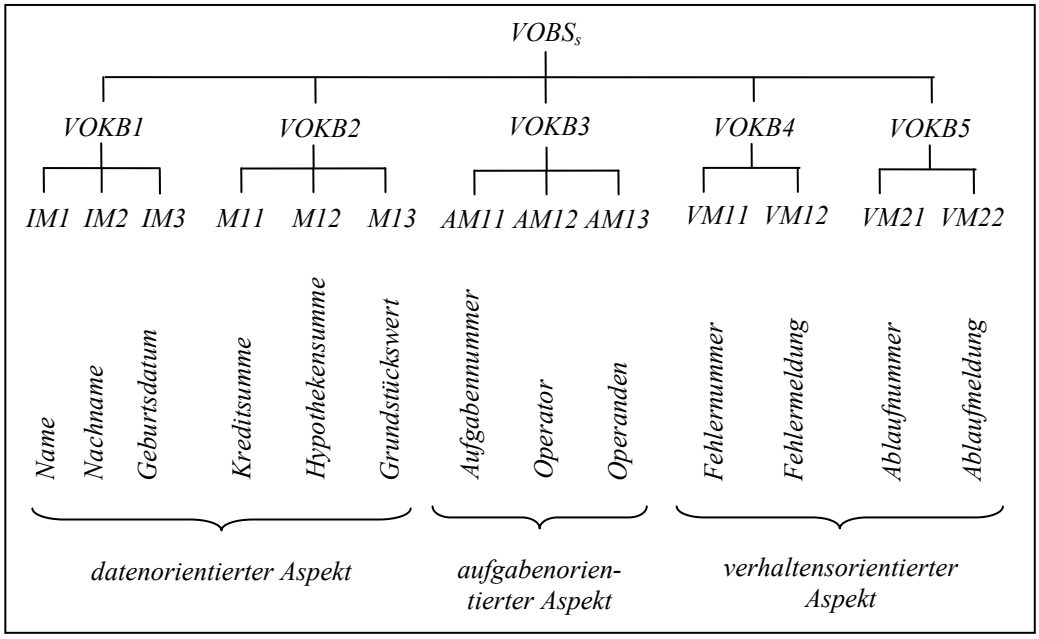


Abb. 2.5: Beispiel für eine Verwaltungsobjektbeschreibungsstruktur

Eine erste Schwierigkeit ergibt sich durch die Transformation des Bürgerobjektes (z. B. der Antrag in Briefform), wie es der Bürger erstellt hat, in die Sichtweise der öffentlichen Verwaltung, zu einem Verwaltungsobjekt (gleiches gilt für die Rücktransformation). Dies kann direkt im Bürgerbüro bei einem Gespräch erfolgen oder nach einem schriftlichen Eingang im Backoffice. Die Transformation ist notwendig, um die enthaltenen Informationen zu nutzen und den entsprechenden Rechtsrahmen zuzuordnen. Die Verwaltungsaufgaben werden dann aus dem erstellten Verwaltungsobjekt abgeleitet oder sind bei rechnergestützten Verwaltungsprozessen bereits im Vorfeld festgelegt. In dieser Arbeit wird der zweite Fall angenommen.

Die **Wissenskomponente** (W_i bzw. W_j) dient zum Sammeln und speichern von Wissen im und über den Prozess. Damit wird ermöglicht, dass einmal gesammelte Erfahrungen bei der Bearbeitung eines speziellen Verwaltungsprozesses (z. B. eines Verwaltungsaktes[4]) für spätere Verwaltungsvorgänge festgehalten werden. Diese Komponente wird aufgrund des Umfangs im Folgenden nicht weiter betrachtet.

[3] entspricht der grafischen Darstellung der Verwaltungsobjektbeschreibung

[4] „[Ein] Verwaltungsakt ist jede Verfügung, Entscheidung oder andere hoheitliche Maßnahme, die eine Behörde zur Regelung eines Einzelfalls auf dem Gebiet des öffentlichen Rechts trifft und die auf unmittelbare Rechtswirkung nach außen gerichtet ist.“ (§35 Verwaltungsverfahrensgesetz (BVwVfG) vom 05.05.2004)

Da in einem modernen E-Government die Frage der Kostenübernahme für die verschiedenen Systeme berücksichtigt und die Transparenz der Prozesse gewährleistet werden müssen, werden entsprechende **Beobachtungsfunktionen** eingeführt. So können durch den Vergleich der CPU- bzw. der Start-Stopp-Zeiten (Zeitkomponente t_i bzw. t_j) eines rechnergestützten Verwaltungsprozesses die Kosten ermittelt werden, die dann wiederum dem Bürger in Rechnung gestellt werden können. Die Ortskomponente (l_i bzw. l_j) dient der Lokalisierung der momentanen Ausführung des jeweiligen Verwaltungs-(teil-) prozesses. Zur Steigerung der Transparenz dient zusätzlich das **Verwaltungsverhalten** (VV_{ij}). Es ermöglicht die Protokollierung der Ausführung von Verwaltungsprozessen und deren Ergebnisse (Ablaufprotokoll, Fehler- und Nachrichtenprotokoll).

2.1.2 Systemebene - Verwaltungssystem (VS)

Der Verwaltungsprozess wird über einem Verwaltungssystem ausgeführt. Dieses beinhaltet die entsprechenden abstrakten Beschreibungen der Komponenten (Komponentenbeschreibungen) und damit die entsprechenden Verwaltungssystemschnittstellen. Das Verwaltungssystem (VS_{ij}) besteht aus den Zustandsbeschreibungen ZB_i, ZB_j und der Relation zwischen diesen beiden. Per Definition wird diese Relation mit dem Verwaltungsverfahren (VVf_{ij}) gleich gesetzt.

$$rel\left(ZB_i, ZB_j\right) \stackrel{!}{=} VVf_{ij}$$

Daraus ergibt sich:

$$VS_{ij} = (ZB_i, ZB_j, rel(ZB_i, ZB_j)) = (ZB_i, ZB_j, VVf_{ij}).$$

2.1.3 Verfahrensebene - Verwaltungsverfahren (VVf)

Das Verwaltungsverfahren besteht aus dem Anfangspunkt *i* und dem Endpunkt *j*. Es beinhaltet die einzelnen Funktionalitäten zur Bearbeitung der in der Verwaltungsaufgabe gestellten Teilaufgaben. Damit besteht das Verwaltungsverfahren aus einer Menge von Funktionskomplexen aus Nutzersicht (FK^N) und darunter liegenden Funktionen. Sie werden später in die konzeptuelle Ebene (als FK^K) überführt. Ein Funktionskomplex enthält die einzelnen Funktionen, um z.B. die Suche nach einem Grundstückeigentümer in einem Rechercheprozess durchzuführen.

Wichtig ist vor allem der Funktionskomplex *FK0*, welcher das Teilrechtssystem (*TRS*) enthält. Die bisherige Systementwicklung ignoriert weitestgehend die Implementierung

von bestehenden Rechtsvorschriften und überlässt damit die Einhaltung dem jeweiligen Nutzer. Innerhalb des Prozessmodells wird dagegen das TRS direkt mit implementiert, um damit mehr (Rechts-) Sicherheit zu schaffen und das Verwaltungssystem rechtskonform ablaufen zu lassen.

2.1.4 Zusammenfassende Darstellung

Nachdem das erweiterte Prozessmodell erläutert wurde, wird dieses in Abb. 2.6 als Fünf-Komponenten-Modell zusammenfassend schematisch dargestellt. Es wird noch einmal deutlich, dass ein direkter Zusammenhang zwischen dem Verwaltungsprozess (VP_{ij}), dem Verwaltungssystem (VS_{ij}) und dem Verwaltungsverfahren (VVf_{ij}) besteht.

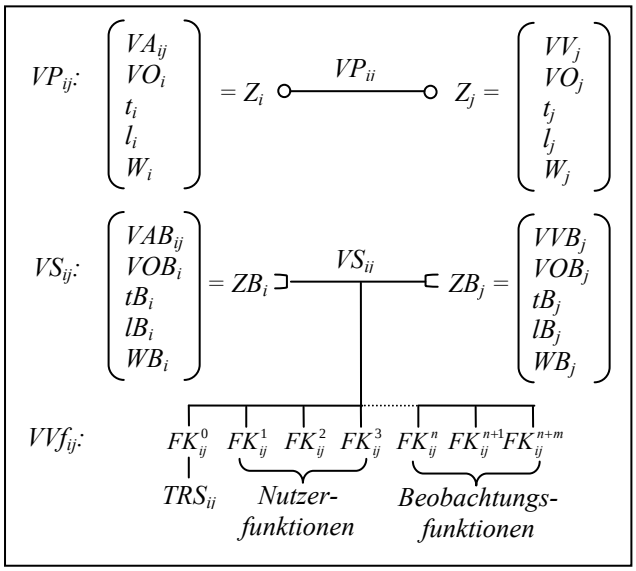


Abb. 2.6: erweitertes Prozessmodell als 5-Komponenten Modell

2.2 Telekooperation

Nachdem die Grundlagen des erweiterten Prozessmodells erläutert wurden, wechselt der Betrachtungsfokus auf die Telekooperation und damit auf die Zielstellung der nutzerorientierten Anwendungsdienstentwicklung. „[Die] Telekooperation bezeichnet die durch den Einsatz von Informations- und Kommunikationstechnik unterstützte Zusammenarbeit zwischen räumlich verteilten Kooperationspartnern“ (Engel (2001), S. 317).

Sollen systemübergreifende bzw. verteilte Anwendungen realisiert werden, muss ein geeignetes Verfahren für den Informationsaustausch sowie der Aufgabenübermittlung zwischen den jeweiligen kooperierenden Verwaltungssystemen existieren. Dabei muss

sichergestellt werden, dass die Verwaltungsprozesse verknüpfbar, das sendende und das empfangende Verwaltungssystem koppelbar und die Verwaltungsverfahren abstimmbar sind (siehe Kapitel 2.5). Dazu werden die Begriffe des Transportprozesses, des Telekooperationsprozesses sowie des -systems eingeführt.

2.2.1 Transportprozess (TP)

Der Transportprozess $TP_i^{E}{}_k^{A}$ stellt einen speziellen Verwaltungsprozess (das Transportobjekt TO_j^{E} entspricht dem TO_k^{A}) aus Nutzersicht dar und übernimmt die Übertragung der in einem anderen Verwaltungsprozess auszuführenden Aufgaben, sowie wenn notwendig eines Verwaltungsobjektes, an den aufzurufenden Verwaltungsprozess ($VP_k^{E}{}_{,l}^{A}$). Dazu muss der Verwaltungsprozess VP_{ij}^{A} ausgangsseitig ein entsprechendes Verwaltungsobjekt VO_j^{A} bereitstellen, welches sowohl die zu transportierenden Informationen inklusive der entsprechenden Verwaltungsaufgaben $VA_k^{E}{}_l^{A}$ des Zielverwaltungsprozesses $VP_k^{E}{}_l^{A}$ als auch die entsprechende Transportaufgabe $TA_j^{E}{}_k^{A}$ beinhaltet. Bei der Übergabe des VO_j^{A} an den Transportprozess $TP_j^{E}{}_k^{A}$, werden dann die Bestandteile wieder getrennt. Die enthaltene Transportaufgabe wird der Transportaufgabe $TA_j^{E}{}_k^{A}$ und die zu übermittelnden Informationen sowie die Verwaltungsaufgabe $VA_k^{E}{}_l^{A}$ dem Transportobjekt TO_j^{E} zugeordnet. Ist das Transportobjekt am Ziel eingetroffen, werden die enthaltenen Komponenten den jeweiligen Komponenten des Zielverwaltungsprozesses $VP_k^{E}{}_l^{A}$ übergeben ($TO_k^{A} \rightarrow VA_k^{E}{}_l^{A}, VO_k^{E}$). Sollte kein Verwaltungsobjekt übergeben worden sein, kann dies auch vom Zielverwaltungsprozess selbst aus externen Informationsquellen gefüllt werden. Die Abb. 2.7 stellt diesen Zusammenhang noch einmal mit den zuvor verwendeten Indexen der Komponenten grafisch dar.

Für die Erstellung der Transportaufgabe $TA_j^{E}{}_k^{A}$ muss zunächst das Übertragungsmedium gewählt werden. Die Medienwahl spielt eine große Rolle für die Laufzeit des Transportprozesses und damit für die Durchlaufzeit eines über mehrere örtlich getrennte Verwaltungsprozessgänge durchgeführten Verwaltungsprozesses. Während die persönliche Aktenübergabe, ein Telefonanruf, Fax, eine Email oder andere elektronische Übertragungsverfahren nur wenige Sekunden benötigen, sind es bei dem herkömmlichen Postweg zwei bis drei Tage. Zusätzlich besteht der Vorteil in der digitalen Übertragung darin, einen medienbruchfreien prozessübergreifenden Ablauf zu ermöglichen.

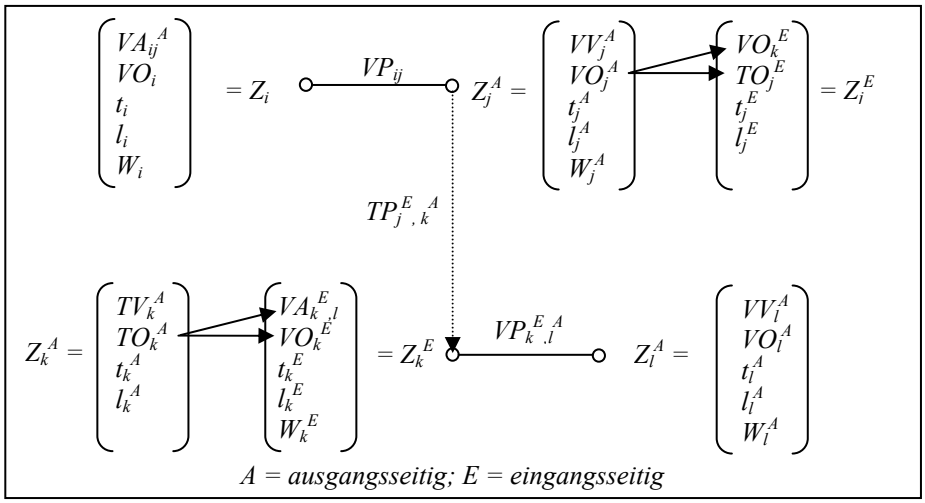


Abb. 2.7: Bsp. für den Zusammenhang zwischen dem Transportprozess und den Verwaltungsprozessen

Weiterhin müssen folgende Anforderungen an einen Transportprozess aus Nutzersicht gestellt werden (die Komponentenbezeichnungen beziehen sich dabei auf die Abb. 2.7):

- Die Teilaufgabe des Transportprozesses soll relativ einfach und kurz ausfallen und keine umfangreichen Strukturen benötigen. Sie dient lediglich der Aktivierung der Teilprozesse.

- Das Transportobjekt des Senders und Empfängers müssen übereinstimmen ($TO_j^E = TO_k^A$). Sollte dies nicht der Fall sein, kann der Empfänger im besten Fall („Best Case“) die Daten nicht auswerten und sendet eine Fehlermeldung. Im schlechtesten Fall („Worst Case“) interpretiert er diese falsch und verarbeitet sie dennoch, welches schwerwiegende Fehler verursachen kann.

- Die Zeitkomponente (t_j^E, t_k^A) besteht wiederum aus einer Struktur, welche unterschiedliche Zeitarten erfasst. Diese Erfassung ist zum einen für das Controlling aber auch für die Ermittlung und Weitergabe der bei der Ausführung des gesamten Verwaltungsprozesses entstandenen Kosten nötig. Dabei unterscheiden sich die Zeitausprägungen (tA_j^E, tA_k^A).

- Als letzte Anforderung ist die Übereinstimmung der Ortsbeschreibung (lB_j^E, lB_k^A) zu nennen. Dies ist wichtig, da z. B. nicht ohne weiteres eine Email abgesendet werden kann, wenn der Empfänger nur einen Faxeingang besitzt. Hierzu werden dann entsprechende Medienwandler benötigt.

Im Falle eines Rücktransportes oder einer weiteren Übergabe enthält das Transportobjekt ($TO_l^E{}_m^A$) die Informationen der Komponenten Verwaltungsverhalten ($VV_k^E{}_l^A$), Verwaltungsobjekt (VO_l^A) (wenn nötig), die Orts- (l_l^A) und Zeitangaben (t_l^A) sowie

das gesammelte Wissen (W_l^A) (wenn vorhanden) des zuvor ausgeführten Verwaltungsprozesses ($VP_k^{E\,A}{}_l$).

2.2.2 Telekooperationsprozess (TKP)

„Ein Telekooperationsprozess (TKP_{ij}) steuert ortsunabhängig kooperierende Verwaltungsprozesse auf der Basis existierender Telekooperationssysteme (*TKS*). Dabei stellen die Zustände Z_i, Z_j zu den Zeiten t_i, t_j die Anfangs- und Endzustände dar. Zusätzlich können die Zwischenzustände als ‚Beobachtungspunkte' angesehen werden" (Lüttich (2007b)). Zu deren Realisierung werden aus Nutzersicht entsprechende Transportprozesse, wie zuvor eingeführt, genutzt. Der Telekooperationsprozess entsteht, äquivalent zu den Verwaltungsprozessen, aus der Vereinigung der beteiligten Telekooperationsprozessgänge.

Die Abb. 2.8 stellt einen möglichen Telekooperationsprozess TKP_{ij} dar und verdeutlicht die folgenden Erläuterungen.

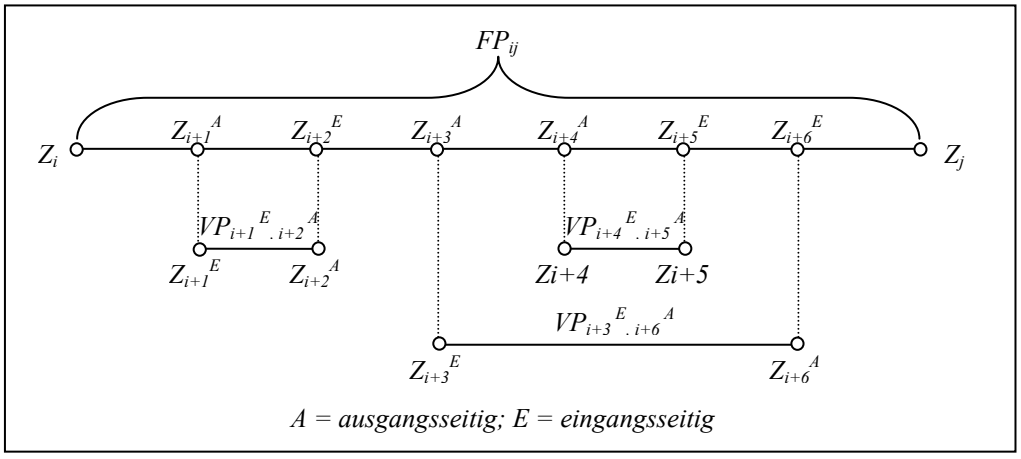


Abb. 2.8: Beispiel für einen Telekooperationsprozess mit darunter liegendem -system

Es gibt verschiedene Auffassungen, was genau unter einem Telekooperationsprozess zu verstehen ist (drei sollen an dieser Stelle genannt werden):

1. Auffassung

Zum einen wird der Telekooperationsprozess im engeren Sinne als Führungsprozess (FP_{ij}) definiert. Dies bedeutet, dass er zwar die Verwaltungsprozesse steuert, diese ihm aber nicht direkt zugeordnet werden. Damit ergibt sich (der Operator *a* entspricht der Ausführung eines Verwaltungssystems):

$$TKP_{ij} = FP_{ij},$$

$$TKP_{ij} = a(FS_{ij}),$$

und aus dem Beispiel: $TKP_{ij} = FP_{ij}$.

2. Auffassung

Zum anderen wird der Telekooperationsprozess im erweiterten Sinne als Vereinigung des Führungsprozess mit den zugehörigen Verwaltungsprozessen angesehen. Daraus resultiert (mit *a* als Operanden für die Ausführung, *k* als Index für die beteiligten Verwaltungsprozesse):

$$TKP_{ij} = FP_{ij} \cup \bigcup_k VP_k \ ,$$

$$TKP_{ij} = aFS_{ij} \cup \bigcup_k a\ \left(VS_k, t^k\right),$$

und aus dem Beispiel: $TKP_{ij} = FP_{ij} \cup VP_{i+1^E, i+2^A} \cup VP_{i+3^E, i+6^A} \cup VP_{i+4^E, i+5^A}$.

3. Auffassung

In einer dritten Auffassung wird unter einem Telekooperationsprozess die Vereinigung des Führungsprozesses mit den jeweiligen Verwaltungsprozessen und den notwendigen Transportprozessen verstanden. Dies kann wie folgt dargestellt werden (mit a als Operanden für die Ausführung, k / l als Index für die beteiligten Verwaltungs- / Transportprozesse):

$$TKP_{ij} = FP_{ij} \cup \bigcup_k VP_k \cup \bigcup_l TP_l \ ,$$

$$TKP_{ij} = aFS_{ij} \cup \bigcup_k a\ \left(VS_k, t^k\right) \cup \bigcup_l a\left(TS_l, t^l\right),$$

und aus dem Beispiel:

$$TKP_{ij} = \begin{array}{l} FP_{ij} \cup VP_{i+1^E, i+2^A} \cup VP_{i+3^E, i+6^A} \cup VP_{i+4^E, i+5^A} \cup TP_{i+1^A, i+1^E} \cup TP_{i+2^E, i+2^A} \cup \\ TP_{i+4^A, i+4^E} \cup TP_{i+6^E, i+6^A} \cup TP_{i+4^A, i+4^E} \cup TP_{i+5^E, i+5^A} \end{array}.$$

Diese verschiedenen Auffassungen können durch die unterschiedlichen Nutzerklassen und damit durch die differenzierten Sichtweisen auf ein Telekooperationssystem begründet werden. Die dritte Auffassung entspricht der Sichtweise der kompetenten Nutzerklasse (siehe Kapitel 2.3).

Unabhängig davon, welcher Überlegung gefolgt wird, muss der Führungsprozess mindestens die Verwaltungsteilaufgaben an die einzelnen beteiligten Verwaltungs- und

Transportprozesse übergeben. Wenn es notwendig ist, kann auch das jeweilige Verwaltungsobjekt mit übergeben werden. Dies wäre der Fall, wenn es an einen für den Teilprozess nicht zugänglichen Ablageort (entweder auf einem lokalen Arbeitsplatzrechner oder in Papierform in einem Büro) liegt.

Der Telekooperationsprozess kann sowohl innerhalb einer Institution, als auch institutionsübergreifend durchgeführt werden und dabei unabhängige als auch abhängige Prozesse steuern. Im Falle von abhängigen Prozessen wird der Führungsprozess zur Koordination und Kontrolle benötigt.

2.2.3 Telekooperationssystem (TKS)

Telekooperationssysteme sind die Softwarewerkzeuge, mit denen Verwaltungsprozesse über räumliche Entfernungen unterstützt und ausgeführt werden können. Dabei liegt der Fokus auf drei Zielgrößen: sie können einzelne Arbeitsplätze, ganze Arbeitsgruppen, oder aber auch die Gesamtorganisation unterstützen. Zusätzlich kann sich die Anwendung des Telekooperationssystems auf eine bestimmte Institution beschränken, oder institutionsübergreifend eingeführt werden.

Wichtig ist, dass die Inbetriebnahme (Planung, Installation, Nutzung) organisiert und abgestimmt durchgeführt wird. Bisherige Praxis ist, dass jede qualifizierte Gemeinde ihre eigenen speziellen Anwendungssysteme einsetzt. So ist es schwierig standardisierte, kooperierende Systeme zu entwickeln und einzuführen.

Telekooperationssysteme sind z. B. Email, Gruppenablagen, verteilte Datenbanken, Audio- und Videokonferenzsysteme mit Application-Sharing, Workflow-Managementsysteme, Vorgangsunterstützungssysteme, Verzeichnisdienste, Protokollierungs- und Benachrichtigungsdienste (vgl. Engel (2001), S. 318). Für diese Arbeit werden vor allem die Web Service Basiskomponenten benötigt. Darunter fallen u. a. die Applikationsserver und die Verzeichnisdienste (siehe Kapitel 6.2).

2.2.4 Erweitertes Trägermodell

Eine zusätzliche Erweiterung des Prozessmodells mittels des Trägermodells nach Lüttich (Lüttich (2006)) bietet die Möglichkeit, einen Verantwortlichen für:

- die Stellung der Verwaltungsaufgabe (Träger der Verwaltungsaufgabe),
- die Ausführung des Verwaltungsprozesses (Träger des Verwaltungsprozesses),

- das Verwaltungssystem (Träger des Verwaltungssystems) und
- des Verwaltungsverfahrens (Träger des Verwaltungsverfahrens),

zu definieren. Dies ermöglicht es ohne Weiteres Nutzerdienste, welche von den einzelnen Nutzerklassen über das Internet genutzt werden können, einzuführen. Nach geltendem Recht in der Bundesrepublik Deutschland muss die Verantwortung für die Sicherheit eines Verwaltungssystems bei der zuständigen Institution liegen. Demnach wurden bisher alle Anwendungen auch dort ausgeführt. Mittels des Trägermodells ist es nun möglich, die Institution als Träger des Verwaltungsverfahrens zu definieren. Der Nutzer eines Verwaltungssystems, aus einer Nutzerklasse NK_{kl}, wird als Träger der Verwaltungsaufgabe definiert. Damit ist dieser für die Richtigkeit der übertragenen Informationen verantwortlich. Die Ausführung des Verwaltungsprozesses sowie des Verwaltungssystems können wiederum in die Verantwortlichkeit einer anderen Institution gelegt werden. Dies wäre z. B. ein Bürgerportal, welches von einem Dienstleister, dem Träger des Verwaltungsprozesses, betrieben wird. Das Verwaltungssystem könnte dabei in einem entsprechenden Rechenzentrum ausgeführt werden.

2.3 Nutzerklassen

Insbesondere die Verwaltungsinformatik steht vor dem Problem, dass Verwaltungssysteme für eine sehr große Masse an nicht homogenen Nutzern erstellt werden müssen. Es kann z. B. nach Nutzern

- mit Internet- und Verwaltungserfahrung,
- ohne Internet- und mit Verwaltungserfahrung,
- mit Internet- und ohne Verwaltungserfahrung,
- ohne Internet- und ohne Verwaltungserfahrung,

unterschieden werden.

Um dieses Problem einzugrenzen und zu lösen, werden Nutzerklassen eingeführt. Eine Nutzerklasse ist eine weitestgehend homogene Menge von Nutzern eines Verwaltungssystems, welche ähnliche Eigenschaften und Sichtweisen auf einen Verwaltungsprozess hat. Als Beispiel, speziell im Bereich der öffentlichen Verwaltungen, können die in Tab. 2.2 genannten Nutzerklassen (NK_{kl}) definiert werden.

Tab. 2.2: Beispiel für verschiedene Nutzerklassen

Nutzerklasse	Repräsentanten
NK_1	Amtsleiter
NK_2	Sachbearbeiter
NK_3	qualifizierte Bürger: verwaltungs- und interneterfahrene Bürger
NK_4	unqualifizierte Bürger: verwaltungs- und internetunerfahrene Bürger

Aufgrund der verschiedenen Nutzersichten muss eine Nutzerklasse definiert werden, aus deren Betrachtungsweise dann die nutzerorientierte Anwendungssystementwicklung stattfindet. Es ist im Beispiel davon auszugehen, dass der Amtsleiter vollständige Kenntnisse über existierende Verwaltungsprozesse, ihre Interaktionen und deren Abläufe sowie die notwendigen Bestimmungen hat. Damit erfüllen die Amtsleiter alle Voraussetzungen, um ein Anwendungssystem aus Nutzersicht generieren zu können und bilden damit die kompetente Nutzerklasse. Die anderen Nutzerklassen haben nur Teilsichten auf einen Verwaltungsprozess und kennen nicht alle Zusammenhänge. Dennoch muss es auch ihnen möglich sein, das entstandene Verwaltungssystem zu nutzen und ein gewünschtes Ergebnis zu erzielen. Ein Beispiel (in Anlehnung an die zugeordneten Nutzerklassen in Tab. 2.2) für diesen Zusammenhang ist in Abb. 2.9 dargestellt.

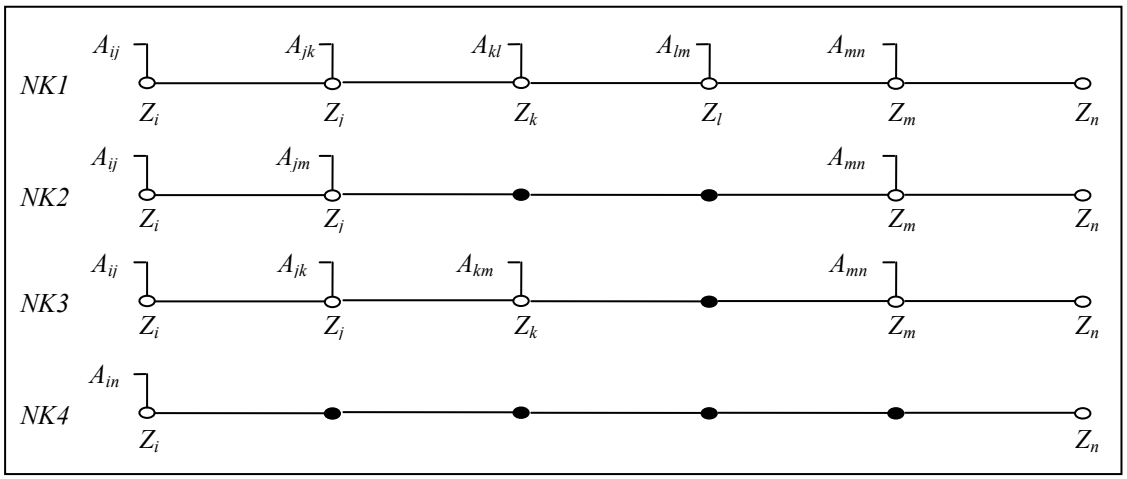


Abb. 2.9: Sichtweisen verschiedener Nutzerklassen auf eine Kette von Verwaltungsprozessen

Die schwarzen Zwischenzustände werden von der jeweiligen Nutzerklasse nicht erkannt. Dabei bildet die Nutzerklasse *NK1* die kompetente Nutzerklasse ab. Die Nutzerklasse *NK4* dagegen könnte einen verwaltungsunerfahrenen Bürger darstellen. Bisherige Ansätze der Softwareentwicklung ermöglichen ein solches Vorgehen nicht, da den Amtsleitern die Kenntnisse einer Programmiersprache zum Erstellen eines Verwaltungssystems in den überwiegenden Fällen fehlen.

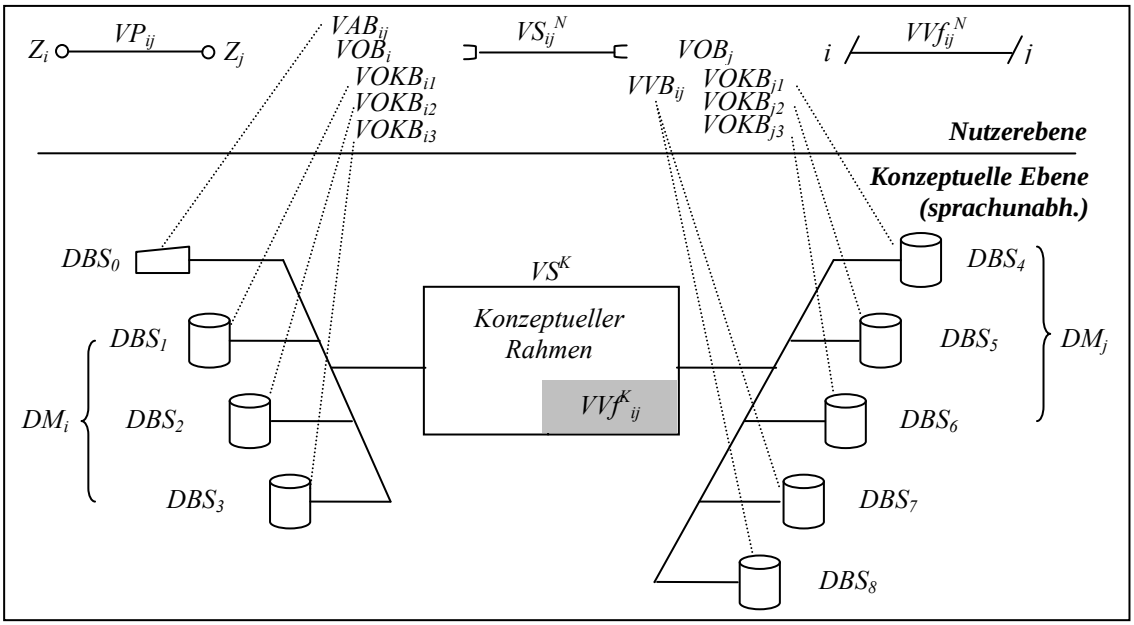
2.4 Überführung eines Verwaltungssystems in die konzeptuelle Ebene

Die nutzerorientierte Systementwicklung nach Lüttich basiert auf dem Konzept des Generierens von Quellcode. Dabei wird der konzeptuelle Rahmen soweit standardisiert, dass dieser vollständig generiert und lediglich die eigentlichen Funktionalitäten des Verwaltungssystems individuell programmiert werden müssen. Hier gibt es zusätzliche Möglichkeiten diesen Aufwand zu reduzieren, in dem bereits erstellte Verwaltungsverfahrenskomponenten oder Verwaltungsteilsysteme wiederverwendet werden.

Bei der Transformation der nutzerorientierten Modellierung in die konzeptuelle Ebene (zunächst sprachunabhängig), werden die einzelnen Verwaltungsobjektkomponentenbeschreibungen ($VOKB_{ix}$ und $VOKB_{jx}$ für x = 1,2,3) des Verwaltungssystems in Datenbeschreibungsstrukturen (DBS_x für x = 1,2,3) und damit in das Datenmodell DM_i bzw. DM_j überführt. Hierzu kommen Softwareentwicklungswerkzeuge zum Einsatz (siehe Kapitel 2.5).

Eine Information besteht aus dem Tripel: Informationsquelle, -senke und der eigentlichen Mitteilung. Diese bestehen wiederum aus einer Beschreibung und einer Ausprägung. Wenn Informationen auf einem Datenspeicher als Daten abgelegt werden, wird im Allgemeinen nur die Informationsausprägung auf einem logischen Datenspeicher abgelegt. Damit können die als Daten abgelegten Informationen nur sinnvoll genutzt werden, wenn das lesende Programm die entsprechende Informationsbeschreibung besitzt und somit die Zuordnung wiederherstellen kann. Ein weiteres Problem besteht in der gängigen Praxis darin, dass bei der Umwandlung der Information in Daten die Informationsquelle und -senke nicht abgebildet werden und somit die Information im eigentlichen Sinne nur eine Mitteilung darstellt, deren Wahrheitswert (aufgrund der Bewertung der Quelle) nicht mehr nachvollzogen werden kann. Als Beispiel sei hier die Speicherung einer Verkehrsflusszählung genannt. Auf dem Datenspeicher ist folgendes Bild abgelegt: $B = \{10, 15, 5, 0\}$. Ohne die Informationsbeschreibung besitzen diese Daten keinen Wert, da sie nicht interpretiert und zugeordnet werden können. Besitzt eine lesende Anwendung die Informationsbeschreibung $IB = \{PKW, LKW, Motorrad, Fußgänger\}$, können die Daten interpretiert werden und besitzen einen entsprechenden Informationsgehalt. Wurden die Ausprägungen der Informationsquelle und -senke nicht mit abgelegt, kann allerdings der Wahrheitsgehalt nicht mehr nachvollzogen werden.

Das Verwaltungssystem auf der konzeptuellen Ebene (VS^K) besteht aus dem konzeptuellen Rahmen und den einzelnen überführten Funktionskomplexen des Verwaltungsverfahrens. In Abb. 2.10 ist der erläuterte Zusammenhang bei der Überführung eines Verwaltungssystems aus der Nutzerebene in die sprachunabhängige konzeptuelle Ebene einmal grafisch dargestellt.



Quelle: in Anlehnung an Lüttich (2007a)

Abb. 2.10: Nutzerorientierter Ansatz zur Entwicklung von Verwaltungssystemen auf der Basis eines zwei Komponenten-Modells der Zustände

Der konzeptuelle Rahmen basiert nach Lüttich auf dem Modulkonzept und besteht aus einem Hauptsteuermodul (*HSM*), welches die Steuerung und Koordinierung der auszuführenden Aufgaben übernimmt. Zusätzlich werden folgende Module bereitgestellt:

- Anfangsmodul (*A*)
  Das Anfangsmodul übernimmt die Initialisierung von Systemparametern (z. B. für die Zeitmessung) sowie das Öffnen der benötigten Eingabequellen (Zugriff auf Dateien oder Datenbanken).

- Eingabemodule (B_x)
  Die Eingabemodule stellen die entsprechenden Schnittstellen zu den benötigten Eingabequellen dar und werden für den Zugriff auf diese benötigt.

- Kontrollmodul (*D*) mit den jeweiligen Erkennungskomplexen (G_x)
  Das Kontrollmodul übernimmt die Kontrolle der Datenstrukturen und überwacht deren Wechsel. Als Beispiel können folgende Module definiert werden: bei einem Dateizugriff: G_1: Objektwechsel, G_2: Komponentenwechsel, G_3: Satzartwechsel, und bei einem Datenbankzugriff: G_1: Datenbankwechsel, G_2: Tabellenwechsel, G_3: Attributwechsel.

- Ausführungsmodul (*E*) mit den jeweiligen Funktionskomplexen (H_x)
  Das Ausführungsmodul beinhaltet innerhalb der einzelnen Funktionskomplexe die eigentlichen Funktionalitäten des Verwaltungssystems.

- Ausgabemodule (J_x)
  Das Ausgabemodul stellt, ähnlich dem Eingabemodul, die Schnittstellen zum Zugriff auf die benötigten Ausgabesysteme bereit.

- Abschlussmodul (*F*)
  Das Abschlussmodul übernimmt, als Gegenstück zum Anfangsmodul, das Schließen der Eingabequellen und der Ausgabesysteme sowie die Auswertung und Speicherung der benötigten Systemparameter.

In Abb. 2.11 ist das Verwaltungssystem aus konzeptueller Sicht einmal grafisch dargestellt. Dabei bildet der weiße Bereich den konzeptuellen Rahmen, welcher nicht programmiert werden muss, sondern generiert werden kann.

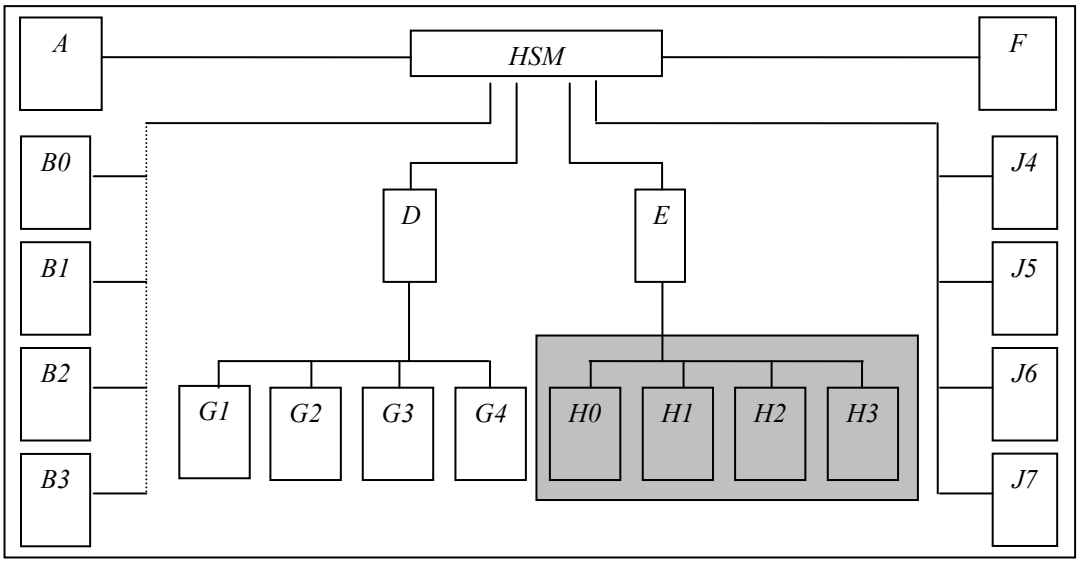


Abb. 2.11: Verwaltungssystem aus konzeptueller Sicht

2.5 Softwareentwicklungswerkzeug

Wie schon im vorherigen Teilkapitel erwähnt, werden Softwarewerkzeuge benötigt, um ein Verwaltungssystem aus Nutzersicht zu modellieren und dann in die konzeptuelle Ebene zu überführen. Dazu müssen einige Anforderungen an ein Softwareentwicklungswerkzeug aus Nutzersicht gestellt werden. Für die Erstellung eines Verwaltungssystems bzw. eines Anwendungsdienstes müssen die verwendeten Verwaltungsverfah-

ren abstimmbar, die Verwaltungssysteme koppelbar und die Verwaltungsprozesse verknüpfbar sein. Dieser Zusammenhang wird im Folgenden erläutert.

Abstimmbarkeit der Verwaltungsverfahren

Die Verwaltungsverfahren realisieren die einzelnen Funktionalitäten der Verwaltungssysteme. Dabei ist es notwendig, dass die zu verarbeitenden Informationen sowohl syntaktisch als auch semantisch richtig interpretiert werden (für weitere Erläuterungen wird u. a. auf Ernst (2000), S. 240 f. verwiesen). Dies bedeutet, dass bei der Übertragung von Informationen nicht nur die richtigen Datentypen verwendet werden, sondern auch die gleichen Bezeichnungen innerhalb der Informationsausprägungen. Ein Beispiel soll dies verdeutlichen: In einem Verwaltungssystem wird die Berufsbezeichnung mittels eines standardisierten Verzeichnisses durch eine Integer-Zahl ausgedrückt. In einem anderen Verwaltungssystem wird dieselbe Information durch eine Zeichenkette abgelegt. Soll nun die Information aus dem ersten in das zweite Verwaltungssystem übertragen werden, muss die Transformation der Integer-Zahl in eine Zeichenkette erfolgen. Dies ist mittels des Verzeichnisses leicht zu realisieren. Dennoch ist im zweiten System nicht sichergestellt, dass die Berufsbezeichnung richtig interpretiert wird, da hier kein standardisiertes Verzeichnis genutzt wird. Somit sind ähnliche Bezeichnungen keine Besonderheit. Dies erschwert allerdings den Informationsaustausch und die -auswertung. Die Abstimmbarkeit wird durch die Kopplung der Verwaltungssysteme gewährleistet.

Kopplung der Verwaltungssysteme

Die Kopplung bzw. Integration von Verwaltungssystemen kann auf unterschiedlichen Ebenen erfolgen. Die älteste Variante ist die manuelle Kopplung. Dabei werden die Ergebnisse des Verwaltungsauftrages im jeweiligen Verwaltungssystem bearbeitet und anschließend für die Weiterbearbeitung in einem anderen Verwaltungssystem ausgedruckt und weitergegeben. Dann müssen diese Informationen dort wieder eingepflegt werden, um den Auftrag weiterbearbeiten zu können. Mit der Weiterentwicklung der EDV wurden verschiedene Formen der automatischen Kopplung auf der Mittel- oder Verarbeitungsschicht und der Persistenz- oder Datenhaltungsschicht entwickelt.

Bei der datenorientierten Kopplung erfolgt die Integration auf der Persistenzschicht. Dabei wird entweder eine gemeinsame Datenbasis in einer Datenbank angelegt und der Zugriff der einzelnen Verwaltungssysteme auf diese umgelenkt oder aber es erfolgt eine redundante Datenhaltung in den einzelnen Verwaltungssystemen. In beiden Fällen wer-

den die Ergebnisse der Bearbeitung in einem System so abgespeichert, dass diese in das andere System automatisch übernommen werden.

Bei der Kopplung auf der funktionalen Ebene erfolgt eine prozessorientierte Kapselung der Funktionalitäten der Verwaltungssysteme in Dienste. Anschließend werden diese durch die Übertragung von Nachrichten (meist XML) oder direkten Funktionsaufrufen (u. a. RPC) aufgerufen. Der Zugriff wird dabei durch die Bereitstellung von Schnittstellen ermöglicht. In dieser Arbeit werden dafür die Web Service Technologien verwendet.

Für eine Kopplung aus der Sicht der nutzerorientierten Systementwicklung ist es notwendig, dass die verwendeten Verwaltungsobjekte nutzerobjektfähig sind und somit zumindest den datenorientierten Aspekt (siehe Kapitel 2.1.1) erfüllen. Damit kann gewährleistet werden, dass diese von verschiedenen Verwaltungssystemen genutzt werden können.

Verknüpfung der Verwaltungsprozesse

Ein Verwaltungsprozess kann erst mit einem oder auch mehreren anderen verknüpft werden, wenn die darunterliegenden Verwaltungssysteme koppelbar und die Verwaltungsverfahren abstimmbar sind. Sind diese beiden Anforderungen gegeben, können die Verwaltungsprozesse verknüpft werden. Dabei wird in der Regel das Verwaltungsobjekt des vorherigen Verwaltungsprozesses zum Verwaltungsobjekt des nachfolgenden. Zusätzlich besteht die Möglichkeit, dass das Verwaltungsobjekt und/oder das Verwaltungsverhalten des Verwaltungsprozesses zur Verwaltungsaufgabe des Nachfolgeprozesses werden.

Nachdem eine Einführung in die nutzerorientierte Systementwicklung erfolgte, wird nun die Web Service Technologie vorgestellt, um dann im nächsten Kapitel den erweiterten Dienst einzuführen und dessen nutzerorientierte Entwicklung voranzutreiben.

3 Die Web Service Technologie

Dieses Kapitel stellt die technischen Grundlagen für eine Verwendung der Web Service Technologie dar, um diese anschließend für die nutzerorientierte Anwendungsdienstentwicklung zu instrumentalisieren. Web Services bieten „[…] eine Schnittstelle auf Softwareoperationen über ein Netzwerk (z.B. Internet), das mittels Nachrichtenbasiertem XML-Datentransfer aufrufbar ist (z.B. SOAP, WSDL), an“ (Dustdar u. a. (2003), S. 113). Dabei werden weit verbreitete Standards (meist an dem Namensvorsatz „WS“ zu erkennen) verwendet, welches einen system- und herstellerübergreifenden Einsatz ermöglicht. Dadurch können hohe Kosten für die Systemanpassung, bedingt durch Inkompatibilitäten unterschiedlicher Produkte, vermieden werden. Die notwendigen Standards werden vom „World Wide Web Consortium” (W3C), der „Web Service Interoperability Organization” (WS-I) sowie der „Organization for the Advancement of Structured Information Standards” (OASIS) verabschiedet.

Die Web Service Technologie stellt eine Realisierungsform der serviceorientierten Architektur (SOA) dar. Diese Architektur stellt sowohl ein Management- als auch ein Technologiekonzept dar und beschreibt die Serviceorientierung von Softwaresystemen und -komponenten. Der Schwerpunkt einer SOA besteht darin, „[…] dass IT-Funktionen in einem Unternehmen [oder einer Institution der öffentlichen Verwaltung] als Dienste (‚Services’) zur Verfügung gestellt werden“ (Kircher (2007), S. 8). Dazu werden die Rollen des „Service Provider“, „Service Consumer“ und des „Service Broker“ unterschieden. Das dreiteilige Konzept kann auch mittels der Web Service Technologie umgesetzt werden und wird im Folgenden genauer erläutert:

Der „**Service Provider“** ist ein Dienstanbieter und stellt die entsprechende IT-Infrastruktur zur Ausführung und Speicherung der Web Services zur Verfügung. Anders als in der freien Wirtschaft, in der Sicherheits- und Vertrauensprobleme existieren wenn Anwendungssysteme mit sensiblen Daten an andere Dienstleister ausgelagert werden sollen, kann dies in der öffentlichen Verwaltung als Vorteil angesehen werden. Hierzu kann der Träger des Verwaltungsverfahrens (siehe Kapitel 2.2.4) der jeweiligen Institution zugeordnet werden, welche laut Gesetz zur Ausführung der zugrunde liegenden Aufgaben zuständig ist. Damit kann der Verwaltungsvorgang von einem Nutzer unabhängig vom Ort (z. B. über das Internet) ausgelöst werden, da dieser den Träger der Verwaltungsaufgabe repräsentiert.

Der **„Service Consumer“** nutzt einen oder entsprechend mehrere Web Services. Dabei können folgende Rollen auf einen Service zugreifen:

- ein Nutzer der Nutzerklasse NK_{kl}, welcher über ein Frontend oder ein Anwendungsprogramm zugreift,

- ein anderer Web Service, welcher diesen nutzt um eigene Funktionalitäten anzubieten.

Der „**Service Broker“** ist in der Regel ein auf dem Standard UDDI basierender Verzeichnisdienst. Es können auch andere Protokolle verwendet werden, wobei dann allerdings von der Standardarchitektur abgewichen wird. Im „Service Broker“ werden alle Web Services von den verschiedenen „Service Providern“ registriert („publish“), damit der „Service Consumer“ diese finden („find“) und nutzen („bind/invoke“) kann. Für die Kommunikation zwischen diesen drei Rollen werden die Protokolle SOAP, XML-RPC und WSDL eingesetzt, welches auf dem XML-Format basieren. Die Abb. 3.1 verdeutlicht diesen Zusammenhang.

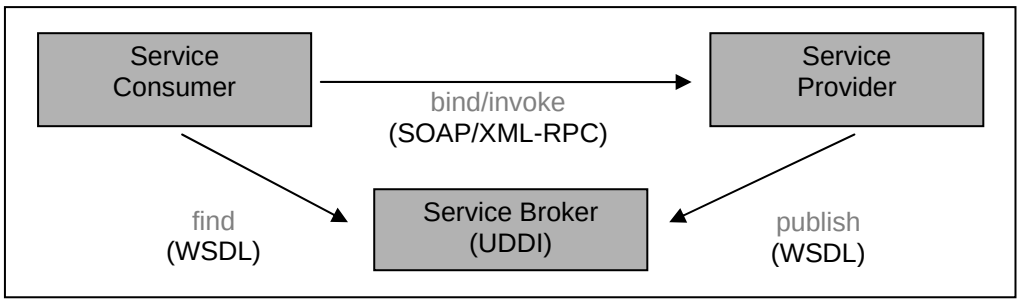


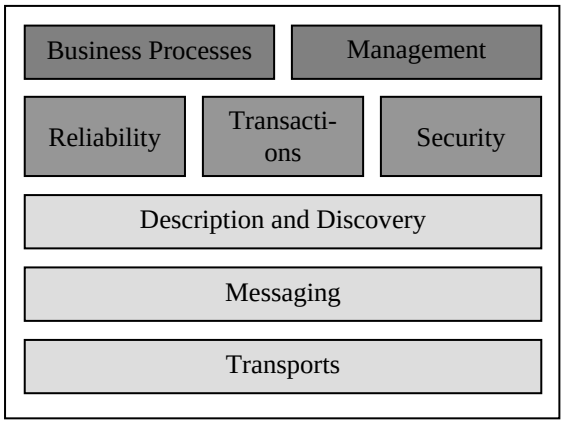
Abb. 3.1**:** Web Service Modell

Da eine Vielzahl von Standards und Spezifikationen für die Nutzung von Web Services existieren, haben verschiedene Hersteller eigene Frameworks spezifiziert um eine effektive Nutzung zu ermöglichen.

3.1 Web Service Framework

Zur Realisierung einer Web Service Infrastruktur, müssen die unterschiedlichen Standards zu einem konzeptuellen Rahmen (Framework) zusammengefügt werden. In dieser Arbeit wird beispielhaft ein Framework von IBM, wie in Abb. 3.2 dargestellt, erläutert.

Die Basis bilden die Transporttechnologien. Darauf setzen die Nachrichtenprotokolle für den Nachrichtenaustausch (Messaging) in dezentralen und verteilten Umgebungen auf. Eine Servicebeschreibung wird im WSDL-Format erstellt und anschließend auf einen „Universal Description, Discovery and Integration“ (UDDI) Server publiziert („Description and Discovery“). Die genannten Komponenten stellen die Grundfunktionalität für die Nutzung von Web Services bereit.



Quelle: vgl. IBM (2008a)

Abb. 3.2**:** IBM Web Service Framework

Zusätzlich wurde der Funktionsumfang um notwendige Business Funktionen erweitert, um eine Alternative zu etablierte Lösungen (z. B. EAI, CORBA) für ein serviceorientiertes Unternehmen darzustellen. Im IBM Web Service Framework sind diese zusätzlichen Funktionen: Reliability, Transactions, Security und die Business Processes. Auf mögliche andere Erweiterungen wird aufgrund des Fokus dieser Arbeit nicht eingegangen. Sie können aber auf den Webseiten der W3C (W3C (2008a)) und des OASIS-Konsortiums (OASIS (2008a)) eingesehen werden. Im Folgenden werden die einzelnen Bestandteile des IBM Web Service Framework näher betrachtet.

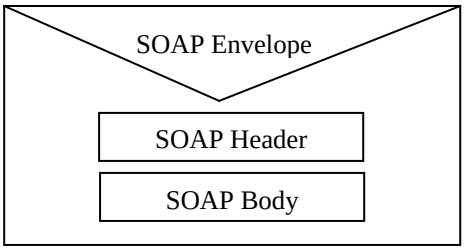
Transports

Für den Datentransport wird bei der Web Service Technologie überwiegend das standardisierte HTTP Protokoll verwendet. Dies hat den Vorteil, dass eine Kommunikation auch durch eine Firewall hinweg erfolgen kann. Zusätzlich wird dieses prinzipiell auf jedem Rechner verwendet, sodass hier keine zusätzlichen Kosten für die Einführung entstehen. Beschränkungen für die Protokollwahl gibt es hier allerdings nicht. Somit sind auch andere Übertragungsmechanismen wie SMTP, FTP, JMS, RMI oder nicht standardisierte Übertragungsprotokolle denkbar. Für weitere Informationen zur Datentransportschicht sei u. a. auf Tanenbaum (2003) verwiesen.

Messaging

Der Nachrichtenaustausch findet mittels SOAP-Nachrichten statt. Diese sind XML-Dokumente, die der SOAP Spezifikation (vgl. W3C (2007b)) unterliegen. Eine solche Nachricht kann als ein Brief visualisiert werden (vgl. Abb. 3.3). Der SOAP-Envelope

(zu Deutsch: Umschlag) beinhaltet die eigentliche Nachricht und bildet das XML-Wurzelelement. Damit kapselt er den „Header“ und den „Body“. Der Umschlag wird durch das Element <env:Envelope …> geöffnet und am Ende durch </env:Envelope> geschlossen. Der „Uniform Resource Identifier“ (URI) gibt den Namensraum, also die zur Verfügung stehenden Elemente einer SOAP-Nachricht, an.



Quelle: Dostal u. a. (2005), S. 48

Abb. 3.3**:** Grundlegender Aufbau einer SOAP-Nachricht

Der SOAP Header ist kein zwingender Bestandteil. Er kann aber genutzt werden, um Informationen, die nicht zu den eigentlichen auszutauschenden Daten gehören, abzulegen. Diese Bestandteile sind nicht in der SOAP Spezifikation enthalten, sodass diese einzeln spezifiziert und standardisiert werden. Durch die Standardisierung werden eine große Kompatibilität und Interaktion mit anderen Systemen ermöglicht. Hier sind u. a. Spezifikationen für die im Framework genannten Komponenten: „Reliability“, „Transactions“ und „Security“ zu nennen. Für jeden Header-Eintrag kann über ein Attribut namens „mustUnderstand“ festgelegt werden, ob der Empfänger dieser Nachricht dieses Attribut zwingend auswerten muss bzw. dieses auch ignorieren kann. Der Headerblock wird durch die Elemente <env:Header> </env:Header> gekapselt.

Im Body werden die zu übertragenden Informationen abgelegt. Hier existieren keine festen Spezifikationen. Wichtig sind hier allerdings die Wohlgeformtheit der XML-Elemente (zu jedem öffnenden, auch ein schließendes Element) und ein definierter Namensraum. Des Weiteren können im Body durch das Element <env:Fault> auch Fehlernachrichten übertragen werden.

Für den Nachrichtenaustausch kommen zwei Verfahren in Frage. Der „Remote Procedure Call“ (RPC), bei dem in der Nachricht die aufzurufende Funktion inklusive Parametern enthalten ist, ermöglicht eine synchrone Kommunikation. Dies bedeutet, dass der Sender so lange seine Ausführung unterbricht, bis er eine Antwort erhält. Die andere Variante ist ein dokumentenbasierter Austausch. Hier erfolgt eine asynchrone Kommunikation und der Empfänger entscheidet anhand der erhaltenen Informationen, welche

Funktionen er ausführen muss und ob eine Antwortnachricht gesendet wird. Im Folgenden ist das Basisgerüst einer SOAP-Nachricht dargestellt:

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-
envelope">
     <env:Header>
          header information+
     </env:Header>
     <env:Body>
          Content+
     </env:Body>
</env:Envelope>
```

Description and Discovery

Das Suchen, Finden und Veröffentlichen von Web Services wird über einen Verzeichnisdienst namens „Universal Description, Discovery & Integration“, oder kurz UDDI, realisiert. Zur Beschreibung eines Services dient ein auf XML basierendes WSDL-Dokument. Diese Informationen spiegeln die technischen Aspekte wieder. Zusätzlich werden im Verzeichnis auch Daten über den Ersteller bzw. das anbietende Unternehmen sowie zusätzliche Informationen, z. B. zu verschiedenen Service Level Agreements (SLA) und Versionen von Services, abgelegt. Des Weiteren kann eine Suche nach alternativen Services realisiert werden, wenn diese Informationen hinterlegt sind. Für nähere Informationen zum Aufbau und der Funktionsweise von UDDI bzw. WSDL sei auf die jeweiligen Spezifikationen in UDDI (2008a) bzw. W3C (2008b) verwiesen.

Reliability

In den Institutionen der öffentlichen Verwaltung ist es, genauso wie in Unternehmen, sehr entscheidend zu wissen, ob eine gesendete Nachricht auch ihre Empfänger erreicht hat. Dazu wurde die Spezifikation „WS-ReliableMessaging“ (OASIS (2004)) eingeführt. Diese ermöglicht eine Rückbestätigung über den erfolgreichen Empfang und damit ein sicheres Zustellen an den Absender. Zusätzlich gibt es die Möglichkeit des Sendens einer Sequenz von Nachrichten. Dann müssen diese genau in der richtigen Reihenfolge beim Empfänger eintreffen. Dazu werden sie nacheinander gesendet und zwar erst dann, wenn der Empfang der vorhergehenden Nachricht erfolgreich war (evtl. wird das Senden einzelner Nachrichten wiederholt). Genauere Erläuterungen bietet die oben genannte Spezifikation.

Transactions

In der Datenbanktheorie gelten für Transaktionen die Eigenschaften der Atomarität, Konsistenz (engl. „consistency“), Isoliertheit und der Dauerhaftigkeit (ACID). Diese Eigenschaften können bei Web Services allerdings nur selten genutzt werden, da hier Vorgänge oft über mehrere Stunden ablaufen. In diesem Fall können Datenbereiche einer Datenbank nicht solange blockiert werden. Daher gibt es zwei unterschiedliche Standards: die „Atomic Transactions“ (OASIS (2007a)) für atomare Transaktionen, die die ACID Eigenschaften erfüllen, und die „Business Activities“ (OASIS (2007b)) für lang laufende Transaktionen. Aufgrund der Langläufigkeit werden die einzelnen Transaktionen sofort durchgeführt. Wenn ein Fehler an einer beliebigen Stelle auftritt, wird die gesamte Transaktion durch ein entsprechend vorher festgelegtes Verhalten der einzelnen beteiligten Services kompensiert.

Security

Die Web Service Technologie ermöglicht eine moderne Realisierung einer SOA. Im Verwaltungsumfeld werden viele sensible oder sogar geheime Informationen genutzt und ausgetauscht. Daher ist es neben dem gesicherten Zustellen sowie der Transaktionsverwaltung wichtig, dass für die Nutzung von Web Services Sicherheitsstandards vorliegen. Es existieren hier folgende Sicherheitsanforderungen (vgl. Dostal u. a. (2005), S. 165):

- Vertraulichkeit: Nur der Empfänger sollte die Nachricht lesen können.
- Berechtigung: Der jeweilige Dienstanforderer muss auch berechtigt sein, den Dienst zu nutzen.
- (Daten-) Konsistenz: Die Nachricht muss unverfälscht ihren Empfänger erreichen.
- Glaubwürdigkeit: Die Nachricht muss klar einem Absender zugeordnet sein und es darf einem Nutzer nicht möglich sein, eine andere Identität anzunehmen.
- Verbindlichkeit: Der Versand muss nachvollzogen werden können (Protokollierung).

Weder SOAP noch WSDL bieten Mittel um diese Anforderungen zu erfüllen. Daher wurde der Standard WS-Security (OASIS (2006)) geschaffen. Dieser Standard schafft

einen Rahmen um vorhandene bzw. zukünftige Ansätze zur Steigerung der Sicherheit in SOAP-Nachrichten einzuführen.

Schon existierende Möglichkeiten, die sich direkt auf die Nachrichten beziehen und sich dort einbinden lassen, sind u. a. die digitale Signatur und die XML-Verschlüsselung. Weitere Möglichkeiten auf der Transportebene sind eine Grundverschlüsselung der Datenkommunikation mit Secure Socket Layer (SSL) bzw. Transport Socket Layer (TSL).

Management

Wird in den Institutionen der öffentlichen Verwaltung eine SOA mittels Web Services eingeführt, entstehen eine Vielzahl von einzelnen Services, die effizient verwaltet werden müssen. Hier gibt es bisher noch keine ausgereiften standardisierten Lösungen. Ansätze bieten die „OASIS Web Services Distributed Management 1.1“ (WSDM) Spezifikation (OASIS (2008b)) sowie die „WS-Management“ Spezifikation (DMTF (2008)).

Das eigentliche Service Management umfasst u. a. folgende Funktionen:

- Überwachung der Servicequalität und von Leistungsmerkmalen
  Um dies zu ermöglichen müssen geeignete Qualitätskriterien erstellt und entsprechend gemessen werden.

- Service Level Agreement (SLA) Management
  Für eine Institution muss das Verhalten eines Services nachvollziehbar und kontrollierbar sein. Dazu werden mit dem Service Provider im Vorfeld bestimmte SLA abgeschlossen. Verstößt der Provider gegen diese, kann es zu Vertragsstrafen kommen. Das Service Management sorgt für die Einhaltung und Kontrolle der Leistungsparameter.

- Unterstützung der Verrechung genutzter Services
  Web Services verursachen Kosten in der Bereitstellung und in der Nutzung. Daher müssen diese Kosten zumindest intern für das Controlling auf bestimmte Kostenstellen umgelegt werden. Eine hohe Bedeutung kommt dieser Funktion zu, wenn die Funktionalität von den Nutzern bezahlt werden muss.

- Fehlerzustände behandeln und Hochverfügbarkeit sicherstellen
  Wenn in einer Institution der öffentlichen Verwaltung Web Services im produktiven Betrieb eingesetzt werden, muss eine nahezu hundertprozentige Verfügbarkeit gewährleistet werden. Dazu gibt es u. a. Failover-Mechanismen, die ei-

nen automatischen Wechsel eines oder mehrerer Services bei einem Ausfall von Services bewirken.

- Versionskontrolle (change management)
  Da Services im Laufe der Zeit ersetzt, oder aktualisiert werden, müssen Abhängigkeiten und notwendige Versionen von Services genau festgehalten und gepflegt werden. Erst wenn kein anderer Service mehr auf den zu ersetzenden Dienst zugreift, kann dieser deaktiviert werden.

Business Processes

Wie schon beschrieben, sind Business Processes (Geschäftsprozesse) meist über einen längeren Zeitraum laufende Transaktionen zwischen mehreren Services. Die Serviceorientierung bietet den Vorteil, dass bei sich ändernden Verwaltungsprozessen, die Services die diese unterstützen, neu kombiniert werden können. Es wird dann von einer Komposition gesprochen, welche die Aspekte der Choreographie und der Orchestrierung beinhaltet (vgl. Dostal u. a. (2005), S. 202).

Die Choreographie beschreibt, „[…] wie mehrere Prozesse miteinander kooperieren“ (Dostal u. a. (2005), S. 203). Um dies zu ermöglichen, muss das Zusammenspiel mehrer Verwaltungsprozesse aus einer globalen Sicht heraus modelliert und beschrieben werden. Dies ermöglicht die „Web Services Choreography Description Language (WS-CDL)“ (vgl. W3C (2007d)).

„Eine Orchestrierung beschreibt die ausführbaren Aspekte eines Geschäftsprozesses [in dieser Arbeit speziell eines Verwaltungsprozesses] aus der Sicht eines Prozesses“ (Dostal u. a. (2005), S. 202). Dies stellt die klassische Workflow-Sicht dar. Es wird dabei ein Führungsprozess (der Dienstprozess) erstellt, der alle Aktivitäten der genutzten Verwaltungsprozesse steuert. Dabei werden die Bedingungen, unter denen der Prozess den Ablauf steuert, erfasst. Zusätzlich werden die Reihenfolge der aufzurufenden Web Services sowie die auszutauschenden Nachrichten beschrieben.

Für die Modellierung der Orchestrierung bzw. Choreographie werden Geschäftsprozessmodellierungssprachen genutzt. Eine Möglichkeit ist die Nutzung der Modellierungssprache WS-BPEL.

3.2 Business Process Execution Language (WS-BPEL)

BPEL ist eine Modellierungssprache, welche das Verhalten von Geschäftsprozessen auf der Grundlage der Interaktionen zwischen dem zu erstellenden Dienstprozess und dessen Partner beschreibt. Dabei wurde die Spezifikation mehrfach erweitert und existiert nun in der Version 2.0. In den vorherigen Versionen wurde die Abkürzung BPEL4WS verwendet. Die Bezeichnung wurde angepasst, um eine einheitliche Bezeichnung der verschiedenen Web Service Standards zu erreichen.

Der BPEL Programmcode wird selber in einem XML-Dokument abgelegt. Dabei wird im Allgemeinen das XML-Schema der Komitee-Spezifikation (siehe OASIS (2008c)) verwendet. Als Ausführungskomponente eines BPEL-Dokuments kommen Prozess-Manager oder Workflow-Engines zum Einsatz, welche die entsprechende Spezifikation interpretieren können. Die Grundstruktur gliedert sich wie folgt:

```
<process name="ProcessName">
     <partnerLinks>     …  </partnerLinks>
     <variables>        …  </variables>
     <correlationSets>  …  </correlationSets>
     <faultHandlers>    …  </faultHandlers>
     <eventHandlers>    …  </eventHandlers>
     <! -- auszuführende Aktivitäten -->
</process>
```

Für die nachfolgenden Codelistings bedeutet das Zeichen „+“, dass dieses Element mehrfach verwendet werden kann. Ein „?“ kennzeichnet dagegen ein optionales Element. Im Folgenden werden die Sprachelemente kurz erläutert. Als zusätzliche Literatur wird vor allem die BPEL-Spezifikation (OASIS (2007c)) empfohlen.

3.2.1 Kommunikationsbeziehungen

Mittels BPEL wird das Zusammenwirken von mehreren bestehenden Diensten beschrieben. Für die Kommunikation zwischen diesen und dem zu erzeugenden Anwendungsdienst wird das Sprachelement <partnerLink> verwendet.

Für die Verwendung müssen in den WSDL- oder einem Wrapper-Dokumenten[5], der zu nutzenden Dienste, das Element <partnerLinkType> vorhanden sein. Der Syntax sieht wie folgt aus:

```
<wsdl:definitions name="NCName" targetNamespace="anyURI">
     ...
```

[5] Importiert das bisherige WSDL-Dokument und erweitert dies um die notwendigen Sprachelemente

```
     <plnk:partnerLinkType name="NCName">
          <plnk:role name="NCName" portType="QName" />
          <plnk:role name="NCName" portType="QName" />?
     </plnk:partnerLinkType>
     ...
</wsdl:definitions>
```

Mit diesen Codezeilen wird jeder Rolle genau ein vorhandener „PortType“ zugeordnet. Damit wird ein Zusammenhang zwischen den jeweiligen ausführbaren Funktionen und dem jeweiligen Dienstnutzer hergestellt.

Soll nun der neue Anwendungsdienst auf diesen Dienst zugreifen, wird folgender Code im BPEL-Dokument verwendet:

```
<partnerLinks>
     <partnerLink name="NCName"
          partnerLinkType="QName"
          myRole="NCName"?
          partnerRole="NCName"?
          initializePartnerRole="yes|no"? />+
</partnerLinks>
```

Hiermit wird die Verbindung zwischen den beiden Diensten hergestellt. Wichtig sind hier die Parameter myRole und partnerRole. Es muss mindestens eine von beiden angegeben werden. Mittels initializePartnerRole wird festgelegt, ob die Verbindung vor der ersten Nutzung initialisiert (hergestellt) werden soll. Wenn die Verknüpfung besteht, können die bereitgestellten Funktionen des Partnerdienstes genutzt werden.

3.2.2 Nutzung von Variablen

Bevor auf die einzelnen Aktivitäten eingegangen wird, muss die Verwendung bzw. Manipulation von Informationen, welche in den Variablen abgelegt sind, erläutert werden. Im BPEL-Dokument werden Variablen für das Ablegen von eingehenden bzw. ausgehenden Informationen benötigt. Zusätzlich müssen oft auch innerhalb des Dienstes Informationen vorgehalten werden, die nur während der Ausführung benötigt werden.

Variablen werden im BPEL-Sprachelement <variables> abgelegt. Eine Variablendefinition ist wie folgt aufgebaut:

```
<variables>
     <variable name="BPELVariableName"
          messageType="QName"?
          type="QName"?
          element="QName"?>
```

```
    </variable>+
</variables>
```

Dabei wird der Variablenname über das Attribut „name“ zugewiesen. Als Datenformat kann ein SOAP-Nachrichtentyp mittels des „messageType“-Attributs, ein einfacher bzw. komplexer Datentyp mittels des „type“-Attributs oder eine globale Variable („properties“) mittels des „element“-Attributs verwendet werden. Die jeweils verwendeten Daten- bzw. Nachrichtentypen müssen entsprechend importiert oder lokal definiert werden.

3.2.3 Correlations

Führt ein über BPEL generierter Dienst eine Dienstleistung aus, welche über mehrere verschiedene Dienste abgewickelt wird, ist es oft erforderlich, dass eine eindeutige Zuordnung der auszutauschenden Informationen zur jeweiligen ausgeführten Instanz eines Dienstes erfolgen kann. Dies wird bei den zustandslosen Web Services über entsprechende Correlations ermöglicht. Diese nutzen die bereits erwähnten „properties“. Sie stellen von der Warteschlange („query“) des jeweiligen Web Services auslesbare globale Variablen dar, die sich wenn sie einmal für einen Vorgang belegt wurden nicht verändert werden dürfen, um eine eindeutige Zuordnung der jeweiligen Instanz eines Dienstes zu möglich. Ähnlich der Rollenzuordnung mittels des <partnerLinkType>-Operators, müssen auch in diesem Fall die „properties“ in der WSDL-Datei des aufzurufenden Dienstes oder einem zusätzlichen Wrapper-Dokument definiert werden.

Der BPEL-Code sieht folgendermaßen aus:

```
<correlations>
    <correlation set="NCName"
    initiate="yes|join|no"?
    pattern="request|response|request-response"? />+
</correlations>
```

Mittels des Attributes „initiate“ wird das Verhalten des aufzurufenden Dienstes in Bezug auf die „properties“ festgelegt:

- yes: ist die Variable bereits belegt, wird eine Fehlermeldung zurückgegeben.
- join: ist die Variable bereits belegt, wird diese verwendet. Andernfalls wird die Variable vom Dienst selber belegt.
- No: ist die Variable noch nicht belegt, wird eine Fehlermeldung zurückgegeben.

Mit steigender Anzahl an Interaktionen nimmt diese Abstimmung stark an Komplexität zu. Hier sollte das Entwicklungswerkzeug unterstützen.

3.2.4 Fehlerbehandlung

Wie schon in Kapitel 3.1 erläutert, handelt es sich bei Web Services und damit auch bei mittels BPEL erzeugten Web Services oft um lang laufende Transaktionen. Da BPEL grundsätzlich dazu ausgelegt ist mehrere Services zu einem neuen zusammenzufassen, muss eine erweiterte Fehlerbehandlung ermöglicht werden.

Dazu wird der <compensationHandler>-Operator eingesetzt, welcher direkt als Operand innerhalb des <invoke> und des <scope> Operators verwendet werden kann. Im Falle eines Fehlers wird der hier hinterlegte Code ausgeführt um die innerhalb des Operators durchgeführten Aktivitäten zu kompensieren.

```
<compensationHandler>
     activity+
</compensationHandler>
```

Innerhalb des durch BPEL erzeugten Prozesses (<process>) wird der <faultHandlers> Operator verwendet um die entsprechenden „compensationHandler“ aufzurufen. Dabei werden sowohl abgeschlossene als auch noch laufende <invoke> oder <scope> Operationen angesprochen.

```
<faultHandlers>
     <catch faultName="QName"?
     faultVariable="BPELVariableName"?
     (faultMessageType="QName" oder faultElement="QName")?>
          activity+
     </catch>
     <catchAll>?
          activity+
     </catchAll>
</faultHandlers>
```

Dabei können einzelne Fehler gezielt behandelt werden (<catch>) oder auch alle Fehlermeldungen gesammelt (<catchAll>) verarbeitet werden. Als Operatoren werden u. a. <compensate> und <compensateScope> eingesetzt, um entsprechende externe Aktivitäten zu kompensieren.

3.2.5 Aktivitäten

Die Modellierungssprache BPEL umfasst die folgenden Sprachelemente zur Ausführung von Aktivitäten:

<receive>

Mittels dieser Aktivität ist es möglich, die weitere Ausführung des Web Services so lange zu unterbrechen, bis eine erwartete Nachricht eintrifft. Der Syntax ist wie folgt festgelegt:

```
<receive partnerLink="NCName"
     portType="QName"
     operation="NCName"
     variable="BPELVariableName"?
     createInstance="yes|no"?
     <correlations>?
          <correlation set="NCName" initiate="yes|join|no"?
          />+
     </correlations>
</receive>
```

Die Attribute „partnerLink“ und „portType“ wurden bereits im Kapitel 3.2.1 erläutert. Mittels „operation“ wird die gewünschte Funktion ausgewählt und „variable“ bestimmt den zu erwartenden Nachrichtentyp. “CreateInstance“ definiert, ob bei Eingang der festgelegten Nachricht eine neue Instanz des Web Services gestartet werden soll. In diesem Zusammenhang sind die im vorherigen Teilkapitel erläuterten Correlations von großer Bedeutung. Der *<receive>*-Operator kann vor allem als Startelement eines BPEL-Dienstes genutzt werden.

<reply>

Das Gegenstück zum *<receive>* bietet dieser Operator. Damit auf eine empfangene Nachricht geantwortet werden kann, kann das Sprachelement <reply> verwendet werden.

```
<reply partnerLink="NCName"
     portType="QName"?
     operation="NCName"
     variable="BPELVariableName"?
     faultName="QName"?
     <correlations>?
```

```
          <correlation set="NCName" initiate="yes|join|no"?
          />+
     </correlations>
</reply>
```

Das zusätzliche Attribut „faultName“ ermöglicht die Festlegung des Fehlernamens und erlaubt damit eine gezieltere Fehlerbehandlung.

<invoke>

Dieses Sprachelement ermöglicht das einfache Versenden einer Nachricht. Dabei ist nur die Angabe von „inputVariable“ notwendig. Sie beinhaltet den zu sendenden Nachrichtentyp. Zusätzlich ermöglicht <invoke> die Zwei-Wege-Kommunikation. In diesem Fall wird die Operation erst beendet, wenn eine Antwortnachricht empfangen wurde. Dabei ist die Angabe der Attribute „inputVariable“ und „outputVariable“ notwendig.

```
<invoke partnerLink="NCName"
     portType="QName"?
     operation="NCName"
     inputVariable="BPELVariableName"?
     outputVariable="BPELVariableName"?>
     <correlations>?
          <correlation set="NCName" initiate="yes|join|no"?
          pattern="request|response|request-response"? />+
     </correlations>
     <catch faultName="QName"?
          faultVariable="BPELVariableName"?
          faultMessageType="QName"?
          faultElement="QName"?>*
          activity
     </catch>
     <catchAll>?
          activity+
     </catchAll>
     <compensationHandler>?
          activity+
     </compensationHandler>
</invoke>
```

Neben den schon erläuterten “Correlations” besteht zusätzlich die Möglichkeit der erweiterten Fehlerbehandlung durch die Auswertung der zurückgelieferten Nachricht.

<assign>

In den meisten Fällen müssen Informationen, die von einem Dienst abgerufen wurden, an einen anderen Dienst weitergesendet werden. Dabei kommen allerdings andere Nachrichtenformate zum Einsatz. Daher müssen die Informationen, die in den Variablen abgelegt werden, untereinander kopiert bzw. verändert werden können.

```
<assign validate="yes|no"?>
     (
     <copy keepSrcElementName="yes|no"? ignoreMissingFrom-
     Data="yes|no"?>
          <from> ... </from>
          <to> ... </to>
     </copy>
     )+
</assign>
```

Das Attribut „validate“ gibt an, ob eine Validierung in Bezug auf das XML-Schema der veränderten Variablen erfolgen soll. Standardmäßig ist diese Option deaktiviert. Mittels „keepSrcElementName“ kann festgelegt werden, dass die Zielvariable in den Namen der Quellvariable umbenannt wird. „ignoreMissingFromData“ ermöglicht die Deaktivierung einer Fehlermeldung, wenn die Quellvariable keinen Inhalt besitzt. Für die Festlegung der Quell- (<from>) und der Zielinformationen (<to>) können unterschiedliche Operanden verwendet werden. Die Wahl hängt davon ab, ob eine Variable, eine „property“ oder gar eine komplette Nachricht in eine andere kopiert oder auch umgeformt werden soll. Auf eine Darstellung der einzelnen Fälle wird hier verzichtet. Für weitere Erläuterungen sei auf (OASIS (2007c), S. 59 ff.) verwiesen.

<exit>

Der <exit>-Operator kann genutzt werden, um eine Ausführungsinstanz eines Web Services ohne Fehler- oder Abschlussbehandlung sofort zu beenden.

```
<exit />
```

<wait>

Oft ist es notwendig, auf ein bestimmtes Ereignis oder bis zu einem bestimmten Zeitpunkt zu warten. Dies wird durch die folgenden Codezeilen ermöglicht:

```
<wait>
<for expressionLanguage="anyURI"?>duration-expr</for>
```

```
</wait> bzw.

<wait>
<until expressionLanguage="anyURI"?>deadline-expr</until>
</wait>
```

Das Attribut „expressionLanguage“ bestimmt dabei die zu verwendende Sprache für die nachfolgenden Ausdrücke innerhalb des Operators.

<empty>

In manchen Fällen ist es erforderlich, dass der Web Service in bestimmten Fällen keine weiteren Funktionen ausführt. Dies ist etwa der Fall um Fehlermeldungen zu vermeiden. Dazu wird ein Entscheidungsoperator genutzt, welcher die Fehler verursachenden Fälle abfragt und dann den <empty>-Operator ausführen lässt.

```
<empty />
```

<sequence>

Für die logische Kapselung einer bestimmten Ausführungsreihenfolge von Operatoren kann der <sequence>-Operator verwendet werden. Damit können insbesondere in parallel laufenden Aktionen oder Schleifen bestimmte Operationen gebündelt werden. Erst wenn die Sequenz vollständig abgearbeitet wurde, wird mit der nächsten Aktivität fortgesetzt.

```
<sequence>
     activity+
</sequence>
```

<if>

Mittels des <if>-Operators kann eine Entscheidung über den weiteren Ablauf des Dienstes getroffen werden. Dazu wird die erste gültige Bedingung ausgeführt. Sollte keine Übereinstimmung gefunden werden, wird der <else>-Bereich (insofern vorhanden) ausgeführt oder es wird sofort mit der nächsten Aktivität fortgesetzt.

```
<if>
     <condition expressionLanguage="anyURI"?> bool-expr
     </condition>
          activity+
```

```
     <elseif>?
     <condition expressionLanguage="anyURI"?>bool-expr
     </condition>
          activity+
     </elseif>
     <else>?
          activity+
     </else>
</if>
```

<while>

Mittels dieses Operators kann eine Schleife erzeugt werden, die solange durchlaufen wird bis die angegebene Bedingung erfüllt ist. Sollte die Bedingung schon bei der ersten Prüfung erfüllt sein, werden die enthalten Aktivitäten gar nicht ausgeführt.

```
<while>
     <condition expressionLanguage="anyURI"?>bool-expr
     </condition>
     activity+
</while>
```

<repeatUntil>

Ähnlich wie bei <while> werden die eingebundenen Aktivitäten solange ausgeführt bis die Bedingung erfüllt ist. Der Unterschied besteht allerdings darin, dass in diesem Fall die Aktivitäten mindestens einmal ausgeführt werden.

```
<repeatUntil>
     activity+
     <condition expressionLanguage="anyURI"?>bool-expr
     </condition>
</repeatUntil>
```

<forEach>

Der <forEach>-Operator verhält sich ähnlich einer Zählschleife in der objektorientierten Softwareentwicklung. Es wird eine Zählvariable („counterName“) festgelegt. Anschließend wird ein Start- (<startCounterValue>) und ein Zielwert (<finalCounterValue>) definiert. Mittels des <completionCondition>-Attributs kann es ermöglicht werden, dass mindestens x (erfolgreiche) Aufrufe erfolgen müssen. Ist diese Zahl erreicht, erfolgen

keine weiteren und die bestehenden werden beendet. Wird der Operator nicht verwendet, wird standardmäßig die komplette Anzahl durchlaufen.

```
<forEach counterName="BPELVariableName" parallel="yes| no">
     <startCounterValue expressionLanguage="anyURI"?>
          unsigned-integer-expression
     </startCounterValue>
     <finalCounterValue expressionLanguage="anyURI"?>
          unsigned-integer-expression
     </finalCounterValue>
     <completionCondition>?
          <branches expressionLanguage="anyURI"?
               successfulBranchesOnly="yes|no"?>?
               unsigned-integer-expression
          </branches>
     </completionCondition>
     <scope> ... </scope>
</forEach>
```

<pick>

Ist es notwendig auf den Eingang einer bestimmten Nachricht (<onMessage>) und / oder das Eintreten eines bestimmten Ereignisses (<onAlarm>) von mehreren zu warten, kann der <pick>-Operator verwendet werden. Tritt einer dieser Fälle ein, werden die eingeschlossenen Aktivitäten ausgeführt. Ein weiteres Eintreffen einer Nachricht oder eines Ereignisses wird von diesem <pick> nicht mehr ausgeführt (Ausnahme in einer Schleife).

```
<pick createInstance="yes|no"?>
     <onMessage partnerLink="NCName"
     portType="QName"?
     operation="NCName"
     variable="BPELVariableName"?
     messageExchange="NCName"?>+
          <correlations>?
               <correlation set="NCName" initiate=
               "yes|join|no"? />+
          </correlations>
          activity
     </onMessage>
     <onAlarm>*
          (
          <for expressionLanguage="anyURI"?>duration-expr
          </for>
          oder
          <until expressionLanguage="anyURI"?>deadline-expr
          </until>
```

```
          )
          activity
     </onAlarm>
</pick>
```

Bei der Verwendung muss mindest ein <onMessage>-Operator verwendet werden. Soll bei der Ausführung eine neue Instanz des Web Services ausgeführt werden, darf kein <onAlarm>-Operator verwendet werden.

<flow>

BPEL ermöglicht auch die Ausführung paralleler Aktivitäten. Erst wenn alle abgeschlossen wurden, wird mit den nachfolgenden Aktivitäten fortgesetzt.

```
<flow >
     activity+
</flow>
```

<scope>

Mittels dieser Aktivität wird ein Container erzeugt, der die darin enthaltenen Codeelemente kapselt und nicht nach Außen sichtbar macht. Es können hier folgende Elemente genutzt und neu erzeugt werden: „variables“, „partner links“, „correlations“, „event handlers“, „fault handlers“, „compensation handler“ und „termination handler“. Auch hier stehen alle erläuterten Aktivitäten zur Verfügung.

```
<scope>
     <faultHandlers>...</faultHandlers>
     activity+
</scope>
```

<validate>

Mittels des <validate>-Operators kann eine Variablen- oder Nachrichtenausprägung auf die Übereinstimmung mit der in der XML- bzw. WSDL-Definition hinterlegten Beschreibung geprüft werden.

```
<validate variables="BPELVariableNames" />
```

<extensionActivity>

Die Modellierungssprache WS-BPEL lässt eine Erweiterung der vorhandenen Aktivitäten zu. Dafür dient der <extensionActivity>-Operator. Dabei ist es wichtig, dass ein anderer XML-Namensraum verwendet wird. Wird nicht explizit mittels des Attributs „mustUnderstand“ verlangt, dass der BPEL-Prozessor diese Aktivität ausführen kann, wird diese als ein <empty>-Operator interpretiert und die folgenden Aktivitäten damit ignoriert, sollte der Prozessor diese nicht verstehen. Dies wird in dieser Arbeit allerdings nicht weiter behandelt.

```
<extensionActivity>
     <anyElementQName standard-attributes>
          standard-elements
     </anyElementQName>
</extensionActivity>
```

<throw>

Mittels <throw> kann eine Fehlermeldung gesendet werden. Dabei muss der Fehlername angegeben werden. Zusätzlich besteht die Möglichkeit eine Variable oder eine Nachricht weiterzugeben um den Fehler genauer eingrenzen zu können.

```
<throw faultName="QName" faultVariable="BPELVariableName"?
/>
```

<rethrow>

Mit dem <rethrow>-Operator kann eine von einem anderen Web Service empfangene Fehlermeldung wiederum an den Aufrufer dieses BPEL-Dienstes weitergeleitet werden. Der Syntax lautet:

```
<rethrow />
```

4 Der erweiterte Anwendungsdienst

Nachdem die Grundlagen der in dieser Arbeit verwendeten Technologien erläutert wurden, wird sich nun der Themenstellung: „die nutzerorientierte Anwendungsdienstentwicklung“ angenähert. Dazu wird zunächst eine einheitliche Dienstbegriffsdefinition gesucht, um dann den erweiterten Dienstbegriff einzuführen und daraus wiederum den erweiterten Anwendungsdienst abzuleiten.

4.1 Begriffsdefinition „Dienst“

Es ist festzustellen, dass keine einheitliche Definition für den Begriff „Dienst“ existiert. Zwei wesentliche Bereiche für die Herkunft werden im Folgenden näher betrachtet:

Wirtschaftswissenschaften

In den Wirtschaftswissenschaften und im allgemeinen Sprachgebrauch bezeichnet ein Dienst eine **Dienstleistung** und erhält einen wirtschaftlichen Charakter. „Dienstleistungen [Hervorhebung i. Original] sind selbständige, marktfähige Leistungen, die mit der Bereitstellung (z. B. Versicherungsleistungen) und/oder dem Einsatz von Leistungsfähigkeiten (z. B. Frisörleistungen) verbunden sind (Potenzialorientierung [Hervorhebung i. Original]). Interne (z. B. Geschäftsräume, Personal, Ausstattung) und externe Faktoren (also solche, die nicht im Einflussbereich des Dienstleisters liegen) werden im Rahmen des Erstellungsprozesses kombiniert (Prozessorientierung [Hervorhebung i. Original]). Die Faktorenkombination des Dienstleistungsanbieters wird mit dem Ziel eingesetzt, an den externen Faktoren, an Menschen (z. B. Kunden) und deren Objekten (z. B. Auto des Kunden) nutzenstiftende Wirkungen (z. B. Inspektion beim Auto) zu erzielen (Ergebnisorientierung [Hervorhebung i. Original]).“ (Meffert, Bruhn (2006), S. 33) Dabei gibt es u. a. Kritik an der Ergebnisorientierung. Es wird davon ausgegangen, dass das Ergebnis eines Dienstleistungsprozesses als Dienstleistung definiert wird, da nur diese am Markt angeboten werden kann.

Zusätzlich wird mit einem Dienst die **Arbeitstätigkeit** im Ausgleich für ein in der Regel festes monatliches Gehalt bezeichnet. Dabei wird ein Dienstvertrag[6] geschlossen, der die Rechte und Pflichten des Angestellten während der Dienstzeit beschreibt. Es wird dabei deutlich, dass es sich bei einem Dienst um ein Ausführungsangebot von verschiedenen Funktionen (Leistungen) handelt.

[6] der Dienstvertrag ist in den §§611 ff. Bürgerliches Gesetzbuch (BGB)

Informationstechnik

Im Fachgebiet der Informationstechnik existieren wiederum unterschiedliche Definitionen für einen Dienst. Eine Definition stammt aus dem **OSI-Referenzmodell**. Dazu wird dieses zunächst eingeführt, um den Begriff des Dienstes aus diesem Betrachtungsfokus zu erläutern.

Das bereits 1984 vorgestellte Referenzmodell bildet die Grundlage für viele aktuelle Kommunikations- und Telekooperationssysteme und besteht aus sieben Schichten:

1. Bitübertragungsschicht (engl. physical layer),
2. Sicherungsschicht (engl. data link layer),
3. Vermittlungsschicht (engl. network layer),
4. Transportschicht (engl. transport layer),
5. Kommunikationssteuerungsschicht (engl. session layer),
6. Darstellungsschicht (engl. presentation layer),
7. Anwendungsschicht (engl. application layer).

Die **Bitübertragungs-, die Sicherungs- und die Vermittlungsschicht** dienen der Realisierung der physikalischen Übertragung und werden in dieser Arbeit nicht näher betrachtet. Sie bieten in der Regel keine Fehlererkennungs- und Fehlerbehebungsfunktionen an. Dafür ist die darüberliegende **Transportschicht** verantwortlich. Bis zu dieser Schicht findet nur eine vollduplexe Datenübertragung statt.

Die **Kommunikationssteuerungsschicht** vermittelt zwischen den unteren Datenübertragungsschichten und den höheren anwendungsorientierten Schichten. Sie stellt Verwaltungs- und Steuerungsdienste bereit und bietet damit einer Anwendung bzw. einem Verwaltungssystem die Möglichkeit ihren Datenaustausch zu ordnen und zu verwalten. Angebotene Dienste dieser Schicht sind vor allem das Starten und Anhalten, das Verlassen und wieder in Gang bringen von Kommunikationsprozessen. Zusätzlich bietet sie Synchronisationspunkte für eine eventuelle Unterbrechung einer Übertragung, um im Fehlerfall an diesen Punkten fortsetzen zu können.

Es existieren unterschiedliche Computersysteme mit unterschiedlichen Datenformaten und -codierungen. Damit ein Informationsaustausch dennoch stattfinden kann, ist die **Darstellungsschicht** dafür verantwortlich, dass die auszutauschenden Anwendungsinformationen in einem für beide Seiten verständlichen und einheitlichen Format übertra-

gen werden. Beispiele dafür sind ein einheitlicher Zeichensatz oder der XML-Nachrichtenstandard.

Die sieben Schichten können als eine Art Stapel (engl. „Stack“) visualisiert werden. Die **Anwendungsschicht** stellt die oberste Schicht und insofern einen Sonderfall dar, dass sie die Dienste des „Stack“ für einen Nutzer (Mensch oder Maschine) nach außen hin zur Verfügung stellt. Alle anderen Dienste der verschiedenen Schichten werden durch die nächsthöhere Schicht genutzt und sind nicht extern zugänglich.

Auf der Anwendungsschicht muss die nutzerorientierte Systementwicklung ansetzen, da diese die direkte Schnittstelle zu den Nutzern darstellt. Dabei darf, abweichend von der bisherigen Systementwicklungspraxis, die Nutzerklasse nicht als eine homogene Menge gesehen werden. Es gibt eine Vielzahl an Nutzerklassen (siehe Kapitel 2.3) mit unterschiedlichem Wissens- und Erfahrungsstand.

Durch die Nutzung der verschiedenen Schichten können einzelne ausgetauscht und somit verschiedene Hard- und Softwaresysteme beliebig genutzt werden. Da durch die Vielzahl an Schichten auch eine Vielzahl an Schnittstellen und damit Implementierungs- und in der Ausführung Rechenaufwand entsteht, wird das OSI Referenzmodell oft kritisiert. Aus diesem Grund werden in vielen neuen Standards nicht alle Schichten einzeln besetzt, sondern es werden mehrere zusammengefasst.

Ein **Dienst** wird nach dem Referenzmodell als eine Sammlung von Funktionen, den sogenannten Dienstelementen, einer Schicht definiert und ist an die konzeptuelle Ebene gebunden. Er stellt der darüber liegenden Schicht bzw. den Dienstnutzern die Elemente zur Verfügung. Zur Ausführung der Funktionen nutzt dieser wiederum die Funktionen der darunter liegenden Schicht. Dabei kann unter einem Dienst sowohl eine einzelne Funktion einer Schicht verstanden werden oder aber die gesamte nach außen verfügbare Funktionalität des Schichtenmodells.

Die einzelnen Dienstelemente werden durch entsprechende Dienstprimitive über entsprechende „Service Access Points“ aufgerufen. Es existieren vier Dienstprimitive:

- das Anforderungs-Dienstprimitiv (dient zur Aktivierung eines Dienstelements),
- das Anzeige-Dienstprimitiv (resultiert aus der Anforderung und führt eine entsprechende Ausgabe auf der Gegenseite durch),
- das Antwort-Dienstprimitiv sowie das Bestätigungsdienstprimitiv (dienen der Bestätigung der Übertragung und werden nicht zwingend benötigt).

Der Dienstnutzer wird durch einen entsprechenden „Connection Endpoint“ identifiziert. Die Abb. 4.1 verdeutlicht den Zugriff von Nutzern auf einen Dienst.

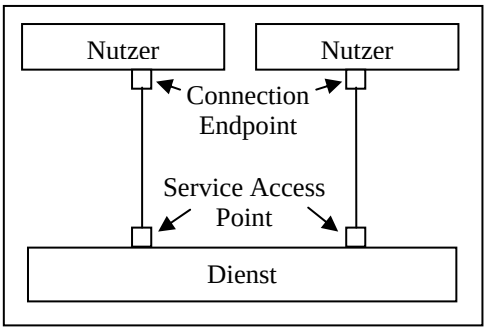


Abb. 4.1: Zugriff mehrerer Dienstnutzer auf einen Dienst

Damit ein Dienst in einer Anwendung bzw. einem Anwendungsprozess genutzt werden kann, muss ein Interface für die Anwendungsschicht dieses Dienstes implementiert werden. Die darunter liegenden Schichten werden durch den Service Provider zusammengefasst und nicht näher betrachtet. In Abb. 4.2 ist die Verknüpfung einer Anwendung mit einem OSI-Dienst grafisch dargestellt.

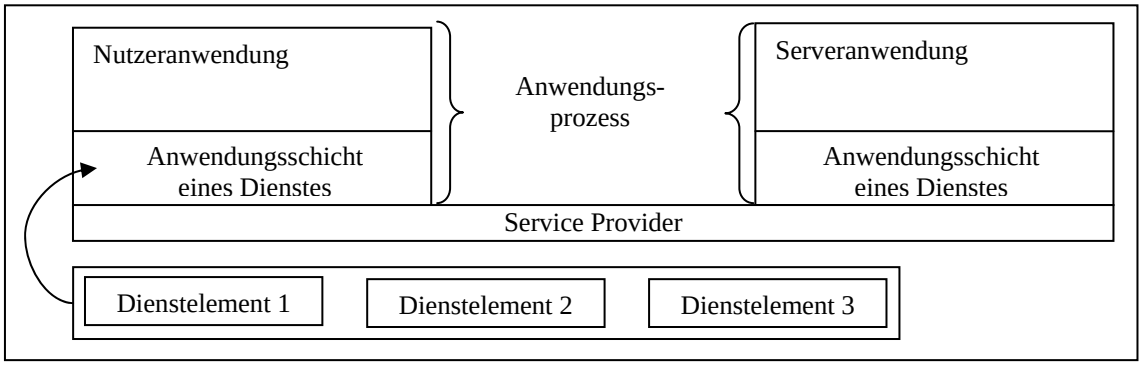


Abb. 4.2: Verflechtung Anwendung – OSI-Schichtenmodell (vereinfacht)

Das OSI-Referenzmodell ist stark auf Telekommunikationsdienste geprägt und der konzeptuellen Entwurfsebene zuzuordnen. Die Dienste sollen den Austausch von Informationen ermöglichen. Zusätzliche Funktionalitäten werden nicht dem Modell zugeordnet, sondern den entsprechenden Anwendungssystemen. Für nähere Erläuterungen zum gesamten Referenzmodell entsprechende Fachliteratur verwiesen.

Für diese Arbeit bedeutet dies, dass die Dienstdefinition nicht vollständig den Modellanforderungen von Lüttich genügt und entsprechend erweitert werden muss. Dennoch bieten das OSI-Referenzmodell sowie dessen Dienstbegriff einen guten Ausgangspunkt für die weitere Themenbearbeitung. Im erweiterten Prozessmodell kann der Informationsaustausch nach diesem Modell wie folgt modelliert werden:

Ein Anwendungsprozess A (AP_A) möchte Informationen an einen Anwendungsprozess B (AP_B) senden. Dazu baut der AP_A eine Verbindung mit dem Service Provider Prozess (SPP) über einen Transportprozess A (TP_A) auf. Dieser wiederum öffnet eine Verbindung zum Zielprozess über einen Transportprozess B (TP_B) (genauer: zu dem darunter liegenden Anwendungssystem). Anschließend findet der Informationsaustausch über den Service Provider (Service Provider Prozess (SPP)) statt. In Abb. 4.3 ist dieser Zusammenhang grafisch dargestellt.

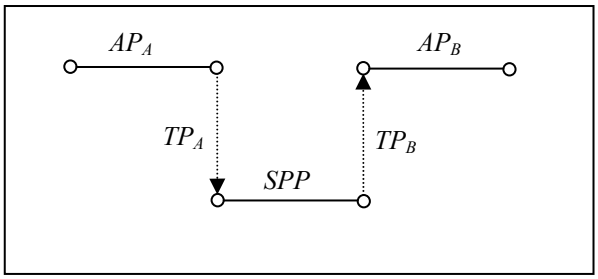


Abb. 4.3: erweiterte Prozessdarstellung des OSI-Dienstes

LÜTTICH selbst verwendet den Begriff des Dienstes in Bezug auf die Verwendung von Fachkomponenten für die nutzerorientierte Systementwicklung als „[…] eine Aktion eines rechnergestützten Verwaltungssystems […], welche die Bewältigung einer bestimmten Menge von Verwaltungsaufgaben unterstützt“ (Lüttich, Turowski (2000), S. 465). Damit stellt ein Dienst, in allen hier erläuterten Definitionen, ein Ausführungsangebot von festgelegten Funktionen dar.

4.2 Erweiterter (Anwendungs-) Dienst aus Nutzersicht

Ein Dienst stellt, wie bereits im vorherigen Teilkapitel erläutert, seine Funktionalität anderen Nutzern und damit auch verschiedenen Nutzerklassen zur Verfügung. Ein Anwendungsdienst aus Nutzersicht stellt eine Kette von verknüpften Verwaltungssystemen dar, deren Verfahren abgestimmt, deren Systeme koppelbar und deren Prozesse verknüpfbar sind. Dabei aggregiert dieser verschiedene Funktionalitäten der Verwaltungssysteme, stellt sie mittels einer Schnittstelle zur Verfügung und ermöglicht damit Dienstleistungen für die einzelnen Nutzerklassen, die diese Systeme alleine nicht leisten können bzw. für die diese nicht vorgesehen sind.

Da der Anwendungsdienst über entsprechende Schnittstellen verfügt, allerdings keine eigene Nutzeroberfläche anbietet, muss im Falle der Nutzerinteraktion ein entsprechendes Frontend in Form eines Anwendungssystems oder eines Portals bereitstehen. In der

Maschine-Maschine Kommunikation wird dieses nicht benötigt, da die entsprechenden auszutauschenden Nachrichten ohne Nutzereingriff erstellt werden.

Um einen Anwendungsdienst aus Nutzersicht entwickeln zu können, wird zunächst das in Kapitel 2 eingeführte erweiterte Prozessmodell auf die Dienste übertragen. Es wird davon ausgegangen, dass ein Dienst aus Nutzersicht mittels der drei Ebenen (Prozess-, System- und Verfahrensebene) beschrieben werden kann: dem Dienstprozess (DP_{ij}), dem Dienstsystem (DS_{ij}) und dem Dienstverfahren (DVf_{ij}). Dabei stellt jede Komponente ein Untersuchungsobjekt dar, welches wiederum aus einer Komponentenbeschreibung und einer Komponentenausprägung besteht. Diese werden im Folgenden näher betrachtet.

Der **Dienstprozess** besteht aus den beiden Zuständen Z_i, Z_j und der Relation zwischen den Zuständen ($rel(Z_i, Z_j)$). Die Dienstsaufgabe (DA_{ij}) enthält die möglichen abzuarbeitenden Tätigkeiten. Durch die Ausführung einer Aufgabe wird das entsprechende Dienstobjekt (DO_i) manipuliert, in das Dienstobjekt (DO_j) überführt und damit das gewünschte Ergebnis erzielt. Hierbei darf das Wort „manipulieren“ allerdings nicht im negativen Sinn interpretiert werden, sondern im Sinne von „verändernd“. Zusätzlich existieren die Beobachtungskomponenten Ort (l_i bzw. l_j), Zeit (t_i bzw. t_j) und das Wissen (W_i bzw. W_j). Es ist möglich auch weitere Komponenten einzuführen, insofern dies notwendig wird (siehe Kapitel 2.1).

Im Endzustand Z_j entfällt die Dienstaufgabe, da diese bearbeitet wurde. Dafür wird hier die Komponente des Dienstverhaltens (DV_{ij}) eingeführt. Dieses enthält neben dem Fehler- und Nachrichtenprotokoll auch das Ablaufprotokoll. Beide haben besondere Bedeutung im E-Governmentbereich, da hier die Transparenz und Nachvollziehbarkeit der Abwicklung von Verwaltungsprozessen gewahrt werden muss.

Ein Dienstprozess (DP_{ij}) setzt sich aus der Vereinigung aller möglichen definierten Dienstgänge ($\bigcup_k DG_{ij}^k$) bzw. deren Ausführung über dem Dienstsystem ($\bigcup_k a_k(DS_{ij})$), zusammen.

$$DP_{ij} = \bigcup_k DG_{ij}^k = \bigcup_k a_k(DS_{ij})$$

Die Relation zwischen den Zuständen Z_i und Z_j bildet das **Dienstsystem.** Es enthält die entsprechenden Zustandsbeschreibungen (die Zustandsausprägungen fallen hier weg) und damit auch die Schnittstellen (SS_i *bzw.* SS_j).

Wiederum unter dem Dienstsystem liegt das **Dienstverfahren**. Das Dienstverfahren enthält die einzelnen Funktionskomplexe und damit die einzelnen Aufrufe der verwen-

deten externen Verwaltungssysteme und Dienste. Nach Lüttich kann hier der Funktionskomplex Null eingesetzt werden. Dieser stellt den entsprechenden Rechtsrahmen bereit und kontrolliert die Abläufe auf Rechtskonformität.

Abb. 4.4 veranschaulicht diese Zusammenhänge noch einmal grafisch. Dabei kann das Dienstsystem auf unterschiedliche externe Verwaltungssysteme (VS_x) oder Dienste (D_{xy}) zugreifen.

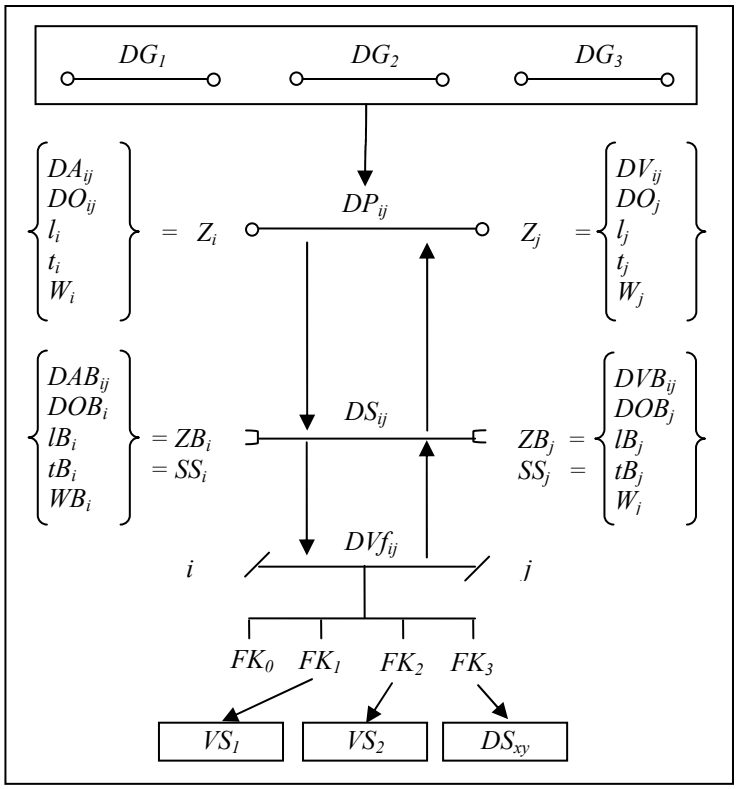


Abb. 4.4: erweiterter Dienst aus Nutzersicht

Wie schon im letzten Teilkapitel erläutert, stellt ein Dienst ein Ausführungsangebot dar. Dieses Angebot wird als ein abstrakter Dienst (aD_{ij}) definiert und enthält lediglich die entsprechende Dienstbeschreibung (DB_{ij}).

$$aD_{ij} = DB_{ij}.$$

Wird ein solcher Dienst genutzt, handelt es sich um einen Nutzerdienst und damit auf der konzeptuellen Ebene um einen **Anwendungsdienst**. Ein Nutzerdienst stellt einen konkreten Dienst (kD_{ij}) dar.

$$kD_{ij} = (DB_{ij}, DA_{ij})$$

Damit ergibt sich der konkrete Dienst aus der Verknüpfung des abstrakten Dienstes mit der konkreten Auftragsausprägung, wobei die Auftragsbeschreibung mit der Aufgabenbeschreibung des abstrakten Dienstes übereinstimmen muss.

$kD_{ij} = aD_{ij} \circ DA_{ij}$

Im Folgenden muss unterschieden werden zwischen dem Dienstentwicklungs- und dem Dienstausführungsprozess.

Dienstentwicklungsprozess

Wird ein Verwaltungssystem aus Nutzersicht entwickelt (siehe Kapitel 2), wird dies als Verwaltungssystementwicklung erster Art bezeichnet. Das dabei entstehende System stellt Schnittstellen bereit, welche entweder als Nutzerschnittstellen direkt vom Nutzer einer Nutzerklasse (NK_{kl}) oder von externen Anwendungen genutzt werden können. Wird nun die Anwendungsdienstentwicklung betrachtet, greift diese im allgemeinen Fall auf mehrere bestehende Verwaltungssysteme und Dienste zu und komponiert die verschiedenen Funktionalitäten zu einer neuen Dienstleistung. Anschließend wird der neue Dienst mittels eines geeigneten Schnittstellenformats den Nutzern einer Nutzerklasse (NK_{kl}) bereitgestellt. Diese Form der Anwendungsdienstentwicklung wird als Verwaltungssystementwicklung zweiter Art bezeichnet. In Abb. 4.5 ist dieser Zusammenhang einmal grafisch dargestellt. Die Werkzeugsysteme werden durch einen klassischen Anwendungssystementwickler der Nutzerklasse NK_5 erstellt. Anschließend können Nutzer der kompetenten Nutzerklasse (NK_3 und NK_4) die entsprechenden Verwaltungssysteme und Anwendungsdienste erstellen. Nachdem diese erstellt wurden, können Menschen der Nutzerklasse NK_1 und NK_2 die jeweiligen Dienstleistungen in Anspruch nehmen.

Zum besseren Verständnis, welche Vorteile die Verknüpfung von verschiedenen Verwaltungssystemen hat, dienen die folgenden Ausführungen: Es wird davon ausgegangen, dass ein Bürger drei Verwaltungsaufträge (VAT_{ij}, VAT_{kl}, VAT_{mn}) stellen will. Bisher war er gezwungen diese jedem einzelnen Verwaltungssystem (VS) selber zu übergeben. Dabei ist es notwendig viele Informationen mehrfach in unterschiedliche Nutzeroberflächen einzugeben. Werden nun die einzelnen Aufträge zu einem Dienstauftrag zusammengefasst, brauchen die Informationen nur einmal eingegeben werden. Diese werden dann implizit an die einzelnen Verwaltungssysteme vom Anwendungsdienst mittels Dienstteilprozessen (DTP_{xy}) weitergegeben.

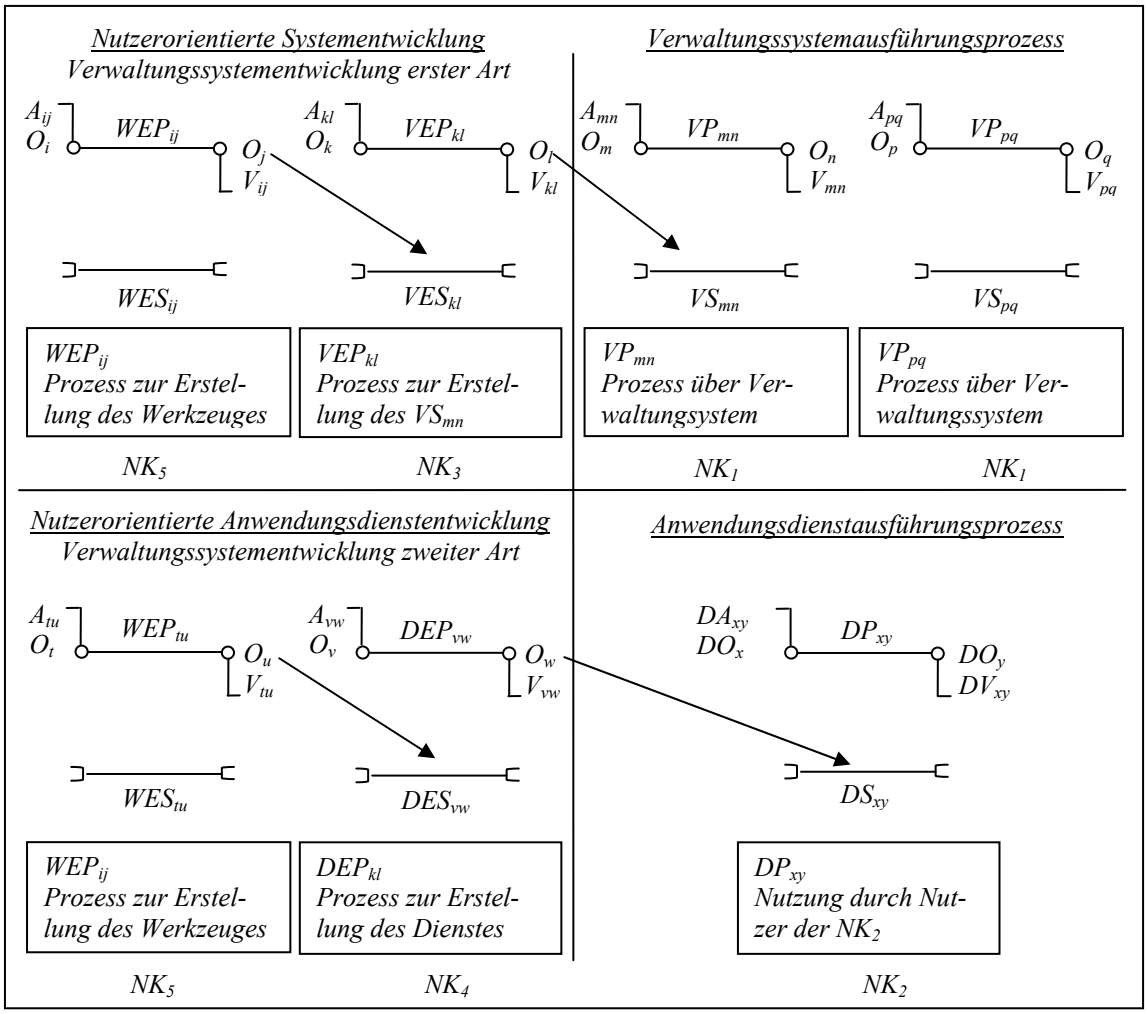


Abb. 4.5: Zusammenhang zwischen der Verwaltungssystem- und der Anwendungsdienstentwicklung

Die Funktionalitäten der in einem Dienst zu koppelnden Teilverwaltungssysteme können in Bezug auf das OSI-Referenzmodell als Dienstprimitive (siehe Kapitel 4.1) betrachtet werden. Die Verknüpfung der einzelnen Dienstteilprozesse ergibt den Dienstprozess (DP_{xy}). In Abb. 4.6 ist dieser Zusammenhang einmal grafisch dargestellt.

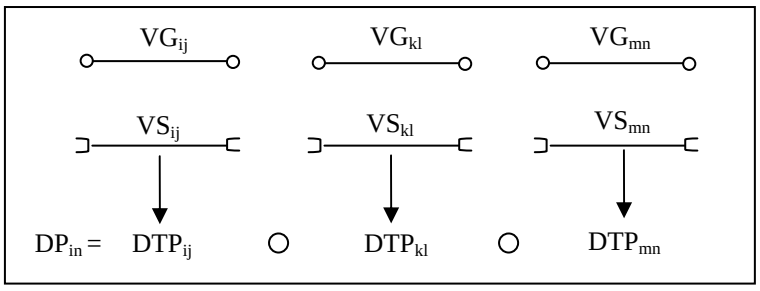


Abb. 4.6: Zusammenhang Dienstprozess - Verwaltungsgang

Damit wird ein Dienstteilprozess mit dem Tripel aus Dienstteilauftrag, Verwaltungsgang und dem entsprechenden Verwaltungssystem beschrieben (AT_{ij}, VG_{ij}, VS_{ij}). Ein Dienstvorgang wird dann als die Verknüpfung der jeweiligen Tripel definiert.

Um einen Dienstgang ausführen zu können, muss dieser zuvor als Kette von Verwaltungsprozessen modelliert und der vollständige Dienst entwickelt worden sein. Für die Dienstentwicklung aus Nutzersicht bietet es sich an, die später zu ermöglichen Prozessgänge einzeln zu modellieren und damit die Entwicklung zu vereinfachen.

Dienstausführungsprozess

Ein besonderes Merkmal der nutzerorientierten Systementwicklung stellt die Einbeziehung der verschiedenen Nutzerklassen dar. So kann ein aus Nutzersicht entwickelter Anwendungsdienst je nach Nutzerklasse verschiedene Dienstprimitive anbieten, welche es ermöglichen, dass Nutzer mit unterschiedlichen Erfahrungen sowohl mit der Technik als auch mit Verwaltungsformalitäten dennoch diese Dienstangebote nutzen können. Dazu sind dann, wie bereits erwähnt, entsprechende Frontend-Systeme für die Nutzer notwendig, da ein Dienst keine visuelle Schnittstelle besitzt. Der Dienstausführungsprozess wird in dieser Arbeit nicht weiter betrachtet.

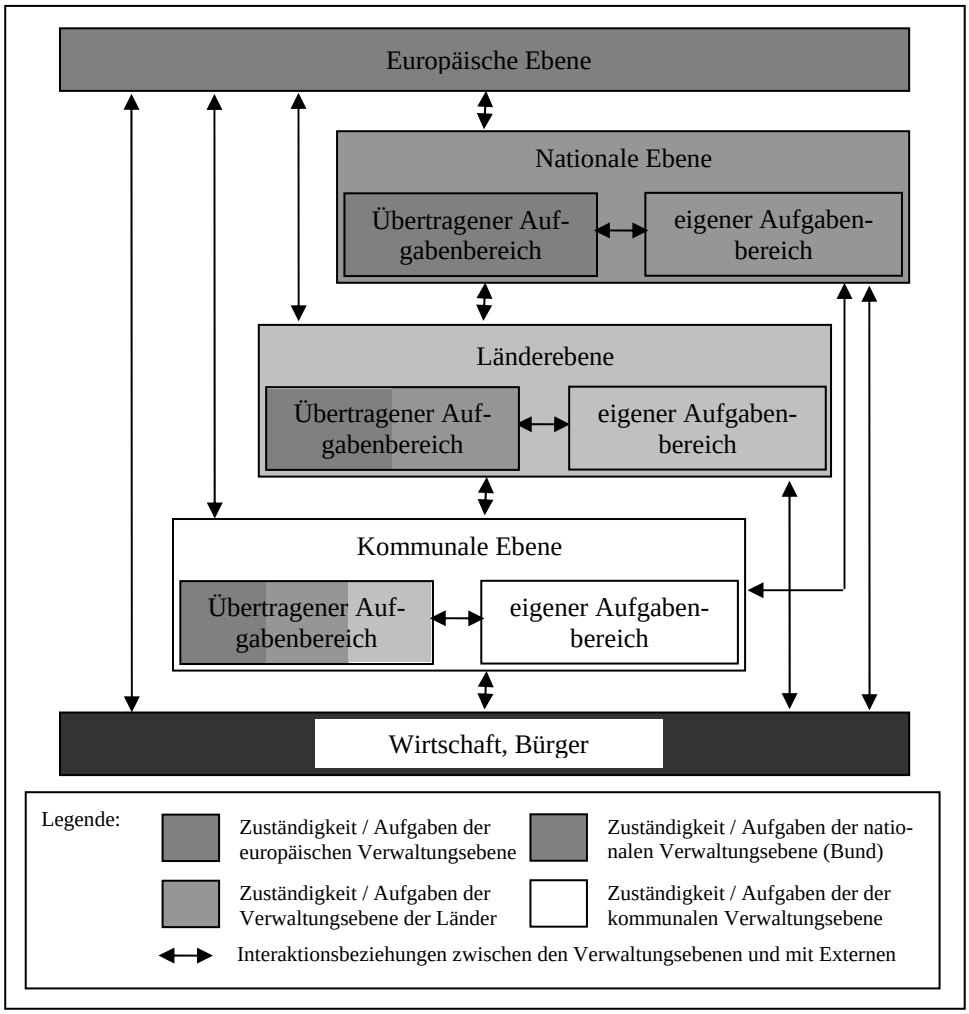
5 Spezifikation der Softwarearchitektur

Nachdem die Grundlagen für eine nutzerorientierte Anwendungsdienstentwicklung erläutert wurden, muss nun eine entsprechende Softwarearchitektur spezifiziert werden, um diese Art der Anwendungsdienstentwicklung zu ermöglichen.

Die Bundesrepublik Deutschland besteht aus drei Verwaltungsebenen: der Bundes-, Landes- und der kommunalen Ebene. Dabei liegt die Verantwortung für Aufgaben, die nicht unbedingt von der Bundesverwaltung zentral bearbeitet werden müssen, bei den Ländern. Diese wiederum können die Verantwortung für eigene oder hoheitliche[7] Aufgaben nach dem Prinzip der Subsidiarität (vgl. Hoffmann-Riem u. a. (2006), S. 960 ff.) auf die kommunale Ebene delegieren. Gleichzeitig besitzen die jeweiligen kommunalen Gebietskörperschaften die folgenden Hoheitsrechte für ihre örtlichen Angelegenheiten: Gebietshoheit, Satzungshoheit, Finanzhoheit, Planungshoheit, Organisationshoheit, Personalhoheit und die Rechtssubjektshoheit (vgl. Art. 28, Abs. 2 sowie Art. 30 Grundgesetz (GG)). Dieses Recht führte dazu „[…], dass viele kommunale Körperschaften selbst Wege und Werkzeuge bei der Erfüllung eigener und teilweise auch übertragener Aufgaben gewählt haben“ (Schneider (2007), S.122). Somit existieren eine Vielzahl von Werkzeugen und Methoden für gleiche oder ähnliche Aufgaben. In Abb. 5.1 wird das Zusammenwirken der verschiedenen Verwaltungsebenen dargestellt. Dabei wird die europäische Ebene als überstaatliche Ebene eingefügt, da diese im Zuge einer europäischen Gemeinschaft zusätzliche Aufgaben wahrnimmt und diese teilweise auf untere Verwaltungsebenen überträgt. Sollen nun im Zuge einer modernen E-Government-Systemarchitektur diese Verwaltungssysteme zusammenarbeiten, entsteht ein hoher Aufwand für die Abstimmung der Fachverfahren, die Kopplung der Verwaltungssysteme und Verknüpfung der Verwaltungsprozesse (siehe Kapitel 2.5).

Das Problem der Vielfalt an Verwaltungs- und Anwendungssystemen in den Institutionen der öffentlichen Verwaltung wurde erkannt. Im Zuge der E-Government Entwicklung in der Bundesrepublik Deutschland wurde der Handlungsleitfaden „Standards und Architekturen für E-Government-Anwendungen (SAGA)“ (BMI (2008a)) herausgegeben, welcher Lösungsmöglichkeiten und Empfehlungen für neue Verwaltungsanwendungen gibt. Die Version 4.0 wurde im März 2008 veröffentlicht und verfolgt die Ziele: Interoperabilität, Wiederverwendbarkeit, Offenheit, Skalierbarkeit, Reduktion von Kosten und Risiken der Verwaltungssysteme.

[7] Hoheitliche Aufgaben sind von einer höheren Ebene verliehene Aufgaben.



Quelle: Schneider (2007), S.126

Abb. 5.1: Aufbau und Zusammenspiel nationaler und europäischer Verwaltungseinheiten

Dazu werden zwei Ansätze verfolgt:

- Festlegung der technischen Normen, Standards und Architekturen für E-Government-Anwendungen,
- Vereinheitlichung von Prozessen und Daten in Verwaltungen.

Weiterhin wird für die Verwaltungssysteme als Nutzer-Frontend der Webbrowser empfohlen. Dabei sollen aktive Inhalte vermieden werden, um auch bei Nutzern einer entsprechenden Nutzerklasse die Ausführbarkeit zu gewährleisten, bei denen hohe Sicherheitseinstellungen auf den Rechnersystemen vorgegeben sind (vgl. BMI (2008a), S. 13).

Die Richtlinie SAGA empfiehlt zur Beschreibung eines E-Government-Systems die Verwendung des Referenzmodells für offene, verteilte Datenverarbeitung (RM-ODP).

Gleichzeitig wurde auch die SAGA-Richtlinie nach dieser Struktur aufgebaut. RM-ODP betrachtet ein System auf fünf Sichtweisen (BMI (2008a), S. 34):

1. Der „Enterprise Viewpoint“ spezifiziert Zweck, Nutzungsbereich und Regeln einer Anwendung.

2. Der „Information Viewpoint“ beschreibt die Struktur und Semantik der zu verarbeitenden Daten, also das Datenmodell.

3. Der “Computational Viewpoint“ stellt die Zerlegung einer Anwendung in funktionale Elemente und deren Interaktionsschnittstellen dar.

4. Der „Engineering Viewpoint“ stellt die Verteilung der einzelnen Elemente des Systems auf physikalische Ressourcen sowie deren Verbindungen dar.

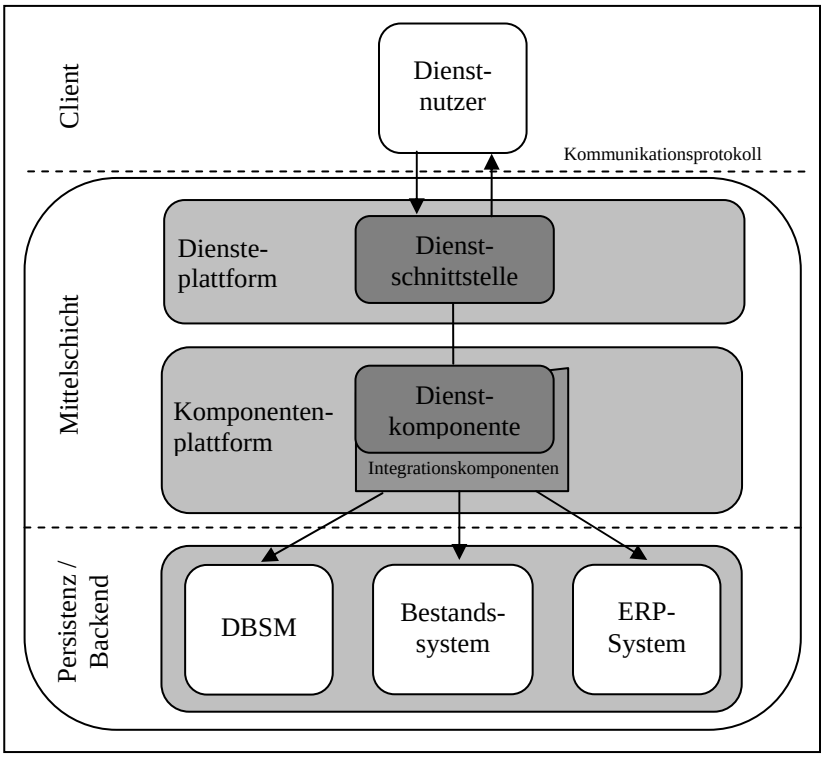
5. Der „Technology Viewpoint“ beschreibt die zur Realisierung des Systems verwendeten Technologien.

Wird sich bei einer neuen Verwaltungssystementwicklung an SAGA orientiert, werden einheitliche Standards verwendet. Diese unterstützen vor allem die Zukunftsfähigkeit und die Interaktionsfähigkeit mit anderen Verwaltungssystemen. Hinzu kommen die Vereinheitlichung von Verwaltungsvorgängen und Daten- sowie Datenaustauschformaten, was zu einer Reduktion der Daten- und Schnittstellenvielfalt führt.

5.1 Serviceorientierte Architektur mittels der Web Service Technologie

In der SAGA-Richtline (Version 4.0) wird die Verwendung einer dienst- bzw. serviceorientierten Architektur (SOA) empfohlen. Bei der Betrachtung eines Dienstes kann von einer Gliederung der Architektur in drei Schichten (siehe Abb. 5.2) gesprochen werden. Es wird in diesem Fall in die Persistenz-, Mittel- und die Clientschicht unterteilt. Zusätzlich wird davon ausgegangen, dass die Präsentation von einem entsprechenden Client-Anwendungssystem übernommen wird und diese somit bei der Erstellung des Dienstes nicht betrachtet werden muss.

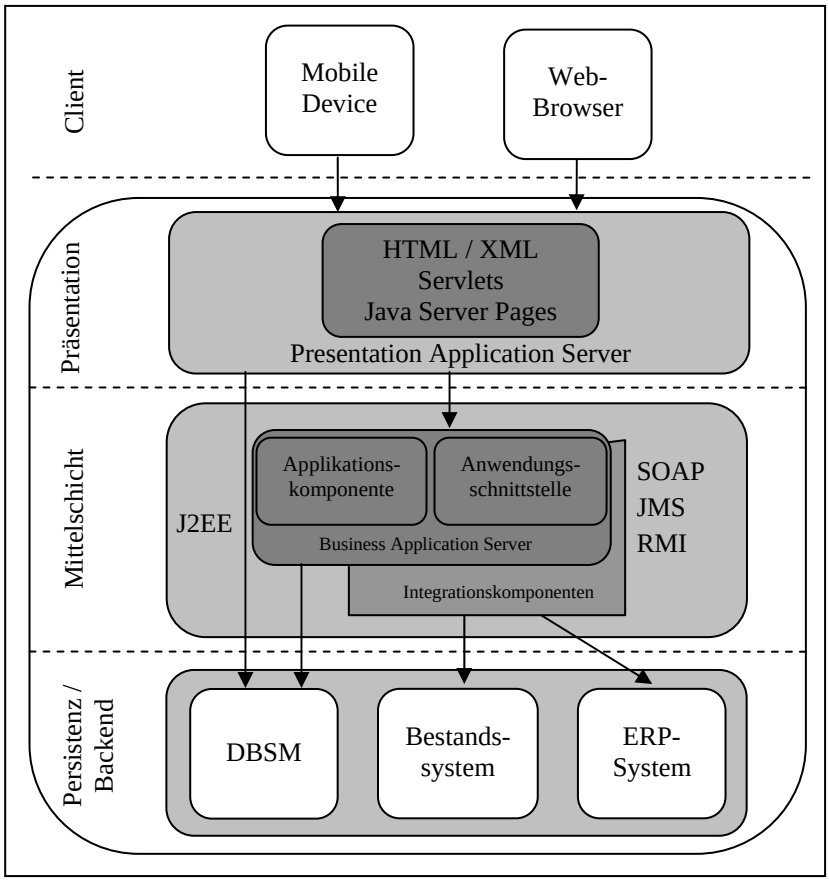
Die Funktionalität der Persistenzschicht wird in unterschiedlichen Dienstkomponenten mittels verschiedener Integrationskomponenten (u. a. Adapter, Schnittstellen) nach funktionalen Aspekten gekapselt. Damit die Dienstkomponenten angesprochen werden, benötigen sie eine entsprechende Dienstschnittstelle. Bei der Nutzung der Web Service Technologie werden u. a. die in Kapitel 3 erläuterten Standards verwendet.



Quelle: BMI (2008a), S. 77, Abbildung 6-4

Abb. 5.2: Modell einer Dreischichtarchitektur von Diensten

Da es im Allgemeinen wenig praktikabel ist auf jedem Client eine entsprechende Software für den Zugriff auf die bereitgestellten Dienste zu installieren, werden entsprechende Portalsysteme eingeführt. Diese übernehmen die Aufbereitung der erforderlichen Informationen für die Dienstnutzung und stellen sie in einer Oberfläche über einen Webbrowser zur Verfügung (entspricht der Empfehlung der SAGA-Referenz). Die Abb. 5.3 zeigt ein in diesem Fall notwendiges vierschichtiges Modell. Die Persistenzschicht enthält die bestehenden Verwaltungs- und Anwendungssysteme. Hier erfolgt die Datenhaltung unter anderm in Datenbankmanagementsystemen (DBMS) und in den Bestands- und ERP-Systemen. Die Mittelschicht übernimmt die Vermittlungsrolle zwischen den verschiedenen Verwaltungssystemen, den einzelnen Diensten und der Präsentationsschicht. Sie besteht ohne Beschränkung der Allgemeinheit aus dem Business Application Server, welcher die einzelnen Fachkomponenten und deren Schnittstellen (zusammen als Dienst bezeichnet) bereitstellt und den verschiedenen Integrationskomponenten, um auf die unterschiedlichen Verwaltungs- und Anwendungssysteme zuzugreifen. Als Protokolle können hier u. a. J2EE, SOAP, JMS, RMI verwendet werden. Die Präsentationssicht besteht aus einem „Presentation Application Server“, welcher mittels dynamischer Webseitengestaltung die einzelnen Dienste für die nicht Maschine-Maschine Kommunikation nutzbar macht. Auf der Clientschicht sind die verschiedenen Zugriffsmöglichkeiten für einen Nutzer angeordnet.



Quelle: vgl. BMI (2008a), S. 78, Abbildung 6-5

Abb. 5.3: Vierschichtenarchitektur eines E-Government-Systems

Damit dieses Schichtenmodell mittels der Web Service Technologie realisiert werden kann, müssen die drei Rollen einer SOA (siehe Kapitel 3) spezifiziert werden.

Service Broker

Der „Service Broker“ übernimmt die Funktion eines zentralen Verzeichnisdienstes und stellt damit Informationen über die zur Verfügung stehenden Dienste bereit. Die SAGA-Referenz empfiehlt hier die Verwendung des „deutschen Verwaltungsdienstverzeichnisses“ (DVDV), welches am 01.01.2007 in Deutschland offiziell in Betrieb genommen wurde. Es basiert auf dem Verzeichnisdienststandard „Open Lightweight Directory Access Protocol (Open-LDAP)“ (Open-LDAP (2008)). Der DVDV dient als Vermittler zwischen dem „Service Consumer“ und dem „Service Provider“. Er wurde speziell für die gegenseitige Identifizierung (technische Verbindungsparameter) von entsprechenden Services zur Maschine-Maschine-Kommunikation entwickelt. Für weitere Informationen über den DVDV-Verzeichnisdienst sei auf DVDV (2008) verwiesen.

Service Consumer

Der Service Consumer bzw. Dienstnutzer kann in unterschiedliche Nutzerklassen eingeteilt werden (vgl. Kapitel 2.3). Es kann speziell im E-Government-Sektor unterschieden werden zwischen:

- dem Bürger:
  Hier existieren beispielsweise die Klassen (nicht) verwaltungserfahren, (keine) Internet-Kenntnisse, (kein) Onlinezugang.

- dem Unternehmensmitarbeiter:
  Dieser tätigt im Auftrag seiner Firma regelmäßig Verwaltungsgänge für die Kunden oder für das Unternehmen selber.

- dem Verwaltungsangestellten:
  Er nutzt regelmäßig bereitgestellte Dienste über eine Nutzeroberfläche und ist in der Regel sowohl verwaltungs- als auch interneterfahren.

Dabei treten diese Nutzer allerdings nicht direkt in Kontakt mit einem Anwendungsdienst, sondern nutzen ein entsprechendes Portal, welches als Vermittler und Übersetzer verstanden werden kann, oder eine Client-Anwendung. Hier liegt die Herausforderung die Dienste nutzerklassengerecht anzubieten, welches allerdings nicht im Fokus dieser Arbeit liegt.

Service Provider

Wie schon in den Ausführungen zur Web Service Technologie aus Kapitel 3 ersichtlich, stellt der Service Provider die einzelnen Services zur Verfügung. Generell sind dabei mehrere Szenarien für den Einsatz in den Institutionen der öffentlichen Verwaltung möglich.

Ein erster Ansatz besteht darin, dass in jeder Institution ein entsprechender „Presentation Application Server“ und ein „Business Application Server“ installiert werden und somit die jeweiligen Funktionalitäten auch direkt von der jeweiligen Institution bereit gestellt werden. Dies bietet den Vorteil, dass die Institutionen weiterhin ihre Entscheidungsfreiheit über die eigenen Soft- und Hardwaresysteme behalten. Der „Web Application Server“ stellt dann die Daten- und Anwendungsdienste bereit.

Ein zweiter Ansatz besteht in der Auslagerung der einzelnen Verwaltungsdienste an externe Dienstleister oder an ein oder mehrere zentrale Rechenzentren in Deutschland.

Dies hätte langfristig den Vorteil, die Kosten für die einzelnen IT-Systeme in den Institutionen zu senken, da Neusysteme nicht mehr lokal vorgehalten werden, sondern in den zentralen Rechenzentren aufgesetzt werden. Auf diesem Weg erfolgt dann allerdings die Reduzierung der Eigenverantwortung für die Verwaltungssysteme. Um dennoch auf die bestehenden Daten und Verwaltungssysteme in den Institutionen zugreifen zu können, müssen Verbindungsmöglichkeiten mittels Datendiensten zu den lokal laufenden Verwaltungssystemen geschaffen werden. Die neuentwickelten Anwendungsdienste werden direkt in den Rechenzentren ausgeführt.

5.2 Dienstarten

Es können drei wesentliche Dienstarten unterschieden werden: Daten-, Unterstützungs- und Anwendungsdienste. Diese werden im Folgenden erläutert.

Datendienste

Der Datendienst ermöglicht die standardisierte Bereitstellung von Daten die für andere Institutionen und deren Anwendungsdienste relevant sind. Extern stellt der Datendienst entsprechende standardisierte Daten zur Verfügung. Intern müssen diese Daten bereitgestellt werden. Dabei können verschiedene Verfahren zum Einsatz kommen:

1. direkter Zugriff auf eine Datenbank (DB) innerhalb der Institution,
2. direkter Zugriff über eine Schnittstelle eines vorhandenen Anwendungssystems,
3. Informationsanforderung per Email oder Brief an einen entsprechenden lokalen Bearbeiter, der dann die Daten einspeist.

Dies gilt auch für die Rückübertragung (Speicherung) von Daten. Abb. 5.4 stellt diesen Zusammenhang grafisch für eine Institution dar. Diese Flexibilität ermöglicht es, die entsprechende Infrastruktur schnell zu errichten und je nach Finanzlage und Prioritäten der einzelnen qualifizierten Gemeinden, Ländern, aber auch des Bundes, diese nach und nach zu automatisieren.

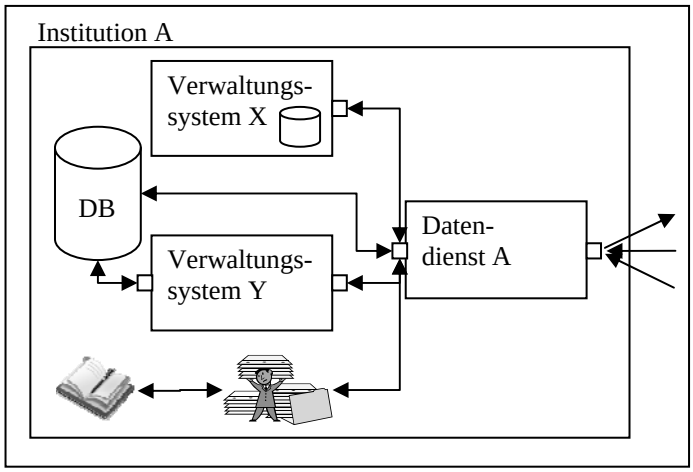


Abb. 5.4: Datendienst und seine Schnittstellen

Anwendungsdienst

Ein Anwendungsdienst stellt die eigentlichen Funktionalitäten der zukünftigen Softwarearchitektur bereit und ist, wie bereits in Kap. 4.2 definiert ein Ausführungsangebot für eine rechnergestützte Dienstleistung. Dabei können auch bereits bestehende Services eingebunden und deren Funktionalitäten genutzt werden. In

Abb. 5.5 ist dieser Zusammenhang einmal beispielhaft dargestellt. Nähere Informationen werden in Kapitel 6 zur nutzerorientierten Anwendungsdienstentwicklung dargestellt.

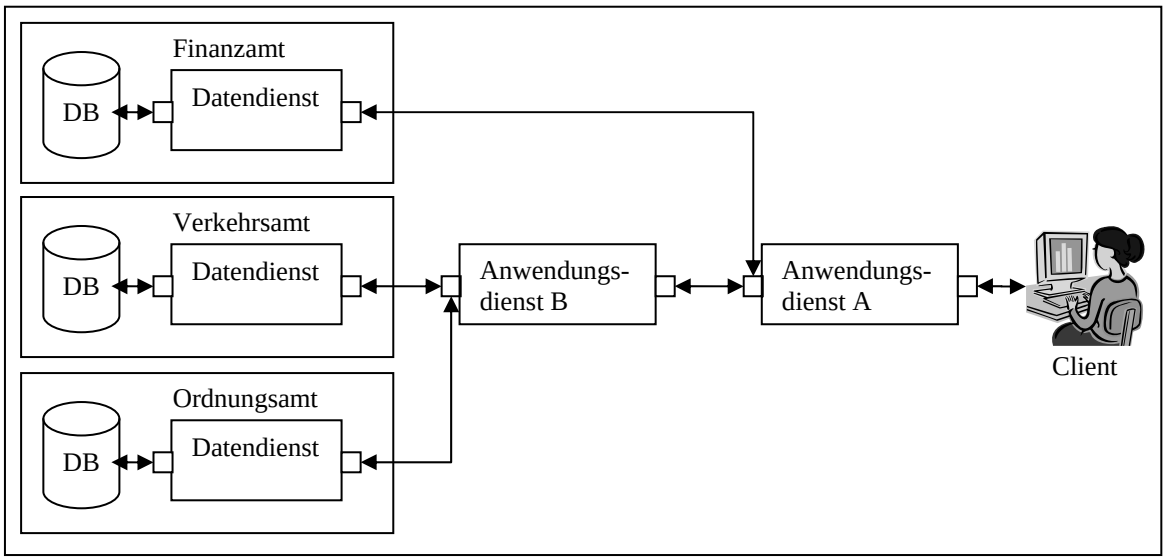


Abb. 5.5: Beispiel für das Zusammenwirken der verschiedenen Dienste

Unterstützungsdienste

In diese Kategorie werden Dienste eingeordnet, die nicht direkt den beiden oberen zugeordnet werden können. Diese können u. a. Dienste zur Datenkonvertierung, Authentifizierung und / oder Datensicherung sein.

5.3 Nachrichtenaustausch und Datensicherheit

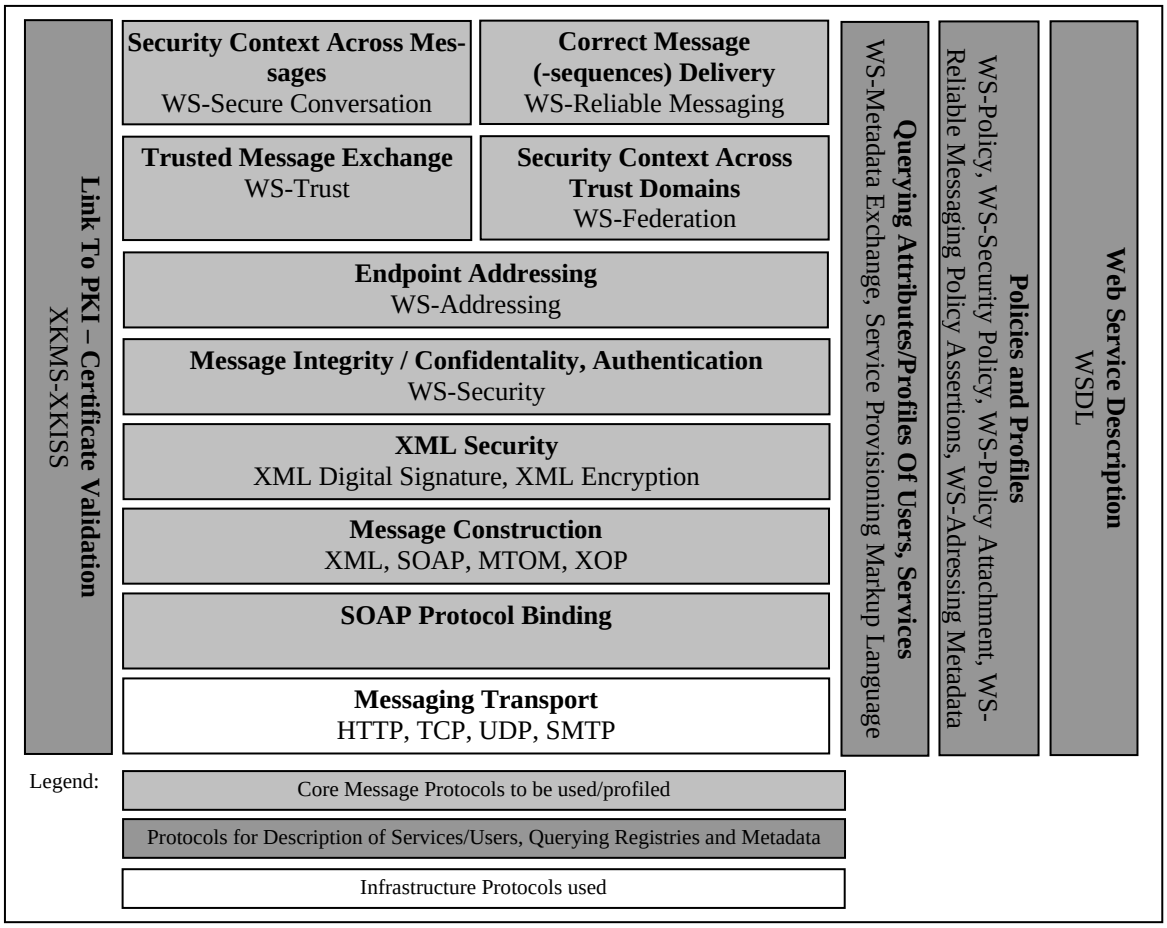
Bisher wurden die Sicherheitsanforderungen, die bereits in Kapitel 3.1 erwähnt wurden, und die Verwendung standardisierter Nachrichtenformate nicht berücksichtigt. Diese Aspekte werden nun näher betrachtet.

Sicherer Datenaustausch

Die erläuterte Architektur ermöglicht eine flexible Nutzung von bestehenden Verwaltungssystemen und entsprechenden Anwendungsdiensten. Dennoch müssen weitere Anforderungen erfüllt werden: „Für Prozessketten im E-Government [und damit auch für die Zusammenarbeit der Anwendungsdienste,] wird eine technische Infrastruktur benötigt, die neben der Vertraulichkeit (Verschlüsselung) auch die Authentisierung (digitale Identität), die Integrität (Schutz vor Verfälschungen) und die Nicht-Abstreitbarkeit der elektronischen Kommunikation in und mit [den Institutionen] der öffentlichen Verwaltung unterstützt“ (BMI (2008b), S. 32).

Damit eine vertrauliche und fälschungssichere Kommunikation gewährleistet werden kann, wurde das OSCI-Transport-Protokoll in Deutschland entwickelt. Die aktuelle Version 1.2 stößt allerdings bereits an ihre Grenzen, da diese nur einen Datenaustausch zwischen einer Informationsquelle und einer -senke über einen entsprechenden Intermediär[8] ermöglicht. Die momentan in der Entwicklung befindliche Version 2.0 setzt keinen Intermediär voraus und ermöglicht auch die Sendung von Informationen an mehrere Empfänger. Dabei wird noch stärker auf die aktuellen Web Service Standards gesetzt und damit eine hohe Zukunftssicherheit erreicht. In Abb. 5.6 sind die verwendeten Standards abgebildet. Auf eine weitere Erläuterung wird an dieser Stelle verzichtet.

[8] Vermittlungsstelle zwischen den beiden Kommunikationspartnern. Dabei müssen beide Partner dem Intermediär direkt vertrauen. Seine Aufgaben sind insbesondere die Zertifikatsprüfung, die Aufbewahrung von Laufzetteln und andere Prüfungen.



Quelle: BMI (2008b), S. 32

Abb. 5.6: OSCI Version 2.0 – verwendete Web Service Standards

Damit ermöglicht der OSCI-Standard in der Version 2.0 eine verschlüsselte, fälschungssichere Kommunikation. Die Anforderung an die Nicht-Abstreitbarkeit kann durch die Einbindung einer entsprechenden Protokollierung gewährleistet werden. Lediglich die Authentifizierung muss durch entsprechende andere Verfahren realisiert werden. Diese Themen werden nicht weiter vertieft. Für weitere Informationen sei u. a. auf BSI (2008), Kapitel IV B: IT und IT-Sicherheit verwiesen.

Standardisierte Nachrichtenformate

Die sichere Kommunikation zwischen den verschiedenen Daten- und Anwendungsdiensten, den Verwaltungssystemen und den Nutzersystemen ist mittels des OSCI-Protokolls möglich. Allerdings müssen die verschiedenen Systeme die auszutauschenden Informationen semantisch und syntaktisch richtig interpretieren. Dazu sind einheitliche Nachrichtenformate notwendig. „Standardisierungsvorhaben in der Wirtschaft haben gezeigt, dass der Versuch, eine vollständige Standardisierung von Datenmodellen

durchzuführen, meist zum Scheitern verurteilt ist“ (BMI (2008a), S. 58). Daher sollte sich auf die notwendigen Austauschformate konzentriert werden.

Im Rahmen der Deutschland Online Initiative wurden im Bereich Standardisierung (vgl. BMI (2008b)) verschiedene XÖV-Standards sowie das XÖV-Framework entwickelt. Das Framework dient der Entwicklung, Auswahl und Anpassung von XÖV-Standards. Bei den entwickelten Standards handelt es sich um auf XML basierende Austauschformate für definierte Fachaufgaben. U. a. entstand das Protokoll XMeld für das Meldewesen. Für weitere Informationen sei auf BMI (2008c) verwiesen.

Datenschutz

Für die Datensicherheit wird in SAGA Version 4.0 eine Aufteilung der Infrastruktur-Hardware in vier Zonen vorgeschlagen:

- die Informations- und Dienstzone: die Systeme dieser Zone stehen im direkten Kontakt mit den externen Netzwerken und Nutzern. Dies sind u. a. die Frontend-Systeme. Sie haben keine Zugriffsrechte auf die Datenzone.

- Logik- und Verarbeitungszone: diese Zone enthält die Systeme, auf denen die Dienste und Verwaltungssysteme ausgeführt werden, welche Daten aus der Datenzone verarbeiten.

- Datenzone: auf diese Zone darf nur von der Logik- und Verarbeitungszone und der Management-Zone zugegriffen werden. Hier werden die Datenbankmanagementsysteme und andere Datenlieferanten vorgehalten. Die Zugriffe aus der Management-Zone müssen zusätzlich überwacht werden, um Manipulationen auszuschließen.

- Management-Zone: hier werden entsprechende Überwachungs- und Administrationssysteme eingeordnet. Zusätzlich können hier entsprechende Dienste für die Nutzerverwaltung und Authentifizierung vorgehalten werden.

Unberücksichtigt bleibt die unberechtigte Datennutzung durch neu entwickelte Dienste bzw. Verwaltungssysteme, Administratoren oder durch andere unautorisierte Zugriffe. Hier müssen entsprechende Zugriffsbeschränkungen nach Nutzerklassen (z. B. Institution A, B, C) eingeführt werden, um eine nicht gesetzeskonforme Informationssammlung zu verhindern. Zusätzlich empfiehlt es sich, dass neue Dienste durch eine höhere Stelle genehmigt und abgenommen werden müssen. Für eine schnelle Entwicklung können entsprechende Beispieldatensätze vorgehalten werden, um Zeitverzögerungen für entsprechende Zugriffserlaubnisse in der Entwicklungsphase zu vermeiden.

6 Nutzerorientierte Anwendungsdienstentwicklung

In diesem Kapitel erfolgt die konzeptuelle Realisierung eines Entwicklungswerkzeuges, welches es einem Nutzer der kompetenten Nutzerklasse NK_{kl} ermöglicht einen Anwendungsdienst ohne eigene Softwareentwicklungserfahrung weitestgehend zu generieren. Das Werkzeug unterstützt den Anwendungsdienstersteller dabei, vorhandene Informationen und Dienste zu finden, zu nutzen und zu erweitern.

6.1 Entwicklung aus Nutzersicht

Für die Realisierung muss eine einfache Modellierungssprache mit einer grafischen Nutzerschnittstelle (im englischen „graphical user interface“ oder kurz GUI) gestaltet werden. Dazu werden zunächst die notwendigen Sprachelemente zur Abbildung des erweiterten Prozessmodells von Lüttich eingeführt. Anschließend erfolgt der Entwurf einer Werkzeugoberfläche.

6.1.1 Sprachelemente

Ausführung eines externen Dienstprozesses

Soll ein externer Dienst in den Ablauf des zu modellierenden Anwendungsdienstes eingebunden werden, muss das Element „Ausführung eines externen Dienstprozesses“ (Darstellung: siehe Abb. 6.1) eingefügt werden. Der Nutzer der kompetenten Nutzerklasse wählt dazu einen Dienst und die gewünschte Funktion aus einer Auswahlliste aus. Für die Auflistung wird auf den Verzeichnisdienst (siehe Kapitel 5.1) zugegriffen.

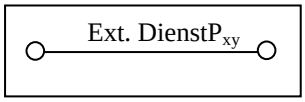


Abb. 6.1: Sprachelement: Ausführung eines externen Dienstprozesses

Transportprozess

Der Transportprozess dient zur Verknüpfung des Anwendungsdienstes mit externen Diensten (Darstellung: siehe Abb. 6.2). Dazu wird ein entsprechender externer Dienst an einen freien ausgangs- und einen eingangsseitigen Transportprozesszustand angebunden.

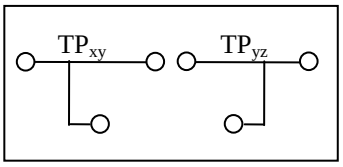


Abb. 6.2: Sprachelement: Ausführung eines Transportprozesses

Bearbeitungsprozess

Dieser Prozesstyp dient der Erstellung eigener Funktionalitäten, welche nicht durch die Weitergabe von Teilaufgaben an andere Dienste oder die veränderte Zuordnung von Informationen erbracht werden können (Darstellung: siehe Abb. 6.3).

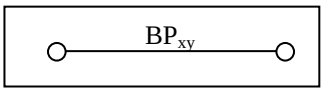


Abb. 6.3: Sprachelement: automatischer Bearbeitungsprozess

Prozesswiederholung

Oft ist es notwendig einen oder auch mehrere Teilprozess(e) zu wiederholen. Dies ermöglicht das in Abb. 6.4 dargestellte Element.

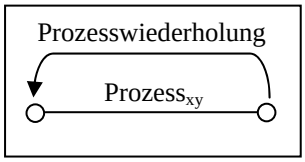


Abb. 6.4: Sprachelement: Prozesswiederholung

Anfangs- und Endprozess

Damit der zu entwickelnde Anwendungsdienst einen definierten Start- und Endpunkt besitzt, werden ein Anfangs- und ein Endprozess eingeführt.

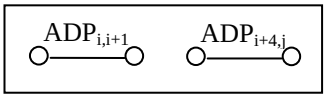


Abb. 6.5: Sprachelement: Anfangs- und Endprozess

Beispielprozess

Mittels dieser Sprachelemente wird es möglich, einen Dienst aus Nutzersicht zu modellieren und somit den Programmierungsaufwand stark zu reduzieren. Lediglich die Funktionalität des Bearbeitungsprozesses muss mittels einer Programmiersprache erfolgen. In Abb. 6.6 sind alle Sprachelemente in einem einfachen Beispiel dargestellt.

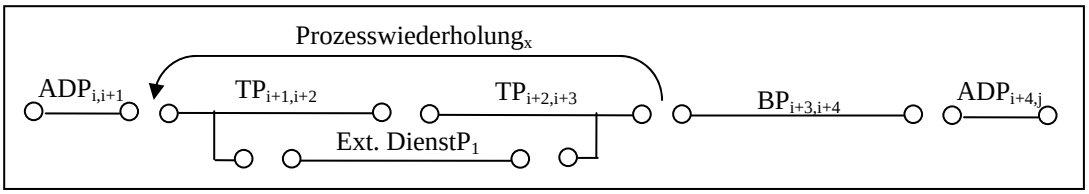


Abb. 6.6: Werkzeugumgebung - Beispielprozess

6.1.2 Entwurf der Werkzeugoberfläche

Das Layout der Werkzeugoberfläche und der -dialoge wird hier nur ansatzweise beschrieben, da es sich um eine Grundlagenarbeit handelt. Für eine vollständige Realisierung müssen weitere Aspekte, wie z. B. die der Software-Ergonomie, betrachtet werden. Grundsätzlich bietet die Oberfläche eine Menüleiste, einen Auswahlbereich für die verfügbaren Elemente und einen Arbeitsbereich (siehe Abb. 6.7).

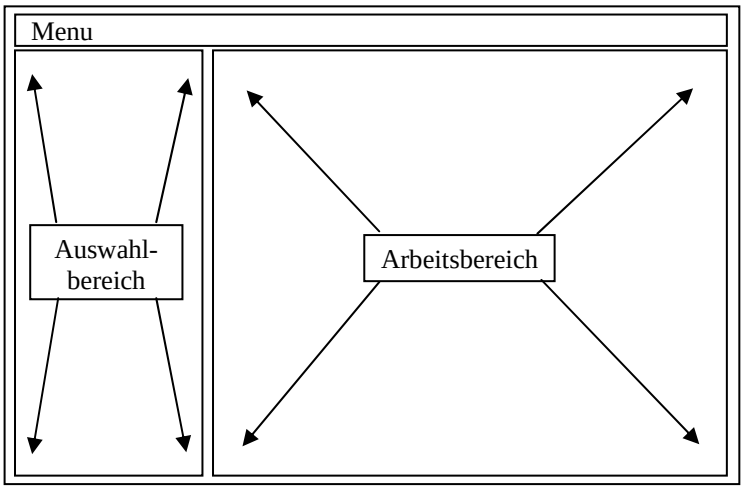


Abb. 6.7: Werkzeugumgebung - Arbeitsbereich

Die verfügbaren Elemente können aus dem Auswahlbereich in den Arbeitsbereich gezogen und dort entsprechend den folgenden Regeln angeordnet werden:

- Ein externer Dienstprozess kann an einem Transportprozessausgangs- und -eingangszustand geknüpft werden.

- Der Bearbeitungsprozess ermöglicht die individuelle Realisierung von Funktionalitäten, die nicht von vorhandenen Diensten abgedeckt werden können. Dieses Sprachelement wird direkt an ein anderes Prozesselement angeknüpft.

- Eine Wiederholung eines oder mehrerer Teilprozesses(e) ist möglich. Mehrere Wiederholungselemente dürfen sich nicht überschneiden.

Nachdem die verschiedenen Prozesse verknüpft wurden, kann bei den Transport- und den Bearbeitungsprozessen per Mausklick ein Dialogfenster geöffnet werden, welches die Zuordnung der zu übergebenden Informationen ermöglicht. Dazu wird das Fenster zweigeteilt. Links befinden sich die bisher im Anwendungsdienst vorhandenen Informationen und rechts die für den externen Dienstprozess benötigten Informationen. Bei einem eingehenden Transportprozess müssen die Zuordnungen von links und rechts vertauscht werden. Im Dialog für den Bearbeitungsprozess müssen beide Varianten abgefragt werden. Mit Hilfe der Maus können nun Verbindungen zwischen den Informationselementen auf beiden Seiten hergestellt werden. Die Werkzeugumgebung überprüft dabei, ob eine automatische Konvertierung zwischen den zugrunde liegenden Datentypen möglich ist. Sollte dies nicht möglich sein, erscheint eine Warnung und der Nutzer muss die Konvertierungseinstellung manuell vornehmen. Sind Informationen im Anwendungsdienst noch nicht vorhanden, können die benötigten Informationsobjekte hier erstellt werden. Erfolgt dann vorgelagert keine Belegung der Informationsobjektausprägung wird dieses Informationsobjekt dem Eingangsverwaltungsobjekt des Anwendungsdienstes zugeordnet.

Das Dialogfenster des Wiederholungselements ermöglicht die Eingabe der Bedingung und ob diese zu Beginn der Wiederholungsfolge oder am Ende geprüft werden soll. Im ersten Fall ist es möglich, dass die Folge gar nicht ausgeführt wird.

Mittels des Endprozesses können Informationen an den aufrufenden Dienst zurückgesendet werden. Hier kann ebenfalls die Auswahl der zu sendenden Informationen durch ein Dialogfenster durchgeführt werden.

Bevor mit der eigentlichen Anwendungsdienstmodellierung begonnen werden kann, müssen in einem Dialog der entsprechende Name des Anwendungsdienstes, der spätere Ausführungsort und der Protokolldienst (der entsprechende URI) festgelegt werden. Für die Auswahlmöglichkeiten greift das Entwicklungswerkzeug auf den „Service Broker“ zu, in welchem alle relevanten Orte verzeichnet sind. Dennoch kann auch ein neuer URI angegeben werden. Diese Informationen müssen auch bei der Anknüpfung eines neuen Bearbeitungsprozesses mitgeteilt werden.

6.2 Konzeptuelle Realisierung

Nachdem die Sprachelemente vorgestellt wurden, muss nun die Übersetzung des damit realisierten Modells in die konzeptuelle Ebene und speziell in die Sprache WS-BPEL erfolgen. Dazu werden zunächst die einzelnen Elemente erläutert. Anschließend erfolgt die Modellierung des Generierungsprozesses als EPK. Anzumerken ist an dieser Stelle, dass der konzeptueller Rahmen (siehe Kapitel 2.4) vollständig durch die BPEL-Ausführungsumgebung gegeben ist und somit nur die Funktionalität des Anwendungsdienstes erstellt werden muss.

6.2.1 Realisierung der nutzerorientierten Sprachelemente

Auf der konzeptuellen Ebene wird das bei der Modellierung entstehende Prozessmodell als eine doppelt verkette Liste verwaltet. Diese ermöglicht neben der schnellen Ergänzung von weiteren Elementen innerhalb des zu erstellenden Anwendungsdienstes auch die leichtere Erweiterung der Werkzeugumgebung um zusätzliche Sprachelemente.

Ausführung eines externen Dienstprozesses

Aus konzeptueller Sicht erhält dieses Element einen Transportprozessausgang und einen -eingang als Listenelement. Für die BPEL-Code Generierung werden dem Dienstprozess ein eindeutiger Name sowie die URI der WSDL-Datei des aufzurufenden Dienstes und die gewünschte Funktion zugeordnet. Damit kann das Element wie in Tab. 6.1 konzipiert werden.

Tab. 6.1: Sprachelement: Ausführung externer Dienst - Datenbeschreibung

Elementbeschreibung	Datentyp
Transportprozessausgangzustand	Listenelement
Transportprozesseingangszustand	Listenelement
Dienstname	String
aufzurufender Dienst (WSDL-Datei)	Link
auszuführende Funktion	String

Transportprozess

Der Transportprozess wird als ausgehender und als eingehender Transportprozess verwendet. In diesem Element müssen neben der Verknüpfung auf die externen Dienste, die Informationen für die zu verknüpfenden Informationen gespeichert werden, damit in

dem Generierungsprozess die entsprechenden Zuordnungen erstellt werden können. Der ausgehende Transportprozess wird, wie in Tab. 6.2 dargestellt, konzipiert. Für die Generierung, in Verbindung mit dem Aufruf des externen Dienstprozesses, im BPEL-Code wird das Sprachelement <invoke> verwendet. Es wird dabei immer eine Rückantwort angefordert, da dies dem Telekooperationsprozess nach Lüttich (siehe Kapitel 2.2.2) entspricht und im Rahmen einer vollständigen Nachvollziehbarkeit der durchgeführten Aufträge für die Protokollierung notwendig ist.

Tab. 6.2: Sprachelement: Transportprozess - Datenbeschreibung

Elementbeschreibung	Datentyp
Prozessvorgängerelement	Listenelement
Prozessnachfolgerelement	Listenelement
Transportprozessausgangzustand	Listenelement
Transportprozesseingangszustand	Listenelement
Dienstname	String
Informationszuordnung ausgehend (lokale Information, Datentyp; Information in Nachricht, Datentyp)	Array[Array[String, String], [String, String]]

Bearbeitungsprozess

Der Bearbeitungsprozess dient der Informationsverarbeitung, welche nicht durch bestehende Dienste realisiert werden kann. Wird dieser Prozesstyp notwendig, muss die eigentliche Funktionalität von einem Softwareentwickler in einem neuen externen Dienst bereitgestellt werden. Es existieren Ansätze, die Modellierungssprache BPEL um Elemente der Programmiersprache Java zu erweitern (vgl. BEA (2004)). Diese werden allerdings nicht weiter beleuchtet.

Für die Anwendungsdienstentwicklung können dennoch weitere Vorgaben gemacht werden. Es kann festgelegt werden, welche Informationen an den Bearbeitungsprozess und damit an den zusätzlich zu entwickelnden Dienst gesendet und welche empfangen werden sollen. Dementsprechend wird die WSDL-Datei weitestgehend generiert und damit die Entwicklungsarbeit wiederum auf die eigentliche Funktionalität (nutzerorientierter Entwicklungsansatz) reduziert. Die konzeptuelle Datenbeschreibung des Bearbeitungsprozesses kann wie in Tab. 6.3 dargestellt werden. Das Beschreibungsfeld dient hierbei der Erläuterung der gewünschten Funktionalität.

Tab. 6.3: Sprachelement: Bearbeitungsprozess - Datenbeschreibung

Elementbeschreibung	Datentyp
Prozessvorgängerelement	Listenelement
Prozessnachfolgerelement	Listenelement
Dienstname	String
Speicherort des Dienstes (WSDL-Datei)	Link
auszuführende Funktion	String
Informationszuordnung ausgehend (lokale Information, Datentyp; Information in Nachricht, Datentyp)	Array[Array[String, String], [String, String]]
Informationszuordnung eingehend (Information in Nachricht, Datentyp; lokale Information, Datentyp)	Array[Array[String, String], [String, String]]
Beschreibung der Funktionalität	String

Prozesswiederholung

Die Prozesswiederholung muss die Informationen über den Anfangs- und Endpunkt hinterlegen. Dazu werden aus konzeptioneller Sicht zwei Listenelemente, welche den Bereich einschließen, eingefügt. Beide enthalten einen internen Bezeichner, so dass das Element eindeutig identifiziert werden kann.

Zusätzlich wird eine Wiederholungsbedingung benötigt. Das erweiterte Prozessmodell nach Lüttich macht an dieser Stelle keine Aussagen über die Art der Wiederholung. Daher wird sich an den Möglichkeiten der Modellierungssprache WS-BPEL orientiert. Diese bietet die Operatoren <while> (prüft die Bedingung am Anfang der Wiederholungsfolge) und <repeatUntil> (prüft die Bedingung nach jeder Ausführung der Wiederholungsfolge).

Wird die Bedingung im Anfangselement abgelegt, handelt es sich um eine vorgelagerte Prüfung und der BPEL-Operator <while> wird verwendet. Ist diese im Endelement abgelegt, wird die nachgelagerte Prüfung realisiert. Damit kann das Datenobjekt wie in Tab. 6.4 realisiert werden.

Tab. 6.4: Sprachelement: Prozesswiederholung - Datenbeschreibung

Elementbeschreibung	Datentyp
Bezeichner	String
Bedingung	String

Anfangs- und Endprozess

Diese Elemente bilden aus Sicht der konzeptuellen Ebene das Anfangs- und Endglied der doppelt verketteten Liste. Das Anfangsprozesselement enthält den Dienstnamen, den späteren Ausführungsort und die Informationszuordnung für die eingehende Nachricht (siehe Tab. 6.5).

Tab. 6.5: Sprachelement: Anfangsprozess - Datenbeschreibung

Elementbeschreibung	Datentyp
Prozessnachfolgerelement	Listenelement
Dienstname	String
Speicherort des Dienstes (WSDL-Datei)	Link
Informationszuordnung eingehend (Information in Nachricht, Datentyp; lokale Information, Datentyp)	Array[Array[String, String], [String, String]]

Das abschließende Element muss lediglich die Informationszuordnung für die ausgehende Nachricht speichern (siehe Tab. 6.6). Durch diese ein- und ausgehende Zuordnung ist es möglich standardisierte Nachrichtentypen (siehe Kapitel 5.3) zu verwenden.

Tab. 6.6: Sprachelement: Endprozess - Datenbeschreibung

Elementbeschreibung	Datentyp
Prozessvorgängerelement	Listenelement
Informationszuordnung ausgehend (lokale Information, Datentyp; Information in Nachricht, Datentyp)	Array[Array[String, String], [String, String]]

6.2.2 Generierungsprozess des BPEL-Codes

Im Generierungsprozess wird zunächst geprüft, ob die Belegung der Variablenausprägungen der zu sendenden Informationen vor dem Senden erfolgt. Ist dies nicht der Fall, wird der Nutzer in einer Bildschirmausgabe gefragt, ob dies korrekt ist. Wird dies bestätigt, erfolgt die Zuordnung der erst später belegten Variablen als Diensteingangsinformationen. Die gleiche Prüfung erfolgt in Bezug auf verwendete aber nicht belegte Variablen.

Nun wird die Umsetzung des Prozessmodells in den BPEL-Code vorgenommen. Dazu erfolgt die Auswahl des Startelements. Die enthaltenen Informationen werden für den vollständigen BPEL-Code zwischengespeichert. Anschließend kann der Nutzer, insofern gewünscht, einen standardisierten Nachrichtentyp auswählen. Daraufhin muss der entsprechende Eingangsnachrichtentyp als WSDL-Datei erzeugt werden und eine Sequenz <sequence> für die weiteren Aktivitäten erzeugt werden. Dann wird für jedes

Listenglied die Art des Werkzeugselements bestimmt und in dessen Abhängigkeit die folgenden Aktionen durchgeführt:

- Transportprozessausgang:
  Zunächst wird eine WSDL-Wrapper-Datei für den aufzurufenden Dienst erzeugt, welche die ergänzenden Informationen über den „partnerLinkType“ sowie die „Correlations“ enthält. Dann werden die weiteren Informationen aus dem Listenelement zwischengespeichert. Nun kann in den BPEL-Code die Konvertierung der Informationen in das notwendige Nachrichtenformat mit Hilfe des <assign>-Operators veranlasst und der Dienst durch ein <invoke> aufgerufen werden.

- Transportprozesseingang:
  An dieser Stelle ist es lediglich notwendig die Zuordnung der empfangenen Informationen zu den lokal vorhandenen, mit Hilfe des <assign>-Operators, hinzuzufügen.

- Bearbeitungsprozess:
  Da der Bearbeitungsprozess aus konzeptueller Sicht einen externen Dienstprozess darstellt, gestaltet sich der BPEL-Code ähnlich. Zunächst werden durch ein <assign> die lokalen Informationen dem neu zu entwickelnden Dienst übergeben. Anschließend wird dieser aufgerufen (<invoke>) und dann die notwendigen zurückerhaltenen Informationen wieder den lokalen zugewiesen (<assign>). Nun erfolgt die Generierung der WSDL-Datei des neuen Dienstes für die Funktionalität des Bearbeitungsprozesses und es wird die Entwicklung veranlasst.

- Anfang Prozesswiederholung:
  An dieser Stelle wird geprüft, ob eine Wiederholungsbedingung enthalten ist. Wenn dies der Fall ist, handelt es sich um eine <while>-Schleife und folgender Code wird eingefügt: „<while><condition …> … </condition>“. Andernfalls muss eine <repeatUntil>-Schleife mit „<repeatUntil>“ als einzufügenden Code verwendet werden. Abgeschlossen wird dieses Sprachelement mittels dem Starten einer Sequenz (<sequence>).

- Ende Prozesswiederholung:
  Zunächst wird die Sequenz beendet (</sequence>). Anschließend erfolgt die Prüfung, welcher Schleifentyp geschlossen werden muss. Die <while>-Schleife erfordert den BPEL-Code „</while>“ und die <repeatUntil>-Schleife „<condition> … </condition></repeatUntil>“.

- Endprozess:
  Wird dieses Element erreicht, wurde das entwickelte Modell vollständig durchlaufen.

Es wird nun die Erstellung des notwendigen Ausgangsnachrichtentyps sowie der Informationszuordnung <assign> veranlasst. Dazu ist in einem Dialogfenster die Auswahl eines Standardnachrichtentyps möglich. Anschließend wird die Sequenz beendet (</sequence>).

Wurden alle Elemente umgesetzt, erfolgt die vollständige Generierung der BPEL-Grundstruktur (siehe Kapitel 3.2). Für die „CorrelationSets“ wird ein automatisierter einheitlicher „Identifier“ vorgeschlagen, da die Verwaltung durch einen Nutzer erweiterte Programmierkenntnisse besitzen muss. Die „faultHandlers“ werden in dieser Arbeit lediglich in der Form realisiert, dass die durchgeführten Aktivitäten rückgängig gemacht werden und eine Fehlermeldung im Protokollierungsdienst (das Fehler- und Nachrichtenprotokoll) hinterlegt wird. Auf die Verwendung des optionalen „eventHandlers“ wird an dieser Stelle verzichtet.

Im Anhang A ist der erläuterte Algorithmus als EPK modelliert. Wird der Beispielprozess aus Abb. 6.6 mittels des Algorithmus umgesetzt, entsteht folgender Code:

```
<process name="ProcessName">
     <partnerLinks>     …  </partnerLinks>
     <variables>        …  </variables>
     <correlationSets>  …  </correlationSets>
     <faultHandlers>    …  </faultHandlers>
     <sequence>
          <while>                                          Prozesswiederholung
               <condition … >        …    </condition>
               <sequence>
                    <assign>         …    <assign>   }
                    <invoke … >      …    </invoke>  }  Ext. Dienstprozess
                    <assign>         …    <assign>   }
                    <assign>         …    <assign>   }
                    <invoke … >      …    </invoke>  }  Bearbeitungsprozess
                    <assign>         …    <assign>   }
               </sequence>
          </while>
     </sequence>
</process>
```

7 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde der nutzerorientierte Systementwicklungsansatz nach Lüttich umfassend dargestellt. Anschließend wurde die Realisierbarkeit dieses Ansatzes mit Hilfe der Web Service Technologie und speziell der Geschäftsprozessmodellierungssprache WS-BPEL untersucht. Aufgrund des weitaus größeren Sprachumfangs von BPEL konnte die Überführung vollständig realisiert werden. Im Zuge der gestiegenen Komplexität und der losen Kopplung von verschiedenen Diensten sollte das Prozessmodell nach Lüttich allerdings um folgende Aspekte erweitert werden:

- Parallele Prozesse:
  Im Zuge einer lose gekoppelten Dienstarchitektur ist es sinnvoll mehrere Aktivitäten gleichzeitig ausführen zu können. Im Entwicklungswerkzeug kann dazu das Transportprozesselement um die Fähigkeit des gleichzeitigen Startens mehrerer externer Dienstprozesse erweitert werden. Zusätzlich wäre die Realisierung von unterschiedlichen Ankunftszeiten der Antwortnachrichten sinnvoll. Aufgrund des bereits geteilten Transportprozesselements ist dies mit geringem Aufwand zu realisieren.

- Entscheidungselemente:
  BPEL bietet bereits die Möglichkeiten durch Entscheidungsstränge unterschiedliche Prozesswege zu realisieren. Im Modell nach Lüttich wird diese Möglichkeit nicht berücksichtigt. Diese Funktionalität muss dort innerhalb der Funktionskomplexe selbst programmiert werden.

- Erweiterte Fehlerbehandlung:
  Auch die Fehlerbehandlung wird nicht berücksichtigt. Treten diese auf, werden sie im Fehler- und Nachrichtenprotokoll festgehalten. Dieses Vorgehen wurde auch in dieser Arbeit realisiert. Aufgrund der gesteigerten Komplexität und der Vielzahl an auszuführenden Diensten und Verwaltungssystemen sollte dieser Aspekt allerdings weitere Beachtung finden.

Für die Erweiterungen müssen allerdings die theoretischen Grundlagen für das Modell von Lüttich erstellt werden. Hier empfiehlt sich eine Anschlussforschungsarbeit.

Dennoch wird vor allem wegen der wenigen Sprachelemente des konzipierten Entwicklungswerkzeuges eine einfache Einarbeitung auch für einen Nutzer ohne Programmierkenntnisse möglich. Dies wird zusätzlich durch die integrierte E-Government-Softwareumgebung erreicht. Für den Nutzer werden die Entwicklungsdetails der konzeptuellen Ebene vollständig verdeckt, so dass eine rein nutzerorientierte Anwendungsdienstentwicklung möglich wird. Funktionalitäten die nicht auf diese Weise realisiert werden

können, werden aus der Werkzeugumgebung automatisch an einen Softwareentwickler weitergeleitet.

Ausblick: Portal-Ansatz für ein modernes E-Government

Der Portalansatz wird im Zuge des E-Governments bereits stark verfolgt. Dabei werden den Portalen folgende Aufgaben zugeordnet:

- Informationsbereitstellung,
- Verwaltungsdienstleistungen,
- Formularbereitstellung.

Aufgrund der neuen EU-Dienstleistungsrichtlinie müssen zusammenhängende Verwaltungsdienstleistungen ab Ende 2009 von einem Ansprechpartner (persönlich und online) erfüllt werden können. In diesem Zusammenhang wird von einem „One-Stop-Government“ gesprochen. Dies bedeutet, dass ein Nutzer einer Nutzerklasse NK_{kl} auf eine Vielzahl von unterschiedlichen Verwaltungssystemen zugreifen müsste.

An dieser Stelle kommt der in dieser Arbeit vorgestellte Entwicklungsansatz für einen Anwendungsdienst zum Tragen. Allerdings fehlt die grafische Nutzeroberfläche. Dazu können entsprechende Portale genutzt werden. Sie bieten den unterschiedlichen Nutzern eine einheitliche Oberfläche zur Nutzung der unterschiedlichen Funktionen. Zusätzlich können aber auch Unternehmen neue Dienstleistungen für den Bürger schaffen. Ein Beispiel dafür wäre der vollständig rechnergestützte Kauf eines Kraftfahrzeuges. Der Kunde wählt das Fahrzeug online aus, führt die Kaufabwicklung durch und kann aus dem Portal heraus die Zulassung des Wagens veranlassen. Dabei greift der Service des Autohauses auf den externen Dienst der Zulassungsstelle, auf der Ebene des Bundes, zu und führt die gewünschte Funktion aus. Die Abb. 7.1 visualisiert noch einmal die vorherigen Ausführungen.

Die Nutzung der nutzerorientierten Systementwicklungsansatzes bietet für die Portalentwicklung vor allem in Bezug auf die automatische Oberflächengenerierung Vorteile. Es empfiehlt sich daher eine weitere Forschungsarbeit über den Aufbau eines entsprechenden Portalsystems mit einer automatischen Oberflächengenerierung zu erstellen.

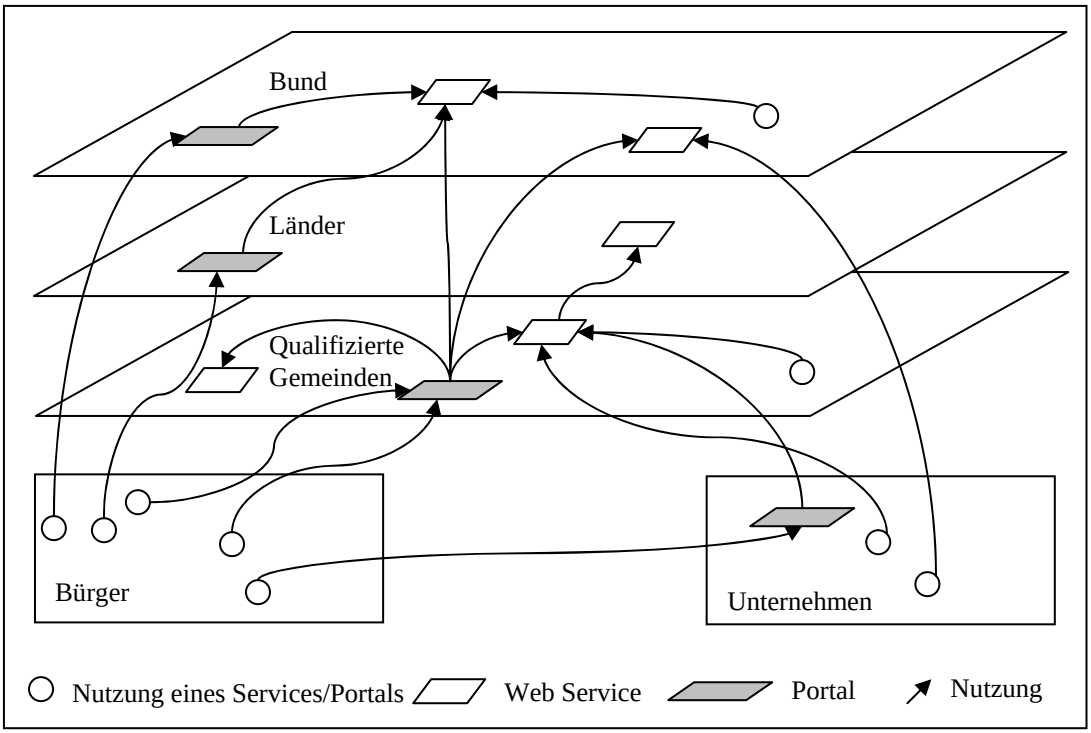


Abb. 7.1: Ebenen- und Portal-Konzept bei der Nutzung von Web Services

Anhang

A EPK des Generierungsprozesses

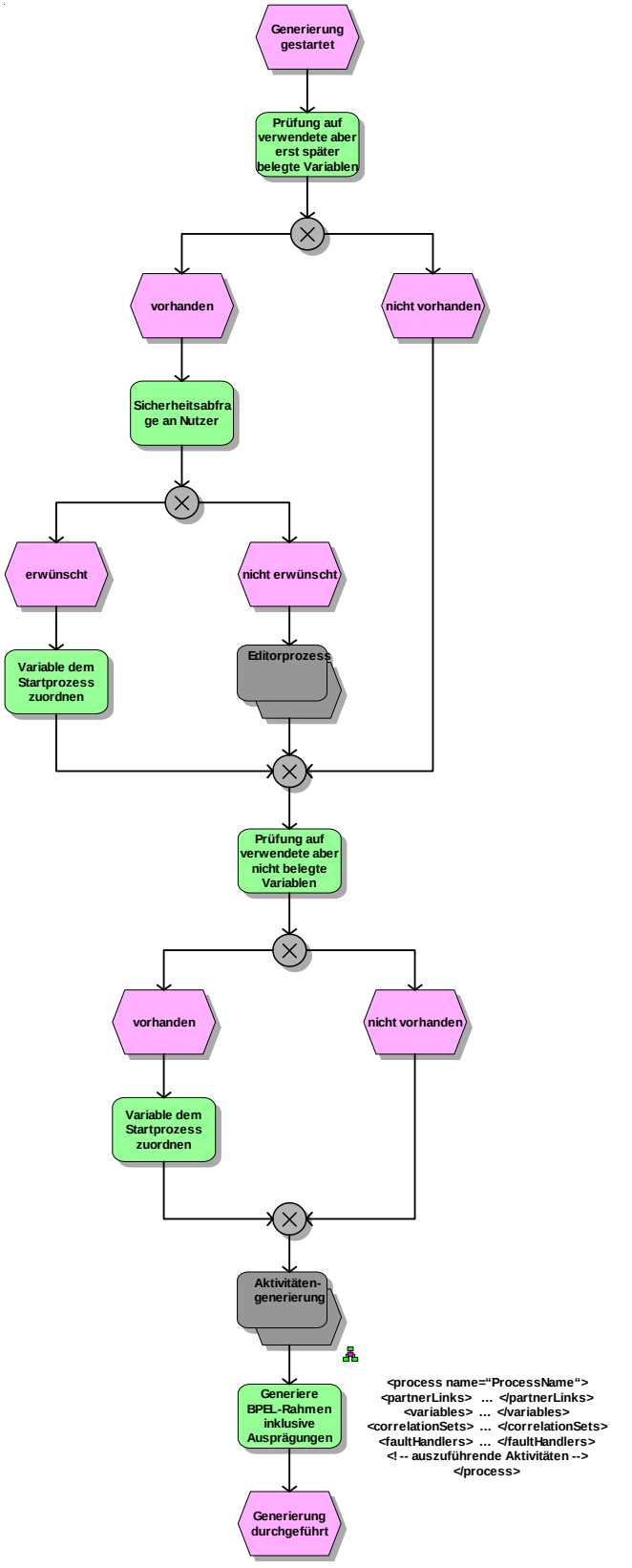


Abb. A.1: EPK - Generierungsprozess

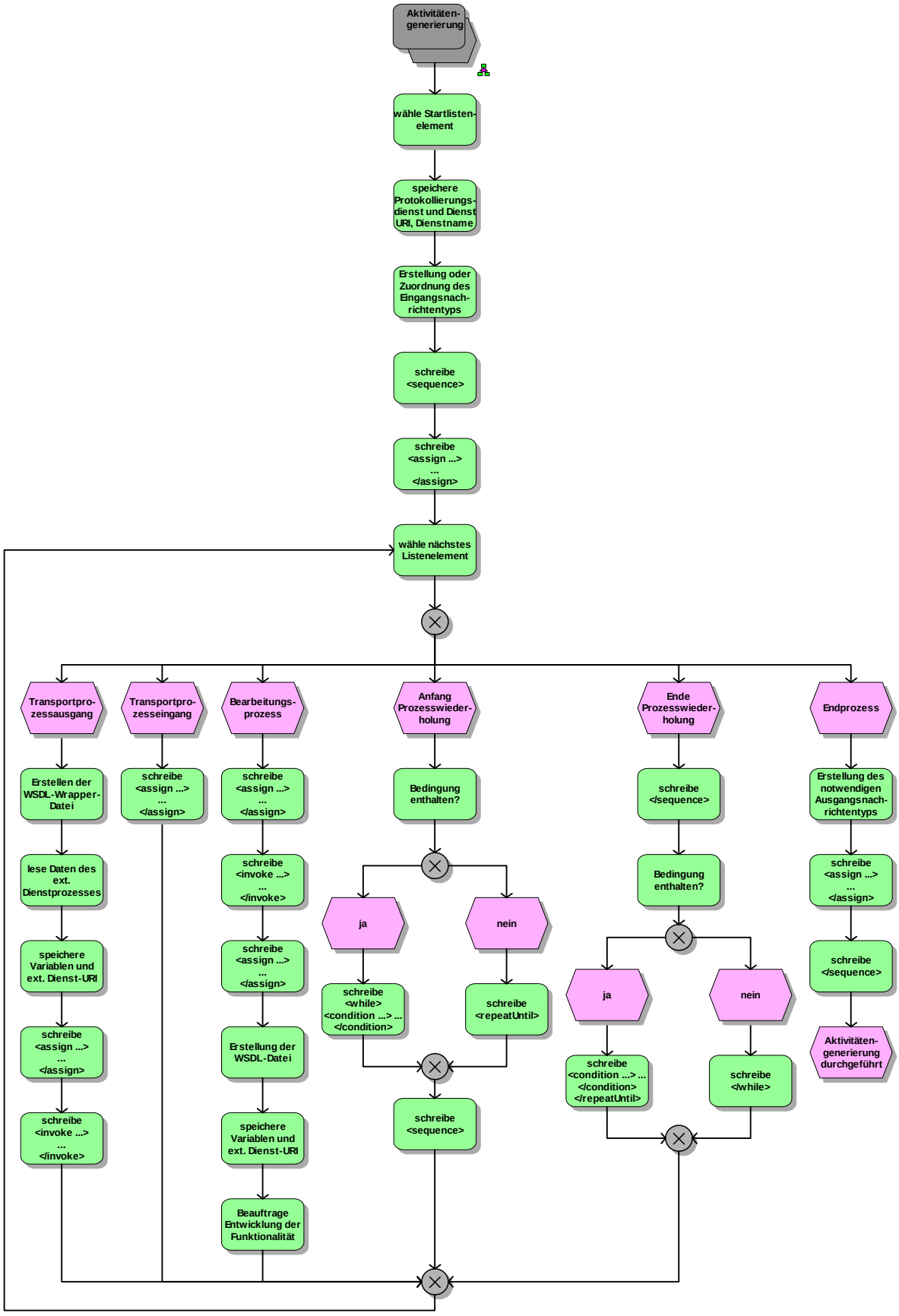


Abb. A.2: EPK - Aktivitätengenerierung

Literaturverzeichnis

BEA (2004): BPELJ: BPEL for Java - A Joint White Paper by BEA and IBM. http://ftpna2.bea.com/pub/downloads/ws-bpelj.pdf. (19.09.2008)

BMI (2007): Handbuch für Organisationsuntersuchungen und Personalbedarfsermittlung. http://www.orghandbuch.de/cln_115/nn_414290/OrganisationsHandbuch/DE/ohb__pdf,templateId=raw,property=publicationFile.pdf/ohb_pdf.pdf

BMI (2008a): SAGA Version 4.0 - Standards und Architekturen für E-Government-Anwendungen. http://gsb.download.bva.bund.de/KBSt/SAGA/SAGA_v4.0.pdf (19.09.2008)

BMI (2008b): Leitfaden für die Entwicklung von Standards für den elektronischen Datenaustausch (XÖV-Standards). Version 1.0. http://www.deutschland-online.de/DOL_Internet/binarywriterservlet?imgUid=32f3e75c-fa63-9114-fbf1-b1ac0c2f214a&uBasVariant=22222222-2222-2222-2222-222222222222 (19.09.2008)

BMI (2008c): Leitfaden für die Entwicklung von Standards für den elektronischen Datenaustausch (XÖV-Standards). Version 1.0. http://www.deutschland-online.de/standardisierung (19.09.2008)

Braun, L. (2002): Statistisches Prozessmanagement - Modellierung betrieblicher Prozessnetzwerke mit multivariaten Methoden. Marburg.

BSI (2006): E-Government-Handbuch. http://www.bsi.bund.de/fachthem/egov/6.htm (19.09.2008)

DMTF (2008): Web Services for Management (WS-Management) Specification. http://www.dmtf.org/standards/published_documents/DSP0226_1.0.0.pdf (19.09.2008)

DVDV (2008): BIT - DVDV. http://www.bit.bund.de/cln_046/nn_373366/BIT/DE/Zentrale__Dienste/DVDV/node.html?__nnn=true. (19.09.2008)

Dostal, W.; Jeckle, M.; Melzer, I.; Zengler, B. (2005): Service-orientierte Architekturen und Webservices. Heidelberg.

Dustdar, S.; Gall, H.; Hauswirth, M. (2003): Software-Architekturen für Verteilte Systeme – Prinzipien, Bausteine und Starndardarchitekturen für moderne Software. Berlin u. a.

Engel, A. (2001): Telekooperation als Herausforderung für das IT-Management. In: Gora, W.; Bauer, H. (2001), S. 317-331.

Ernst, H. (2000): Grundlagen und Konzepte der Informatik, 2. Aufl., Braunschweig u. a.

Gora, W.; Bauer, H. (2001): Virtuelle Organisationen im Zeitlater von E-Business und E-Government – Einblicke und Ausblicke. Berlin u. a.

Henshall, J.; Shaw, S. (1992): OSI praxisnah erklärt - Der Standard für die Computer-Kommunikation. München u. a.

Hoffmann-Riem, W.; Schmidt-Aßmann, E.; Voßkuhle, A. (2006): Grundlagen des Verwaltungsrechts - Band 1: Methoden Maßstäbe Aufgaben Organisation. München

IBM (2008a): Standards and Web services. http://www-128.ibm.com/developerworks/webservices/standards/ (19.09.2008)

Kircher, H. (2007): IT - Technologien Lösungen Innovationen. Heidelberg.

Lenk, K.; Brüggemeier, M.; Reichard, C. (2005): Electronic Government: Ein Konzept zur innovativen Neugestaltung öffentlicher Aufgabenwahrnehmung. Dissertation, Hamburger Universität für Wirtschaft und Politik. Münster

Lucke, J., Reinermann, H. (2000): Speyerer Definition von Electronic Government - Ergebnisse des Forschungsprojektes, Forschungsinstitut für öffentliche Verwaltung bei der Deutschen Hochschule für Verwaltungswissenschaften Speyer. http://foev.dhv-speyer.de/ruvii/Sp-EGov.pdf (19.09.2008)

Lüttich, H.-J. (2007a): Vorlesungsskript Nutzerorientierte Systementwicklung

Lüttich, H.-J. (2007b): Vorlesungsskript E-Government

Lüttich, H.-J. (2007c): Vorlesungsskript Telekooperation

Lüttich, H.-J. (2007d): Vorlesungsskript Werkzeugentwicklung

Lüttich, H.-J.; Turowski, K. (2000): Zur komponentenbasierten Gestaltung rechnergestützter Verwaltungssysteme. In: Lüttich, H.-J.; Rautenstrauch, K. (2000), S. 458-471.

Lüttich, H.-J.; Rautenstrauch, C. (2000): Verwaltungsinformatik 2000 - Verwaltungsinformatik in Theorie, Anwendung und Hochschulausbildung. Halle.

Manthey, R. (2002): Vorlesungsskript Informationssysteme - DB-Entwurf (2.Teil). Universität Bonn. http://www.informatik.uni-bonn.de/III/lehre/vorlesungen/Informationssysteme/WS02/folien/DB4b-1.pdf (19.09.2008)

Meffert, H.; Bruhn, M. (2006): Dienstleistungsmarketing - Grundlagen – Konzepte – Methoden. 5. Aufl., Wiesbaden.

Melzer, I. (2007): Service-orientierte Architekturen mit Web Services - Konzepte – Standards – Praxis. München.

OASIS (2004): Web Services Reliable Messaging TC WS-Reliability 1.1 OASIS Standard, 15 November 2004. http://docs.oasis-open.org/wsrm/ws-reliability/v1.1 (19.09.2008)

OASIS (2006): Web Services Security: SOAP Message Security 1.1 (WS-Security 2004) - OASIS Standard Specification. http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf (19.09.2008)

OASIS (2007a): Web Services Atomic Transaction (WS-AtomicTransaction) Version 1.1 - OASIS Standard incorporating Approved Errata. http://docs.oasis-open.org/ws-tx/wstx-wsat-1.1-spec-errata-os.pdf (19.09.2008)

OASIS (2007b): Web Services Business Activity (WS-BusinessActivity) Version 1.1 - OASIS Standard incorporating Approved Errata. http://docs.oasis-open.org/ws-tx/wstx-wsba-1.1-spec-errata-os.pdf (19.09.2008)

OASIS (2007c): Web Services Business Process Execution Language Version 2.0 - OASIS Standard. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf. (19.09.2008)

OASIS (2008a): OASIS Standards and Other Approved Work. www.oasis-open.org/specs (19.09.2008)

OASIS (2008b): OASIS Web Services Distributed Management (WSDM) TC. http://www.oasis-open.org/committees/wsdm/ (19.09.2008)

OASIS (2008c): Web Services Business Process Execution Language (WS-BPEL). http://docs.oasis-open.org/wsbpel/2.0/process/executable (19.09.2008)

Open-LDAP (2008): Open-LDAP, Main Page. http://www.openldap.org/. (19.09.2008)

Rautenstrauch, C.; Schulze, T. (2003): Informatik für Wirtschaftswissenschaftler und Wirtschaftsinformatiker. Berlin u. a.

Rosemann, M. (1995): Erstellung und Integration von Prozeßmodellen – Methoden-spezifische Gestaltungsempfehlungen für die Informationsmodellierung. Dissertation, Westfälische-Wilhelms-Universität. Münster.

Schneider, C. (2007): eGovernment-Integration - Konzeption einer serviceorientierten Integrationsarchitektur zur Digitalisierung von Verwaltungsprozessen. Dissertation, Julius-Maximilians-Universität Würzburg. München.

Tanenbaum, A. S. (2003): Computernetzwerke. 4. Aufl., München u. a.

UDDI (2008a): UDDI Version 3.0.2. http://uddi.org/pubs/uddi_v3.htm (19.09.2008)

W3C (2008a): Web Services Activity. www.w3.org/2002/ws/ (19.09.2008)

W3C (2008b): Web Services Description Language (WSDL) Version 2.0. http://www.w3.org/TR/wsdl20/ (19.09.2008)

Abschließende Erklärung

Ich versichere hiermit, daß ich die vorliegende Diplomarbeit selbständig, ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Magdeburg, den 30. September 2008