Projektgruppe BundOnline

Musterarchitektur BundOnline
Grundlagen der Integration

Version 1.0
07.03.2005

Das vorliegende Dokument wurde durch die Projektgruppe BundOnline in Zusammenarbeit mit der Firma CSC Ploenzke AG erstellt.

Ansprechpartner:
Dr. Heike Stach
Bundesministerium des Innern
Zentrale Koordination BundOnline

eMail: heike.stach@bmi.bund.de

Inhaltsverzeichnis

1 Management Summary 5

2 Zielsetzung 6

3 Grundmuster der Komponentenintegration 8
3.1 Integrationsgrundmuster 8
3.2 Integrationsebenen 9
3.3 Serviceorientierung 12
3.4 Serviceintegration 15
3.4.1 Serviceintegration auf GUI/Service-Ebene (Integrationsebene 2) 15
3.4.2 Serviceintegration auf Service/Service-Ebene (Integrationsebene 3) 16

4 Abkürzungsverzeichnis 21

5 Referenzierte Dokumente 22

Abbildungsverzeichnis

Abbildung 1: Schichtenmodell eines Systems 8
Abbildung 2: Integrationsmöglichkeiten zwischen Systemen 10
Abbildung 3: Service als Aggregat von Schnittstellen 14
Abbildung 4: Darstellung Service 15
Abbildung 5: Serviceintegration auf Ebene 2 (GUI/Service Integration) .. 15
Abbildung 6: Serviceintegration auf Ebene 3 (Service/Service Integration) 16
Abbildung 7: Service Manager auf JAVA-Basis 17
Abbildung 8: Service Manager auf MOM-Basis 18
Abbildung 9: Service Manager auf BPMS-Basis 19

1 MANAGEMENT SUMMARY

Die Musterarchitektur BundOnline stellt eine idealtypische Architektur für die Implementierung von Online-Dienstleistungen in der Bundesverwaltung dar. Sie beschreibt, wie die BundOnline Basiskomponenten technisch in Online-Dienstleistungen eines bestimmten Typs eingebunden werden können.[1] Die Musterarchitektur muss an die spezifischen z.B. organisatorischen und betrieblichen Gegebenheiten in einem Projekt angepasst werden. Sie gibt IT-Verantwortlichen, die eine Dienstleistung im BundOnline Kontext planen oder konzipieren, konkrete Hilfestellungen bei der technischen Konzeption ihres Vorhabens und der Vorbereitung von Ausschreibungen.

Die Musterarchitektur besteht aus mehreren Dokumenten: In diesem Dokument werden die wesentlichen Grundlagen und Grundmuster der Integration, die für alle Dienstleistungstypen gelten, zusammengefasst dargestellt. Die darauf basierende dienstleistungstypabhängige Spezialisierung – z.B. für Antragsverfahren – erfolgt in gesonderten Dokumenten.

In enger Anlehnung an SAGA[2] hält die BundOnline Architektur die Abhängigkeiten der beteiligten Komponenten so gering wie möglich. Durch Verringerung und Vermeidung von Schnittstellen im Rahmen einer konsequenten Serviceorientierung wird die Grundlage geschaffen, um flexible und erweiterbare Online-Dienstleistungen zu realisieren. Die technische und funktionale Skalierbarkeit der Architektur ermöglicht sowohl die Konzeption einfacher als auch komplexer Systeme, die verteilt betrieben werden. Sie bewirkt, dass mit relativ geringem Aufwand Erweiterungen der Funktionalitäten und Anpassungen an veränderte Mengengerüste durchgeführt werden können. Somit wird ein hoher Investitionsschutz sichergestellt. Die lose Kopplung der Komponenten reduziert die Anforderungen an die Verfügbarkeit der einzelnen Komponenten der Online-Dienstleistung. Dadurch wird ein wirtschaftlicher Betrieb einer Online-Dienstleistung begünstigt.

[1] Beschreibung und Dokumentation der BundOnline Basiskomponenten Datensicherheit, Formularserver, Content Management, Zahlungsverkehr und Portal www.bund.de sowie auch der Dienstleistungstypen findet sich im Wissensmanagement BundOnline unter www.wmsbundonline.de

[2] Standards und Architekturen für eGovernment-Anwendungen

2 ZIELSETZUNG

Die Musterarchitektur BundOnline beschreibt das Zusammenspiel von BundOnline Basiskomponenten und der weiteren, notwendigen Komponenten von Dienstleistungen bestimmter Typen (Antragsverfahren, Förderverfahren, ...). Somit ist die Musterarchitektur vor allem eine Integrationsarchitektur, bei der die Basiskomponenten als Black Boxes verstanden werden.

Das Ziel dieses Grundlagendokumentes besteht darin, grundlegende Integrationsmechanismen vorzustellen, die für verschiedene Dienstleistungstypen relevant sind. Dabei werden Begriffe geklärt, die in den Dokumenten zu den einzelnen Dienstleistungstypen verwendet werden. Gleichzeitig werden Empfehlungen gegeben, die die Realisierung zukunftssicherer, wirtschaftlicher Online-Dienstleistungen aus architektonischer Sicht unterstützen.

Grundlegendes Ziel der Musterarchitektur ist, die Dienstleistungen bei der Einbindung der Basiskomponenten und der Konzipierung einer beherrschbaren und wirtschaftlichen Architektur zu unterstützen. Dazu werden in enger Anlehnung an SAGA [1] folgende architektonische Prinzipien befolgt:

- **Serviceorientierung**

Jedes System/Basiskomponente (kurz Komponente) exportiert fachlich orientiert bestimmte funktionale Bereiche (Services) über geeignete, möglichst Standard-basierte Schnittstellen (oder auch Adapter) für die "Außenwelt". Die Services resp. die korrespondierenden Schnittstellen sind dabei grobkörnig, d.h. nicht jedes technische Detail ist exportwürdig, nur sinnvolle Gesamtzusammenhänge werden nach außen gelegt. Dieser Ansatz stellt den in der Integrationstheorie geforderten Black Box-Charakter einer Komponente sicher.

- **Lose Kopplung der Services,**

über z.B. Servicedirectories und asynchrone Kommunikation verringern gegenseitige Abhängigkeiten der Komponenten untereinander, räumlich verteilte Betriebsszenarien der einzelnen Komponenten werden ermöglicht. Durch die asynchrone lose Kopplung der Systeme untereinander, ist es für das Zusammenspiel des Gesamtverbundes nicht mehr zwingend notwendig, dass alle beteiligten Services zur gleichen Zeit verfügbar sind. Kosten für Hochverfügbarkeitslösungen (Hardware, Software u. Betrieb), die für die Zielerreichung konstant verfügbarer Services im Sinne des konzertierten Zusammenspiels anfallen würden, lassen sich durch lose, asynchrone Kopplungen weitgehend vermeiden.

- **Flexibilität der Integrationsstrukturen**

Flexible Integrationsstrukturen auf der Basis der Serviceorientierung und loser Kopplung ermöglichen die einfache Erweiterung der Gesamtfunktionalität durch neue Services. Hierzu ist es wichtig, dedizierte Integrationskomponenten (vgl. auch SW-Referenzarchitektur, SAGA 2.0 [1]) einzuführen.

- **Beachtung relevanter Standards**

Durch die Unterstützung offener moderner Standards im Bereich der Integration wird eine breite Toolunterstützung und ein maximaler Investitionsschutz sichergestellt. Für BundOnline sollte hier immer SAGA [1] die Referenz für Standards sein. Ein wesentlicher Standard in diesem Kontext ist der Web Service-Standard.

- **Skalierbarkeit**

Eine stabile Gesamtarchitektur zeichnet sich durch Skalierbarkeit aus, d.h. dass die dargestellten Konzepte eine funktionale Erweiterung (z.B. durch zusätzliche Integration) ausdrücklich unterstützen. Weiterhin sollten sowohl kleinere Umsetzungsszenarien als auch große, Plattformorientierte Ansätze mit hohem Datendurchsatz architektonisch unterstützt werden.

Die Musterarchitektur soll Systemarchitekten und IT-Verantwortlichen, die eine Dienstleistung im BundOnline Kontext planen oder konzipieren wollen, konkrete Hilfestellungen bei der Umsetzung dieser Prinzipien in ihrem Vorhaben liefern. Dabei wird das Zusammenspiel aller Kompo-

nenten von Dienstleistungen eines Typs aus verschiedenen Blickwinkeln dargestellt. Diese Blickwinkel (Viewpoints) orientieren sich an dem ISO-Architekturstandard RM-OPD[3] (Reference Model of Open Distributed Processing). Die allgemeingültigen RM-ODP-Viewpoints wurden in SAGA 2.0 [1] auf die Belange des eGovernment hin konkretisiert.

[3] ISO/IEC 10746

3 GRUNDMUSTER DER KOMPONENTENINTEGRATION

3.1 Integrationsgrundmuster

Im Rahmen einer Online-Dienstleistung wirken viele Systeme[4] koordiniert zusammen. Eine wesentliche Aufgabenstellung beim Design einer Online-Dienstleistung ist die Konzeption der Integration der verschiedenen Systeme. Bei der softwaretechnischen Integration können verschiedene Ebenen unterschieden werden. Bevor nun im Einzelnen auf die verschiedenen Integrationsebenen eingegangen wird, erfolgt eine kurze Darstellung des Schichtenmodells eines Systems.

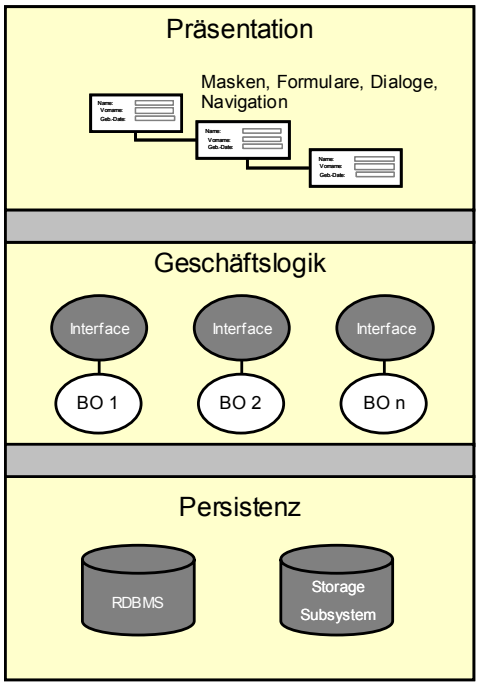

Abbildung 1: Schichtenmodell eines Systems

Grundsätzlich kann wie in Abbildung 1 dargestellt ein modernes DV-System in drei logische Schichten unterteilt werden.

Die **Präsentationsschicht** stellt das Benutzerinterface (User Interface, UI) des Systems bereit. Hier werden die Eingabemasken, Formulare, Dialoge sowie die Navigation und Menüführung des Systems vorgehalten. Die Präsentationsschicht eines modernen DV-Systems kann als so genannter Complex Client[5] z.B. als Java-Swing-Applikation ausgeführt sein, der auf einem PC installiert ist und der sich i.d.R. der Möglichkeiten moderner Client-Betriebssysteme bedient, wie etwa Drag und Drop oder Kontext-sensitiver Menüs.

[4] BundOnline Basiskomponenten, Fachverfahren, Vorgangsbearbeitungssysteme

[5] alternativ auch Fat Client, Rich Client

Eine weitere Ausprägung der Präsentationsschicht, gerade für den Online-Teil einer Dienstleistung relevant, stellt der Web Client[6] dar. Hierbei ist die logische Schicht Präsentation in die zwei physische Schichten - die Client Side Präsentation und die Server Side Präsentation – aufgeteilt. Im Rahmen der Client Side Präsentation wir das Benutzerinterface über einen Web Browser dem Benutzer in Form von HTML-Seiten dargestellt. Die Erzeugung der HTML-Seiten sowie die Steuerung der Navigation erfolgt innerhalb der Schicht Server Side Präsentation.

Neben den Complex Client und Web Client Ausprägungen einer Präsentationsschicht existieren in vielen Organisationen auch noch eine Reihe ältere Host-Systeme, die i.d.R. ihr Zeichen-orientiertes UI via Terminalemulation (z.B. EHLLAPI) dem Benutzer als Präsentationsschicht zur Verfügung stellen.

Die Elemente der Präsentationsschicht kommunizieren in geeigneter Form mit den Interfaces der **Geschäftslogik** (API). Die Geschäftslogik beinhaltet Geschäftsobjekte und Geschäftsregeln (hier zusammengefasst zu Business Objects, BO). Geschäftsobjekte repräsentieren zusammengefasste Datenobjekte der Persistenzschicht, deren dynamisches Zusammenwirken durch Geschäftsregeln abgebildet wird. Die Schicht der Geschäftslogik kann physisch zusammen mit den Elementen eines Complex Clients auf einem Benutzer-PC oder, bei moderneren Systemen, auf zentralen Servern beherbergt sein. Die technische Ausprägung der Geschäftslogik kann sehr unterschiedlich sein. So können hier z.B.: CORBA-Objekte, J2EE-Enterprise Java Beans, .NET-Objekte, via RPC propagierte C/C++ Objekte, COM-Objekte oder proprietäre Datenbank-Programm-Objekte zum Einsatz kommen.

Die hierzu entsprechenden Interfaces der Schicht Geschäftslogik sind in vielen Systemen bzgl. ihres strukturellen Aufbaus exakt und ausschließlich für die Bedürfnisse der Elemente der Präsentationsschicht des Systems zugeschnitten.

Die **Persistenz**-Schicht eines modernen Systems beinhaltet i.d.R. ein relationales Datenbank Managementsystem (RDBMS) zur Speicherung der BO. Daneben können hier gerade für VBS-/Archivsysteme weitere Storage Subsysteme (z.B. optische Plattensysteme) existieren. Die Persistenz-Schicht liegt physisch i.d.R. auf einem zentralen Serversystem.

3.2 Integrationsebenen

Die softwaretechnische Integration zwischen Systemen erfolgt in der Praxis auf vier Ebenen.

[6] Thin Client

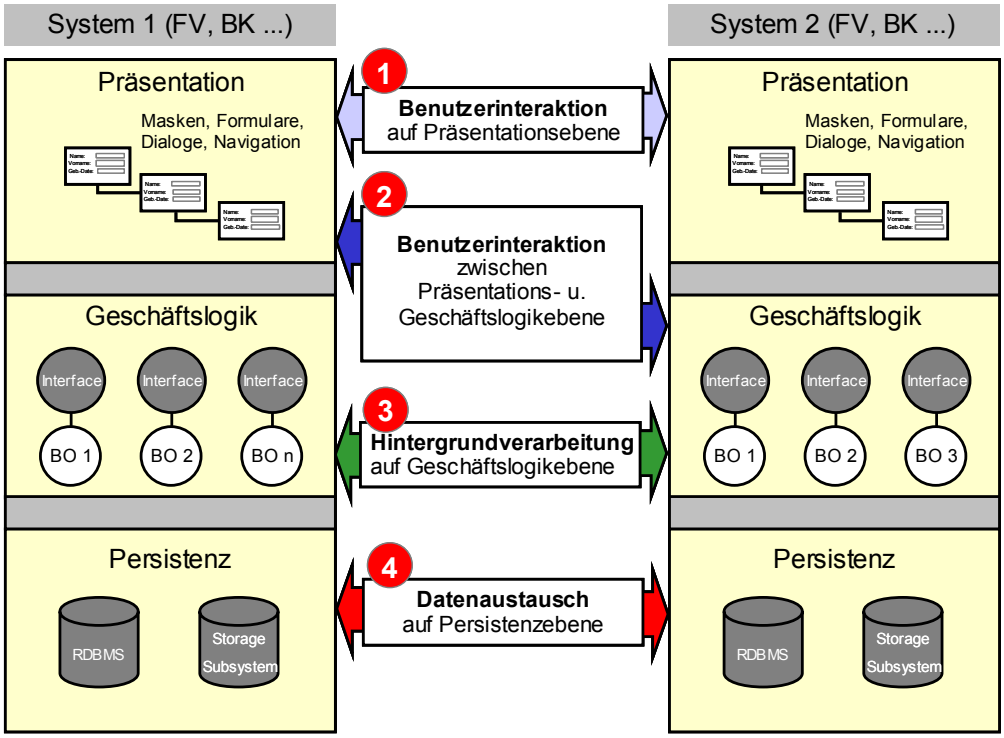

Abbildung 2: Integrationsmöglichkeiten zwischen Systemen

Integrationsebene 1:

Integration durch Benutzerinteraktion auf Präsentationsebene zwischen 2 Systemen.

Indikationen:

- Keine andere technische Kopplungsmöglichkeit bei Einbezug von Altsystemen (z.B. Integration Host-Verfahren via Terminalemulation)
- Ergonomische Notwendigkeit (Benutzer müsste ansonsten im Rahmen eines UseCases kleinteilige Arbeitsschritte in unterschiedlichen Systemumgebungen durchführen)
- Verschiedene Informationen werden auf einen Blick benötigt

Technische Möglichkeiten, z.B.:

- Komposition neuer, integrierter UI auf der Basis von Complex Client Komponenten, wie OCX-Controls oder Swing-basierter Java Beans
- Steuerung/Navigation einer Präsentationsschicht durch eine andere, z.B. durch EHLLAPI oder OLE-Automation.
- Portaltechnologie (z.B. via Portlets)

Einige der hier beschriebenen technischen Möglichkeiten dürfen aber nur auf dieser Integrationsebene verwendet werden. Die Verwendung von OCX-Controls bei einer Online-Kommunikation in der Integrationsebene 2 wäre beispielsweise ein Verstoß gegen SAGA-Vorgaben [1].

Bei der Integration von BundOnline2005-Basiskomponenten spielt diese Integrationsebene nur eine untergeordnete Rolle, so dass hier auf die Nennung weitere Details verzichtet wird.

Integrationsebene 2:

Integration durch Benutzerinteraktion zwischen der Präsentationsebene von System 1 und Geschäftslogikebene von System 2

Indikationen:

- Erweiterung des funktionalen Umfangs von System 1
- Initiieren von Prozessen in System 2 durch System 1 mit manueller Freigabe

Technische Möglichkeiten, z.B.:

- SOAP[7]

Für eine Kommunikation in einem organisationsinternen Netzwerk (LAN) außerdem:

- JMS
- RMI[8]

Es gibt weitere technische Möglichkeiten, die allerdings potentiell einen Verstoß gegen SAGA-Konformitäten darstellen können und daher nur nach eingehender Prüfung verwendet werden dürfen (z.B.: TCP/IP-Sockets, RPC, DCOM). Diese haben für die Integration von BundOnline-Basiskomponenten keine Bedeutung.

Integrationsebene 3:

Integration zwischen den Geschäftslogikebenen von System 1 und System 2

Indikationen:

- Bidirektionaler Datenaustausch zwischen Systemen
- Weiterleitung von Prozessaktivitäten
- Hintergrundverarbeitung ohne Nutzerinteraktion

Technische Möglichkeiten, z.B.:

---

[7] SOAP ist aufgrund der XML-basierten statuslosen Kommunikation per HTTP insbesondere bei der Kommunikation über zahlreiche Firewalls (organisationsübergreifend, WAN) geeignet.

[8] Firewalls können die statusbehaftete Kommunikation per RMI verhindern (z.B. Portsperren, Paketfilterung), daher ist dieses Protokoll im WAN (durch organisationsfremde Firewalls) ungeeignet. Daneben existiert keine garantierte Systemverfügbarkeit bei organisationsübergreifender Kommunikation, was eine synchrone Kommunikation via RMI erschwert. Eine Paketfilterung und Portsperren können auch die Kommunikation per JMS erschweren.

Für eine Kommunikation im LAN gilt, dass ggf. vorhandene von der Kommunikation betroffene Firewalls entsprechend zu konfigurieren sind. Andernfalls sollte auch hier auf RMI bzw. JMS verzichtet werden.

- SOAP[9]

Für eine Kommunikation in einem organisationsinternen Netzwerk (LAN) außerdem:

- JMS
- RMI-IIOP
- RMI[10]

Auch auf dieser Integrationsebene gibt es weitere technische Möglichkeiten, die in Bezug auf die Musterarchitektur nicht zu empfehlen sind, da sie nicht den SAGA-Standards entsprechen. Diese sind daher innerhalb von Online-Dienstleistungen zu vermeiden (z.B. Datenaustausch auf Dateiebene, TCP/IP-Sockets, RPC, DCOM) und haben für die Integration von BundOnline-Basiskomponenten keine Bedeutung.

Integrationsebene 4:

Die Integration auf Ebene der Persistenz zwischen System 1 und System 2

Indikationen:

- Bidirektionaler Datenaustausch zwischen Systemen
- Hintergrundverarbeitung ohne Nutzerinteraktion
- Statische Datensicht ohne Geschäftsobjekt-Kontext

Technische Möglichkeiten:

- JDBC

Diese Integrationsebene spielt im BundOnline 2005-Kontext keine Rolle. Denn ein Zugriff auf Ebene der RDBMS birgt die Gefahr von Inkonsistenzen, da unbekannt ist, wann in einem Fremd-System die Persistierung erfolgt. Außerdem würde dieser Zugriff detaillierte Kenntnis des jeweiligen Fremd-Datenmodells voraussetzen und eine starke Abhängigkeit zwischen beiden betrachteten Systemen schaffen. Eine derartige Integration steht also im Widerspruch zur losen Kopplung von Systemen.

Im Rahmen einer Online-Dienstleistung sind aus den genannten Gründen die Integrationsebenen 2 und 3 zu bevorzugen. Daher werden diese im Folgenden detailliert dargestellt.

3.3 Serviceorientierung

Wie in Kap. 3.1 dargestellt, kommunizieren in einer 3-schichtigen Softwarearchitektur die Elemente der Präsentationsschicht in geeigneter Form mit den Interfaces der **Geschäftslogik** über ein Application Programming Interface (API). Die APIs der Schicht Geschäftslogik sind in vielen Systemen bzgl. ihres strukturellen Aufbaus exakt und ausschließlich für die Bedürfnisse der E-

[9] s. 7

[10] s. 8

lemente der Präsentationsschicht des eigenen Systems zugeschnitten. I.d.R. sind diese APIs strukturell feingranular aufgebaut, so dass eine fachlich hoch detaillierte und ergonomische Ablaufsteuerung im Rahmen der zugehörigen Präsentationsschicht des Systems unterstützt wird. Diese hohe strukturelle Granularität wird zum Zwecke der Integration zu anderen Systemen in den meisten Fällen gar nicht benötigt, sie stellt hierfür i.d.R. sogar ein Problem dar. Hohe Granularität muss zum Zwecke der Integration zu andern Systemen oft durch geeignetes Wrapping hin zu grobkörnigen (Teil-)Strukturen angepasst werden. Das hohe Maß an Granularität bedeutet einen hohen Grad der Spezialisierung im Hinblick auf die Kommunikation der Präsentationsschicht mit der Schicht der Geschäftslogik. Diese Spezialisierung führt zu einer großen Abhängigkeit zwischen Präsentationsschicht und Schicht der Geschäftslogik, die eine Integration erschwert. Hier ist eine Generalisierung in grobkörnige (Teil-)Strukturen notwendig, um die Komplexität einer Integration zu reduzieren.

Die technische Ausprägung der API kann, wie in Kap. 3.1 dargestellt, höchst unterschiedlich ausgeführt sein, so dass auf dieser Ebene Interoperabilität zu anderen Systemen oft nicht gegeben ist. Eine Interoperabilität ist daher auf einer API-Basis und damit einer engen Kopplung und starken Abhängigkeit nicht oder nur unter sehr hohem Aufwand zu realisieren.

Als Forderung für die Integrationsebenen 2 und 3 kann zusammenfassend Folgendes abgeleitet werden:

- Grobkörnige fachlich/funktionale Strukturen zum Zwecke der Integration
- Eine möglichst auf Standards basierte technische Form der Bereitstellung dieser Strukturen

Jedes System im Rahmen einer Online-Dienstleistung sollte fachlich orientiert seine funktionalen Bereiche (Services) über geeignete, möglichst Standard-basierte Schnittstellen (oder auch Adapter) für die „Außenwelt“ exportieren. Die Services resp. die korrespondierenden Schnittstellen sind dabei grobkörnig, d.h. nicht jedes technische Detail ist exportwürdig, nur sinnvolle Gesamtzusammenhänge werden nach außen gelegt. Hierdurch lässt sich eine max. mögliche Kapselung der Systeminterna erreichen.

Dieser Ansatz stellt den Black-Box-Charakter einer Komponente sicher. Es wird erreicht, dass bei einer Integration zwischen der Geschäftslogikebene von System 1 und der von System 2 die gegenseitigen Abhängigkeiten minimiert werden. Durch die Verwendung von Standard-basierten Service-Schnittstellen[11] wird eine breite, herstellerunabhängige Toolunterstützung und damit ein entsprechender Investitionsschutz sichergestellt.

Ein Service wird im Rahmen des vorliegenden Architekturdokuments wie folgt definiert:

[11] in Anlehnung an SAGA 2.0 [1] basieren diese beispielsweise auf SOAP und Web Services.

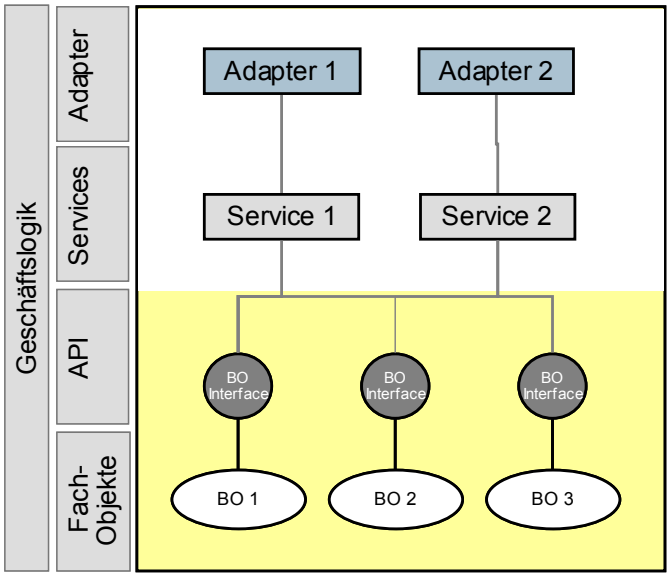

Abbildung 3: Service als Aggregat von Schnittstellen

Wie in Kap. 3.1 dargestellt, befinden sich auf der Ebene der Geschäftslogik diverse Business Objekte, die detailliert und feingranular die fachlichen Zusammenhänge eines Systems technisch abbilden. Business Objekte exportieren Interfaces, die i.d.R. zur Ankopplung an die systemeigene Präsentationsebene vorgesehen sind. Ein Service stellt die fachlich sinnvolle Aggregation mehrerer Business Objekt Interfaces dar und ist logisch wie technisch auch auf Ebene der Geschäftslogik angesiedelt. Konkret können solche Aggregationen mit den Mitteln der Programmier- und Ablaufumgebung des betreffenden Systems umgesetzt werden (z.B. Java, C++), oder auch (z.B. bei älteren Systemen) mit anderen, im Verhältnis zum Altsystem, i.d.R. neueren Techniken.

Ein Service weist einen oder auch mehrere (technisch unterschiedliche) Adapter auf. Ein Adapter stellt letztlich die Schnittstelle eines fachlich/funktionalen Bereiches der Geschäftslogik eines Systems zum Zwecke der Integration zu anderen Systemen bereit. I.d.R. sollte ein komplexes System aus Gründen einer sauberen Strukturierung mehrere Services via Adapter bereitstellen. Die technische Ausgestaltung eines Adapters sollte, in enger Anlehnung an SAGA 2.0 [1], auf der Basis von SOAP/Web Services erfolgen.

Sowohl die strukturelle Konzeption eines Services, als auch die technische Ausgestaltung des entsprechenden Adapters, sollten universellen Charakter haben und wenn möglich, nicht ausschließlich nur einen bestimmten Verwendungszweck fokussieren. Der Adapter eines Services nimmt immer eine passive Rolle ein, d.h. er selbst ruft von sich aus keine Adapter (oder Connectoren, vergl. Ausführungen weiter unten) anderer System auf. Adapter werden immer aufgerufen. Von welchen Komponenten ein Adapter aufgerufen werden kann, wird in Kapitel Serviceintegration erläutert.

Ein Service inkl. Adapter wird im Rahmen des vorliegenden Architekturdokuments wie folgt grafisch dargestellt:

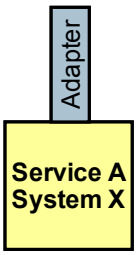

Abbildung 4: Darstellung Service

3.4 Serviceintegration

Die Bereitstellung von Services inkl. Adaptern durch ein System stellt an sich noch keine Integration dar. Services bilden aber die Grundlage der Integrabilität eines Systems für die in Kap. 3.2 dargestellten Integrationsebenen 2 und 3.

3.4.1 Serviceintegration auf GUI/Service-Ebene (Integrationsebene 2)

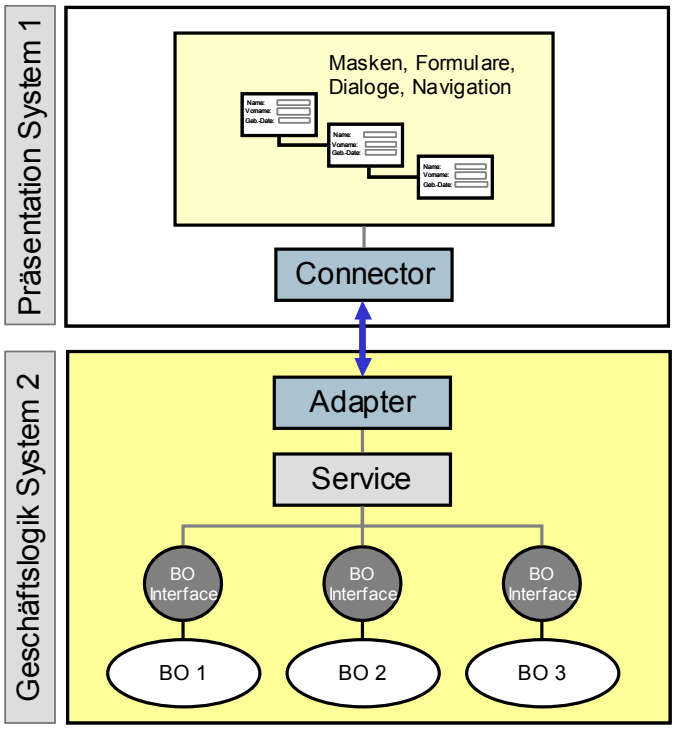

Abbildung 5: Serviceintegration auf Ebene 2 (GUI/Service Integration)

Bei der Serviceintegration auf der Ebene 2 kommuniziert die Präsentationsschicht von System 1 mit der Schicht der Geschäftslogik von System 2. Der Zugriff auf den/die Service Adapter von System 2 sollte in System 1 in einer dedizierten Komponenten strukturell gekapselt sein, um die Abhängigkeiten zu System 2 auf klar begrenzte Bereiche von System 1 zu beschränken. Im Folgenden wird diese Komponente „Connector“ genannt. Grundsätzlich ist der in Abbildung 5 ge-

zeigte Mechanismus auch auf die Anbindung eines GUIs zu einem Service Manager - also zu einer Aggregation von Services - übertragbar.

3.4.2 Serviceintegration auf Service/Service-Ebene (Integrationsebene 3)

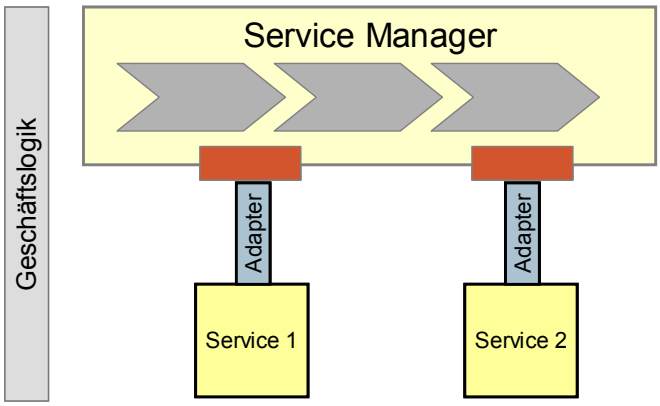

Abbildung 6: Serviceintegration auf Ebene 3 (Service/Service Integration)

Damit die Services zweier Systeme miteinander kommunizieren können, sind dedizierte Integrationskomponenten notwendig, die mittels der Adapter die Services verbinden. Im Gegensatz zu den universellen Services/Adaptern dienen Integrationskomponenten immer ganz speziellen, Projekt-individuellen Integrationszielen. Inhaltlich bewerkstelligen Integrationskomponenten (im Rahmen der Integrationsebene 3 im Folgenden „Service Manager“ genannt) die Steuerung der technischen Aktivitäten (auch technische Mikroprozesse genannt), die zum Datenaustausch zwischen Services mehrerer Systeme notwendig sind. Darüber hinaus können hier auch Datentransformationen und Mappings vorgenommen werden (ggf. kann hierfür auch der Einsatz dedizierter technischer Support Services sinnvoll sein. Hierdurch wird einer Entlastung der Service Manager sowie eine Kapselung erreicht).

Der Service Manager ist die aktive Komponente, die sich mit Hilfe passiver Adapter der Services verschiedener Systeme bedient. Die technische Ausgestaltung der Service Manager kann jeweils sehr unterschiedlich sein. In Abhängigkeit der projektbezogenen, der Organisationsweiten oder sogar der Organisations-übergreifenden Rahmenbedingungen sind hier verschiedenen technische Grundkonzepte zielführend.

Im Folgenden werden drei, in Abhängigkeit der Anforderungslage, mögliche technische Grundstrukturen von Service Managern aufgezeigt.

- Service-Manager auf JAVA-Basis
- Service-Manager auf MOM-Basis
- Service-Manager auf BPMS-Basis

Grundsätzlich sind durch eine strikte Einhaltung der Serviceorientierung auch Migrationen zwischen diesen verschiedenen Service Manager Grundstrukturen mit überschaubarem Aufwand möglich.

a) Service Manager auf Java-Basis

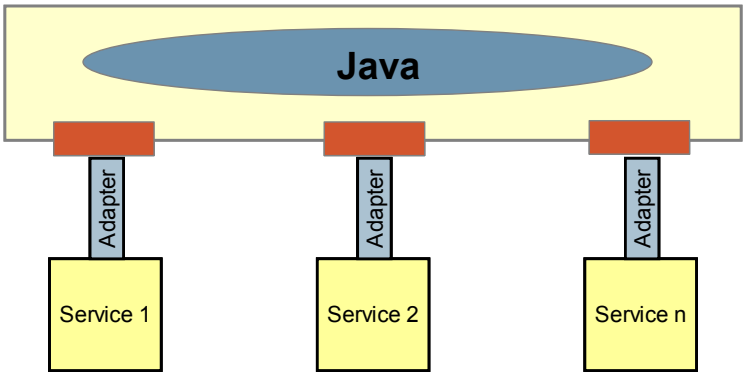

Abbildung 7: Service Manager auf JAVA-Basis

Die Umsetzung der technischen Mikroprozesse eines Service Managers und damit die Integration von Services kann auf Java-Basis realisiert werden. Hierzu bieten sich z.B. einfache Java Tasks an, die periodisch von einem Scheduler (z.B. UNIX Cron Job) gestartet werden. Service Manager auf Java-Basis können auch in Form von Java Servlets umgesetzt werden. Hierbei erfolgt die Aktivierung des Service Managers z.B. über die Schaltfläche einer HTML-Seite. Eine weitere Variante eines Java-basierten Service Managers stellt z.B. die Ausführung als Message Driven Bean im Rahmen eines J2EE-Applikationsservers dar. Für diese Variante steht zur Umsetzung der technischen Mikroprozesse das mächtige Leistungs- und Funktionsspektrum eines J2EE-Applikationsservers zur Verfügung. Die Aktivierung des Service Managers kann also von externen Events getriggert werden.

Die Kopplung der Services erfolgt bei einem fest programmierten, Java-basierten Service Manager synchron. Der Gesamtkopplungsgrad ist hierbei als eher fest zu charakterisieren.

Folgende Anforderungen bzw. Rahmenbedingungen sind Indikatoren[12] für den Einsatz eines Java-basierten Service Managers:

- Die Anzahl der involvierten Services ist gering
- Die zu erwartende Änderungsdynamik des Gesamtverbundes ist gering
- Alle involvierten Services befinden sich an einem Standort (innerhalb eines LANs)[13]

[12] Die Indikatoren sind allgemeingültig nicht exakt quantifizierbar, so dass hier Relation zu den anderen Service Managern bedeutsam ist.

[13] Eine synchrone Java-basierte Kommunikation (RMI) wird dem Überschreiten organisatorischer Grenzen (WAN) nicht gerecht.

b) Service Manager auf MOM-Basis

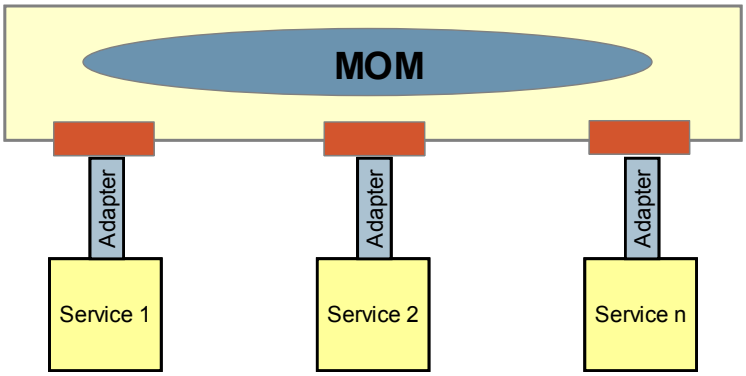

Abbildung 8: Service Manager auf MOM-Basis

Die Umsetzung der technischen Mikroprozesse eines Service Managers und damit die Integration von Services kann auf der Basis von MOM-Systemen (Message Oriented Middleware) erfolgen.

Bei diesem Nachrichten-basierten Verfahren werden Nachrichten in Form von XML-Datensätzen über Queues verarbeitet. Zu einer Queue gehören je ein Message Producer und ein Message Consumer. Message Producer integrieren Adapter von Services, lesen über diese Daten aus den Services, formen sie ggf. in XML um und stellen sie als Nachricht in eine dedizierte Queue. Message Producer können Nachrichten aus Queues entgegennehmen und den Nachrichteninhalt wieder über einen Adapter einem Service zur Verfügung stellen. Die Kommunikationsregeln innerhalb eines MOM-Systems sind konfigurierbar.

Ein MOM-basiertes Verfahren bietet gegenüber der unter a) beschriebenen Variante den Vorteil der asynchronen Kommunikation. Durch asynchrone Serviceintegration wird eine weitere technische und organisatorische Entkopplung erreicht. Gerade bei räumlich verteilten Betriebsszenarien der Services in einem WAN[14] wird hierbei durch den fehlertoleranteren Grundcharakter asynchroner Kommunikation bzgl. temporärer Nichtverfügbarkeit eines Services die effektive mittlere Verfügbarkeit des Integrationsverbundes erheblich gesteigert. D.h., ist durch den Ausfall z.B. einer Netzwerkkomponente im WAN oder durch ein Wartungsfenster bei einem Service-Standort bedingt, ein Service temporär nicht verfügbar, wird die entsprechende Nachricht solange in einer Queue vorgehalten, bis sie bei Verfügbarkeit des Services versendet werden kann. Dieses Standardverhalten eines MOM-Systems kann nur mit erheblichen Aufwänden im Rahmen eines unter a) beschriebenen Service Managers nachprogrammiert werden.

Weiterhin kann auf der Grundlage eines MOM-basierten Service Managers die horizontale Skalierung eines Serviceverbunds[15] realisiert werden. Dies ist z.B. bei hohen Durchsatzanforderungen erforderlich, bei denen ein Service (z.B. zur Kryptografie) gleichzeitig Anfragen von vielen Service Managern oder viele Anfragen von einem Service Manager verarbeiten muss. Eine transparente horizontale Skalierung erreicht man in diesen Fällen, indem man den hoch belasteten Service und die entsprechenden Message Producer und Message Consumer auf mehreren Rechnerknoten instanziert, so dass zur Abarbeitung der Anfragen mehr Kapazität zur Verfügung steht. Diese Maßnahme erfordert (neben der Installation neuer Hardware) lediglich Konfiguration

[14] Herstellerabhängig werden unterschiedliche Protokolle beim Austausch der Nachrichten zwischen Queues angeboten. Bei der Wahl eines geeigneten Produktes ist ggf. die Kommunikation über Firewalls (Portsperren und Paketfilter) zu berücksichtigen.

[15] Ein Serviceverbund stellt mehrerer Instanzen eines Service bereit.

der MOM-Infrastruktur bzgl. des betroffenen Services. Für den restlichen Verbund ist diese Maßnahme transparent.

Die meisten Anbieter von J2EE-Applikationsservern bieten heute MOM-Erweiterungen, z.T. schon im Standardlieferumfang ihrer Produkte auf der Grundlage von JMS, an.

Folgende Anforderungen bzw. Rahmenbedingungen sind Indikatoren[16] für den Einsatz eines MOM-basierten Service Managers:

- Die involvierten Services sind räumlich innerhalb eines WAN verteilt
- Es ist davon auszugehen, dass der Durchsatz des Gesamtsystems oder einzelner Services durch horizontale Skalierung sichergestellt oder im Betrieb erhöht werden muss (z.B. durch steigende Benutzerzahlen)

c) Service Manager auf BPMS-Basis

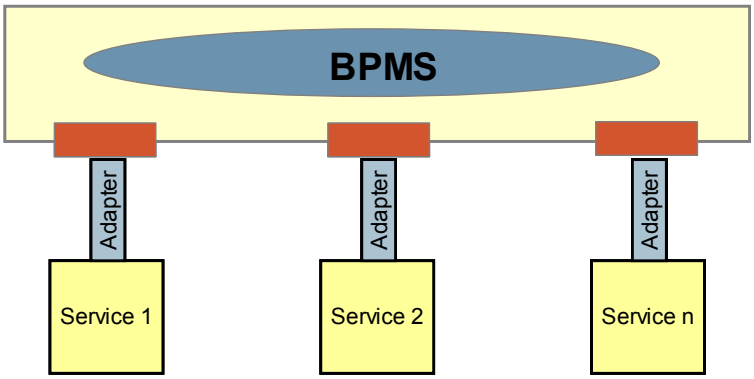

Abbildung 9: Service Manager auf BPMS-Basis

Die Umsetzung der technischen Mikroprozesse eines Service Managers und damit die Integration von Services kann auf der Basis von BPMS (Business Process Management System) erfolgen. Technisch basieren BPMS meist auf MOM oder Workflow-Managementsystemen. Der Fokus eines BPMS liegt auf der Modellierung der Mikroprozesse[17], die zur Integration von Standard-basierten Services/Adaptern benötigt wird. Standards für die Modellierung der Mikroprozesse sind z.B. BPML und BPEL4WS. Service Manager auf der Basis von BPMS gelten als die nächste Evolutionsstufe zu den MOM-basierten Service Managern. Sie „erben“ von ihnen alle Vorteile und Möglichkeiten der asynchronen Kommunikation, die konkrete Ausgestaltung einer Integration von Services ist aber weniger technisch, sondern eher Prozess-orientiert. Hierdurch wird die Integration auf Integrationsebene 3 auf ein fachliches Niveau transferiert, welches diverse weitere Vorteile und Möglichkeiten bietet. So kann, z.B. selbst im Rahmen komplexer Verbünde von Services durch Änderung eines Prozessmodells, einfach und damit schnell und kostengünstig, ein weiterer Service hinzugefügt oder der Informationsfluss im Verbund geändert werden.

[16] s. 12

[17] Ggf. auch der makroskopischen Geschäftsprozesse

Folgende Anforderungen bzw. Rahmenbedingungen sind Indikatoren[18] für den Einsatz eines BPMS-basierten Service Managers:

- Die Anzahl der involvierten Services ist hoch
- Die zu erwartende Änderungsdynamik des Gesamtverbundes ist hoch
- Die involvierten Services sind räumlich innerhalb eines WAN verteilt
- Es ist davon auszugehen, dass der Durchsatz des Gesamtsystems oder einzelner Services durch horizontale Skalierung sichergestellt oder im Betrieb erhöht werden muss (z.B. durch steigende Benutzerzahlen)

Vorgangsbearbeitungssysteme (VBS) befassen sich heute mit der Umsetzung und Abwicklung geschäftsvorfallbezogener Prozesse. Im Rahmen eines integrierten BPMS zur Orchestrierung sowie Steuerung und Ausführung dieser Prozesse ergeben sich unter Nutzung von Serviceorientierung neue Perspektiven: Tendenzen zum Ausbau von VBS zu BPMS oder die Integration von BPMS in die VBS sind bereits bei einigen Standardsoftwareherstellern zu verzeichnen.

Damit könnten VBS mit den ihnen eigenen Workflow-Management-Komponenten in der Zukunft als BPMS basierte Service-Manager die für eine Online-Dienstleistung notwendigen Services integrieren und auf diese Weise über beteiligte Systeme hinweg die Prozesse der einzelnen Dienstleistungstypen steuern.

[18] s. 12

4 ABKÜRZUNGSVERZEICHNIS

BK	Basiskomponente
BO	BusinessObjects
BPM	Business Process Management
BPMS	Business Process Management System
BPML	Business Process Modelling Language
BPEL4WS	Business Process Execution Language for Web Services
CORBA	Common Object Request Broker Architecture
DCOM	Distributed Common Object Model
DL	Dienstleistung
EHLLAPI	Extended High Level Language Application Programming Interface
GUI	Graphical User Interface
IIOP	Internet Inter-ORB Protocol
ISO	International Organsization for Standardization
J2EE	Java 2 Platform; Enterprise Edition
JMS	Java Message Service
JSP	JavaServer Pages
KBSt	Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung
LAN	Local Area Network
MOM	Message Oriented Middleware
OCX	OLE Control Extension
OLE	Object Linking and Embedding
RDBMS	Relationales Datenbank Management System
RMI	Remote Method Invocation
RMI-IIOP	RMI over IIOP
RM-ODP	Reference Model of Open Distributed Processing
RPC	Remote Procedure Call
SAGA	Standards und Architekturen für eGovernment-Anwendungen
SOAP	Simple Object Access Protocol
UI	User Interface
VBS	Vorgangsbearbeitungssystem
WAN	Wide Area Network

5 REFERENZIERTE DOKUMENTE

[1] SAGA, Standards und Architekturen für E-Government Anwendungen, Version 2.0, Schriftenreihe der KBSt, Band 59, Dezember 2003