Bundesverwaltungsamt

IsyFact-Standard

IsyFact –
Technische Architektur eines IT-Systems

Version 1.2
25.04.2016

IsyFact Standards

Bundesverwaltungsamt
Der zentrale Dienstleister des Bundes

„IsyFact – Technische Architektur eines IT-Systems“
des Bundesverwaltungsamts ist lizenziert unter einer
Creative Commons Namensnennung 4.0 International Lizenz.

Ansprechpartner:

Referat Z II 2
Bundesverwaltungsamt
E-Mail: isyfact@bva.bund.de
Internet: www.isyfact.de

Dokumentinformationen

Dokumenten-ID:	IsyFact-Referenzarchitektur-IT-System.docx

Inhaltsverzeichnis

1. Einleitung

Die in diesem Dokument beschriebene technische Architektur eines IT-Systems, erläutert wie ein IT-System der Anwendungslandschaft zu beschreiben und umzusetzen ist.

1.1. Kurzüberblick Referenzarchitektur eines IT-Systems

Dieser Abschnitt gibt einen kurzen Überblick über die wesentlichen Inhalte der technischen Architektur, die in späteren Kapiteln ausführlich dargestellt werden.

Die individuell erstellten Anwendungssysteme der Anwendungslandschaft sollen technisch gleichartig aufgebaut sein. Daher wird in diesem Dokument die Referenzarchitektur definiert (s. Abbildung 1), nach der alle Anwendungssysteme der IsyFact software-technisch in IT-Systeme umgesetzt werden.

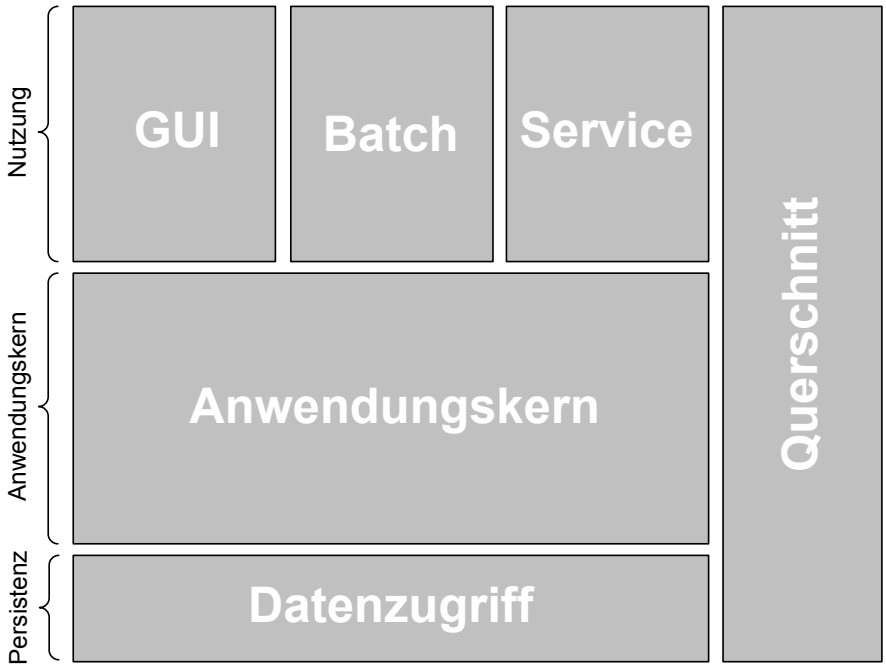

Abbildung 1: Schichten eines IT-Systems

Die Referenzarchitektur für ein IT-System basiert auf der bekannten Drei-Schichten-Architektur. Diese drei Schichten sind:

Persistenz: Diese Schicht enthält alle Funktionalitäten zum Erzeugen, Suchen, Bearbeiten und Löschen von Datenobjekten.

Anwendungskern: Der Anwendungskern umfasst die fachliche Logik zur Datenbearbeitung, die mit dem System realisiert werden soll.

Nutzung: Die Nutzungsschicht ist eine Erweiterung der klassischen Drei-Schichten-Architektur, in der die dritte Schicht als GUI oder

Präsentation bezeichnet wird. In der IsyFact-Referenzarchitektur für ein IT-System wird diese Schicht erweitert um Batch und Service.

Die IsyFact-Referenzarchitektur ist eine vollwertige JEE-Architektur. Jedoch wird eine zentrale Spezifikation von JEE nicht genutzt: die EJB-Spezifikation aus dem Bereich Enterprise Application. Dies hat vor allem Performance- und Komplexitätsgründe. Es hat zur Folge, dass als Application Server ein Servlet-Container ausreichend ist.

2. Technische Architektur

Die technische Architektur beschreibt die Strukturierung der IsyFact-Plattform in IT-Systeme, den Aufbau einzelner IT-Systeme und die verwendeten übergreifenden technischen Konzepte.

2.1. Die Referenzarchitektur eines IT-Systems

Die individuell erstellten Anwendungssysteme der Anwendungslandschaft müssen technisch gleichartig aufgebaut sein. Daher wird im Folgenden die Referenz-Architektur beschrieben, nach der diese Anwendungssysteme software-technisch in IT-Systeme umgesetzt werden.

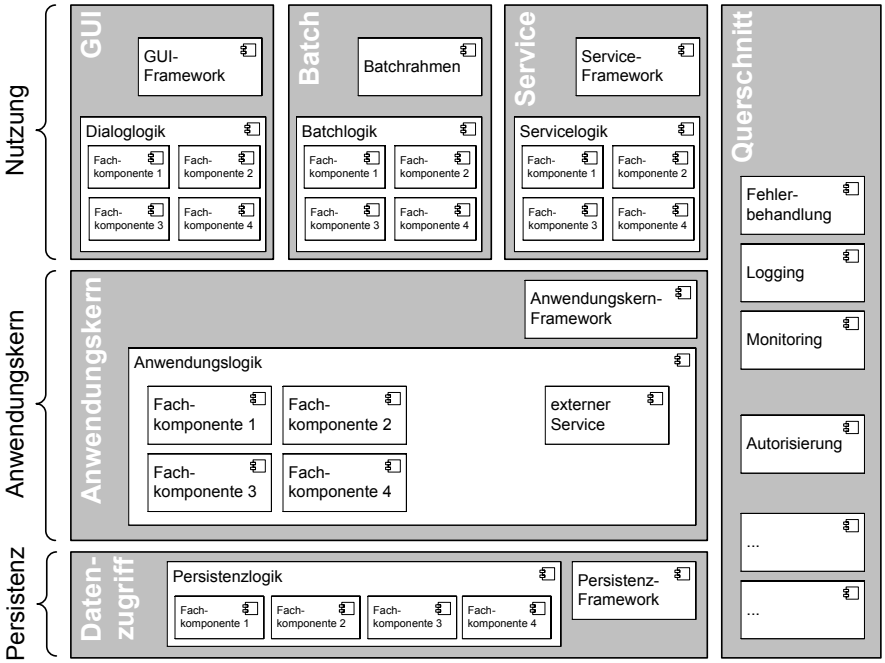

Abbildung 2: Referenzarchitektur eines IT-Systems

Die Referenzarchitektur für ein IT-System basiert auf der bekannten Drei-Schichten-Architektur:

Persistenz: In dieser Schicht ist alle Funktionalität enthalten, die zum Erzeugen, Suchen, Bearbeiten und Löschen von Datenobjekten benötigt wird. Der Datenzugriff auf das DBMS ist in dieser Schicht vollständig gekapselt. In Richtung Datenbank kommuniziert diese Schicht mittels JDBC und SQL, in Richtung des Anwendungskerns stellt die Schicht Geschäftsobjekte zur Verfügung. Die Persistenz nutzt dazu das Persistenzframework Hibernate über die Java Persistence API (JPA).

Anwendungskern: Der Anwendungskern umfasst die fachliche Logik zur Datenbearbeitung, die mit dem System realisiert werden soll. Dazu gehören zum Beispiel Datenvalidierungen oder Funktionen zum Verarbeiten von Geschäftsobjekten, die über die Datenhaltung geladen wurden. Innerhalb dieser Anwendungslogik befinden sich aber nur Funktionalitäten, die allgemein verwendbar sind und die von

mehreren Nutzern genutzt werden könnten. Daneben gibt es spezielle Dialogabläufe, zum Beispiel dialogspezifische Bearbeitungsverfahren wie die Nutzerführung über Wizards. Diese Prozesse, die nur durch einen einzigen Dialog genutzt werden, sind nicht Teil des Anwendungskerns, sondern Teil der Schicht Nutzung. Damit stellt der Anwendungskern innerhalb des Systems allgemein nutzbare Dienste zur Verfügung.

Nutzung: Die Schicht der Nutzung ist eine Erweiterung der klassischen Drei-Schichten Architektur, in der die dritte Schicht als GUI oder Präsentation bezeichnet wird. In der Referenzarchitektur für ein IT-System wird diese Schicht noch zusätzlich erweitert um Batch und Service.

GUI: Aufgabe der GUI ist es, die Funktionalität der Anwendung für einen menschlichen Nutzer zur Verfügung zu stellen. Dabei unterstützt die GUI den Bearbeitungsprozess einer fachlichen Aufgabe. Innerhalb des Bearbeitungsprozesses stellt sie Dialoge dar und transferiert die durch den Anwendungskern gelieferten Datenobjekte in eine präsentable Form. Dies kann sowohl durch eine Web-basierte HTML-GUI als auch durch eine sogenannte Rich-Client Anwendung geschehen.

Batch: Aufgabe des Batches ist es, automatisierte Prozesse ohne einen direkten menschlichen Nutzer durchzuführen. Dabei nutzt der Batch ebenfalls die Funktionen des Anwendungskerns. Auch hier gibt es spezielle Batch-Schritte, die exklusiv vom Batch benötigt werden und deshalb nicht Bestandteil des Anwendungskerns sind. Eine Besonderheit beim Batch ist, dass die Steuerung des Batch-Prozesses in der Regel über ein Scheduling-Werkzeug erfolgt.

Services: Der dritte Nutzer von Anwendungslogik sind andere Anwendungen, die mittels Services auf den Anwendungskern zugreifen. Dieser Zugriff erfolgt nicht direkt auf den Anwendungskern, sondern über die Nutzungsschicht „Service“. Dadurch können auf einfache Art technische Attribute in die Außenschnittstelle aufgenommen werden und der Service des Anwendungskerns kann auf die Kommunikationstechnik abgebildet werden, die für die Kommunikation zwischen IT-Systemen verwendet wird.

Details zur Konstruktion eines IT-Systems werden in [IsyFactSystementwurf] festgelegt und für das jeweilige IT-System dokumentiert.

2.2. Übergreifende technische Konzeption

Für den Aufbau eines IT-Systems gelten einige übergreifende technische Regeln, die im Folgenden vorgestellt werden.

Strukturierung des IT-Systems: Aufgrund der durch die Fachlichkeit festgelegten Funktionalität können IT-Systeme sehr komplex werden. Um diese Komplexität beherrschen zu können, wird das IT-System in Komponenten zerlegt, die jeweils genau festgelegte Teilfunktionalitäten erfüllen. Damit die gesamte geforderte Funktionalität erreicht wird, müssen die einzelnen Komponenten miteinander kommunizieren und sich kennen. Dadurch entsteht ein Komponentengeflecht. Für die Verwaltung der Komponenten innerhalb des IT-Systems und deren Kopplung wird ein Framework verwendet. Details hierzu werden in [AnwendungskernDetailkonzept] festgelegt. Technisch lässt sich ein IT-System in Schichten unterteilen, fachlich in Teilanwendungen. Details hierzu werden in Kapitel 2.3 beschrieben.

Transaktionen: Veränderungen am Datenbestand gehören zur Kernaufgabe von Informationssystemen. Diese Veränderungen können jedoch nicht beliebig erfolgen – sie müssen gewisse Eigenschaften haben, damit die Datenbestände nicht an Qualität verlieren: Sie müssen atomar, konsistent, isoliert und dauerhaft sein. In der Fachliteratur sind diese Eigenschaften als ACID-Eigenschaften (atomicity, consistency, isolation, durability) bekannt.

Änderungen am Datenbestand können komplex sein, das heißt es können viele Informationen und Zusammenhänge von den Änderungen betroffen sein. Solche komplexen Änderungen werden vom Informationssystem meist in einzelne, einfache Änderungsschritte zerlegt. Damit die ACID-Eigenschaften der komplexen Änderung erhalten bleiben, muss das Informationssystem entweder alle Änderungsschritte ausführen – oder keinen. Damit es dazu in der Lage ist, werden die Änderungsschritte zunächst zu einer logischen Einheit zusammengefasst – einer Transaktion.

Für die Referenzarchitektur wird festgelegt, dass die Nutzungsschicht für die Steuerung der Transaktionen zuständig ist. Der Anwendungskern selbst steuert in der Regel keine Transaktionen. Er bietet der nutzenden Schicht eine Schnittstelle zur Transaktionssteuerung an, d. h. der Anwendungskern wird durch die Nutzungsschicht gesteuert.

Zugriffe auf die Datenbank: Bei Zugriffen auf die Datenbank werden die Objekte des Anwendungskerns auf die relationalen Tabellenstrukturen der Datenbank abgebildet. Dabei müssen folgende Aufgaben erfüllt werden:

Objekt-relationales Mapping: die Daten der relationalen Datenbank werden Attributen von Objekten zugeordnet. Dabei werden Datentypen der Datenbank in ihre Objekt-Äquivalente umgesetzt, eine Abbildung von Vererbung im Objektmodell auf relationale Strukturen durchgeführt und Fremdschlüsselbeziehungen

zwischen Tabellen durch Relationen zwischen Objekten abgebildet.

Objektidentität: Wenn ein Datensatz im Verlauf der Bearbeitung mehrmals gelesen wird, so muss dafür gesorgt werden, dass nicht mehrfach gleiche Objekte in einer Anwendung erzeugt werden.

Transaktionssteuerung und Sperren: Es müssen fachliche Transaktionen auf technische Transaktionen gegen die Datenbank abgebildet werden. Dabei wird eine optimistische Sperrstrategie umgesetzt. Ein übliches Verhalten ist zum Beispiel das folgende: Erstreckt sich eine fachliche Transaktion über mehrere Dialogschritte, wird die Zugriffsschicht erst beim Verlassen des letzten Dialogs eine technische Datenbanktransaktion auslösen. Um zu prüfen, ob die Daten in der Zwischenzeit nicht durch einen anderen Nutzer verändert worden sind, wird die Zugriffsschicht beim Schreiben Zeitstempel oder Versionszähler der Datensätze überprüfen.

Um diesen großen Funktionsumfang abzudecken, wird ein Produkt zum objekt-relationalen Mapping gemäß JPA-Standard eingesetzt. Details hierzu sind in [DatenzugriffDetailkonzept] dokumentiert.

2.3. Der Anwendungskern

Zentraler Baustein eines IT-Systems ist der Anwendungskern. Im Anwendungskern werden die fachlichen Funktionen der Anwendung realisiert. Dazu gehören zum Beispiel Datenvalidierungen oder Funktionen zum Verarbeiten von Geschäftsobjekten, die über die Datenhaltung geladen wurden. In Richtung Nutzungsschicht bietet der Anwendungskern nur Funktionalitäten an, die allgemein verwendbar sind und die von mehreren Nutzern verwendet werden könnten. Weitere Details können auch dem Detailkonzept [AnwendungskernDetailkonzept] entnommen werden.

In Abbildung 3 sind drei Aspekte des Anwendungskerns dargestellt. Er besteht aus zwei wesentlichen Bestandteilen: Den Fachkomponenten und dem Anwendungskern-Framework. Weiterhin ist dort dargestellt, dass externe Services als Komponente in den Anwendungskern integriert sind. Diese drei Aspekte werden im Folgenden detailliert beschrieben.

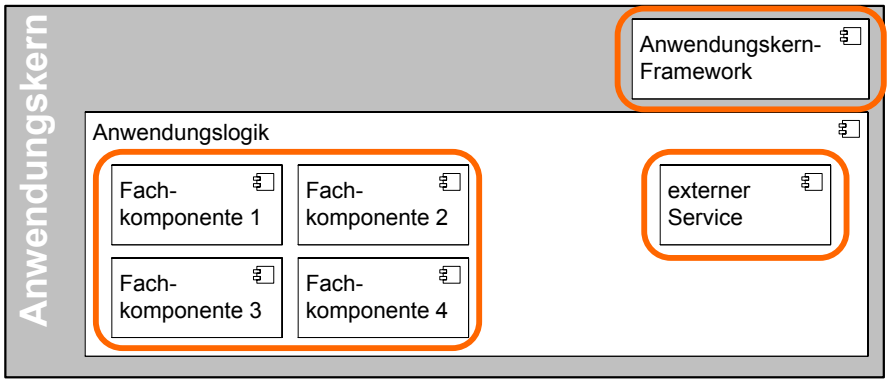

Abbildung 3: Bestandteile des Anwendungskerns

Fachkomponenten: Die Fachkomponenten entsprechen dem fachlichen Komponentenschnitt aus der fachlichen Architektur. Diese Komponenten implementieren weitgehend reine Fachlichkeit und trennen so Anwendungslogik und Technologie. Die Umsetzung einer Komponente aus der fachlichen Architektur erfolgt durch eine Fachkomponente. Dies ist der Schlüssel für gute Wartbarkeit und einfache Weiterentwickelbarkeit des Anwendungskerns. Diese Struktur findet sich auch in anderen Schichten wieder:

In der Persistenz-Schicht sind die Fachkomponenten auch gemäß den Komponenten aus der fachlichen Architektur strukturiert. Die Komponenten des Anwendungskerns besitzen die Datenhoheit auf Objekte, die ihnen eindeutig zuzuordnen sind. Diese Objekte stammen aus den korrespondierenden Komponenten in der Persistenz-Schicht. Nur die korrespondierende Anwendungskern-Komponente darf Änderungen an den entsprechenden persistenten Objekten vornehmen. Die persistenten Objekte dürfen nicht über Komponentengrenzen hinweg herausgegeben werden. In diesem Fall wäre nicht sichergestellt, dass keine Änderungen außerhalb der Komponente passieren. Für den Transfer über Komponentengrenzen hinweg müssen eigene, nicht-persistente Schnittstellen-Objekte erzeugt werden, die dann aus den persistenten Objekten mittels des Bean-Mappers Dozer befüllt werden können.
Wenn Objekte von mehreren Komponenten genutzt werden und keiner einzelnen Komponente zugeordnet werden können, sollten sie in einem eigenen querschnittlichen Package abgelegt werden.

Vereinfachend können Anwendungskern-Komponenten persistente Daten an die Service-Schicht (Kapitel 2.6) weitergeben. Dies ist insbesondere dann angebracht, wenn die Komponente ausschließlich durch eine Service-Komponente genutzt wird.
Da die Service-Schicht keine Logik enthält und daher ohnehin keine Änderung an solchen Entitäten vornehmen darf und eindeutig der Anwendungskern-Komponente zugeordnet ist, bedeutet dies keine Verletzung der Datenhoheit. Da die Anwendungskern-Komponente an Ihrer Schnittstelle nun potentiell persistente und nicht persistente Entitäten bereitstellt müssen Verwechselungen vermieden werden, z.B. in dem solche Methoden in getrennten Interfaces deklariert

werden. Im Zweifel sollte darauf verzichtet werden persistente Entitäten aus der Anwendungskern-Komponente herauszugeben.

Komponenten-Framework: Für querschnittliche Funktionalität innerhalb des Anwendungskerns wird das Spring-Framework genutzt. Hauptaufgabe des Frameworks ist es, die Komponenten zu konfigurieren und miteinander bekannt zu machen. Dadurch wird die Trennung zwischen Fachlichkeit und Technik verbessert. Beispiel für querschnittliche Funktionalität ist die deklarative Steuerung von Transaktionen.

Externe Services: Wenn der Anwendungskern fachliche Services benötigt, die von anderen IT-Systemen innerhalb der Plattform angeboten werden, so werden diese Services als Komponente im Anwendungskern abgebildet. Dadurch ist die Funktionalität sauber gekapselt, was die Wartbarkeit erhöht. Wenn der externe Service ausgetauscht werden soll, ist keine Änderung der gesamten Anwendung notwendig – es ist lediglich eine interne Änderung der externen Service-Komponente notwendig. Für andere fachliche Komponenten des Anwendungskerns ist nicht zu unterscheiden, ob es sich beim Aufruf einer Komponentenschnittstelle um eine in dieser Komponente implementierte Funktion oder um einen Serviceaufruf handelt. Komponenten, die externe Services kapseln, sind im Idealfall von außen nicht von fachlichen Komponenten des Anwendungskerns unterscheidbar. Diese Komponenten haben damit zwei Hauptaufgaben: Sie müssen die technische Aspekte der Kommunikation umsetzen und sie müssen Schnittstellendaten und Exceptions der aufgerufenen Services in die Datenformate der Anwendung transformieren.

2.4. GUI

Bei graphischen Benutzeroberflächen (engl. Graphical User Interface, GUI) gibt es eine Vielfalt unterschiedlichster Komplexitäten: von einfachen Stammdatensystemen über Dialogsysteme mit vielen einfachen Dialogen, die aber intensiv miteinander interagieren, bis zu Clients mit wenigen, aber sehr komplexen Dialogen. Eine gute Architektur muss für alle relevanten Varianten einen tragfähigen Rahmen schaffen.

Im Wesentlichen müssen innerhalb einer graphischen Benutzeroberfläche verschiedene Aufgaben erledigt werden:

- Die Masken und Informationen müssen am Bildschirm angezeigt werden.
- Der Dialog muss auf Benutzerinteraktionen reagieren. Die Validierung von Eingabewerten erfolgt in der Regel aber im Anwendungskern.
- Einzelne Dialoge müssen ggf. zu Dialogabläufen zusammengefasst werden und benötigen Kontext-Informationen

wie den aktuell angemeldeten Benutzer. Die Dialogabläufe bilden einen Workflow. Dieser ist in der Regel in der Dialogsteuerung abgebildet, er kann auch durch eine Workflow-Komponente gesteuert werden.

- Der Dialog muss direkt mit dem Anwendungskern kommunizieren, um Daten zu lesen, die veränderten Daten zu speichern und komplexere fachliche Funktionen auszuführen.

Die in Abbildung 4 dargestellte GUI-Architektur besteht aus 4 Komponenten, die diese wesentlichen Aufgaben übernehmen. Diese Komponenten sind der Dialograhmen, die Dialoge, die GUI-Bibliothek und die Dialoglogik.

Dialograhmen: Die Komponente Dialograhmen definiert die Ablaufumgebung für Dialoge. Der Dialograhmen kennt die Dialogabläufe und die notwendigen Kontextinformationen.

Dialog: Aufgaben der Dialogkomponenten sind die Reaktion auf Benutzerinteraktionen und die Datenhaltung des aktuellen Dialogs.

GUI-Bibliothek: Die Komponente GUI-Bibliothek ist in der Lage, Masken am Bildschirm darzustellen und die einzelnen Elemente der Masken mit Informationen zu versorgen.

Dialoglogik: Die Komponente Dialoglogik enthält die vom Dialog benötigten Fachklassen und übernimmt die direkte Kommunikation mit dem Anwendungskern. Liegt der Anwendungskern auf einem Server, so speichert der Anwendungskern auch Login-Informationen, Session-Daten und ähnliches.

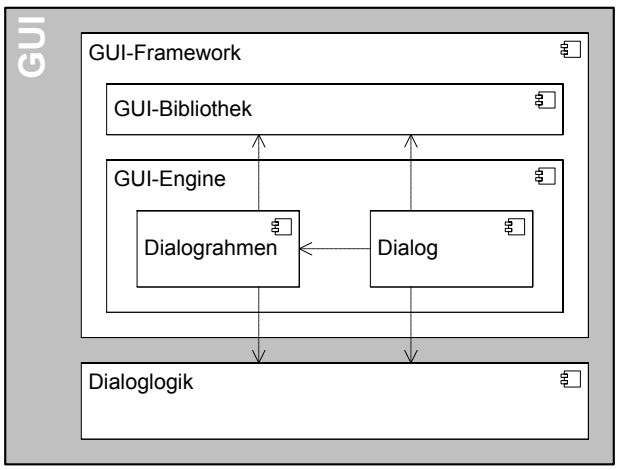

Abbildung 4: Komponenten der GUI-Architektur

Die GUI-Architektur setzt eine Trennung der Dialogsteuerung und des Layouts um. Diese Trennung hat den Vorteil, dass das Layout der Bildschirmmasken bei Bedarf relativ einfach ausgetauscht werden kann. Während der Entwicklung können Spezialisten für das Layout unabhängig von den Spezialisten für die Umsetzung der Dialogsteuerung arbeiten. Für

die Dialogsteuerung von Web-GUIs wird das Framework SpringWebFlow (GUI-Engine) verwendet, für das Layout JSF (GUI-Bibliothek).

Ein am Client durchgeführter Arbeitsprozess besteht in der Regel aus mehr als einem Dialogschritt und damit aus mehr als einem Aufruf des Servers. Dabei sind zum Beispiel folgende Aufgaben zu lösen:

- Über mehrere Dialogschritte muss ein „Gedächtnis“ gehalten werden.
- Ergebnisse von aufwändigen Operationen sollen gecached werden.

Dieser Zustand muss innerhalb der Anwendung abgebildet werden. Hierzu wird ein zustandsloser Server realisiert und der Zustand wird in der Datenbank gehalten. Der Serverprozess selbst hat keinen Zustand. Sobald ein Aufruf durch den Client erfolgt, muss der Server zunächst den aktuellen Zustand rekonstruieren. Dies erfolgt dadurch, dass der Client eine Session-ID übergibt und der Server die benötigten Daten aus einem Datenbank-Zwischenspeicher unter diesem Schlüssel nachschlägt. Mit dieser Lösung lassen sich sehr einfach Loadbalancing- und Failover-Lösungen über Rechner-Cluster realisieren.

Weiterführende technische Details zur GUI sind im Dokument [WebGUIDetailkonzept] enthalten. Vorgaben zum Layout sind in [Styleguide] beschrieben. von Details zur Umsetzung von Berechtigungen sind in [Berechtigungskonzept] enthalten.

2.5. Batch

Ein Batch realisiert eine eigenständige Verarbeitung ohne direkten Benutzereingriff während des Ablaufes. An einen Batch werden verschiedene Anforderungen gestellt: Ausführungszeitpunkt, Abhängigkeiten, Datenvolumen, ausgeführte Funktionalität, Eingaben, Ausgaben usw.

Aus Architektur-Sicht werden diese Anforderungen durch zwei Komponenten abgedeckt: der Batchrahmen und der Batchlogik.

Batchrahmen: Der Batchrahmen stellt die Schnittstelle für den Aufruf der Batchfunktionalität zur Verfügung. Er übernimmt auch die Transaktionssteuerung und die Steuerung für einen Restart.

Batchlogik: Die Batchlogik wird vom Batchrahmen aufgerufen, um die Funktionalität des Batchverarbeitungsprogramms zu aktivieren. Die Funktionalität, das heißt die fachliche Logik und die Arbeitsschritte eines Batches, wird als Anwendungsfälle erfasst. Wenn diese Anwendungsfälle auch von anderen Nutzern benötigt werden, dann sind sie im Anwendungskern implementiert.

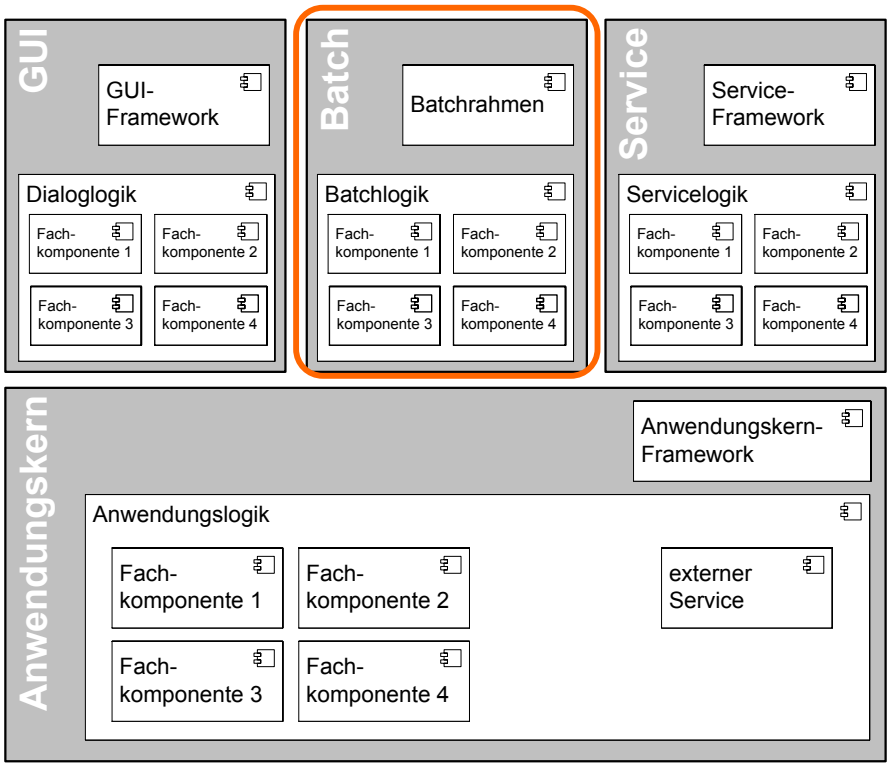

Abbildung 5: Bestandteile von Batchverarbeitungsprogrammen

Batches werden als eigener Prozess auf einem eigenen Server ausgeführt, das heißt sie laufen nicht in der virtuellen Maschine des Application Servers ab. Batches werden somit in einer eigenen Ablaufumgebung ausgeführt und greifen direkt auf die Datenbank oder auch auf Dateien zu. Die benötigte Funktionalität des Anwendungskerns wird dem Batch als Bibliothek zur Verfügung gestellt und nicht über einen Remoteaufruf genutzt. Der Grund für diese Entscheidung liegt in den Datenmengen, die normalerweise von einem Batch verarbeitet werden: Die Übermittlung dieser Datenmengen über eine remote genutzte Schnittstelle ist ein möglicher Flaschenhals in der Anwendung. Dieser Flaschenhals wird durch die Nutzung der Anwendungskernfunktionalität als Bibliothek vermieden.

Batch-Abläufe bestehen aus einem oder mehreren Batch-Schritten. Die einzelnen Batch-Schritte werden von einem Scheduler aufgerufen und zum vollständigen Batch-Ablauf verbunden. Ein Batch-Schritt wird von einem Programm implementiert, das mit entsprechenden Parametern vom Scheduler aufgerufen werden kann. Die hier beschrieben Batches sind genau diese Batch-Schritte.

Die Batch-Schritte haben eine genormte Schnittstelle für Aufruf und Rückgabewerte. Sie sind in der Regel restart-fähig. Es gibt einen Batch-Rahmen, der dies unterstützt.

Weiterführende technische Details zum Batch sind im Dokument [BatchDetailkonzept] enthalten.

2.6. Servicezugriffe

Services des Anwendungskerns, die vom IT-System zur Verfügung gestellt werden sollen, werden durch die Komponenten von „Service“ innerhalb der Schicht „Nutzung“ angereichert und nach außen gegeben. Dabei können alle Dienste des Anwendungskerns genutzt werden. Die Service-Komponenten werden entsprechend den Anwendungskern-Komponente geschnitten, d.h. für jede Komponente des Kerns, die einen Service anbieten soll, wird eine eigene Service-Komponente implementiert. Service-Komponenten werden nicht mehrere Anwendungskern-Komponenten. Dies würde dem Gebot, in den Services keine Logik zu implementieren widersprechen.

Der Aufbau der Schicht Service ist in Abbildung 6 dargestellt. Intern besteht diese Schicht aus zwei Komponenten, dem Service-Framework und der Service-Logik.

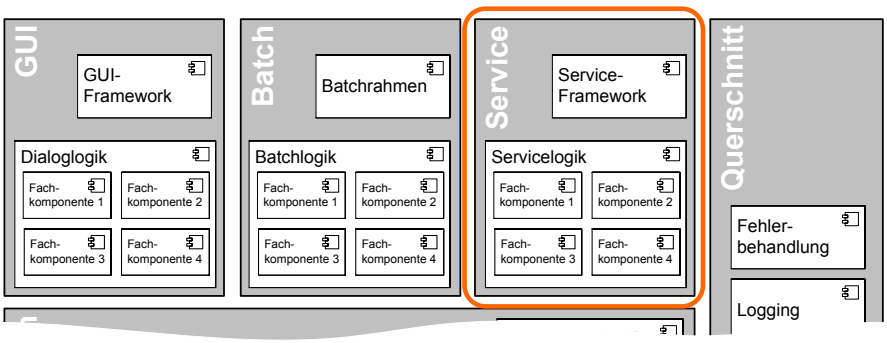

Abbildung 6: Die Bestandteile von Services

Service-Framework: Das Service-Framework dient als Kapsel für die Technologie, mit der die Services des Anwendungskerns zur Verfügung gestellt werden. Hierfür wird das Framework Spring HTTP-Invoker verwendet. In der Regel wird ein extern angebotener Service noch durch zusätzliche Daten oder Logik ergänzt. Diese werden in der Komponente Service-Logik implementiert.

Service-Logik: Die Komponente Service-Logik enthält Daten und Funktionalität, die für die Bereitstellung des Services relevant sind. In der Service-Logik wird keine Fachlogik implementiert, sie nutzt die Funktionalität des Anwendungskerns, um den Dienst bereitzustellen. Die eigentliche Funktionalität des Dienstes ist also im Anwendungskern implementiert. Die Schnittstelle zwischen den Schichten „Service“ und „Anwendungslogik“ ist daher eine interne Service-Schnittstelle. Eine Kernaufgabe der Service-Logik ist die Umsetzung der internen Datenstrukturen und Exceptions des IT-Systems auf Transportobjekte und Exceptions der Service-Schnittstelle sowie die Autorisierung der Nutzung von angebotenen Services.

2.7. Unterstützung technischer Funktionalitäten

Neben der GUI, Services, Batch, Anwendungskern und Datenhaltung benötigt ein IT-System mehrere querschnittliche Funktionalitäten. Diese querschnittlichen Komponenten sind in jeweils eigenen Dokumenten beschrieben. Eine Übersicht dazu liefert das Dokument [IsyFactEinstieg].

3. Quellenverzeichnis

[AnwendungskernDetailkonzept]

Detailkonzept der Komponente Anwendungskern
10_Blaupausen\technische_Architektur\Detailkonzept_Komponente_Anwendungskern.pdf.

[BatchDetailkonzept]

Detailkonzept Komponente Batch
10_Blaupausen/technische_Architektur/Detailkonzept_Komponente_Batch.pdf.

[Berechtigungskonzept]

Berechtigungskonzept
Muss projektspezifisch erstellt werden.

[DatenzugriffDetailkonzept]

Detailkonzept Komponente Datenzugriff
10_Blaupausen\technische_Architektur\Detailkonzept_Komponente_Datenzugriff.pdf.

[IsyFactEinstieg]

IsyFact-Einstieg
00_Allgemein\IsyFact-Einstieg.pdf.

[IsyFactSystementwurf]

Vorlage für Systementwürfe
40_Methodik\20_Systementwurf\IsyFact-Vorlage_Systementwurf.dot.

[Services]

Detailkonzept Komponente Service
10_Blaupausen\technische_Architektur\Detailkonzept_Komponente_Service.pdf.

[Styleguide]

Styleguide
20_Bausteine\Styleguide\Styleguide.pdf.

[ÜberwachungKonfigKonzept]

Konzept Überwachung und Konfiguration
20_Bausteine\Ueberwachung_Konfiguration\Konzept_Ueberwachung-Konfiguration.pdf.

[WebGUIDetailkonzept]

Detailkonzept Komponente Web-GUI
10_Blaupausen/technische_Architektur/Detailkonzept_Komponente_Web_GUI.pdf.

4. Abbildungsverzeichnis