Bachelorarbeit zum Thema:

Cloud Integration mit Integrationsplattformen (iPaaS) und die Auswirkung agiler Organisation auf die Technologiewahl

zur Erlangung des akademischen Grades

Bachelor of Science

eingereicht im Fachbereich Mathematik, Naturwissenschaften und Datenverarbeitung
an der Technischen Hochschule Mittelhessen

Name: Arda Türktorun

E-Mail: arda.tuerktorun@mnd.thm.de

Matrikelnummer: 5231638

Studiengang: Wirtschaftsinformatik

Referent: Herr Prof. Dr. Frank Kammer

Korreferent: Herr Prof. Dr. Harald Ritz

Semester: Sommersemester 2022

Abgabedatum: 01.08.2022

Abstract

Integrationen von Daten und Anwendungen haben in den letzten Jahren immer mehr an Bedeutung gewonnen. Besonders in den heutigen, von homogenen Systemen geprägten, hybriden IT-Landschaften ist das Thema der (Daten-)Integration unumgänglich, um Informationssilos zwischen geschäftskritischen Anwendungen zu vermeiden. In dieser Bachelorarbeit werden verschiedene Integrationsansätze beschrieben, erläutert und verglichen. Dabei wird von den unterschiedlichen Funktionen für Integrationen bis hin zu den technischen Details heruntergebrochen. Zu diesem Zweck wurden Informationen und Inhalte aus verschiedenster Literatur sowie Anbietern von Integrationslösungen aufgegriffen und unterstützend anhand eines Praxisbeispiel mit der SAP Cloud Platform Integration dargestellt. Als Ergebnis stellt sich heraus, dass traditionelle Enterprise Application Integration Ansätze wie der Enterprise Service Bus zunehmend durch cloudbasierte Integrationsplattformen ergänzt oder ersetzt werden. Integration Platform as a Service bietet eine modernere Plattform zur Integration mittels einer grafischen Bedienoberfläche, leicht konfigurierbaren Adaptern zur Anbindung von Anwendungen und weiteren Vorteilen von Cloudplattformen wie die Skalierbarkeit. Des Weiteren helfen agile Organisationsmuster dabei, die Infrastruktur flexibler zu gestalten, sodass auf Marktanforderungen schnell reagiert werden kann.

Inhaltsverzeichnis

Abbildungsverzeichnis iv

Abkürzungsverzeichnis v

1. Einleitung 1

1.1 Einführung in das Thema 1

1.2 Aufgabenstellung und Zielsetzung 1

1.3 Vorgehensweise 2

2. Grundlagen des Cloud Computing 2

2.1 Servicekategorien für Cloud-Dienste 4

2.2 Infrastructure as a Service (IaaS) 5

2.3 Platform as a Service (PaaS) 6

2.4 Software as a Service (SaaS) 6

2.5 Cloud-Bereitstellungsmodelle 7

3. Integrationsgegenstände und -Merkmale 8

3.1 Prozessintegration 8

3.2 Anwendungsintegration 9

3.3 Datenintegration 9

3.4 Die Integrationsrichtung als Integrationsmerkmal 9

3.5 Die Integrationsreichweite als Integrationsmerkmal 10

4. Enterprise Application Integration 11

4.1 Enterprise Service Bus als Integrationsansatz 13

4.2 Funktionale Bestandteile einer EAI-Lösung 15

5. Hybride Integration – Eine Strategie 18

6. iPaaS – Cloudbasierte Integrationsplattform 20

6.1 Die iPaaS-Architekturvarianten 22

6.2 Schnittstellen zum Datenaustausch (REST, OData und SOAP) 24

6.3 Ein Überblick der Potenziale und Herausforderungen durch iPaaS 28

6.4 Der Vergleich zwischen iPaaS und EAI 29

7. Der Markt für Integration Platform as a Service 31

7.1 Kriterien zur Anbieterwahl 33

8. SAP – Cloud Integration (iPaaS) 35

8.1 SAP Cloud Platform Integration (CPI) 35

8.2 (Daten-)Integration am Beispiel von SAP HCM und SuccessFactors 39

9. Agile Organisation 44

9.1 Grundsätze der Agilität und das agile Manifest 44

9.2 Einführung in Scrum 46

9.3 Agile Integration 47

10. Fazit und Ausblick 48

Literaturverzeichnis 50

Abbildungsverzeichnis

Abbildung 1 – Multi-Tenant-Architektur....3

Abbildung 2 – Überblick Cloud-Services....5

Abbildung 3 – Integrationsreichweiten im Überblick....10

Abbildung 4 – Hub and Spoke Ansatz....12

Abbildung 5 – iPaaS Integrationsansatz....20

Abbildung 6 – Überblick der iPaaS Architekturen....23

Abbildung 7 – Aufbau einer SOAP Nachricht....27

Abbildung 8 – Gartner Magic Quadrant iPaaS....32

Abbildung 9 – Discover CPI....37

Abbildung 10 – Design CPI....37

Abbildung 11 – Monitor CPI....38

Abbildung 12 – Funktionsbaustein im SE 80 Paket Explorer....39

Abbildung 13 – ABAP Ausschnitt eines Select-Befehls....40

Abbildung 14 – Service Definition 40

Abbildung 15 – iFlow Kostenstellen nach SF....41

Abbildung 16 – SOAP Adapterkonfiguration....41

Abbildung 17 – Quellformat aus SAP-HCM....42

Abbildung 18 – Strukturanpassung mit Groovy....43

Abbildung 19 – Adapter für Upload nach SuccessFactors....43

Abbildung 20 – Finale Dateistruktur nach dem Prozess....44

Abkürzungsverzeichnis

ABAP	-	Advanced Business Application Coding
API	-	Application Programming Interface
CPI	-	Cloud Platform Integration
CRM	-	Customer Relationship Management
EAI	-	Enterprise Application Integration
ERP	-	Enterprise Resource Planning
ESB	-	Enterprise Service Bus
HCM	-	Human Capital Management
HTTP	-	Hypertext Transfer Protocol
IaaS	-	Infrastructure as a Service
iFlow	-	Integration Flow
iPaaS	-	Integration Platform as a Service
JSON	-	JavaScript Object Notation
OData	-	Open Data Protocol
PaaS	-	Platform as a Service
REST	-	Representational State Transfer
RPC	-	Remote Procedure Call
SaaS	-	Software as a Service
SF	-	SuccessFactors
SOA	-	Service oriented Architecture
SOAP	-	Simple Object Access Protocol
WSDL	-	Webservice Description Language
XML	-	Extensible Markup Language

1.Einleitung

1.1 Einführung in das Thema

In unserem digitalen Zeitalter hat die Cloud mit ihren vielen Lösungen für verschiedenste Zwecke einen nahezu obligatorischen Wert in der IT eingenommen. Bereits im Jahre 2015 war der Trend deutlich zu sehen. Laut der Bitkom-Untersuchung „Cloud Monitor 2015“ hatten damals bereits 44% der deutschen Unternehmen Cloud-Computing im Einsatz und weitere 24% planten den Einsatz von Cloud-Lösungen.[1] Insgesamt findet eine Verlagerung am Markt von On-Premise zu Cloud-Lösungen statt. Durch stätige Konkurrenz, in der von Leistungsdruck geprägten Wirtschaft und der Geschwindigkeit des dynamischen Marktes mit immer komplexer werdenden Anforderungen, müssen IT-Systeme untereinander kommunizieren, um so eine höhere Effizienz für die immer kürzer werdenden Innovationszyklen zu gewährleisen.

Da einige Lösungen nur als Cloud-Optionen existieren oder für Unternehmen zur Optimierung bestehender Prozesse in Frage kommen, entstehen oftmals hybride IT-Landschaften bestehend aus Eigenentwicklungen, On-Premise-Lösungen und Cloud-Lösungen. Für die bereits angesprochene essenzielle Kommunikation zwischen den Systemen muss eine Integration erfolgen, welche die Prozesse, Anwendungen und insbesondere die Daten eines Systems für andere möglichst automatisiert bereitstellt. Cloudbasierte Integrationsplattformen (Integration Platform as a Service) ermöglichen durch den Austausch von Daten zwischen Cloud- oder On-Premise-Anwendungen systemübergreifende Integrationen.[2] Es geht nämlich meist um die Kombination neuer Technologien mit bestehenden Prozessen und Daten, sowie die Synchronisation der Daten aus verschiedenen Domänen.[3] Um diesen schnellen hybriden Modellen gerecht zu werden, stellen viele Unternehmen auch ihre Organisation parallel zur IT-Infrastruktur um. So werden vermehrt agile Organisationmuster implementiert, um sich dem dynamischen Markt anzupassen und schnell auf Anforderungen zu reagieren.[4]

1.2 Aufgabenstellung und Zielsetzung

Die vorliegende Bachelorarbeit soll aufzeigen, wie iPaaS-Lösungen, auch im Vergleich zu anderen Integrationslösungen, neben Applikationen und Prozessen insbesondere Daten integrieren. Dabei wird ebenfalls betrachtet, was eine hybride Integration ist und

[1] vgl. [Bitk/2015], S.5

[2] vgl. [Seub/2018], S.185

[3] vgl. [Seub/2018], S.177

[4] vgl. [DrKS/2013], S.15

welche Herausforderungen bzw. Mehrwerte unterschiedliche Integrationsmöglichkeiten mit sich bringen. Allgemein soll neben den Technologien für die Integrationen auch ein Überblick über den Markt geschaffen werden.

1.3 Vorgehensweise

Um den Einstieg in das Thema „Cloud Integration“ für den Leser zu erleichtern, wird zunächst auf die Grundlagen des Cloud-Computing mit seinen Servicekategorien eingegangen. Anschließend wird der Begriff Integration mit seinen Gegenständen und Merkmalen erläutert, um so eine Basis für die folgenden Integrationsmöglichkeiten zu schaffen. Speziell die Integrationsvarianten „Enterprise Application Integration (EAI)“ sowie die „Integration Platform as a Service (iPaaS)“ werden anhand von Kriterien miteinander verglichen. Auch die Herausforderungen der hybriden Integration werden thematisiert. Ein wichtiger Aspekt sind die Schnittstellen zum Datenaustausch, welche durch praxisnahe Beispiele erläutert werden. Abschließend folgt die Vorstellung der SAP-Integration Suite und ein eigens entwickeltes Praxisbeispiel der Datenintegration vom SAP Human Capital Management System zu SAP SuccessFactors. Zum Schluss wird die organisatorische Veränderung eines Unternehmens hinsichtlich der Verwendung von Cloud-Technologien behandelt.

2. Grundlagen des Cloud Computing

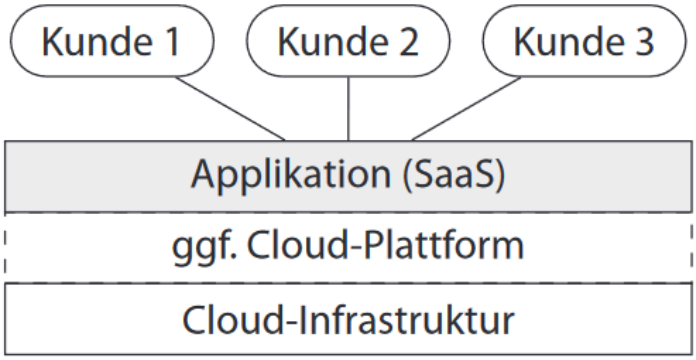
Cloud Computing ist an sich keine neue Technologie, vielmehr handelt es sich dabei um ein neues Geschäftsmodell. Stetige Rechenleistungssteigerungen, die hohe Verfügbarkeit von Breitband und neue Möglichkeiten der Virtualisierung sind nur einige Aspekte, die das Modell zum Erfolg führen.[5] Das National Institute of Standards and Technology (NIST) definiert Cloud Computing als „ein Modell für den allgegenwärtigen, komfortablen und bedarfsgerechten Netzzugang zu einem gemeinsam genutzten Pool konfigurierbarer Rechenressourcen (bspw. Speicher, Anwendungen und Dienste), die mit minimalem Verwaltungsaufwand und ohne Interaktionen mit dem Dienstanbieter schnell bereitgestellt und freigegeben werden können“[6]. So sind die wichtigsten Merkmale des Cloud Computing und Cloud Services zum einen das Zusammenfassen von Ressourcen. Hiermit ist eine Zentralisierung der Ressourcen wie bspw. in Server- und Speicherfarmen gemeint. Aus diesem Pool bedienen sich mehrere Nutzer, was die Auslastung einzelner

[5] vgl. [TeVo/2011], S.23

[6] vgl. [NIST/2011], SP 800-145

Ressourcen steigert und in einer Reduktion des insgesamten Energiebedarfes resultiert. Realisiert wird dies mit der sogenannten Multi-Tenant-Architektur, bei welcher alle Kunden als getrennte Mandanten auf derselben Hardwareinfrastruktur laufen. Es ist die Dimensionierung einer Instanz, in der sich alle Kunden eine Plattform teilen (Abbildung 1).

Abbildung 1 – Multi-Tenant-Architektur



Quelle der Abbildung: [Hase/2012], S.26

Vorteil dieses „one to many" – Systems ist bspw. die Kosteneinsparung bei Updates, da eine Anpassung des Systems für mehrere Kunden gelten und so die Entwicklungszeit und Personalkosten minimiert werden können.[7] Das nächste Merkmal ist die Skalierbarkeit des Ressourcenbedarfes. Falls zu hohe Lasten die Kapazitäten eines Cloud-Systems überschreiten, müssen diese trotzdem abgefangen werden. Auch externe Ressourcen können hierfür verwendet werden. Es ist leichter, weitere logische Ressourcen hinzuzufügen, als die bestehenden Ressourcen aufzurüsten.[8] Selbstbedienung bei Bedarf ist ein weiteres Merkmal, welches beschreibt, dass der Nutzer die benötigte Menge an Ressourcen durch Aktivierung, Reservierung oder Deaktivierung selbst bestimmt und jederzeit ohne den Dienstleister ändern kann. Der nächste Aspekt ist die Flexibilität bei Netzzugriffen. Da Breitbandzugänge bereits in vielen Ländern zur Verfügung stehen, sollte der Zugriff auf die Ressourcen flexibel aus dem lokalen Netzwerk oder Fernbereich gewährleistet sein. Das letzte Merkmal ist die verbrauchsabhängige Verrechnung. Es wird nur so viel gezahlt, wie auch tatsächlich genutzt wird. Durch ausgereifte Messverfahren wird so auch die Preistransparenz gegenüber den Nutzern geboten.[9]

[7] vgl. [TeVo/2011], S.22 f.

[8] vgl. [Hase/2012], S.29

[9] vgl. [TeVo/2011], S.24

Diese Merkmale deuten auf die vielen Vorteile von Cloud Computing und allgemeinen Cloud-Lösungen hin. Neben den niedrigen Kosten durch die Einsparungen von bspw. Energiekosten und Personalkosten gegenüber einer On-Premise Infrastruktur und der hohen Nutzeranzahl, wodurch Anbieter wie Amazon oder Salesforce ihre Preise am Markt halten können[10], sind es die nahezu unbegrenzten Kapazitäten und schnelle Bereitstellung der Ressourcen, welche die Cloud für Unternehmen attraktiv machen. „Cloud-Anbieter können häufig voll funktionsfähige Instanzen innerhalb von Minuten aufbauen“[11]. Doch eine Anschaffung von Cloud-Diensten bringt nicht nur Vorteile mit sich. Gerade Unternehmen müssen sich vor oder auch noch nach der Anschaffung einigen Herausforderungen stellen. Zum einen ist es die Abhängigkeit vom Anbieter. Beim Wechsel von einem Anbieter auf einen anderen können Probleme entstehen und erfordern meist auch zusätzliche organisatorische Kapazitäten. Auch die Integration/Migration auf eine andere Cloudplattform oder die Zusammenarbeit zweier Anbieter können zu Kompatibilitätsstörungen führen.[12] Datenschutz und -sicherheit ist ebenfalls ein nicht zu vernachlässigender kritischer Aspekt, da gerade Unternehmen eine Vielzahl von Vorgaben über die Handhabung mit (sensiblen) Daten erfüllen müssen. Beim Abspeichern von Kundendaten muss sichergestellt werden, dass die Rechenzentren in einer Region sind, in der das Datenschutzgesetzt den Anforderungen entsprechen.[13]

Vor dem Umstieg oder der Einbindung von Clouddiensten müssen deshalb Nutzen und Risiken individuell abgewogen werden.

2.1 Servicekategorien für Cloud-Dienste

Bereits in den frühen Jahren des Cloud-Computings wurde ersichtlich, dass die Bedürfnisse der immer stetig steigenden Anzahl von Kunden nicht durch ein einzelnes Servicemodell universell abgedeckt werden können. Deshalb spezialisierten sich viele Anbieter mit der Zeit auf ein Gebiet wie bspw. Online-Datensicherung oder Cloud-Speicher. Heute werden IT-Leistungen als Services bereitgestellt. Diese werden hierarchisch in drei wesentliche Ebenen unterteilt: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) und Software as a Service (SaaS). In der Abbildung 2 ist ein Überblick der Cloud-Services dargestellt.[14]

[10] vgl. [BöB+/2016], S.24

[11] vgl. [Wood/2015], S.37

[12] vgl. [Hark/2018], Chapter I

[13] vgl. [BöB+/2016], S.25 f.

[14] vgl. [Wood/2015], S.30

Abbildung 2 – Überblick Cloud-Services

Quelle der Abbildung: [TeVo/2011], S.24

2.2 Infrastructure as a Service (IaaS)

IaaS-Dienste dienen zur Bereitstellung von virtualisierten Infrastrukturkomponenten. Es handelt sich um ein Service-Modell, mit dem grundlegende Ressourcen wie Server oder Festplattenspeicher bereitgestellt werden. Trotz dessen, dass es sich hierbei in der Regel um virtualisierte Ressourcen handelt (siehe Kapitel 2 - Multi-Tenant-Architektur), erhält der Nutzer ein identisches Erlebnis wie mit einer vor Ort konfigurierten Infrastruktur.[15] Kunden haben durch das Wegfallen der Anschaffungskosten von Computer-Hardware Einsparungen. Zudem verfügen sie bei der Nutzung des Dienstes über modernste Technologien, die von den Anbietern für die Infrastrukturkomponenten genutzt werden. Auch hier ist für den Ausgleich bei mehr Leistungsbedarf gesorgt, indem externe Ressourcen zugeschaltet werden können. Die Verantwortung der Nutzer liegt bei den Daten, Anwendungen, Services und der Laufzeitumgebung. Das Betriebssystem kann bei virtuellen Maschinen auf Wunsch dazugebucht werden. Für den Betrieb und die Instandhaltung der Server, Virtualisierung, Speicher etc. ist der Provider verantwortlich.[16] Gängige IaaS-Provider sind bspw. Amazon Webservice (AWS), Microsoft Azure (Azure) und die Google Cloud-Plattform.[17]

[15] vgl. [Wood/2015], S.31

[16] vgl. [TeVo/2011], S.25 f.

[17] vgl. [Seub/2018], S.42

2.3 Platform as a Service (PaaS)

Mit einer Platform as a Service werden primär Entwickler als Nutzer angesprochen, die Anwendungen implementieren oder bereitstellen wollen. Neben einer kompletten Entwicklungsumgebung, mit der Anwendungen implementiert werden können, wird auch die Laufzeitumgebung für diese Anwendungen bereitgestellt. So sind für die Entwicklung und den Betrieb der Anwendungen keine Eigenanschaffungen von Hard- oder Software nötig. Eine besondere Eigenschaft von PaaS-Providern stellt der Fakt dar, dass sie dem Anwender meist keinen Zugriff auf das Betriebssystem gewähren. Der Service wird über APIs (Application Programmining Interfaces) oder grafischen Weboberflächen konfiguriert. Von der Entwicklung, dem Testen und der Auslieferung bis hin zum Betrieb der Cloud-Anwendung: All diese Prozesse finden auf der Platform as a Service statt. Um seine eigens entwickelte Cloud-Anwendung zu vermarkten, stellen einige PaaS-Provider einen sogenannten „Marktplatz“ zu Verfügung.

Generell ist eine Unterscheidung oder strikte Trennung von IaaS und PaaS Produkten nicht ganz möglich. Viele PaaS-Provider bieten auch die darunterliegende IaaS an. Aber auch die Nutzung unterschiedlicher Anbieter von IaaS und PaaS ist dank des Multi-Cloud Prinzips möglich. Hierfür wurde die Cloud-Foundry-PaaS, eine Open-Source-PaaS, entwickelt. Durch die standardisierte Architektur für die Umsetzung einer PaaS ist diese bei unterschiedlichen IaaS-Anbietern wie AWS oder Azure lauffähig.[18]

2.4 Software as Service (SaaS)

Bei Software as a Service beziehen die Kunden z.B. Geschäftsanwendungen über das Internet. Dieses Modell beinhaltet Lizenz für die Nutzung der Soft- und Hardware sowie die Pflege und Betrieb der Plattform seitens des Anbieters. Nach der Zahlung ist dieser Service meist sofort verfügbar. Der Anbieter muss neben der Software also auch die Plattform/Infrastruktur bereitstellen und betreiben. So werden die Kunden entlastet und können sich auf ihr Kerngeschäft fokussieren. Ein Nachteil ist jedoch, dass der Kunde meist nicht viel Spielraum für individuelle Anpassungen hat. Doch die Anbieterseitige Wartung der Infrastruktur und Software in Kombination mit geringen Investitionskosten machen dieses Modell für Unternehmen attraktiv. Insbesondere Human Resource (HR) und Customer-Relationship-Management (CRM) Anwendungen sind die am weitesten verbreiteten SaaS-Produkte.[19]

[18] vgl. [Seub/2018], S.41 f.

[19] vgl. [TeVo/2011], S.26

2.5 Cloud-Bereitstellungsmodelle

Die verschiedenen Cloud-Servicemodelle (IaaS, PaaS und SaaS) können laut dem National Institue of Standards and Technology auf drei unterschiedliche Arten bereitgestellt werden. Die Bereitstellungsmodelle lauten Public Cloud, Private Cloud und Hybride Cloud.[20]

In einer Public Cloud bereitgestellte Cloud-Services sind für jeden über das Internet zugänglich. Diese Services laufen alle auf einer gemeinsamen Infrastruktur, wobei die Ressourcen durch die Mandantenfähigkeit (Multi-Tenant-Architektur) auf mehrere Kunden virtuell aufgeteilt werden. Bei Public Clouds besitzt und verwaltet der Anbieter die Software, Hardware und andere Infrastrukturkomponenten. Kunden teilen sich die Infrastruktur (Speicher, Server etc.) untereinander. Ein Nutzer hat also keine dediziert zugewiesene Infrastruktur.[21] Vorteile dieses Bereitstellungsmodells belaufen sich auf die Erhöhung der Skalierbarkeit, welche in einer höheren Kosteneffizienz resultiert. Zudem ist es die hohe Verfügbarkeit und Ausfallsicherheit für den Kunden durch die gemeinsame Nutzung von Ressourcen, da die für diese Zwecke ausgebauten Servernetzwerke vor Ausfällen schützen. Durch diese Aspekte sollen Public Cloud-Services Unternehmen bei der schnelleren Umsetzung von (Kunden-)Anforderungen unterstützen.[22]

In einigen Geschäftsbereichen und Branchen können Public Clouds jedoch aus datenschutzrechtlichen Gründen nicht verwendet werden. Unternehmen, die dennoch von den Vorteilen der Cloud profitieren möchten, setzen auf die sogenannte Private Cloud. Der Unterschied zur Public Cloud liegt darin, dass der Zugriff auf die Infrastruktur exklusiv für einen Kunden (bspw. Unternehmen) zur Verfügung steht. So kann eine Private Cloud ebenfalls von einem externen Anbieter bezogen werden, aber auch im unternehmensinternen Rechenzentrum betrieben werden. Um dies umzusetzen, müssen Technologien für die Virtualisierung sowie Schnittstellen etc. implementiert werden. Der Dienst kann durch diese Möglichkeiten individuell auf die Anforderungen und Datenschutzbestimmungen angepasst werden.[23]

Die Hybride Cloud ist eine Kombination von privaten und öffentlichen Clouds. Hier wird die durch Private Clouds zur Verfügung gestellte, eigene Infrastruktur mit der Skalierbarkeit einer Public Cloud eines externen Anbieters verbunden. Viele

[20] vgl. [NIST/2011], SP 800-145

[21] vgl. [Seub/2018], S.43 f.

[22] vgl. [Hark/2018], Chapter II

[23] vgl. [BöB+/2016], S.224

Anwendungen/Projekte mit „geringeren“ Anforderungen an die Datensicherheit können in der Public Cloud genutzt werden, wobei sich sensible Daten und geschäftskritische Bereiche wie das Finanzwesen in der Private Cloud befinden. Durch sogenanntes Load Balancing können zudem weitere Kosten eingespart werden[24]. Gemeint ist damit, dass bei zu erwartenden Spitzenauslastungen wie z.B. Marketingkampagnen oder Massentest neuer Anwendungen, genau diese Prozesse mit hohem Leistungsbedarf auf die beliebig skalierbare Public Cloud ausgelagert werden, um die kurzzeitig benötigte Leistung nicht mit Mehrkosten für die Aufrüstung der eignen Infrastruktur zu belasten. Mit dieser Kombination ist das Unternehmen weiterhin in der Lage, vertrauliche Prozesse, Daten sowie Anwendungen in der privaten Infrastruktur zu behalten und dabei immer noch auf günstigere externe Ressourcen zurückzugreifen.[25]

3. Integrationsgegenstände und -Merkmale

Wenn Cloud Services genutzt werden, müssen sich diese untereinander, aber auch mit bestehenden IT-Systemen, integrieren lassen. Dabei gibt es folgende Integrationspunkte und Integrationsmerkmale: Prozess-, Anwendungs- und Datenintegration sowie die Integrationsrichtung und Integrationsreichweite.

3.1 Prozessintegration

Bei der Prozessintegration müssen Geschäftsprozesse klar definiert sein. Es geht hierbei nämlich primär um die Integration bisher voneinander abgekapselten Geschäftsprozesse. So optimieren Unternehmen ihre Prozesse, welche durch sogenannte Workflows abgebildet werden. Cloud-Lösungen, gerade Integration Platform as a Services, müssen in der Lage sein, diese Workflows technisch auszuführen und durch bspw. Datenintegration zu fördern[26]. Das anzustrebende Ziel bei der Prozessintegration ist die ineinandergreifende Automatisierung der Geschäftsprozesse. Beispielsweise können bisher getrennte Aufgaben während eines Geschäftsprozesses zusammengeführt werden. So könnten Daten aus dem Designprozess eines Produktes ebenfalls zur Herstellung von Stücklisten/technischen Handbücher genutzt werden[27]. Dabei müssen Risiken wie die

[24] vgl. [Hark/2018], Chapter II

[25] vgl. [TeVo/2011], S.29

[26] vgl. [Sche/2019], S.8

[27] vgl. [Kaib/2002], S.17

Nichtverfügbarkeit von Modulen/Prozessen durch Systemausfälle und die daraus resultierenden Verzögerungen bei Folgeprozessen beachtet werden.[28]

3.2 Anwendungsintegration

Die Anwendungsintegration beschreibt die Abstimmung einzelner Programme eines Anwendungssystems. Für die Integration der Anwendungen werden APIs und eine Modularisierung benötigt, um eine Funktionalität über mehrere Anwendungen hinweg nutzen zu können. Eine Service-Oriented Architecture (SOA) ist ein Design für solch eine Modularisierung. SOA ist eine Art Softwartesysteme zu designen, um Funktionalitäten als Services für Applikationen oder andere im Netzwerk verteilte Services bereitzustellen. Dazu werden isolierte Softwarekomponenten in eine miteinander verbundene Sammlung einfacher Dienste umgewandelt. Jeder Dienst ist über eine Standardschnittstelle und einem Messaging Protokoll (SOAP, JSON…) aufrufbar. So können bestehende und zukünftige Anwendungen je nach Bedarf auf die einzelnen SOA-Dienste zugreifen.[29] Zusammengefasst sind SOA-Dienste eigenständige Softwarekomponenten, die wiederverwendbar sind und bestimmte Funktionen erfüllen.[30]

3.3 Datenintegration

Die Integration der Daten dient zur Bereitstellung der gleichen Datenbestände für die integrierten Anwendungssysteme. Daten sollen primär automatisiert von System zu System übertragen werden. Dabei wird zwischen folgenden Datenintegrationen unterschieden: Die manuelle Datenweitergabe, die automatisierte Datenintegration durch die Verwendung von Schnittstellen und die Nutzung einer gemeinsamen Datenbasis.[31]

3.4 Die Integrationsrichtung als Integrationsmerkmal

Ein Unternehmen kann bezüglich der Aufbauorganisation in einer Pyramide dargestellt werden. Hier wird zwischen horizontaler und vertikaler Integration differenziert. Die horizontale Integration beschreibt die Verbindung der Systeme in der Wertschöpfungskette. Dabei unterscheidet man als Integrationsgrad zum einen nach der Anzahl der Verbundenen Bereiche und zum anderen, wie eng die Bereiche untereinander verbunden sind. Die erste Stufe stellt die Daten aus einem Bereich auch für weitere

[28] vgl. [TeVo/2011], S.42

[29] vgl. [TeVo/2011], S.42

[30] vgl. [Papa/2012], S.30

[31] vgl. [Kaib/2002], S.17

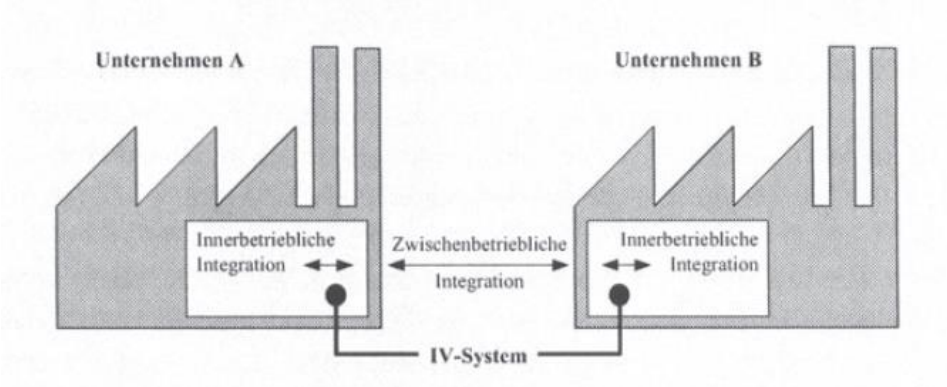
Bereiche bereit. Daten werden hier entlang der Wertschöpfungskette im Unternehmen geteilt.

Bei der vertikalen Integration geht es um die Datenversorgung der Planungs- und Kontrollsysteme aus den Dispositions- und Administrationssystemen. Die Informationssysteme der verschiedenen Ebenen haben unterschiedliche Verdichtungs- und Detailierungsgrade. Von der operativen Ebene zur Planungs- und Kontrollebene eines Unternehmens erhöht sich die Verdichtung der Daten. Zusammengefasst kann gesagt werden, dass sich der Informationsgehalt bei der vertikalen Integration verändert und bei der horizontalen Integration der Informationsgehalt gleichbleibend ist.[32]

3.5 Die Integrationsreichweite als Integrationsmerkmal

Die Integrationsreichweite unterscheidet zwischen einer innerbetrieblichen und zwischenbetrieblichen Integration. Eine innerbetriebliche Integration bezieht sich nur auf ein eigenständiges Unternehmen. Hier kann sich die Integration auf einzelne Bereiche oder auf das gesamte Unternehmen auswirken. Bei der zwischenbetrieblichen Integration wird das gesamte Betrachtungsfeld auf zwei oder mehrere, in einer wirtschaftlichen Beziehung stehenden, Unternehmen erweitert (Abbildung 3).

Abbildung 3 – Integrationsreichweiten im Überblick



Quelle der Abbildung: [Kaib/2002], S.19

Die Integration kann bspw. durch die Nutzung gemeinsamer Datenbestände gestaltet werden.[33]

[32] vgl. [Kaib/2002], S.18 f.

[33] vgl. [Kaib/2002], S.19 f.

4. Enterprise Application Integration

Eine innerbetriebliche Integration wird auch als Enterprise Application Integration (EAI) bezeichnet, jedoch geht der EAI-Ansatz auch über unternehmensgrenzen hinweg. Mit der Zeit steigt die Komplexität der IT-Landschaft von Unternehmen durch die Erneuerung bzw. Inbetriebnahme neuer Systeme. Dadurch steigt der Integrationsbedarf ebenfalls stetig. Nicht miteinander kommunizierende Informationssysteme führen zu sogenannten Informationssilos, welche die Durchlaufzeit von Geschäftsprozessen erhöhen. Durch eine Integration, also die Herstellung eines Ganzen durch das Zusammenfügen von Systemen[34], können die Informationssilos vermieden werden. EAI ist für die automatisierte Unterstützung von Geschäftsprozessen zuständig. Die Integration der Anwendungssysteme stellt für den Endnutzer eine einzelne virtuelle Anwendung dar. Neben der Kommunikation der verschiedenen Systeme unterstützt EAI ebenfalls die operative Integration von Geschäftsprozessen. Bei der Integration sollen keine oder nur minimale Veränderungen an Daten oder Anwendungen entstehen.[35] EAI ermöglicht die Integration von Individualentwicklungen, kommerzieller Standardsoftware, Dateien, Datenbanken und weiteren Systemen. Durch das „any to any"-Prinzip ist es dabei nicht relevant, von welchem Anbieter die Entwicklungen/Systeme stammen. Alles wird untereinander verbunden. Durch dieses Prinzip wird die bei traditionellen Ansätzen zur Anwendungsintegration genutzte „Punkt zu Punkt"-Integration (P2P) abgelöst. Bei der P2P-Integration werden Anwendungen je nach Bedarf direkt miteinander verbunden, was meist das Resultat angepasster Anforderungen an Geschäftsprozesse darstellt. EAI basiert jedoch auf dem Middleware-Konzept, wodurch Anwendungen und Systeme nicht direkt miteinander verbunden sind. Sie sind unabhängig, lose gekoppelt und zentral über eine sogenannte Informationsscheibe vernetzt. Dieses Konzept bringt die nötige Flexibilität in ein Unternehmen, um schnell auf veränderte Anforderungen auf dem Markt zu reagieren.[36]

Technisch gesehen laufen alle Kommunikationen über die zentrale EAI-Software. Die integrierten Anwendungen haben im Idealfall also nur eine Schnittstelle zur EAI-Software. Eine minimale Anzahl von Schnittstellen sowie der Einsatz von Standards führen zu einer vereinfachten Wartung. Neben der individuellen Einführung ermöglicht diese Architektur auch die individuelle Abkopplung bzw. Veränderung an einzelnen Anwendungssystemen, ohne das restliche Gesamtsystem zu beeinträchtigen. Zudem

[34] vgl. [Herd/2006], S.11

[35] vgl. [Kaib/2002], S. 80 f.

[36] vgl. [Kaib/2002], S. 83

können auch bereits existierende Altsysteme eines Unternehmens in die Integrationslösung eingebunden werden, was zu Kosteneinsparungen durch längere Nutzung eines bereits erworbenen Systems führt.[37] Möglich wird dies durch die in der EAI enthaltenen Adapter für bspw. Datenformationen, mittels denen die Daten aus dem Ausgangssystem an die Struktur des Zielsystems angepasst werden.[38]

Einer der ersten EAI-Integrationsansätze lautet Hub-and-Spoke. Alle integrierten Anwendungen sind über den Hub verbunden. Über diesen Hub tauschen die Anwendungen Nachrichten untereinander aus. Über die Spokes (zu Deutsch Speichen), die als Bindeglied zwischen dem Hub und den Anwendungen fungiert, senden die Anwendungen Daten zum zentralen Hub (Abbildung 4). Hierbei wird vor der Verteilung an andere Systeme das Format der Daten aus dem Ausgangssystem so angepasst, dass sie mit der Struktur des Zielsystems übereinstimmen.

Abbildung 4 – Hub and Spoke Ansatz

Quelle der Abbildung: [Ghin/2019]

Durch diese Architektur gilt es für jede zusätzliche Anwendung auch nur eine weitere Schnittstelle zu Entwickeln. Im Gegensatz dazu müssen bei der „Punkt zu Punkt“-

[37] vgl. [Kaib/2002], S. 85 f.

[38] vgl. [Ghin/2019]

Integration je nach Bedarf für eine Anwendung mehrere Schnittstellen eingefügt werden. Aus diesem Grund ist ein Vorteil des Hub-and-Spoke Ansatzes der geringere Implementierungsaufwand und folglich auch die geringeren Wartungs- bzw. Betriebsaufwände im Laufe der Zeit, da alle zu integrierenden Systeme nur einmal miteinander verbunden werden müssen.[39] Gleichzeitig stellt der zentrale Hub jedoch auch die größte Schwachstelle dieser Architektur dar. Da der Hub die zentrale Anlaufstelle für die Daten aus allen integrierten Anwendungssystemen ist, die weitergeleitet werden müssen, entsteht hier viel Last, welche sich aufgrund der zentralen Architektur auf die Perfomance aller Anwendungen auswirkt. Darüber hinaus ist es ein sogenannter Single Point of Failure, da bei einem Ausfall des Hubs die Kommunikation zwischen allen Systemen ausfällt. Eine Multihub-Konfiguration (hinzufügen weiterer Hubs) kann dabei helfen, die Perfomance auch bei Höchstlasten aufrecht zu erhalten. Das Risiko eines Ausfalls ist trotzdem sehr hoch.[40]

4.1 Enterprise Service Bus als Integrationsansatz

Der Enterprise Service Bus ist ein weiterer EAI-Integrationsansatz, welcher auch als Weiterentwicklung des Hub-and-Spoke Ansatzes betrachtet werden kann. Die Anforderungen zur Bereitstellung einer leistungsfähigen und kontrollierbaren Integrationsinfrastruktur für Web-Services sowie SOA werden durch das Konzept des Enterprise Service Bus (ESB) realisiert. Er ermöglicht die Implementierung, Ausführung und das Management von serviceorientierten Architekturen bzw. SOA-Diensten. Schwerpunkt des ESB liegt demnach bei der allgemeinen Zusammen- bzw. Bereitstellung und Verwaltung verteilter serviceorientierter Architekturen.

Ein Enterprise Service Bus ist eine Art durch Middleware-Technologien implementierte Infrastruktur, die eine SOA überhaupt erst ermöglicht und Inkompatibilitäten in einer heterogenen IT-Landschaft zwischen Anwendungen die unterschiedliche Datenformate nutzen minimiert. Auf Standards basierte Adapter und Schnittstellen ermöglichen dies durch bspw. Umwandlungen von Daten in das Format unterschiedlicher Systeme, um eine Interoperabilität zwischen Anwendungssystemen zu schaffen. Dadurch dient der „Bus“ nicht nur als Transportvermittler, sondern auch als Transformationsvermittler, der die Verteilung der Daten/Dienste verschiedener Systeme ermöglicht. Primär geht es beim ESB um die Konfiguration der Anwendungen, statt diese fest durch Programmierung

[39] vgl. [Müll/2005], S. 53

[40] vgl. [Kaib/2002], S. 87

miteinander zu verbinden. Es ist eine dedizierte Serviceinfrastruktur, die eine sogenannte „plug-and-play“-Funktionalität bietet. Dieser Mechanismus erleichtert das Hinzufügen neuer Anwendungen[41]. Eine lose Kopplung der an der Integration beteiligten Systeme und die dadurch entstehende Aufteilung der Integrationslogik in übersichtlichere Teile ist das Kernkonzept hinter dem Enterprise Service Bus. Er ist für die Steuerung, den Fluss und Übersetzung aller Nachrichten zwischen den Integrationsobjekten zuständig. Also fungiert der ESB als Vermittler zwischen dem Service-Provider und Service-Nutzer. Das Resultat ist auch hier eine Reduktion der Schnittstellenanzahl und die Wiederverwendbarkeit der Schnittstellenkomponenten. Der Vorteil ist, dass ein Service-Provider, der eine bestimmte Funktion ausführt, ohne Anpassungen am Service-Nutzer vorzunehmen ausgetauscht werden kann. Zwar wird beim Enterprise Service Bus noch immer eine zentrale Komponente für die Verteilung der Nachrichten genutzt, so sind im Gegensatz zum Hub-and-Spoke Ansatz jedoch weitere Komponenten für andere Integrationsaufgaben wie die Sicherheit und Monitoring der Vorgänge vorhanden.[42]

Um eine serviceorientierte Architektur als Enterprise Service Bus erfolgreich zu implementieren, müssen einige Aspekte bei der Umsetzung dringend beachtet werden. Zum einen muss eine Entwicklungsmethode genutzt werden, die serviceorientiert ist, um bspw. Funktionen so flexibel wie möglich als SOA-Dienste in andere Systeme zu integrieren und die Nutzung bereits existierender Anwendungen weiterhin zu gewehrleisten. Zudem sollte bestimmt werden, welche einzelnen Elemente/Funktionen von Anwendungen als solche Dienste bereitgestellt werden müssen. Verteilte Services müssen einheitlich definiert und konfiguriert werden. Des Weiteren sollte der Schwerpunkt bei solchen Projekten auf der Sicherheit, Zuverlässigkeit und Skalierbarkeit liegen. Während dem Betrieb müssen Services möglichst ohne Veränderungen am Quellcode der Anwendung gewartet und auch neu konfiguriert werden.[43]

Ein ESB hat in der Regel folgende Kernfunktionalitäten:

In einem Enterprise Service Bus werden viele unterschiedliche Datenformate von den integrierten Systemen genutzt. Der ESB spielt eine große Rolle bei der Transformation der Daten, seien es Standard XML-Formate oder individuelle Formate. Diese Funktionalität nennt sich Transformation and Mapping.[44]

---

[41] vgl. [WuTa/2010], S.90 f.

[42] vgl. [Papa/2012], S.255 ff.

[43] vgl. [Papa/2012], S.261

[44] vgl. [WuTa/2010], S.91

Eine zuverlässige Kommunikation ist dringen für die Sicherstellung der Nachrichtenübertragung an Zielsysteme notwendig. Gerade bei einer asynchronen Kommunikation, bei welcher der Sender auf keine Rückmeldung des Empfängers wartet. Außerdem können so Events wie automatisierte Bestellvorgänge zuverlässiger ausgelöst werden.[45] Dafür können essenziell fehlende Informationen in Nachrichten, basierend auf vorherigen Nachrichten, durch sogenanntes Message enhancement ergänzt werden.[46]

In serviceorientierten Umgebungen gehen die verwendeten Anwendungen und Dienste meist über Organisationsgrenzen (Abteilungen, Unternehmen etc.) hinweg. Auch sind diese durch die Modularisierung leicht ersetzbar. Ein ausgeprägtes Monitoring hilft dabei, Fehler frühzeitig zu erkennen. So wird ein ESB immer zuverlässiger.[47]

Die nächste Kernfunktionalität ist das Service Enablement. Gemeint ist, dass auch Altsysteme mit teilweise obsoleten Technologien SOA-Fähig gemacht werden können, da diese oft geschäftskritische Prozesse unterstützen.[48]

Auch die Authentifizierung, Berechtigungen und Verschlüsselungen sind Bestandteil eines ESB, um Nachrichten zu schützen.[49]

Eine weitere Kernfunktionalität ist die Skalierbarkeit. Bei einer verteilten serviceorientierten Architektur müssen einzelne Dienste oder die Gesamtinfrastruktur skalierbar sein, um die Integrationsanforderungen jederzeit zu erfüllen. Beispielsweise sind Transformationsservices sehr Ressourcenintensiv und benötigen daher skalierbare Instanzen, die bei Bedarf zugeschaltet werden. Durch die dezentralisierte Architektur bei der Nutzung von SOA können Ressourcen für Services individuell skaliert werden, ohne dass eine Skalierung der Gesamtinfrastruktur nötig ist.[50]

4.2 Funktionale Bestandteile einer EAI-Lösung

In diesem Abschnitt werden die wesentlichen Bestandteile von EAI-Lösungen näher erläutert.

[45] vgl. [Papa/2012], S.262

[46] vgl. [WuTa/2010], S.91

[47] vgl. [Papa/2012], S.264

[48] ebd.

[49] vgl. [WuTa/2010], S.91

[50] vgl. [Papa/2012], S.264

Ein wesentlicher Bestandteil sind die Adapter. Sie sind vorgefertigte Softwarebausteine, welche die Integrationsobjekte, bspw. Standardsoftware, mit der EAI-Laufzeitumgebung verbinden. Die Adapter sind anwendungs- und technologiespezifisch, weshalb für verschiedene Integrationsobjekte auch individuelle Adapter existieren.[51] Zudem werden die Adapter immer weiterentwickelt. Neben den Aufgaben, die Systeme miteinander zu koppeln, übernehmen immer mehr ausgereiftere Adapter auch die Formatierung der Dateistruktur für die jeweiligen Zielsysteme.[52]

Ein zentraler Bestandteil ist die Middleware. Sie ermöglicht den Austausch von Daten zwischen Systemen in einer heterogenen IT-Landschaft. Dabei kann diese Kommunikation synchron oder asynchron ablaufen. Bei einer synchronen Kommunikation wartet der Sender der Nachricht auf die Rückmeldung des Empfängers. Währenddessen ist das Sendersystem blockiert und kann nicht mit der Verarbeitung weiterer interner Prozesse fortfahren. Diese Abhängigkeit ist bei einer asynchronen Kommunikation nicht existent. Die interne Verarbeitung des Senders ist unabhängig von der Rückmeldung des Empfängers.[53]

Es wird zwischen fünf Kategorien von Middleware differenziert:

Remote Procedure Call (RPC)

Remote Procedure Calls ermöglichen den Aufruf von Funktionen/Prozeduren einer Anwendung aus anderen Anwendungen über das Netzwerk.[54] Da die Kommunikation synchron abläuft, muss die ständige Netzwerkverbindung zwischen Sender und Empfänger gewährleistet sein, da der Sender sonst unnötig auf eine Antwort wartet und interne Verarbeitungen blockiert sind.

Datenzugriffsorientierte Middleware

Die Datenzugriffsorientierte Middleware kann als klassische Middleware betrachtet werden. Durch einheitliche Schnittstellen ermöglicht sie den Zugriff auf verteilte Daten in einer einzelnen virtuellen Datenbank.[55]

---

[51] vgl. [Herd/2006], S. 55

[52] vgl. [Kaib/2002], S. 101

[53] vgl. [Kaib/2002], S. 102

[54] vgl. [Herd/2006], S. 55

[55] vgl. [Müll/2005], S. 55

Message-oriented Middleware

Hier kommunizieren heterogene Systeme durch den Austausch von Nachrichten. Im Gegensatz zu RPC können hier auch asynchrone Kommunikationen realisiert werden, wodurch der Programmablauf nicht gestört wird. Mittels dem Message Queing Modells werden die Nachrichten über das Netzwerk in der richtigen Reihenfolge versandt.[56]

Transaktionsorientierte Middleware

Transaktionen können entweder erfolgreich abgeschlossen oder fehlerhaft sein. Diese Eigenschaft unterstützt dabei Abfolgen von Funktionen nachzuvollziehen, indem sie als Transaktionen abgebildet werden. So kann bspw. geprüft werden, ob eine Funktion erfolgreich durchgeführt wurde.[57]

Komponentenorientierte Middleware

Eine Komponentenorientierte Middleware kann als objektorientierte Erweiterung der RPCs bezeichnet werden. „Sie dient zur Orchestrierung von verteilten Komponenten, die z.B. als Web-Services implementiert sein können“[58].

Der nächste Bestandteil einer EAI-Lösung ist das Nachrichtenmanagement. Es dient zur Realisierung der Programmintegration. Insbesondere geht es hier um die Datentransformation sowie der Synchronisation des zeitlichen Ablaufs der integrierten Systeme. Durch sogenannte Transformationsdienste werden die von den Anwendungssystemen gelieferten Daten in die richtige Zielstruktur für weitere Anwendungen gebracht. Die Ergebnisse der Datentransformation werden durch Transaktionen ebenfalls an die Systeme weitergeleitet. Hierbei werden traditionelle Boolesche Variablen (true und false) verwendet, um den Erfolg oder Misserfolg zu übermitteln.[59]

Das Prozessmanagement als Bestandteil einer EAI-Lösung ist eine Komponente zur Unterstützung verteilter Geschäftsprozesse. Als Workflows dargestellte Arbeitsabfolgen von bestimmten Funktionen können als Teil eines Geschäftsprozesses definiert werden. Durch eine Prozessmodellierung werden einem Prozess Ressourcen zugeordnet und

[56] vgl. [Kaib/2002], S. 109

[57] vgl. [Herd/2006], S. 55

[58] [Herd/2006], S. 55

[59] vgl. [Kaib/2002], S. 119

Abläufe definiert. Über die Prozesssteuerung wird das Modell ausgeführt. Die Prozesskontrolle überwacht den Ablauf (Monitoring).

In einer Metadatenbank werden schließlich Informationen wie das Nachrichtenschema und Integrationsbeziehungen der Anwendungssysteme durch Schnittstellen untereinander gespeichert.[60]

5. Hybride Integration – Eine Strategie

EAI-Integrationslösungen wie der Enterprise Service Bus können durch die weiterentwickelten Adapter zwar auch Cloud-Lösungen integrieren, werden jedoch primär zur Integration von On-Premise Lösungen eingesetzt. On-Premise bedeutet, dass die Systeme oder Anwendungen im eigenen Rechenzentrum betrieben werden und nicht fremdbezogen sind.[61] Neben dem Fakt, dass EAI-Lösungen generell sehr kostenintensiv in der Umsetzung sind, scheitern zudem fast 70% der EAI-Projekte. Gründe hierfür sind die Nichteinhaltung von Deadlines, dadurch entstehende Mehrkosten oder, dass das Ergebnis nicht den Erwartungen und Bedürfnissen des Unternehmens entsprechen. Die meisten Gründe für das Scheitern sind eher managementbedingt.[62] Außerdem muss für ESB-Integrationen viel spezifischer Code geschrieben werden, um Abläufe/Workflows zu priorisieren oder Nachrichten aus unterschiedlichen Systemen zusammenzufassen. Dies erhöht die Umsetzungsdauer und wirkt sich negativ auf die Wettbewerbsfähigkeit des Unternehmens aus, da dieses in der Entwicklungszeit nicht auf neue Marktanforderungen reagieren kann.[63]

Wie bereits zu Beginn dieser Arbeit erwähnt, setzen immer mehr Unternehmen auf Cloud-Lösungen. Gründe hierfür sind die dadurch gewonnene Agilität für immer kürzer werdende Innovationszyklen und Kosteneinsparungen.[64] Auch wenn ein Wandel stattfindet, kann dieser nicht von heute auf morgen alle geschäftsprozesskritischen Systeme ersetzen. In der Zwischenzeit baut sich eine IT-Umgebung auf, bei der sich Anwendungen und Daten in der externen Cloud und Lokal in der unternehmensinternen Infrastruktur befinden. Es entwickelt sich eine hybride IT-Landschaft. So können Unternehmen ihre Geschwindigkeit beim Umstieg Richtung „only Cloud“, also die reine

[60] vgl. [Herd/2006], S. 56

[61] vgl. [BöB+/2016], S.31

[62] vgl. [PaKh/2018], S.133

[63] vgl. [Info/o.J], S.8

[64] vgl. [PaKh/2018], S.133

Nutzung von Cloud-Lösungen, selbst festlegen.[65] Mit dieser hybriden IT-Landschaft muss sich auch die Integrationsstrategie verändern. Es ist die sogenannte hybride Integration. Denn die benötigten Daten für Cloud-Anwendungen befinden sich noch auf den alten On-Premise Systemen. Dass hierfür Schnittstellen zur Datenübertragung benötigt werden, ist logisch. Doch für die vollständige Integration werden auch optimierte Workflows benötigt. Eine hybride Strategie umfasst deshalb nicht nur IT-Spezialisten. Auch die Endnutzer können bei der Entwicklung helfen. Sie haben ein genaues Verständnis der Arbeitsabläufe. Falls die genutzte Integrationsplattform eine einfache Bedienoberfläche bietet, können die Nutzer, auch Citizen Integrators genannt, einfache, nicht unternehmenskritische Integrationen selbst übernehmen.[66] Dieses Vorgehen hat darüber hinaus den Vorteil, dass Benutzer Anwendungen, Daten etc. je nach Bedarf im Geschäftsgeschehen eigenständig und schnell integrieren können.[67] Während der Übergangsphase können die Cloud- und lokale Anwendungen für dieselben Funktionalitäten zuständig sein. Dabei kann es dazu kommen, dass bspw. Daten redundant gepflegt werden. Eine automatisierte Synchronisation, unterstützt durch die jeweilig genutzte Integrationsplattform, könnte Abhilfe schaffen.[68] Die hybride Integration sorgt damit dafür, dass alle Unternehmensanwendungen, seien es Software as a Service Produkte oder On-Premise Anwendungen, Daten an andere Systeme liefern können. Auch Unternehmen, bei denen der Umzug auf die moderne Cloud-Technologie noch nicht ganz vollzogen ist, profitieren so bereits in den Anfängen von den Vorteilen der Cloud. Für dieses Vorgehen wird eine (hybride) Integrationsplattform für die Entwicklung, Ausführung, Überwachung und Weiterentwicklung der Integrationen benötigt. Diese Plattform wird ebenfalls zur Bewältigung einiger Integrationshürden eingesetzt. Einer dieser Hürden ist bspw. die Datenkonsistenz zweier oder mehrere miteinander integrierten Systeme. Unter Umständen kann es bei der Synchronisation dazu kommen, dass unterschiedliche Aktualität der Daten in den Systemen herrschen. Dies kann der Fall sein, falls aufgrund von Netzwerkproblemen ein System nicht erreichbar ist. Regelmäßige Batch-Läufe, also automatisierte und routinierte Synchronisationen, die mithilfe der Integrationsplattform eingerichtet und geplant werden, können hier helfen.

[65] vgl. [Wood/2015], S.37

[66] vgl. [Börn/2018], S.4

[67] vgl. [Info/o.J], S.23

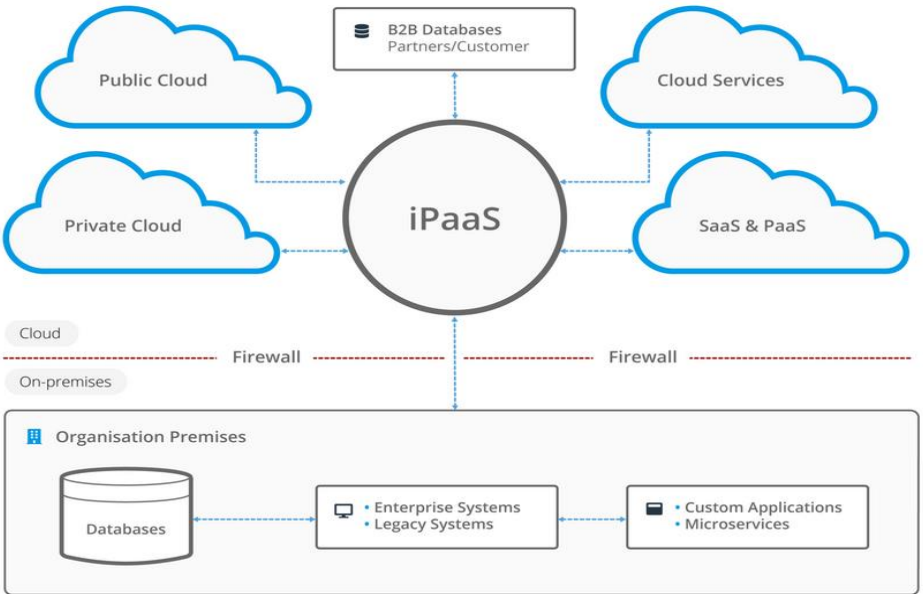
[68] vgl. [Wood/2015], S.38

Auch die Datentransformation vom Quell- zum Zielschema können Integrationsplattformen erleichtern.[69]

6. iPaaS – Cloudbasierte Integrationsplattform

Eine Integration Platform as a Service (iPaaS) ist eine cloudbasierte Integrationsplattform und wird als moderne Variante der Enterprise Application Integration betrachtet. Genauer ist iPaaS eine Suite von Cloud-Services für die Entwicklung, Ausführung und Verwaltung von Integrationen. IPaaS-Lösungen versprechen neben den ausgereiften EAI-Funktionalitäten für die Integrationen auch die Vorteile einer SaaS-Anwendung.[70] Dabei können zwischen Domänen wie On-Premise- und Cloud-Anwendungen, unterschiedlichen Cloud-Anwendungen und Geschäftspartnern Applikationen, Daten, Dienste und Prozesse integriert werden, ohne eigene Hardware installieren zu müssen.[71] Nutzer eines Integration Platform as a Service Produktes teilen sich durch die Multi-Tenant-Architektur (Kapitel 2) eine Cloud-Infrastruktur, welche vom Anbieter verwaltet und zur Verfügung gestellt wird.[72] Eine beispielhafte Integration ist in Abbildung 5 dargestellt.

Abbildung 5 – iPaaS Integrationsansatz



Quelle der Abbildung: [Code/2021]

[69] vgl. [VoHa/2012], S.86 f.
[70] vgl. [EbW+/2017], S.375
[71] vgl. [Seub/2018], S.178
[72] vgl. [EbW+/2017], S.376

Die iPaaS stellt eine Art „Middleware“ zwischen den unternehmensinternen On-Premise Systemen wie Datenbanken und Anwendungen sowie den extern bezogenen Cloud-Anwendungen dar. Des Weiteren wird die gemeinsame Datenbank des Unternehmens mit seinen Partnern ebenfalls in die Integration einbezogen, um immer einen aktuellen Datenbestand zu haben. Um dies zu ermöglichen, müssen Integration Platform as a Service Produkte einige Funktionalitäten enthalten.

Einer der wichtigsten Funktionalitäten einer iPaaS sind die umfangreichen Möglichkeiten zur Transformation von Daten. Da nicht alle Systeme dasselbe Datenformat oder gar Datentypen unterstützen, müssen bei der Integration Anpassungen vom Quell- zum Zielschema vorgenommen werden. Zum einen gibt es die Möglichkeit eines sogenannten Value Mapping der Daten. Hier können die Attribute des Quellsystems mit dem jeweiligen Gegenstück im Zielsystem meist grafisch auf der Benutzeroberfläche zugeordnet werden. So werden auch Regeln eingerichtet und eingeführt, wie die Daten zukünftig zu übertragen sind. Die Filterung der Daten für bestimmte Zwecke (bspw. überflüssige Daten) gehört ebenfalls dazu.[73] Dieses Verfahren nimmt besonders bei Standardprozessen, die häufig in Unternehmen verwendet werden, viel Entwicklungszeit ab. Einige iPaaS-Produkte schlagen sogar je nach Themengebiet etc. bereits definierte Mappings vor. Realisierbar ist dies durch die gemeinsame Nutzung einer Infrastruktur durch die Kunden. So können die Anbieter Analysedaten sammeln und ihre Produkte stetig verbessern.[74] Für das individuelle Mapping von Daten bieten die Integrationsplattformen Skripts an, mit denen in unterstützen Programmiersprachen die Datentransformation festgelegt wird.[75] Ebenfalls nicht zu vernachlässigen ist die Funktion der Integrationslogik, also wann die entwickelte Integration ausgeführt werden soll. Natürlich kann eine Integration jeder Zeit manuell ausgeführt werden oder in vom Nutzer festgelegten Zeitabständen. Aber gerade in Unternehmen bzw. der Industrie, in welcher immer mehr Systeme miteinander verknüpft sind und automatisierte Vorgänge die Arbeit erleichtern, sind eventbasierte Integrationen fundamental. Als Praxisbeispiel für solch eine Integration kann angenommen werden, dass wenn ein neuer Kunde im System angelegt wird, anschließend auf dieses Event basierend eine Synchronisierung der neuen Daten auf weitere Systeme angestoßen wird. Um die Verbindung zwischen den zu integrierenden Systemen und Anwendungen überhaupt zu realisieren, existieren vorgefertigte und konfigurierbare Adapter. Diese bilden bei der Erstellung von

[73] vgl. [EbW+/2017], S.376
[74] ebd.
[75] vgl. [EbW+/2017], S.377

Integrationen grafisch die Schnittstellen zwischen den jeweiligen Systemen und der Integration Platform as a Service. Um auch hier die Entwicklungszeit zu verkürzen, haben die meisten Anbieter einen „Market-Place“, bei welchem oft benötigte Adapter einsatzbereit vorhanden sind.[76] Durch die visualisierten Elemente wie die Adapter und einfachen Konfigurationsmöglichkeiten wie Drag&Drop-Features in einer grafischen Bedienoberfläche unterstützen iPaaS-Lösungen das Citizen Integrators Prinzip. Bedeutet, dass die IT-Spezialisten entlastet werden, weil auch Endnutzer mit wenigen Aufwand Integrationen anlegen können. Viele Kunden, die iPaaS beziehen, werden deshalb auch DIY-Kunden (Do it yourself) genannt. Sie kennen ihre Prozesse sehr gut und können kleinere Probleme selbständig lösen.[77] Dazu gehört ein ausgereiftes Monitoring, welches verständlich aufgebaut ist und mit sinnvollen Fehlermeldungen zur Problemlösung dient. Definierte Schnittstellen, Adapter und alle Integrationen werden zudem zentral verwaltet und können wiederverwendet werden. Auch die Sicherheit spielt eine entscheidende Rolle für die Nutzer von iPaaS-Produkten, da viele Daten mit diesen verarbeitet werden. Die Anbieter haben sich darauf spezialisiert und verfügen über das nötige Fachwissen bezüglich Sicherheitsmechanismen wie Authentifizierung und Standards, da sie bei Verstößen ebenfalls Haftungspflichtig werden können.[78]

Viele Elemente und Funktionen ähneln den EAI-Integrationsansätzen Hub and Spoke bzw. Enterprise Service Bus sehr, sind jedoch mit den Vorteilen der Cloud seinen „Vorgängern“ überlegen.

6.1 Die iPaaS-Architekturvarianten

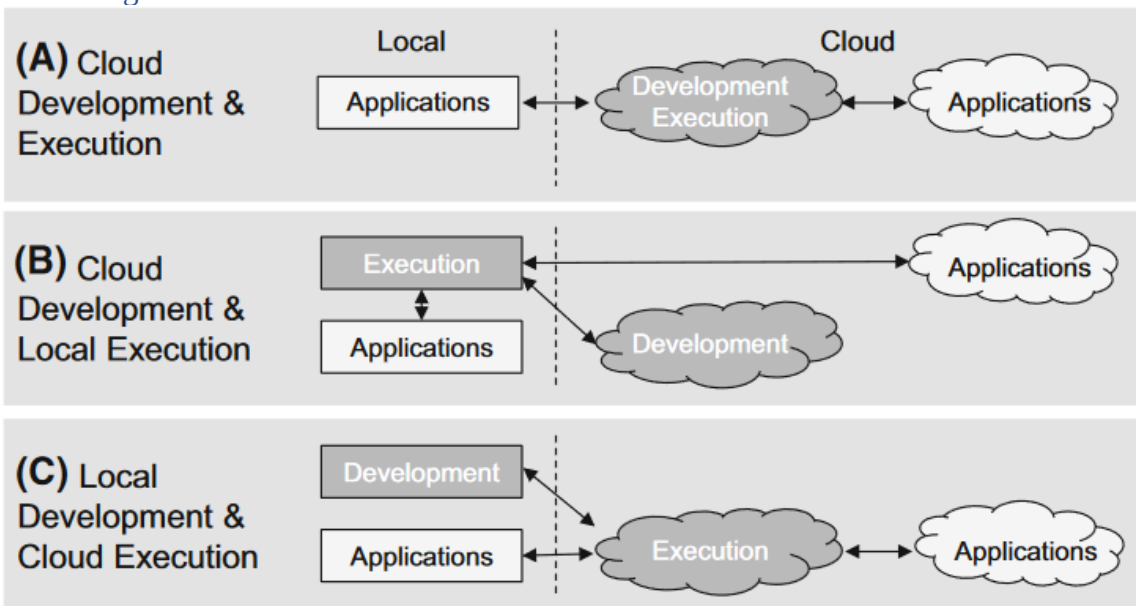
Nicht alle iPaaS-Lösungen sind in ihrer Architektur rein cloudbasiert. Es wird zwischen drei Architekturvarianten von Integration Platform as a Service unterschieden, welche in Abbildung 6 mit den Bezeichnungen (A), (B) und (C) voneinander getrennt sind.

In Architektur (A) wird die Entwicklung der Integrationen durch webbasierte Designtools unterstützt, welche Metadaten wie ValueMappings (Kapitel 6) und Prozessdefinitionen in der Cloud speichern. Während der Ausführung der Integration werden auch Daten aus den zu integrierenden Anwendungen über die iPaaS-Cloud transportiert. Die hier genutzte Infrastruktur kann durch die Anwender flexibel je nach Bedarf und Datenmenge skaliert werden, ohne dass sie dabei selbst eine eigene Infrastruktur für Integrationen

[76] vgl. [EbW+/2017], S.376
[77] vgl. [HyKo/2021], S.5
[78] vgl. [HyKo/2021], S.5 f.

warten müssen. Von allen drei Architekturen hat diese Variante (A) die kürzeste Dauer zwischen Entwicklung und Ausführung der Integrationsprozesse. Dell Boomi oder SAP-CPI sind beispielhafte Produkte, welche auf dieser Architektur basieren.[79]

Abbildung 6 – Überblick der iPaaS Architekturen



Quelle der Abbildung: [EbW+/2017], S.378

In Architektur (B) ist zwar die Entwicklung cloudbasiert, so wird der Integrationsprozess an sich jedoch lokal ausgeführt. Diese Prozesse werden in einer lokalen Laufzeitumgebung ausgeführt, die vollständig vom Nutzer verwaltet wird. Der Vorteil ist, dass wenn nur On-Premise Anwendungen miteinander integriert werden, keine Daten das Unternehmen verlassen. Der Aufbau der lokalen Infrastruktur und Umgebung kann einiges an Zeit in Anspruch nehmen und die Skalierbarkeit leidet ebenfalls bei diesem Verfahren. Bei der Architektur (C) findet die Entwicklung lokal statt. Dadurch haben Anwender meist Zugang zu umfangreicheren Entwicklungswerkzeugen, die mehr Individualisierungsmöglichkeiten bieten. Anschließend werden die Integrationsprozesse, ähnlich wie bei der Variante (A), in der Cloud eines Anbieters ausgeführt. Der Nutzer muss hier keine Infrastruktur zur Ausführung der Prozesse selbst verwalten und kann diese trotzdem nach Bedarf skalieren.[80]

[79] vgl. [EbW+/2017], S.377 f.
[80] vgl. [EbW+/2017], S.378

6.2 Schnittstellen zum Datenaustausch (REST, OData und SOAP)

Bei jeder Integration von Anwendungssystemen geht es darum, diese zu verbinden, sodass sie Daten untereinander austauschen können. REST, OData und SOAP sind Begriffe, welche bei der Anbindung bzw. des Datenaustauschs oft verwendet werden. Es sind Ansätze für die Online-Datenübertragung.

Representational State Transfer (REST) ist kein Protokoll, sondern ein Paradigma, dass primär für Webservices in verteilten Architekturen genutzt wird. Als vollkommen REST-Konform wird eine Architektur jedoch erst bezeichnet, wenn sie die sechs Richtlinien bzw. Eigenschaften erfüllt. Die erste Eigenschaft ist die Client-Server-Architektur. Hier trennt eine einheitliche Schnittstelle Client und Server voneinander. Die nächste Eigenschaft ist die Zustandslosigkeit. Zustandslosigkeit bedeutet, dass der Server keine Daten zu den Abfragen eines Clients abgespeichert. Der Client muss bei seiner Abfrage immer alle benötigten Informationen für die Verarbeitung mitschicken. Pufferbarkeit wird die dritte Eigenschaft bezeichnet. Die Antworten des Servers müssen stets die Zeit definieren, wie lange Daten gespeichert werden dürfen. Dies verhindert, dass Clients falsche bzw. veraltete Daten nutzen. Zudem müssen Client und Server so nicht dauerhaft kommunizieren. Auch das Mehrschichtsystem ist eine zentrale Eigenschaft. Der Client weiß nicht, ob er mit dem Server direkt verbunden ist, an den er die Daten liefert oder nur mit einem Server verbunden ist, der zwischen den beiden geschaltet ist. Die Struktur von Client und Server ist hierarchisch erweiterbar. Eine der wichtigsten Eigenschaften sind die einheitlichen Schnittstellen zwischen Client und Server. Die Informationen werden in einer standardisierten Form übertragen. Die letzte Eigenschaft nennt sich Programmcode bei Bedarf (Code-on-Demand). Dadurch, dass der Server bei Bedarf ausführbaren Programmcode an den Client übertragen kann, erweitert er seine Funktionalität (optional). Dabei ist es nicht relevant wie die Eigenschaften umgesetzt und implementiert werden, solange sie vorhanden sind.[81] REST arbeitet mit Befehlen, wobei jeder Befehl eine Abfrage vom Client an den Server darstellt. Die wichtigsten Operatoren lauten GET, POST, PUT, DELETE und PATCH.[82]

GET ist der bedeutsamste Befehl. Er dient zur Ermittlung eines einzelnen Eintrags oder einer Liste von Einträgen[83], die dann in Form einer Entity (Repräsentationsart wie JSON oder XML) abgeholt wird. Oft wird behauptet, dass GET-Operationen keine serverseitigen Zustandsänderungen auslösen dürfen. Doch zu einer serverseitigen

[81] vgl. [BöD+/2014], S.66
[82] vgl. [BöD+/2014], S.67
[83] ebd.

Zustandsänderung zählt auch schon ein Eintrag in die Logdatei für das Monitoring. Wichtig ist nur, dass der Client keine Anforderung zur Zustandsänderung des Servers gibt. Mit ca. 95% aller Interaktionen im World Wide Web (WWW), die einen abfragenden bzw. lesenden Charakter haben, ist die GET-Operation die verbreitetste.[84] Die POST-Operation dient zum Erzeugen eines neuen Eintrags bzw. sendet Daten zum Server für die Verarbeitung. Über den HTTP-Statuscode 201 (Created) informiert der Server den Client, dass der Eintrag erfolgreich angelegt wurde.[85] PUT-Operationen dienen zur Veränderung eines existierenden Eintrags. Diese Operation ist wie GET und DELETE idempotent. Dies bedeutet, dass man bei ihrer Ausführung immer dasselbe Ergebnis erwarten kann. Auch wenn die Operation mehrmals ausgeführt wird, hat dies denselben Effekt, als wenn sie einmal ausgeführt wurde. Der Vorteil hiervon ist, dass wenn bspw. der Client keine Antwort auf seine Anfrage bekommt. Es entstehen zwei Möglichkeiten: Entweder die Anfrage wurde korrekt verarbeitet, aber die Antwort ist nicht angekommen oder bei der Verarbeitung ist etwas schiefgelaufen. Da bei einer weiteren Anfrage keine anderen Ergebnisse oder Veränderungen zu erwarten sind, kann diese erneut geschickt werden.[86] Durch PATCH werden nur einzelne Attribute eines existierenden Eintrags aktualisiert.[87] Durch die DELETE-Operation werden existierende Einträge gelöscht, welche bei der Anfrage in der URI (Universal Resource Identifier) angegeben werden. Wie bereits beschrieben ist DELETE ebenfalls idempotent, da das Löschen eines Eintrags immer denselben Effekt aufweist.[88] Serverseitig werden diese Operationen durch entsprechende Create-, Retrieve-, Update- und Delete-Methoden verarbeitet.[89] Heutzutage ist REST aufgrund der Flexibilität durch die nicht verpflichtenden Richtlinien sehr beliebt.

Bei der Anwendung von Schnittstellen können Daten in unterschiedlichen Formaten ausgegeben und verarbeitet werden. XML und JSON sind die verbreitetsten Formate. XML steht für eXtensible Markup Language und ist ein einfaches flexibles Textformat, welches im Jahre 1995 seinen Ursprung hat. Es ist seit langem ein Industriestandard, da es eine standardisierte und plattformübergreifende Sprache ist. Die Daten werden als Baumstruktur angezeigt, welche mit Elementen und Attributen beschrieben werden. Dadurch verlieren die Daten bei Prozessen nie ihre Hierarchiebeziehung.[90] JSON

[84] vgl. [Tilk/2011], S.51 f.
[85] vgl. [Tilk/2011], S.54
[86] vgl. [Tilk/2011], S.55
[87] vgl. [BöD+/2014], S.67
[88] vgl. [Tilk/2011], S.55
[89] vgl. [BöD+/2014], S.67
[90] vgl. [BoYa/2012], S.1174 f.

hingegen steht für JavaScript Object Notation und basiert, wie der Name bereits verrät, auf der Programmiersprache JavaScript. Mittlerweile wird das JSON-Format von fast allen Programmiersprachen unterstützt, was es universell einsetzbar macht[91]. Generell ist JSON einfach zu lesendes Format. Mit JSON werden Daten hauptsächlich in Arrays und Objekten beschrieben. Der Fokus liegt hier hauptsächlich auf den Datenstrukturen und nicht auf Text.[92]

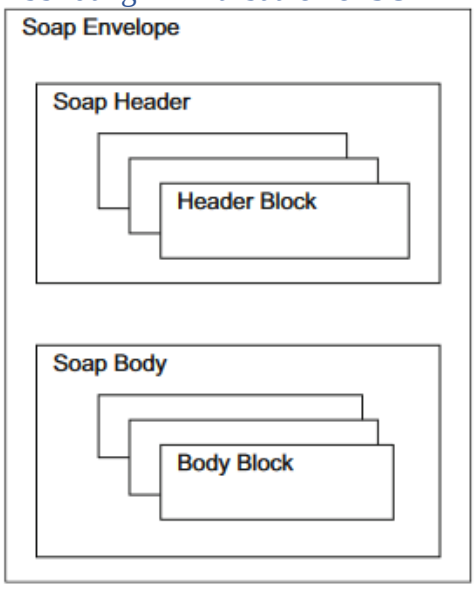
Das Open Data Protocoll, auch OData genannt, basiert auf REST und wurde von Microsoft veröffentlicht. Da es auf Industriestandards wie XML und JSON aufbaut, ist das Verständnis und die Akzeptanz hoch. Des Weiteren erlaubt OData auch Daten aus unterschiedlichen Quellen zusammenzuführen und dem Konsumenten in einer einheitlichen Form zu liefern. Durch diese Vereinheitlichung der Daten verbessert sich die Interoperabilität zwischen den Systemen, was ebenfalls Integrationen erleichtert. Der OData-Service stellt den Endpunkt dar, welcher vom Client für seine Anfragen genutzt wird.[93] Auch hier werden die beschriebenen REST-Befehle unterstützt. Durch Standards und Best-Practices können sich die Anwendung auf die Geschäftslogik fokussieren, ohne sich mit verschiedenen Schnittstellen zu beschäftigen.[94]

Anwendungen müssen heutzutage miteinander kommunizieren, um Informationen untereinander auszutauschen und so Prozesse zu optimieren als auch zu automatisieren. Dies kann jedoch nur gelingen, wenn proprietäre Systeme in einer heterogenen Infrastruktur Daten untereinander austauschen können. Für dieses Ziel nutzen viele Anwendungen Webservices, die auf SOAP basieren. SOAP steht für Simple Object Access Protocol und ist ein Standardprotokoll mit integrierten Regeln (z.B. Sicherheitsstandards), welches auf XML basiert und dient zum Austausch von Nachrichten zwischen Anwendungen. Dabei ist SOAP unabhängig von verwendeten Betriebssystemen oder Programmiersprachen. Es stellt Nachrichten in XML-Form bereit, die meist in dezentralen und verteilten IT-Umgebungen genutzt werden. Hauptverwendungszweck von SOAP liegt primär in der Kommunikation zwischen Systemen wie z.B. bei einer Integration. Über HTTP werden Datenanforderungen an die SOAP-Schnittstelle geschickt. Vereinfacht dargestellt ist eine SOAP-Nachricht eine einfache HTTP-Anfrage und Antwort. Der SOAP-Endpunkt, welcher bei Anfragen angegeben wird, ist eine HTTP-basierte URL. Die Antwort/Nachricht auf Datenanfragen

[91] vgl. [Tilk/2011], S.91
[92] vgl. [BoYa/2012], S.1175
[93] vgl. [BöD+/2014], S.72-74
[94] vgl. [Micr/2022]

wird immer im XML-Format zurückgegeben.[95] Diese XML-Nachrichten bestehen aus den drei Teilen Envelope, Header und Body (Abbildung 7).

Abbildung 7 – Aufbau einer SOAP-Nachricht



Quelle der Abbildung: [Beng/2014], S.216

Der Envelope bildet den „Umschlag“, welcher die Mechanismen bereitstellt, um die tatsächlichen Daten im XML-Format mit zusätzlichen Informationen zu versorgen, sodass diese ihr Ziel garantiert erreichen. Damit sich Server und Client verstehen, müssen sie den gleichen Namensraum/Vokabular nutzen. Er enthält Deklarationen, die für alle Attribute und Elemente gelten. Im Header befinden sich Verarbeitungshinweise, welche für den Endpunkt relevant sind und spezifiziert bspw. Sicherheitskonventionen oder Authentifizierungsinformationen, die eingehalten werden sollen (optional). Der Body beinhaltet die tatsächlichen Daten für den Austausch zwischen Anwendungen.[96]

SOAP ist also einfach gestrickt, da es auf XML basiert und einfach zu verarbeiten ist. SOAP ist unabhängig von genutzten Anwendungen, Systemen und Plattformen einsetzbar. Mittlerweile können die meisten Systeme XML verarbeiten. Dadurch ist, neben dem Fakt, dass viele Altsysteme auf SOAP setzen und diese Technologie weiter genutzt wird, die Akzeptanz bei den Anwendern hoch. Es ist jedoch zu beachten, dass auch SOAP zustandslos ist. Bei jeder Anfrage muss sich die Anwendung erneut vorstellen, was zu höherem Datenverkehr führen kann.[97] Zur Beschreibung von

[95] vgl. [Papa/2012], S.126
[96] vgl. [Beng/2014], S.217
[97] vgl. [Papa/2012], S. 146 f.

Webservices, die über SOAP-Nachrichten angesprochen werden, wird die Webservice Description Language (WSDL) eingesetzt. Sie basiert ebenfalls auf XML und ist im Laufe der Zeit zum Standard geworden. WSDL wird verwendet, um die Funktionen eines SOAP-Webservices zu beschreiben. Genau beschreibt eine WSDL zum einen, was ein Service für Funktionen (Operations) bietet und welche Daten von welchem Typen enthalten sind. Des Weiteren ist der Endpunkt, wo sich der Webservice befindet, angegeben. Die WSDL-Datei kann symbolisch als Vertrag zwischen Service-Nutzer und Service-Anbieter gesehen werden und regelt deren Beziehung durch die beschriebenen Informationen.[98] In Kapitel 8.2 ist ein praktisches Beispiel beschrieben, welches erläutert wo und wie eine WSDL-Datei eingesetzt wird.

6.3 Ein Überblick der Potenziale und Herausforderungen durch iPaaS

Allgemein betrachtet liegen einige Vorteile von iPaaS-Lösungen mit denen von Cloud-Anwendungen in einer Linie. IPaaS-Lösungen werden identisch zu SaaS-Anwendungen bereitgestellt, wodurch keine eigene Hardwareinfrastruktur benötigt wird. Dadurch können Kosten eingespart werden. Entwicklungswerkzeuge und vorgefertigte Softwarekomponenten erleichtern die Integrationen, auch durch Citizen Integrators. Das Resultat neben einer einheitlichen Struktur bei der Entwicklung, Wartung und Ausführung der Integrationsprozesse sind unter anderem verbesserte Workflows, da sich alles wichtige wie Entwicklungsumgebung und Monitoring auf einer zentralen Plattform befindet. Die Mandantenfähigkeit erlaubt es, virtuelle Instanzen wie Entwicklungs- und Testumgebungen, ohne weitere Hardwareinfrastruktur in Anspruch zu nehmen, einzurichten. Falls doch mehr Ressourcen benötigt werden, sind iPaaS-Lösungen skalierbar. Durch die schnelle Bereitstellung von iPaaS und den effizienten Entwicklungsmethoden (z.B. grafische Oberflächen) ist es möglich, schnell auf Marktanforderungen zu reagieren und Kundenanfragen umzusetzen. Mittels regelmäßigen Updates durch den Anbieter werden iPaaS-Lösungen sicher gehalten, indem Sicherheitsstandards gepflegt und Sicherheitslücken geschlossen werden.[99]

Doch die größte Herausforderung liegt trotz den Bemühungen der Anwender noch immer beim Thema Sicherheit. Anwendungsdaten werden über die Cloud bzw. das Internet ausgetauscht, was eine potenzielle Schwachstelle für die Datensicherheit darstellt, bei der sich die Kunden voll auf die Anbieter der Plattformen verlassen müssen. Ein weiterer

[98] vgl. [Papa/2012], S.151 ff.
[99] vgl. [Kunt/2021]

Aspekt ist, dass Meta-Daten wie Passwörter für den Zugang zu Anwendungen auch mit Drittanbietern geteilt werden können. Dieser Drittanbieter kann bspw. die Cloud-Infrastruktur bereitstellen, auf der die iPaaS-Lösung aufgebaut ist. Sicherheitsstandards wie die ISO 27001 helfen Anbietern zwar beim Gewährleisten der Sicherheit, jedoch sind die Informationen über die genutzten Standards, gerade bei kleinen Anbietern, oft nicht transparent kommuniziert oder für solche Produkte nicht geeignet.[100] Zudem hat eine schlechte Internetanbindung oder ungenügende Performance der vom Anbieter bereitgestellten Plattform einen direkten Einfluss auf die Integrationsprozesse und deren Stabilität. Wesentlich ist auch der Punkt des Vendor Lock-In. Anbieter benutzen oft proprietäre Formate und Technologien, sodass sie mit den Produkten anderer Anbieter nicht kompatibel sind. Ein Wechsel fällt deshalb schwer, da im schlimmsten Fall alle Prozesse/Integrationen wieder manuell angelegt werden müssen. So machen Anbieter die Kunden von sich abhängig.[101]

6.4 Der Vergleich zwischen iPaaS und EAI

Integration Platform as a Service (iPaaS) und Enterprise Application Integration (EAI) verfolgen beide dasselbe Ziel der Integration von Informationssystemen. Die meistimplementierte Form von EAI ist der Enterprise Service Bus.

Im Vergleich zu iPaaS wird der ESB-Ansatz als weniger flexible Architektur angesehen. IPaaS gilt als der agilere Integrationsansatz. Der ESB ist eng mit einer serviceorientierten Architektur verbunden. Diese eignet sich primär für die Integration von On-Premise Anwendungen untereinander. ESB war nämlich die Antwort auf die Punkt-zu-Punkt Integrationen, durch die keine Flexibilität gegeben war. Jede Anwendung musste einzeln mit der anderen verbunden werden. Der ESB koordiniert die Daten in einem Unternehmen, indem alle Anwendungen bzw. Services über diesen Bus miteinander verbunden sind. Der cloudbasierte Service (iPaaS) ist nicht nur für On-Premise Integrationen, sondern auch für fremdbezogene/gehostete Cloud-Anwendungen optimiert und ist dabei weniger komplex aufgebaut. Dieser Ansatz fokussiert sich auf die nahtlose Verbindung von Anwendungen in Unternehmen oder über Unternehmensgrenzen hinweg, seien es alte Legacy-Systeme oder Cloud-Anwendungen.[102] IPaaS fördert die Nutzung moderner Schnittstellen wie REST, welche für den Datenaustausch von

[100] vgl. [EbW+/2017], S.378
[101] vgl. [EbW+/2017], S.379
[102] vgl. [IBMC/2019]

cloudbasierten Systemen optimiert ist. ESB-Integrationen setzen eher auf synchrone Protokolle, welche sich eher für On-Premise Anwendungen eignen und nicht für Echtzeitbearbeitungen geeignet ist. Zudem bietet iPaaS Möglichkeiten, ältere Formate wie XML in modernere cloudfähige Formate wie JSON umzuwandeln. So können bereits bestehende Anwendungen im Unternehmen weitergenutzt werden und trotzdem von den Vorteilen von iPaaS profitieren. Die Daten bereits bestehender Systeme können für Cloud-Anwendungen zugänglich gemacht werden und so in einer neuen, moderneren Oberfläche genutzt werden. Ein weiterer Vergleichsaspekt ist die Mandantenfähigkeit beider Integrationsansätze. Mit iPaaS muss nicht für jede Abteilung oder Unternehmenspartner/Kunden eine neue physische Instanz aufgebaut werden. Durch die Virtualisierung ist es möglich, verschiedene Instanzen auf einer Infrastruktur zu erstellen, auf die von überall zugegriffen werden kann. Ein ESB hingegen ist lokal implementiert und unterstütz deshalb die Mandantenfähigkeit ohne große Aufwände nicht über Unternehmensgrenzen hinweg.[103] Auch die Kostenmodelle sind unterschiedlich und haben einen anderen Schwerpunkt, was sich ebenfalls in der Skalierbarkeit widerspiegelt. Cloud-Anwendungen (SaaS), in welcher Form auch Integration Platform as a Service bereitgestellt wird, werden meist monatlich verrechnet. Das sogenannte Pay-per-Use Preismodell besagt, dass der Kunde nur das Zahlt, was er tatsächlich nutzt. Dabei variieren die Preise je nach benötigten Ressourcen (Speicher, Rechenkapazität usw.).[104] Die Einstiegskosten halten sich aufgrund dieser „Fixpreise“ im Vergleich gering. Da der ESB in der unternehmensinternen Infrastruktur betrieben wird, fallen die Einstiegskosten durch Hardware- und Lizenzanschaffungen hoch aus. Zwar sinken die Kosten nach der Zeit, nachdem die Hardware erworben wurde, so fallen jedoch laufend Wartungen und Personalkosten an. Auch Updates müssen bezahlt werden, während diese bei iPaaS im Preis inklusive sind.[105]

Werden die Vergleichsaspekte im Ganzen betrachtet, so kann gesagt werden, dass bei einer reinen On-Premise Integration der ESB-Ansatz präferiert werden sollte. Dadurch verlassen keine Daten bei der Integration das Unternehmen, was zu einer leichteren Einhaltung des Datenschutzes führt. Bei einer reinen Cloud-Integration oder wenigen On-Premise Systemen, auch hybride IT-Landschaft genannt, lohnt sich eine iPaaS. Aber iPaaS kann auch zur Unterstützung eines bereits implementierten ESB genommen werden. So können unternehmensinterne (lokale) Anwendungen mit einem ESB

[103] vgl. [Kunt/2021]
[104] vgl. [Höll/2011], S.205
[105] vgl. [Höll/2011], S.207

integriert werden und Cloud-Anwendungen/Daten werden mittels iPaaS für andere Anwendungen bereitgestellt.[106]

7. Der Markt für Integration Platform as a Service

Nach Gartner sollte ein iPaaS-Anbieter möglichst hohe Verfügbarkeit, Optionen zur Datensicherung und Sicherheitsfunktionen bieten. Auch die Verwaltung der Plattform durch Patches und Updates ist durch den Anbieter zu gewährleisten. Der Markt für iPaaS wächst stetig, da immer mehr Unternehmen auf Cloud-Anwendungen umsteigen, wodurch sich ihr Integrationsbedarf erhöht. Bereits im Jahre 2009 beträgt der Umsatz für iPaaS ähnliche Produkte/Services (z.B. Integration as a Service) ungefähr 945 Millionen US-Dollar. Schon damals wurde erwartet, dass der Markt für iPaaS-Lösungen in den nächsten fünf Jahren aufgrund des steigenden Integrationsbedarfs stark expandieren würde. Im Jahr 2013, also schon nach kurzer Zeit, waren größere Unternehmen vermehrt auf der Suche nach flexibleren und im Vergleich zu traditionellen On-Premise Integrationsansätzen effizienteren Lösungen cloudbasierte Anwendungen zu integrieren. IPaaS wurde langsam eingeführt und zunächst eher als Ergänzung für existierende EAI-Integrationen eingesetzt, um Risiken bei einem Umstieg zu minimieren. Die Prognose zum Jahr 2015 besagte, dass ein Umschwung am Integrationsmarkt herrschen wird und bis 2016 ungefähr 35% der Groß- und mittelständischen Unternehmen mindestens eine iPaaS-Lösung einsetzen werden.[107] Diese Prognosen bewegten einige Unternehmen dazu, dem Trend relativ früh zu folgen. Zu langes Warten, bis der Markt eine gewisse Reife erreicht hat, kann nämlich gefährlich sein. Konkurrenten können durch die frühe Nutzung von iPaaS über einen längeren Zeitraum ihre Effizienz steigern und so Kosten einsparen. Bevor iPaaS jedoch eingeführt wird, muss zwischen technologischen und anbieterseitigen Risiken die Sinnhaftigkeit des Einsatzes dieser Plattform abgewogen werden.[108] Aktuell wird der Umsatz am iPaaS-Markt auf 3,47 Milliarden US-Dollar geschätzt, was ein Wachstum von ca. 39% zum Vorjahr bedeutet. Bis zum Jahr 2025 soll der Umsatz schätzungsweise auf neun Milliarden US-Dollar steigen.[109] Demnach haben die Prognosen aus den früheren Jahren Recht behalten.

Durch dieses Wachstum ist der Markt für Integration Platform as a Service jedoch sehr unübersichtlich geworden, da viele Anbieter unterschiedliche Funktionen und Preise

[106] vgl. [IBMC/2019]
[107] vgl. [Mate/2012], S.1904
[108] vgl. [Mate/2012], S.1907
[109] vgl. [ThGu/2021], Chapter Market Overview

anbieten. Für eine Übersicht der Anbieter sorgt das Magic Quadrant von Gartner. Er unterteilt Anbieter, welche die essenziellen Kriterien wie z.B. die Synchronisierung von Daten zwischen Anwendungen, eventbasierte Integrationen, die Möglichkeit On-Premise sowie Cloud-Dienste zu integrieren und auch technischen Support für Kunden erfüllen, in Leaders, Challengers, Visionaries und Niche Players (Abbildung 8). Der Quadrant soll dabei kleinere Anbieter nicht schlecht darstellen, sondern lediglich einen Überblick liefern. Kleinere Anbieter können bspw. genau die gewünschten Funktionen zu niedrigeren Preisen anbieten und sind bei der Anbieterauswalt deshalb nicht zu vernachlässigen. Die Bewertungskriterien sind dabei zum einen die Fähigkeit des Anbieter Funktionen umzusetzen, sowie deren Vision.

Abbildung 8 – Gartner Magic Quadrant iPaaS

Quelle der Abbildung: [ThGu/2021], Chapter Magic Quadrant

Niche Players sind in der Regel nur auf einen kleinen funktionalen oder regionalen Bereich spezialisiert und bedienen somit nur einen geringen Teil des Marktes. Oft sind es kleinere Unternehmen (unteranderem Celigo), die sich auf spezifische Anwendungsfälle fokussieren, was zu einer hohen Kundenzufriedenheit führt. Sie pflegen eine enge Beziehung mit den Kunden und können diesen dadurch besser beraten. Durch ihre

Spezialisierung sind Produkte von Niche Players oft das Ziel von Übernahmen durch größere Unternehmen, die so ihr Angebotsspektrum erweitern.[110]

Visionäre haben ein gutes Verständnis für neue Technologien und Trends. Sie bieten Funktionen an, bei denen die Nachfragen aktuell hoch sind und kreieren so Lösungen, die sich mit der Zeit zu einem umfassenden Produkt entwickeln. Wegen fehlenden Kenntnissen im Bereich des Marketings oder Vertriebs ist es für diese Anbieter schwer, neue Kunden dazuzugewinnen. Visionäre auf dem Markt übernehmen deshalb gerne kleinere Anbieter, entwickeln die Produkte weiter, um so präsenter auf dem iPaaS-Markt zu sein. Jitterbit und IBM sind laut Gartner Visionaries.[111]

Challengers sind Anbieter mit ausgereiften und sich stets in der Weiterentwicklung befindenden Produkten, die sich bereits bei Kunden bewährt haben. Sie verfügen über genügend finanzielle Mittel für aggressives Marketing und bleiben so konkurrenzfähig. Jedoch haben sie meist nicht den gesamten iPaaS-Markt im Blick, wodurch ihre Angebote eingeschränkter als die von den Leadern sind.[112]

Leaders wie bspw. SAP sind die Marktführer am iPaaS-Markt. Ihre Produkte haben hohe Erfolgsraten. Auch sind sie in der Lage den Markt durch ihren Einfluss zu lenken. Die Produkte der Marktführer sind mit Road-Maps vorausgeplant und geben dem Kunden Zukunftssicherheit. So bleiben die Kunden und die Anbieter festigen ihre Position. Nicht nur bieten sie für aktuelle Probleme Lösungen, sondern auch für neue, zukunftsorientierte Probleme. Die Integrationsplattformen bieten daher viele ausgereifte Funktionen, die mit regelmäßigen Updates verbessert werden. Um auf dem schnelllebigen iPaaS-Markt führend zu bleiben, ist dieses Vorgehen dringend notwendig.[113]

7.1 Kriterien zur Anbieterwahl

Die Anbieterwahl im iPaaS-Segment kann sich als Herausforderung herausstellen, da jedes Unterhemen andere Anforderungen an eine Integrationsplattform hat. Jedoch gibt es einige Kriterien, die als Standardfunktionen und Services bei iPaaS-Produkten inkludiert sein sollten, die als Art Vermittler zwischen unterschiedlichen Anwendungen fungiert. Zunächst sollte das Kosten-Nutzen-Verhältnis stimmen. Besonders kleinere Unternehmen mit proprietären Anwendungen benötigen individuell entwickelte

[110] vgl. [ThGu/2021], Chapter Quadrant Descriptions
[111] vgl. [ThGu/2021], Chapter Quadrant Descriptions
[112] vgl. [ThGu/2021], Chapter Quadrant Descriptions
[113] vgl. [ThGu/2021], Chapter Quadrant Descriptions

Konnektoren/Adapter, um diese Anwendungen zu integrieren. Es ist nicht sinnvoll, iPaaS einzuführen und nicht von effizienteren Integrationsprozessen zu profitieren. Eine Funktionalität ist demnach eine breite Auswahl an Konnektoren und die Möglichkeit nicht verfügbare, aber zwingend benötigte Adapter schnell auf Anfrage zu bekommen. Anbieter, die freikonfigurierbare Adapter liefern sind hier im Vorteil. So haben die Kunden die Option, eigenständig und unkompliziert individuelle Integrationen zu entwickeln.[114]

Falls das Datenschema der zu integrierenden Systeme nicht miteinander kompatibel ist, sollte es eine einfache (optimalerweise grafische) Funktion zum Mapping geben. Für Standardanwendungen sollte das Mapping intelligent automatisch im Hintergrund durchführbar sein. Zu Automatisierungszwecken sollten die Daten bei der Integration zentral gespeichert werden, um so ein Muster für spätere Integrationsprozesse zu schaffen, welche die intelligenten Mappings unterstützt. So kann bspw. eine Datenanreicherung stattfinden, die sonst manuell durchgeführt werden müsste.[115] Ein Monitoring sollte ebenfalls eine ausgereifte Funktionalität einer iPaaS sein, sodass erfolgreiche Integrationen angezeigt und Fehler durch nachvollziehbare Hinweise behoben werden können.[116]

Um ungewollte Ausfälle zu vermeiden, sollte auch das Thema Servicequalität bei der Anbieterwahl betrachtet werden. Wird im Vertrag z.B. die Verfügbarkeit der iPaaS auf 365 Tage verteilt mit 99,5% angegeben, sind es ca. 44 Stunden innerhalb denen die iPaaS nicht verfügbar sein kann, ohne dass es rechtliche Konsequenzen für den Anbieter hat. Der Service sollte schnell und einfach zu erreichen sein.[117] Weiterhin sind Schulungen für das Produkt ein weiteres, nichttechnisches Kriterium. Für ein schnelles On-Boarding und die Nutzung des vollen Potentials durch die Anwender sind Schulungen ein wichtiges Mittel zur Wissensübermittlung, welche auch den Anbieter beim Support entlasten.[118]

[114] vgl. [NeL+/2021], S.50
[115] vgl. [NeL+/2021], S.52
[116] vgl. [NeL+/2021], S.52
[117] vgl. [Höll/2011], S.231
[118] vgl. [NeL+/2021], S.51

8. SAP-Cloud Integration (iPaaS)

SAP ist einer der führenden Anbieter für Integration Platform as a Service. In diesem Kapitel folgt eine Vorstellung der SAP Cloud Platform Integration (CPI) und ein Integrationsszenario, bei dem Daten aus dem SAP HCM nach SuccessFactors übertragen werden.

8.1 SAP Cloud Platform Integration (CPI)

Die SAP Cloud Platform Integration ist eine Plattform, die systemübergreifende Integrationen, seien es On-Premise- oder Cloud-Anwendungen, ermöglicht. Sie unterstützt nicht nur SAP-Anwendungen, sondern auch Anwendungen anderer Anbieter. Auch Daten aus sozialen Medien sind in der CPI verwendbar. Die Verwendung der Daten von Geschäftspartnern im Integration-Service durch Schnittstellen ist ebenfalls möglich. Um behördliche Pflichten zu erfüllen, stellt die CPI vorgefertigte Möglichkeiten bereit.[119] Die CPI ist zudem in der Lage, Nachrichten in Echtzeit zu verarbeiten[120]. Die Integrationen werden durch sogenannte Integration Flows (iFlows) realisiert. In den Integration Flows wird festgelegt, wie Daten unterschiedlicher Systeme verarbeitet und an das Zielsystem weitergeleitet werden. Für Standardprozesse gibt es in der CPI vorgefertigte iFlows, die nur kleinere Anpassungen wie bspw. das Eintragen der Endpunkte für die Anwendungen erfordern. Die meisten vorgefertigten iFlows sind für SAP-Standardintegrationsszenarien entworfen. Dazu gehören die Datenübertragung mit ERP-Systemen, SAP-SuccessFactors, Concur und Ariba. Neben den genannten Beispielen gibt es auch die Option eigene Integrationsszenarien mit iFlows umzusetzen.[121] Die bedeutendsten Operatoren, um eigene Integration Flows zu erstellen sind die Transformation von Daten, Aufrufe externer Systeme/Prozesse, das Routing der Daten zu den Empfängern, das Speichern der Daten und die Datensicherheit. Die Transformation der Daten in eine andere Struktur kann mit dem Mapping-Operator durchgeführt werden. Dies ist ein grafischer Editor, in welchem die Attribute des Quellsystems dem Gegenstück des Zielsystems zugewiesen werden. Nicht vom Editor unterstützte Formate oder individuelle Mappings können im iFlow mit dem Script-Operator bzw. mit Groovy oder JavaScript programmiert werden. Im Operator Content Modifier besteht die Möglichkeit Daten bspw. im XML-Header mit weiteren Daten anzureichern. Die konfigurierbaren Conversion-Operatoren können Daten von

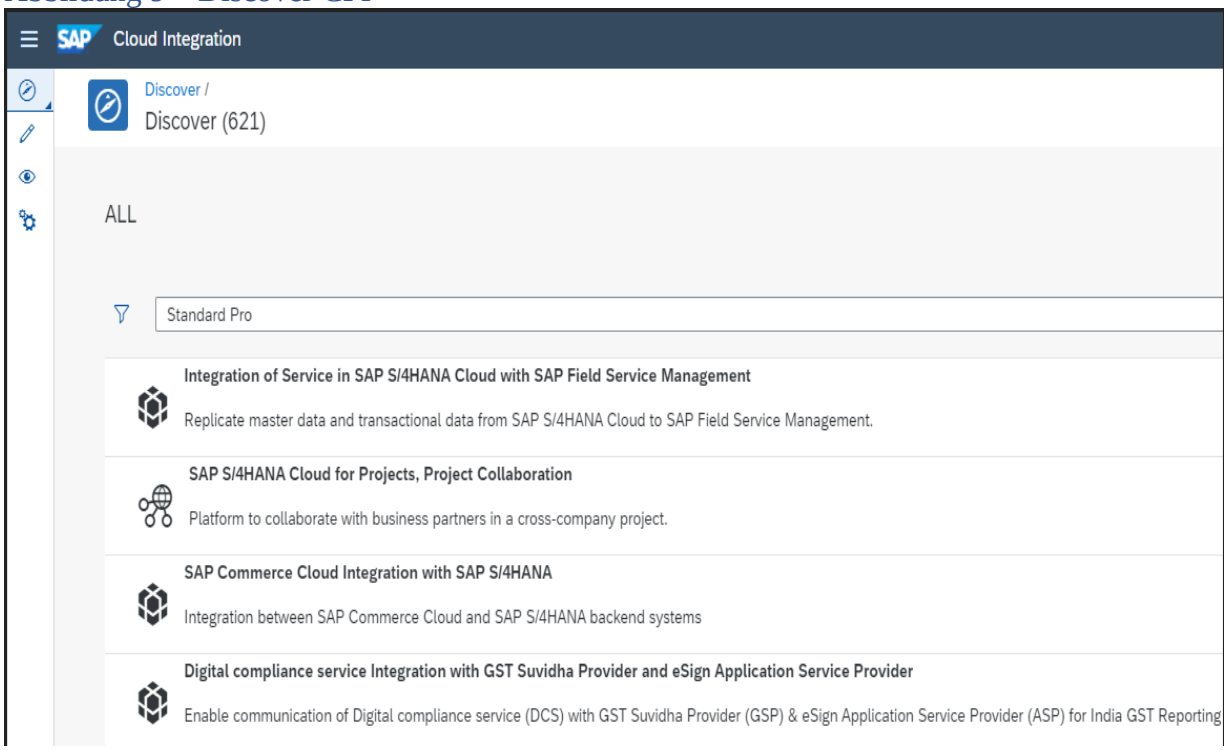
[119] vgl. [Seub/2018], S.185
[120] vgl. [SAP/2022]
[121] vgl. [Seub/2018], S.186

unterschiedlichen Formaten in einem einfachen Schritt umwandeln. Die CPI unterstützt hier Formate wie XML, JSON und CSV.[122] Um externe Systeme/Prozesse aufzurufen, bietet die CPI den Operator Request Reply. Hier wird ein synchroner Prozessaufruf zu einem externen System ausgeführt und der Operator erhält eine Antwort vom System. Falls Daten empfangen werden, beinhalten sie die Daten, welche sich dann in der CPI befinden. Werden Daten an ein externes System gesendet, ist die Antwort entweder eine Erfolgsmeldung oder ein Fehlercode.[123] Der Router-Operator ermöglicht die Weiterleitung der in der CPI verarbeiteten Daten zu mehreren Zielsystemen (Multicast). Die Konfiguration ermöglicht es dem Anwender festzulegen, ob die Weiterleitung parallel oder zeitversetzt erfolgen soll. Auch durch den Inhalt der Nachrichten kann bestimmt werden, ob und an welches System die Daten gesendet werden sollen. Mit dem Persist-Massage-Operator können Daten für eine spätere Verwendung oder Analysezwecke gespeichert werden.[124] Datensicherheit ist in der Integration ein großer Faktor. Um diese zu gewährleisten, wird mit Verschlüsselungsmethoden durch Encryptor- und Decryptor-Operatoren gearbeitet.[125] Die Integrations-Suite von SAP läuft im Web. Die Weboberfläche der CPI ist in die vier Bereiche Discover, Design, Monitor und Settings unterteilt.

Durch SAP vordefinierte Integrationsinhalte befinden sich im Bereich Discover (Abbildung 9). Neben reinen Integration Flows stehen hier auch Mappings und weitere Integrationsobjekte zur Verfügung. Über ein Suchfeld gelingt das Auffinden des gesuchten Integrationsobjektes schnell. Für eine zuverlässige Suche sind weitere Informationen angegeben, welche auch zum Filtern genutzt werden können. Die Artifacts sind die tatsächlichen iFlows oder Mappings, die zur Auswahl stehen. Im Bereich Documents befinden sich Leitfäden und Links zur Hilfe, welche Möglichkeiten das ausgewählte Szenario bietet. Tags helfen dabei, die Integrationsszenarien/Inhalte zu klassifizieren. Angaben sind hierbei Felder wie unterstützte Plattformen oder Geschäftsbereiche.[126]

[122] vgl. [SAP/2022], Chapter Message Transformation
[123] vgl. [Seub/2018], S.188
[124] vgl. [SAP/2022], Chapter Routing and Storing
[125] vgl. [Seub/2018], S.189
[126] vgl. [Wood/2015], S.453-456

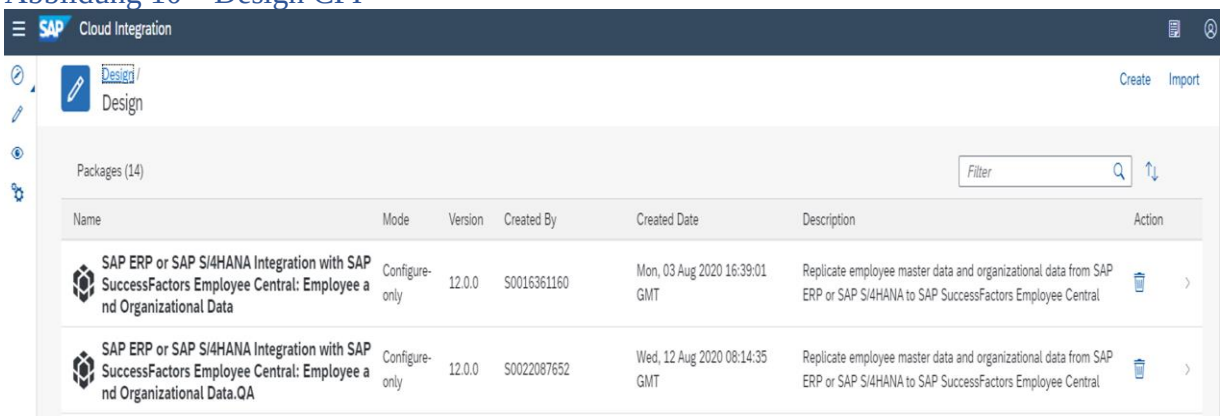
Abbildung 9 – Discover CPI



Quelle der Abbildung: Eigene Aufnahme

Individuell angelegte Integration Flows und Mappings oder die hinzugefügten vordefinierten Integrationsszenarien aus Discovery befinden sich im Bereich Design[127]. Hier hat der Anwender eine Import- und Export-Funktion von iFlows usw., um diese in anderen Paketen/Szenarien wiederzuverwenden (Abbildung 10).

Abbildung 10 – Design CPI



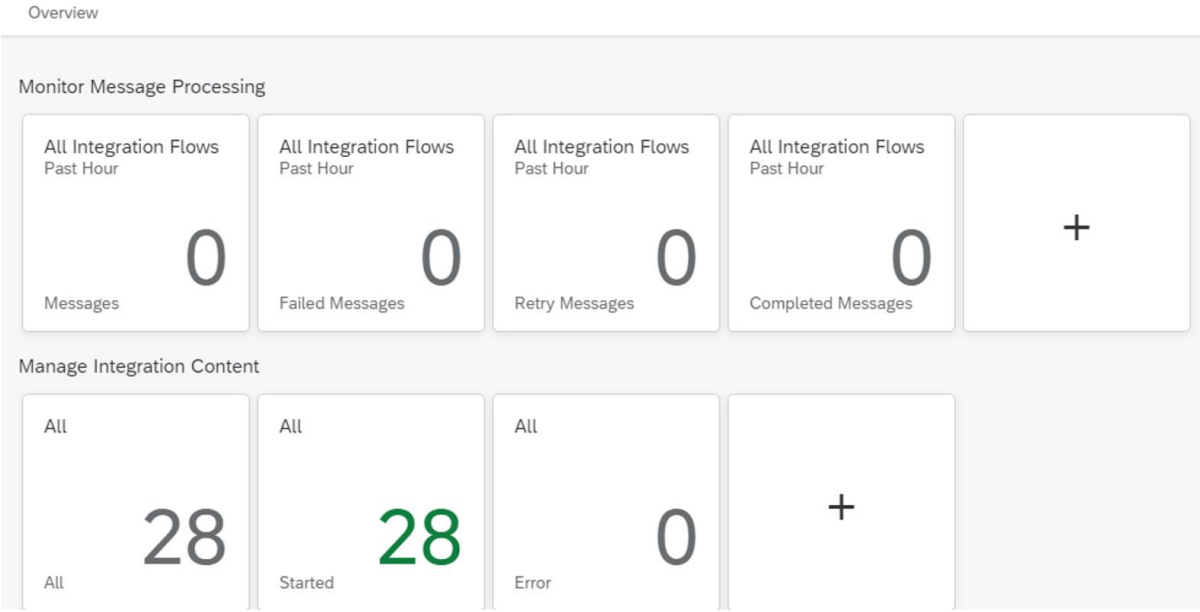
Quelle der Abbildung: Eigene Aufnahme

[127] vgl. [Wood/2015], S.456 f.

Über die Auswahl eines Pakets gelingt der Nutzer unter Artefacts in den Entwicklungsbereich. Die einzelnen Operators können per Listenauswahl in die freie Entwicklungsoberfläche hinzugefügt und über Drag and Drop miteinander verbunden werden, um so einen Flow zu erstellen. Mit einem Doppelklick auf einen Operator/Baustein kann dieser konfiguriert werden.

Der Bereich Monitor erlaubt es, alle Integrationen zu überwachen und Fehler schnell zu erkennen (Abbildung 11). Des Weiteren können hier Sicherheitsrelevante Credentials angelegt werden. Diese bestehen meist aus einem Benutzername und einem Passwort. Sie werden für die Authentifizierung an den externen Systemen verwendet.[128]

Abbildung 11 – Monitor CPI



Quelle der Abbildung: Eigene Aufnahme

In den Settings gibt es allgemeine Einstellungen für bspw. die Anzeigegröße von Elementen. Da die iPaaS von SAP Mandantenfähig ist, nutzen viele Unternehmen die Option eine Entwicklungs-, Test- und Produktivumgebung auf derselben Infrastruktur einzurichten. Für die Verwaltung dieser Instanzen wird das SAP BTP Cockpit genutzt.

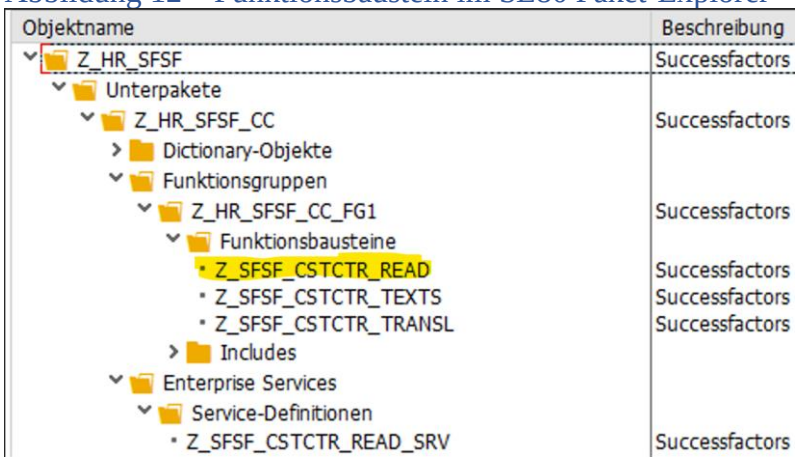
[128] vgl. [Wood/2015], S.460

8.2 (Daten-)Integration am Beispiel von SAP HCM und SuccessFactors

SAP hat bereits vor einigen Jahren das Wartungsende für ihre Kernanwendung, dem SAP ERP, angekündigt. Nach dem Jahr 2027 wird der offizielle Support eingestellt und Kunden haben die Möglichkeit eine Extended-Wartung bis Ende 2030 abzuschließen.[129] Diese Nachrichten haben viele Unternehmen dazu bewegt, auf modernere Produkte umzusteigen und den Umschwung für sich zu nutzen. Besonders im Bereich Human Resources, der in im Laufe der Zeit immer mehr an Bedeutung gewonnen hat, ist der Umstieg von SAP Human Capital Management (HCM) auf SAP-SuccessFactors zu beobachten. Im SAP HCM werden Unternehmensstrukturen, Mitarbeiterstrukturen und Organisationsstrukturen gepflegt[130]. SuccessFactors bietet ebenfalls viele Module zur Stammdatenverarbeitung von Mitarbeiterdaten wie zum Beispiel Entgeltabrechnung, Zeitwirtschaftsmodule oder Learningmodule als Cloud-Option. Als Integrationsszenario wird nun dargestellt, wie Kostenstellendaten vom SAP HCM nach SuccessFactors mit der CPI übertragen werden, um diese auf einem moderneren System mit den Vorteilen der Cloud zu nutzen.

Bevor die Daten an die CPI zur Modifizierung an die Zielstruktur übergeben werden können, müssen die benötigten Kostenstellendaten aus dem HCM-System ausgelesen werden. Dazu wird in der SAP eigenen Entwicklungssprache Advanced Business Application Programming (ABAP) ein Funktionsstein wie in Abbildung 12 entwickelt. In der ABAP-Workbench, die mit dem Transaktionscode SE80 aufrufbar ist, wird dieser Funktionsbaustein angelegt.

Abbildung 12 – Funktionsbaustein im SE80 Paket-Explorer



Quelle der Abbildung: Eigene Aufnahme

[129] vgl. [SAPN/2020]

[130] vgl. [Wolf/2018], S.460

Durch das unterschiedliche Pflegen befinden sich die Daten meistens in verschiedenen Tabellen. Um die Daten nun lesen zu können, werden diese über Select-Statements und Filter im ABAP-Code in eine neu erstellte Zieltabelle geschrieben (Abbildung 13). Dort befinden sich nach der Ausführung des ABAP-Funktionsbausteins alle benötigen Daten.

Abbildung 13 – ABAP Ausschnitt eines Select-Befehls

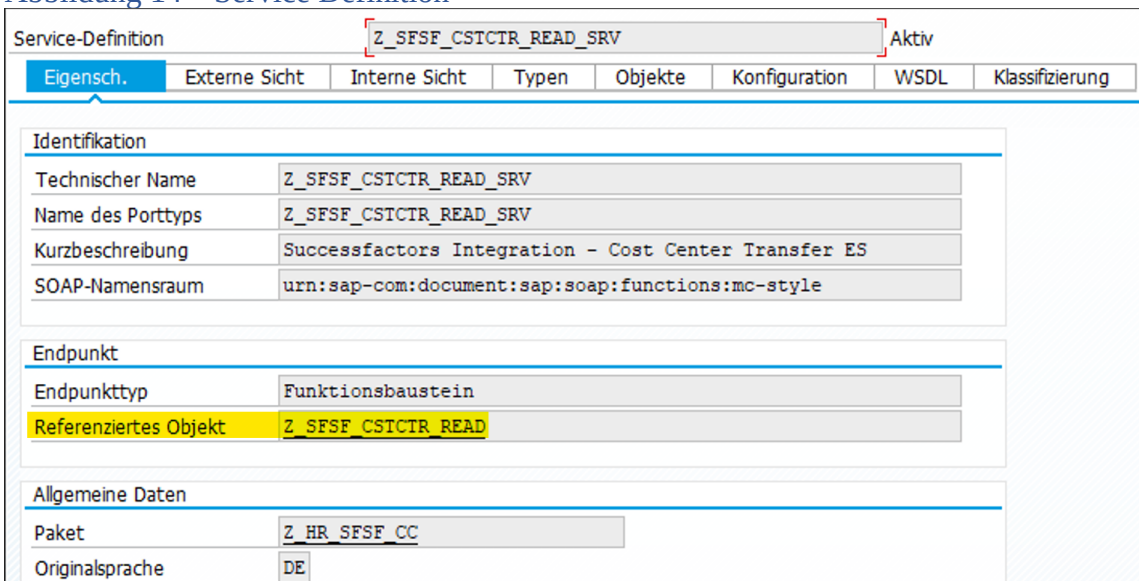
```
SELECT * FROM csks INTO TABLE lt_csks  " Kostenstellen werden gelesen
  WHERE kostl IN lt_r_kostl
   AND datbi >= lv_ftsd " Zeitscheiben berücksichtigen
   AND bkzkp EQ ''.

 IF sy-subrc = 0.
```

Quelle der Abbildung: Eigene Aufnahme

Anschließend muss für diesen Funktionsbaustein ein Enterprise Service/Webservice angelegt werden, um diesen die Fähigkeit zu geben, sich mit der CPI zu verbinden und Daten auszutauschen. Enterprise Services werden in SAP unter der Transaktion SOAMANAGER erstellt. In Abbildung 14 ist der Service für den Funktionsbaustein Z_SFSF_CUSTCTR_READ dargestellt. Der Funktionsbaustein wird als referenziertes Objekt angeben und wird ausgeführt, sobald eine Datenanfrage eingeht. Mit diesem Verfahren erhält die CPI bei jeder Anfrage die aktuell im System gepflegten Daten.

Abbildung 14 – Service Definition

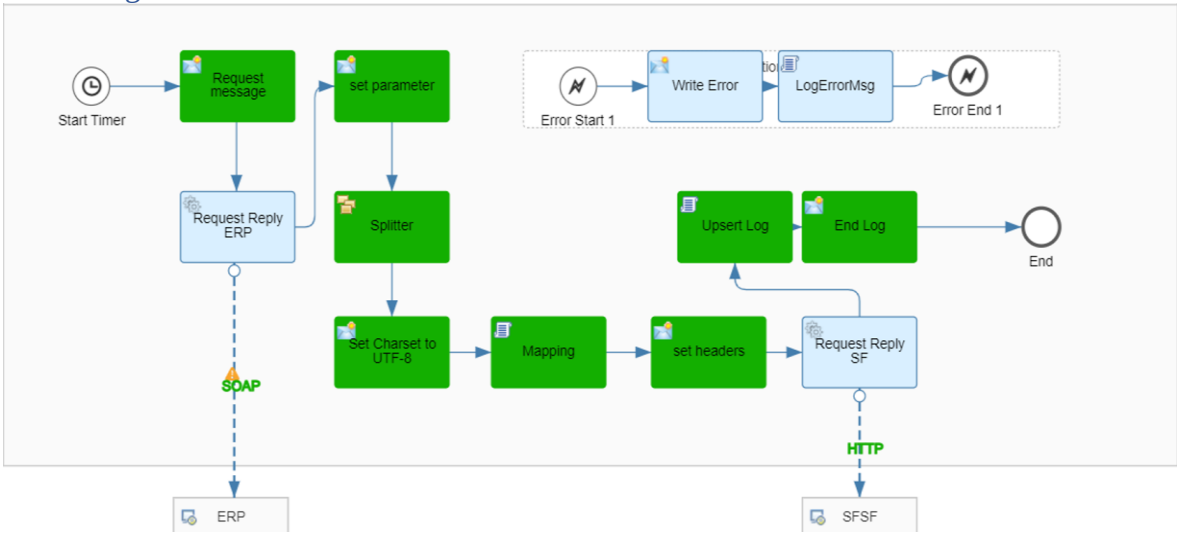


Quelle der Abbildung: Eigene Aufnahme

Wie bereits zuvor beschrieben, ist ein Webservice eine Methode der Kommunikation über das Internet. In diesem Fall wird als Protokoll SOAP verwendet. Über HTTP werden

Datenanforderungen an die Schnittstelle gesendet und die Antwort wird im XML-Format zurückgegeben. Ein weiteres Resultat des angelegten Service ist eine WSDL-Datei. Im Laufe der Integration wird die WSDL-Datei in die CPI als Ressource eingefügt, damit die CPI die Kenntnisse darüber verfügt, wie sie den Service abrufen kann bzw. wo er sich befindet. Als nächstes wird in der CPI ein Integration Flow erstellt, der die Daten vom HCM aufnimmt, verarbeitet und nach SuccessFactors (SFSF) weiterleitet. Wie in Abbildung 15 dargestellt, wird im ersten Schritt eine Datenabfrage an den zuvor erstellten Enterprise Service über SOAP gestartet.

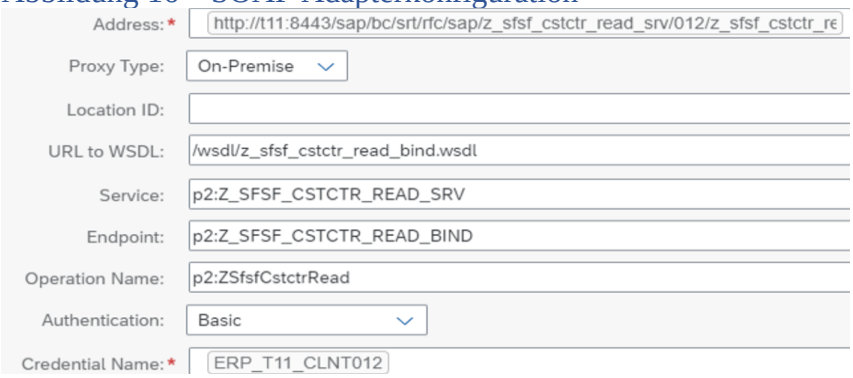
Abbildung 15 – iFlow Kostenstellen nach SF



Quelle der Abbildung: Eigene Aufnahme

Der Adapter „Request Reply ERP“ wird mit dem Reciver „ERP“ verbunden. Als Verbindungstyp wird SOAP gewählt und die Konfiguration (Abbildung 16) durchgeführt.

Abbildung 16 – SOAP Adapterkonfiguration



Quelle der Abbildung: Eigene Aufnahme

Die Daten durch den Service aus dem SAP-HCM kommen im XML-Format an. Das Zielsystem SuccessFactors benötigt die Daten jedoch im JSON-Format, um diese verarbeiten zu können. In Abbildung 17 ist ein Ausschnitt des Ausgangsformats abgebildet. Durch die WSDL-Datei hat die CPI die Kenntnisse über das Format, die Attribute, den Ort des Service und den Datentypen. Als nächstes folgt der Schritt „set parameters“, bei den Optionen eingerichtet werden, welche Daten geschickt werden sollen.

Abbildung 17 – Quellformat aus SAP-HCM

```
<n0:ZSfsfCstctrReadResponse xmlns:n0="urn:sap-com
  <ET_CSTCTR>
    <item>
      <MANDT>002</MANDT>
      <KOSTL>0000730003</KOSTL>
      <KOSTX>Entwicklung Frankrei</KOSTX>
      <KOSTL_CONCAT>000300000730003</KOSTL_CONCAT>
      <KOKRS>0003</KOKRS>
      <BUKRS>0003</BUKRS>
      <PERSA>
        <item>
          <PERSA>0003</PERSA>
          <BEGDA>1900-01-01</BEGDA>
          <ENDDA>9999-12-31</ENDDA>
        </item>
      </PERSA>
```

Quelle der Abbildung: Eigene Aufnahme

Mit dem Splitter wird die große Nachricht in kleinere Teile aufgeteilt. Dadurch wird die Arbeitslast aufgeteilt und Performanceprobleme werden umgangen.

Da Quell- und Zielschema der Daten nicht übereinstimmen wird ein Mapping der Daten benötigt. Ein grafisches Mapping ist in diesem Falle wegen der starken Individualität des Integrationsszenarios nicht möglich. Ein Script-Operator wird verwendet, um mit Groovy (Java ähnliche Scriptsprache) das Mapping von Hand zu programmieren. In Abbildung 18 ist ein Ausschnitt des Codes zu sehen, der die Struktur der Daten dem Zielformat entsprechend aufbaut.

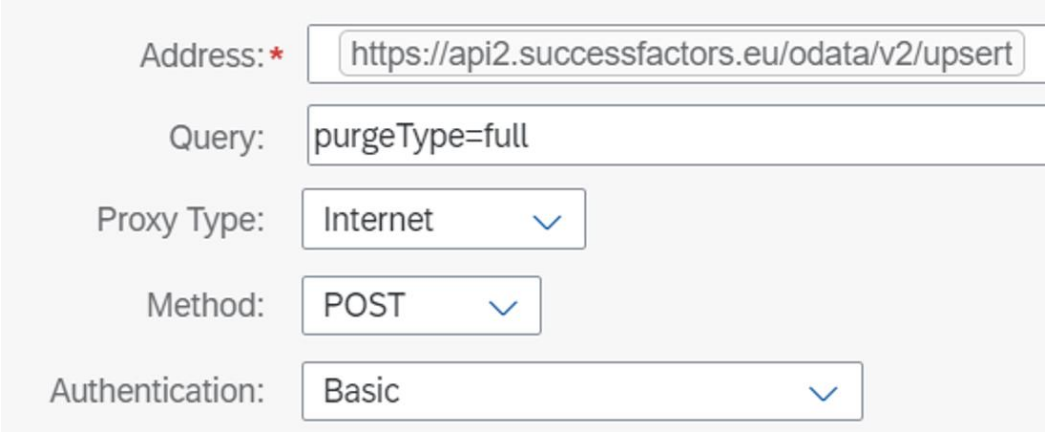
Abbildung 18 – Strukturanpassung mit Groovy

```
str1 = comma + "{ \"__metadata\": { \"uri\": \"FOCostCenter\"" + " }," +
     "\"startDate\": " + startDate + "," +
     "\"endDate\": " + endDate + "," +
     "\"externalCode\": " + externalCode + "," +
     "\"costcenterExternalObjectID\": " + extID + "," +
     "\"name\": " + name + "," +
     "\"name_defaultValue\": " + name_defaultValue + "," +
     "\"status\": " + status + "," +
    // "\"costcenterManager\": " + manager + "," +
     "\"name_en_US\": " + nameUS + "," +
     "\"name_de_DE\": " + nameDE  + "," +
     "\"cust_controllingArea\": " + contrArea + "}" + eol
    //  "\"name_zh_CN\": " + nameZH  + "," +
    //  "\"name_cs_CZ\": " + nameCZ + "," +
    //  "\"name_pt_PT\": " + namePT  + "," +
    //  "\"name_ru_RU\": " + nameRU + "," +
    //  "\"name_es_ES\": " + nameES  + "," +
```

Quelle der Abbildung: Eigene Aufnahme

Nachdem die Anpassungen im Script durchgeführt wurden, folgt ein „Request Reply SF“. Die im Script angepassten Daten werden in diesem Schritt via HTTP nach SuccessFactors geladen. In SuccessFactors wird diese Operation Upsert (Update and Insert) genannt. Für den Upload der Daten wird die „POST“-Methode in der Adapterkonfiguration (Abbildung 19) gewählt.

Abbildung 19 – Adapter für Upload nach SuccessFactors



Quelle der Abbildung: Eigene Aufnahme

Der Integrationsprozess hat die Daten aus dem SAP-HCM, welche im XML-Format in der CPI angekommen sind, nach JSON konvertiert und so cloudfähig gemacht. In Abbildung 20 ist die finale Struktur der Daten, wie sie nach SuccessFactors hochgeladen wurden, abgebildet.

Abbildung 20 – Finale Dateistruktur nach dem Prozess


```
{
    "__metadata": {
        "uri": "FOCostCenter"
    },
    "startDate": "/Date(1641030368000)/",
    "endDate": "/Date(253402249568000)/",
    "externalCode": "0000730003",
    "costcenterExternalObjectID": "00030000730003",
    "name": "0000730003",
    "name_defaultValue": "Entwicklung Frankrei",
    "status": "A",
    "costcenterManager": "",
    "name_en_US": null,
    "name_de_DE": "Entwicklung Frankrei",
    "cust_controllingArea": "0003",
    "cust_toLegalEntity": [
        {
            "__metadata": {
                "uri": "FOCompany(externalCode='0003'
                "type": "SFOData.FOCompany"
            }
        }
    ]
```


Quelle der Abbildung: Eigene Aufnahme

9. Agile Organisation

Wenn Unterhemen Cloud-Technologien einsetzen, wächst ihr Integrationsbedarf. Bei vielen Anwendungen und Systemen muss auch die Organisation umgestaltet werden, um schnell und flexibel auf neue Anforderungen zu reagieren. Dafür müssen auch Integrationen agil gestaltet werden. Denn eine Integration ist immer eine Antwort auf neue Anforderungen an Geschäftsprozesse. In diesem Kapitel werden die Grundsätze der Agilität erläutert und die agile Methode Scrum näher beleuchtet. Anschließend wird die agile Integration vorgestellt.

9.1 Grundsätze der Agilität und das agile Manifest

Agile Methoden verfolgen die Intention, dass Änderungen an Projekten jederzeit, seien es kurzfristige oder eher gegen Ende des Projektes auftretende Veränderungen, keinerlei Probleme darstellen. Das allgemeine Ziel ist es, die Softwareentwicklung auf eine flexible Ebene zu bringen, sodass Kundenwünsche, welche sich nach dem schnelllebigen und aktuellen Markt richten, ebenfalls durch Integrationen realisiert werden können.[131] Zusätzlich liegt der Sinn in der Einführung agiler Methoden dabei, eine Kultur des

[131] vgl. [Stey/2010], S.9

Lernens einzubringen. Das Ziel hierbei ist es, den tatsächlichen Entwicklungsprozess in den Vordergrund zu rücken, statt das sich der Fokus nur auf dem zu entwickelnden Produkt befindet. Menschlichen Interaktionen, besonders unter Teammitgliedern, wird eine enorme Signifikanz zugewiesen und bildet einen primären Wert einer agilen Vorgehensweise. Es ist die Aufgabe aller an einem Projekt beteiligten Personen sich damit auseinanderzusetzen, wie ihre Effektivität bezüglich der Zusammenarbeit miteinander optimiert werden kann. Um das agile Mindset zu normieren, haben sich im Jahre 2001 insgesamt 17 Vertreter unterschiedlichster Softwareentwicklungsmethoden in dem Bundestaat Utah der Vereinigten Staaten von Amerika zusammengeschlossen, um das sogenannte „agile Manifesto" auszuarbeiten. Dabei sind es diese vier Werte, unter dem das Wort „agil" durch die Vertreter der Entwicklungsmethoden definiert worden ist:

- „Individuen und Interaktionen mehr als Prozesse und Werkzeuge
- Funktionierende Software mehr als umfassende Dokumentation
- Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlungen
- Reagieren auf Veränderungen mehr als Befolgen eines Plans"[132]

Diese vier Werte stellen den grundlegenden Rahmen für agiles Arbeiten dar und sollten deshalb stets beachtet werden. Dieses Manifest beinhaltet die Mentalität, welche in allen agilen Methoden aufzufinden ist.[133] Beim ersten Aspekt werden den Individuen und Interaktionen ein höherer Wert beigemessen als die Prozesse und Werkzeuge, die bei der Entwicklung angewendet werden. Es wird ersichtlich, dass der Mensch im Vordergrund steht. Der menschliche Teil ist also ein Erfolgsfaktor eines Teams, der durch begleitende Mittel wie bspw. „Werkzeuge" unterstützt werden kann. Ein weiterer Wert, der für das Arbeiten mit agilen Praktiken von Bedeutung ist, lautet Kommunikation. Dabei geht es nicht nur speziell um die offene und ehrliche Kommunikation unter den Projektbeteiligen, also den Teammitgliedern. Auch der Kontakt zu den Kunden kann für den Erfolg entscheidend sein, da dieser den Umfang des Projektes und damit eingehend auch den Umfang der Arbeit für das Team und seine Mitglieder bestimmt. Somit wird ein weiterer Aspekt aus dem agilen Manifest aufgegriffen, der maßgeblich für den Erfolg beim Umgang mit agilen Methoden ist. Es geht um die stätige, ununterbrochene Zusammenarbeit und Kommunikation mit dem Kunden. Statt Vertragsverhandlungen mit dem Kunden zu führen, lässt man ihn Teil der Entwicklung werden. Denn nur der Kunde als Abnehmer der Entwicklung weiß, was das Entwicklerteam zu programmieren hat.[134]

---

[132] [Ecks/2012], S.14 f.

[133] vgl. [Stey/2010], S.17

[134] vgl. [WoBi/2011], S.10

Dieses Vorgehen dient zur Sicherstellung, dass im Falle der Softwareentwicklung immer ein passendes softwaretechnisches Artefakt geliefert wird. Auch auf Änderungswünsche des Kunden kann auf diesem Wege besser eingegangen werden. Zudem können Missverständnisse, welche unnötig längere Lieferzeiten schaffen, durch direkte Kommunikation und einbeziehen des Kunden in die Entwicklung bewusst vermieden werden.[135] Das Ziel, Risiken bei der Entwicklung frühzeitig zu identifizieren und dem entgegenzuwirken, wird aktiv vom Kunden unterstütz. Auf diesen Werten aufbauend haben sich im Laufe der Zeit viele Vorgehensmodelle und Projektmanagement-Frameworks wie beispielsweise Scrum geformt.

9.2 Einführung in Scrum

Bereits in den 1990er Jahren hatten Ken Schwaber und Jeff Sutherland, die zudem ihren Beitrag zum agilen Manifest geleistet haben, daran gearbeitet, sich mit der Idee zu beschäftigen, wie man Produkte vor allem schnell, aber auch flexibel entwickeln könnte.[136] Auch wenn Sie, vor allem in der Anfangszeit, allein an Möglichkeiten gearbeitet hatten, war ihre gemeinsame Idee, die über die Zeit entstand, „Scrum“. Scrum ist ein aus dem Rugby stammender Begriff, bei dem bereits kleinste Regelverstöße dazu führen, dass ein Spiel erneut gestartet werden muss.[137] Scrum, so ist es von den Entwicklern definiert, beschreibt und ist an sich kein Prozess, sondern ein agiles Managementframework, welches im Laufe der Zeit seinen Einsatz hauptsächlich in der Softwareentwicklung findet und über eindeutig festgelegte Regeln verfügt, die das Zusammenspiel der beteiligten Rollen sowie der verschiedenen Artefakte beschreiben. Dabei ist zu beachten, dass die während eines Projektes auftretenden Risiken auch mit Scrum als Framework nicht ganz unterbunden werden. Jedoch ist anzumerken, dass Scrum ein empirischer Prozess ist, wobei nicht nur die zu entwickelnden Produkte, aber auch die Arbeitsweise aller Beteiligten am Projekt immer wieder in regelmäßigen Abständen evaluiert werden, damit eingehend Fehler aufgedeckt oder von anderen offengelegt werden und bei Bedarf eine Nachjustierung erfolgt.[138] Durch dieses Vorgehen wird das Wissen aus den Erfahrungen gewonnen. Um vom angewandten Empirismus zu profitieren, müssen die Grundsätze der Transparenz, der Überprüfung und der Anpassung, die in Scrum vertreten sind, beachtet werden.[139] Im Gesamten betrachtet ist

[135] vgl. [Stey/2010], S.18
[136] vgl. [DrKS/2013], S.15
[137] vgl. [Pich/2008], S.2
[138] ebd.
[139] vgl. [DrKS/2013], S.14

Scrum ein Kulturwandel, der das Ziel verfolgt Offenheit und Vertrauen zwischen den Beteiligten in der Entwicklung von Produkten zu stärken. Die Hauptbeteiligten an einem Projekt mit Scrum ist das sogenannte Scrum-Team. Dieses besteht aus insgesamt nur drei Rollen: Dem Product Owner, dem Scrum Master und dem Entwicklerteam.[140] Die Vorteile, die durch die Anwendung von Scrum in Projekten gezogen werden können, belaufen sich darauf, dass beispielsweise eine Software schneller auf den Markt kommt (Time-To-Market), eine höhere Effizienz gewährleistet ist und als Resultat die Qualität des gelieferten Produktes steigt.[141]

9.3 Agile Integration

Wie bereits beschrieben führen agile Methoden, wenn korrekt umgesetzt, zu mehr Feedback und dadurch zu mehr Flexibilität, um auf das Marktgeschehen reagieren zu können. Agile Integrationen sind auch verteilte Integrationen. Nicht nur die Integrationsteams, sondern auch die Integrationsarchitektur muss sich diesen Umständen anpassen. In einem traditionellen Integrationsansatz wie dem Enterprise Service Bus herrscht oft eine zentrale Infrastruktur. Erhoffte Kosteneinsparungen können meist nicht eingehalten werden, da sich die Kosten für bspw. Schnittstellenentwicklungen höher als geplant herausstellen und die Wiederverwendbarkeit durch starke Individualisierungen darunter leidet. Durch den steigenden Integrationsbedarf wegen erhöhten Inbetriebnahmen neuer Cloud-Anwendungen steigt auch die Komplexität der Integrationen parallel an. Ein einziges Spezialisten-Team reicht nicht mehr aus, um die Integrationen großer Organisationen mit vielen heterogenen Systemen zu bewältigen.[142] Jedoch führen serviceorientierte Architekturen zu genau dieser Bildung von zentralen Integrationsteams, die sich zwar mit den Technologien zur Integration auskennen, aber meist nicht mit den zu integrierenden Anwendungen. Die Lösung für dieses Problem ist der dezentrale Besitz von Integrationen. Verschiedene Teams sind für jeweils bestimmte Teile der gesamten Integration zuständig. Die Teammitglieder der kleineren Teams sind interdisziplinär aufgestellt, was bedeutet, dass sie zusammen über das komplette Wissen verfügen, welches sie für die Erfüllung der Anforderungen benötigen. Neben reinen Integrationsspezialisten sind demnach auch Anwender Teil des Teams, da sie die Anforderungen klar definieren können.[143] Durch diese Anpassung muss sich auch die

[140] vgl. [Stey/2010], S.22
[141] vgl. [Pich/2008], S.7
[142] vgl. [IBMC/2018], S.6
[143] vgl. [IBMC/2018], S.8-11

Integrationsarchitektur anpassen. Da jedes Team einen bestimmten Teil der Integrationsaufgaben erfüllt, muss die zentrale Architektur aufgeteilt werden. Prinzipien einer Mikroservice-Architektur schaffen hier Abhilfe. Die Integration als Ganzes wird in kleinere Komponenten aufgeteilt, um diese skalierbar und ausfallsicherer zu gestalten.[144] Diese Art Integrationsbereitstellung bietet den Vorteil, dass die Teams auch technisch unabhängig voneinander arbeiten können. Änderungen/Anpassungen an einzelnen, einem Team zugewiesenen, Integrationsszenario können so schneller umgesetzt werden, ohne dabei andere Teile der Gesamtintegration zu beeinflussen. Dies geling durch die technische Dezentralisierung.[145] Eine Integration Platform as a Service (iPaaS) kann Unternehmen bei dieser Herausforderung unterstützen. Mandantenfähige Systeme für mehrere Teams sowie der Zugriff auf die Plattform von jedem Ort gehen mit den agilen Prinzipien wie Flexibilität einher. So lohnt es sich bei wachsender Anzahl von Systemen, die Integriert werden müssen, auf den agilen Integrationsansatz aufzusteigen.

10. Fazit und Ausblick

Wegen der Inbetriebnahme neuer Cloud-Anwendungen steigt der Integrationsbedarf von Unternehmen. Cloud-Technologien bieten durch ihre simple Skalierbarkeit für Kunden die Möglichkeit effizient Kosten zu sparen. An den jeweiligen Bedarf gerichtete Preismodelle machen Cloudoptionen für viele Anwender immer attraktiver. Unternehmen können ihre eigene IT-Infrastruktur so immer weiter abbauen und lösen sich von bspw. Wartungen für die Hardware. Doch der Umstieg in die Cloud ist ein Prozess, welcher sich über einen längeren Zeitraum ziehen kann, da alte Legacy-Systeme noch immer geschäftskritische Prozesse unterstützen. In der Zwischenzeit entsteht eine hybride IT-Landschaft mit On-Premise- und Cloud-Anwendungen, welche Informationen untereinander austauschen müssen, um Informationssilos zu vermeiden und die Automatisierung von Geschäftsprozessen voranzutreiben. Traditionelle Integrationsansätze wie der Enterprise Service Bus (ESB) kommen hier an ihre Grenzen. Der ESB ist lokal in der eigenen IT-Infrastruktur implementiert und gewährleistet damit die Datensicherheit, da diese das Unternehmen während der Integration nicht verlassen. Doch der ESB ist zu unflexibel. Zudem wächst die Komplexität und Last mit zunehmender Anzahl an Integrationsobjekten. Alle Anwendungen werden über einen zentralen Hub, dem sogenannten Bus, miteinander verbunden. Genutzte Schnittstellen

[144] vgl. [IBMC/2018], S.7
[145] vgl. [IBMC/2018], S.12

basieren häufig auf SOAP oder REST. Durch den zentralen Hub entsteht ein Single Point of Failure, was die Zuverlässigkeit eines ESB einschränkt. Aus diesen Gründen werden ESB-Integrationen heutzutage fast ausschließlich für die Integration von On-Premise Anwendungen eingesetzt. Integration Platform as a Service als eine cloudbasierte Plattform bietet ESB-Funktionalitäten wie Datentransformationen und die Erstellung eventbasierter Integrationsszenarien mit den allgemeinen Vorteilen der Cloud in einer grafischen Bedienoberflächen. IPaaS stellt viele bereits vorgefertigte und konfigurierbare Adapter für eine einfache Erstellung von Integration Flows, auch für Citizen Integrators, bereit. Für die Infrastruktur und Updates mit neuen Funktionen ist der iPaaS-Anbieter zuständig. Mit iPaaS-Lösungen werden sowohl On-Premise Anwendungen als auch Cloud-Anwendungen beliebig miteinander für die Kommunikation verbunden. Ein führender iPaaS-Anbieter am Markt ist z.B. SAP und ihre Cloud Platform Integration Suite, mit welcher in Kapitel 8 ein praktisches Integrationsszenario entwickelt wurde. Um das volle Potenzial ausschöpfen zu können, führen Unternehmen die agile Integration ein. Dabei werden Integrationsgebiete unter mehreren, interdisziplinär aufgestellten Teams aufgeteilt. Auch auf Architekturebene werden die Integration wie Mikroservices aufgeteilt. Durch dieses Vorgehen erreichen Unternehmen die maximale Stufe der Flexibilität, da Unabhängigkeit zwischen den Systemen herrscht. Bei Anpassungen durch neue Marktanforderungen können die betroffenen Integrationsteams diese, ohne andere Integrationen zu beeinflussen, schnell umsetzen. Die dadurch resultierende kürzere Time-to-Market Zeit gibt Unternehmen den Vorsprung konkurrenzfähig zu bleiben. Eine iPaaS-Lösung unterstützt dieses Vorgehen mit seinen Vorteilen.

Es bleibt weiterhin interessant zu beobachten, wie sich die genutzten Technologien und Ansätze zur Integration weiterentwickeln. Da sich Cloud-Anwendungen bereits etabliert haben und immer häufiger genutzt werden, wird der Integrationsbedarf auch in Zukunft weiter steigen. Der Fokus wird sich von On-Premise, stark in Richtung Cloud-to-Cloud Integrationen wandeln. IPaaS-Lösungen werden dann, wie auch in den vergangenen Jahren, erneut Umsatzrekorde brechen.

Literaturverzeichnis

[Beng/2014] Bengel, Günther: Grundkurs Verteilte Systeme, Wiesbaden: Springer Fachmedien, 2014

[Bitk/2015] Bitkom Research GmbH: Cloud Monitor 2015, Bitkom, 2015, online im Internet: URL: https://www.bitkom.org/sites/default/files/file/import/Cloud-Monitor-2015-KPMG-Bitkom-Research.pdf [Stand 28.05.2022]

[BöB+/2018] Bögelsack, André; Baader, Galina; Prifti, Loina; Zimmermann, Ronny; Krcmar, Helmut: SAP-Systeme in der Cloud, Bonn: Rheinwerk Publishing, 2016

[BöD+/2014] Bönnen, Carsten; Drees, Volker; Fischer, Andre; Heinz, Ludwig; Strothmann, Karsten: OData und SAP Gateway, Bonn: Galileo Press, 2014

[Börn/2018] Börnert, Carsten: An Architectural and Practical Guide to IBM Hybrid Integration Platform, USA: International Business Machines Corporation, 2016, online im Internet: URL: https://www.redbooks.ibm.com/redbooks/pdfs/sg248351.pdf [Stand 24.06.2022]

[BoYa/2012] Boci, Lin; Yan, Chan: COMPARISION BETWEEN JSON AND XML IN APPLICATIONS ON AJAX, China: IEEE, 2012, online im Internet: URL: https://ieeexplore.ieee.org/document/6394535 [Stand 30.06.2022]

[Code/2021] Codeless Platforms: What is iPaaS? A Complete Guide to Integration Platform Service, United Kingdom: Codeless Blog, 2021 (abgerufen am 24.06.2022 https://www.codelessplatforms.com/blog/what-is-ipaas/)

[DrKS/2013] Drähter, Rolf; Koschek, Holger; Sahling, Carsten: Scrum kurz&gut, Beijing; Cambridge; Farnham; Köln; Sebastopol; Tokyo: O‘ Reilly, 2013

[EbW+/2017] Ebert, Nico; Weber Kristin; Koruna, Stefan: Integration Platform as a Service; Wiesbaden: Springer Fachmedien GmbH, 2017, online im Internet: URL: https://link.springer.com/content/pdf/10.1007/s12599-017-0486-0.pdf [Stand 28.06.2022]

[Ecks/2012] Eckstein, Jutta: Agile Softwareentwicklung in großen Projekten Teams, Prozesse und Technologien – Strategien für den Wandel im Unterhemen, Heidelberg: dpunkt-Verl., 2012

[GHIN/2019] Ghinet, Bogdan: Enterprise Application Integration (EAI), Accesa, 2019, online im Internet: URL: https://medium.com/accesa/enterprise-application-integration-eai-df3e0d660482 [Stand 14.06.2022]

[Hark/2018]: Harkut, Dinesh: Cloud Computing – Technology and Practices, London: IntechOpen, 2018, online im Internet: URL: http://dx.doi.org/10.5772/intechopen.81247 [Stand 02.06.2022]

[Hase/2012] Haselmann, Till: Cloud-Services in kleinen und mittleren Unternehmen Nutzen, Vorgehen, Kosten; Münster: DBIS, 2012

[Herd/2006] Herden, Sebastian: Software-Architekturen für das E-Business, Berlin;Heidelberg: Springer-Verlag, 2006

[Höll/2011] Höllwarth, Tobias: Cloud Migration; Heidelberg: mitp, 2011

[HyKo/2021] Hyrynsalmi, Sonja; Koskien, Kari: Towards the utilization of Cloud-based Integration Platforms, United Kingdom: IEEE, 2021, online im Internet: URL: https://ieeexplore.ieee.org/document/9570235 [Stand 24.06.2022]

[IBMC/2018] IBM Cloud: Agile Integration: Ein moderner Ansatz zur Hybridintegration in Unternehmen, IBM Cloud, 2018, online im Internet: URL: https://www.ibm.com/downloads/cas/ZMQR8VY4 [Stand 02.07.2022]

[IBMC/2021] IBM Cloud Education: iPaaS (Integration-Platform-as-a-Service), IBM Cloud Education, 2019, online im Internet: URL: https://www.ibm.com/de-de/cloud/learn/ipaas#toc-ipaas-vort-_Uxn7sGv [Stand 02.07.2022]

[Info/o.J.] Informatica GmbH: Die Anleitung des Cloud-Architekten zu iPaaS, o.J., online im Internet: URL: https://www.cloudcomputing-insider.de/hybride-datenintegration-eine-anleitung-d-35734/ [Stand 24.06.2022]

[Kaib/2002] Kaib, Michael: Enterprise Application Integration Grundlagen, Integrationsprodukte und Anwendungsbeispiele, Wiesbaden: Springer Fachmedien, 2002

[Kunt/2021] Kuntz, Martin: iPaaS vs. ESB: Welche Integrations-Lösung ist die beste für Ihr Unternehmen?, Seeburger Blog, 2021, online im Internet: URL: https://blog.seeburger.com/de/ipaas-vs-esb-welche-integrations-loesung-ist-die-beste-fuer-ihr-unternehmen/ [Stand 02.07.2022]

[Mate/2012] Matei, Marian: iPaaS: Different Ways of Thinking, ScienceDirect, 2012, online im Internet: URL: https://www.sciencedirect.com/science/article/pii/S2212567112002791?ref=cra_js_challenge&fr=RR-1 [Stand 03.07.2022]

[Micr/2022] Microsoft: Welcome to OData, USA: Microsoft, 2022, online im Internet: URL: https://docs.microsoft.com/en-us/odata/overview [Stand 30.06.2022]

[Müll/2005] Müller, Joachim: Workflow-based Integration, Heidelberg: Springer-Verlag, 2005

[NeL+/2021] Neifer, Thomas; Lawo, Dennis; Bossauer, Paul; Gadatsch, Andreas: Decoding IPaaS: Investigation of User Requirements for Integration Platforms as a Service, Siegen: Verbraucherinformatik Research Group, 2021, online im Internet: URL: https://pdfs.semanticscholar.org/6b3c/5ffea3a04446ed8ada6159d58c102ceb966d.pdf [Stand 03.07.2022]

[NIST/2011] National Institute of Standards and Technology: The NIST Definition of Cloud Computing https://csrc.nist.gov/publications/detail/sp/800-145/final (aufgerufen am 30.05.2022)

[PaKh/2018] Pathak, Ramesh; Khandelwal Pankaj: A Model for Hybrid Cloud Integration With a Case Study for IT Service Management (ITSM), India: IEEE, 2018, online im Internet: URL: https://ieeexplore.ieee.org/document/8332564 [Stand 24.06.2022]

[Papa/2012] Papazoglou, Michael: Web Services & SOA Principles and Technology Second Edition; England: Pearson Education Limited, 2012

[Pich/2008] Pichler, Roman: Scrum Agiles Projektmanagement erfolgreich einsetzen, Heidelberg: dpunkt-Verl., 2008

[SAP/2022] SAP: What Is SAP Cloud Integration?, 2022, online im Internet: URL: https://help.sap.com/docs/CLOUD_INTEGRATION/368c481cd6954bdfa5d0435479fd4eaf/e12c09cc8e9b4574b092d8964b049ce6.html?locale=en-US [Stand 04.07.2022]

[SAPN/2020] SAP NEWS: SAP gibt erweiterte Innovationszusage für SAP S/4HANA und bietet Klarheit und Wahlmöglichkeiten für SAP Business Suite 7, 2020, online im

Internet: URL: https://news.sap.com/germany/2020/02/wartung-s4hana-sap-business-suite-7/ [Stand 04.07.2022]

[SCHE/2019] Scheuch, Rolf: Cloud-basierte Integration (iPaaS), Whitepaper, Opitz Consulting, 2019, online im Internet: URL: https://www.opitz-consulting.com/sites/default/files/Kompetenz/Whitepaper/PDF/whitepaper-cloud-basierte-integration-neu.pdf [Stand 06.06.2022]

[Seub/2018] Seubert, Holger: SAP Cloud Platform, Bonn: Rheinwerk Publishing, 2018

[Stey/2010] Steyer, Manfred: Agile Muster und Methoden: agile Softwareentwicklung maßgeschneidert, Frankfurt am Main: Entwickler.press, 2010

[TeVo/2011] Terplan, Kornel; Voigt; Christian: Cloud Computing, Heidelberg: mitp-Verlag, 2011

[ThGu/2021] Thoo, Eric; Guttridge, Keith: Magic Quadrant for Enterprise Integration Platform as a Service, Gartner, 2021, online im Internet: URL: https://www.gartner.com/doc/reprints?id=1-27KRBNJG&ct=211004&st=sb?mkt_tok=ODY3LU1BTy02MzQAAAGFYbOtxT-Lz2pDd6fjGQD3VexXpOxF_3m298WrPbBGIz69kKlbJHB9bnJzx1VPN53mPti9U9UHMhL4bFRGJi0onDBvZZJZ9224yh-SZhxdkdxOdDBFtog [Stand 03.07.2022]

[Tilk/2011] Tilkov, Stefan: REST und HTTP Einsatz der Architektur des Web für Integrationsszenarien, Heidelberg: dpunkt.verlag, 2011

[VoHa/2012] Vossen, Gottfried; Haselmann, Till: Cloud Computing für Unternehmen Technische, wirtschaftliche, rechtliche und organisatorische Aspekte; Heidelberg: dpunkt.verlag, 2012

[WoBl/2011] Wolf, Henning; Bleek, Wolf-Gideon: Agile Softwareentwicklung: Werte, Konzepte und Methoden, Heidelberg: dpunkt-Verl., 2011

[Wolf/2018] Kanngießer, Wolf: Schnelleinstieg in SAP HCM, Gleichen: Espresso Tutorials, 2018

[Wood/2018]: Wood, James: SAP HANA Cloud Plattform, Bonn: Rheinwerk Publishing, 2015

[WuTa/2010] Wu, Jieming; Tao, Xiaoli: Research of Enterprise Application Integration Based-on ESB, Shenyang: IEEE, 2010

Eidesstattliche Versicherung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Dietzenbach, 17.07.2022

Ort, Datum

Unterschrift