Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Grzegorz Markiewicz

Design von Schnittstellen für Datenmigration zwischen CRM und ERP Systemen

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Grzegorz Markiewicz

Design von Schnittstellen für Datenmigration zwischen CRM und ERP Systemen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Ulrike Steffens
Zweitgutachter: Prof. Dr. Stefan Sarstedt

Eingereicht am: 19. Dezember 2015

Grzegorz Markiewicz

Thema der Arbeit
Design von Schnittstellen für Datenmigration zwischen CRM und ERP Systemen

Stichworte
Unternehmensanwendungsintegration, Customer-Relationship-Management, Enterprise-Ressource-Planning, Geschäftsprozesse, Schnittstelle, Ereignisgesteuerte Prozesskette

Kurzzusammenfassung
Anwendungslandschaften wachsen heutzutage mit steigender Geschwindigkeit und werden somit immer komplexer. Deswegen ist die reibungsfreie Kommunikation einzelner Systeme untereinander von höchster Wichtigkeit. Diese Arbeit setzt sich mit dem Begriff der Integration auseinander. ERP-Systeme und CRM-Systeme werden untersucht und ihre Funktionalitäten abgegrenzt. Der Aufbau von Schnittstellen zwischen zwei Anwendungen wird erläutert. Darüber hinaus wird ein Geschäftsprozess analysiert, um daraufhin eine angepasste Schnittstellensoftware entwickeln zu können. Schließlich wird die besagte Software implementiert.

Grzegorz Markiewicz

Title of the paper
Design of Interfaces for datamigration between CRM and ERP systems

Keywords
Enterprise-Application-Integration, Customer-Relationship-Management, Enterprise-Ressource-Planning, Business process, Interface, Event-driven process chain

Abstract
Application environments nowadays grow with increasing speed and are becoming more complex. Therefore, the error-free communication of individual systems with each other is of paramount importance. This work discusses the concept of integration. ERP-systems and CRM-systems are examined and their scope is defined. The structure of interfaces between two applications is described. In addition, a business process is analyzed in order to develop an according interface software. Finally, the aformentioned software is implemented.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Vorwort

Anwendungslandschaften wachsen heutzutage mit steigender Geschwindigkeit und werden somit immer komplexer. Deswegen ist die reibungsfreie Kommunikation einzelner Systeme untereinander von höchster Wichtigkeit. Hinsichtlich der Realisierung dieser Schnittstellen eröffnen sich mehrere Optionen, die jeweils ihre eigenen Vor- und Nachteile besitzen. Hinzukommen Einschränkungen, die sich aus dem Aufbau der Softwarelandschaft sowie aus der Implementierung der einzelnen Systeme ergeben. Lösungen, die in der Vergangenheit des Unternehmens als korrekte Entscheidung angesehen wurden, können sich im Laufe der Zeit und der damit verbundenen Expansion des Gesamtsystems als Fehler herausstellen. Genau aus diesem Grund ist eine korrekte Auswahl unter der Berücksichtigung der Geschäftsprozesse des Unternehmens ungemein wichtig.

Einen besonderen Stellenwert in heutigen IT-Systemen hat die Schnittstelle zwischen den CRM- und ERP-Systemen. Zwar existieren Softwarelösungen, die CRM- und ERP-Komponenten in einer Applikation vereinigen, aber dennoch betreiben viele Unternehmen voneinander getrennte Systeme, da diese historisch mit der Firma gewachsen sind, um sich von einem Anbieter weniger abhängig zu machen oder weil eine Eigenentwicklung Wettbewerbsvorteile verspricht [Gronau (2012)]. Diese beiden Systeme haben unterschiedliche Aufgabenbereiche. CRM-Systeme arbeiten sehr kundennah. Sie beinhalten Funktionen für Marketing, Vertrieb, Reklamation und Wartung. Sie speichern die Kontaktdaten der Kunden und Leads, also von potentiellen Kunden/Interessenten, wohingegen ERP-Systeme ausschließlich die Daten der Realkunden, also der Kunden die bereits eine Auftrag ausgestellt haben, festhalten. ERP-Systeme verwalten die Unternehmenseigenen Ressourcen und speichern deswegen die Kundendaten erst, sobald diese mit einen konkreten Auftrag in Verbindung stehen. Die ERP-Systeme managen die Ressourcen, z.B Zulieferer und Arbeitskräfte, die für die Produktion von Bedeutung sind. Dennoch sind Teilmengen der Daten zwischen beiden Systemen identisch. So muss einem potentiellen Kunden ein konkretes Angebot zukommen gelassen werden und ein Lead

aus dem CRM-System muss nach einer erfolgreichen Auftragserstellung in das ERP-System eingetragen werden. Aufgrund dessen ist eine Betrachtung der konkreten Geschäftsprozesse und eine dementsprechende Implementierung der Schnittstelle zwischen beiden Systemen für den reibungsfreien Betrieb in einem Großunternehmen essentiell.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist es, eine konkrete Schnittstelle zwischen den CRM- und ERP-Systemen zu entwickeln und implementieren. Als Vorarbeit soll ein bestehender Geschäftsprozesses analysiert und unter der Berücksichtigung der unterschiedlichen Anforderungen an die beiden Systeme untersucht werden. Dies soll die wichtigsten Aspekte des Geschäftsprozesses im Hinblick auf die Überschneidung der Daten beider Systeme herauskristallisieren.

1.3 Gliederung der Arbeit

Es werden zuerst die Grundlagen der Schnittstellentypen erörtert. Dabei wird der Begriff „Schnittstelle“ im Rahmen der Anwendungslandschaft abgegrenzt. Anschließend werden die verschiedenen Schnittstellentypen sowie deren Aufbau erläutert.

Anschließend werden die Kernaspekte von CRM- und ERP-Systemen vorgestellt und ihre Funktionalitäten abgegrenzt. Der weitere Teil der Analyse umfasst die Auseinandersetzung mit einem ausgewählten Geschäftsprozess. Hinzu kommt eine Vorstellung von Methoden für die Modellierung von Geschäftsprozessen. Anschließend eine Separation der Daten die für beide Systeme relevant sind. Die Frage nach der Bedeutung von Geschäftsprozessen für die Vorgänge in der Wirtschaft und der damit konkret verbundenen Folgen für die Umsetzung von Software in Unternehmen werden erläutert. Es wird eine Analyse von Daten aus CRM-Systemen vorgenommen, um sie auf ihren Informationsgehalt bezüglich der ERP-Systeme zu untersuchen, und um die Anforderungen an unterschiedliche Schnittstellentypen abzugrenzen.

Es wird dann anhand eines konkreten Geschäftsprozesses eine Analyse zur Klärung der Frage vorgenommen, welcher Synchronisationstyp für die Implementierung unserer Schnittstelle am sinnvollsten ist. Das hier verwendete ERP-System ist nur ein Mockup. Dennoch betrachtet man die Problematik, als ob man es mit zwei reellen Systemen zu tun hätte. Das in der Arbeit verwendete System „Vtiger“ wird kurz vorgestellt, um im Anschluss die Software der Schnittstelle zu präsentieren und ihre Funktionsweise zu erklären. Zum Ende werden die Ergebnisse der Arbeit bewertet.

2 Grundlagen

Das Schnittstellendesign zwischen CRM und ERP-Systemen ist ein sehr umfangreiches Feld. Im Zuge dieses Grundlagenkapitels werden wir uns zunächst mit dem Begriff der Geschäftsprozesse auseinandersetzen, da diese die Basis unserer Analyse darstellen. Wir werden im Anschluss daran jeweils CRM- und ERP-Systeme erörtern. Die aus diesem Vorgang erlangten Erkenntnisse, werden einen Überblick für den Sachverhalt verleihen. Anschließend wird der Funktionsumfang der beiden Systeme Informationen liefern, welche Daten zwischen den Systemen ausgetauscht werden müssen. Im Rahmen der Betrachtung der ERP-Systeme wird darüber hinaus auf den Begriff der Integration im Kontext der Vereinigung eingegangen. Dies geschieht um die treibende Kraft der Anwendungsintegration hervorzuheben. Schließlich wird auf die Enterprise-Application-Integration eingegangen und währenddessen die Integration im Sinne der Kopplung zweier Anwendungssysteme näher betrachtet. Dadurch wird ein Überblick über verschiedene Realisierungsarten und Strukturen von Schnittstellen gewonnen.

2.1 Geschäftsprozesse

2.1.1 Begriffsabgrenzung

In der Literatur finden sich verschiedene Definitionen von Geschäftsprozessen. Die Kernbedeutung eines Geschäftsprozesses ist jedoch sehr deutlich. Aus diesem Grund halten wir die Begriffsabgrenzung kompakt. Der Prozessbegriff wird von [Becker u. a. (2012)] an einem betriebswirtschaftlichen relevanten Objekt festgemacht. Eine zeitlich und inhaltlich abgeschlossene Folge von Aktivitäten die zur Bearbeitung dieses Objektes notwendig wird als Prozess bezeichnet.

Die Definition eines Geschäftsprozesses gemäß [Leimeister (2015)] lautet: „Als Geschäftsprozess ist eine zielgerichtete zeitlich-logische Folge oder Vorgangkette von Tätigkeiten (andere Bezeichnungen: Aktivitäten, Geschäftsvorgänge) definiert, die für das Unternehmen einen Beitrag zu Wertschöpfung leistet beziehungsweise aus der Unternehmensstrategie abgeleitet ist. In der Regel sind Prozesse am Kunden orientiert, das heißt, dass sie auch für den Kunden

einen Wert schaffen. “

Des weiteren beschreibt [Leimeister (2015)] die Geschäftsprozesse als Basis für die Wertschöpfung eines Unternehmens. Er geht auf die Merkmale und Strukturierung eines Geschäftsprozesses ein. Jeder Geschäftsprozess besitzt einen definierten Anfang, den sogenannten „Auslöser“ und ein Ergebnis. Ein Geschäftsprozess zeichnet sich darüber hinaus dadurch aus, dass er kein einmaliges Ereignis ist, sondern eine Aufgabe innerhalb eines Unternehmens darstellt, die immer wiederholt wird. Als Beispiel kann das Verkaufsgespräch in einem Callcenter zwischen Verkaufspersonal und einem Kunden genommen werden, dass im späteren Verlauf dieser Arbeit analysiert wird.

2.1.2 Prozesskategorien und Typisierung

Getreu [T. Davenport] lassen sich Geschäftsprozesse grundsätzlich in drei Dimensionen qualifizieren. Diese sind die Einheit, das Objekt und die Aktivität.

Die Organisationseinheit unterscheidet drei Typen oder Ebenen [vgl. T. Davenport], die die Tragweite eines Geschäftsprozesses verdeutlichen. Die unterste Ebene ist individuelle Ebene. Geschäftsprozesse auf dieser Ebene werden meist durch einen einzelnen Angestellten durchgeführt. Als Beispiel könnte ein Kundengespräch, das zum Verkauf einer Ware führt, dienen. Funktionsübergreifende Geschäftsprozesse greifen über mehrere Bereiche eines Unternehmens. Bei ihnen arbeiten mehrere Angestellte aus verschiedenen Gebieten zusammen um ein neues Produkt zu entwickeln oder führen eine andere Tätigkeit aus, die einen Mehrwert für das Unternehmen erzeugt und auf die Kompetenzen aus verschiedenen Bereichen konzentriert. Auf der letzten Ebene sind die Unternehmensübergreifenden Geschäftsprozesse. Ein typisches Beispiel für einen solchen Vorgang ist die Bestellung der zur Produktion benötigten Materialien von einem Zulieferer.

In der zweiten Dimension wird zwischen physischen und informationellen Prozessen unterschieden [T. Davenport]. Bei physischen Geschäftsprozessen wird ein konkreter Gegenstand erstellt, während bei informationellen die immateriellen Prozesse, wie der Strategieentwicklungsprozess, im Vordergrund stehen [Koch (2015)]. Gemäß [Koch (2015)] wird es in den modernen Unternehmen immer schwieriger Geschäftsprozesse in diese Kategorie einzuordnen, weil heutzutage jeder materielle Geschäftsprozess von einem IT-Prozess unterstützt wird.

Die letzte Dimension unterscheidet zwischen zwei Typen von Prozessen, den Operativen- und den Steuerungsprozessen [T. Davenport]. Die operativen Geschäftsprozesse sind tägliche, routinierte Unternehmensziele wie der Verkauf des von der Firma vertriebenen Produktes. Zu Steuerungsprozessen zählen alle Prozesse, die eine Verbesserung der Planung und der Kontrolle der Unternehmensabläufe als Ziel haben.

Daraus resultierend teilt [Koch (2015)] Geschäftsprozesse grundsätzlich in die drei Kategorien. Am Anfang befinden sich die Kernprozesse, die direkt an der Wertschöpfungskette beteiligt sind. Die Unterstützungsprozesse, die die Kernprozesse beim Ablauf unterstützen, und die Führungsprozesse, welche den Unterstützungsprozessen ähneln, konzentrieren sich allerdings auf gesamt unternehmensstrategische Ziele.

Hinsichtlich dieser drei Klassifizierungen geht [Leimeister (2015)] nochmals auf die Wichtigkeit der einzelnen Typen ein. Er stellt die Kernprozesse als die essentiellen dar, da sie für den unmittelbaren Kundennutzen verantwortlich sind. Für Kernprozesse bezieht er sich auf Produkte und Services, die direkt vom Kunden genutzt werden. Er stellt diese Prozessart über die Support und die Managementprozesse. Die Supportprozesse haben eine unterstützende Natur. Nach [Leimeister (2015)] sind sie für die Dokumentation des Ablaufs und deren Unterstützung verantwortlich. Ein Beispiel hierfür sei die Buchhaltung in einer Firma. Die Letzten sind die Managementprozesse. Da sie nicht direkt am Wertschöpfungskette oder ihrer Unterstützung beteiligt sind, ist ihre Priorität am niedrigsten. Dennoch sollte man sie vor allem in Großkonzernen nicht unterschätzen, da in diese Kategorie die Geschäftsprozesse wie das Qualitätsmanagement und Risikomanagement fallen.

2.1.3 Prozessmanagement

Grundsätzlich beschäftigt sich Prozessmanagement mit der Steigerung der Qualität und Produktivität von Prozessen innerhalb eines Unternehmens durch die Organisation, Planung und Steuerung von Ressourcen [Becker u. a. (2012)]. Zudem gehören auch die Dokumentation und andere Teilbereiche, die aber alle das selbe Ziel der Qualitätssteigerung haben. Dadurch können Produkte effizienter und flexibler Produziert werden, was letztendlich zur Senkung der Produktionskosten und damit resultierenden Ertragssteigerung führt.

Im Rahmen der Anwendungsintegration ist das Prozessmanagement eine Komponente, die das Zusammenspiel einzelner Anwendungen und somit die Prozessintegration ermöglicht. Es ist für die Modellierung, Überwachung, Steuerung und die Durchführung von Geschäftsprozessen innerhalb einer Anwendungssystem Infrastruktur verantwortlich. Die in der Anwendungsland-

schaft integrierten Anwendungen implementieren die durch Geschäftsprozesse abgebildete Arbeitsschritte [ArndtWI].

2.1.4 Ereignisgesteuerte Prozessketten (EPK)

Die überwiegende Modellierungsform von Geschäftsprozessen ist die Ereignisgesteuerte Prozesskette (EPK) [Leimeister (2015)]. Sie ist eine semiformale, graphische Beschreibungssprache.

Die einzelnen Elemente der EPK werden nachfolgend aufgelistet [Rewissen]:

- **Ereignis:** Ein Auslöser für eine Funktion/ ein Status, der erreicht wurde
- **Funktion:** Ablauf/Auftrag der von einer bestimmten Person ausgeführt wird und ein Input braucht. Er führt zu einem bestimmten Zustand
- **Logische Operatoren:** Entscheidungen oder Verbindungen zwischen Funktionen und Ereignissen
- **Organsitationseinheit:** Stellt Rollen oder Personen dar, die für eine Funktion verantwortlich sind
- **Informationsobjekt:** Input oder Output einer Funktion
- **Prozesspfad:** Repräsentiert komplexe Aktivitäten, die vielleicht später mit einem separaten EPK beschrieben werden.

EPK müssen gemäß [Rewissen] gewisse Regeln einhalten:

- **Regel 1:** Muss mit einem Start-Ereignis beginnen
- **Regel 2:** Muss mit einem End-Ereignis enden
- **Regel 3:** Funktionen und Ereignisse sollen in abwechselnder Reihenfolge vorkommen
- **Regel 4:** Verbindungen zwischen Funktionen und Ereignissen dürfen jeweils nur ein Input- und Outputkonnektor besitzen. Diese Regel gilt nicht für die Verbindung zu anderen Elementen, wie logischen Operatoren
- **Regel 5:** Ein Ereignis muss immer passiv sein, es hat keine Entscheidungsgewalt, Entscheidungen werden von Funktionen durchgeführt

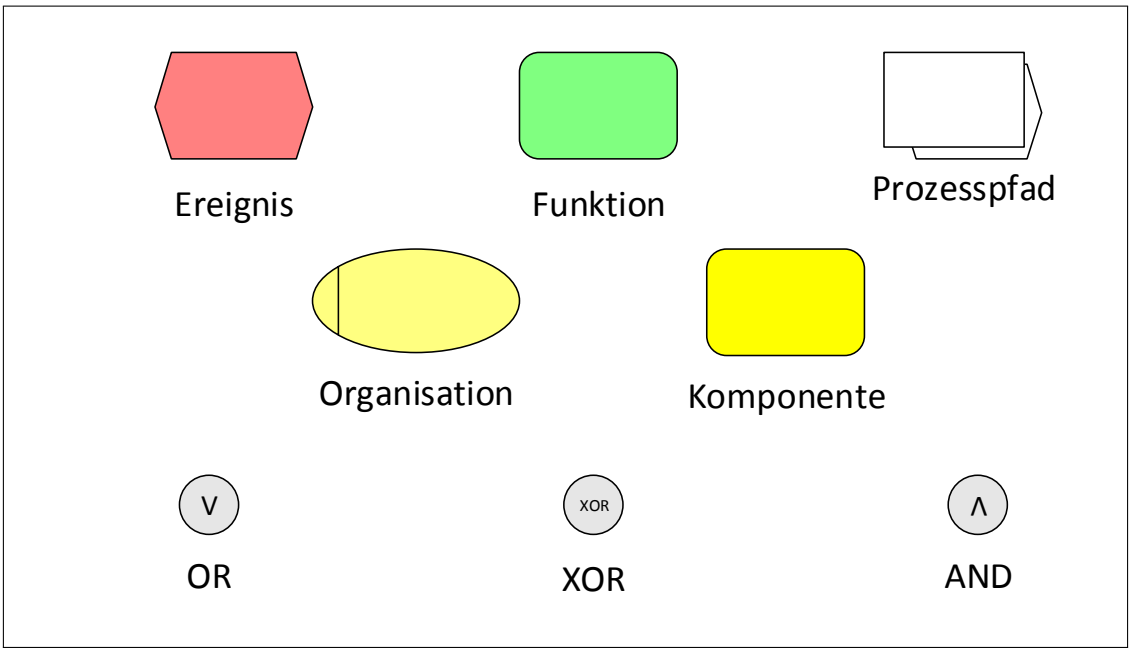

Abbildung 2.1: EPK Elemente in Übersicht

Mit Hilfe der EPK wird der Geschäftsprozess, der als Beispiel für die Abläufe in dieser Arbeit dient, modelliert.

2.2 Informationslandschaften

2.2.1 Integration von Anwendungssystemen

Bei ständig wachsenden Unternehmen, insbesondere bei Großkonzernen, wird die Realisierung von Geschäfts- oder Produktionsprozessen ohne die Unterstützung durch die richtige Software schwer vorstellbar. Auf Kundenbetreuung spezialisierte Anwendungssysteme, wie Customer-Relationship-Management, und die auf die Unterstützung der Wertschöpfungskette spezialisierten Enterprise-Ressource-Planning ergänzen sich durch ihren spezifischen Aufgabenfokus und ihre Funktionalität [Hahn2012]. Wenn im späteren Verlauf der Arbeit also eine Schnittstelle zwischen zwei solchen Systemen entwerfen werden soll, muss man sich zunächst mit der Integration von Informationssystemen auseinandersetzen. Dies wird ausgiebig in den Abschnitten über ERP- und CRM-Systeme durchgeführt.

2.2.2 Definition Informationsintegration

Gemäß [Vajna (2014)] wird durch Anwendungsintegration sichergestellt, dass für jede Aufgabe, die in einem Unternehmen anfällt, das jeweilige IT-System zur Verfügung steht, um es zu unterstützen. [Hahn2012] definiert ein Informationssystem folgendermaßen: „ein Informationssystem dient der rechnergestützten Erfassung, Speicherung, Verarbeitung, Pflege, Analyse, Benutzung, Disposition, Übertragung und Anzeige von Informationen“

2.2.3 Serviceorientierte Architektur (SOA)

Bei der Serviceorientierten Architektur werden Geschäftsprozesse in den Vordergrund gestellt. Diese können durch (Web-)Services dargestellt werden. SOA Services grenzen sich von anderen (Web-)Services durch ihre grobe Granulierung, hohe Wiederverwendbarkeit und einen hohen Datenaustausch ab. Sie sind sehr Modular und als Dienste (Services) gekapselt [Rausch (2004)]. Die Industrie blickt auf SOA mit großem Interesse. So besteht die Hoffnung, das durch SOA Fach- und IT-Bereiche besser miteinander arbeiten können, wodurch die Entwicklung neuer Produkte dieser Unternehmen beschleunigt und gleichzeitig die Kosten von dieser Entwicklung, gesenkt werden sollen [Kohnke u. a. (2008)]. Das Hauptaugenmerk liegt dabei in der Idee, Systeme über Services lose miteinander zu koppeln. Dies entspringt einem der klassischen Probleme mit denen, sich Unternehmen im IT-Bereich konfrontiert sehen. Eines dieser Probleme ist, dass sich ihre IT-Landschaften über die Jahre zu komplexen Gebilden mit hoher Kopplung unter ihren Komponenten entwickelt haben. Daraus folgt, dass die Erweiterung oder Überarbeitung dieser Systeme sehr Zeit- und Kostenaufwändig ist. Auf einem sich mit rasender Geschwindigkeit verändernden Markt, die der Bereich der IT-Dienstleistung umfasst, ist die Fähigkeit sich anzupassen um Konkurrenzfähig zu bleiben unverzichtbar. Diese Notwendigkeit führte zur Entwicklung der SOA, bei der die Komponenten nicht direkt miteinander kommunizieren, sondern über eine Integrierte Plattform Zugriff auf die Funktionen der anderen Komponenten besitzen [Zeppenfeld u. a. (2009)].

2.2.4 Webservices

Auf der Grundlage des Konzeptes der Service-orientierten Architektur sind WebServices bestens dafür geeignet, den Datenaustausch zwischen verschiedenen Anwendungen zu ermöglichen. WebServices können dabei direkt Aufgaben, die man anhand von Geschäftsprozessen abbilden kann, übernehmen. Sie können für Nachrichtenaustausch zwischen mehreren Anwendungen sorgen sowie im Bereich der Integration für die Registrierung und Nachrichtentransformation sorgen. Sie liefern auch die eigentliche Infrastruktur in Form von Basisdiensten, auf der dann

jeweils andere Dienste aufbauen können [Oesterle u. a. (2003)]. WebServices sollen nach ihrer Implementierung für den Endnutzer unsichtbar sein. So muss kein Nutzer den WebService steuern, da die Kommunikation direkt von Maschine zu Maschine läuft und damit verborgen bleibt [Zeppenfeld u. a. (2009)].

Entsprechend [Melzer (2010)] basieren WebServices auf drei Standards:

- **WSDL** - ist eine XML-basierte Beschreibungssprache, die verwendet wird, um eine Schnittstelle der Webservices zu spezifizieren
- **SOAP** - ist ein XML-basierendes Nachrichtenformat, um Prozedurenaufrufe zu übermitteln
- **UDDI** - ist ein zentraler Verzeichnisdienst, bei dem Informationen über WebServices eingezogen werden können, die helfen ein Webservice aufzufinden

2.2.5 Representational State Transfer REST

Um die Kommunikation im Web zu ermöglichen, ist es notwendig, dass gewisse Standards und Protokolle eingehalten werden. Am Anfang nutzte man hauptsächlich RPC, dies wurde aber mit dem Wachstum des Internets immer unpopulärer, da sich Probleme bei der Performance und der Skalierbarkeit herausgestellt haben. Mit der Einführung von REST hat man gewisse Qualitäten geschaffen, die den Zugriff auf Daten erleichterten. Hauptsächlicher Kernpunkt ist die Abstraktion von einer Ressource [Jakl (2008)]. Als zentralen Aspekt von REST stellt [Dazer] die Forderung nach einer gemeinsamen Schnittstelle. Um dies zu erreichen, kann jede Entität als Ressource dargestellt werden. Der Zustand eines RESTful Clients, wird durch das Abfragen, Erstellen, Updaten oder Löschen einer Ressource bestimmt. Ressourcen werden durch eine URI (Unified Ressource Identifier) adressiert. Durch diese URI ist eine Ressource eindeutig Identifizierbar, das bedeutet eine zweite Anfrage mit der selben URI führt zum Aufruf der selben URI, die den gleichen Inhalt repräsentiert. Eine wichtige Eigenschaft von REST ist die Zustandslosigkeit. Durch den Verzicht aus Usersessions ist eine einfache Skalierung durch die Verteilung auf mehrere Server einfach realisierbar.
Auf eine Ressource können eine Hand von verschiedenen Operationen ausgeführt werden. Die geläufigsten unter ihnen sind:

- GET - Anforderung einer Ressource
- POST- Fügt eine neue Ressource an die angegebene an

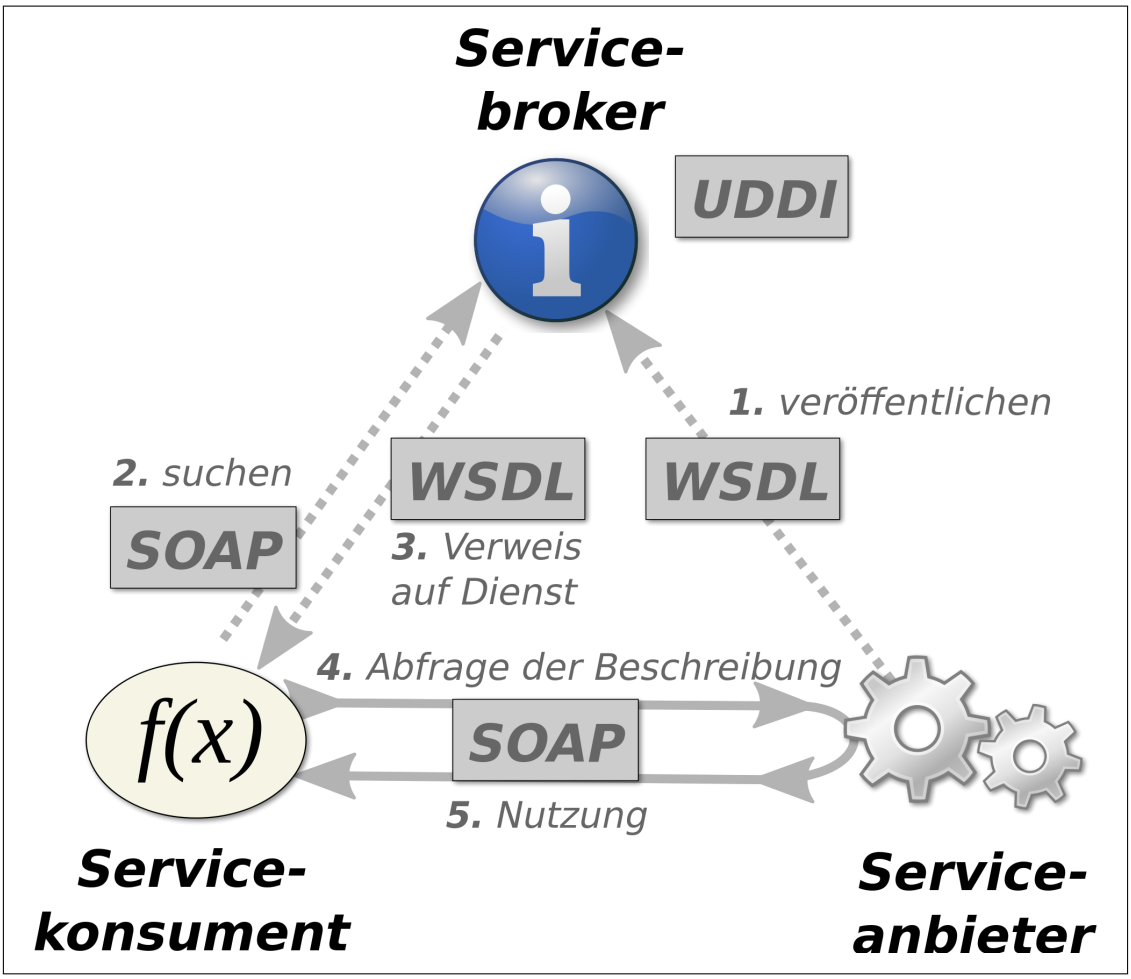

Abbildung 2.2: Webservice Aufbau aus (Wikipedia)

- PUT - Anlegung oder Veränderung einer Ressource
- DELETE - Löscht die angegebene Ressource

Es können noch weitere Operationen nach Bedarf ausgeführt werden [Jakl (2008)].

2.3 CRM Systeme

Customer Relationship Management (CRM) Systeme dienen der Verarbeitung und Pflege von Kundenkontakten innerhalb eines Unternehmens. Durch sie werden die Aktualität und Pflege der Kundendaten gewährleistet, ohne die der Kontakt zu den Kunden erschwert wäre. CRM Systeme werden bei jedem Unternehmen benötigt, das eine größere Anzahl an Kunden

besitzt und bei dem die Kundenbindung ein wichtiger Bestandteil des Geschäftsmodells ist. Ein Kioskbesitzer hat zwar viele Kunden, muss aber nicht die Daten und Vorlieben seiner Kunden verfolgen. Meist reicht bei ihm ein genereller Überblick über seinen eigenen Lagerbestand. Andererseits muss ein Dienstleistungsanbieter im Telekommunikationsbereich einen ständigen Kontakt zu seinem Kunden pflegen, da er sie sonst an die Konkurrenz verlieren könnte. Er muss im Blick behalten, welche Vorlieben und Interessen seine Kunden haben. Auf die Weise kann er seine Produkte sowie Services besser den Bedürfnissen seiner Kunden anpassen, um einen Vorteil im Wettkampf um die Marktanteile zu erlangen.

2.3.1 Definition CRM

Durch CRM Software soll der Kundenbearbeitungsprozess in einem Unternehmen schneller und effizienter durchgeführt werden können. Dabei können Statistiken aus den Informationen des Kunden zusammengeführt und nach Bedarf bereitgestellt werden. Aus den gewonnen Informationen ist eine bessere Planung der Prozesse, die um den Kundenkontakt aufgebaut sind. Durch diese Maßnahmen sollen Kosten eingespart werden [GablerCRM].

2.3.2 IT- Anforderungen an CRM Systeme

Bei der großen Auswahl an potentiellen CRM-System, die auf dem Markt existieren, ist es notwendig, Entscheidung hinsichtlich der Anschaffung der richtigen Software treffen zu können. Obwohl viele CRM-Systeme den durch die Geschäftsprozesse eines Betriebs gestellten Anforderungen zunächst gewachsen erscheinen, gibt es einige Punkte, die oft als Ausschlusskriterium zu sehen sind. Zuallererst beruhen viele Systeme auf einer alten technischen Basis, was spezifisches Know-how erfordert, dass in den Unternehmen oft nicht vorhanden ist. Dies führt zu einem hohen Aufwand für die IT-Abteilungen, die mit der Verwaltung der Software sowie der Konfigurierung und der Fehlerbehandlung beschäftigt sind [Helmke u. a. (2012)].

Eine weitere Anforderung ist die technische Offenheit des Systems. Die früher nicht bekannten Vertriebsstrategien über Soziale Netzwerke gewinnen in den letzten Jahren immer mehr an Bedeutung. Deswegen ist die Anbindung von Erweiterungen an das System sehr wichtig [Hippner u. a. (2011)].

Heutzutage bieten moderne Systeme Schnittstellen an, die ein Austausch von Daten mit einem anderen System vereinfachen. Diese ermöglichen ein einfaches Importieren von bereits bestehenden Kundendatensetzen, was die Initialisierung neuer Systeme vereinfacht. Sie erlau-

ben außerdem die Migration von Daten zu anderen Systemen, wie zum Beispiel ERP-Systemen [Helmke u. a. (2012)].

2.3.3 Spezifische Aufgaben der CRM-Systeme

Der heutige Markt bieten eine große Auswahl möglicher Customer-Relationship-Management-Systeme. Um konkurrenzfähig zu bleiben, erweitern immer mehr Anbieter den Funktionsumfang ihres Produktes. Als Konsequenz davon verschwimmt der Übergang zwischen den klassischen CRM-Systemen und anderen Applikationen, wie den ERP-Systemen einer Anwendungslandschaft. Damit ein Unternehmen die richtige Wahl nicht nur bezüglich eines Systems sondern einer gesamten Anwendungslandschaft und mit ihr eingehenden Schnittstellen zwischen den Einzelsystemen treffen kann, ist es notwendig die Kernfunktionalität der CRM-Systeme abzugrenzen.

In der Abbildung (2.3) werden die einzelnen Komponenten eines CRM-Systems nach [Helmke u. a. (2012)] dargestellt. In den folgenden Abschnitten werden die einzelnen Punkte dieser Konstellation durchgegangen und ihr Aufgabenbereich erläutert.

Lead Management

Ein Lead ist ein potentieller Kunde, der Interesse an einem Produkt oder einem Service zeigt, die das Unternehmen anbietet. Die Zielsetzung von Leadmanagment ist es, ein Lead in einen Kundenkontakt umzuwandeln, indem man einen Verkauf abgeschlossen hat. In diesem Fall werden in einem CRM System die Daten eines Leads in Kundendaten umgewandelt [LeadsVT]. In [Hippner u. a. (2011)] wird Lead-Management folgendermaßen definiert: „Lead-Management beschäftigt sich mit der Erfassung, Qualifizierung, Priorisierung und Weiterleitung von Interessenbekundungen der Kunden, die aus Kampagnen und anderen Unternehmensmaßnahmen entstanden sind“. Da das Ziel eines Mitarbeiters ist, einen Lead in einen Kunden umzuwandeln und die Arbeitszeit der Mitarbeiter ein Kostenfaktor ist, ist es notwendig Kontakte bzw. Leads qualifizieren zu können. So kann das limitierte Zeitkontingent der Mitarbeiter genutzt werden, um sich auf die vielversprechenden Leads konzentrieren zu können [Hippner u. a. (2011)]. Deswegen muss ein CRM-System die Möglichkeit bieten, die Kommunikationshistorie mit dem Kunden abbilden zu können [Helmke u. a. (2012)].

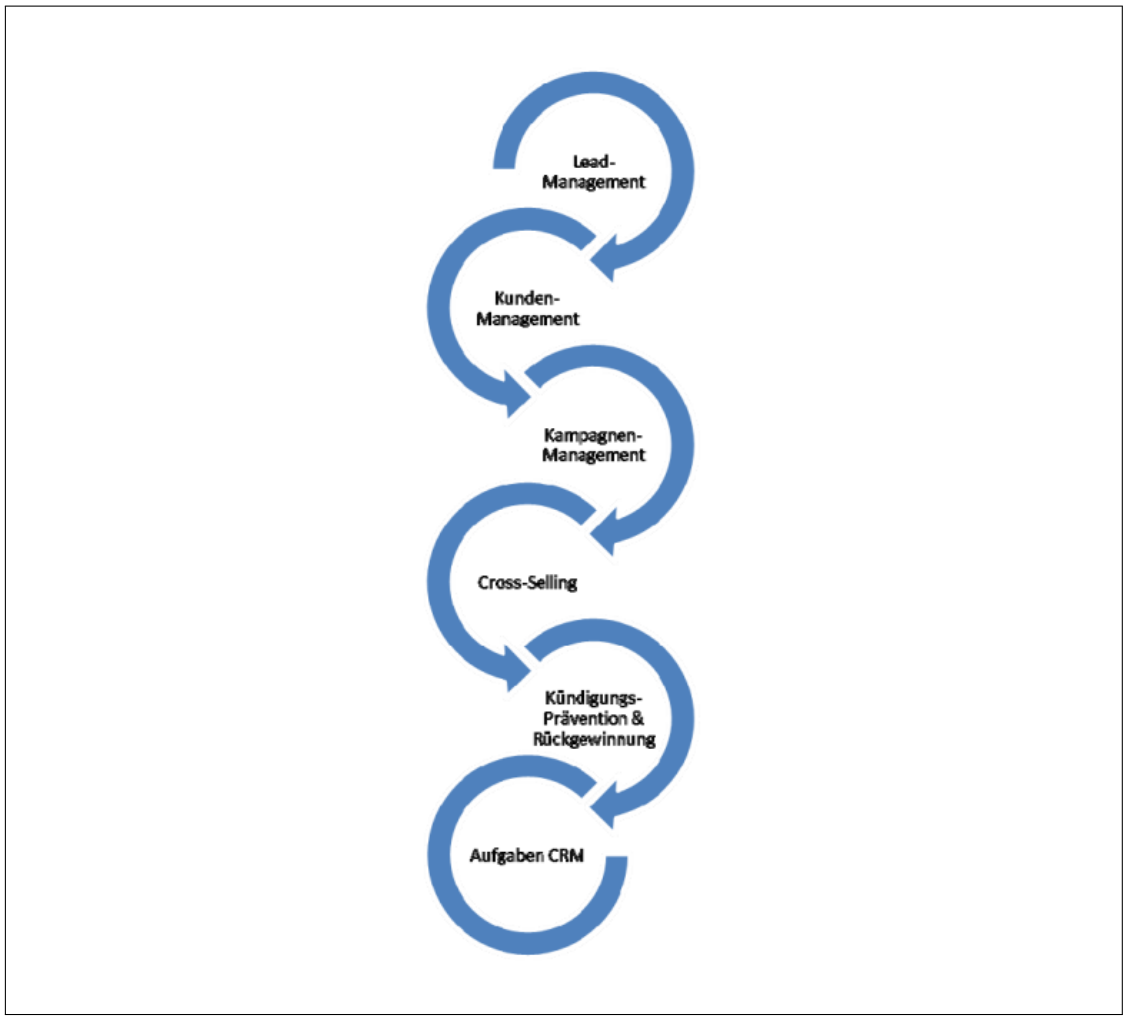

Abbildung 2.3: Aufgaben eines CRM-Systems aus (Helmke u. a., 2012)

Kunden Management

Zum Abschluss des Verkaufs kann ein Lead zu einem Kunden konvertiert werden. Manche CRM-Systeme, wie Vtiger [LeadsVT] bieten die Möglichkeit dies automatisch zu tun. Nachdem ein Lead erfolgreich zu einem Kundenkontakt wurde, fällt er in den Bereich des Kundenmanagements. Hierbei sollte es immer erkennbar sein, welche Produkte oder Services ein Kunde gekauft hat und welche Marketing Strategien erfolgreich angeschlagen haben. Darüber hinaus sollte das CRM-System das Verkaufspersonal durch automatisierte Vorgänge, wie die Erinnerung an den regelmäßigen Kundenkontakt, unterstützen [Hippner u. a. (2011)]. In diesem Bereich sind Funktionen wie Terminplanung und deren Protokollmöglichkeiten sowie die

Angebotserstellung vorgesehen. Hierbei treten oft Überschneidungen zu den anderen Systemen der Anwendungslandschaft auf. Diese müssen dann in die jeweils anderen Systeme übertragen werden.

Kampagnenmanagement

Gemäß [Hippner u. a. (2011)] ist die Aufgabe des Kampagnenmanagements „...die Planung, Durchführung und Analyse aller Kampagnen mit dem Ziel, Interressensbekundungen (Leads) von Bestandskunden und Neukunden zu generieren...“. Darüber hinaus wird die Kundenansprache im Rahmen der strategischen Zielsetzung eines Unternehmens hervorgehoben. Die Mitarbeiter sollen die Kunden mit Informationen und Werbung auf den neuesten Stand der Angebote halten. Das CRM-System muss deshalb Möglichkeiten zur Zielgruppenanalyse bereitstellen. Durch dies wird es ermöglicht, einzelne Kundeninformationen zu verwerten und aus diesen Information, Anstrengungen um ein Einzelkunden zu Kampagnen des Marketings zu verbinden. Die Anwendungen brauchen die Option, Kunden nach speziellen Kriterien auswählen zu können, um sie zu einer Zielgruppe zu verbinden. CRM-Systeme müssen hierbei auch den Ausschluss von gesonderten Kunden, die keine Werbung erhalten möchten, da dies in Zeiten des heutigen allgegenwärtigen Marketings eine Abneigung bei Kunden hervorruft [Helmke u. a. (2012)].

Cross Selling

Beim Cross Selling wird der bestehende Kundenkontakt genutzt, um dem Kunden weitere Produkte oder Services anzubieten, die mit der von ihm bereits erworbenen Ware in Verbindung stehen und vom Unternehmen zusätzlich angeboten werden. Hierbei können zum Beispiel zusätzlich zum Verkauf eines Kraftfahrzeugs weitere Serviceleistungen angeboten werden. Hierbei werden gleich mehrere positive Aspekte deutlich. Zunächst ist die direkte Steigerung des Kundenwertes durch den Verkauf multipler Produkte an den Einzelkunden und die daraus resultierende Gewinnsteigerung hervorgehoben. Darüber hinaus wird der Kunde durch den Mehrfachkauf stärker an das Unternehmen gebunden, was zu einem erhöhten Kaufpotential in der Zukunft führt. Schließlich besteht die Gelegenheit durch die gleichen Vertriebswege Kosten zu sparen [Schawel und Billing (2014)]. Der Kunde erhält auf diese Weise ebenfalls Vorteile, da er weitere Angebote von einem Unternehmen erhält, dem er bereits vertraut und dessen Produkte er schätzt. Deshalb fällt bei ihm die Suche nach weiteren Anbietern womöglich weg. Bei Unternehmen mit direktem Kundenkontakt, wie zum Beispiel Callcentern, ist Cross-Selling besonders wirksam. Dabei bietet das CRM-System die Möglichkeit nach einem Verkäufer-Kunden-Gespräch die Eindrücke des Mitarbeiters über den Kunden zu hinterlegen, aus denen

später Schlüsse für die weitere Angebotsunterbreitung gezogen werden können [Helmke u. a. (2012)].

Kündigungsprevention

Durch das Sammeln von Daten in CRM-Systemen soll es möglich sein frühzeitig zu erkennen, ob ein Kunde unzufrieden mit der Leistung des Unternehmens ist und abspringen will. Dies ermöglicht unter anderem durch das Sammeln von Beschwerden und das Auswerten dieser Informationen ein Profil eines Kunden zu erstellen, den die Firma Gefahr läuft zu verlieren. Ein solches Profil kann genutzt werden, um gezielt auf Kunden einzugehen und ihr Bild vom Unternehmen zu verbessern. Darüber hinaus kann eine qualifizierte Entscheidung getroffen werden, ob der Kunde die in ihn gesteckten Ressourcen durch spätere Käufe wett macht oder es aus unternehmensstrategischen Gründen besser ist, den Absprung des Kunden zu riskieren [Helmke u. a. (2012)].

2.4 ERP Systeme

Hauptsächlich lassen sich ERP-Systeme als Anwendungssysteme, die die Integration von Funktion, Daten und aufgaben in eine Informationssystem übernehmen. Ihr größter Vorteil ist die Automatisierung von Abläufen und die Standarisierung von Prozessen [Gronau (2004)]. Ein ERP-System sollte hierzu die für die Durchführung der Geschäftsprozesse notwendigen Informationen über Ressourcen wie Material, Personal und Kapazitäten verwalten [Gronau (2012)]. Die Unterstützung wesentlicher operativer Funktionen und Führungsfunktionen in einem integrierten Gesamtsystem, wird von [Leimeister (2015)] als ERP-System definiert. Er gliedert dies in ein Basissystem, dass durch Module wie Rechnungswesen, Controlling, Beschaffung, Produktplanung und Vertrieb erweitert wird.

Diese Systeme ermöglichen unter anderen die Planung von Fabrikstandorten für effektivere Zulieferungsstrategien und den Transport der Waren an dem Kunden, wodurch eine Reduzierung der Kosten ermöglicht wird.

Früher standen ERP Systeme ausschließlich Großunternehmen zur Verfügung. Mit den Zuwachs an Kostengünstigen ERP Lösungen und mit dem mit ihnen verbundenen Wettbewerbsvorteil gibt es kaum noch Mittelständische- und Kleinunternehmen, die nicht auf ein ERP System zugreifen [Gronau (2012)].

Sobald ein Unternehmen sich für eine ERP-Software entscheidet, entstehen für es laut [Gronau (2012)] starke Vorteile. Diese Vorteile umfassen die schnellere Auftragsbearbeitung in einem Unternehmen, da die Daten nun zentral gelagert werden. Das Supply Chain Managment wird durch eine Verbindung zu Lieferanten vereinfacht. Außerdem erleichtern solche Gesamtsysteme die Schnittstellen zu anderen Anwendungen wie CRM-Systemen, da sie die Funktionalität von mehren Systemen zusammenfassen und dadurch nur eine Schnittstelle zu einem anderen System notwendig ist.

Als Nachteil von Gesamtlösungen, laut [Gronau (2012)], kann eine zu umfangreiche Funktionalität der Software gesehen werden, die vom Unternehmen nicht benötigt wird. Diese Überforderung durch Funktionalität erschwert die Anpassung an die spezielle Geschäftssituation des Unternehmens. Ein Unternehmen macht sich darüber hinaus, über einen gewissen Zeitraum von einem einzelnen Hersteller abhängig. Gewisse Branchen sind auf Speziallösungen angewiesen, um konkurrenzfähig zu bleiben [Bunjes u. a. (2002)].

2.4.1 BWL Definition

Die Definition von ERP-Systemen nach [GablerWL] ist „Ein Enterprise-Resource-Planning-System oder kurz ERP-System dient der funktionsbereichsübergreifenden Unterstützung sämtlicher in einem Unternehmen ablaufenden Geschäftsprozesse. Entsprechend enthält es Module für die Bereiche Beschaffung/Materialwirtschaft, Produktion, Vertrieb, Forschung und Entwicklung, Anlagenwirtschaft, Personalwesen, Finanz- und Rechnungswesen, Controlling usw., die über eine (in Form einer relationalen Datenbank realisierte) gemeinsame Datenbasis miteinander verbunden sind. Durch die unternehmensweite Konsolidierung der Daten ist eine Unterstützung der Planung über sämtliche Unternehmensebenen hinweg (von der Konzernebene über verschiedene Werke, Sparten und Abteilungen bis hin zu einzelnen Lagerorten) möglich.“

2.4.2 Aspekte der Integration

Bevor wir zu einer Integration und Betrachtung von einer Schnittstelle zwischen einem ERP- und einem CRM-System kommen, müssen wir uns des Aufbaus eines ERP-Systems klar werden. Der Übergang zwischen ERP- und CRM-Systemen ist heutzutage fliesend, wodurch eine Abgrenzung des Aufgabenbereiches an Bedeutung gewinnt. Um den Funktionsumfang einer ERP-Systems zu verstehen, muss man sich zunächst mit dem Begriff der Integration auseinandersetzen. Dieser ist aufbauend auf den Geschäftsprozessen eines Unternehmens und führte

geschichtlich zur Entstehung von ERP-Systemen. Durch Vereinigung bestehender Informationssysteme wurden die einzelnen Komponenten mit gleicher oder sich überschneidender Funktionalität zu Gesamtsystemen zusammengefasst. Dies geschah aus Gründen der Reduzierung der Anzahl der Komponenten als auch sowie der zwischen den Einzelkomponenten bestehenden Schnittstellen [Fischer und Stelzer (2008)].

Zur Charakterisierung und Eingrenzung der Integration schlägt [Fischer und Stelzer (2008)] den Begriff „Dimension“ vor. Hingegen kann die Integration gemäß [Jacob u. a. (2013)] in vier Aspekte eingeteilt werden. Diese vier Aspekte sind:

- Gegenstand
- Vereinigung/Ausrichtung
- Nutzungsbezug/Reichweite
- Bereich

Gegenstand der Integration

Der Gegenstand der Integration unterteilt sich nach [Jacob u. a. (2013)] weiter in vier unterschiedliche Formen von Integration. Die erste Form ist die Datenintegration. Sie legt fest, welche der Daten von mehreren Abteilungen oder Systemen genutzt werden. Ihr Ziel ist eine redundanzarme und zugrifffreundliche Strukturierung der gemeinsam genutzten Daten. Durch Datenintegration wird eine verbesserte Integrität der Daten, eine redundanzarme Datenspeicherung, keine Mehrfacherfassung bei Bewegungs- und Stammdaten, eine kürzere Durchlaufzeit eines Vorgangs und gleiche Aktualität der Daten erreicht [Jacob u. a. (2013)].

Die zweite Form - die Funktionsintegration - teilt sich in zwei Stufen auf. Die Sachbearbeiterdateneingabe fasst die Funktionen Datenerfassung, Sachbearbeitung sowie die Steuerung der Verarbeitung zusammen. Bei der zweiten Stufe werden einzelne Funktionen von einem Arbeitsplatz zusammengefasst. Daraus ergeben sich Vorteile wie ein niedriger Koordinationsaufwand, sowie ein erhöhter Zugang zu Informationen. An dieser Stelle fallen auch Arbeitserleichterungen in den Geschäftsprozessen an die zur Strukturierung der Sachbearbeitung führen [Jacob u. a. (2013)].

Integration der Benutzeroberfläche beinhaltet eine einheitliche Schnittstellengestaltung für den Anwender jener Informationssysteme. Um möglichst kurze Einarbeitungszeiten zu

ermöglichen und eine effektive und unkomplizierte Funktion des Systems zu gewährleisten, ist die Einheitlichkeit in Bereich der Kommandos, Funktionstastenbelegung, Fehler- sowie Rückmeldungen und einheitlichen Dialogformen eine wichtige Form der Gestaltung [Jacob u. a. (2013)]. Dies ist eine wichtige Eigenschaft, die bei Gestaltung der in dieser Arbeite zu entwickelnden Schnittstellensoftware zu berücksichtigen ist.

Die Prozessintegration unterscheidet sich zu Funktionsintegration, indem es die Funktionen zu Prozessen zusammenfasst. Der Unterschied liegt darin, dass bei der Prozessintegration die Schnittstellen zwischen den Prozessen betrachtet werden [Jacob u. a. (2013)].

Integrationsausrichtung

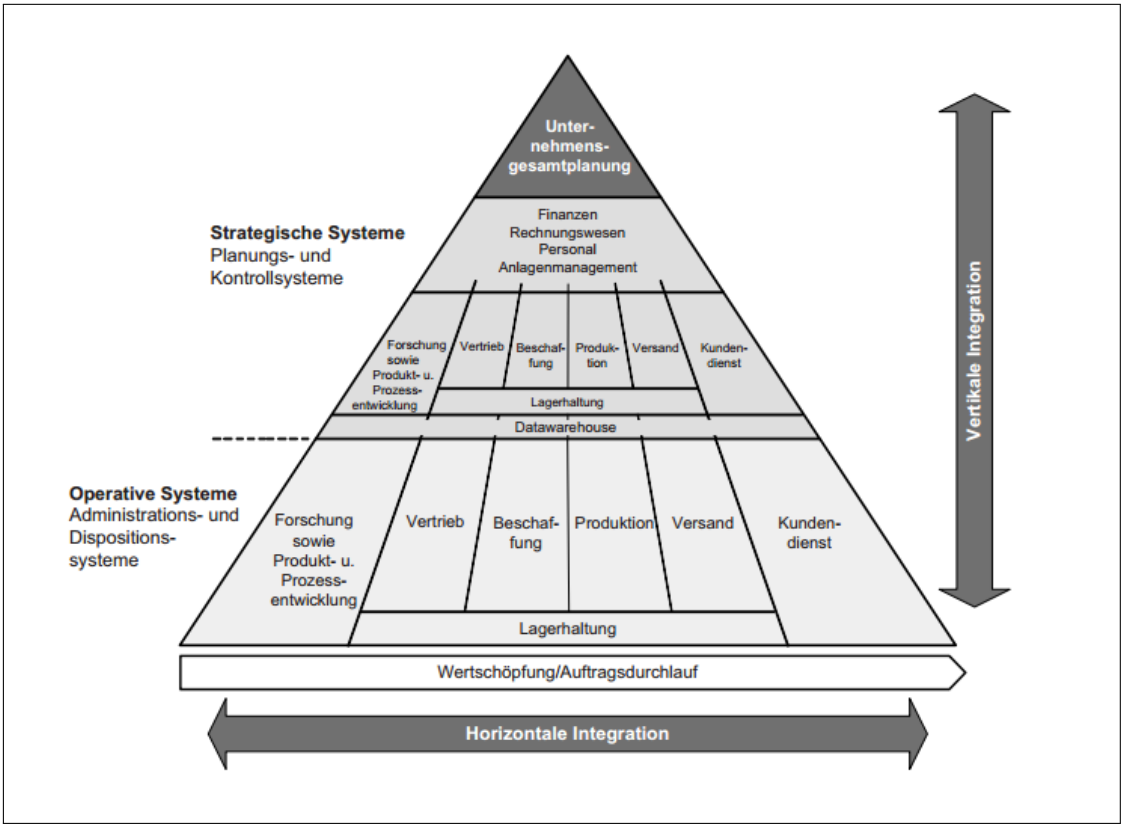

Abbildung 2.4: Vertikale und horizontale Integration innerhalb der Ebenen eines Unternehmens aus (Fischer und Stelzer, 2008)

Die Integrationsausrichtung wird von [Fischer und Stelzer (2008)] durch eine Positionierung zwischen den Ebenen eines Unternehmens dargestellt. Er unterscheidet zwei Arten der Rich-

tung, die horizontale und vertikale. Die horizontale Integration bezieht sich auf eine Ebene des Unternehmens und orientiert sich an der Wertschöpfungskette. Vertikale Integration hingegen bezieht sich auf die Integration zwischen den Ebenen eines Unternehmens. Sie umfasst die strategischen Systeme wie Planung und Controlling [Jacob u. a. (2013)]. Sie bildet daher die Geschäftsprozesse der Management Kategorie ab vgl. (2.1.2).

Reichweite der Integration

Laut [Fischer2007] lässt sich die Reichweite der Integration in vier Stufen aufteilen. Die Bereichsintegration, die bereichsübergreifende und die innerbetriebliche Integration umfassen alle drei die Integration innerhalb eines einzelnen Unternehmens. Dabei steigt die Reichweite von der Bereichsintegration bis hinzu der innerbetrieblichen Integration an. Die unternehmensübergreifende Integration stellt eine nochmalige Erhöhung der Reichweite da. Dies wird nochmal in Abbildung (2.5) verdeutlicht.

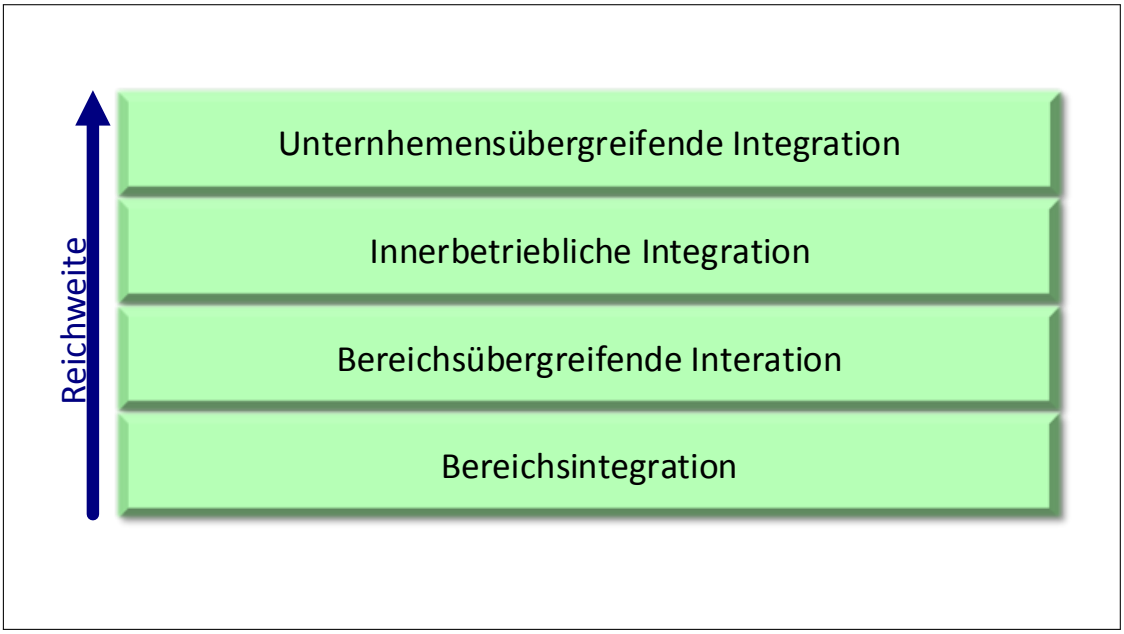

Abbildung 2.5: Reichweite der Integration

Integrationsbereich

Bei dem Integrationsbereich handelt es sich um die Perspektive der Betrachtung der Integration [Fischer und Stelzer (2008)]. Die Bereiche sind in Anlehnung an [Jacob u. a. (2013)] Technik,

Anwendungsnutzung, Entwicklung von Informationssystemen und Informationsmanagement. Bei der Technik wird vor allem die Kopplung von unterschiedlicher Hardware geachtet. Die Anwendungsnutzung hat starken Bezug zu der Datenverarbeitung. Hingegen wird beim Bereich der Entwicklung ein Blick auf die Methoden und Werkzeuge, die während des Betriebs eines System verwendet werden, geworfen. Schließlich wird beim Informationsmanagement eine Abbildung der Informationen durch einen konkreten Zweck verdeutlicht.

2.4.3 Struktur ERP-Systeme

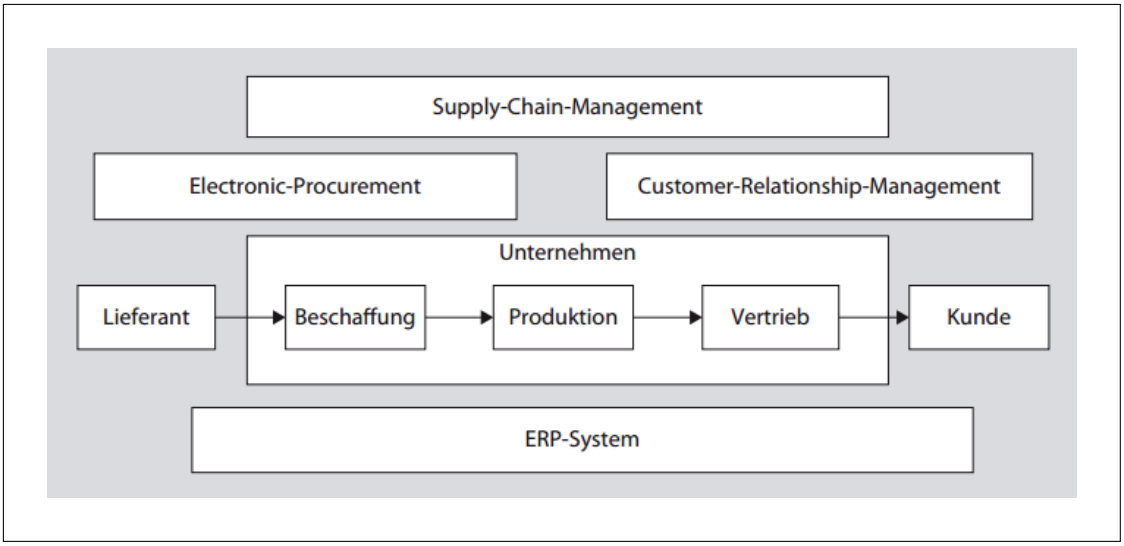

Abbildung 2.6: Struktur von ERP-Systmen aus (Leimeister, 2015)

In der Abbildung (2.6) sieht man den typischen Aufbau eines ERP-Systems. Der harte Kern eines ERP-Systems ist dabei die Produktionsplanung und die dazugehörigen Produktionssteuerungssysteme, die die Arbeitsgebiete Lagerung, Beschaffung also der Einkauf, Materialbedarfsplanung, Fertigungsplanung und Fertigungssteuerung umfassen [Leimeister (2015)]. Anhand der Abbildung sieht man, dass sich der Aufgabenbereich eines ERP-Systems mit den Aufgabenbereichen anderer Systeme überschneidet. Dies resultiert aus dem Bestreben zur Vereinigung (siehe 2.2.1) von bestehenden Modulen in einem Unternehmen, dass zu einem möglichst integrierten gesamt ERP-System strebt. Allerdings setzen sich auch in Unternehmen Lösungen durch, bei denen ein kompakter Teil der Funktionalität in einem ERP-System gebündelt wird und separate Systeme Aufgaben wie des CRM und anderer Bereiche übernehmen.

2.4.4 Die Rolle der ERP-Systeme

Gemäß [Leimeister (2015)] geht die Tendenz dahin, dass ERP-Systeme um immer weitere Module erweitert werden. Diese gehen über den klassischen Bereich des ERP-Umfangs immer weiter hinaus. Dies ist durch den Drang zur Integration sämtlicher Geschäftsprozesse durch die Integration laut [Gronau (2004)] immer stärker ausgeprägt. Aufgrund dieser Entwicklung werden viele Systeme wie das Customer-Relationship-Management mit ihrer Funktionalität als Bestandteil des ERP-Systems integriert. Als Folge dessen sind CRM-Systeme für Großunternehmen immer unattraktiver, da die branchenspezifischen Gesamtlösungen der Anbieter die volle Funktionalität anbieten und dabei die Kosten und Probleme, die bei Anwendungsintegration und der Anpassung an die Bedürfnisse der Unternehmen in beiden Systeme anfallen würden, wegfallen.

Dennoch gibt es Ausnahmen. Durch das Verlangen nach erreichen von Wettbewerbsvorteilen setzen Unternehmen oft auf selbst entwickelte oder eingekaufte Speziallösungen von Systemen. Darüber hinaus zeigen einige etablierte ERP-Lösungen eine externe Finanzbuchhaltung und ein separates Personalverwaltungssystem, da sich die Spezialsysteme in diesem Bereich als sehr Vorteilhaft erwiesen haben. Diese Vorteile überwiegen die Nachteile die die geringere Integration mit sich bringt [Gronau (2012)]. Außerdem werden durch Fusionierung und Übernahmen von Konzernen, wegen des Datenerhalts und des Risikos bei der Migration der Daten auf ein einzelnes System, oft mehrere ältere Systeme miteinander gekoppelt. Hierdurch wird die Notwendigkeit von Enterprise-Application-Integration deutlich [ArndtWI].

2.5 IT Schnittstellen - Enterprise Application Integration

Zu Beginn dieses Abschnittes wird auf das Problem der Definition von EAI eingegangen. Nach [Strüver (2006)] hat sich keine eindeutige Definition etablieren können. Während der Bearbeitung dieser Arbeit sind die Gründe, für diesen Definitionsmangel klar geworden. Bei der Integration laut [Jacob u. a. (2013)] wird von Vereinigung von einzelnen Modulen, deren Funktionalitäten sich überschneiden, zu einem gemeinsamen ERP-System gesprochen. Nach [Mantel und Schissler (2002)] geht die Entwicklung soweit, dass man den Lieferanten einen Zugriff auf sein eigenes Integriertes System bietet. Laut [Gronau (2004)] ist das Ziel der Integration von Anwendungen in einem Unternehmen (EAI) ein möglich integriertes System zu schaffen. Was ebenfalls als EAI bezeichnet wird, ist die Kopplung, also die Anbindung unterschiedlicher Systeme wie CRM und Supply-Chain-Management unter Verwendung von IT-Schnittstelle [Jacob u. a. (2013)]. In diesem Abschnitt wird demzufolge auf die Anwendungsintegration im

Sinne der Kopplung durch Schnittstellen eingegangen. Es werden die Gründe für die Kopplung diskutiert und die Probleme, die dabei auftreten können, analysiert. Es wird immer wieder auf den Begriff der Integration im Sinne der Vereinigung aus dem Abschnitt (2.2.1) Bezug genommen, da die Dimensionen der Integration zwischen Kopplung und Vereinigung Parallelen aufzeigen. Darüber hinaus werden die verschiedenen konzeptionellen Lösungen der Anwendungsintegration untersucht.

2.5.1 Anordnungsarten der Kopplung von Anwendugssystemen

Eine Kopplung zweier Anwendungen kann in Anlehnung an [ArndtWI] auf Grundsätzlich drei verschiedene Arten durchgeführt werden.

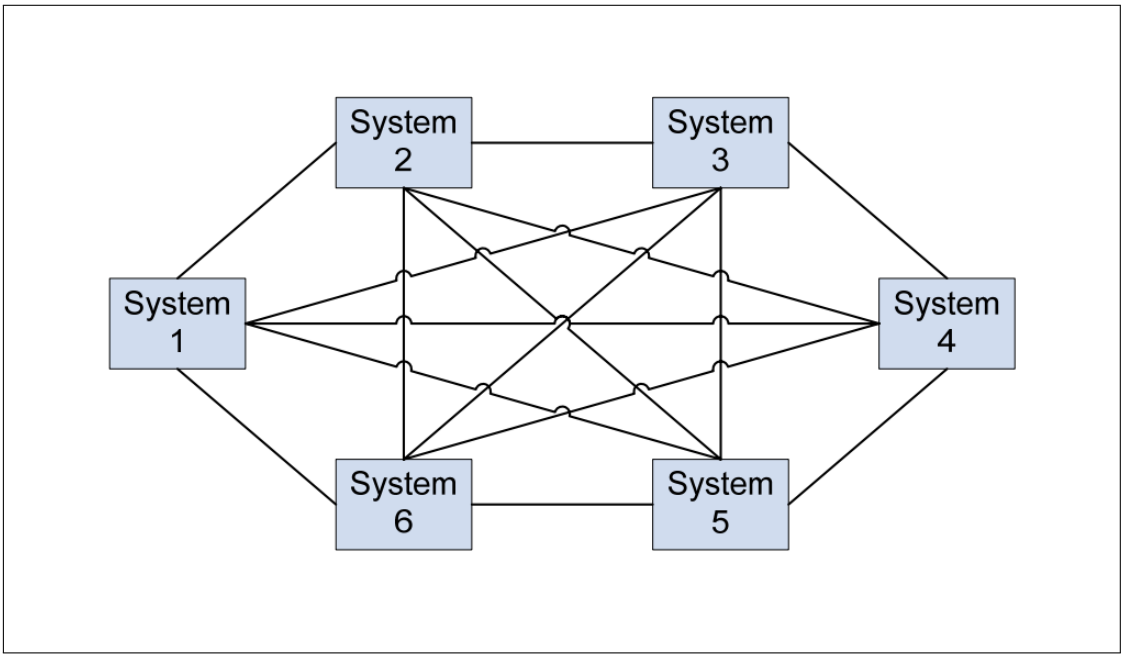

Abbildung 2.7: Punkt zu Punkt Integration aus (ArndtWI)

Die erste Art ist die Punkt-zu-Punkt Integration, hier in Abbildung (2.7) zu sehen. Geschichtlich gesehen ist dies die erste Art der Integration zwischen Anwendungen gewesen [Itransparent]. Das wesentliche Merkmal der Punkt-zu-Punkt Integration eine direkte Kopplung zweier Systeme. Die Schnittstellensoftware kümmert sich dabei ausschließlich um den Austausch und die Transformation der Daten zwischen Quell- und Zielsystem. Eine solche

Schnittstelle ist die am einfachsten und vor allem am schnellsten zu realisierende Art einer Schnittstelle. Sie bringt jedoch wesentliche Nachteile mit sich. Während die Anzahl der Schnittstellen bei zwei integrierten Systemen nur Eins beträgt, steigt die Anzahl auf n*(n-1)/2 Schnittstellen bei n Systemen. Hierdurch steigen die Wartungskosten und der Anpassungsaufwand mit jedem weiteren integrierten System rasant an [ArndtWI]. Ein Unternehmen kann sich in seinen Anfangsjahren also dafür entscheiden, zwei Anwendungssysteme auf diese Weise zu koppeln, um kurzfristig Kosten zu sparen. Es würde die Entscheidung aber wahrscheinlich in den kommenden Jahren bereuen, weil es sich im stetigen Wandel und Anpassungsdruck des Marktes für zusätzliche spezialisierte Lösungen seiner Systeme entscheiden müsste [vgl. Gronau (2012)]. Demnach stellt sich die Frage, ob es sinnvoll ist, diese Art der Integration zu verwenden. Trotz der Nachteile offenbaren sich gewisse Vorteile solcher Schnittstellen. Zunächst sind diese am schnellsten implementiert, was niedrigere Kosten für das Unternehmen bedeutet, welches ein nicht zu unterschätzender Vorteil in den Aufbaujahren einer Firma mit begrenzten Budget ist. Unternehmen müssen sich ein Kosten/Nutzen Faktor vor die Augen halten, welcher ihnen nicht erlaubt, Geld in eine zukunftssichere Lösung zu investieren. Darüber hinaus ist eine Punkt-zu-Punkt Integration eine valide Lösung, falls man eine Architektur anstrebt, bei der man mehrere Systeme an ein einzelnes ERP-System anbindet, ohne direkte Verbindungen zwischen den anderen Systemen herzustellen. Als Beispiel könnte ein CRM-System ausschließlich mit dem ERP-System kommunizieren, während auf der anderen Seite das ERP-System mit einem Supply-Chain-Management-System in Verbindung steht. Dies zeigt das diese Art der Integration nur in einem begrenzen Ausmaß eingesetzt werden kann.

Der Integration um ein zentrales ERP-System ähnelnd ist die Broker- bzw. Hubarchitektur. Bei dieser Art sind die Systeme um ein zentralen Broker angesiedelt, der die Kommunikation zwischen den Einzel-Systemen regelt (siehe Abbildung 2.8). Hierbei wird eine standardisierte Schnittstelle angeboten, an die die Systeme angebunden werden [ArndtWI]. Bei dieser Lösung sieht man, dass die Anzahl der anzubindenden Schnittstellen immer überschaubar bleibt, da für die zukünftig hinzukommenden Anwendungen maximal eine einzige Schnittstelle implementiert werden muss. Damit bleibt der Aufwand, den man für die Integration eines Systems betreiben muss, gering. Allerdings ist der ursprüngliche Aufwand größer, da selbst wenn man zu Beginn nur zwei Systeme koppeln möchte, trotzdem der Broker als System implementiert werden muss. Dies führt also zu höheren Initialkosten. Hinzukommt, dass es bei einer hohen Anzahl an Anfragen zum Datenstau kommen kann. Deswegen gibt es bei dieser Art der Integration ein „Single-Point-of-Failure“[Itransparent].

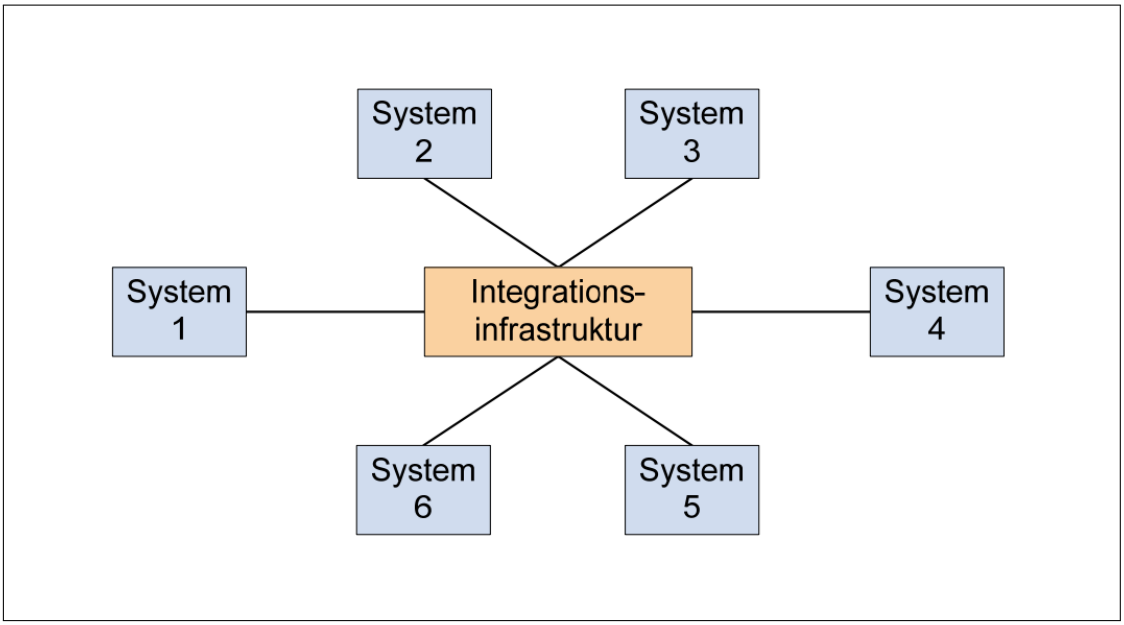

Abbildung 2.8: Integration mit einem zentralen Broker aus (ArndtWI)

Im letzten Punkt wird die Bustopologie betrachtet. Hier unter Abbildung (2.9). Ein Bussystem unterscheidet sich von einem Broker darin, dass die einzelnen Anwendungen nicht über ein einzelnes System kommunizieren zu dem sie Schnittstellen anbinden, sondern jedes einzelne System eine eigene angebundene Komponente hat, die die Kommunikation zu den anderen Systemen reguliert [ArndtWI]. Diese Komponenten sind auf den Kommunikationsbedarf der einzelnen Systeme konfigurierbar, wodurch „Bottleneck-Probleme“ reduziert werden können. Sie sind wie der Broker mit höheren Anschaffungskosten verbunden und erfordern zusätzlich einen höheren Aufwand was die Konfiguration der einzelnen Komponenten betrifft. Sie sind aber flexibler als der Broker und sparen Kosten durch ihre Zuverlässigkeit.

2.5.2 Integrationsebenen der Anwendungsintegration

Bei der Anordnungsart wurde die Topologie der Schnittstellen in einem Anwendungssystem und ihre Folgen analysiert. Von dieser „Oben-Sicht“ geht man nun auf die möglichen Integrationsebenen in Betrachtung einer einzigen Schnittstelle, also wie man eine konkrete Schnittstelle umsetzen kann. Nach [ArndtWI] unterscheidet man zwischen drei Integrationsebenen, der Datenebene, der Funktionsebene und der Benutzerschnittstellenebene.

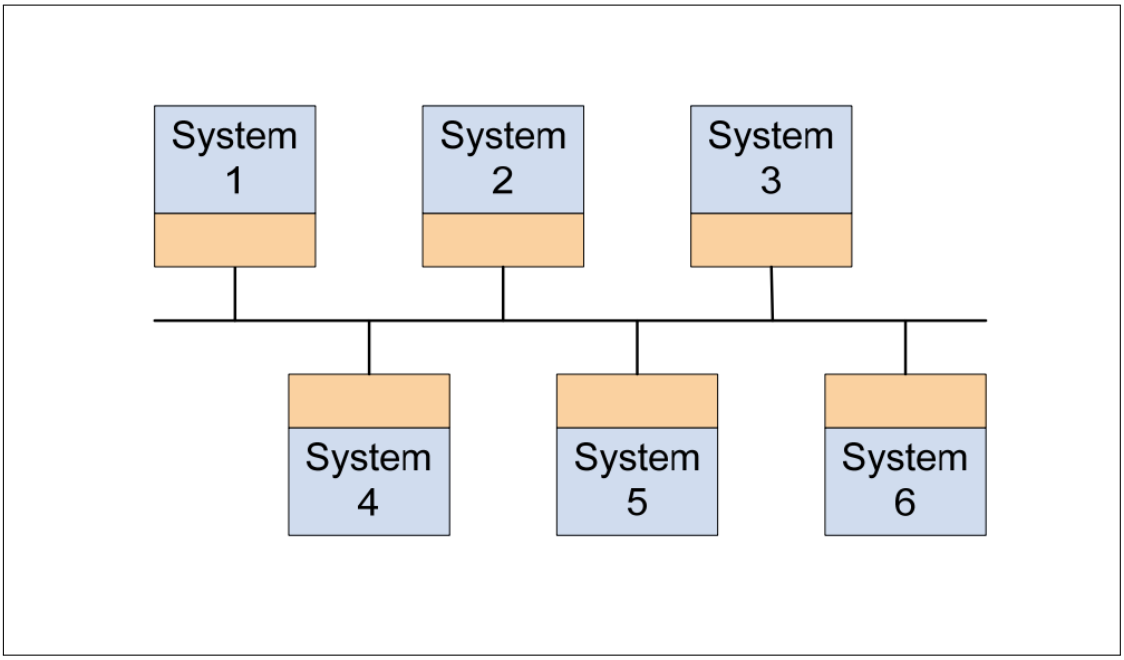

Abbildung 2.9: Integration einer Bustopologie aus (ArndtWI)

Die erste Ebene stellt den Zugriff über die Persistenzebene den Datalayer dar. Diese Form der Integration ist empfehlenswert, falls ein System keine API zur Extrahierung der Daten zur Verfügung stellt und der Quellcode des Systems nicht vorhanden ist [ArndtWI]. Dabei reicht bereits ein Zugriff auf die Datenbank. Allerdings muss bei dieser Integration darauf geachtet werden, dass die Datenbestände entweder das gleiche Schema aufweisen oder dass die Daten nach der Extraktion aus dem Quellsystem in das Schema des Zielsystems überführt werden (vgl. Abschnitt 2.5.4). Es ergeben sich außerdem viele Nachteile. So umgeht der Entwickler die Anwendungslogik eines Systems [nach ArndtWI] also den Businesslayer. Ein Zugriff auf die Daten ist nur sicher, wenn diese ausgelesen werden. Beim Schreiben auf dem Persistenzlayer kann es zu unvorhergesehenen Konsequenzen auf der Logikebene einer Applikation kommen. Darüber hinaus führen Änderungen an den Datenstrukturen einer Anwendung zu einer Verkettung von Änderungen von allen angebundenen Systemen, was diese Form der Integration wenig flexibel macht [ArndtWI].

Die zweite Ebene ist gemäß [ArndtWI] die flexibelste Art der Integration. Unter diesem Begriff sammeln sich sowohl die Integration durch Anpassung des Quellcodes und Zugriffe über Techniken wie Java RMI als auch die Benutzung von den Anwendungen selbst angebotenen Schnittstellen. Es kann dabei auf interne Datenformat und Integritätsprüfungen der

Anwendungen zugegriffen werden, wodurch Probleme der Datenkonsistenz vermieden werden können [vgl. ArndtWI]. Allerdings erfordert eine Integration auf dieser Ebene eine angebotene Schnittstelle oder das Vorhandensein des Quellcodes der Anwendung, wodurch die Möglichkeit der Realisierung nicht immer gegeben ist.

Als absolute Notlösung erwähnt [ArndtWI] noch die Integration auf Präsentationsebene. Auf dieser Ebene werden die grafischen Darstellungen der Anwendungen ausgewertet. Dies kommt nur bei sehr alten Systemen unter Mangel anderer Möglichkeiten und bei HTML-Seiten zum Einsatz [ArndtWI]. Diese Lösung ist nicht empfehlenswert, sei aber an dieser Stelle vollständigkeitshalber erwähnt.

2.5.3 EAI - Integrationserver

Auf dem Markt existieren bereits Lösungen, die einem erlauben Software zu integrieren. Anwendungen, die dafür entwickelt wurden, nur einen Teil eines Geschäftsprozesses abzubilden, sollen mit ihnen so Integriert werden, als ob diese von Anfang an dafür entworfen wurden miteinander zu arbeiten [Keller (2002)]. Dies sind die sogenannten EAI-Integrationserver.

Als ein Beispiel soll hier „BizTalk“ von Microsoft aufgeführt werden. Sein Kern besteht aus dem Server-Modul. Dieses besteht aus zwei Elementen. Das erste ist die Nachrichtenübermittlungskomponente. Diese ermöglicht durch den Einsatz von Adaptern eine Kommunikation zwischen verschiedenen Anwendungen. Dabei werden unterschiedliche Protokolle und Datenformate unterstützt. Das andere Element ist eine grafische Oberfläche zur Gestaltung von Prozessen, die die Logik des Geschäftsprozesses steuert. Unterstützt wird das Hauptmodul durch weitere Module. Eine „Business Rules Engine“, die Regelsätze auswertet, ein Modul mit dem die Entwickler die Abläufe überwachen können, und ein Modul zur Authentifizierung zwischen Windows und anderen Systemen [Microsoft2013 (2013)]. Entstanden ist BizTalk dabei aus dem Bereich des „B2B“ also dem Business-to-Business. Nach [Keller (2002)] lassen sich die Architekturen, die für B2B als auch für EAI eingesetzt werden, aufeinander übertragen. Unternehmensübergreifende Integration und unternehmensinterne Integration besitzen die Gemeinsamkeit, dass die Einzelsysteme wenig oder nichts von dem Gesamtablauf wissen.

2.5.4 Aufbau einer Schnittstelle

Grundsätzlich ist der Aufbau jeder Schnittstelle gleich. Im ersten Schritt werden die Daten aus dem Quellsystem extrahiert. Anschließend werden die Daten validiert, dass heißt auf ihr

Datenformat überprüft. Meistens werden Eigenschaften wie Korrektheit und Vollständigkeit überprüft. Dies kann durch Zuhilfenahme von Metadaten von der Schnittstellensoftware geschehen oder wird vom Quellsystem selbst erledigt. Im nächsten Schritt werden die Daten transformiert. Bei der Transformation werden die Daten, die vom Zielsystem benötigt werden, ausfindig gemacht und aus dem vorher bestehenden Datensatz extrahiert. Der nächste Schritt besteht aus einer Konvertierung der Daten in ein Format, das vom Zielsystem verarbeitet werden kann. Diese werden wieder validiert, um sicherzustellen, dass das Zielsystem diese versteht. Im letzten Schritt werden die neuen Daten in das Zielsystem importiert [Liebhart].

Aus diesem Ablauf kann man feststellen, in welche einzelnen Komponenten sich der Aufbaue einer Schnittstellensoftware unterteilt (siehe Abbildung 2.10).

- **Extract/Export:** Übernehmen die Datenextraktion aus dem Quellsystem. Sie nutzen gegebenenfalls die Services des Systems wie zum Beispiel eine REST Schnittstelle oder sie greifen auf die Persistenz des Quellsystems zu
- **Validate Source:** Prüfen die Daten anhand von Metadaten oder durch Mechanismen die das Quellsystem selbst bereitstellt durchgeführt
- **Transform:** Ist für die Zusammenstellung von Daten für ein Zielsystem aus den Daten von einem oder mehreren Quellsystemen verantwortlich. Dies kommt typischerweise bei Middleware wie Bussystemen vor, da dort von verschiedenen Systemen die Daten extrahiert und dann auf das System, dass diesen neuen Datensatz benötigt übertragen werden
- **Convert:** Eine sehr schmale Komponente, die sich ausschließlich um die Umwandlung von einem Datenformat in ein anderes kümmert.
- **Validate Target:** Das Gegenstück von „Validate Source“, dass für die Korrektheit der Daten im Zielsystem verantwortlich ist
- **Load/Import:** Ähnlich dem „Extract/Export“ für die Überführung der Daten ins Zielsystem durch unterschiedliche Techniken verantwortlich
- **Transport:** Der Transport umfasst den Ablauf zwischen der Validierung von Quelldaten und der Validierung der Zieldaten. Er kann auf unterschiedliche Weisen gelöst werden.
- **Monitoring:** Zuständig für die Überwachung des Transports aber nicht selbst am Vorgang beteiligt. Diese Komponente kommt nicht in Punkt-zu-Punkt Schnittstellen vor

sondern, beim Einsatz von aufwändigeren Schnittstellen, bei denen mehr Probleme während des Ablaufs auftreten. Sie kommt demnach in Middleware Software wie Brokern oder Bussystemen vor

- **Tracing/Logging:** Wie das Monitoring ist auch das Tracing/Logging nicht direkt am Transport beteiligt, sondern dessen Überwachung dient. Diese Komponente protokolliert den Systemlauf und ermöglicht das Nachvollziehen von möglichen Fehlern oder Störquellen

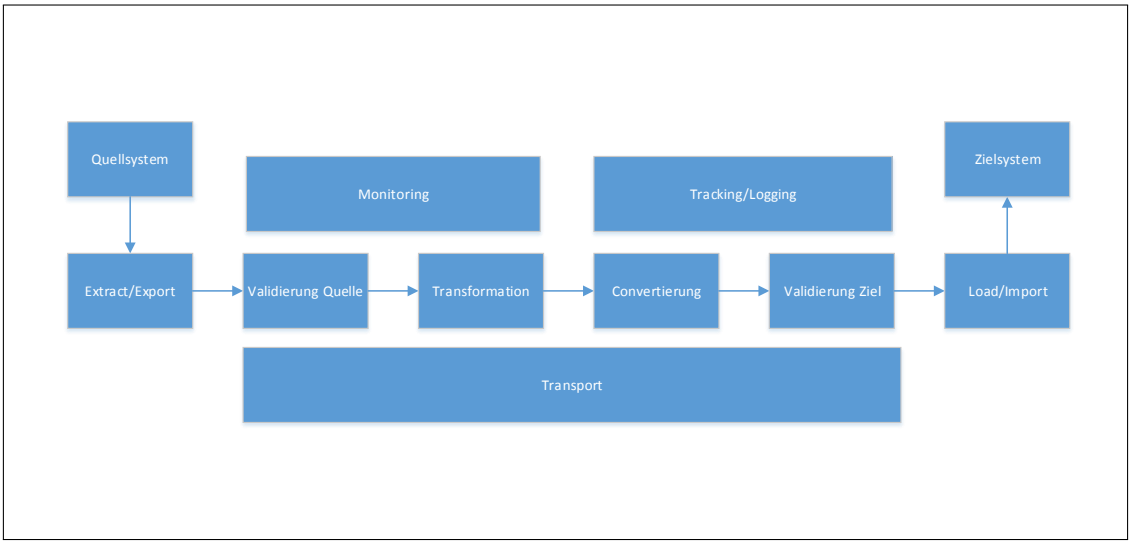

Abbildung 2.10: Komponenten einer Schnittstellensoftware nach (Liebhart)

2.5.5 Schnittstellentypen

Es existieren drei Schnittstellentypen:

- methodenorientierte
- meldungsorientierte
- ressourcenorientierte

Methodenorientierte Schnitstellentypen oder Call Interfaces sind an Funktion und Methodenaufrufen im Call and Return Stil orientiert. Sie eignen sich vor allem für den Austausch einfacher Daten. Meldungsorientierte Schnittstellentypen tauschen strukturierte Daten aus.

Sie werden auch als Document Oriented Interfaces bezeichnet. Ressourcenorientierte Schnittstellentypen sehen Daten als Ressourcen an. Alle dabei nötigen Operationen werden implizit ausgelöst. Ein Beispiel dafür ist REST [Liebhart].

3 Analyse des Geschäftsprozesses

3.1 Ablauf der Analyse

Um eine Schnittstelle ordnungsgemäß modellieren zu können, muss zunächst der ihr zugrunde liegende Geschäftsprozess betrachten werden. Aus diesem können dann die erforderlichen Schlüsse für die Gestaltung der Software gezogen werden. Oft haben die in den Unternehmen dokumentierten Geschäftsprozesse eine geringe Ähnlichkeit mit den tatsächlichen Abläufen im Betrieb. Dies ist auf Mängel der Praxiserfahrung der für den Entwurf der Prozesse verantwortlichen Angestellten zurückzuführen. Ein weiterer Grund ist die Vernachlässigung der Dokumentation besagter Geschäftsprozesse in der firmengeschichtlichen Entwicklung [Schacher und Grässle (2006)].

Aus diesem Grund werden wir ein Geschäftsprozess in möglichst simpler Form nehmen und ihn von Grund auf aufbauen. Wir werden die aus dem Grundlagenkapitel gewonnen Erkenntnisse verwenden, um eine kluge Wahl bezüglich der Schnittstellenart zu treffen. Die Erkenntnisse aus der Abgrenzung des Aufgabenbereiches beider Systeme werden die Anforderungen an die Schnittstelle erleichtern.

Die gesamten Erkenntnisse werden schließlich zusammenfassen, um eine Schnittstelle zu modellieren. Dabei wird die Schnittstelle möglichst kompakt gehalten.

3.2 Gegenüberstellung von Gesamtlösungen und spezialisierten Systemen

Bei der Anschaffung von Anwendungen für Informationslandschaften bietet sich heutzutage eine Fülle von Möglichkeiten unterschiedlichster Lösungen. Diese Auswahl an Software unterscheidet sich in Kategorien Anschaffungskosten, Wartbarkeit und Benutzerfreundlichkeit [Gronau (2012)].

Ein Unternehmen kann sich für Gesamtlösungen, wie sie SAP oder OpenZ anbieten, entscheiden oder für eine spezialisierte Lösung für jede Abteilung. Beide Szenarien haben Vor-

und Nachteile. So entscheiden sich Unternehmen besonders in den Anfangsjahren für Gesamtlösungen, müssen aber feststellen, dass obwohl Gesamtlösungen die Anforderung der Unternehmen decken, diese leider an ihre Grenzen stoßen. Diese weisen Beschränkungen in Bereichen wie Skalierbarkeit und Funktionalität auf. Besonders in Branchen, die eine ständige Weiterentwicklung und Anpassung benötigen, wird es deutlich [Bunjes u. a. (2002)]. Natürlich bieten die Gesamtsysteme ebenfalls gewisse Vorteile für den Betrieb. So entstehen bei ihrer Verwendung oft geringere Kosten. Diese Kostensenkung entsteht sowohl durch einen geringeren Betreuungs- und Wartungsaufwand als auch durch eine reduzierten Anzahl an Personal, die das nötige Wissen und die Ausbildung zum Betrieb der Systeme benötigt. Dies ist eine notwendige Entwicklung, weil das Budget der IT-Abteilungen in den letzten Jahren sinkt [Stannat und Petri (2004)]. Die Gefahr, die dabei entsteht ist, dass eine Software erworben wird, die mehr bietet, als vom Unternehmen benötigt wird und somit unübersichtlich bei der Handhabung ist [Gronau (2012)].

Daraus ergibt sich die Notwendigkeit der Analyse von Geschäftsprozessen, die einen Ablauf zwischen zwei unterschiedlichen Systemen darstellen. Der Anreiz Spezialsysteme zu verwenden ist definitiv existent. Die Schlussfolgerung aus dieser Beobachtung ist die Schnittstellen zwischen diesen Systemen so zu gestalten, dass sie ein möglichst effizienten Geschäftsfluss ermöglichen.

3.3 Zusammenhänge zwischen Business- und Datenlayer beim Datenaustausch

Die Erklärung der Integration auf unterschiedlichen Schichten wurde bereits im Abschnitt (2.5.2) durchgeführt. In diesem Abschnitt wird die Entscheidung bezüglich des Zugriffs auf die Datenbestände getroffen.

Um die Zusammenhänge beider Layer zu verstehen, müssen zuerst die unterschiedlichen Aufgabenbereiche der Schichten abgegrenzt werden. Der Datalayer ist verantwortlich für die Kommunikation zur Perstistenz. In der Verantwortlichkeit vom Datalayer liegt wie man Objekte schreibt und liest. Das bedeutet, dass der Datalayer die genauen Tabellen in der Datenbank kennt mit den zugehörigen Spaltenbezeichnungen kennt, um die korrekten Attribute einzutragen zu können. Im Datenlayer wird also ausschließlich entschieden, wo Objekte abgespeichert werden. Der Datalayer kommuniziert mit dem Businesslayer. In dem Letzten wird festgehalten, welche Regeln bei der Speicherung der Daten beachten werden müssen. Beispielsweise erfor-

dert das Vtiger-Modul „Contacs“, dass zumindest der Nachname eines Kunden eingetragen wird. Der Businesslayer beinhaltet demgemäß die Geschäftslogik. Diese spiegelt wieder, wie ein Programm arbeitet. Der Businesslayer setzt die Geschäftsregeln durch. Eine solche Regel kann sein, welche Umsatzsteuer in einem Land gilt.

Wieso ist diese Unterscheidung im Kontext von Datenmigration so wichtig? Eine Möglichkeit der Datenmigration besteht darin, die Daten durch einen direkten Zugriff auf die Datenbank, z.B durch das Auslesen der Tabellen mit JDBC und ein anschließendes Überragen in ein weiteres System. Diese Daten können zwar erfolgreich übertragen werden aber dabei wird die Businesslogik umgangen. Dies führt dazu, dass in einem System unmögliche, beziehungsweise unlogische Einträge entstehen, die den Programmverlauf stören können und ferner statistische Erhebungen unbrauchbar machen.

Um dieses Problem zu bewältigen, wurde im Rahmen dieser Arbeit die von Vtiger angebotene REST-API zur Migration der Daten ausgesucht. Diese erlaubt einen begrenzten Zugriff auf die Datenbestände, ohne dabei die Businesslogik zu verletzen. Dies geschieht durch die Abstraktion der im Datalayer gespeicherten Entitäten durch sogenannte Vtiger Objects. Diese setzen sich aus dem Modul und der RessourcenID zusammen. Hierdurch wird der Abruf der Daten aus den unterschiedlichen Modulen ermöglicht. Die Daten, die zum Austausch zwischen dem ERP- und CRM-System festgestellt wurden, befinden sich in den Modulen: „Leads“, „Contacts“ und „Products“

3.4 Betrachtung eines konkreten Geschäftsprozesses

Nach der vorangegangen Analyse soll nun ein Geschäftsprozess untersuchen werden, um die wichtigsten Abhängigkeiten zwischen den beiden System-Typen zu schildern. Anhand dieses Geschäftsprozesses, soll gezeigt werden, an welchen Zeitpunkten Daten von einem System in ein anderes übertragen werden müssen. Die Herkunft des entsprechenden Geschäftsprozesses ist [**?**]. Bei dem Szenario handelt es sich um einen "potentiellen Kunden", der bei einem Call-Center-Mitarbeiter anruft und Fragen zur Bestellung eines bestimmten Produktes zu einem spezifizierten Termin stellt.

1. Als erstes stellt der Call-Center-Agent (CCA) fest, dass es sich um einen Neukunden handelt, da noch keine Daten über ihn im CRM-System eingetragen sind. Zur Erstellung eines Angebots müssen nun die Kundendaten ins CRM-System eingespeichert werden. Diese Daten werden später ebenfalls im ERP-System benötigt.

2. Die vom potentiellen Kunden gewünschten Produkte müssen nun vom CCA im elektronischen Katalog gesucht werden. Der CCA gibt die erforderlichen Produktdaten, sowie die Stückzahlen und den Liefertermin in das CRM-System ein, um ein Angebot zu erstellen.

3. Daraufhin müssen diese Daten in das ERP-System überführt werden, da dieses für die weitere Auftragsabwicklung zuständig ist.

4. Im ERP-System werden die Daten aus dem CRM-System geprüft. Sind die Produkte verfügbar? Welche Konditionen gelten für Neukunden? Das ERP-System informiert: Ein Teil der gewünschten Produkte kann nicht zum geforderten Termin geliefert werden.

5. Der CCA teilt dies dem Kunden mit und muss nun mit dem Kunden eine entsprechende Modifizierung des Auftrags besprechen.

6. Nach der Zustimmung des Kunden über die Änderungen, werden diese vom CCA ins CRM-System eingegeben.

7. Diese Daten müssen ebenfalls wieder im ERP-System geändert werden, um eine korrekte Ausführung zu gewährleisten.

Dieses Beispiel macht schnell deutlich, dass eine Eintragung der Daten von Hand einen viel größeren Arbeitsaufwand für den Mitarbeiter bedeuten würde. Der Mitarbeiter könnte weniger Kunden in seiner Arbeitszeit betreuen. Dies würde eine höhere Zahl an Personal nach sich ziehen und endgültig höheren Kosten zur Folge hätte [?]. Deshalb ist der Einsatz einer Schnittstelle zwischen beiden Systemen empfehlenswert. Da die Daten ad-hoc zwischen beiden Systemen ausgetauscht werden müssen, reicht in diesem Anwendungsfall eine täglich einmalige Synchronisation der Daten nicht aus. Eine Echtzeitsynchronisation ist notwendig, damit die Auftragsabwicklung mit dem Kunden zügig verläuft.

3.5 Schnittstellen Analyse

Um den Geschäftsprozess besser zu Untersuchen, wurde gemäß des Abschnitts (2.1.4) eine Modellierung in eine EPK vorgenommen.

Ein wie in Abbildung (3.1) dargestellter großer Geschäftsprozess ist auf den ersten Blick sehr unübersichtlich. Um aber die entsprechenden Entscheidungen bezüglich des Designs der Schnittstelle treffen zu können, müssen zunächst die kritischen Stellen herausgefiltert werden, die die Interaktionen von Maschine zu Maschine auf Kommunikationsebene beinhalten. Diese müssen wiederum hinsichtlich ihrer realisierbaren Möglichkeiten bei paralleler Abwägung

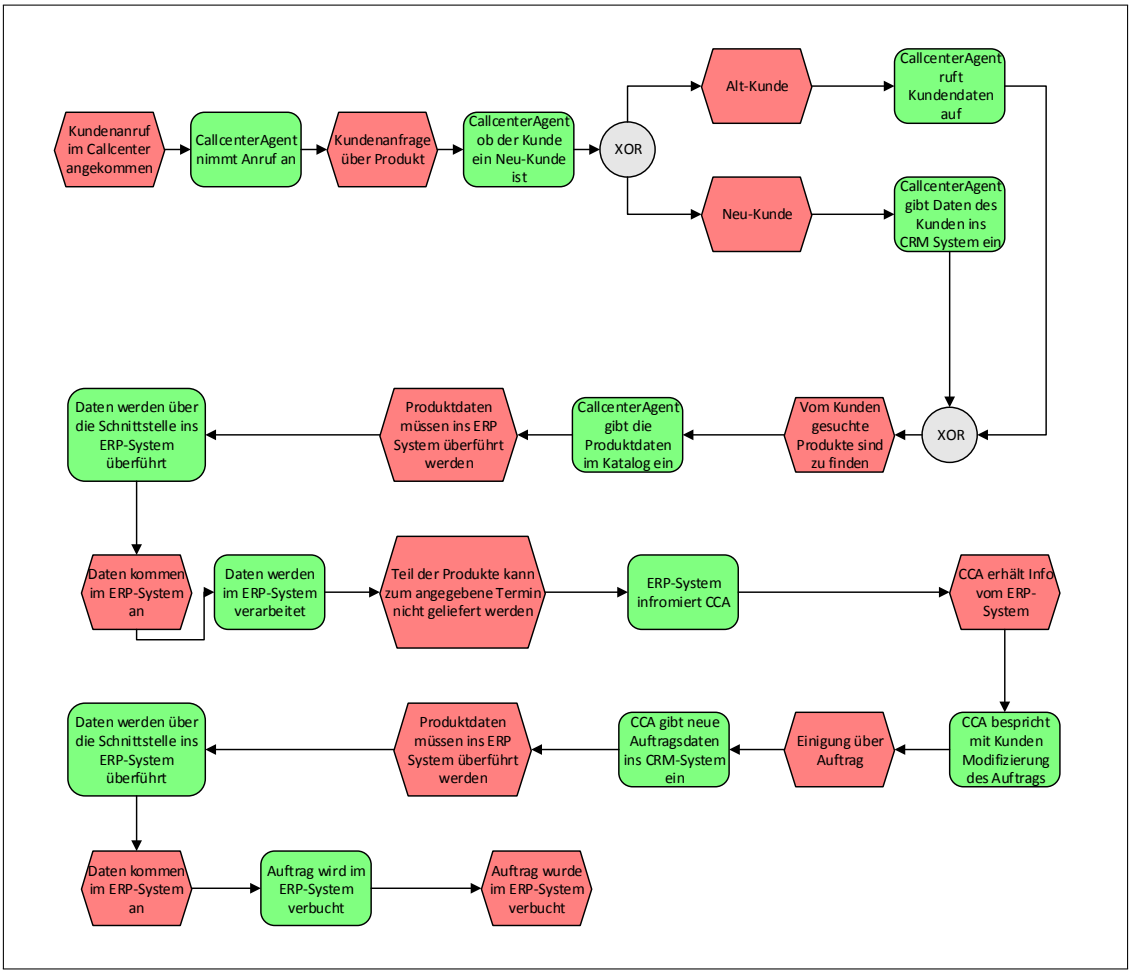

Abbildung 3.1: Callcenter Geschäftsprozess

ihrer Vor- und Nacheile betrachtet werden. Am Ende dieses Prozesses kann dann auf Basis der zugrunde liegenden Daten eine Entscheidung für unseren Software Entwurf getroffen werden.

Der Geschäftsprozess beginnt mit der Interaktion zwischen dem Kunden und dem Callcenter-Agent. Da die Mensch-zu-Mensch Kommunikation für diesen Analyseteil nicht relevant ist, wird die Aufmerksamkeit auf den Abbildung (3.2) gerichtet.

In diesem Geschäftsprozessausschnitt ist zu sehen, dass der Callcenter-Agent im CRM System prüft, ob die Kundendaten vorhanden sind. Es ist sichtbar, dass hier die Daten direkt aus dem CRM-System abgerufen werden, ohne dass das ERP-System beteiligt wird. Hieraus

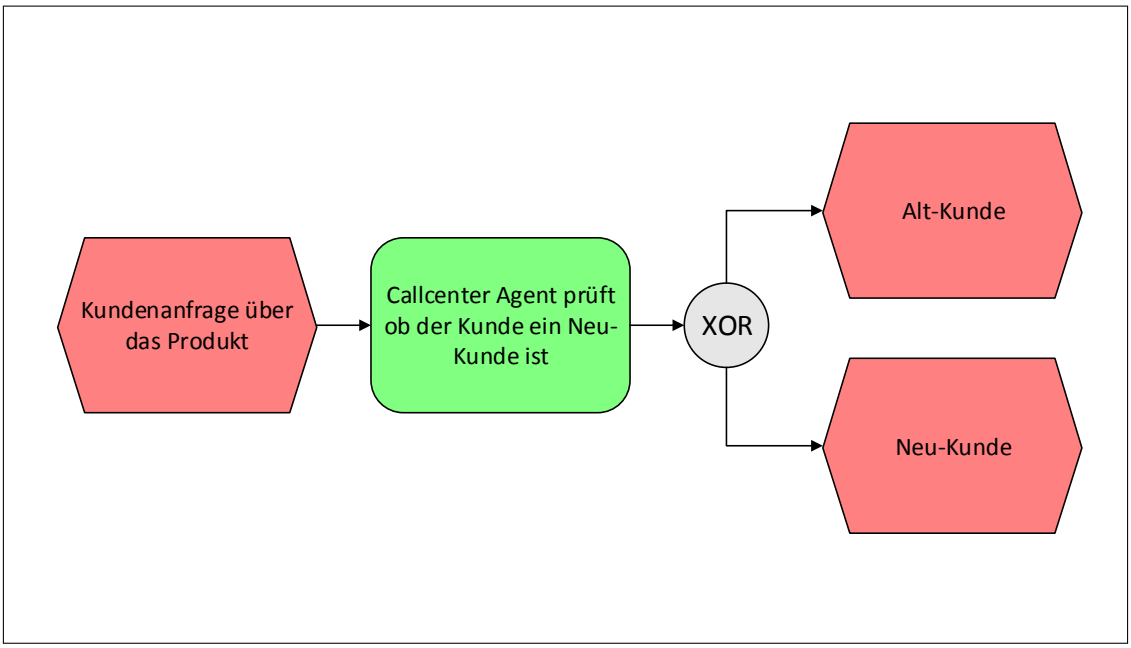

Abbildung 3.2: Der erste Teilausschnitt der EPC

resultieren keine direkten Schlüsse über die Schnittstelle, dennoch muss diese Interaktion betrachten werden, da hier im Fall eines Neu-Kunden Daten erstellt werden. Diese müssen womöglich im späteren Verlauf des Prozesses ins ERP-System übertragen werden. Dies könnte Design Entscheidungen hinsichtlich des Datenaustausch beeinflussen, da sowohl die Menge der Daten als auch die Unterschiede bei der Handhabung der Daten eine Auswirkung auf die spätere Entscheidung bezüglich des Entwurfs haben können.

Anhand des Ausschnitts (3.3) ergibt sich eine eindeutige Übergabe der Daten zwischen den beiden Systemen. Bei der Lagerprüfung stößt das CRM-System an seine Grenzen und benötigt die Informationen aus dem ERP-System. Eine Feststellung ist, dass das ERP-System nicht nur der Lieferant der Daten sein muss, sondern es ebenfalls die Daten aus dem CRM-System benötigt. Es muss also ein Datenaustausch in beide Richtungen stattfinden. An dieser Stelle werden die Daten allerdings zuerst ins ERP-System übertragen, um dort die Kundendaten und die Bestellung abzuspeichern. Die Antwort an das CRM-System findet an einer späteren Stelle im Geschäftsprozess statt. Dennoch wird erfasst, dass das CRM-System in diesem Moment eine Antwort erwartet. Dies ist eine Information, die für das Design der Schnittstelle berücksichtigt werden muss.

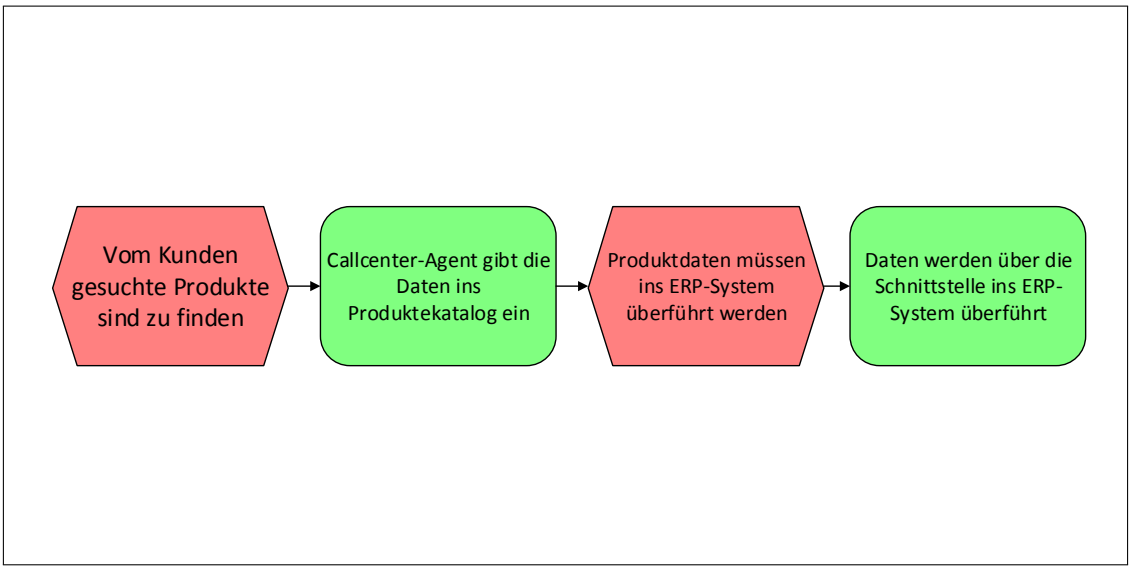

Abbildung 3.3: Übergabe von Daten an das ERP-System

Die Abbildung (3.4) beschäftigt sich mit der Ankunft der Daten ins ERP-System. Die Verarbeitung der Daten ist Teil des internen Aufgabenbereichs des ERP-Systems. Die eingehenden Informationen kommen ins ERP-System an, wo zuerst der Kunde im ERP-System verbucht wird. Anschließend wird anhand der übergebenen Parameter eine Prüfung der Möglichkeiten einer Lieferung vorgenommen. Diese wird durch Bezugnahme auf den derzeitigen Lagerbestand, die Aussicht auf die zukünftige Fertigstellung des Produktes sowie die derzeitige Situation der Zulieferer durchgeführt. In diesem Beispiel ist ein Teil der Produkte zu dem vom Kunden gewünschten Termin nicht verfügbar. Diese Nachricht muss an den Callcenter-Agent, wie auch nach Möglichkeit an das CRM-System, weitergeleitet werden. Eine wichtige Frage, die sich bei Betrachtung des hier analysierten Geschäftsprozesses stellt ist, ob das CRM-System die Möglichkeit bietet, alle im ERP-System benötigten Daten bereitzustellen oder, ob weitere Formen der Dateneingabe benötigt werden. Angaben wie die Menge des Produktes, dass der Kunde erwerben möchte und das Lieferdatum sind für die Verarbeitung im ERP-System erforderlich. Für das Design sollte deshalb beachten werden, dass die Eventualität besteht, dass die Software der Schnittstelle ein Userinterface benötigt, um die gebrauchten Daten ans ERP-System weiterleiten zu können.

In dem darauf folgendem Abschnitt (3.5) wird verdeutlicht, dass das ERP-System den Callcenter Agent benachrichtigt. Es stellt sich immer noch die Frage, ob die Benachrichtigung direkt vom ERP-System kommt, über die weitergeleitet wird oder, ob die Information

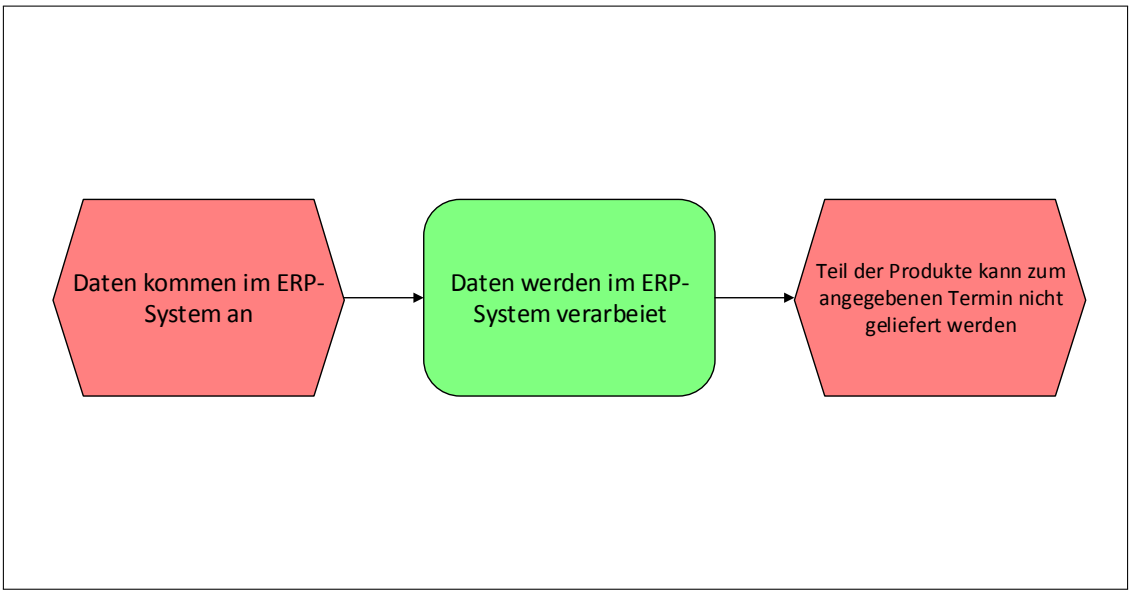

Abbildung 3.4: Ankunft der Daten im ERP-System

vom CRM-System aufgegriffen wird und dieses den Callcenter-Agent in Kenntnis setzt. Diese Fragestellung muss nach ihrer Vor- und Nachteilen beurteilt werden, sowie im Blickwinkel der Implementierungsmöglichkeiten analysiert werden. Da der Geschäftsprozessteil konkret von der Interaktion zwischen Mensch und ERP-System spricht, kann ein Eingreifen des CRM-Systems mit hoher Wahrscheinlichkeit ausgeschlossen werden. Hierauf bezugnehmend bleiben ausschließlich die Optionen von einer durch das ERP-System hervorgerufenen Nachricht sowie eine durch die Schnittstellensoftware ausgelöste Benachrichtigung des Callcenter-Agent. Entsprechend sollte geklärt werden, wie sehr die Schnittstelle einem Service-Bus entspricht oder wie viel sie sich als eigenständige Anwendung verhält, die autonom funktioniert und ihre eigene Geschäftslogik besitzt.

In der Grafik (3.6) findet wieder Mensch-zu-Mensch Kommunikation statt. Der Callcenter Agent bespricht mit dem Kunden die Probleme mit dem Auftrag und klärt mit ihm, ob er eine abgeänderte Bestellung aufgeben will oder ob er auf einen Kauf verzichtet. Um dies korrekt ausführen zu können, muss er konkrete Informationen vom ERP-System erhalten haben, die ihm erlauben einen geänderten Auftrag mit dem Kunden zu besprechen. Ein Parameter dieser Daten ist die Information, zu welchem Zeitpunkt die benötigte Menge des Produktes verfügbar sein wird. Bei dem vorliegenden Szenario kann sich der Callcenter-Agent aufgrund der neu

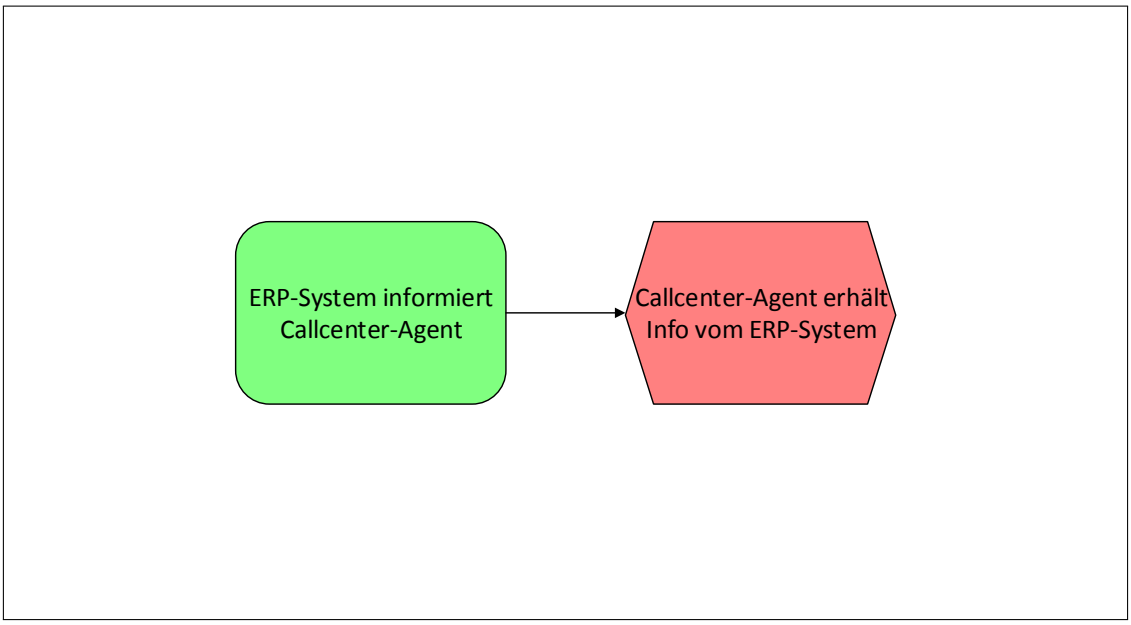

Abbildung 3.5: Benachrichtigung des CCA durch das ERP-System

erhaltenen Daten mit dem Kunden einigen.

Die im vorangegangenen Abschnitt geänderten Auftragsdaten müssen nun in das CRM-System eingetragen werden (Abbildung 3.7), um anschließend ein weiteres mal in das ERP-System überführt zu werden. Der Vorgang ist mit dem im zweiten Abschnitt analysierten Fall identisch (siehe Abbildung 3.3) und muss daher nicht ein weiteres mal analysiert werden, da er keine weitere Anforderung an die Schnittstelle stellt. Dieser Teil ist darüber hinaus ausschließlich nötig, weil das ERP-System ein Problem mit der Auftragsabwicklung festgestellt hat. Ebenfalls könnte der Fall eintreten, dass die Waren zum angegebenen Zeitpunkt im Lager vorhanden sind. Dies ist für das Design der Schnittstellensoftware wichtig. Der duplizierte Vorgang ermöglicht aber den Datenfluss im Konfliktfall zu beobachten und dient aus diesem Grund der Gesamtanalyse.

Schließlich treffen die Daten ins ERP-System an. An dieser Stelle sei es zu vermerken, dass die Interaktion zum Kunden bereits vorüber ist. Ist der Fall eingetreten, dass die Schnittstelle fehlerfrei implementiert ist, und ist das ERP-System zuverlässig in Betrieb, wird an dieser Stelle im Bezug auf den Auftrag keine Komplikation mehr auftreten. Das Kundengespräch kann in dem im fünften Abschnitt besprochenen Geschäftsprozessteil beendet werden. Ein mögliches

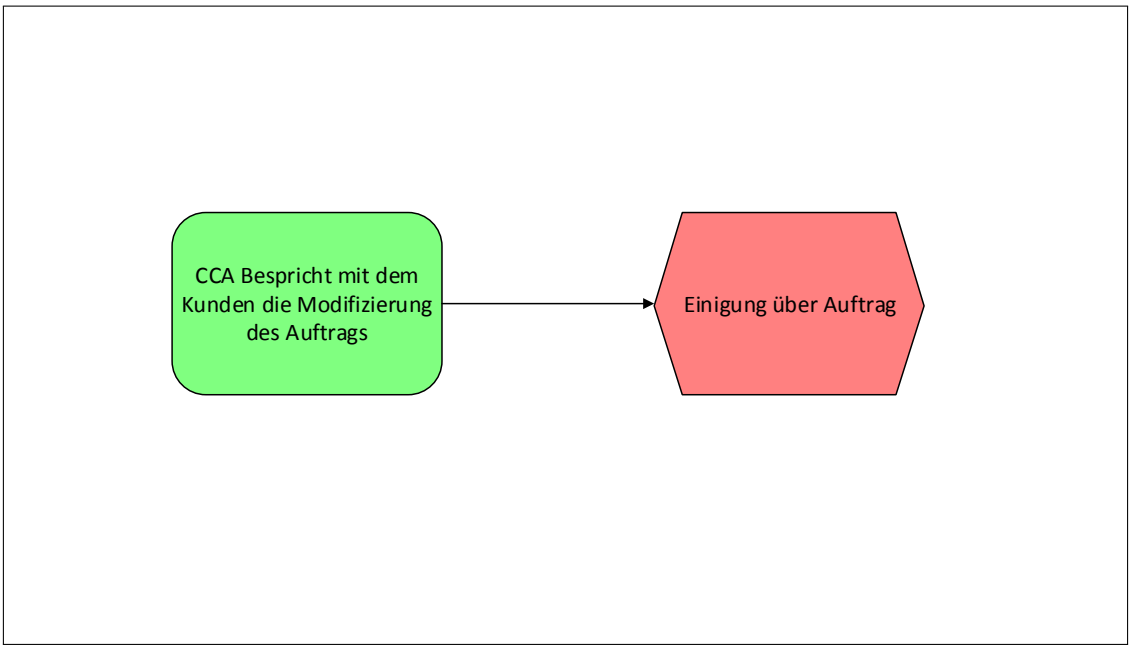

Abbildung 3.6: Abklärung von neuem Liefertermin

Szenario, welches verhindert werden sollte ist, dass während des Gespräches mit dem Kunden ein anderer Callcenter-Agent das gleiche Produkt für den selben Zeittraum für einen anderen Kunden beansprucht. Um dies zu verhindern, wäre eine Reservierung der nicht vorhandenen Produkte zum Zeitpunkt der Übermittlung der ursprünglichen Auftragsparameter benötigt. Ob das ERP-System eine solche Funktion bereitstellt, ist aber außer Reichweite der Verantwortung der Schnittstelle. Die Möglichkeit besteht, dass das ERP-System nach diesem Schritt einen neuen Konflikt mit dem Auftrag registrieren könnte, welcher die Geduld des Kunden unnötig strapazieren würde und das Unternehmen schlecht aussehen ließe.

3.6 Abgrenzung der Aufgabenbereiche zwischen CRM- und ERP-Systemen

Aus den Abschnitten (2.4) und (2.3.3) lassen sich die Aufgabenbereiche beider Systeme relativ klar definieren. Das CRM-System übernimmt in diesem Szenario definitiv eine frontale Rolle, während das ERP-System zumindest aus Sicht des Callcenter-Agent als unsichtbares System agiert.

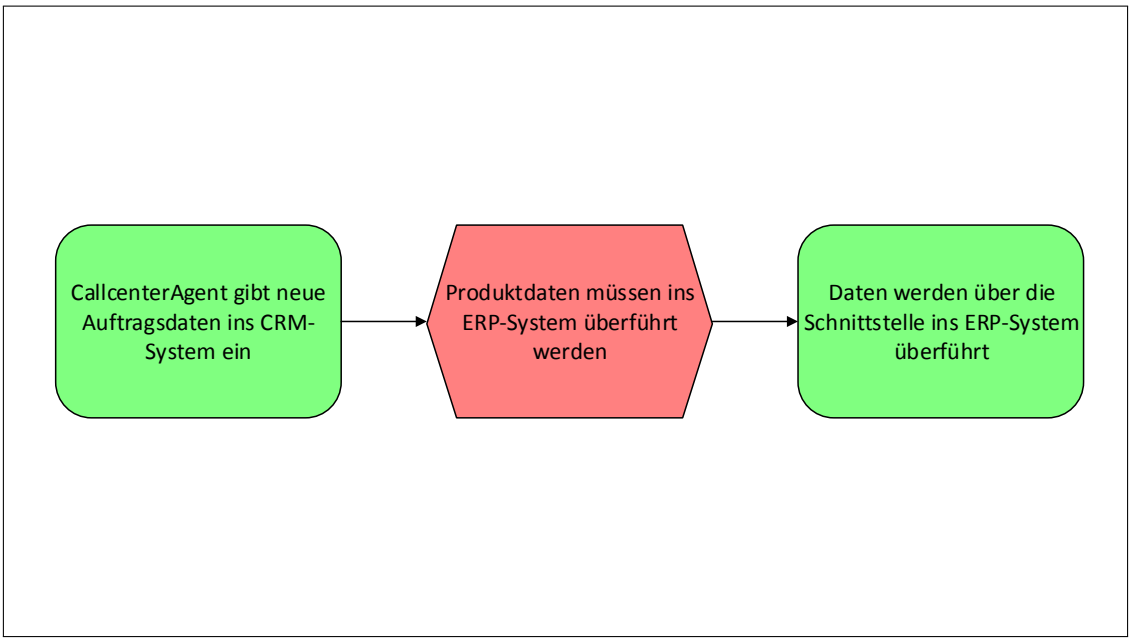

Abbildung 3.7: Änderung der Auftragsdaten

Durch Betrachtung der Module von Vtiger kommt man zu folgenden kritischen Datensätzen vom CRM-System:

- Produktname
- ProduktID
- Lagerbestand
- Menge pro Stück
- Preis pro Stück
- Produkt Code

Damit das ERP-System die Verarbeitung der in unserem Szenario dargestellten Aufgabe ordnungsgemäß ausführen kann, müssen zwei weitere Parameter hinzukommen:

- Bestellmenge
- Lieferdatum

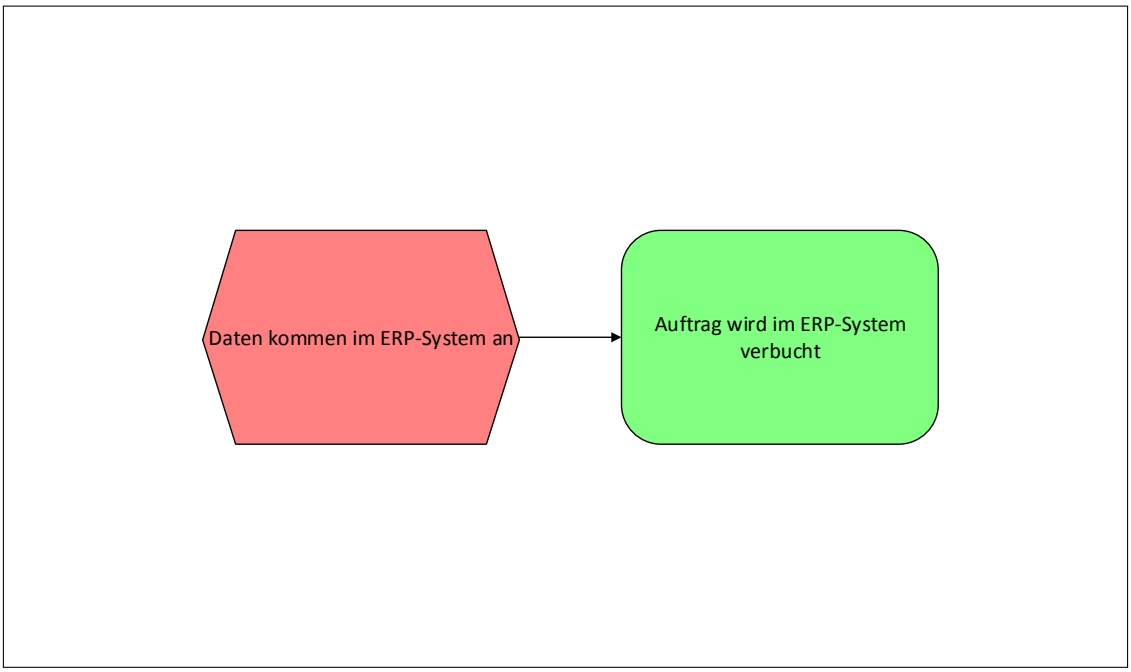

Abbildung 3.8: Speicherung der neuen Auftragsdaten im ERP-System

Anzumerken ist, dass obwohl die Daten wie „Lagerbestand“ oder „Menge pro Stück“ im CRM-System gespeichert werden, ist das ERP-System der Hauptverantwortliche für diese Informationen. Aus diesem Grund werden bei Betrieb der Schnittstellensoftware die Daten aus dem CRM-System aktualisiert. Dies geschieht bei der Anfrage bezüglich des Vorhandensein der vom Kunden gewünschten Ware.

Schließlich soll bei erfolgreichem Abschluss des Verkaufsgespräches der Kunde in das ERP-System übertragen werden. Die hierfür benötigten Daten vom Kunden sind:

- Vorname
- Nachname
- Straße
- Stadt
- Land
- Postleitzahl

- Kundennummer

Besonders die Adressinformationen sind in diesem Zusammenhang sehr wichtig, da diese von der Rechnungskomponente eines ERP-Systems benötigt werden.

Ein Problem das sich oft bei Customer-Relationship-Management-Systemen zeigt, ist der Mangel an Möglichkeiten, die Produktinformationen aus dem vom Kunden aufgegebenen Auftrag zu speichern beziehungsweise zu verwalten. Klassische CRM-Lösungen stellen sich dieser Tatsache, da sie sich meistens auf eine reine Verwaltung der Kundendaten spezialisieren. Hieraus ergibt sich die Frage, auf welche weise Daten, wie der vom Kunden gewünschte Lieferzeitpunkt, ins ERP-System überragen werden. Der zugrunde liegende Sinn dieses Szenarios ist, dass der CCA keinen Kontakt mehr zum ERP-System hat. Aus diesen Annahmen kann entnommen werden, dass die Benutzeroberfläche des CRM-Systems erweitert werden muss oder die Schnittstellensoftware eine solche Gelegenheit der Eingabe bereitstellen muss.

Diese Problematik ist bei Anwendungslandschaften, bei denen viele individuelle Einzellösungen kooperativ arbeiten, nicht unüblich. Der Anwender muss sich demnach im schlimmsten Fall mit mehreren Benutzeroberflächen gleichzeitig auseinandersetzen. Der Idealfall hingegen sieht in einer solchen Konstellation den Bau einer MiddleWare vor, die eine neue Benutzeroberfläche anbietet, welche das eingesetzte System um die nötige Funktionalität erweitert [Bunjes u. a. (2002)].

Das ERP-System wiederum versorgt das Anwendungssystem mit den nötigen Informationen bezüglich der Fähigkeit des Unternehmens, die vom Kunden gewünschte Ware zum geforderten Termin zu liefern. Es beansprucht dabei nicht nur die Parameter des vom Kunden gewünschten Produktes sondern ebenfalls die Kundendaten selbst. Der Grund hierfür ist, dass das ERP-System die Daten braucht, um sie selbst zu verwalten oder sie gegebenenfalls an weitere Systeme, wie das für die Rechnung zuständige System (Billing) weiterzugeben [Leimeister (2015)]. Hierzu wurde im Abschnitt (2.4.3) bereits Stellung bezogen.

3.7 Vergleich zwischen Echtzeitkopplung und Synchronisation zu bestimmten Zeiten

Betrachtet man Schnittstellen zwischen zwei Anwendungen stehen sich zwei Grundtypen gegenüber. Auf der einen Seite sind Schnittstellen, die den Datenaustausch zwischen zwei Systemen zur Echtzeit durchführen, und auf der anderen Seite stehen Schnittstellen, die in vorgegebenen Zeiten oder in gewissen Abständen Daten aus beiden Systemen sammeln um

diese anschließend in beiden Anwendungen zu aktualisieren. Dies wurde im Abschnitt (2.5.2) näher erläutert.

Synchronisation, die nicht Echtzeitgesteuert ist, hat ihre Vorteile bei Systemen bei denen Aktualität nicht zu den kritischsten Anforderungen gehört. Sollte ein Unternehmen entscheiden, dass die Informationen aus dem Billing-System nicht umgehend in ein anderes System übertragen werden müssen, sondern nur ein mal täglich werden aktualisiert werden sollen, so kann ein Update der Persistenzen der beiden Anwendungen zu einer festgelegten Zeit umgesetzt werden, ohne dass der Geschäftsfluss beeinträchtigt wird. Ein solches Vorgehen ist allerdings nur dann empfehlenswert, wenn eine Interaktion mit Personen als Aktoren nicht vorliegt oder ein Individuum nicht direkt an einem Geschäftsprozess beteiligt ist. In einem solchen Fall kann ein Zugriff und Austausch der Daten direkt durch ein Datenbankenzugriff oder unter Beachtung der Businesslogik über die zur Verfügung gestellten APIs durchgeführt werden. Ebenso wird bei Abruf von Daten, die zur Bildung von Langzeitstrategien eines Unternehmens verwendet werden, der umgehende Austausch zu einer sekundären Anforderung und ist damit verzichtbar.

Auf der anderen Seite sind Prozesse, bei denen der Kundenkontakt oder die Interaktion zu anderen Aktoren im Mittelpunkt steht. Diese Prozesse sind dringend auf einen gegenwartsnahen Stand der Informationen angewiesen. Wie an unserem Beispiel erkennbar ist, braucht ein Mitarbeiter bei der Verkaufsabwicklung den aktuellen IST-Stand des Lagers und kann somit einem potentiellen Kunden keine Auskunft auf Basis von veralteten Informationen geben.

3.8 Schlussfolgerung und Zielsetzung für die Implementierung

Für eine Entscheidung bezüglich des Entwurfs einer Schnittstelle werden die bisher gewonnen Erkenntnisse aus der Analyse zusammenfasst. Aus diesen gesammelten Informationen wird der Entschluss für den Aufbau der Schnittstelle vorgenommen.

Wir konnten feststellen, dass Speziallösungen für Anwendungslandschaft eine attraktive Option für Unternehmen darstellen [Stannat und Petri (2004)]. Dies wurde durch Attribute wie Anpassungsfähigkeit, Erweiterbarkeit und Übersicht für die Angestellten die diese Systeme verwenden, hervorgehoben. Unternehmen können dadurch in dynamisch ändernden Märkten konkurrenzfähig bleiben. [Bunjes u. a. (2002)]

Im weiteren Verlauf kamen wir zum Schluss, dass das Umgehen der Businesslogik durch direkte Datenbankabfragen nachteilig ist. Der Einsatz von MiddleWare, die sich Technologien

wie JDBC bedient, um einen direkten Zugriff auf die Persistenz zu erlangen, kann Logik der Anwendung aushebeln und somit statistische Ergebnisse verfälschen bzw. zu Fehlern bei den einzelnen Anwendungen nach einer Synchronisation führen. Dieser Einsatz eignet sich wenn man Daten ausliest, da die Gefahr der Kompromittierung nicht besteht.

Anschließend wurde ein konkreter Geschäftsprozess betrachtet. Bei seiner Analyse wurde zunächst der Aufbau studiert. Dann wurde der in Textform vorliegende Geschäftsprozess in ein EPK umgewandelt, damit er effizienter beurteilt werden konnte. Die einzelnen Ausschnitte wurden inspiziert, in dem man den Geschäftsprozess in sieben Teilprozesse aufgespielte um sie einer genaueren Analyse zu unterziehen. Dieser Analyse entsprangen die folgenden wichtigen Punkte:

- Der Callcenter-Agent hat ausschließlich Kontakt mit dem CRM-System und womöglich mit der Schnittstellenanwendung
- Der Informationsfluss führt zwischen den beiden Anwendungen in beide Richtungen
- Produktinformationen wie Verfügbarkeit und andere Parameter wie Lieferanten werden im ERP-System verarbeitet
- Auskunft des Callcenter-Agents findet nicht vom CRM-System statt, sondern vom ERP-system, wodurch eine Benutzeroberfläche in Erwägung gezogen werden sollte

Der letzte Punkt wird durch eine bekannte Problematik in der Industrie verdeutlicht. Systeme werden um eine weitere Oberfläche erweitert. Eine Vereinigung von der CRM-System-Benutzeroberfläche und dieser Oberfläche der Schnittstellensoftware zu einer gemeinsamen Benutzeroberfläche wäre optimal, würde den Rahmen dieser Bachelorarbeit sprengen. Die Benutzeroberfläche der Schnittstellensoftware soll einerseits die benötigten Zusatzinformationen für das ERP-System aufnehmen, andererseits für die Benachrichtigung des Callcenter-Agents zuständig sein. Parallel dazu werden die eingegebenen Daten durch Datensätze aus dem CRM-System erweitert. Diese werden durch eine vom CRM-System zur Verfügung gestellte API abgerufen. Während des Programmablaufs werden diese Daten über eine zentrale Komponente der Schnittstellensoftware an das ERP-Systems weitergegeben. Der Adapter des ERP-Systems fällt schmaler aus, da eine Mensch-Maschine Kommunikation nicht stattfindet und wir darüber hinaus das ERP-System nur simulieren.

4 Implementierung und Design

4.1 Vtiger

Vtiger CRM ist eine freie Open-Source-Software für das Customer-Relationship-Management (CRM), die von dem gleichnamigen Unternehmen Vtiger, im Jahre 2004 entwickelt wurde. In der Arbeit wird die Version (6.1.0) benutzt.

Geschichte

Vtiger ist eine Abspaltung von SugarCRM mit dem Ziel, eine quelloffene CRM Software zu erschaffen, die den Funktionsumfang von SugarCRM und Salesforce.com abdeckt.

Aufbau

Die Software basiert auf der Scriptsprache PHP und auf dem Datenbankmanagementsystem MySQL. Die unterstützten Betriebssysteme sind Windows, Linux sowie Unix-Systeme.

Weitere Informationen können unter http://www.vtiger.de/de/ gefunden werden.

4.2 Flussdiagramm des Ablaufs mit der neuen Software

Basierend auf der durchgeführten Analyse wurde ein Flussdiagramm erstellt, in dem der Arbeitsablauf des Callcenter-Agents, der mit unserer neuen Software arbeiten sollte, aufzeigt wird. Zuerst loggt sich der Mitarbeiter in Vtiger ein, um anschließend mit den ihm ersichtlichen Zugangsdaten ein Einloggen in unsere Schnittstellensoftware durchzuführen. Der „Login-Vorgang“ wird näher im Abschnitt (4.6.2) erläutert.
Nach dem erfolgreichen Einloggen ist der Callcenter-Agent bereit, Kundenanrufe entgegenzunehmen. Mit der Annahme des Kunden, startet er eine neue Kundensession. Der Mitarbeiter prüft in Vtiger, ob der Kunde bereits im System registriert ist. Falls das nicht der Fall ist und es sich somit um einen Neukunden handelt, trägt er die Kundendaten in Vtiger ein. Nach diesem Vorgang kann er die Kundendaten in die Schnittstellensoftware übertragen, indem er

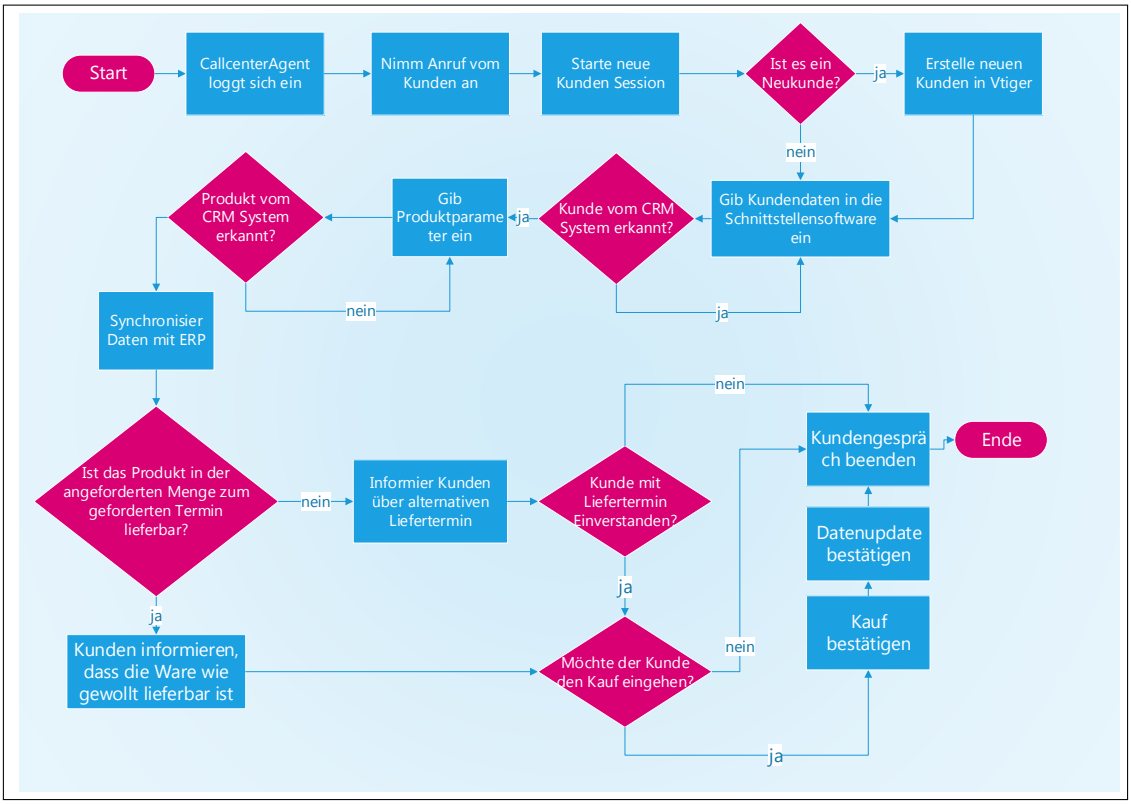

Abbildung 4.1: Flussdiagramm des Gesamtablaufs

den Kundennamen im ersten Fenster ausfüllt. Sollte der Kunde bereits bekannt sein, reicht ein simples Eintragen in die Schnittstellensoftware. Ist der Fall eingetreten, dass der Kunde nicht von der Schnittstellensoftware erkannt wurde, ist beim Eintragen der Daten in Vtiger oder beim Eintrag des Kundennamens in die Schnittstellensoftware ein Fehler begangen worden und der Mitarbeiter muss an dieser Stelle nachbessern.
Als nächstes zeigt die Schnittstellensoftware ein neues Fenster für den Callcenter-Agent, wo er das vom Kunden angefragte Produkt zusammen mit dem vom Kunden gewünschten Termin und der von ihm gewollten Stückzahl eintragen kann. Zunächst wird die Software die Produktdaten überprüfen um, im Anschluss diese zu validieren. Den Erfolg wird an der Benutzeroberfläche bestätigt. Sind die Daten inkorrekt, wird der Callcenter-Agent aufgefordert, diese zu überarbeiten. Nachdem korrekte Daten eingetragen wurden sind, ermöglicht die Schnittstellensoftware dem Mitarbeiter eine Synchronisation der Produktdaten mit dem ERP-System. Auf diese Weise erhält er Auskunft, ob das Produkt zum gewünschten Termin lieferbar ist. Ist eine Lieferung zum gewollten Zeitpunkt nicht möglich, liefert die Benutzeroberfläche der

Schnittstellensoftware dem Callcenter-Agent den nächstmöglichen Liefertermin. Ist der Kunde mit dem Liefertermin einverstanden, kann der Callcenter-Agent mit einem Knopfdruck den Bestellvorgang erfolgreich abschließen. Hiermit werden die erforderten Daten sowohl im CRM als auch im ERP System synchronisiert. Anschließend sieht der Mitarbeiter das Fenster, auf dem er eine neue Kundensession erstellen oder das Programm beenden kann. Tritt dieser Fall nicht ein und der Kunde den Handel nicht durchführt, wird die Kundensession verworfen und der Callcenter-Agent gelangt ebenfalls im Fenster mit der Kundensession-Erstellung an.

4.3 Design der Schnittstelle aus dem Ergebnis der Analyse

Aus der Analyse des betrachteten Geschäftsprozesses, kamen wir zum Schluss, dass die von uns entwickelte Schnittstelle keine reine Integration auf Datenebene durchführen sollte, da die Abwicklung des Verkaufs die nötigen aktuellen Informationen dem Callcenter-Agent zur Verfügung stellen muss. Eine simple Synchronisation der Datenbanken wäre damit ausgeschlossen, weil diese nur den Lagerbestand seit dem letzten Update widerspiegeln. Hieraus resultierend bleibt die Anbindung zur Laufzeit, bei der in unserem Fall das ERP-System die Funktionalität des CRM-Systems um Produktinformationen erweitert.

Das Lösungskonzept könnte auf unterschiedlichste Art umgesetzt werden. Die erste Möglichkeit war eine Änderung des Vtiger Codes. Diese erschien aus mehren Gründen wenig attraktiv. Zuerst mangelte es an Kenntnissen der Plattformsprache PHP. Dazu kam die Unvorhersehbarkeit von Änderungen am Programmcode und deren Auswirkungen. Aus diesem Grund wurde entschieden, die Applikation um eine weitere Benutzeroberfläche zu erweitern. Die optimale Lösung hierfür wäre eine gemeinsame Vereinigung von Vtiger-GUI und der in der Arbeit entwickelten Benutzeroberfläche. Das war jedoch wegen des zu erwartenden erheblichen Arbeitsaufwand im Rahmen dieser Bachelorarbeit nicht durchführbar. Dies würde zusätzlich durch die Problematik von PHP als Programmiersprache erschwert werden.

Der Anwender nutzt die native Benutzeroberfläche von Vtiger, um die Kundendaten zu verwalten und ruft die Oberfläche der Schnittstellensoftware auf, um die Produktdaten einzutragen. Die Benutzeroberfläche der Schnittstelle löst den weiteren Funktionsverlauf aus. Die vom Anwender angeforderten Informationen werden ebenfalls auf der Schnittstellen-GUI eingeblendet. Es sei an dieser Stelle vermerkt, dass die Vtiger-Software eine umfangreiche Modulsammlung zur Verfügung stellt, die die Kernanwendung um weitere Funktionalität erweitert. Das Produkt- sowie das Lead-Module wurden in Vtiger aktiviert. Das Lead-Module

erlaubt die Verwaltung von potentiellen Kunden. Das Produkt-Modul ermöglicht die Darstellung der Produktdaten in der CRM-Anwendung. Außerdem aktualisiert man die Produktdaten wie den Lagerbestand, in diesem Modul mit den Informationen, die man als Antwort vom ERP-System erhalten hat.

Die Schnittstellensoftware soll nun die Kundendaten und die Produktdaten aus den CRM- und ERP-System austauschen können. Dabei hält sich unser Entwurf an den durch den Geschäftsprozess vorgegebenen Ablauf. Die Benutzeroberfläche wird kompakt gehalten, damit der Mitarbeiter nicht mehr Funktionalität auszusetzen als nötig.

Die verwendete Programmiersprache zur Implementierung der Schnittstelle ist Java. Die JRE Stufe ist 8. Die zum programmieren benutzte IDE ist IntelliJ IDEA (Version 14).

Die Zugriff auf die Kundendaten auf Seite von Vtiger wird durch den auf Grundlage der Webservices angebotenen **REST** API durchgeführt. Hierbei wird die von Vtiger angebotene Java-Library **vtwsclib** v1.5 verwendet, die wiederum auf der **HTTP Components** Library basiert. Außerdem wird die **json-simple** Library verwendet.

4.4 Komponenten

- **Initialisierungsmanagement** - Das Initialisierungsmanagement enthält die Mainfunktion. Diese Komponente ruft nacheinander die anderen Komponenten über ihre Interfaces I(Komponentenname)Mgmt auf und gibt die benötigten Logikschnitstellen I(Komponentenname)Logic and die Komponenten, die die jeweilige Funktionalität benötigen weiter. Außerdem enthält die Komponente zwei Enums, die jeweils die zu initialisierenden CRM- und ERP-Systeme definieren.
- **Kernmanagement** - Prüft die Daten aus den Quellsystem und wandelt sie nach Bedarf in das Datenformat des Zielsystems. An dieser Komponente sind die Adapter der Anwendungen angebracht.
- **Vtiger Adapter** - Übernimmt die gesamte Kommunikation mit Vtiger über die REST API. Dafür implementiert sie den WebserviceClient von Vtiger. Sie übernimmt somit die Aufgaben des Login, das Auslesen der Kundendaten und deren Update. Sie wird über ihr Logik-Interface von der Fassade aufgerufen. Diese Komponente übernimmt außerdem

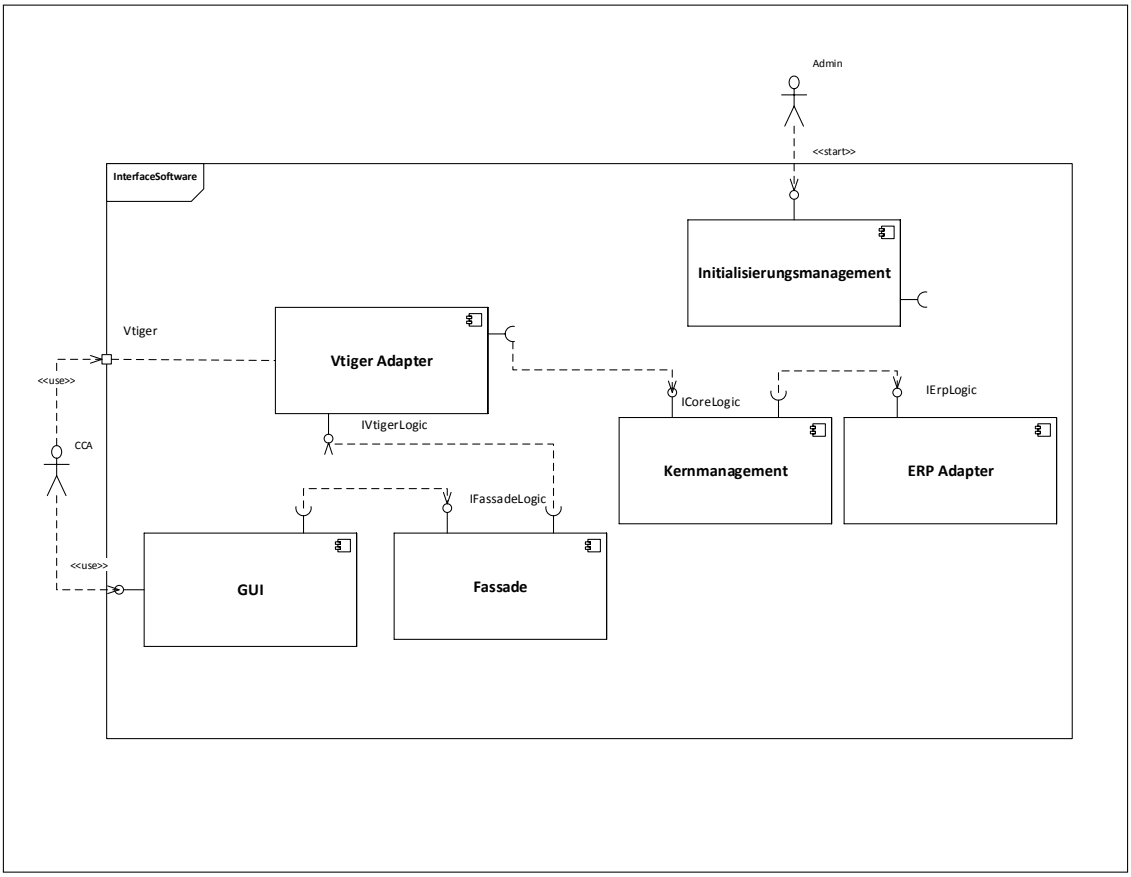

Abbildung 4.2: Komponentendiagramm der Schnittstelllensoftware

die Erzeugung des CustomerSession Objektes mit dem dazugehörigen Customer und Product Objekten und übernimmt deren Aktualisierung

- **Fassade** - Kapselt die Funktionalität von ICRMLogic, auf die die GUI zugreift, und überprüft die Formatierung der vom Benutzer eingegebenen Daten.
- **GUI** - Die Benutzeroberfläche, die der Callcenter-Agent zur Erweiterung der Funktionalität von Vtiger nutzt. Sie enthält die Fenster zum Login, der Eingabe der Kundendaten sowie der Produktdaten. Darüber hinaus enthält sie ein Menü-Fenster. Die Erstellung der GUI wurde mithilfe von JavaFX durchgeführt.
- **ERP Adapter** - Der Adapter für die Schnittstelle zum ERP-System. In unserem Fall ist die konkrete Implementierung ein gemocktes System

4.5 Initialisierungsreihenfolge

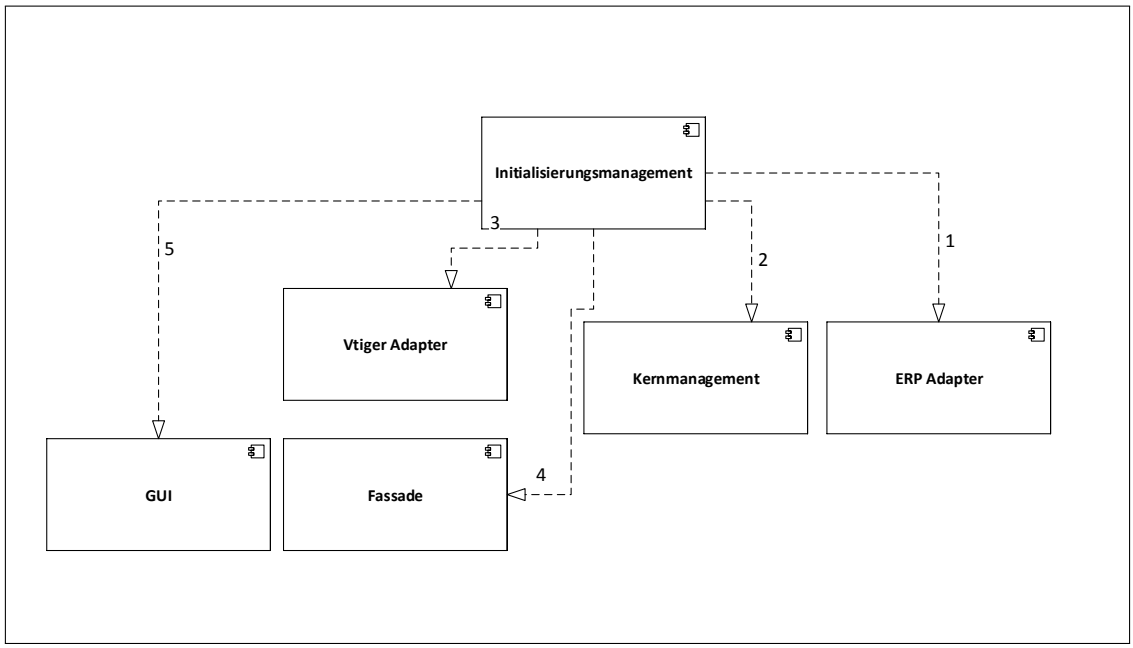

Abbildung 4.3: Initialisierungsreihenfolge der Komponenten

In der Abbildung (4.3) ist die Aufrufreihenfolge aufgezeigt, nach der die einzelnen Komponenten initialisiert werden. Dies wird durch die Komponente **Initialisierungsmanagement** durchgeführt. Die einzelnen Komponenten kapseln die Initialisierung in ihren Management Interfaces zB: **ICoreMgmt** ab. Diese Schnittstellen werden ausschließlich vom Initialisierungsmanagement genutzt, das diese aufruft und die Logik-Interfaces zB. **ICoreLogic** ,erzeugt und diese an die Komponenten weiterreicht, die Zugriff auf die entsprechende Komponente haben dürfen.

4.6 Laufzeitsicht

Die Laufzeitsicht wird von [Starke und Hruschka (2011)] als Beschreibung von Komponenten zur Laufzeit vorgestellt. In dieser Sicht sollten Szenarien, die die wichtigsten Abläufe darstellen, vorgestellt werden. Um diese herauszufinden, sollten Kriterien betrachtet werden, wie der wichtigste Anwendungsfall. Darüber hinaus sollten Fragen geklärt werden, wie die Laufzeitkomponenten mit der Umwelt interagieren oder wie das System gestartet wird. Dabei sollten nach Möglichkeit nur die wichtigsten Fälle betrachtet werden. Jedes Szenario sollte weiterführend durch Diagramme mit zusätzlicher Dokumentation erläutert werden.

Aus diesem Grund hat man sich entschieden, die Zustandsübergänge der Benutzeroberfläche als Zustandsdiagramm darzustellen, da die Benutzeroberfläche für den Anwender das meiste Berührungspotential beinhaltet. Überdies erstellen wir zwei Sequenzdiagramme, wovon eines den „Login-Vorgang“ erläutert und der andere sich mit dem Anwendungsfall eines Kundengespräches auseinandersetzt .

4.6.1 Zustandsübergänge der Benutzeroberfläche

Die Abbildung (4.4) zeigt einen simplen Zustandsautomaten, der sich einzig mit den Übergängen der Benutzeroberfläche der Schnittstellensoftware auseinandersetzt. Jeweils denkbare Einbezüge der Vtiger-Benutzeroberfläche sind absichtlich nicht dargestellt. In dieser Darstellungsform werden wir im Gegensatz zu den Sequenzdiagrammen mögliche Verzweigungen im Ablauf darstellen.

Durch Starten der Anwendung landet der Benutzer im Fenster mit dem Titel „Login“. Hier kann er durch Eingabe seiner Zugangsparameter ein Einloggen der Software zur Vtiger Schnittstelle erreichen, siehe dazu (4.6.2). Nach Betätigen der „Login“ Taste wechselt die GUI in die „Menu“ Stage. Hier steht dem Anwender die Wahl entweder das Programm mit Drücken der „Exit“ Taste zu schließen oder durch das Betätigen der “Next Customer„ Taste den Übergang in den „Customer“ Stage zu vorführen.

Im „Customer“ Stage bietet sich dem Anwender eine simple Nutzeroberfläche, bei der er durch Eingabe eines Vor- und Nachnamens von einem Kunden dessen Existenz in der Vtiger Datenbank überprüfen kann. Im Eintreten des Anwendungsfalls, dass der spezifizierte Kunde vorhanden ist, wird durch Auslösen der „Selec“ Taste ein Wechsel in den „Product“ Stage vorgeführt. Sollte der Kunde auflegen oder das Verkaufsgespräch auf andere Art beenden,

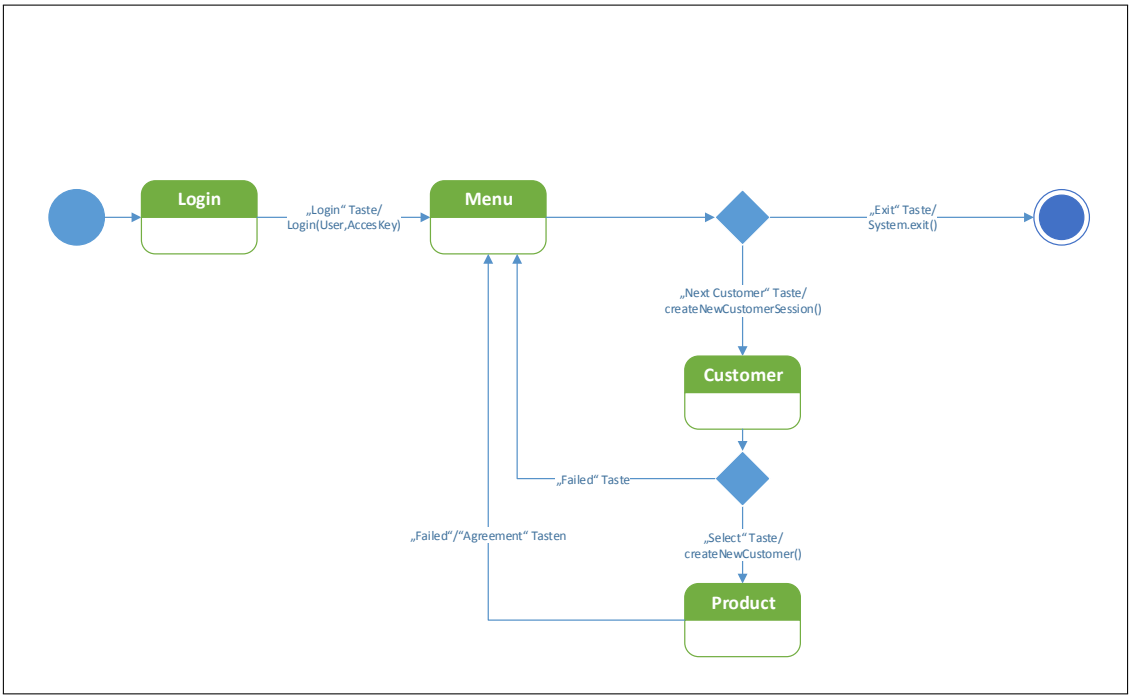

Abbildung 4.4: Zustandsdiagramm der Benutzeroberfläche

so kann der Callcenter-Agent durch Drücken der „Failed“ Taste wieder in den „Menu“ Stage wechseln. Eine fehlerhafte Eingabe oder ein im System nicht registrierter Kunde werden durch eine Ausgabe in einem Textfeld visualisiert.

Bei der Ankunft des Mitarbeiters im „Product“ Stage, sieht er drei Texteingabefelder vor sich, die er mit den Angaben zum Produktnamen, der Anzahl vom Kunden gewünschten Produktmenge und dem vom Kunden vorgeschlagenen Liefertermin füllen soll. Ist die Eingabe vollständig, wird durch Betätigen der „Select“ Taste der Prüfvorgang des Produktnamens gestartet. Durch die erfolgreiche Erkennung des Produktnamens, wird eine weitere, zuvor deaktivierte Taste „Synchronize“ aktiviert. Durch Drücken der Taste wird die Anforderung der Produktparameter aus dem ERP-System durchgeführt. Diese liefert eine Bestätigung der Auswahl oder die Information über einen alternativen Liefertermin auf dem entsprechendem Informationstextfeld. Ebenfalls wird die „Agreement“ Taste aktiviert. Durch den Input dieser Taste wird der Kauf abgeschlossen und eine Synchronisation der Lagerbestände sowie Kundendaten beider Systeme durchgeführt. Ein Fokussieren der Eingabefelder deaktiviert die „Synchronize“ und „Agreement“ Tasten, um ein Abgleich unstimmiger Daten zu verhindern. Der Anwender ist gezwungen, über die „Select“ Taste seine Auswahl wieder zu bestätigen.

Damit ist die GUI für den korrekten Programmdurchlauf verantwortlich. Die Kundensession kann, wie im Vorgängerstage, jederzeit durch Drücken der „Failed“ Taste beendet werden, wodurch der Anwender im „Menu“ Stage landet. Dort kann er eine neue Kundensession erstellen oder die Anwendung beenden.

4.6.2 Login Vorgang

Der auf Seiten unserer Schnittstellensoftware in Abbildung (4.5) aufgeführte Login Vorgang, hat zwei Stufen.

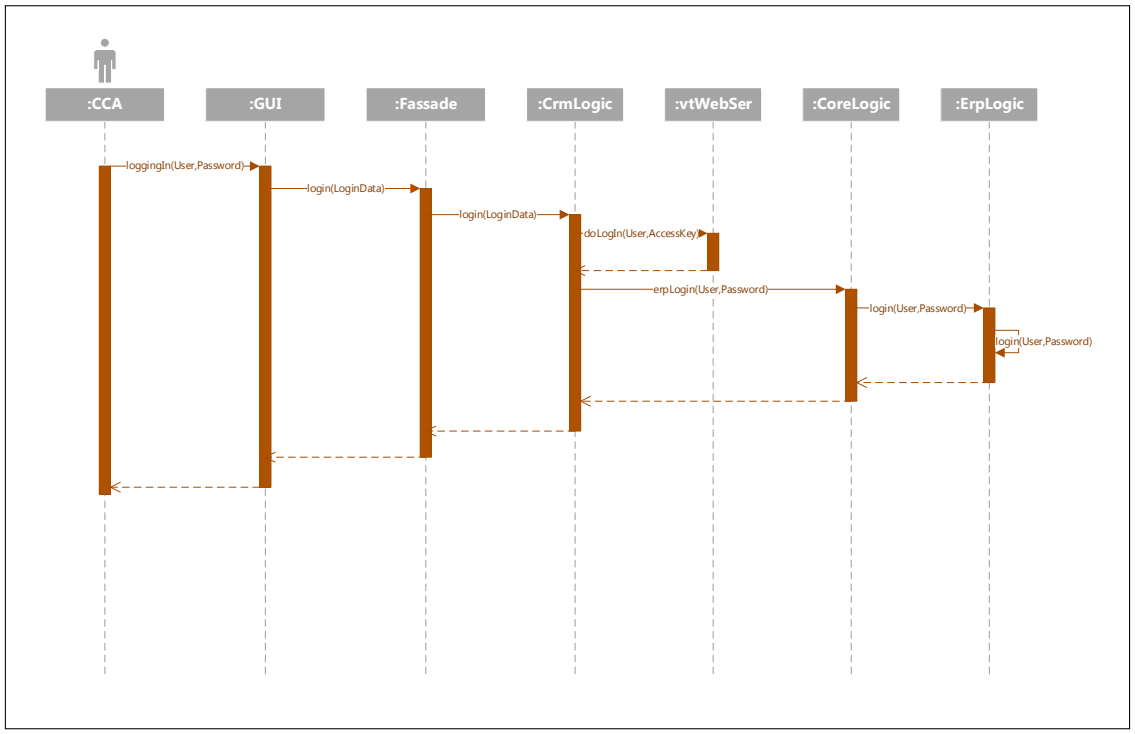

Abbildung 4.5: Sequenzdiagramm des Login-Vorgangs

Die erste Stufe ist ein Einloggen in Vtiger. Man gibt zunächst im Vtiger Webclient seinen Benutzernamen und sein Vtiger Passwort ein. Erfolgreich eingeloggt findet man unter dem Menüpunkt: “Meine Einstellungen“ seinen Zugangsschlüssel (siehe Abbildung 4.6). Mit diesem startet der Anwender unsere Schnittstellensoftware. Beim Login-Verfahren (siehe Abbildung 4.7) der Schnittstellensoftware soll der Callcenter-Agent (CCA) den selben Benutzernamen eingeben, wie er es von Vtiger gewohnt ist und verwendet den aus Vtiger gewonnen Zugangs-

schlüssel als Passwort in der Schnittstellensoftware.

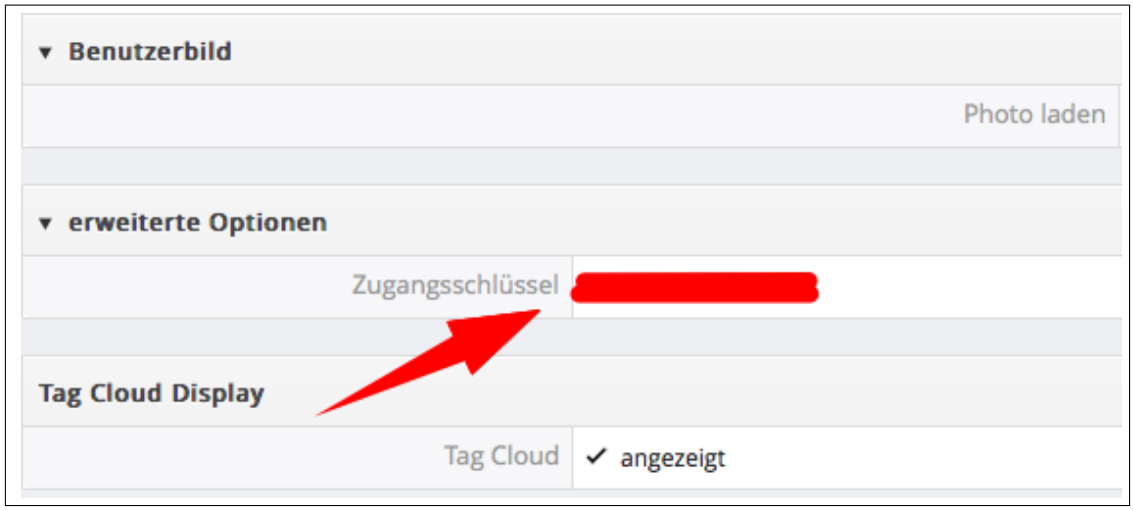

Abbildung 4.6: Ausschnitt aus dem Menüpunkt: Meine Einstellungen Vtiger

Das in Abbildung (4.5) dargestellte Szenario wird nun ausgelöst. Die in der GUI vom Anwender eingegebenen Daten werden an Fassade weitergegeben. Diese greift auf die Logik vom CRM System zu. Diese implementiert die Webservice API. Die CrmLogik speichert die Login-Daten in zwei Attributen User und AccesKey und führt anschließend die doLogin() Methode des Webclients aus. Anschließend wird ein Testlogin durchgeführt, um auf diese Weise festzustellen, ob das vom Benutzer eingegebene Passwort zum Nutzernamen passt. Später ruft die CrmLogik die CoreLogic auf, die wiederum die ErpLogic aufruft. Dort wird, weil unser ERP-System gemockt ist, mit den Login-Daten ein simulierter Login-Vorgang durchgeführt. Um diesen simulierten Vorgang zu vereinfachen, nehmen wir an, dass im ERP-System die gleichen Login-Datenwie im ERP-System benötigt werden. Wenn in der CrmLogik die Antworten vom VTWebclient sowie vom ERP System zurückkommen, prüft die CrmLogic, ob beide erfolgreich waren. Im Fall vom ERP-System ist aus vorher erwähnten Gründen der Vorgang immer erfolgreich. Sowie beide Methodenaufrufe erfolgreich sind, schließt die GUI den Vorgang ab und wechselt in das nächste Fenster.

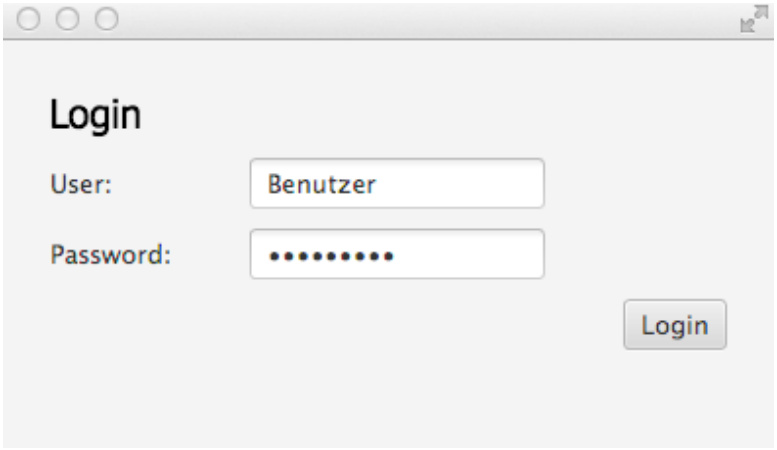

Abbildung 4.7: Login-Fenster der Schnittstellensoftware

4.6.3 Kundensession

Die Kundensession ist das Kernszenario der gesamten Anwendung. In diesem Anwendungsfall liegt das Herz der Schnittstelle. Nach Ausführen des im Abschnitt (4.6.2) vorgestellten Login-Vorgangs, steht der Anwender vor der Menu-Seite (siehe Abbildung 4.8). Das Szenario wird in der Abbildung (4.15) dargestellt. Hier wird durch Drücken der „Next Customer“ Taste eine neue Kundensession gestartet. Dabei wird die CrmLogik über den Umweg durch die Fassade aufgerufen und ein neues Objekt vom Typ ICustomerSession erstellt. Dieses besitzt zwei Attribute, customer sowie product, die vorerst keinen konkreten Wert zugewiesen bekommen (siehe Abschnitt 4.7.1). Nach erfolgreicher Erzeugung der Kundensession wechselt die Benutzeroberfläche ins Customer-Fenster (siehe Abbildung 4.9). Hier wird durch Eingabe des Kundennamens und Drücken der Select-Taste, die Methode findCustomer(CustomerPara) ausgelöst. Intern bereitet die CrmLogik die Daten auf und sendet über den WSclient von Vtiger eine Query Anfrage bezüglich der Kundendaten, die zu dem ausgewählten Kunden gehören. Anschließend extrahiert CrmLogik die benötigten Daten und erstellt mit ihnen ein Objekt vom Typ ICrmCustomer, das im CustomerSession-Objekt als Attribut gesetzt wird. Nach erfolgreichem Durchführen dieses Prozesses bekommt die Benutzeroberfläche eine positive Rückmeldung und wechselt in den nächsten Zustand.

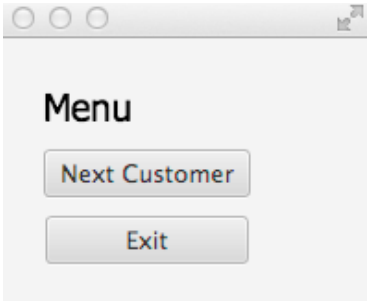

Abbildung 4.8: Menu-Fenster der Schnittstellensoftware

In diesem Fenster (siehe Abbildung 4.10) gibt der Anwender die Produktparameter ein. Diese setzen sich aus dem Produktnamen, der vom Kunden gewünschten Stückzahl sowie dem vom Kunden bevorzugten Termin zur Lieferung zusammen. Durch Drücken der Select-Taste wird die findProduct(ProductData) Methode der Fassade aufgerufen. Diese prüft zunächst, ob das Format vom Datum stimmt. Ist dieses korrekt, ruft die Fassade die findProduct(ProductData) Methode von CrmLogik auf. Dort wird nach der gleichen Struktur wie zuvor bei den Kundendaten eine Prüfung im CRM-System durchgeführt. Dabei wird geprüft, ob das Produkt in der Datenbank vorhanden ist. Genau wie bei den Kundendaten wird im Erfolgsfall ein Product-Objekt erstellt und dem Kundensession-Objekt übergeben. Nun kehrt der Programmverlauf zur Benutzeroberfläche zurück, wodurch eine weitere Option auf demselben Fenster freigeschaltet wird.

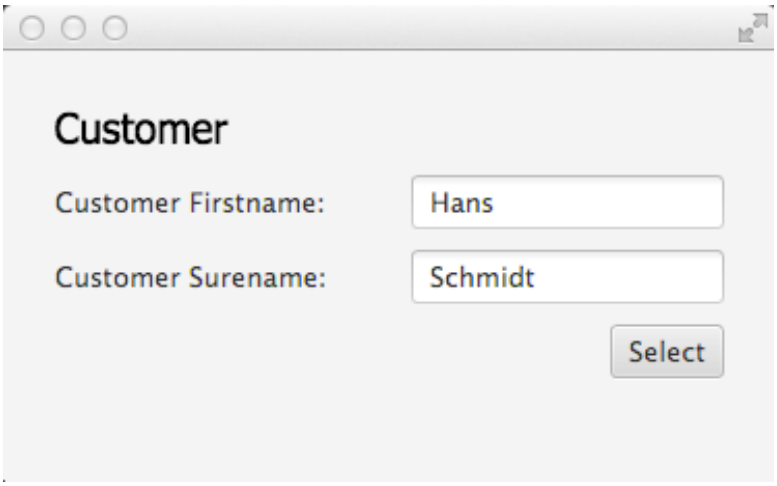

Abbildung 4.9: Kunden-Fenster der Schnittstellensoftware

Der Anwender hat zu diesem Zeitpunkt mit den bestätigten Daten die Möglichkeit eine Synchronisation mit dem ERP-System durchzuführen. Dabei wird durch Aufruf der synchronizeData() Methode in der CrmLogik das ICrmProduct Objekt aus der Kundensession gelöst und mit dem Aufruf von synchronizedata() an die CoreLogik übergeben. Dort wird zunächst überprüft, ob alle geforderten Felder belegt sind. Trifft dies zu, werden die Produktdaten wie der Produktname und die angeforderte Stückzahl in ein Transportobjekt vom Typ JSONObject umgewandelt. Der Gedanke dabei ist, dass die konkrete Implementierung und Struktur aus dem CRM-System nicht ins ERP-System übertragen werden soll. Nach der Synchronisation mit dem ERP-System liefert die ErpLogik eine Antwort, die ebenfalls vom Typ JSONObject ist. Dort sind Daten über Erfolg beziehungsweise Fehlschlag der Synchronisation mit dazugehörigen Error-Daten enthalten. Außerdem befinden sich dort die Daten mit einen möglichen geänderten Liefertermin. Diese werden in CrmLogik im Produkt-Objekt aktualisiert. Meldet der Funktionsaufruf ein Erfolgsfall, öffnet sich dem Anwender die Möglichkeit, den Kauf zu bestätigen. Darüber hinaus wird ihm der Liefertermin auf der Benutzeroberfläche angezeigt.

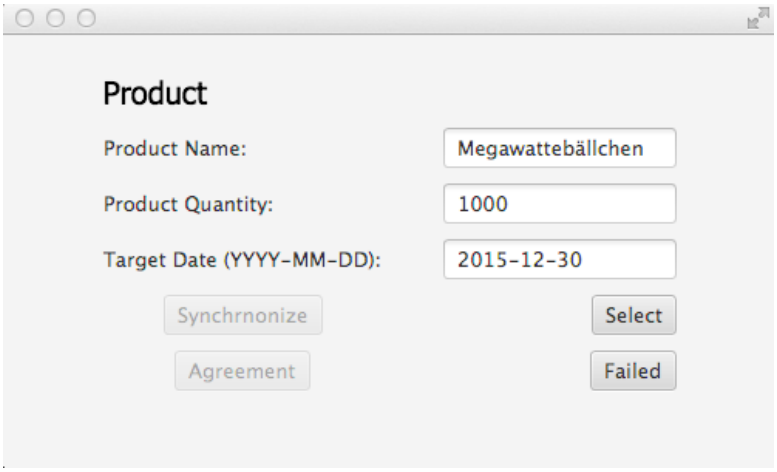

Abbildung 4.10: Produkt-Fenster der Schnittstellensoftware

Durch Drücken der Agreement-Taste wird die Fassade über die Methode updateSystems() von CrmLogik aufgerufen. Intern besteht eine Aufteilung in die Methoden updateCrmSystem() und updateErpSystem(). Bei updateCrmSystem() werden die in der Kundensession gespeicherten Produktdaten ins CRM System übertragen. Da wir von einem abgeschlossenen Kauf ausgehen, wird eine neue Menge des Produktes im CRM-System aktualisiert, die die vom Kunden erworbene Warenmenge mitberücksichtigt. Parallel dazu läuft eine Übertragung der Daten ins ERP-System. In diesem Fall wird die gesamte CustomerSession übertragen, da bei

erfolgreichem Kauf die Informationen über den Kunden ins ERP-System übertragen werden sollen. Da unser ERP-System gemockt ist, liefert die update Methode immer true zurück. Kehrt der abgeschlossene Methodenaufruf zur Benutzeroberfläche zurück, ist dieses Szenario beendet und der Anwender befindet sich zum im Menü Fenster.

4.7 Programmier Pattern

Entwurfsmuster sind nach [Eilebrecht und Starke (2014)] am besten dazu geeignet Probleme, die immer wieder beim entwerfen oder programmieren einer Anwendung auftreten, zu meistern. Sogenannte „Pattern“ stellen durch Erfahrung gewonnene Muster dar, die aus erprobten Designentscheidungen herauskristallisiert wurden. Durch sie wird ein Risiko von Endwurffehlern minimiert, die Probleme bei späteren Implementierung oder Erweiterungen einer Anwendung bereiten könnten. Sie umfassen ebenfalls Techniken zur besseren Implementierung einer Applikation.

Wir wollen nun drei im Rahmen dieser Arbeit angewendeten Programmier-Pattern vorstellen. Hierzu wird das jeweils verwendete Pattern kurz vorstellen. Im Anschluss wird anhand von Codeausschnitten unsere Umsetzung dieser Pattern näher beleuchten. Zum Schluss wird erläutert, weswegen wir uns für das jeweilige Pattern entschieden haben.

4.7.1 Nullobject-Pattern

Nach (Eilebrecht und Starke (2014)) lässt sich das Null-Object-Pattern dadurch erklären, dass eine Klasse implementiert wird, die nichts tut. Der heutige Entwickler sieht sich mit der Notwendigkeit konfrontiert, defensiv programmieren zu müssen. Das bedeutet, er muss bevor er Methoden auf einen Objekt aufruft, zuerst eine Abfrage durchführen, ob das Objekt nicht den Wert null besitzt. Wird dieser Vorgang von ihm nicht ausgeführt, riskiert er gravierende Laufzeitfehler wie NullPointerExceptions. Die Möglichkeit ein Objekt vorerst nicht zu initialisieren ist aber in vielen Fällen unverzichtbar. Das Null-Object-Pattern liefert hier die Lösung. Durch Implementierung einer Klasse, deren Methoden fachlich gesehen nichts ausführen, wird verhindert, dass man „null“ verwenden muss.

```
public class CustomerSession implements ICustomerSession {

    private ICrmCustomer customer;
    private ICrmProduct product;

    public CustomerSession(){
        customer = ICrmCustomer.NULL;
        product = ICrmProduct.NULL;
    }
```

Abbildung 4.11: Ausschnitt aus der Klasse: CustomerSession

Die Abbildung (4.11) zeigt ein Codeausschnitt unserer Klasse „CustomerSession“. Eine Instanz dieser Klasse wird erzeugt sobald der Callcenter-Agent die „Next Customer“ Taste drückt. Zu jenem Zeitpunkt existieren die Attribute „product“ vom Typ ICrmProduct und „customer“ vom Typ ICrmCustomer noch nicht, da diese erst im Verlauf des weiteren Programms initialisiert werden.

```
public interface ICrmCustomer {

    default boolean isNil(){ return false;}

    public static final ICrmCustomer NULL = new ICrmCustomer() {
        @Override
        public boolean isNil() { return true; }

        @Override
        public boolean isValid() { return false; }

        @Override
        public String getFirstName() { return "EMPTYNAME"; }

        @Override
        public String getSureName() { return "EMPTYNAME"; }
```

Abbildung 4.12: Ausschnitt aus Interface: ICrmCustomer

In der Abbildung (4.12) wird gezeigt, wie an das Problem herangegangen ist. Um eine typsichere Null-Variante von ICrmCustomer zu haben, wird im Interface ICrmCustomer eine anonyme Innere Klasse implementiert, diese existiert nur einmal und kann ausschließlich von

Interface erzeugt werden. Darüber hinaus kann ihr Wert nicht verändert werden. Java 8 erlaubt Default-Methoden zu definieren. Deswegen kann eine isNil() Methode definieren werden, die alle Klassen, die dieses Interface implementieren, automatisch besitzen und die den Wert „false“ liefert. In der anonymen inneren Klasse wird diese Methode überschreiben, so dass sie den Wert “true“ liefert. Auf diese Weise sind isNil() Abfragen, falls gewollt, nach wie vor möglich. Des weiteren überschreiben wir die anderen deklarierten Methoden und lassen sie Rückgabewerte liefern, die für ein NullObject fachlich Sinn machen. So liefert isValid() natürlich den Wert „false“ und Methoden die ein String Getter darstellen liefern „EMPTYSTRING“.

4.7.2 Factory-Pattern

Da die Schnittstellensoftware möglichst unabhängig von den Verwendeten ERP oder CRM-Sytemen gestalten werden sollte, hat man sich für die Verwendung vom Factory-Pattern entschieden. Erhält man gemäß [Eilebrecht und Starke (2014)] dadurch eine Menge in sich geschlossener Klassenverbände. Dadurch lassen sich konkrete Klassen vom Rest des Codes abschirmen. Die konkreten Klassen sollen erst später festgelegt oder austauschbar sein. Darüber hinaus soll der Konstruktionsprozess von der Repräsentation getrennt werden.

Aus diesen Vorgaben heraus entschieden wir uns für ein spezielles Design unserer Anwendung. Die einzelnen Komponenten besitzen alle eine Implementierung einer Erzeuger-Klasse, die die Endung -Mgmt besitzen. Als Beispiel hierfür dienen CrmMgmt oder CoreMgmt. Diese implementieren die Interfaces ICrmMgmt und ICoreMgmt, die von der Komponente Initialisierungsmanagement zugegriffen wird. Initialisierungsmanagement entscheidet in der Main-Methode anhand der ihr vorliegenden Parameter, welche konkrete Implementierung der Logik der einzelnen Komponenten durchgeführt wird und an welche Komponenten diese in der entsprechenden Reihenfolge übergeben werden (vgl 4.3). Hierzu wird im Fall von Erzeugung der Logik-Implementierungen ein Enum vom Typ CRM herangezogen. In diesem sind die jeweils verwendeten möglichen CRM-Systeme als Werte definiert.

In der Abbildung (4.13) wird die Factory-Methode getCrmLogic() aus der Factory-Klasse CRMMgmt vorgestellt. Anhand des übergebenen Parameters crm vom EnumTyp CRM wird entschieden, welche Konkrete Logik Klasse der CRM Komponente erzeugt wird. Der andere Parameter wird benötigt, weil die Implementierung von ICrmLogic eine Referenz auf die Logik aus dem Kernmanagement benötigt. Auf diese Weise erhält man eine Trennung der für die Erzeugung beauftragten Klassen und derer, die für die eigentliche Programmlogik zuständig sind. Dadurch halten wir unser Programm anpassungsfähig. Außerdem ist der Vorgang der In-

```
@Override
public ICRMLogic getCrmLogic(ICoreLogic coreLogic, CRM crm) {
    switch(crm) {
        case VTIGER:
            this.logic = VtigerLogic.getVtigerLogic(coreLogic);
            break;
        default:
            this.logic = VtigerLogic.getVtigerLogic(coreLogic);
            break;
    }
    return logic;
}
```

Abbildung 4.13: Factory-Methode aus CRMMgmt

itialisierung von den anderen Komponenten entkoppelt und dadurch überschaubar. Hierbei sei erwähnt, dass durch die Abstrahierung der Erzeuger-Klassen durch Interfaces eine Anpassung auf dieser Ebene ebenfalls möglich ist.

4.7.3 Singleton-Pattern

Den Angaben von [Eilebrecht und Starke (2014)] geht das Factory-Patern mit dem Singleton-Pattern Hand in Hand. Grund hierfür ist, dass man bei Verwendung des Factory-Pattern, größtenteils nur eine einzelne Erzeugerklasse haben will. Das Singleton-Pattern wird eingesetzt, wenn man von einer Klasse genau eine Instanz haben möchte und keine weiteren erzeugt werden sollen. Dieses Pattern ist einfach zu implementieren, was ein nicht zu unterschätzender Vorteil ist. Die Nachteile von Singleton treten laut [Eilebrecht und Starke (2014)] vor allem bei Mutli-Thread Anwendung auf, wobei es bei zu vielen Benutzern zu Performance Problemen kommen kann. Darüber hinaus der Zerstörung eines Singleton-Objektes, wobei Referenzen in den anderen Clients auf dieses Verbleiben könnten.

Diese Probleme können bei unserer Anwendung nicht auftreten, weshalb es keinen Grund gibt, auf das Pattern zu verzichten. Aufgeführt in Abbildung (4.14) sieht man unsere Umsetzung anhand der oben bereits erwähnten Erzeuger-Klasse CRMMgmt. Die Klasse besitzt mit der Klassen-Variable „instance“ eine Referenz auf die einzige Instanz von sich selbst. Der Konstruktor ist private und kann über die static Methode getCRMMgmt() aufgerufen werden. Dort findet eine Prüfung statt, ob „instance“ bereits initialisiert wurde (vgl. 4.7.1) um dann entsprechend eine neue Instanz zu erzeugen oder die bereits vorhandene zurückzugeben.

```
public class CRMMgmt implements ICRMMgmt {
    private static ICRMMgmt instance = ICRMMgmt.NULL;
    private ICRMLogic logic;

    public static ICRMMgmt getCRMMgmt(){
        if(instance.isNil()){
            instance = new CRMMgmt();
        }
        return instance;
    }

    private CRMMgmt(){

    }
```

Abbildung 4.14: Ausschnitt aus CRMMgmt

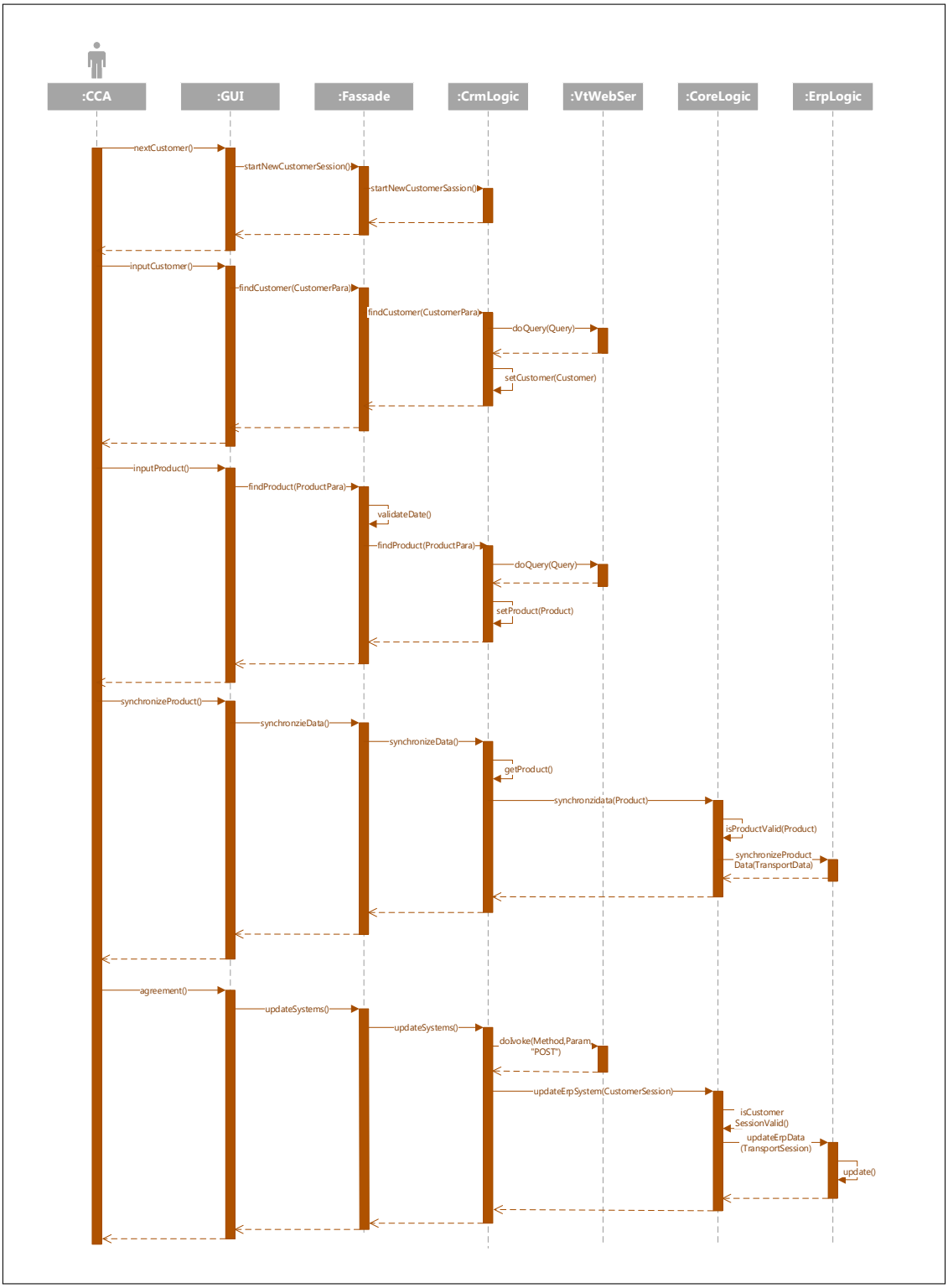

Abbildung 4.15: Sequenzdiagramm während der Kundenbetreuung

5 Fazit

Anwendungsintegration umfasst ein breites Themengebiet, dass durch die Unterstützung von Geschäftsprozessen in Unternehmen geprägt wurde. Im Rahmen dieser Arbeit haben wir beobachtet, wie ERP-Systeme im Laufe der Zeit aus den vereinzelten Anwendungen zur Unterstützung von Prozessen innerhalb eines Unternehmens, entstanden sind. Der Drang nach Vereinigung der einzelnen Komponenten zu einer Anwendungslandschaft, führt zu immer komplexeren Softwaregebilden, die ein breites Spektrum von Funktionalität in sich kombinieren. Eine hierfür treibende Kraft ist der Versuch, Abläufe im Unternehmen auf die Weise zu optimieren, dass Kosten eingespart werden können. Gemäß diesem Eifer, ist eine Verschmelzung von Anwendungen innerhalb einer IT-Landschaft erstrebenswert. Durch das Zusammenführen einzelner Systeme mit sich überschneidenden Aufgabenbereichen und Datensätzen, können aufgrund von Personaleinsparung und durch Reduzierung von Wartungsarbeiten, letztendlich Kosten eingespart werden.

Trotz dieser Entwicklung besteht nach wie vor der Bedarf getrennte Systeme miteinander zu koppeln. Durch Fusionen oder Übernahmen von Unternehmen entsteht die Notwendigkeit separate Systeme miteinander in Einklang zu bringen, die nicht zu diesem Zweck entwickelt wurden. Resultierend aus diesem Bestreben, wurde in der Arbeit eine Anwendung zum Austausch von Daten zwischen ERP- und CRM-Systemen entwickelt. Um die Aufgaben dieser Software abzugrenzen, wurde ein Geschäftsprozess betrachtet. In der durchgeführten Analyse wurde der besagte Geschäftsprozess als EPK modellierten. Anschließend wurden die Schlüsse für die Umsetzung der Anwendung gezogen. Diese erweitert den Funktionsumfang von der CRM-Software Vtiger um die notwendigsten Merkmale. Zu den größten Vorteilen dieser Umsetzung gehört die Kompaktheit der Anwendung. Darüber hinaus ist die einfache Benutzung der Software positiv hervorzuheben, da sie den Benutzer beim Vorgang an die Hand nimmt. Auf diese Weise können Kosten für die Schulung des Personals gering gehalten werden.

Nach der Testphase sind einige wenige Verbesserungsmöglichkeiten der Schnittstellensoftware aufgefallen. Dazu gehört vor allem die Möglichkeit, den aus zwei Stufen bestehen Login-Vorgang durch eine Stufe der Anmeldung zu ersetzen. Darüber hinaus könnten die

Daten zwischen den beiden Systemen synchronisiert werden, ohne dass der Callcenter-Agent aktiv am Vorgang beteiligt wird.

An diese Forschung anschließend, wäre eine Erweiterung der Systemlandschaft möglich. Wir sind in unserer Arbeit darauf eingegangen, dass eine Punk-zu-Punkt Topologie bei erhöhter Anwendungszahl an ihre Grenzen stößt. Das ERP-System könnte eine zweite Schnittstelle zu einem weiteren System, wie dem Rechnungswesen, erhalten. Diese könnte eine Migration der Daten auf einer anderen Ebene durchführen, wodurch neue Aspekte hinsichtlich der Kooperation unterschiedlicher Implementierungsmöglichkeiten erforscht werden könnten. Anschließend wäre eine Auswertung des Zusammenspiels aller drei Systeme erdenklich. Die daraus gewonnenen Einsichten könnten zu einer neuen Betrachtungsweise der Gesamtproblematik führen.

Literaturverzeichnis

[GablerCRM] : *Customer Relationship Management (CRM)*. – URL http://wirtschaftslexikon.gabler.de/Archiv/5072/customer-relationship-management-crm-v10.html. – Zugriffsdatum: 2015-12-11

[Itransparent] : *Enterprise Application Integration*. – URL http://www.itransparent.de/kompetenzen/enterprise-service-bus/enterprise-application-integration. – Zugriffsdatum: 2015-12-09

[GablerWL] : *Enterprise-Ressource-Planning-System*. – URL http://wirtschaftslexikon.gabler.de/Definition/enterprise-resource-planning-system.html?referenceKeywordName=ERP-System. – Zugriffsdatum: 2015-12-04

[ArndtWI] : *Grundlagen und Konzepte des EAI*. – URL https://www.wi1.uni-muenster.de/pi/lehre/ss06/eai/folien/eai_servlets/Vorlesung_EAI.pdf. – Zugriffsdatum: 2015-12-04

[Fischer2007] : *Ilmenauer Integrationsmodell für Informationssysteme*. – URL http://www.db-thueringen.de/servlets/DerivateServlet/Derivate-10606/IBzWI_2007-01.pdf. – Zugriffsdatum: 2015-12-04

[Hahn2012] : *Integration von Informationssstemen*. – URL http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/daten-wissen/Informationsmanagement/Informationsmanagement--Aufgaben-des/Informationssystem--Integration
– Zugriffsdatum: 2015-12-04

[LeadsVT] : *Leads Module*. – URL https://wiki.vtiger.com/index.php/Leads_Module. – Zugriffsdatum: 2015-12-03

[Wikipedia] : *Webservice.* – URL https://de.wikipedia.org/wiki/Webservice. – Zugriffsdatum: 2015-12-03

[Microsoft2013 2013] : *Einführung in BizTalk Server.* 2013. – URL https://technet.microsoft.com/de-de/library/aa547058.aspx. – Zugriffsdatum: 2015-12-15

[Becker u. a. 2012] Becker, Jörg ; Kugeler, Martin ; Rosemann, Michael (. ; Becker, Jörg ; Kugeler, Martin ; Rosemann, Michael: *Prozessmanagement - Ein Leitfaden zur prozessorientierten Organisationsgestaltung.* 7. Aufl. Berlin Heidelberg New York : Springer-Verlag, 2012. – ISBN 978-3-642-33844-1

[Bunjes u. a. 2002] Bunjes, Bernd ; Friebe, Jörg ; Götze, Rainer ; Harren, Arne: Integration von Daten, Anwendungen und Prozessen am Beispiel des Telekommunikationsunternehmens EWE TEL. In: *Wirtschaftsinformatik* 44 (2002), Nr. 5, S. 415–424. – URL http://dx.doi.org/10.1007/BF03250863. – ISSN 0937-6429

[Dazer] Dazer, Michael: RESTful APIs-Eine Übersicht.

[Eilebrecht und Starke 2014] Eilebrecht, Karl ; Starke, Gernot: *Patterns kompakt - Entwurfsmuster für effektive Software-Entwicklung.* 4. Aufl. Berlin Heidelberg New York : Springer-Verlag, 2014. – ISBN 978-3-642-34718-4

[Fischer und Stelzer 2008] Fischer, Daniel ; Stelzer, Prof. Dr. D.: *Unternehmensübergreifende Integration von Informationssystemen - Bestimmung des Integrationsgrades auf elektronischen Marktplätzen.* 2009. Aufl. Berlin Heidelberg New York : Springer-Verlag, 2008. – ISBN 978-3-834-91285-5

[Gronau 2004] Gronau, Norbert: *Enterprise resource planning und Supply-chain-Management - Architektur und Funktionen.* München : Oldenbourg Verlag, 2004. – ISBN 978-3-486-27265-9

[Gronau 2012] Gronau, Norbert: *Handbuch der ERP-Auswahl* -. Berlin : GITO mbH Verlag, 2012. – ISBN 978-3-942-18337-6

[Helmke u. a. 2012] Helmke, Stefan ; Uebel, Matthias ; Dangelmaier, Wilhelm: *Effektives Customer Relationship Management - Instrumente - Einführungskonzepte - Organisation.* 5. Aufl. Berlin Heidelberg New York : Springer-Verlag, 2012. – ISBN 978-3-834-94176-3

[Hippner u. a. 2011] Hippner, Hajo ; Hubrich, Beate ; Wilde, Klaus D. (. ; Hippner, Hajo ; Hippner, Hajo ; Hubrich, Beate ; Hubrich, Beate ; Wilde, Klaus-Dieter ; Arndt, Dirk ; Becker, Jörg ; Binder, Jochen ; Boenigk, Silke ; Bruhn, Manfred ; Diller, Hermann ; Gary, Alexander ; Georgi, Dominik ; Götz, Oliver ; Gouthier, Matthias ; Grieser, Lukas ; Gründling, Christian ; Günter, Bernd ; Haas, Alexander ; Hanke, Robert ; Hansen, Ursula ; Helm, Sabrina ; Hesse, Frank ; Jaeck, Horst-Florian ; Knackstedt, Ralf ; Krafft, Manfred ; Leusser, Wolfgang ; Link, Jörg ; Merzenich, Melanie ; Mink, Moritz ; Münster, Jan ; Rentzmann, René ; Rühl, Denise ; Sauer, Achim ; Schmidt, Inga ; Schögel, Marcus ; Schöler, Andreas ; Siegl, Marcus ; Stauss, Bernd ; Terpin, Jürgen ; Wilde, Klaus D.: *Grundlagen des CRM - Strategie, Geschäftsprozesse und IT-Unterstützung*. 3. Aufl. Berlin Heidelberg New York : Springer-Verlag, 2011. – ISBN 978-3-834-96618-6

[Jacob u. a. 2013] Jacob, Herbert ; Becker, Jörg ; Krcmar, Helmut: *Integrierte Informationssysteme* -. Berlin Heidelberg New York : Springer-Verlag, 2013. – ISBN 978-3-322-84583-2

[Jakl 2008] Jakl, Michael: REST Representational State Transfer. (2008)

[Keller 2002] Keller, Wolfgang: Enterprise Application Integration. In: *Erfahrungen aus der Praxis. dpunkt* (2002)

[Koch 2015] Koch, Koch: *Einführung in das Management von Geschäftsprozessen - Six Sigma, Kaizen und TQM*. 2. Aufl. Berlin Heidelberg New York : Springer-Verlag, 2015. – ISBN 978-3-662-44450-4

[Kohnke u. a. 2008] Kohnke, Oliver ; Scheffler, Dipl-Wirtsch-Ing T. ; Hock, Dipl Wirtsch-Inf C.: SOA-Governance–Ein Ansatz zum Management serviceorientierter Architekturen. In: *Wirtschaftsinformatik* 50 (2008), Nr. 5, S. 408–412

[Leimeister 2015] Leimeister, Leimeister: *Einführung in die Wirtschaftsinformatik* -. 12. Aufl. Berlin Heidelberg New York : Springer-Verlag, 2015. – ISBN 978-3-540-77847-9

[Liebhart] Liebhart, Daniel: *Schnittstellen sind ein Alptraum*. – URL http://www.trivadis.com/sites/default/files/downloads/pr/Netzwoche_Schnittstellen_DAL_091111.pdf. – Zugriffsdatum: 2015-06-10

[Mantel und Schissler 2002] Mantel, Stephan ; Schissler, Martin: Application Integration. In: *Wirtschaftsinformatik* 44 (2002), Nr. 2, S. 171–174

[Melzer 2010] Melzer, Ingo: *Service-orientierte Architekturen mit Web Services - Konzepte - Standards - Praxis.* 4. Aufl. Berlin Heidelberg New York : Springer-Verlag, 2010. – ISBN 978-3-827-42550-8

[Oesterle u. a. 2003] Oesterle, Hubert ; Alt, Reiner ; Heutsch, Rogeri: WebServices - Hype oder Lösung. In: *io new management* (2003)

[Rausch 2004] Rausch, Till: Service Orientierte Architektur - Übersicht und Einordnung. (2004)

[Rewissen] Rewissen: *EPK-Modellierung.* – URL http://www.re-wissen.de/opencms/Wissen/Techniken/EPK-Modellierung.html. – Zugriffsdatum: 2015-06-10

[Schacher und Grässle 2006] Schacher, Markus ; Grässle, Patrick: *Agile Unternehmen durch Business Rules - Der Business Rules Ansatz.* 2006. Aufl. Berlin Heidelberg : Springer Science Business Media, 2006. – ISBN 978-3-540-25676-2

[Schawel und Billing 2014] Schawel, Christian ; Billing, Fabian: *Top 100 Management Tools - Das wichtigste Buch eines Managers Von ABC-Analyse bis Zielvereinbarung.* 5. Aufl. Berlin Heidelberg New York : Springer-Verlag, 2014. – ISBN 978-3-834-94691-1

[Stannat und Petri 2004] Stannat, Annette ; Petri, Christian: Trends in der Unternehmens-IT. In: *Informatik-Spektrum* 27 (2004), Nr. 3, S. 227–237

[Starke und Hruschka 2011] Starke, Gernot ; Hruschka, Peter: *Software-Architektur kompakt - - angemessen und zielorientiert.* 1. Aufl. 2009. Berlin Heidelberg : Springer Science Business Media, 2011. – ISBN 978-3-827-42093-0

[Strüver 2006] Strüver, Sven C.: *Standardbasiertes EAI-Vorgehen am Beispiel des Investment Bankings* -. 1. Aufl. Berlin : GITO mbH Verlag, 2006. – ISBN 978-3-936-77181-7

[T. Davenport] T. Davenport, J. S.: *THE NEW INDUSTRIAL ENGINEERING: INFORMATION TECHNOLOGY AND BUSINESS PROCESS REDESIGN.* – URL http://dspace.mit.edu/bitstream/handle/1721.1/48613/newindustrialeng00dave.pdf?sequence=1. – Zugriffsdatum: 2015-12-09

[Vajna 2014] Vajna, Sándor: *Integrated Design Engineering - Ein interdisziplinäres Modell für die ganzheitliche Produktentwicklung.* 1. Aufl. Berlin Heidelberg New York : Springer-Verlag, 2014. – ISBN 978-3-642-41104-5

[Zeppenfeld u. a. 2009] Zeppenfeld, Klaus ; Finger, Patrick ; Finger, Patrick: *SOA und WebServices* -. 1. Aufl. Berlin Heidelberg New York : Springer-Verlag, 2009. – ISBN 978-3-540-76991-0

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 19. Dezember 2015 Grzegorz Markiewicz