Westsächsische Hochschule Zwickau
University of Applied Sciences
HOCHSCHULE FÜR MOBILITÄT | UNIVERSITY FOR MOBILITY

Fakultät für Physikalische Technik / Informatik

Fachgruppe Informatik

# Masterarbeit

## Migration und Erweiterung eines Systems zur Unterstützung der Warenauslieferung

Florian Keßler

Chemnitz, den 05.07.2022

**Matrikelnummer:** 41197
**E-Mail-Adresse:** florian.kessler.kah@fh-zwickau.de

**Prüfer 1:** Prof. R. Laue
**Prüfer 2:** Prof. W. Golubski

# Firma erfal GmbH & Co. KG

Die Firma erfal GmbH & Co. KG ist seit mehr als 30 Jahren als Hersteller von Dekoration, Sonnen- und Insektenschutz im Maßbereich nach Kundenwunsch tätig. Diese wird im weiteren Verlauf der Arbeit als erfal bezeichnet. Das Unternehmen besitzt einen Produktions- und Verwaltungsstandort mit circa 500 Mitarbeitern, welcher sich in Falkenstein im Vogtland im Gewerbering 8 befindet. Geografisch erstrecken sich die Produktions- und Verwaltungsbereiche auf insgesamt fünf Gebäude, ein sechstes befindet sich bereits im Bau und ist als zentrales Logistik- und Warenlager geplant.

Erfal vertreibt die Produkte ausschließlich an Fachhändler, welche sowohl über den Webshop als auch per Telefon, Fax oder E-Mail bestellen können. Nach der Fertigstellung der Produkte müssen diese ausgeliefert werden. Dabei können diese zwei unterschiedlich Varianten beliefert werden:

1. Auslieferung über einen Speditionsdienstleister wie z.B. DHL, Hermes, TNT, UPS etc.
2. Der Kunde wird mit dem internen Fuhrpark beliefert.

Der Fuhrpark von erfal setzt sich aus drei LKWs à 12t mit Anhänger, 45 Transportern und 12 Anhängern zusammen, welche insgesamt eine jährliche Kilometerlaufleistung von über drei Millionen Kilometern aufweisen. Die Auslieferung über den internen Fuhrpark erfolgt sowohl deutschlandweit als auch darüber hinaus. Darunter fallen z.B. die Schweiz, Österreich, Luxemburg und die Niederlande.

# Inhaltsverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| API | Application Programming Interface |
| BPMN | Business Process Model and Notation |
| DBMS | Datenbankmanagementsystem |
| DMS | Dokumentenmanagement-System |
| EAL | Elektronische Auslieferung |
| ERD | Entity-Relationship-Diagramm |
| ePS | erfal Produktionssystem |
| ERM | Entity-Relationship-Modell |
| ERP | Enterprise Resource Planning |
| FTP | File Transfer Protocol |
| GB | Gigabyte |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure |
| MES | Manufacturing Execution System |
| PDF | Portable Document Format |
| PHP | Hypertext Preprocessor |
| XML | Extensible Markup Language |

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Anhangsverzeichnis

# 1 Einleitung

## 1.1 Motivation

Durch die steigenden Agilitätsanforderungen an eine Software sind die Innovationszyklen bzw. Veröffentlichungen von Updates in immer kürzerer Zeit zu erbringen. Deshalb sind das kontinuierliche Erweitern sowie das Verbessern einer Software ein Teil zur langfristigen Etablierung am Markt.

„Der Code wird im Laufe der Zeit modifiziert, und die Integrität des Systems – die dem ursprünglichen Design entsprechende Struktur – geht allmählich verloren. Der Code wird nicht mehr sorgfältig entwickelt, und die Programmierung wird allmählich zum <<Hacken>>." [1, S. 20] Dies hat zur Folge, dass die Systeme (teils unbeabsichtigt) unübersichtlich und komplex in ihrer Weiterentwicklung und Wartung werden. Technische Schulden sind die Konsequenzen einer schlechten Software-Architektur, welche aus den o.g. Gründen entstehen kann. [1, S. 20] Nach Fowler kann hierbei zwischen vier verschiedenen Arten unterschieden werden:

| | **rücksichtlos** | **umsichtig** |
|---|---|---|
| **vorsätzlich** | "Wir haben keine Zeit für Design" | "Wir müssen liefern und mit den Konsequenzen leben" |
| **versehentlich** | "Was ist eine Schichtenarchitektur?" | "Jetzt wissen wir, wie wir es hätten tun sollen" |

**Tabelle 1 Arten von technischen Schulden [2]**

Technische Schulden beschleunigen das Projekt auf kurze Sicht, jedoch hindern diese technischen Schulden langfristig daran, eine weniger komplexe und wartbare Software zu entwickeln. Desto länger eine bestimmte technische Schuld besteht, umso schwieriger bzw. kostenintensiver ist die Behebung dieser. Das Verhältnis der Zeit des Bestehens einer technischen Schuld und der Aufwand diese zu bereinigen (monetär bzw. nicht monetär) weist exponentielles Verhalten auf. [2]

Um dies zu beheben, kann das Design einer bestehenden Anwendung mittels Refactoring verbessert und bei Bedarf auf ein aktuelles System migriert werden.

## 1.2 Aufbau der Arbeit

Das Ziel dieser Arbeit ist es, eine Strategie für die Migration und Erweiterung eines Altsystems zur Warenauslieferung zu entwickeln. Dieses ist in das bestehende Produktionsleitsystem der Firma erfal zu integrieren. Dabei erfolgt zunächst die

grundlegende Betrachtung des aktuellen Istzustandes. Ebenso werden hierbei die verschiedenen Probleme des alten Systems näher beleuchtet.
Im dritten Kapitel stehen grundlegende Migrations-Strategien und Modelle im Mittelpunkt. Dabei werden ausgewählte Strategien und Modelle beschrieben.
Im vierten Kapitel werden mögliche Refactoring-Methoden und deren Nutzen erläutert. Anschließend werden die Strategien und Modelle auf die Tauglichkeit hinsichtlich der Problemstellung betrachtet und geeignete ausgewählt.
Das fünfte Kapitel behandelt die Anforderungsanalyse in Bezug auf die neue Software. Hierbei werden verschiedene Methoden zur Gewinnung von Anforderungen genutzt, welche zuvor grundlegend erläutert werden.
In den nächsten drei Kapiteln wird die Migration und Erweiterung des neuen Systems zur Unterstützung der Warenauslieferung beschrieben. Ebenso werden grundlegende Konzeptentwicklungen, Datenhaltung und Datenbeschaffung mitbetrachtet. Ein weiterer Bestandteil sind die Herausforderungen und Fehler, die in der Migration und Erweiterung des Warenauslieferungssystems auftreten können.
Das letzte Kapitel beinhaltet die Auswertung der Ergebnisse, sowie einen Ausblick auf die zukünftige Nutzung der Software.

# 2 Problemstellung

## 2.1 Beschreibung des Ist-Zustandes

Ein Kunde gibt bei erfal seine Bestellung über den Webshop, Fax, Telefon oder E-Mail auf. Nach einer kundenspezifischen Auftragsbestätigungsfrist wird der Auftrag in die Produktion geschickt. Die Auftragsbestätigungsfrist bzw. AB-Zeit dient dazu, um eventuelle Änderungen an dem Auftrag durchzuführen wie z.B. bei fehlerhafter Maßangabe etc. Abhängig davon, welche Produktkategorie(n) der Kunde bestellt hat, wird dieser auf die jeweilige(n) Abteilung(en) (wie z.B. Plissee oder Insektenschutz) aufgeteilt. Sobald die Ware fertig produziert wurde, wird diese mittels internen Transportwagen zu der Verladung oder dem Versand gebracht. Dies ist abhängig davon, wie der Kunde beliefert wird. Es existieren zwei verschiedene Belieferungsmöglichkeiten. Die erste Variante ist die Auslieferung über einen Speditionsdienstleister. Diese sind in Bezug auf die Paketgröße limitiert bzw. eingeschränkt. Somit ergeben sich bei dem Transport von größeren Produkten erhöhte Kosten oder der Transport ist aufgrund der Abmaße der Pakete schlichtweg nicht möglich. Aufgrund dessen gibt es als Alternative bei erfal einen internen Fuhrpark, welcher die Ware zu den jeweiligen Kunden transportiert.

Welche dieser beiden Varianten genutzt wird, ist von mehreren Faktoren abhängig. Grundsätzlich ist im Kundenprofil die bevorzugte Belieferungsart hinterlegt. Von dieser kann aber abgewichen werden, wenn zum Beispiel die Auslastung der Tour zu gering ist und sich somit wirtschaftlich nicht rechnet. Dagegen kann es ebenfalls passieren, dass aufgrund von krankheitsbedingten Ausfällen die Produkte per Spedition verschickt werden müssen.

Zur Unterstützung der Fahrer gibt es das elektronische Auslieferungssystem, welches seit 2016 im Einsatz ist. Durch dieses benötigen die Fahrer keine Ladelisten in Papierform mehr und können auf einen Blick erkennen, welche Ware in welchem Fach für einen Kunden vorgesehen ist. Außerdem ermöglicht die geokoordinierte Adresse, den Fahrer über ein Online-Kartensystem direkt zum Ziel zu navigieren. Zusätzlich dazu ist bei der Zustellung ein Scannen der Pakete vorgesehen, womit in der Theorie kein Paket auf dem Auto vergessen werden kann. Sobald der Barcode der Pakete gescannt wird, wird der aktuelle Standort des mobilen Endgerätes erfasst und als Zustellort vermerkt. Eine Bestätigung der Zustellung bei dem Kunden kann durch die Erfassung einer Unterschrift erfolgen, welche ebenfalls auf dem mobilen Endgerät möglich ist.

## 2.2 Auslieferungsprozess

Der Auslieferungsprozess von Waren über den Fuhrpark von erfal ist nachfolgend in einem BPMN Diagramm dargestellt.

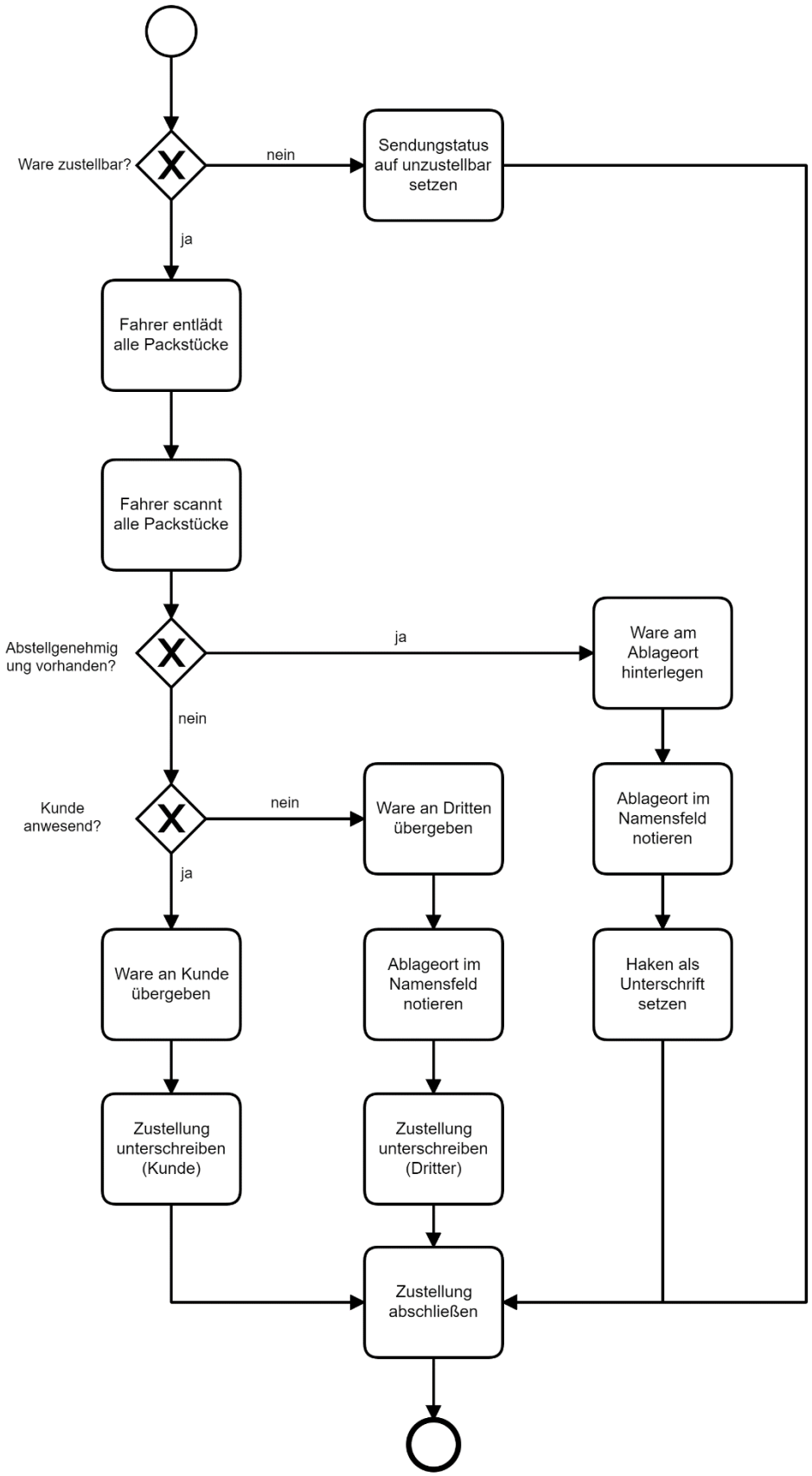



**Abbildung 1 BPMN-Diagramm Zustellprozess**

Wie eingangs erwähnt, existiert ein Mechanismus, der verhindern soll, dass Pakete im Auto vergessen werden. Durch das Scannen der einzelnen Pakete wird sichergestellt, dass alle verfügbaren Waren zugestellt werden. Allerdings stellt dieser Mechanismus die Fahrer vor eine Herausforderung. Angenommen, ein Fahrer muss an den Kunden X 15 Pakete in Berlin ausliefern, parkt in zweiter Reihe und steht somit unter erhöhtem Zeitdruck. Durch die Vielzahl an Paketen ist das Scannen dieser zeitintensiv. Deshalb umgehen die Fahrer dies, indem alle Pakete markiert werden und der Status auf nicht zustellbar gesetzt wird. Damit entfällt ein Scannen der einzelnen Pakete. Anschließend setzt er den Status der gesamten Sendung auf zugestellt und fährt weiter. Dies erschwert die Nachforschungen jeglicher Sendungen, aber insbesondere für verlorengegangene bzw. nicht zugestellte Sendungen, da der generierte Sendungsbericht widersprüchlich ist. Jedes einzelne Paket steht auf nicht zugestellt, allerdings steht der Status der gesamten Sendung auf zugestellt. Dies stellt das Qualitätsmanagement von erfal ebenfalls vor eine große Herausforderung. Dieses kann bei Nichtauffinden der Ware nicht zweifelsfrei nachweisen, dass die Lieferung korrekt zugestellt worden ist. Dadurch ergeben sich Mehrkosten, wie z.B. das erneute Produzieren der Ware, welche das Unternehmen zu tragen hat. Der aus den übertragenen Daten generierte Sendungsbericht kann dem Kunden nicht ausgehändigt werden, da dieser Bericht gegensätzliche Aussagen beinhaltet. Die gesamte Sendung hat den Status nicht zugestellt und als Grund steht Ware Kunde X übergeben. Eine Manipulation dieser Daten ist ebenfalls nicht zulässig. Der prinzipielle Ablauf der Zustellung auf den mobilen Endgeräten liegt nicht im Anpassungsbereich der Firma erfal, sondern wird von einem externen Dienstleister übernommen. Nachfolgend wird das System zur elektronischen Auslieferung näher beschrieben.

## 2.3 Schnittstellenbeschreibung

Das elektronische Auslieferungssystem ist grundlegend in zwei Systeme aufgeteilt. Die Software bzw. App, welche auf den Mobilen Endgeräten (MDE) ausgeführt wird, stammt von einer externen Firma mit dem Namen GHT. Diese stellt sowohl die App als auch die Infrastruktur bereit, welche über das Internet erreichbar ist.

Diese Cloud-Anwendung ist über eine Schnittstelle bei erfal angebunden. Durch diese werden die notwendigen Informationen zu einer Sendung wie z.B. Paketnummer, Empfänger-Adresse und Tour an das GHT-Portal übertragen. Eine Sendung kann mehrere Aufträge mit mehreren Packstücken enthalten, welche an einen Kunden geliefert werden sollen. Zusätzlich dazu importiert die Schnittstelle die bereitgestellten

Daten von Zustellinformationen des GHT-Portals, wie z.B. den Status der Sendung, bei Zustellung der Sendung ein Bild der Unterschrift und den aktuellen Standort.
Die Daten werden im XML-Format ausgetauscht. Nachfolgend sind diese Schnittstellen näher erläutert und hinsichtlich ihrer Funktionalität unterteilt. Im weiteren Verlauf der Arbeit wird das GHT-Portal synonym mit den Begriffen GHT, Portal, Cloud-Anwendung verwendet. Des Weiteren ist die Bezeichnung der Schnittstellen im Legacy-System aus der Sicht der externen Firma beschrieben. Das bedeutet konkret: die Schnittstelle, welche Daten von erfal zu GHT überträgt, wird als Importschnittstelle bezeichnet. Analog dazu wird die Schnittstelle, welche Daten von dem GHT-Portal zu erfal transferiert, als Exportschnittstelle bezeichnet.

**Datenübertragung von erfal zu GHT**

Während des Verpackungsprozesses der Produkte bei erfal werden Packstücke gebildet. Dies ist die Grundlage für die Übertragung der Daten zu GHT, denn es können nur Produkte ausgeliefert werden, für die auch ein Packstück existiert. Diese Daten werden 18:30 Uhr für den nächsten Tag erzeugt und an das Portal übertragen. Anschließend erfolgt bis 22 Uhr jede halbe Stunde eine erneute Übertragung der Daten, um Änderungen zu übergeben. Es werden unter anderem folgende Daten an das Cloud-Portal übermittelt:

1. Adresse (Land, Straße, Hausnummer, Postleitzahl und Ort) des Senders und Empfängers
2. Paketnummer
3. Paketanzahl
4. Fachnummer / Lagerplatz
5. Paketabmessungen

Nachdem die Daten übertragen worden sind, werden diese in zweifach redundanter Form abgespeichert. Die gesendete Datei wird auf dem Server abgelegt und in unregelmäßigen Abständen per Hand komprimiert und archiviert. Zusätzlich dazu werden die übermittelten Informationen in einer Datenbank persistiert.

**Datenübertragung von GHT zu erfal**

Während des Zustellprozesses ist das MDE über eine mobile Datenverbindung mit dem GHT-Portal verbunden. Bei Veränderung des Zustellstatus werden die geänderten Informationen an das Portal übertragen. Dieses stellt anschließend eine XML-Datei mit u.a. folgenden Informationen zur Verfügung:

1. Zustellstatus
2. Paketnummer
3. Bei Zustellung die Adresse inkl. Geokoordinaten
4. Bei Zustellung ein Bild der Unterschrift

Diese XML-Datei wird anschließend in einer Datenbank abgespeichert und separat in einem Ordner als Datei abgelegt. Zudem erfolgt eine azyklische sowie händische Komprimierung und Archivierung der Daten. Zusätzlich dazu wird ein Sendungsbericht generiert und zentral zur Verfügung gestellt. Detailliertere Informationen zu der Datenhaltung des Systems sind in Punkt 6 Datenhaltung und Datenbeschaffung aufgeschlüsselt.

Aufgrund firmeninterner personeller Änderung ist der damalige Entwickler dieser Software nicht mehr im Unternehmen. Zusätzlich dazu existieren im Unternehmen wenig bis keine Dokumentationen für dieses Programm. Eine fachliche Expertise für diese Programmiersprache bzw. Software fehlt ebenfalls. Bei auftretenden Schnittstellenfehlern ist es deshalb kompliziert, diese gezielt zu identifizieren und anschließend zu beheben. Konkret ist hier der Fehler der doppelten Übertragung zu nennen. Es kann das Problem auftreten, dass Daten mehrfach an das GHT-Portal übermittelt werden. Dies führt zu keinen spürbaren Auswirkungen bei dem Auslieferungsprozess. Allerdings tritt dieses Problem ebenfalls bei der Generierung des Sendungsberichtes auf. Hierbei werden Sendungsberichte doppelt generiert und importiert. Aufgrund dessen gibt es in den nachfolgenden Systemen Mechanismen, welche die Bereinigung dieser Duplikate vornehmen. Zusätzlich dazu ist seit Anfang April 2022 die Generierung der Sendungsberichte gestört. Dies stellt ein Problem dar, da somit keine Sendungsberichte vorliegen, welche die Zustellungen der Sendungen belegen. Der Aufwand für die Behebung war zu hoch, da alle Wissensträger aus dem Unternehmen ausgeschieden sind. Aufgrund dessen wurde mit dem GHT-Portal gearbeitet, welches die Zustellinformationen ebenfalls beinhaltet. Allerdings muss die Anfrage dieser Belege über den Fuhrpark stattfinden, da die Standortinformationen der Mitarbeiter über die MDE-Geräte jederzeit verfügbar sind. Das Berechtigungskonzept des GHT-Portals bietet nicht die Möglichkeit, ausschließlich auf die Sendungsdaten zuzugreifen. Dadurch ist der Prozess der Anfrage eines Sendeberichtes langwieriger als zuvor.

## 2.4 Produktionssystem ePS

Das erfal-Produktionssystem, kurz ePS, ist das hauseigene entwickelte MES bzw. Produktionsleitsystem. Dieses hat die zentrale Aufgabe, die Arbeiter zu unterstützen und einen reibungslosen Ablauf der Produktionsprozesse zu ermöglichen. Um Abhängigkeiten von proprietären Herstellern zu vermeiden, setzt das System auf Open-Source-Software. Dadurch wird es ermöglicht, dass dieses auf die individuellen Bedürfnisse der Firma angepasst werden kann.

Das System basiert auf einem modularen Klassenkonzept, welches in PHP entwickelt worden ist. Für die Darstellung wird HTML genutzt. Als Datenbankmanagementsystem wird MySQL von Oracle eingesetzt.

Die Schnittstellen für die elektronische Auslieferung sollen in dieses System integriert werden. Durch die unterschiedlichen Systemumgebungen (Windows Server 2012 und Ubuntu 18.04) ist eine Migration des Altsystems notwendig. In dem nachfolgenden Kapitel werden die theoretischen Grundlagen einer Migration erläutert.

# 3 Migration

## 3.1 Begriffsabgrenzung

Migration ist ein Prozess, der die Umstellung der bisherigen technologischen Umgebung in eine neue darstellt. Dabei kann es das gesamte System oder nur einen Teil betreffen. Im Regelfall handelt es sich bei solchen Systemen um Altsysteme bzw. Legacy-Systeme. [3, S. 3] Diese genügen den neuen Anforderungen an Hardware, Software und Geschäftsprozessen nicht mehr. Durch das Fehlen von Wissensträgern der älteren Software wird der Umstand der Überalterung zusätzlich verstärkt. [4, S. 33] Das typische Alter eines Legacy-Systems wurde 1995 auf zehn Jahre angegeben. [3, S. 7] Durch die immer kürzer werdenden Innovationszyklen muss diese Zeitspanne auf fünf Jahre korrigiert werden. [4, S. 3] Wie bereits oben beschrieben, wird ein System nicht allein aufgrund seines Alters als Legacy-System deklariert. Ausschlaggebend bei der Betrachtung eines Systems ist, ob das System gegenüber neuen Anforderungen bestehen bleiben kann. [3, S. 3]

Sollte eine Anwendung als Legacy-System deklariert werden, so gibt es zwei Möglichkeiten. Entweder wird das bestehende System weiterverwendet, mit dem Risiko, dass zukünftige Anforderungen nicht oder nur teilweise erfüllt werden und die Möglichkeit besteht, dass die Wettbewerbsfähigkeit gegenüber der Konkurrenz (stark) abnimmt. Oder es erfolgt eine Anpassung an die neuen Technologien. Wird die letztere Möglichkeit erwählt, so gibt es hierbei zwei verschiedene Varianten: eine Migration oder eine Neuentwicklung. In der Theorie wird versucht, diese beiden Varianten strikt voneinander zu trennen, allerdings ist eine Überschneidung möglich. Die folgende Definition einer Migration verdeutlicht diesen Sachverhalt: „Softwaremigration bezeichnet die Überführung eines Softwaresystems in eine andere Zielumgebung oder in eine sonstige andere Form, wobei die fachliche Funktionalität unverändert bleibt.“ [4, S. 25] Dies zeigt auf, dass es essenziell ist, dass bei einer Migration keine neuen Funktionen hinzugefügt werden. Fälschlicherweise wird eine Migration häufig als Anlass für das Hinzufügen neuer Funktionalitäten betrachtet. [4, S. 1 ff., 32 ff.]. Zusätzlich dazu wird eine Migration wie folgt eindeutig von einer Neuentwicklung abgegrenzt: „Die Neuentwicklung bestehender Legacy-Systeme ist keine Migration, allenfalls eine Entwicklung unter Wiederverwendung von Teilen des alten Systems.“ [4, S. 25] Dies ist notwendig, damit die Testbasis während einer Migration identisch bleibt. Durch eine einheitliche Basis ist es möglich, genaue Vergleiche bzw. Testszenarien der Systeme zu bewerten. Bei unterschiedlichen Ergebnissen der Tests ist dies ein Hinweis darauf, dass das neuentwickelte System ein anderes Verhalten

aufweist. Dies kann auf ein Programmierfehler oder ein Verständigungsproblem bei der Anforderungsermittlung zurückgeführt werden. Sollte die Testbasis nicht identisch sein, besteht das erhöhte Risiko, dass das Migrationsprojekt scheitert. Das Erweitern bestehender oder das Hinzufügen von neuer Funktionalität sollte demnach erst nach Abschluss einer Migration erfolgen. [4, S. 26]

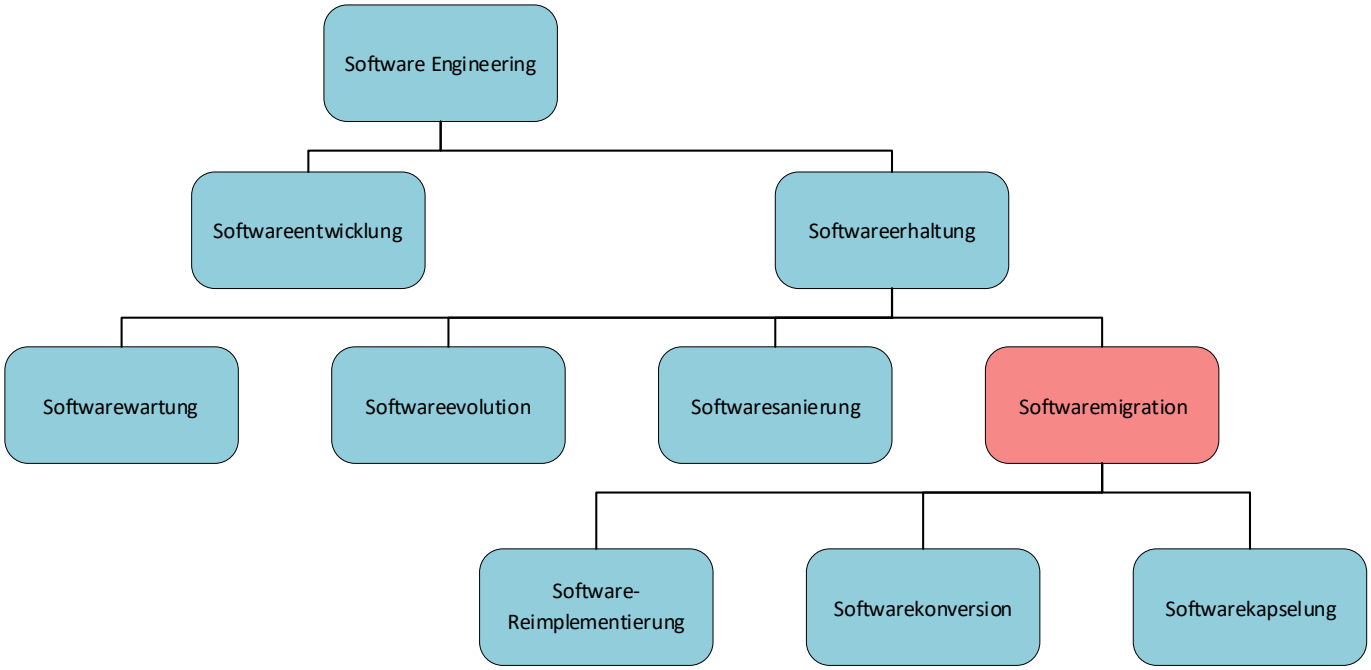



**Abbildung 2 Begriffliche Einordnung der Softwaremigration [4, S. 30]**

In Abbildung 2 ist ersichtlich, dass eine Softwaremigration zu einer von vier möglichen Maßnahmen der Softwareerhaltung gehört. Dabei sind die Begriffe untereinander zu unterscheiden. Eine Evolution bzw. eine Weiterentwicklung bestehender Software bezeichnet eine diskontinuierliche Anpassung. Der Auslöser für eine Weiterentwicklung ist, wenn die Architektur des Systems neuen bzw. veränderten Anforderungen nicht mehr standhalten kann. Bei einer maximalen Anpassung des Quelltextes in Höhe von 20 Prozent wird das Verfahren als Evolution bezeichnet. Überschreiten die Anpassungen diese Marke, handelt es sich um eine Neuentwicklung. Infolgedessen ist eine Softwareevolution das Vorstadium einer Software-Neuentwicklung. [4, S. 25–26]

Wartung bezeichnet die Korrektur von Fehlern bzw. Bugs des bestehenden Systems, welche so marginal sind, dass nicht von einer Softwareevolution gesprochen werden kann. [4, S. 27]

Bei einer Sanierung bzw. Reengineering einer Software wird die Verbesserung der Qualität des bestehenden Systems anvisiert. Analog zu einer Softwaremigration wird

die Funktionalität bzw. das sichtbare Verhalten nicht verändert. Gegenüber einer Migration wird bei einer Sanierung der Code und die Architektur verbessert. Mittels der Durchführung einer Sanierung vor einer Migration kann diese vereinfacht werden. Diese Methode weist Kongruenz zum in Kapitel 4 beschriebenen Refactoring auf. [4, S. 28–29]

## 3.2 Migrationsarten

Abhängig von dem zu migrierenden (Teil-) System kann die Migration in vier Arten unterteilt werden: [4, S. 38]

- Datenmigration
- Schnittstellenmigration
- Oberflächenmigration
- Programmmigration

Im Falle einer Migration des gesamten Systems, der komplexesten Form, handelt es sich um eine Systemmigration. [4, S. 38]

Bei einer Datenmigration bleiben die Programme in ihrer Funktionalität und Aufbau unverändert. Dabei werden ausschließlich die Daten in ein neues System übertragen. Ein Wechsel von einem relationalen Datenbankmanagementsystem, kurz DBMS, zu einem anderen stellt sich vergleichsweise einfach heraus. Dies kann mittels vollautomatisierten Tools ohne größeren Aufwand erfolgen. Im Gegensatz dazu stellt eine Migration eines veralteten Systems eine höhere Komplexität dar, wie z.B. die Migration einer Flat-File-Datenbank. Diese Daten liegen in einer Datei ohne Mechanismen zur Indizierung vor und haben ein festes Format ähnlich einer Text Datei. Diese zu einer (nicht) relationalen Datenbank zu migrieren, ist aufwändig. Hierbei handelt es sich häufig nicht um eine reine Datenmigration, da sich die Zugriffsarten auf die Daten verändern. [4, S. 38 ff.]

Die Schnittstellenmigration überführt jene Schnittstellen, über die das System mit anderen kommuniziert bzw. Daten austauscht, in das neue System. Ein Grund zur Durchführung einer Schnittstellenmigration ist, dass sich die außerhalb des Systemkontexts befindlichen Schnittstellen verändern. Ob es sich dabei um neu entwickelte Systeme mit neuen Protokollen oder migrierte Systeme handelt, ist irrelevant. [4, S. 43 ff.]

Analog zu einer Datenmigration bleibt bei einer Oberflächenmigration das Programm unverändert. Zusätzlich dazu werden die Daten ebenfalls nicht angepasst. Hierbei werden die Benutzerschnittstellen angepasst, was eine klare Trennung von der

Oberfläche bzw. des Frontends und der Programmlogik bzw. des Backends darstellt. Sollte eine zu starke Abhängigkeit des Frontends vom Backend bestehen, so muss zusätzlich zu der Oberfläche die Programmlogik migriert werden. [4, S. 42 ff.]

Eine Programmmigration überführt ausschließlich die Funktionalitäten, wobei die Oberfläche, Schnittstellen und Daten unberührt bleiben. Hierbei kann zwischen drei Varianten unterschieden werden: Einem Wechsel der Programmiersprache auf der gleichen Systemumgebung, einer Änderung der Umgebung bei gleicher Programmiersprache oder einem Wechsel der Programmiersprache auf eine andere Umgebung. [4, S. 41 ff.]

Nachfolgend sind die oben beschriebenen Migrationsarten kompakt dargestellt.

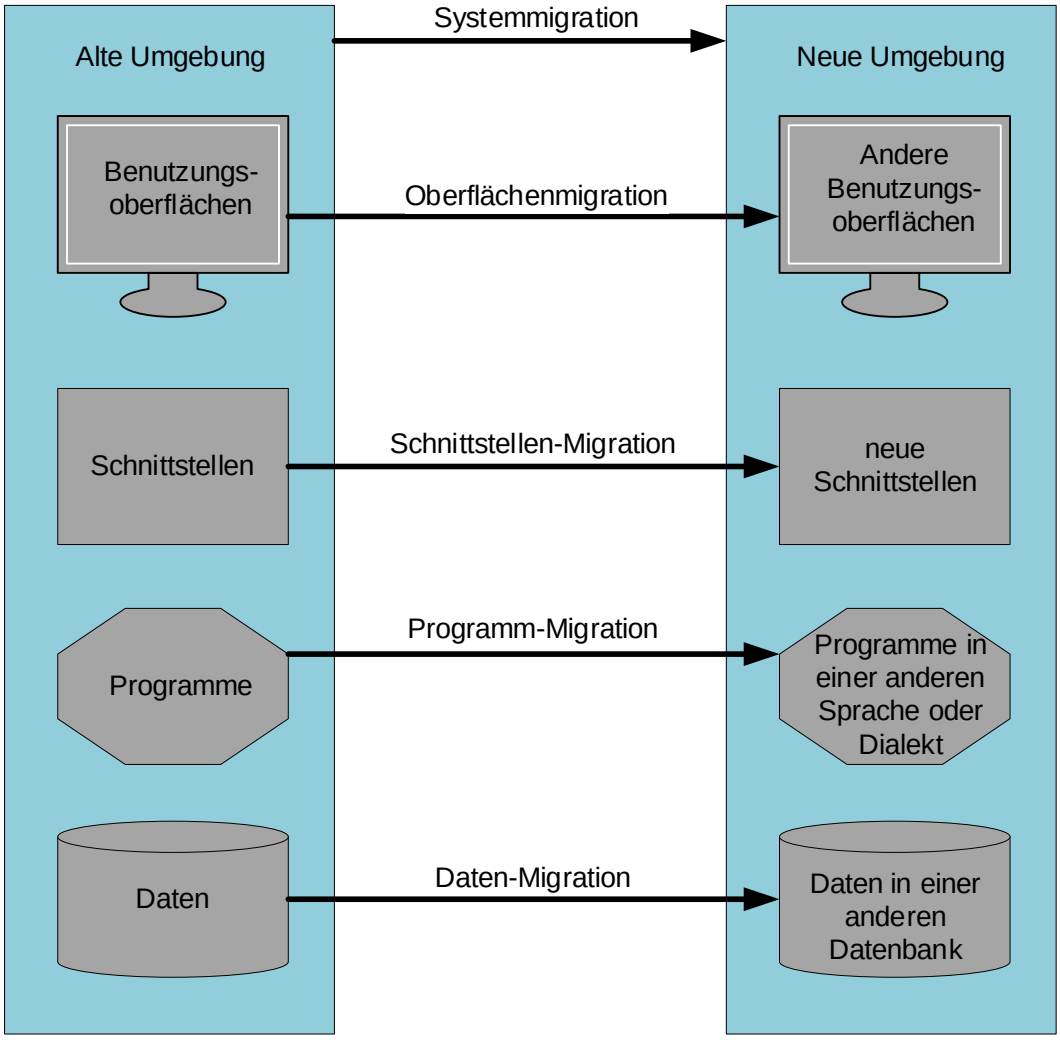



**Abbildung 3 Migrationsarten [4, S. 39]**

## 3.3 Migrationsstrategien

Die Herausforderung, die Begrifflichkeiten eindeutig voneinander abzugrenzen, welche in Punkt 3.1 bereits erläutert wurden, wird zusätzlich verdeutlicht bei den drei möglichen Strategien einer Migration. Diese lauten: [5, S. 20]

- Konversion
- Kapselung
- Reimplementierung

Für die letzte Strategie wird in der Literatur ebenfalls der Begriff Neuentwicklung benutzt. Hierbei wird dieser im Kontext nicht als Neuentwicklung eines Systems von Grund auf verstanden, sondern als eine von drei möglichen Strategien einer Migration eines Legacy-Systems. [5, S. 20]

Eine Migration kann sowohl einen Schwerpunkt auf Daten als auch auf Programme setzen. Dabei werden in Bezug auf eine Neuentwicklung bzw. Reimplementierung die Daten einer grundlegend neuen Struktur zugeordnet und ggf. formatiert. Ausschließlich die Semantik der Daten bleibt, im Vergleich zu der des Altsystems, identisch. Im Gegensatz dazu ist es bei den Programmen möglich, Funktionen neu zu definieren, den Zugriff auf die Datenbank zu verändern, sowie die eingesetzte Programmiersprache und Kommunikationssysteme auszutauschen. [5, S. 20]

Reimplementierung stellt demnach eine freie Übersetzung des Systems dar, wobei einzelne Bestandteile des Altsystems migriert werden. Die anschließend entstehenden Bestandteile des neuen Systems, müssen eine Übereinstimmung mit denen des Ursprungssystems aufweisen, damit dies eine Migration darstellt. [4, S. 35]

Die Strategie der Konversion hat das Ziel, die Daten und Programme unverändert zu migrieren. Lediglich die Form des Systems wird verändert. [4, S. 11] Im Gegensatz zu der Reimplementierung handelt es sich hierbei um eine präzise Transformation. Das bedeutet, die Datentypen des Altsystems werden analog in das neue System, ggf. mit einer anderen Programmiersprache, übersetzt. Die Anweisungsabfolge der migrierten Software wird zur Laufzeit in der gleichen Reihenfolge wie im Ausgangsprogramm ausgeführt. [4, S. 36]

Die dritte Migrationsstrategie, die Kapselung, stellt einen Mittelweg zu den obigen vorgestellten Strategien dar. Bei dieser werden weder Programme noch Daten reimplementiert oder konvertiert. Diese bestehen weiterhin in der alten Umgebung, werden aber von einem Wrapper, der auch als Softwareschale bezeichnet werden kann, umschlossen. Dadurch ist es möglich, dass die Oberfläche mit der Verarbeitungslogik reimplementiert wird und über Schnittstellen des Wrappers an die bestehende Programmlogik angebunden wird. Dieser ist folglich für die Transformation der Daten zuständig, welche über die Schnittstellen ausgetauscht werden. Somit ist es möglich, dass das Frontend bzw. die Benutzeroberfläche in

einer anderen Sprache neuentwickelt wird, wohingegen das Backend, die Programmlogik, auf dem alten System bestehen bleibt.[4, S. 12], [5, S. 22] Grundsätzlich ist die Strategie einer Konversion oder Kapselung, in Bezug auf die Daten, immer möglich. Diese können sich jedoch als zu aufwändig herausstellen, wenn die bisherige Datenstruktur ungewöhnlich oder veraltet ist. Analog zu der Programmstruktur ist es möglich, dass die Komplexität des Altsystem so hoch ist, dass eine Konversion oder Kapselung zu aufwändig ist bzw. zu hohe Kosten verursacht. Die benötigten Refactorings bzw. Sanierungsmaßnahmen würden einen solchen Aufwand verursachen, dass dies betriebswirtschaftlich nicht effizient ist. [5, S. 19]
In der nachfolgenden Abbildung 4 sind die verschiedenen Migrationsstrategien und deren Ergebnisse bildlich dargestellt.

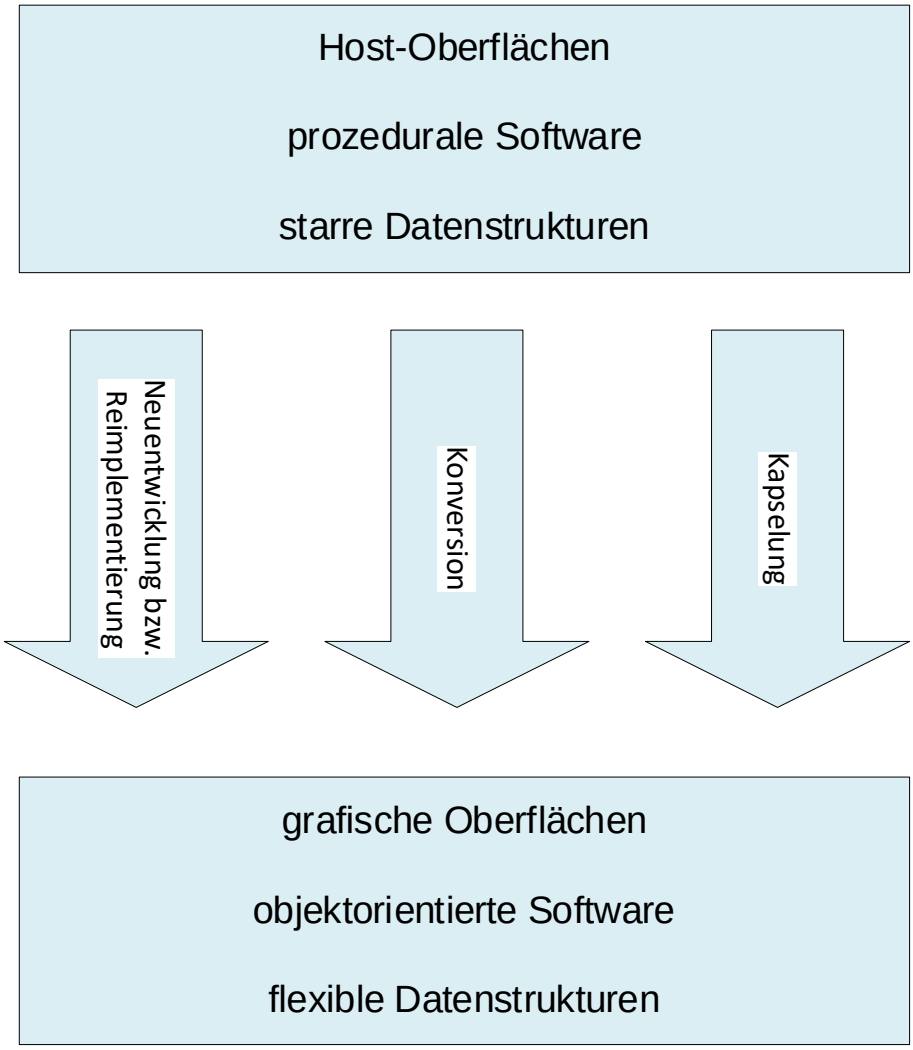



**Abbildung 4 Die drei Migrationsstrategien [5, S. 19]**

Des Weiteren können die drei verschiedenen Migrationsstrategien unterschiedlich, hinsichtlich des Vorgehens, umgesetzt werden. Davon unabhängig ist die gewählte Strategie.

## 3.4 Vorgehensmodelle in der Migration

### 3.4.1 Cold-Turkey-Ansatz

Der Cold-Turkey-Ansatz ist eine der klassischen Vorgehensweisen bei einer Migration, übersetzt bedeutet dies „kalter Entzug“. Dabei wird das gesamte Legacy-System in dem neuen Zielsystem reimplementiert. Während dieser Entwicklungszeit bleibt das Altsystem parallel bestehen und kann ohne Einschränkungen verwendet werden. Nachdem die Entwicklung des neuen Systems abgeschlossen ist, wird in einem einzigen Schritt, dem sogenannten „Big Bang“, von dem alten System auf das Neue umgestellt. [3, S. 50 ff.]

Durch diesen Ansatz ergeben sich aber Risiken für eine erfolgreiche und funktionierende Migration. Wie bereits im vorherigen Kapitel erwähnt, sind die Altsysteme meist unzureichend dokumentiert. Dadurch benötigt die Reimplementierung mehr Zeit. Innerhalb dieser Entwicklungszeit können Änderungen am Altsystem durchgeführt werden, um die Bedürfnisse der Prozesse bzw. des Unternehmens zu erfüllen. Dadurch entsteht die Schwierigkeit, dass diese Änderungen ebenfalls in das neue System überführt werden müssen. Dies stellt die Entwickler vor eine Herausforderung, da der Vorgang fehleranfällig und kostenintensiv ist. Ein weiteres Problem für die Entwickler sind die Abhängigkeiten zwischen den Systemen. Das Altsystem kann in Verbindung mit anderen Systemen stehen, welche im Laufe der Entwicklungszeit gewachsen und nicht dokumentiert sind. In Gesamtheit können diese Schnittstellen bzw. Abhängigkeiten von unkritisch bis geschäftskritisch reichen. [3, S. 9 ff.]

Weist das Altsystem einen geringen Umfang bzw. eine geringe Komplexität auf, sind genügend personelle sowie monetäre Ressourcen verfügbar und stellen die neuen Funktionalitäten einen Mehrwert für das Unternehmen dar, so kann eine Neuentwicklung auf Basis des Cold-Turkey Ansatzes gerechtfertigt werden. [4, S. 8 ff.]

### 3.4.2 Chicken-Little-Ansatz

Im Gegensatz zu dem im Punkt 3.4.1 vorgestellten Cold-Turkey-Ansatz weist der Chicken-Little-Ansatz eine inkrementelle Umstellung auf. Dadurch ist eins der

Hauptrisiken des Cold-Turkey-Ansatzes, welche im schlimmsten Falle zum Scheitern des Migrationsprojektes beitragen, eliminiert. „Die Migration des bestehenden Informationssystems erfolgt in kleinen, inkrementellen Schritten, bis das gewünschte langfristige Ziel erreicht ist. Chicken Little erlaubt es den Planern, das Risiko Schritt für Schritt zu kontrollieren, indem sie die Größe der Inkremente bestimmen können.“ [3, S. 13] (aus dem Englischen Übersetzt)

Das Legacy-System und das neue Zielsystem werden während des gesamten Migrationsprozesses parallel betrieben. Dadurch können bereits migrierte Elemente in dem Zielsystem verwendet werden, wobei die noch nicht migrierten Funktionalitäten im Altsystem weiterhin funktionsfähig bleiben. Daraus hervorgehend kann sich die Herausforderung ergeben, dass die Daten, welche von dem Altsystem verarbeitet und geschrieben werden, dem neuen System ebenfalls zur Verfügung stehen. Ist dies notwendig, so werden Gateways eingesetzt, welche die Datenkonsistenz der beiden Systeme untereinander gewährleisten. Dies hat zur Folge, dass die Komplexität der Migration zunimmt, wodurch der Zeitaufwand und die Kosten steigen. Durch die inkrementellen Schritte scheitert das Migrationsprojekt dagegen nie im Ganzen, sondern ausschließlich der vorgenommene Schritt, welcher nach genauerer Betrachtung wiederholt werden kann. [4, S. 51]

Dabei spielt die Größe der Schritte eine entscheidende Rolle. Die Grundlage bei der Wahl der Größe des durchzuführenden Schrittes ist das „divide et impera“ (teile und herrsche) – ein Verfahren der Informatik. Dabei wird ein unlösbares bzw. unbeherrschbares Problem so lang zerlegt, bis die entsprechende Unteraufgabe nahezu trivial erscheint. [6] Dadurch sind auch auftretende Probleme leichter erkennbar und anschließend zu beheben als bei dem zuvor erklärten Cold-Turkey-Ansatz. Die Größe des zu wählenden Schrittes ist von folgenden zwei Faktoren abhängig [3, S. 13 ff.]:

- Die Höhe des einzugehenden Risikos der Migrationsplanung.
- Die Zerlegbarkeit des Systems, das bedeutet in welchem Umfang kann das System in separate Teile aufgesplittet werden. Dies ist wiederum abhängig von der Systemarchitektur.

Um den zweiten Faktor, die Zerlegbarkeit, konkret beurteilen zu können, ist im Vorfeld eine intensive Analyse notwendig. Diese soll beurteilen, wie stark verschiedene Komponenten aber auch Funktionsschichten miteinander verbunden sind. Dadurch unterscheidet man zwischen drei verschiedenen Systemarchitekturen [3, S. 14]:

- zerlegbare
- semi-zerlegbare und
- nicht-zerlegbare

Bei zerlegbaren Architekturen weisen die Komponenten eine Unabhängigkeit untereinander auf und sind durch Schnittstellen an andere Systeme, wie z.B. Datenbank, Benutzer- und Systemschnittstellen, angebunden. Ein nicht-zerlegbares System kann auch als Black-Box bezeichnet werden, da das System in seiner Funktionsweise nicht verständlich ist. Dagegen können semi-zerlegbare Systeme nur teilweise aufgeteilt werden. Die Applikations- und Datenbankschicht sind so stark miteinander verflochten, dass eine Trennung schlichtweg nicht möglich ist. Dabei können Schnittstellen voneinander getrennt betrachtet werden, wie z.B. Benutzer- und Systemschnittstellen. [3, S. 13–17]

Der Chicken-Little-Ansatz ist in elf Schritte unterteilt, wobei die Abfolge dieser Schritte im Vergleich zum Cold-Turkey-Ansatz nicht die Migration des gesamten Systems beschreibt. Der Ablauf dieser Schritte ist für jeden Teil separat zu wiederholen. Somit ergibt sich eine vollständige Migration aus einer Vielzahl an Durchläufen dieser elf Schritte. [3, S. 32]

1. Teil-Analyse des Altsystems
2. Teil-Zerlegung der Struktur des Altsystems
3. Teil-Entwurf der Schnittstellen des Zielsystems
4. Teil-Entwurf der Applikation des Zielsystems
5. Teil-Entwurf der Datenbank bzw. der Struktur des Zielsystems
6. Teil-Implementierung des Zielsystems
7. Teil-Entwicklung und Einrichtung der Gateways
8. Teil-Migration der alten Datenbank
9. Teil-Migration der alten Applikationen
10. Teil-Migration der alten Schnittstellen
11. Teil-Umstellung auf das neue Zielsystem

In Abhängigkeit des jeweiligen Migrationsprozesses können einzelne Schritte vernachlässigt oder sogar vollständig ignoriert werden. Wird ausschließlich eine Schnittstellenmigration vorgenommen, so muss kein Entwurf, keine Implementierung und Migration der Applikationen und ggf. Datenbank erfolgen. Zusätzlich dazu ist die Reihenfolge der Schritte nicht zwingend in der Abfolge zu vollziehen, wie oben beschrieben. Der Chicken-Little-Ansatz beinhaltet das Ziel, diese Schritte unabhängig voneinander ausführbar zu machen, analog der Aufgabe der Gateways.

Gesamtheitlich minimiert dieser Ansatz das Risiko eines Migrationsprojektes, hat aber gleichzeitig zur Folge, dass erhöhte Kosten und Zeitaufwände notwendig sind, um die Gateways zu entwickeln. [3, S. 30]
Zusätzlich zu den zwei vorgestellten Migrationsmodellen existiert ein weiteres, der sogenannte Butterfly-Ansatz. Da sich dieser ausschließlich auf die Datenmigration bezieht und das Ziel dieser Arbeit eine Migration einer Software ist, bleibt eine Vorstellung des dritten Ansatzes aus.

# 4 Refactoring

## 4.1 Begriffsabgrenzung

Unabhängig von der Abgrenzung der Begriffe Migration und Refactoring ist es möglich, die beiden Methoden gemeinsam zu verwenden. Dabei ist aber zu beachten, dass neue Funktionalitäten erst nach einer Migration hinzugefügt werden. Ein strukturiertes Refactoring kann die benötigte Zeit, um eine Migration durchzuführen, verringern. [4, S. 26]

Refactoring bedeutet, dass die interne Struktur einer Software verändert wird, ohne dabei die Funktionalität, sowie das sichtbare Verhalten, zu beeinflussen. Damit wird das Programm verständlicher und ist leichter zu erweitern. Refactoring weist eine große Gleichartigkeit mit der Performance-Optimierung auf. Beide Methoden verändern den Code, ohne dabei die Funktionalität der Software zu verändern. Allerdings setzt die Performance-Optimierung auf die Beschleunigung des Codes. Durch diese Veränderungen kann das Programm leichter oder schwerer verständlich werden. Auch kann die Erweiterbarkeit positiv oder negativ beeinflusst werden. Im Umkehrschluss ist es bei einem Refactoring möglich, dass die Performance der Software zu- oder abnimmt. [1, S. 79–80]

Einer der Hauptgründe, um ein Refactoring durchzuführen, ist das Aufrechterhalten des Softwaredesigns. Ohne ein Refactoring droht dessen Zerfall. Oftmals wird der Code verändert, um kurzfristige Ziele zu erreichen, ohne dabei die vollständige Struktur des Programmes zu verstehen. Die Software verliert dadurch Stück für Stück ihre Architektur, wobei sich dieser Effekt noch weiter verstärkt, je mehr Anpassungen vorgenommen werden. Das Design der Software weicht mit jeder Anpassung, die vorgenommen wird, von dem ursprünglichen ab. Umso schwieriger wird es, die Struktur der Software zu verstehen und zu bewahren. Bei einer schlechten Softwarearchitektur ist häufig mehr Code notwendig, um das gleiche Ergebnis zu erzielen als im Vergleich zu einer guten. Bei der Entwicklung von Programmen wird häufig vernachlässigt, dass auch andere Menschen diesen Quelltext verstehen müssen, um spätere Anpassungen daran vornehmen zu können. Dabei macht es einen großen Unterschied, ob der zukünftige Entwickler einen Großteil der Arbeitszeit in das Verständnis oder in die Weiterentwicklung des Programmes investiert. Dabei reicht bereits ein geringfügiger Zeitaufwand aus, um das Programm verständlicher und besser zu strukturieren. Dieser geringe Zeitaufwand hilft ebenso, Bugs (Logik- und Programmfehler) schneller zu identifizieren und anschließend zu beheben. Ein Refactoring kann die

Verständlichkeit des Quelltextes erhöhen, um schneller Fehler zu finden und zu beheben. Zusätzlich kann es dazu beitragen, dass die Entwicklungszeit verringert wird. Mangelhaftes Design weist in der Regel mehrere technische Schulden auf. Diese technischen Schulden beschleunigen die Entwicklung kurzfristig. In dieser Zeit hat eine Software mit gutem Design ein schwächeres Wachstum der kumulativen Funktionalität. Wird das Bestehen der Software und deren Code auf langfristige Zeit betrachtet, so ist ersichtlich, dass die Weiterentwicklung von Software mit mangelhaftem Design wesentlich langsamer vonstattengeht. Dies lässt sich auf die technischen Schulden zurückführen, welche die Weiterentwicklung am Anfang beschleunigt haben. Dies ist grafisch in Abbildung 5 dargestellt. [1, S. 81–84]

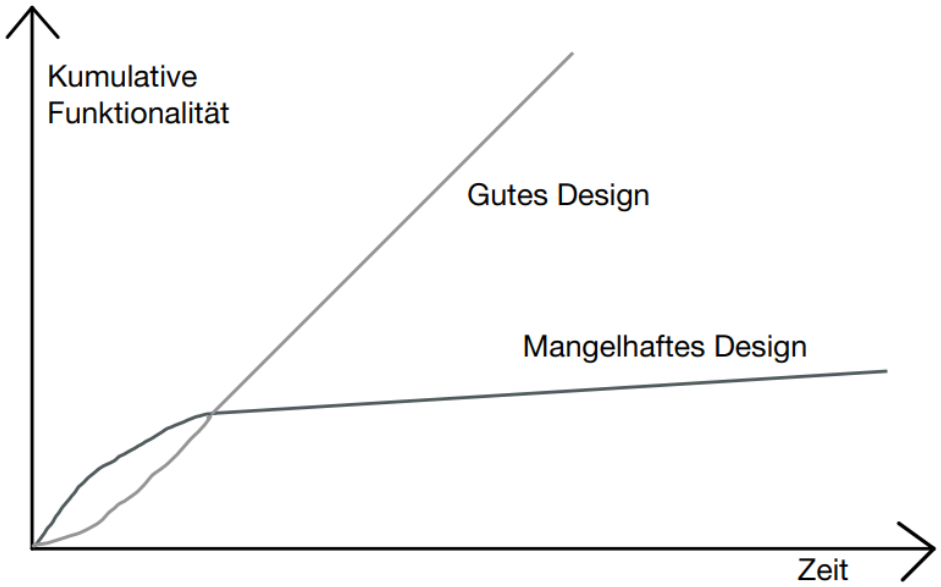



**Abbildung 5 Gutes und Mangelhaftes Design Vgl. [1, S. 83]**

Bei einer guten Software ist es aufgrund von Modularität nur notwendig, einen kleinen Teil der Codebasis zu verstehen und nicht das komplette Programm. Es ist schnell ersichtlich, an welchen Stellen Änderungen vorgenommen werden müssen, um eine neue Funktion bzw. ein Feature zu implementieren. Dabei sinkt die Wahrscheinlichkeit Fehler zu machen. Sollten dennoch Softwarefehler auftreten, so ist es einfacher, das Programm zu debuggen und diese Fehler zu beseitigen. [1, S. 82–84]

## 4.2 Refactorings

### 4.2.1 Katalog

Der Katalog der möglichen Refactorings ist sehr umfangreich. In dem Buch von Martin Fowler „Wie Sie das Design bestehender Software verbessern“ [1] sind auf über 300 Seiten eine Vielzahl von möglichen Refactorings aufgelistet. Der Aufbau

dieser verschiedenen Methoden ist immer gleich: Zuerst wird die Motivation der spezifischen Methode erläutert, anschließend erfolgt die Beschreibung des grundlegenden Vorgehens und zuletzt wird die Methode an einem spezifischen Beispiel durchgeführt.
Aufgrund der hohen Anzahl an Refactorings werden nachfolgend nur solche vorgestellt, welche eine Affinität zum Projekt aufweisen. Bei Bedarf kann der Katalog aus dem o.g. Buch ab Seite 141 verwendet werden.
Des Weiteren sind nachfolgend alle Beispiele in der Programmiersprache PHP verfasst.

### 4.2.2 Funktion extrahieren

Motivation

Die Methode *Funktion extrahieren* bezieht sich sowohl auf funktionale als auch objektorientierte Ansätze, welche die beiden hauptsächlichen Varianten von Programmiersprachen sind.[1, S. 145]
Eine Extraktion einer Funktion sollte dann vollzogen werden, wenn der Zweck bzw. die Aufgabe des Codeabschnittes nicht klar ersichtlich ist. Die Benennung der Funktion sollte der Aufgabe entsprechen. Dadurch ist es auf den ersten Blick ersichtlich, welche Aufgabe diese Funktion vollzieht und muss nicht im Detail verstanden werden. Diese kann ggf. als Blackbox betrachtet werden. Der eigentliche Inhalt bzw. das Verständnis, wie die Funktion ihre Aufgabe erfüllt, wird dadurch irrelevant. Dabei ist die Länge des Codes belanglos. In einigen Fällen kann es durchaus sinnvoll sein, dass eine Funktion nur aus einer Codezeile besteht.[1, S. 146–147] Dies kann notwendig werden, wenn „… zwischen dem Zweck des Codes und der Implementierung ein großer semantischer Abstand …“[1, S. 147] besteht.
Als Beispiel: der Name einer Funktion aus einer Bibliothek bzw. des Systems beschreibt die Aufgabe nicht bzw. fehlerhaft. In diesem Fall stellt der Aufruf einer Funktion über einer anderen eine mögliche Lösung dar, wobei die übergeordnete Methode einen eindeutigeren Namen besitzt. [1, S. 146–147]
Durch das Verwenden von kurzen Funktionen können Zweifel aufkommen, dass die Performance unter der Vielzahl an Funktionsaufrufen leidet. In der Regel beeinflusst dies die Performance von Programmen nicht. Durch die kurzen Funktionen kann modernen Compilern das Caching erleichtert werden und dies führt zu besseren Ergebnissen.[1, S. 147]

Vorgehen

Zuerst wird eine neue Funktion deklariert und ihrer Aufgabe entsprechend benannt. Die Benennung sollte sich hierbei nach der Funktionalität richten und nicht, wie diese Aufgabe durchgeführt wird. Ein erster Hinweis, ob die Funktion extrahiert werden sollte, gibt der Funktionsname. Ist es nicht möglich, der Funktion anhand ihrer Aufgabe, einen prägnanten Namen zu geben, dann kann dies ein Hinweis darauf sein, dass die Funktion nicht extrahiert werden sollte. Wenn die benutzte Programmiersprache verschachtelte Funktionen unterstützt, so sollte dieser Mechanismus genutzt werden. [1, S. 147]
Im zweiten Schritt wird der extrahierte Quelltext in die neue Funktion kopiert. „Suchen Sie im extrahierten Code nach Verweisen auf Variablen, die sich innerhalb des Gültigkeitsbereichs der Quellfunktion befinden, aber nicht in den Gültigkeitsbereich der Zielfunktion fallen. Übergeben Sie diese Variablen als Parameter.“. [1, S. 148] Sollte die genutzte Programmiersprache verschachtelte Funktionen unterstützen, so entfällt dieser Schritt. [1, S. 148]
Darauffolgend sollte der Code kompiliert werden, vorausgesetzt die Entwicklungsumgebung unterstützt Überprüfungen zur Laufzeit. Sollte dies zu keinem Fehler führen, so wird der Code der Quellfunktion durch einen Aufruf der extrahierten Funktion ersetzt. Anschließend müssen Tests durchgeführt werden. Anschließend sollte nach weiterem Code gesucht werden, welcher mit dem extrahierten Code identisch oder ähnlich ist. Anschließend wird dieser nach dem o.g. Prinzip ersetzt. [1, S. 148]

Beispiel:

Im nachfolgenden Beispiel wird einer Funktion ein Array (eine Liste mit einem numerischen Schlüssel) übergeben. Diese Funktion hat die Aufgabe die Summe aller Werte anzuzeigen.

```
function sumValues(ARRAY $values) {
   $sum = 0;
   echo "Die Summe des Arrays ist : ";

   foreach ($values as $key => $value) {
      $sum += (int) $value;
   }
```

```
    echo "Ergebnis: $sum";
}
```

Der Name der Funktion *sumValues* spiegelt nicht ihr gesamtes Verhalten wider. Dahingehend kann die Funktion wie folgt einem Refactoring unterzogen werden:

```
function printDetails(INT $sum) {
    echo "Die Summe des Arrays ist : ";
    echo "Ergebnis: $sum";
}

function sumValues(ARRAY $values) {
    $sum = 0;

    foreach ($values as $key => $value) {
        $sum += (int) $value;
    }

    printDetails($sum);
}
```

Nachdem die Funktion extrahiert wurde, ist die Übersichtlichkeit und der Zweck der Funktionen klar ersichtlich.

### 4.2.3 Funktion inline platzieren

Motivation

Die Methode *Funktion inline platzieren* ist das Komplement der Methode *Funktion extrahieren*. Diese wird dann eingesetzt, wenn die Anzahl der Indirektionen (die Aufrufe der Funktion) unübersichtlich hoch wird und nahezu jede Funktion Aufgaben an eine andere übergibt. Durch eine solche Verschachtelung leiden die Übersichtlichkeit und Verständlichkeit des Codes. [1, S. 156]

Vorgehen

Zuerst muss sichergestellt werden, dass es sich bei der Funktion um keine polymorphe Funktion (eine Funktion, welche in anderen Klassen überschrieben werden kann) handelt. Denn diese kann nicht inline platziert werden. Anschließend werden alle Aufrufe der Funktion recherchiert und etappenweise ersetzt. Dabei ist es wichtig, nach jedem Austausch Tests durchzuführen, um die Funktionsfähigkeit der

Software zu gewährleisten. Sind alle Funktionsaufrufe ersetzt worden, kann die Funktionsdefinition gelöscht werden. [1, S. 156]
Das Verfahren ist grundlegend simpel aufgebaut, kann allerdings z.B. bei Rekursion oder mehreren return-Anweisungen zu Schwierigkeiten führen. Die Empfehlung von Fowler lautet hierbei dieses Refactoring nicht durchzuführen. [1, S. 157]

Beispiel:
Im nachfolgenden Beispiel soll die Funktion die Preiskategorie eines übergebenen Artikels errechnen.

```
function getPriceCategory($article) {
   return getPriceArticle($article) ? 2 : 1;
}

function getPriceArticle($article) {
   return $article->price > 1000;
}
```

In dieser Verschachtelung von Funktionen leidet die Übersichtlichkeit bzw. das Verständnis. Deswegen kann dieses Konstrukt refactort werden.

```
function getPriceCategory($article) {
   return ($article->price > 1000) ? 2 : 1;
}
```

Vgl. [1, S. 155–156]

Nachdem die zwei Funktionen zu einer zusammengefasst wurden, ist dies nicht nur kürzer, sondern auch wesentlich übersichtlicher.

### 4.2.4 Variable extrahieren

Motivation
Durch eine lokale Variable ist es möglich komplexe Ausdrücke zu adaptieren und diese verständlicher und übersichtlicher zu gestalten. Dadurch ist ein Debuggen der Software einfacher, da diese Variable explizit überwacht werden kann. [1, S. 159]

Vorgehen

Dieses Refactoring ist ideal, um eine Variable innerhalb der zu bearbeitenden Funktion bereitzustellen. Sollte diese Bereitstellung über mehrere Ebenen notwendig sein, so wird dies klassischerweise über eine Funktion abgebildet. Möchte man einen Ausdruck extrahieren, so muss zuallererst sichergestellt werden, dass der zu extrahierende Term keine Nebeneffekte verursacht. Anschließend wird eine unveränderliche Variable deklariert, welche eine Kopie des Ausdrucks erhält. Danach wird schrittweise der ursprüngliche Ausdruck ersetzt und anschließend wird das Programm bzw. der Abschnitt getestet. Dies wird solange wiederholt, bis alle Vorkommnisse des Ausdruckes ersetzt sind. [1, S. 160]

Beispiel:

Das folgende Beispiel berechnet den Gesamtpreis einer Bestellung.

```
return $order->quantity * $order->articlePrice
      - max(0, $order->quantity - 1000) * $order->quantity * 0.02
      + max(5.99, 0.02 * $order->quantity * $order->articlePrice);
```

Nach dem Refactoring:

```
$basicPrice = $order->quantity * $order->articlePrice;
$quantityDiscount = max(0, $order->quantity - 1000) *
                    $order->quantity * 0.02;
$shippingCosts = max(5.99, 0.02 * $basicPrice);
return $basicPrice - $quantityDiscount + $shippingCosts;
```

Vgl. [1, S. 162]

Das Refactoring hat nicht nur die Übersichtlichkeit verbessert, sondern auch die Bedeutung der Codezeilen verdeutlicht.

### 4.2.5 Variable inline platzieren

Motivation

Das Refactoring *Variable inline platzieren* ist die Umkehrung von *Variable extrahieren*. [1, S. 163]

„Manchmal ist der Name allerdings gar nicht aussagekräftiger als der Ausdruck selbst. Mitunter behindert eine Variable sogar das Refactoring des benachbarten Codes.“.[1, S. 164]

Vorgehen

Das Vorgehen ist in umgekehrter Reihenfolge analog der *Variable extrahieren* zu verstehen. Hierbei wird zuerst wieder geprüft, ob die Zuweisung zu keinerlei Nebeneffekten führt. Zusätzlich ist zu prüfen, ob eine Wertzuweisung nur einmal durchgeführt wird. Dies kann durch einen Test erreicht werden, wobei die Variable zuvor als unveränderlich deklariert wird. War der Test erfolgreich, so können stufenweise alle Vorkommnisse der Variable durch den Ausdruck ersetzt werden, wobei nach jeder Ersetzung ein Test erfolgt. Sobald alle Referenzen der Variable gelöscht worden sind, wird die Deklaration gelöscht und anschließend ein letztes Mal getestet.

Beispiel:

```
$articlePrice = $article->price;
return $articlePrice > 1000;
```

Nach dem Refacotring

```
return $article->price > 1000;
```

# 5 Anforderungsanalyse

## 5.1 Begriffsabgrenzung

„Das Ermitteln von Anforderungen ist ein Akt der Verführung. Sie müssen Ihr Gegenüber dazu verführen, sein Wissen preiszugeben, seine Visionen mit Ihnen zu teilen. Es ähnelt damit einem Flirt, denn der Flirt ist ein Spiel, bei dem man nicht weiß, ob man noch in der Qualifikation ist oder schon im Finale. Technisch gesehen lässt sich das auch trockener formulieren. Ziel ist es, mit möglichst geringem Aufwand und angepasst an die Rahmenbedingungen des Vorhabens, die Ziele und Anforderungen zu erfassen, um ein System zu entwickeln, das den Stakeholdern möglichst viel Gewinn bringt.“ [7, S. 130]

## 5.2 Das Kano-Modell

In der Anforderungsermittlung wird zwischen drei Wissensarten unterschieden [7, S. 132]:

- Unterbewusstes Wissen (Basisfaktoren) ist dem Wissenden nicht aktiv bekannt, kann aber durch einen Bewusstmachungsprozess aufgerufen werden.
- Bewusstes Wissen (Leistungswissen) ist aktiv abrufbar und die Bedeutung ist vollständig erkennbar.
- Unbewusstes Wissen (Begeisterungsfaktoren) sind ähnlich unbekannten Wünschen. Diese werden erst als Anforderung erkannt, wenn sie von einer dritten Partei herangetragen werden.

Bereits 1978 erstellte Dr. Noriaki Kano das Kano-Modell, welches die Funktionen eines Produktes in drei Kategorien untergliedert. Diese haben einen unterschiedlichen Einfluss auf die Zufriedenheit des Kunden mit dem Produkt und sind analog der zuvor erklärten Wissensarten anzusehen. Kano unterteilt Features in folgende Kategorien [7, S. 132]:

- Basisfaktoren sind selbstverständlich und unterbewusst vorausgesetzte Features.
- Leistungsfaktoren sind die bewusst verlangte Sonderausstattung.
- Begeisterungsfaktoren sind Features des Produkts, die dem Stakeholder (der Person, die die Anforderung stellt) noch unbewusst sind und, die er erst während der Benutzung als angenehme Überraschung entdeckt.

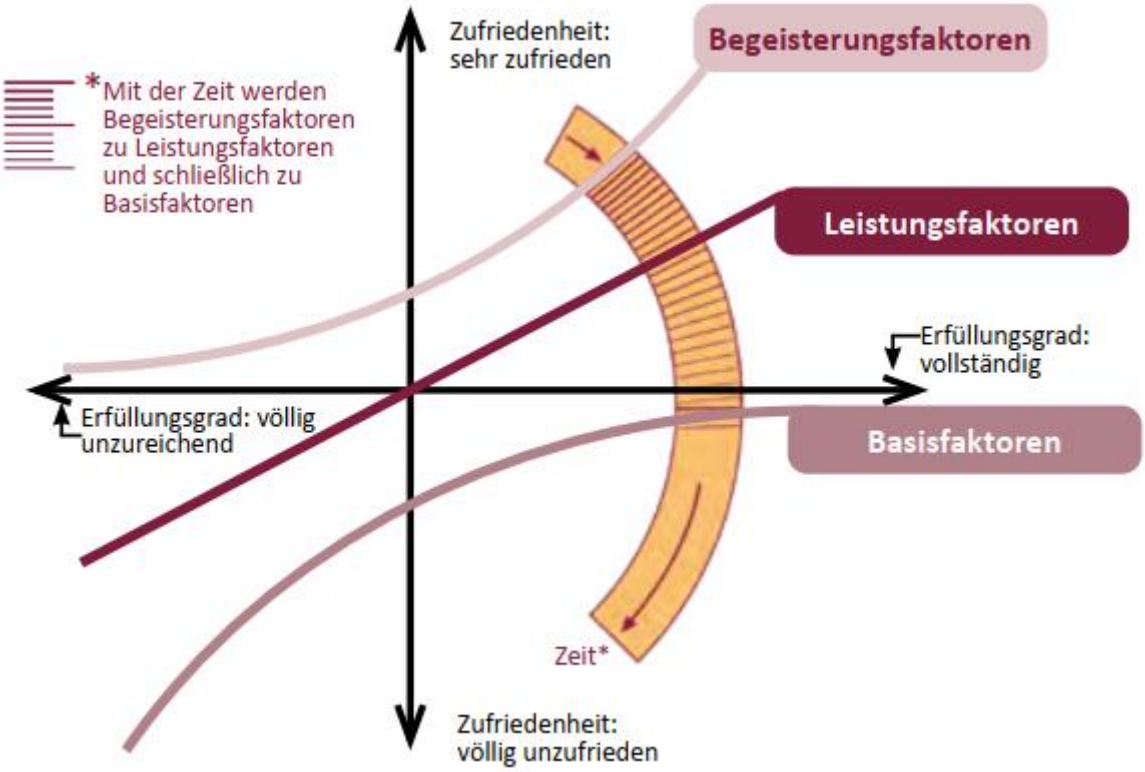



**Abbildung 6 Das Kano-Modell [7, S. 132]**

1992 war das Feature bzw. die Funktion eine SMS zu verschicken noch exotisch. Mit der Zeit wurde dies immer häufiger gefordert. Nach einer gewissen Zeitspanne wurde diese zu einem Basisfaktor, welcher implizit voraussetzt, dass ein neues Handy bzw. Smartphone diese Funktionalität besitzt. Ist dies nicht der Fall, dann sinkt die Zufriedenheit des Käufers stark. Um dies zu vermeiden, werden verschiedenste Ermittlungstechniken eingesetzt. Im nachfolgenden Kapitel sind die wichtigsten dieser Methoden erläutert.

## 5.3 Ermittlungstechniken

### 5.3.1 Befragungstechniken

Fragebogen

Um Anforderungen mit einem Fragebogen zu ermitteln, ist dieser im ersten Schritt zu erstellen. Dabei können sowohl offene und geschlossene Fragen wie auch Multiple-Choice-Fragen Bestandteil sein. Zu beachten ist, dass die Antwortmöglichkeit „weiß ich nicht“ miteinbezogen werden sollte. Suggestivfragen, Fragen, welche durch ihre Formulierung den Befragten die Antwort ganz oder teilweise vorgeben, sind zu unterlassen. Nach der Erstellung müssen die Fragebögen gezielt an die jeweiligen Stakeholder ausgegeben werden. Zusätzlich dazu ist es wichtig, einen festen

Rückgabetermin zu definieren. Damit die Stakeholder sich an diesen Rückgabetermin halten, kann es hilfreich sein, die Leitungsebene mit einzubeziehen. Die Rückläufer werden anschließend ausgewertet. [7, S. 141–142]
Diese Methode eignet sich besonders um die Stakeholder zeit- und kostensparend mit einzubeziehen. Weiterhin ist die elektronische Verteilung der Bögen und deren Auswertung hilfreich bzw. zeitsparend. Allerdings ist diese Methode schlecht geeignet, um implizites Wissen zu ermitteln. Zudem kann die Fragestellung die Antwort des Stakeholders beeinflussen.[7, S. 142]

Interview

Ein Interview ist im Gegensatz zu einem Fragebogen individueller. Dabei ist der prinzipielle Ablauf für ein Interview derselbe. Dieses muss vorbereitet und anschließend mit dem entsprechenden Stakeholdern durchgeführt werden. Hierbei ist das Protokollieren des Interviews hervorzuheben, zur Unterstützung kann ggf. eine Audioaufzeichnung verwendet werden. Die Nachbereitung und anschließende Auswertung sind analog zu einem Fragebogen durchzuführen. Der einzige Unterschied hierbei ist, dass das entstandene Protokoll von dem Befragten unterzeichnet werden sollte. Dies verhindert, dass fehlerhafte Anforderungen aufgrund von Kommunikationsfehlern aufgenommen werden. [7, S. 143–147]
Das Interview kann individuell auf die einzelnen Stakeholder angepasst werden. Dabei ist die Wahrscheinlichkeit, dass Fragen unbeantwortet bleiben, geringer als bei einem Fragebogen. Allerdings geht diese Individualität auch mit einem wesentlich höheren Zeitaufwand einher. Zusätzlich dazu ist die Effektivität dieser Methode stark von dem Interviewer abhängig. [7, S. 147]

### 5.3.2 Beobachtungstechniken

Feldbeobachtung

Durch das Beobachten von Prozessen und Arbeitsabläufen können sprachlich schwer beschreibbare Tätigkeiten einfach nachvollzogen werden. Außerdem fallen dadurch Abweichungen von Prozessen gegenüber den Dokumentationen leicht auf. Als Hilfsmittel zur Nachbereitung kann hierbei die Videoaufzeichnung (mit vorherigem Einverständnis) genutzt werden. Bei schwer beobachtbaren Vorgängen stellt die Feldbeobachtung kein geeignetes Mittel dar. Außerdem können sich Stakeholder durch die Anwesenheit eines Beobachters unwohl fühlen und führen ggf. Schritte anders aus als gewohnt.[7, S. 149–150]

Apprenticing

Im Gegensatz zu der Feldbeobachtung wird bei dieser Methode der Stakeholder (Meister) nicht beobachtet, sondern dieser schult den Analytiker (Lehrling). Der Lehrling kann unklare Handlungsschritte sofort hinterfragen und wird von dem Stakeholder in den Arbeitsablauf eingearbeitet. [7, S. 151–152]

Diese Technik ist besonders geeignet, wenn Stakeholder ihr Wissen sprachlich nicht adäquat beschreiben können. Außerdem ist das Machtverhältnis zwischen dem Analytiker und Stakeholder umgekehrt. Allerdings eignet sich diese Methode nicht für jeden Arbeitsbereich. Zusätzlich dazu ist dies sehr zeitaufwändig. [7, S. 152]

### 5.3.3 Artefaktbasierte Techniken

Systemarchäologie

Durch Dokumentationen wie Benutzerhandbücher oder technische Systembeschreibungen, Quelltexte, Verträge etc. können Informationen des Alt- oder Konkurrenzsystems abgeleitet werden. [7, S. 153–154]

Durch eine fokussierte Analyse können irrelevante Dokumente unberücksichtigt bleiben. Dies ist sehr zeitaufwendig und lohnt nicht bei einer Vielzahl an neuen Anforderungen, da diese grundlegend neu erfasst werden müssen. [7, S. 154]

Perspektivenbasiertes Lesen

Ähnlich wie in der Methode der Systemarchäologie, werden auch hier Informationen aus Dokumenten extrahiert. Dies geschieht aus mehreren Perspektiven bei einem Dokument. Folgende können verwendet werden [8, S. 332–333]:

- Realisier bzw. Entwickler
- Tester
- Architekt
- Auftraggeber
- Anwender

Die Vorteile sind dieselben wie bei der Technik der Systemarchäologie. Allerdings kann das Einnehmen von verschiedenen Rollen den Leser vor eine große Herausforderung stellen. [8, S. 332–333]

Wiederverwendung

Das Übernehmen bereits erarbeiteter Anforderungen von Altsystemen ist ebenfalls möglich. Dabei werden diese gesammelt, strukturiert und anschließend in die Anforderungsermittlung mit einbezogen. [7, S. 155]

Das stellt eine Kostenersparnis dar, da die Anforderungen bereits vorliegen. Im Gegensatz zu neuen Anforderungen kann das Prüfen und Korrigieren dieser (bei ausreichender Qualität) reduziert werden. Ein großer Nachteil dieser Methode ist die Möglichkeit, dass fehlerhafte Anforderungen übernommen werden. Dies kann analog mit der Software-Engineering-Methode des Prototypings verglichen werden. Dabei ist es möglich, dass unausgereifte bzw. fehlerhafte Codeteile mit in die neue Software einfließen. Oftmals reichen bereits ermittelte Anforderungen in ihrer Qualität nicht aus. [7, S. 155]

### 5.3.4 Kreativitätstechniken

Brainstorming

Eine Gruppe von fünf bis zehn Personen sammelt in einer vorgegebenen Zeit mithilfe eines Moderators Ideen, welche dieser wertungsfrei notiert. Jede Idee wird anschließend reflektiert und ggf. werden Maßnahmen zur Umsetzung abgeleitet. [7, S. 156–157]

Durch diese Methode werden viele Ideen in kurzer Zeit gesammelt. Die Teilnehmer können ihre Ideen gegenseitig weiterentwickeln. Dies setzt allerdings eine gute Gruppendynamik voraus. Ist diese nicht gegeben, so wird die Sammlung von Ideen schwierig. Es ist ebenfalls möglich, einen solchen Termin mit räumlicher Distanz durchzuführen, benötigt von Seiten des Moderators aber mehr Aufwand. [7, S. 157]

Brainstorming paradox

Diese Technik nimmt das klassische Brainstorming Thema und invertiert es. Dabei werden Ideen gesammelt, die das Projekt innerhalb kürzester Zeit scheitern lassen. Dadurch können Maßnahmen abgeleitet werden, welche den Eintritt dieser Ereignisse verhindern. [7, S. 156–157]

Die Vor- und Nachteile sind identisch mit denen eines normalen Brainstormings. Ergänzend dazu wird den Grundanforderungen des Kanon-Modells mehr Beachtung geschenkt, da die Perspektive der Stakeholder eine andere ist. Durch diesen Überblick werden mögliche Gefahren und Risiken eines Projektes ersichtlich. [7, S. 157]

Wechsel der Perspektiven

Wie der Name der Methode es bereits ausdrückt, werden in dieser die Perspektiven der Stakeholder gewechselt. Dabei gibt es folgende Sichtweisen [9, S. 89–90]:

- Objektiv und neutral → Fakten (weiß)
- Subjektive Meinung → Gefühle (rot)
- Negative Argumente → fachlich (schwarz)
- Positive Eigenschaften (gelb)
- Neue Ideen → Brainstormer (grün)
- Moderator → Organisator (blau)

Dadurch ist es möglich, festgefahrenen Stakeholder andere Perspektiven aufzuzeigen und eine andere Denkweise anzunehmen. Dies stellt introvertierte und unaufgeschlossene Stakeholder vor eine Herausforderung. [9, S. 90]

Analogietechnik (Bionik/Bisozation)

In dieser Methode wird zu dem eigentlichen Thema eine Analogie gesucht, welche den Teilnehmern vorgestellt wird. Dabei wird das eigentliche Problem bzw. die Frage auf eine andere Situation projiziert. Anschließend werden Ideen gesammelt, um das Problem oder die Frage zu lösen. Danach wird das eigentliche Problem vorgestellt und die Ideen, welche zuvor gesammelt wurden, auf das Problem transferiert. Dadurch können Maßnahmen abgleitet werden, welche auf den Ideen basieren. [9, S. 91–92]
Komplexe Problemstellungen und Zusammenhänge können dadurch verständlicher aufgezeigt werden und Lösungen aus einem anderen Umfeld transferiert werden. Allerdings ist diese Methode sehr zeitaufwendig und es ist möglich, dass aufgrund fehlerhafter Rücktransformation die Ergebnisse zu falschen oder unanwendbaren Lösungen führen. [9, S. 91–92]

## 5.4 Auswahl geeigneter Ermittlungstechniken

„Da im Laufe eines Vorhabens die unterschiedlichsten Arten von Anforderungen unter wechselnden Rahmenbedingungen aus unterschiedlichsten Quellen ermittelt werden, reicht eine einzelne Ermittlungstechnik nicht aus – ein Allheilmittel gibt es nicht.“ [7, S. 131] Deshalb werden nachfolgend mehrere Techniken genutzt, um die Anforderungen zu ermitteln. Da nicht jede Methode zur Gewinnung von Anforderungen für jeden Stakeholder bzw. für jede Situation geeignet ist, müssen zuerst die Stakeholder identifiziert werden. Anschließend können die verschiedenen Ermittlungstechniken hinsichtlich ihrer Nutzbarkeit bewertet werden. Anhand des Pflichtenheftes zur

Umsetzung der elektronischen Auslieferung wurde eine Abteilung identifiziert, mit welcher zwangsläufig eine Anforderungsermittlung durchgeführt werden muss, der interne Fuhrpark. Neben diesem zentralen Stakeholder gibt es noch weitere Abteilungen, welche von dem internen Fuhrpark als Stakeholder benannt wurden. Das Qualitätsmanagement ist bei erfal der erste Ansprechpartner, wenn Kunden Probleme mit Ihrer Lieferung des Fuhrparks haben. Von der Aushändigung des Zustellberichtes bis hin zur telefonischen Kontaktaufnahme mit dem Fahrer, welcher das Paket ausgeliefert hat, ist das Qualitätsmanagement der richtige Ansprechpartner. Zusätzlich dazu ist das Bestellcenter, welches die erste Anlaufstelle für Kunden bildet, ein weiterer Stakeholder. Um zu verhindern, dass gewisse Abteilungen für diese Anforderungsermittlung ausgeklammert werden, ist eine Rücksprache mit den verbleibenden Abteilungs- bzw. Teamleitern durchzuführen. Diese Befragung lieferte keine zusätzlichen Stakeholder.
Unter Zuhilfenahme der Matrix aus Anhang 1 SOPHIST Ermittlungstechnikenauswahlmatrix sind geeignete Ermittlungstechniken evaluiert worden. Hierbei wurden folgende Punkte betrachtet:

- Das Wissen des Stakeholders (besonders des Fuhrparks) ist groß.
- Die Schnittstelle zur Nutzung der elektronischen Warenauslieferung ist zwingend zu nutzen, da sonst keine Datenübertragung stattfinden kann.
- Die Stakeholder sind zeitlich flexibel und vor Ort tätig.
- Die Übergabe der Daten über die Schnittstelle und deren Verarbeitung kann nicht aktiv miterlebt werden.
- Der gesuchte Detailgrad der Informationen ist so fein wie möglich.

Aufgrund der oben genannten Rahmenbedingungen sind folgende Methoden ausgewählt worden:

- Einzelinterview
- Gruppeninterview
- Systemarchäologie
- Wiederverwendung von Anforderungen

## 5.5 Durchführung

### 5.5.1 Systemarchäologie

Damit das Grundverständnis des Altsystems und dessen Aufgaben gegeben ist, wurde diese Methode als Initialschritt durchgeführt. Das Projekt zur elektronischen

Auslieferung ist, wie eingangs erwähnt, seit 2016 im Einsatz. In dem Ticketsystem von erfal *Jira* sind mehr als 300 Tickets zu diesem Thema angelegt worden. Dabei sind verschiedene Fehler oder neue Anforderungen dokumentiert worden. Diese betreffen sowohl die App auf den mobilen Endgeräten, das GHT-Portal sowie die Schnittstelle von und zu erfal. Essenziell für diese Betrachtung sind ausschließlich die Tickets, welche die Schnittstelle von und zu erfal beinhalten. Dadurch konnte die zu betrachtende Ticketanzahl verringert werden. Die System- und Benutzerdokumentation wurden explizit mitbetrachtet, da diese ein grundlegendes Verständnis des Prozesses der Auslieferung enthalten. Dies sind nachfolgende Dokumente:

- ght_Benutzerdokumentation_4mobile OnTour Erfal.pdf
- ght_Pflichtenheft Servicflotte_erfal_V4.0.pdf
- ght_TechnischeDokumentation_erfal.pdf

Des Weiteren konnten Abbildungen für die aktuelle Datenstruktur gefunden werden. Durch diese ist ein erster Anhaltspunkt gegeben, um die verwendeten Datenquellen zu identifizieren. Die Betrachtung der Datenhaltung und deren Beschaffung sind separat in Kapitel 6 aufgeschlüsselt.
Durch die Verwendung eines Versionsverwaltungssystems ist der gesamte Quelltext, inklusive deren Revisionen, verfügbar. Damit ist ein Einblick in das Programm und dessen Funktionalitäten möglich.
Alle erwähnten Dokumente bzw. Daten sind auf der beigefügten CD bzw. dem Archiv vorhanden.

### 5.5.2 Einzelinterview

Es wurde mit allen Stakeholdern ein Einzelinterview durchgeführt. Dabei ist vorrangig der Fuhrpark der Stakeholder, welcher die Funktionalitäten der Schnittstelle vorgibt. Dies ist darauf zurückzuführen, dass andere Abteilungen ausschließlich den Zustell- bzw. Sendungsbericht nutzen. Aufgrund dessen ist das Interview mit dem Fuhrpark ausführlicher, als jene, die mit dem Bestellcenter oder dem Qualitätsmanagement geführt worden sind.
Nachfolgend werden einzelne Punkte aus den Interviews wiedergegeben, die vollständig aufbereiteten Protokolle dieser befinden sich in den Anhängen 3 bis 5.

Fuhrpark

Zu Beginn des Interviews wurde der generelle Ablauf des Prozesses der Auslieferung erläutert. Dadurch war es möglich, einen tiefgreifenden Einblick in die Arbeitsweise der Mitarbeiter zu bekommen. Innerhalb dieser Prozessbeschreibung sind zeitliche Rahmenparameter aufgekommen, welche den Import der Daten von GHT aber auch den Export der Daten zu GHT definieren. Diese Rahmenbedingungen gilt es beizubehalten.

Durch die Reimplementierung der GHT-Schnittstelle ergeben sich die grundlegenden Anforderungen, dass die Funktionalitäten der bisherigen Schnittstelle beibehalten werden sollen. Dies bildet die Grundlage für die weitere Arbeit.

Für die Datenübertragung zu GHT wird gefordert, dass geplante Rückholungen von Waren übertragen werden. Diese werden bereits im erfal-Produktionssystem ePS gepflegt.

Bestellcenter

Das Design des Sendungsberichtes muss überarbeitet und mit einem erfal-Branding versehen werden. Des Weiteren müssen alle Informationen des Berichtes auf dem neuen vorhanden sein.

Qualitätsmanagement

Analog zu dem Interview mit dem Bestellcenter bestehen hierbei die gleichen Anforderungen an den Sendungsbericht. Zusätzlich dazu besteht die Anforderung einer organisatorischen Richtlinie, welche die Fahrer anweist, den Mechanismus des Scannens der Pakete nicht zu umgehen. Diese Richtlinie ist kein Bestandteil dieser Arbeit und wird von einer anderen Abteilung angefertigt. Diese Richtlinie ist ebenfalls im Anhang 6 Arbeitsanweisung elektronische Auslieferung zu finden.

### 5.5.3 Gruppeninterview

Das Gruppeninterview mit allen Stakeholdern diente der Überprüfung der allgemeinen Anforderungen und zur Abstimmung des Auslieferungsprozesses. Dabei wurde der grundlegende Ablauf der Zustellung beleuchtet und anschließend detailliert auf die Probleme dieses Prozesses eingegangen wie z.B. das Umgehen der Maßnahmen, dass alle Pakete bei Auslieferung gescannt werden müssen. Durch die Tatsache, dass dieses Problem organisatorisch geregelt werden muss, wird das Qualitätsmanagement eine Arbeitsanweisung verfassen. Diese ist von dem Fuhrpark

an die Fahrer weiterzureichen. Anschließend wird ein Unterweisungsnachweis der Fahrer gefordert, wodurch die Kenntnisnahme der Richtlinie schriftlich festgehalten wird.

### 5.5.4 Wiederverwendung von Anforderungen

Mittels des Ticketsystems ist es möglich gewesen, einige Anforderungen aus technischer Sicht wiederzuverwenden, welche von keinem Stakeholder direkt beschrieben worden sind. Als konkretes Beispiel ist hier die Tournummer zu erwähnen. Diese ist in den Systemen von erfal alphanumerisch. Das GHT-Portal akzeptiert ausschließlich numerische Tournummern, wodurch eine Konvertierung notwendig ist.

## 5.6 Anforderungen

### 5.6.1 Exportschnittstelle

Aus den Interviews und anderen Methoden sind folgende neue Anforderungen hervorgegangen:

Um eine unmissverständliche natürlichsprachliche Dokumentation der Anforderungen zu erreichen, sind nachfolgend Anforderungsschablonen eingesetzt worden. [7, S. 357–388]

Im weiteren Verlauf wird die Schnittstelle, welche Daten von erfal zu GHT überträgt, als Exportschnittstelle benannt. Im Gegensatz dazu wird, wie bereits in Punkt 2.3 erwähnt, diese Schnittstelle im Altsystem als Importschnittstelle bezeichnet.

<u>Funktionale Anforderungen:</u>

1. Die Exportschnittstelle muss fähig sein, geplante Rückholungen an das GHT Portal zu übertragen.
2. Die Exportschnittstelle muss fähig sein, den hinterlegten Ablageort des Kunden an das GHT Portal zu übertragen.
3. Die Exportschnittstelle muss fähig sein, den hinterlegten Ansprechpartner des Kunden an das GHT Portal zu übertragen.
4. Die Exportschnittstelle muss fähig sein, die hinterlegte Telefonnummer des Kunden an das GHT Portal zu übertragen.
5. Die Exportschnittstelle muss fähig sein, die hinterlegten Öffnungszeiten des Geschäftes des Kunden an das GHT Portal zu übertragen.

6. Die Exportschnittstelle muss fähig sein, die alphanumerische Tournummer von erfal in eine numerische GHT-Tournummer zu konvertieren.
7. Die Exportschnittstelle muss nicht fähig sein, die Daten des Informationsfeldes zu übertragen.

Nicht funktionale Anforderungen

1. Innerhalb des Zeitraumes von 16:30 Uhr bis 22:30 Uhr muss die Exportschnittstelle alle zwei Stunden die Daten auf den FTP Server des GHT-Portals exportieren.
2. Am Folgetag muss die Exportschnittstelle die Daten um 4 Uhr einmalig übertragen.

Wie im Abschnitt 2.2 bereits erläutert, wird bei einer größeren Anzahl an Paketen das Scannen jedes einzelnen Paketes umgangen, indem jedes Paket als unzustellbar deklariert wird und anschließend die gesamte Sendung auf zugestellt gesetzt wird. Dies kann die Exportschnittstelle technisch nicht verhindern. Aufgrund dessen hat das Qualitätsmanagement von erfal die Aufgabe bekommen, eine Arbeitsanweisung zu erstellen, welche die Fahrer zwingend zu befolgen haben. Dies stellt eine organisatorische Richtline dar und hat damit nicht die gleiche Effektivität wie eine technische.

### 5.6.2 Importschnittstelle

Im Nachfolgenden wird die Schnittstelle, welche Daten von GHT zu erfal importiert, als Importschnittstelle bezeichnet. Im Gegensatz dazu wird, wie bereits in Punkt 2.3 erwähnt, diese Schnittstelle im Altsystem als Exportschnittstelle benannt.

Funktionale Anforderungen

1. Die Importschnittstelle muss fähig sein, die Sendungsdaten von GHT in das erfal Produktionssystem zu importieren.
2. Die Importschnittstelle muss fähig sein, ungeplante Rückholungen in das erfal Produktionssystem zu importieren.
3. Die Importschnittstelle muss fähig sein, die Stechzeiten der Mitarbeiter in eine CSV-Datei zu schreiben.

Nicht funktionale Anforderungen

1. Die Importschnittstelle muss alle zwei Minuten die zur Verfügung stehenden Sendungsdaten importieren.

Ergänzungen:

Eine Minimierung der Datenhaltung ist aufgrund eines redundanten Speicherungssystems notwendig.

Zusätzlich dazu muss der Sendungsbericht ein neues Design erhalten. Der Sendungsbericht muss ein erfal-spezifisches Design inklusive des Logos der Firma erhalten. Außerdem muss dieser Sendungsbericht alle Daten des bisherigen Berichtes enthalten. Die Ausarbeitung dieses Designs übernimmt die Marketing-Abteilung der Firma erfal.

# 6 Datenhaltung und Datenbeschaffung

## 6.1 Analyse des Altsystems

### 6.1.1 Importschnittstelle

Das Altsystem nutzt für die Datenübertragung zu GHT drei verschiedene Datenquellen als Grundlage. Aus diesen werden die jeweiligen Informationen einzeln abgerufen und anschließend, in der Software selbst, zusammengeführt. Zusätzlich dazu werden bei Datenbankabfragen über mehrere Tabellen keine Joins genutzt. Das dabei entstandene kartesische Produkt, auch Kreuzprodukt genannt, beinhaltet dabei alle möglichen Kombinationen der verwendeten Tabellen. Das bedeutet, es entstehen auch Ergebnisse, welche semantisch nicht zueinander passen. Zum Beispiel wird ein Auftrag X mit einem Paket Y verknüpft, welches allerdings zu Auftrag Z gehört. Um diese semantisch fehlerhaften Datensätze auszuschließen, erfolgt eine Einschränkung dieser über die WHERE Klausel in einer SQL-Abfrage.

Nachdem die XML-Dateien generiert und übertragen worden sind, werden diese in eine Datenbank geschrieben. Die nachfolgende Abbildung 7 stellt die Datenstruktur bildhaft dar. Der gelbe Schlüssel symbolisiert den Primary Key einer Tabelle. Die orangenen Rauten stellen eine Fremdschlüsselbeziehung (Foreign Key) dar und die blauen Rauten stellen einfache Entitäten, Werte, dar.

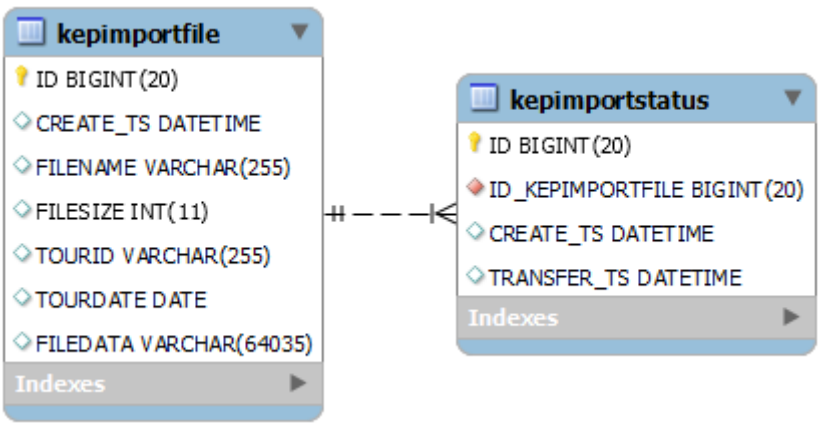



**Abbildung 7 KepImport ERD**

Die verwendeten Begriffe in dem Altsystem sind, wie bereits in Punkt 2.3 erwähnt, aus Sicht des GHT-Portals. Aufgrund dessen wird die Datenübertragung von erfal zu GHT als Import bezeichnet. Im nachfolgenden Punkt ist diese Schnittstelle als Exportschnittstelle gekennzeichnet.

Auf Basis der im Anhang 2 ghtKepImportAbholung dargestellten grafischen Dokumentation (aus einem Ticket) sind die verschiedenen Datenquellen hervorgegangen. In nachfolgender Tabelle ist kompakt aufgeschlüsselt, woher das Altprogramm welche Daten bezieht.

| Genutzte Datenquelle | Datenkategorie |
|---|---|
| **Erfal Produktionssystem** | Auftragsdaten (wie z.B. Auftragsnummer, Absender- und Lieferadresse) |
| | Paketdaten (wie z.B. Paketnummer, Abmessungen und Lagerorte der Pakete) |
| | Adressdaten ohne Geokoordinaten |
| | |
| **Smartour Tourenoptimierung** | Adressdaten inkl. der Geocodierung |
| | |
| **ERP Microsoft Dynamics AX** | Dauer einer Tour |
| | Adressdaten ohne Geokoordinaten |

**Tabelle 2 Datenquellen Export**

Anhand der Tabelle 2 ist ersichtlich, dass ein Großteil der genutzten Daten bereits im Produktionssystem von erfal vorhanden sind. Die restlichen benötigten Daten werden aus dem ERP-System und der Tourplanungs-Software abgerufen.

### 6.1.2 Exportschnittstelle

Die Sendungsdaten, welche von GHT bereitgestellt werden, werden in dreifacher Form redundant gespeichert. Diese werden als XML-String in einer Datenbank gesichert und es wird ein Sendungsbericht generiert, der anschließend in das Dokumentenmanagement-System von erfal gespeichert wird. Der Inhalt ist der gleiche, allerdings werden die Daten in dem Sendungsbericht grafisch aufgearbeitet und für das menschliche Auge besser dargestellt. Zusätzlich dazu werden die Originalen XML-Daten auf dem Server in einem separaten Verzeichnis gespeichert. Nachfolgend ist die Datenstruktur der zugehörigen Tabellen grafisch dargestellt.

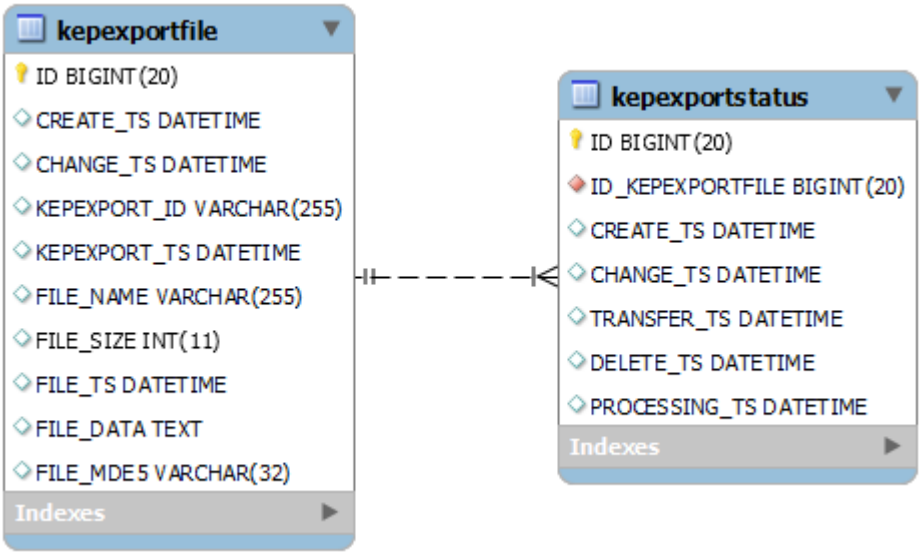



**Abbildung 8 KepExport ERD**

Das Altsystem wurde in dem Kontext des Dienstleisters GHT entwickelt. Dadurch ist die Datenübertragung von GHT zu erfal, wie zuvor in Punkt 2.3 genannt, als Export benannt.

## 6.2 Planung der neuen Datenstruktur

### 6.2.1 Exportschnittstelle

Um eine einheitliche Datenhaltung zu gewährleisten, werden Schnittstellen geschaffen, um die restlichen Informationen in das Produktionsleitsystem zu integrieren. Eine Integration von mehreren Datenquellen wäre ebenfalls möglich, wird aber aufgrund von Flexibilitätsgründen vernachlässigt.

Für die Übergabe von Stammdaten aus dem ERP-System zu dem Produktionssystem existiert bereits eine Schnittstelle. Diese wird um die Übergabe der Touren-Stammdaten erweitert.

Ebenso ist die Tourplanungs-Software Smartour bereits an das Produktionssystem angebunden. Hierbei wird die bestehende Schnittstelle um die Übergabe der geokodierten Adressdaten erweitert.

Da diese Schnittstellen von anderen IT-Mitarbeitern der Firma erfal programmiert worden sind, wird nachfolgend nicht näher auf diese und deren Erweiterung eingegangen.

### 6.2.2 Importschnittstelle

Die Zustellungsdaten werden in das Produktionssystem von erfal übertragen. Dadurch kann eine Verbindung zu den Aufträgen hergestellt werden und falls notwendig der Sendungsbericht generiert werden.

Wie bereits im Punkt 6.1.2 erwähnt, speichert das Altsystem die Daten redundant ab. Hierbei bestand die Vermutung, dass dies datenschutzrechtliche Vorgaben begründete. Nach Rücksprache mit dem internen Compliance-Beauftragten der Firma gibt es aus datenschutzrechtlicher Perspektive keine Notwendigkeit, diese Daten redundant abzuspeichern. Hier gilt der Grundsatz der Datenminimierung, welcher in Artikel 5 Absatz 1 Satz 1c der Datenschutzgrundverordnung DSGVO verankert ist. Dadurch werden die Daten einmalig in der Datenbank gespeichert und die originalen Daten von dem GHT-Portal nach einer Karenzzeit gelöscht. Nachfolgend ist die Datenstruktur in einem Entity Relationship Diagramm dargestellt.

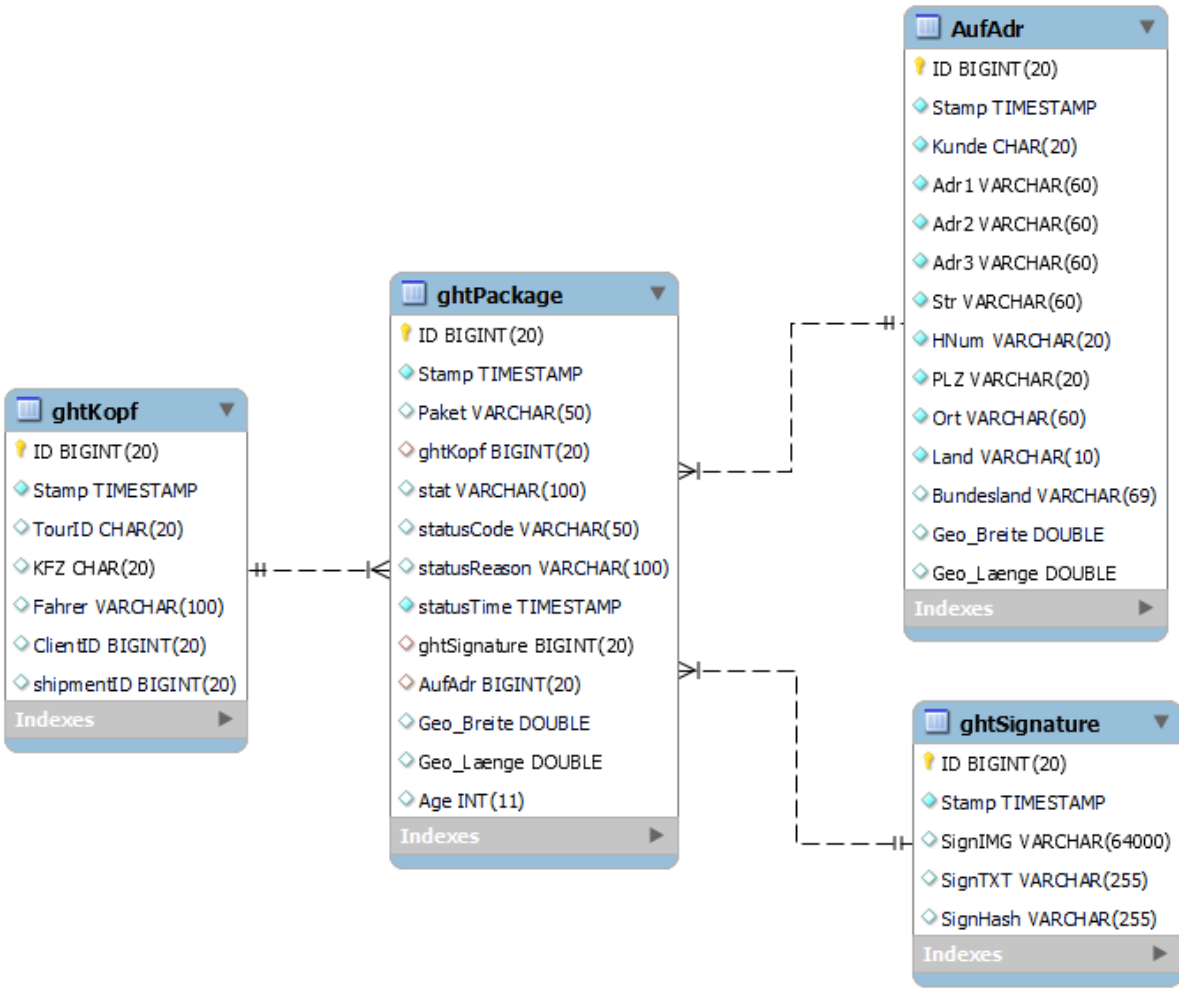



**Abbildung 9 ERD**

Auf Basis des bisher eingesetzten Datenbankmanagementsystems MySQL von Oracle ist die maximale Größe eines Datensatzes auf 64 Kilobytes (KB) beschränkt.

Aufgrund dessen und der Normalisierung wurde die Signatur in eine separate Tabelle ausgelagert.

Die Versionierung der Daten ist im gesamten System mit einem zusätzlichen Feld gelöst, welches die Aktualität des Datensatzes widerspiegelt. Hierbei ist der Wert „0“ der aktuelle. Bevor ein Datensatz geschrieben wird, werden die zutreffenden Einträge um eins erhöht.

# 7 Softwarearchitektur

## 7.1 Auswahl einer geeigneten Migrationsstrategie und Modell

Wie bereits in dem Punkt 2.3 erwähnt, fehlt in dem Unternehmen die fachliche Expertise für diese Software bzw. Programmiersprache. Aufgrund dessen und der Vorgabe, dass diese Schnittstelle in das Produktionsleitsystem der Firma erfal integriert werden soll, scheiden die Strategien der Kapselung und Konversion aus. Außerdem ist, wie in Punkt 6.1 erläutert, die Datenstruktur des Altsystems ungewöhnlich, sodass deren Konvertierung einen erhöhten Aufwand darstellen würde. Eine Verknüpfung mit anderen Systemen besteht nicht, sodass angrenzende Systeme nicht angepasst werden müssen. Die bisherige Programmstruktur beinhaltet keine Benutzeroberfläche, somit ist in diesem Fall eine Kapselung nicht zielführend. Dagegen stellt die Konversion eine konkrete Transformation der Funktionalitäten und der Daten dar. Allerdings ist die Struktur der Software und deren Komplexität nicht optimal ausgerichtet, sodass auch hier Anpassungen vorgenommen werden müssen.

Nachfolgend wird die Migration auf Basis der Reimplementierung aufgebaut. Hierbei ist konkret die Integration der Schnittstellen in das Produktionssystem ePS und deren Programmiersprache PHP als Ziel gesetzt.

Ein Ausfall bzw. ein Fehler der Schnittstellen zur Übertragung der Daten an das GHT-Portal ist als kritisch anzusehen. Die Fahrer haben hierbei das Problem, dass bei einem Ausfall der Schnittstelle, mit Ladelisten gearbeitet werden muss, was den Auslieferungsprozess unnötig kompliziert macht. Des Weiteren ist es bei einem Ausfall nicht möglich, den Zustellort bzw. einen Zustellnachweis anzufertigen, da dieser ausschließlich über das MDE-Gerät erfolgen kann. Dagegen ist im Vergleich ein Ausfall der Schnittstelle, welche die Daten des GHT-Portals importiert, als weniger problematisch anzusehen. Hierbei besteht die Möglichkeit, mit dem Portal von GHT zu arbeiten (wie bereits in Punkt 2.3 beschrieben).

Durch diesen Umstand eignet sich der Cold-Turkey-Ansatz für diese Migration nicht. Durch den großen „Big-Bang“ besteht die Möglichkeit, dass das System zur elektronischen Auslieferung komplett zum Erliegen kommt. Aufgrund dessen und der Besonderheit, dass die Schnittstellen (Import und Export der Daten) zwei voneinander getrennte Programme sind, wird der Chicken-Little-Ansatz als geeignetes Modell ausgewählt.

Durch die Softwarearchitektur des Legacy-Systems ist der Schritt 2 *Teil-Zerlegung des Altsystems* nicht notwendig. Aufgrund dessen, dass das Altsystem keine

Benutzeroberfläche besitzt, entfällt der Schritt 4 *Teil-Entwurf der Applikation des Zielsystems* und Schritt 9 *Teil-Migration der alten Applikation*. Des Weiteren ist die *Teil-Entwicklung und Einrichtung der Gateways* (Schritt 7) nicht notwendig, da die Originaldaten im Rohformat vorliegen. Diese werden initial vor der Inbetriebnahme der Schnittstelle importiert. Dadurch entfällt ebenfalls der Schritt 8 die *Teil-Migration der alten Datenbank*. In Punkt 3.4.2 sind die Schritte des Ansatzes vollständig aufgeführt. Die *Teil-Analyse des Altsystems*, Schritt 1, ist im vorherigen Kapitel 6 auf Basis der Datenstrukturen geschehen. Zusätzlich dazu ist in diesem Kapitel der Schritt 5 *Teil-Entwurf der Datenbank* bzw. der Struktur des Zielsystems abgearbeitet. Nachfolgend werden die verbleibenden Schritte 3 *Teil-Entwurf der Schnittstellen des Zielsystems*, 6 *Teil-Implementierung des Zielsystems*, 10 *Teil-Migration der alten Schnittstellen* sowie Schritt 11 *Teil-Umstellung auf das neue Zielsystem* der Reihenfolge nach, für die jeweilige Schnittstelle, durchgeführt.
Wie bereits in Punkt 2.3 erwähnt, ist die Generierung des Sendungsberichtes seit geraumer Zeit gestört. Aufgrund dessen wird zuerst die Schnittstelle migriert, welche die Daten des GHT-Portals verarbeitet, den Sendungsbericht generiert und anschließend in das Dokumentenmanagementsystem speichert. Anschließend wird die Schnittstelle migriert, welche die Daten von erfal zu GHT überträgt.

## 7.2 Auswahl geeigneter Refactorings

Wie bereits in Kapitel 4 beschrieben, ist der Katalog der Refactorings umfangreich. Aufgrund dessen wurden nur ausgewählte Refactorings in Punkt 4.2 vorgestellt. Die Auswahl von brauchbaren Refactorings richtet sich stark nach dem persönlichen Programmierstil des Entwicklers bzw. nach dessen Systemumgebung. Dabei werden nachfolgend die Methoden *Funktion extrahieren* und *Variable extrahieren* genutzt, um die Übersichtlichkeit des Quelltextes zu erhöhen.

## 7.3 Strukturbeschreibung

### 7.3.1 Importschnittstelle

In nachfolgender Abbildung ist der grundlegende Ablauf der Importschnittstelle dargestellt. Diese hat als zentrale Aufgabe, die bereitgestellten Daten des GHT-Portals in das erfal Produktionssystem zu importieren.

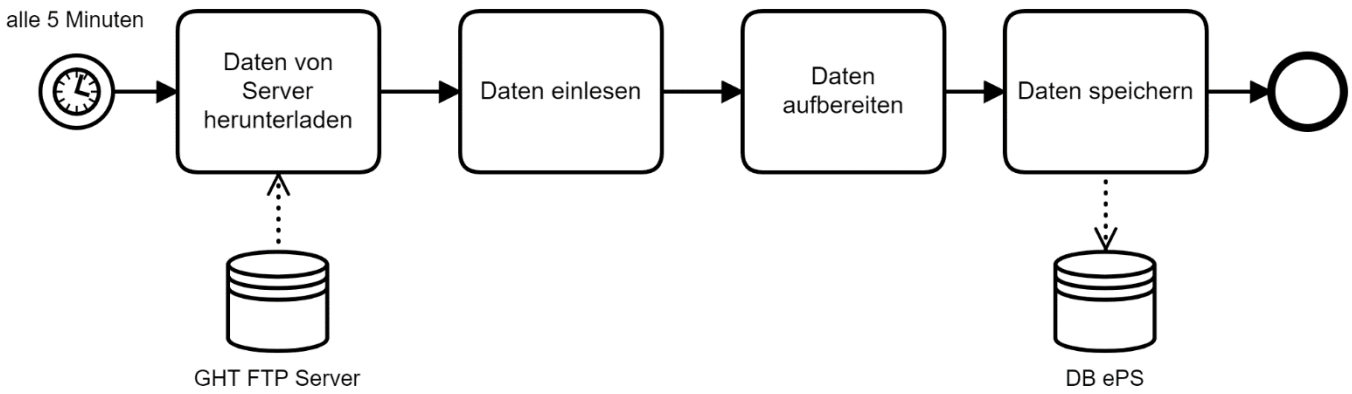



**Abbildung 10 BMPN-Diagramm Importschnittstelle**

Bei den bereitgestellten Daten von dem GHT-Portal handelt es sich um zyklisch exportierte XML-Daten. Diese werden von dem FTP-Server (ein Server der Daten bereitstellt) zunächst in ein temporäres Verzeichnis gespeichert, welche nach einer Karenzzeit (eine Woche) automatisch entfernt werden. Anschließend werden die Daten eingelesen, wobei darauf geachtet werden muss, dass diese Daten redundanzfrei sind. Das bedeutet, es dürfen keine doppelten Informationen gespeichert werden. Als exemplarisches Beispiel ist hier die Signatur anzusehen. Diese wird bei jedem einzelnen Paket mitübergeben, sodass bei einer Sendung mit 14 Paketen, die Signatur ebenfalls 14-mal vorhanden ist. Der Vorteil hierbei ist nicht nur der geringere Speicherplatzverbrauch, sondern auch der logische Zusammenhang. Bei einer Sendung von 14 Paketen unterschreibt der Empfänger einmal für alle Waren und nicht für jedes Paket separat. Auf Basis, der in Punkt 6.2.2 entworfenen Datenstruktur, werden die Daten anschließend in die Datenbank des Produktionssystems ePS importiert.

### 7.3.2 Exportschnittstelle

In der nachfolgenden Abbildung ist der grundlegende Ablauf der Export-Schnittstelle in einem BPMN-Diagramm dargestellt. Diese Schnittstelle hat die Aufgabe, die auszuliefernden Pakete inklusive der Adressdaten an das GHT-Portal zu übertragen.

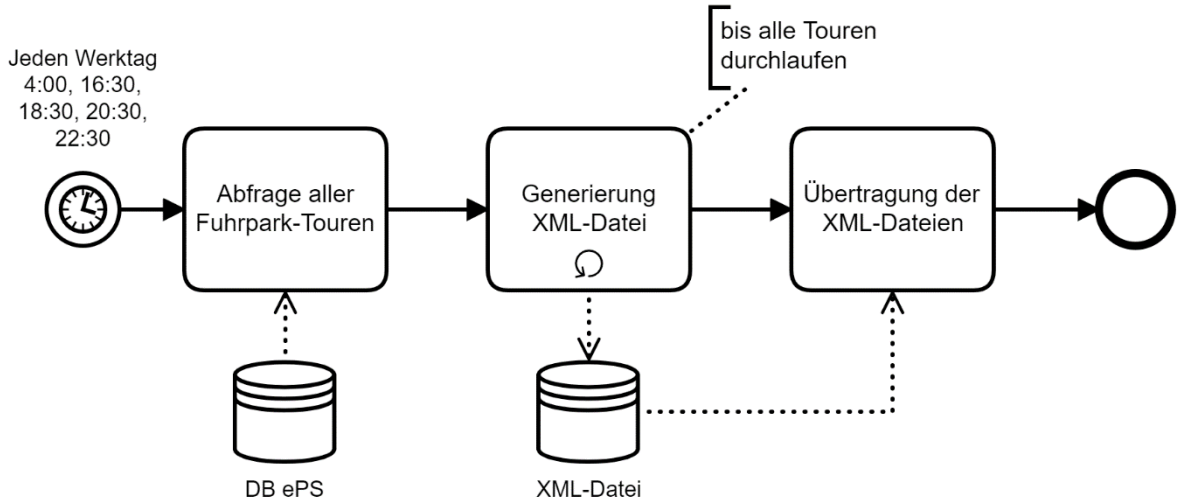



**Abbildung 11 BMPN-Diagramm Exportschnittstelle**

Es gibt Touren, auf welchen das elektronische Auslieferungssystem noch nicht genutzt wird. Dies hat mehrere Gründe, wie z.B. das Fehlen von mobilen Endgeräten oder dass mit dieser Tour ausschließlich ein Großkunde beliefert wird. Aufgrund dessen werden im ersten Schritt alle Touren abgefragt, welche das System zur elektronischen Warenauslieferung benutzen. Anschließend wird in einer Schleife für jede Tour die auszuliefernden Pakete inklusive der Adressdaten des Senders und Empfängers ausgelesen. Durch das vorgegebene XML-Format werden die Ergebnisse in diesem Format als Datei gespeichert. Nachdem alle Dateien generiert worden sind, werden diese gebündelt an das GHT-Portal übertragen.

# 8 Implementierung

## 8.1 Import-Schnittstelle

Wie in Punkt 7.1 erwähnt, ist die Generierung des Sendungsberichtes seit geraumer Zeit gestört. Aufgrund dessen und der im zuvor genannten Punkt erläuterten Gründe wurde diese Schnittstelle zuerst implementiert und anschließend migriert.

Eine Migration der alten Daten wurde, wie bereits in Punkt 7.1 erwähnt, aufgrund der unterschiedlichen Datenstrukturen nicht durchgeführt. Des Weiteren liegen alle Daten im Rohrformat vor. Dadurch ist es möglich, einen Import der alten Daten über die neue Schnittstelle durchzuführen. Nach Rücksprache mit den Mitarbeitern bei erfal (Fuhrpark und Qualitätsmanagement) sind die Daten des letzten Jahres zusätzlich zu importieren. Die Sendungsbelege, welche von dem Legacy-System in dem Dokumentenmanagement System von erfal gespeichert wurden, sind weiterhin verfügbar. Zum Zeitpunkt des Importes beläuft sich die Anzahl der archivierten Dateien (von dem 01.01.2021 bis zum Zeitpunkt des Importes) auf 40.000 Dateien, welche insgesamt 25 GB Speicherplatz belegen. Nach dem Import der neuen Schnittstelle beläuft sich der verwendete Speicherplatz der Datenbanktabellen inklusive der Indexe auf rund ein Gigabyte. Damit wurde eine Komprimierungsrate von 25 erreicht, was insgesamt eine Platzersparnis von 96% erbringt. Ein Vergleich mit der alten Datenstruktur ist nicht möglich, da in der Datenbank Daten ab 2019 enthalten sind. Die Gesamtgröße der Tabelle beläuft sich auf 38 GB.

## 8.2 Export-Schnittstelle

Nach der Implementierung der Importschnittstelle folgt die Exportschnittstelle. Diese ist dafür zuständig, alle auszuliefernden Pakete inklusive der geforderten Daten, wie z.B. Adressdaten, an das GHT-Portal zu übertragen.

Durch die Nutzung einer Datenquelle stellte die Abfrage der Daten über mehrere Tabellen kein Problem dar. In der Datenbank des Produktivsystems ePS waren bis auf wenige Ausnahmen alle benötigten Informationen vorhanden. Wie bereits in 6.2.1 erwähnt, wurden die fehlenden Daten über Anpassungen von Schnittstellen in das erfal Produktionssystem importiert. Nachdem diese Daten vorhanden waren, konnte die Entwicklung der Schnittstelle vonstattengehen. Zuerst wurden die benötigten Daten aus der Datenbank abgerufen. Anschließend erfolgte die Konvertierung in eine programminterne Array Struktur, welche nachfolgend für die Umwandlung in das XML-Format genutzt wurde. Nachdem alle Daten generiert worden sind, werden die Daten gebündelt auf den FTP Server von GHT hochgeladen.

Ein Vergleich der Performance der Exportschnittstelle, auf Basis von 700 Logdateien des Legacy-Systems, ergab, dass das Altsystem im Durchschnitt 315 Sekunden benötigte, um die Dateien zu generieren. Im direkten Vergleich benötigte das neue System im Durchschnitt 2,9 Sekunden. Das bedeutet, das Programm ist um den Faktor 108 schneller.

## 8.3 Qualitätssicherung

Durch das bereitgestellte Testsystem seitens GHT ist es möglich, die neuen Schnittstellen auf deren Funktionalität zu testen. Hierbei wurden sowohl Live-Daten aus dem Produktionssystem als auch aus dem erfal eigenen Testsystem genutzt.
Die Importschnittstelle wurde mittels archivierter Dateien getestet. Die Dateien wurden in das Test-Verzeichnis des FTP-Servers von GHT hochgeladen. Damit wurde die Generierung der XML-Dateien, seitens des GHT-Portals, simuliert. Der anschließende Vergleich der Ergebnisse der neuen und alten Schnittstelle konnte nicht auf Datenbankebene durchgeführt werden. Die Datenstrukturen der Schnittstellen sind grundlegend verschieden, wodurch ein Vergleich dieser mit einem hohen manuellen Aufwand verbunden wäre. Die Entwicklung eines automatisierten Abgleichs beider Datenstrukturen wurde aufgrund von Komplexitätsgründen vernachlässigt. Die detaillierten Datenstrukturen sind in Punkt 6.1.2 und 6.2.2 zu finden. Aufgrund dessen werden die Sendungsberichte miteinander verglichen. Hierbei wurde darauf geachtet, dass die zur Verfügung gestellten Informationen aus der XML-Datei korrekt übernommen worden sind.
Die Exportschnittstelle generiert XML-Dateien, welche an das GHT-Portal übertragen werden. Durch das einheitliche Datenformat und deren Struktur bietet sich eine automatisierte Auswertung an. Im Gegensatz dazu ist der manuelle Vergleich von zwei XML-Dateien mit keinerlei Entwicklungsaufwand verbunden. Der Aufwand, diese Dateien händisch miteinander zu vergleichen, scheint gegenüber der Entwicklung eines Programmes zunächst als gering. Desto öfter ein Vergleich stattfindet, umso mehr invertiert sich dieses Verhältnis. Damit eine verlässliche Grundaussage getroffen werden kann, ob die neue Schnittstelle verlässlich funktioniert, ist ein Vergleich mit einer hohen Anzahl von Dateien sinnvoll. Aufgrund dessen wurde ein Programm zum Vergleich von zwei XML-Dateien entwickelt. Mit dieser Software ist es möglich, Unterschiede bei den generierten Daten zu finden.

## 8.4 Aufgetretene Probleme und Lösungen

Die Komplexität des Legacy-Programmes ist aufgrund hochdynamischer Klassen und Methoden sehr hoch. Die Einarbeitung in das Programm und das Herausfinden wie Daten miteinander verknüpft worden sind, ist nur nach aufwendigen Gesprächen mit verschiedenen Stakeholdern möglich gewesen. Die Vereinheitlichung der Datenhaltung stellte sich ebenfalls als eine Herausforderung dar, da es rechtliche Aspekte zu betrachten gab. Dies wurde zusätzlich durch die fehlende Dokumentation erschwert. Außerdem wurden einige Daten aus Systemen abgerufen, welche nicht gepflegt oder nur schwer zugänglich waren.

Die Kommunikation mit dem externen Dienstleister GHT stellte sich als langwierig dar. Anfragen wurden stark zeitverzögert beantwortet bzw. musste hierbei mehrfach nachgefragt werden. Eine Antwort auf die erste Anfrage bezüglich der Schnittstellen und deren Funktionalität wurde nach sechs Wochen und sechs Tagen beantwortet. Hierbei erfolgte noch keine direkte Beantwortung der Fragen, sondern lediglich eine Erklärung der langen Antwortzeiten. Ein Ausschnitt der E-Mail-Kommunikation mit GHT folgt: „…Bitte entschuldigen Sie, dass die Bearbeitungszeit aktuell deutlich länger ist als gewöhnlich. Wir sind gerade dabei, die AP und Beratungskompetenzen hinsichtlich des damaligen Projektes mit Ihrem Hause neu aufzustellen und zu strukturieren, sind die ursprünglichen Projektleiter bei allen beteiligten Partnern nicht mehr zugegen bzw. wurden neu besetzt. …“. Die Fragen wurden nach einer weiteren Woche teilweise beantwortet. Weitere Antworten stehen noch aus.

Zusätzlich dazu konnten einige geforderte Funktionalitäten aus Punkt 5.5, wie z.B. die Übertragung der Öffnungszeiten und des Abstellortes, nicht produktiv umgesetzt werden. Begründet wurde dies dadurch, dass wie oben beschrieben, die Kompetenz des externen Dienstleisters erneut aufgebaut werden muss. Die geforderten Funktionalitäten der GHT-Schnittstellen sind bereits im Testsystem implementiert, diese jedoch in das Produktivsystem zu überführen, ist nicht ohne weiteres möglich. Nachfolgend ein weiterer Auszug aus der Kommunikation mit dem Dienstleister GHT: „… Da Produktiv- und Dev-Instanzen aktuell unterschiedliche Datenbankversionen sowie auch unterschiedliche Schnittstellenimplementierungen durchgeführt wurden, muss hier eine genauere Betrachtung durchgeführt werden. Das letzte Update für die Dev-Instanz war im März 2019. Erste Eindrücke ergeben jedoch, dass gravierende Unterschiede da sind und diese einen erheblichen Aufwand fordern. …“. Deshalb könnten die geforderten Funktionalitäten zwar implementiert, jedoch nicht getestet

werden. Ein produktiver Einsatz ist erst nach der Umsetzung seitens GHT und einer intensiven Testreihe möglich.
Für die Auslieferung von mehreren Aufträgen an einen Kunden nahm das Altsystem eine Aggregierung vor. Diese sah vor, dass, falls ein Kunde mehrere Aufträge hat, diese unter einer Sendungsnummer zusammengefasst werden müssen. Dies war an keiner Stelle dokumentiert, sodass der Unterschied erst bei dem Vergleich der XML-Dateien des Altsystems gegenüber den Dateien des neuen Systems aufkam. Eine Anpassung des neuen Systems war ohne Probleme möglich. Wäre diese Besonderheit nicht aufgefallen, so hätte der Fahrer bei der Auslieferung mehrere Sendungen für einen Kunden gehabt. Dies wäre kein kritisches Problem gewesen, hätte allerdings die Auslieferung für den Fahrer aufwändiger gemacht.

# 9 Fazit und Ausblick

Das Ziel dieser Arbeit war die Entwicklung einer Strategie für die Migration und Erweiterung eines Systems zur Warenauslieferung. Die Erarbeitung und Umsetzung der Strategie zu einer Migration des Legacy-Systems in eine neue Umgebung wurde erfolgreich umgesetzt. Das neue System arbeitet korrekt und fehlerfrei, welches in den oben beschriebenen Tests nachgewiesen wurde. Die Rückmeldungen zu der Migration von dem Auftraggeber sind durchweg positiv. Aufgrund von externen Abhängigkeiten mit dem zuständigen Dienstleister konnte eine produktive Inbetriebnahme der geforderten Erweiterungen noch nicht umgesetzt werden. Hierbei wird auf die Zuarbeit der Firma GHT gewartet, welche den Aufwand und somit die Kosten abschätzen muss. Demzufolge fehlt die Freigabe für das Projektbudget. Die Umsetzung der neuen Funktionalitäten, siehe Punkt 5.6, seitens GHT ist für Mitte bzw. Ende des vierten Quartals diesen Jahres, seitens erfal, geplant.

Die Schnittstelle zur Unterstützung der elektronischen Auslieferung wird erfal die nächsten Jahre weiterhin begleiten. Die Schnittstelle wird bei Bedarf erweitert werden, um neue Funktionalitäten für die Fahrer zu hinterlegen. Dabei ist eine enge Abstimmung mit dem Fuhrpark bzw. dessen Nutzern wichtig. Die geforderten neuen Funktionalitäten werden in enger Abstimmung mit GHT in naher Zukunft entwickelt und anschließend getestet. Zusätzlich dazu muss die App auf den mobilen Endgeräten weiterentwickelt werden, da diese auf neueren Android-Versionen nicht einwandfrei funktionieren bzw. benutzbar ist. Deshalb wird in der Zukunft eine weitere Abstimmung mit dem Fuhrpark und dem Dienstleister GHT notwendig sein.

Des Weiteren existieren Anforderungen für die Erfassung der Arbeitszeiten der Fahrer per MDE-Gerät. Die Arbeitszeit der Fahrer wird aktuell über eine papiergeführte Stundenliste erfasst. Die Fahrer schreiben auf diese ihre geleisteten Stunden ein und senden die Liste am Monatsende zu dem Fuhrpark-Leiter. Anschließend trägt dieser die Stunden bei jedem Fahrer einzeln im ERP-System ein. Damit dieser zeitaufwändige Ablauf optimiert wird, soll die Möglichkeit geschaffen werden, dass die Fahrer über ihr mobiles Endgerät die Arbeitszeit erfassen. Diese Daten sollen ebenfalls von dem GHT-Portal erfasst und exportiert werden. Anschließend müssen die Daten aufbereitet und in das ERP-System importiert werden. Die Planung zur Umsetzung dieser Funktionalität wird in Abstimmung der Erweiterungen mit GHT geplant.

## Literaturverzeichnis


[1] M. Fowler, Refactoring: wie Sie das Design bestehender Software verbessern, 2. Auflage. Frechen: mitp, 2020.

[2] M. Fowler, „TechnicalDebtQuadrant“. https://martinfowler.com/bliki/TechnicalDebtQuadrant.html (zugegriffen 30. März 2022).

[3] M. L. Brodie und M. Stonebraker, Migrating legacy systems: gateways, interfaces & the incremental approach. San Francisco, Calif. : [S.l.]: Morgan Kaufmann Publishers ; IT/Information Technology, 1995.

[4] H. M. Sneed, E. Wolf, und H. Heilmann, Softwaremigration in der Praxis: Übertragung alter Softwaresysteme in eine moderne Umgebung, 1. Auflage. Heidelberg: dpunkt.Verlag, 2010.

[5] H. M. Sneed, Objektorientierte Softwaremigration: praxiserprobtes Konzept für die Migration von Legacy-Systemen ; Sanierung der Oberflächen, Daten und Programme ; Konversion von 3-GL-Programmen in oo-Programme ; Kapselung alter Programme und Datenbanken, 1. Aufl. Bonn: Adddison-Wesley Longman, 1999.

[6] N. Wirth, „Program development by stepwise refinement“, Commun. ACM, Bd. 14, Nr. 4, S. 221–227, Apr. 1971, doi: 10.1145/362575.362577.

[7] C. Rupp und SOPHIST-Gesellschaft für Innovatives Software-Engineering, Requirements-Engineering und -Management: das Handbuch für Anforderungen in jeder Situation, 7., Aktualisierte und Erweiterte Auflage. München: Hanser, 2021.

[8] J. Feldhusen, G. Pahl, und W. Beitz, Hrsg., Pahl/Beitz Konstruktionslehre: Methoden und Anwendung erfolgreicher Produktentwicklung, 8., Vollst. überarb. Aufl. Berlin Heidelberg: Springer Vieweg, 2013. doi: 10.1007/978-3-642-29569-0.

[9] „Requirements Engineering & Management, 5. Auflage - Kap 5 Ermittlung.pdf“.

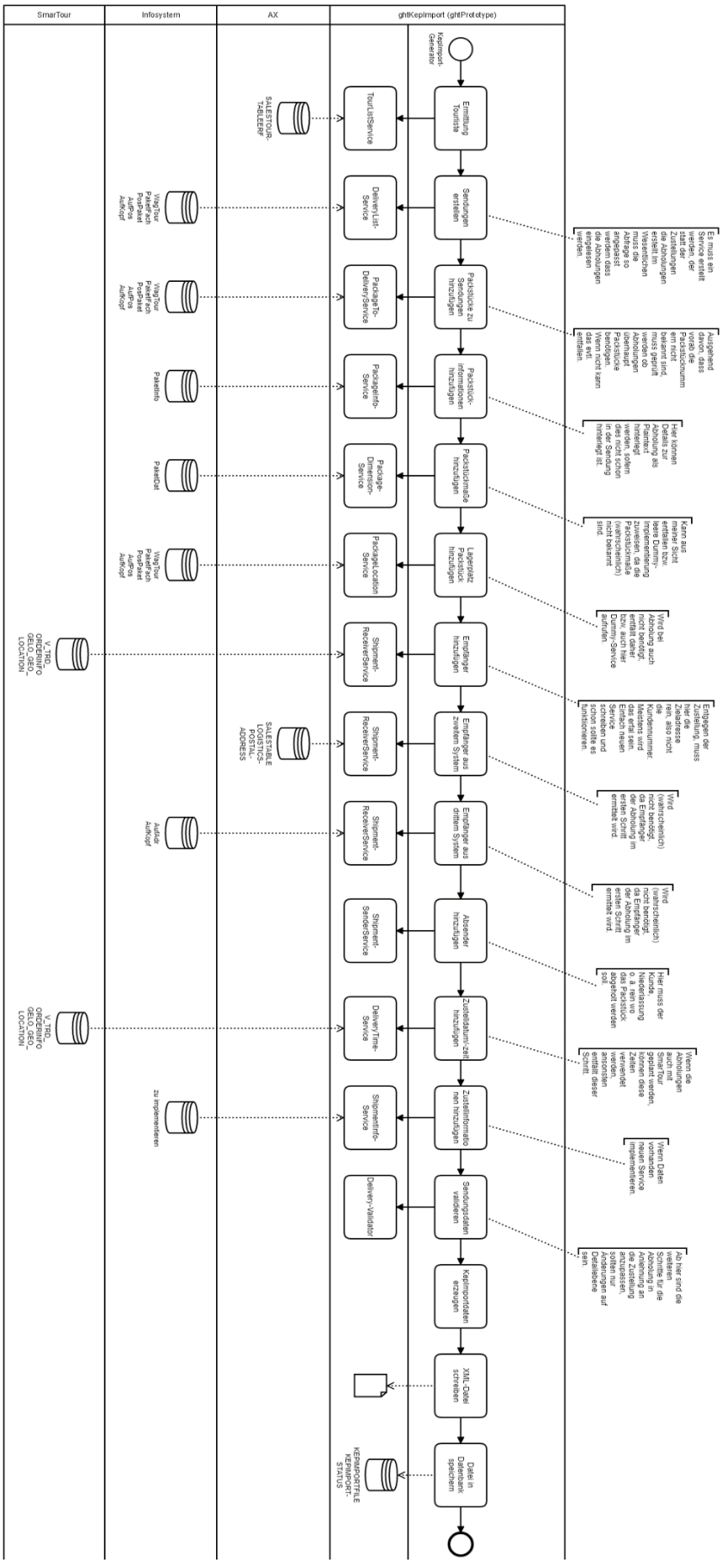
# Anhang 1 SOPHIST Ermittlungstechnikenauswahlmatrix


| | Feldbeobachtung | Contextual Inquiry | Apprenticing | Einzelinterview | Gruppeninterview | Fragebogen | Systemarchäologie | Reuse von Anforderungen | Brainstorming/ Brainstorming Paradox | Methode 6-3-5 | CrowdRE | Design Thinking | Living Lab |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Wissen des Stakeholders bezüglich des Ermittlungsziels | | | | | | | | | | | | | |
| - gering | - | - | - | - | - | - | + | + | - | - | - | o | o |
| - groß | o | + | + | + | + | + | o | - | - | o | + | o | o |
| Freiheitsgrad für Nutzung des Systems | | | | | | | | | | | | | |
| - zwingende Nutzung | + | + | o | + | + | o | - | - | o | o | o | + | + |
| - freiwillige Nutzung | o | o | o | o | o | + | o | o | o | o | + | o | + |
| Motivation des Stakeholders | | | | | | | | | | | | | |
| - motivierbar durch Ermittlungstechnik | o | + | o | + | + | o | o | o | + | o | + | + | + |
| - nicht motivierbar | + | - | o | - | o | - | + | + | - | - | - | - | o |
| Bevorzugter Interaktionsgrad des Stakeholders | | | | | | | | | | | | | |
| - aktiv handlungsorientiert | + | o | - | - | - | - | o | o | - | - | - | + | + |
| - aktiv sprachorientiert | - | o | o | + | + | + | o | o | + | + | + | o | o |
| - aktiv gemischt | - | + | + | o | o | - | o | o | o | o | + | + | + |
| Örtliche Verteilung der Stakeholder | | | | | | | | | | | | | |
| - nur virtuell präsent | - | - | - | o | + | o | o | o | o | + | o | - | - |
| - keine gleichzeitige Kommunikation möglich | - | - | - | + | - | + | + | + | - | + | + | - | - |
| Zeitliche Verfügbarkeit des Stakeholders | | | | | | | | | | | | | |
| - flexibel | o | o | o | o | + | o | o | o | + | o | o | + | + |
| - begrenzt | o | + | + | + | o | o | o | o | o | + | + | o | o |
| - sehr beschäftigt | + | + | - | + | - | + | + | + | - | + | + | - | - |
| Anzahl Meinungen | | | | | | | | | | | | | |
| - eine Meinung/gemeinsames Verständnis | o | o | o | + | o | - | o | o | o | o | - | - | - |
| - viele Meinungen | - | o | o | o | + | + | - | - | o | + | + | + | + |
| Gruppendynamik zwischen den Stakeholdern | | | | | | | | | | | | | |
| - problematisch | o | o | o | + | o | + | + | + | - | + | o | + | - |
| Kommunikative Fähigkeit des Stakeholders | | | | | | | | | | | | | |
| - gering | + | o | o | o | - | o | + | + | - | - | - | - | o |
| - hoch | o | o | o | + | + | o | o | o | + | o | o | o | o |
| Abstraktionsvermögen des Stakeholders | | | | | | | | | | | | | |
| - gering | + | + | + | - | o | - | o | o | - | - | - | - | + |
| Komplexität des Marktes | | | | | | | | | | | | | |
| - Nutzer unbekannt | - | - | - | - | - | o | + | + | o | o | + | - | + |
| Komplexität des Sachverhalts | | | | | | | | | | | | | |
| - hoch | - | + | o | + | + | - | + | + | o | o | - | o | + |
| Erlebbarkeit der Abläufe | | | | | | | | | | | | | |
| - nicht erlebbar | - | - | - | + | + | o | + | + | o | o | o | o | - |
| - erlebbar | + | + | + | o | o | o | o | o | o | o | o | o | o |
| Detailgrad der gesuchten Informationen | | | | | | | | | | | | | |
| - grob | + | + | o | o | o | + | + | o | + | + | o | + | + |
| - fein | o | + | + | o | o | - | + | + | o | o | - | o | o |

[7, S. 164]

„Ist für eine Technik ‚nicht empfohlen' (-) eingetragen, sollten Sie diese Technik bei gegebenen Kriterien nicht anwenden. Der Eintrag ‚empfohlen' (+) bezeichnet eine Technik, die unter den gegebenen Kriterien effizient eingesetzt werden kann. Ist eine Technik mit ‚kein Einfluss' (0) bewertet worden, so wirkt sich das Kriterium nicht auf die Technikauswahl aus. Sie ist zwar anwendbar, aber typischerweise gibt es geeignetere Techniken." [7, S. 163]

# Anhang 2 ghtKepImportAbholung



# Anhang 3 Einzelinterview Fuhrpark



## Schnittstelle elektronische Auslieferung GHT-Portal

Masterthesis Florian Keßler

### 1. Prozessmerkmale

Am Vortrag werden die Autos mit der entsprechenden Ware bestückt, damit die Fahrer am nächsten Tag ihre Tour abfertigen können. Dabei besitzt jeder Fahrer ein MDE-Gerät welches von erfal verwaltet und bereitgestellt wird. Die Auslieferung der Waren über den Fuhrpark wird von Montag bis Freitag durchgeführt. Dabei ist ein Großteil der Touren über einen Tag, wobei es hierbei Außnahmen gibt. Die maximale Tourlänge ist zwei Tage.

Die Datenübertragung der Auslieferungsdaten zu dem GHT-Portal findet von Montag bis Freitag von 16:30 Uhr bis 22 Uhr jede halbe Stunde statt. Dagegen läuft der Import der Daten allle zwei Minuten.

Der Fahrer wählt aus einer Liste seine Tournummer aus, anschließend sieht er alle Pakete welche auf dem Auto verladen worden sind. Anschließend fährt er Kunde für Kunde an und liefert die entsprechenden Pakete dort ab.

Auf eine detaillierte Erfassung des Auslieferungsprozesses wurde verzichtet, da diese für die Schnittstelle irrelevant ist.

### 2. Anforderungen an die Schnittstelle

Die bisherigen Schnittstellenfunktionalitäten sollen bestehen bleiben. Dies betrifft sowohl die Übertragung an das GHT-Portal sowie der Import der Zustellungsdaten inklusive die Möglichkeit, einen Sendungsbericht herunterzuladen.

Die konkreten Anforderungen an diese Schnittstelle können seitens der externen Logistik nicht präzise formuliert werden, da die Abstimmung der Schnittstellen nicht mit dieser Abteilung stattgefunden haben.

Desweiteren bestehen folgende neue Anforderungen für die Übertragung an das GHT-Portal:

- Die Daten des Informationsfeldes müssen nicht mehr übertragen werden. Dies wird von den Fahrern nicht benutzt da die Übersichtlichkeit nicht gegeben ist. Auf Nachfrage gibt es hier nicht den Bedarf diese Informationen auf eine andere Art und Weise zur Verfügung zu stellen.
- Am Kundenstamm ist der Ablageort, die Telefonnummer und ein Zustellfenster gepflegt. Diese Daten müssen mitübertragen werden.
- Geplante Rückholungen müssen mitübertragen und dem Fahrer angezeigt werden.

Für den Import der Daten von GHT bei erfal bestehen ebenfalls neue Anforderungen:

- Ungeplante Rückholungen müssen von der Schnittstelle verarbeitet werden und in das ePS persistiert werden.
- Das Layout des Sendungsberichtes muss überarbeitet werden, welches ein erfal Branding erhalten soll

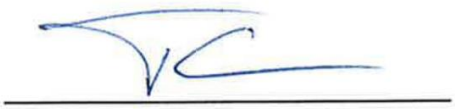
- Der Sendungsbericht muss nicht mehr in dem hauseigenen Dokumentenmangementsystem gespeichert werden. Diese Erstellung bzw. Generierung soll über das Produktionssystem ePS möglich sein.

## 3. Ergänzungen und Anmerkungen

Eine Anforderungen von der IT-Abteilung intern ist die Datenminimierung nach DSGVO Artikel 5.

## 4. Abnahme Gesprächsprotokoll

| | |
|---|---|
| Datum | 21.04.2022 10 Uhr bis 11:30 Uhr |
| Gesprächsteilnehmer | Florian Keßler (Leiter IT)<br>Tamma Andre (Leiter Externe Logistik) |



Unterschrift Andre Tamma

Unterschrift Florian Keßler

# Anhang 4 Einzelinterview Bestellcenter

**Anforderungsanalyse**

## Schnittstelle elektronische Auslieferung GHT-Portal

Masterthesis Florian Keßler

### 1. Prozessmerkmale

Der zugrundliegende Prozess der Schnittstelle ist im weiteren Verlauf des Interviews von keiner Relevanz, da die Mitarbeiter des Bestellcenters nur den Sendungsbericht benutzen.

### 2. Anforderungen an den Sendungsbericht

Es sollen alle Informationen, welche auf dem aktuellen Sendungsbericht vorhanden sind, ebenfalls auf dem Sendungsbericht enthalten sein. Das Design soll ein erfal spezifische Form erhalten. Dieses Design wird von der Marketing-Abteilung erstellt.

### 3. Ergänzungen und Anmerkungen

Weitere Anforderungen bestehen nicht.

### 4. Abnahme Gesprächsprotokoll

| | |
|---|---|
| Datum | 04.05.2022 |
| Gesprächsteilnehmer | Florian Keßler (Leiter IT)<br>Kati Schmidt (Leiterin Bestellcenter) |

K. Schmidt

Unterschrift Kati Schmidt

Florian Keßler

Unterschrift Florian Keßler

# Anhang 5 Einzelinterview Qualitätsmanagement



## Schnittstelle elektronische Auslieferung GHT-Portal

Masterthesis Florian Keßler

### 1. Prozessmerkmale

Der zugrundliegende Prozess der Schnittstelle ist im weiteren Verlauf des Interviews von keiner Relevanz, da die Mitarbeiter des Qualitätsmanagementes nur den Sendungsbericht benutzen.

### 2. Anforderungen an den Sendungsbericht

Es sollen alle Informationen, welche auf dem aktuellen Sendungsbericht vorhanden sind, ebenfalls auf dem Sendungsbericht enthalten sein. Das Design soll ein erfal spezifische Form erhalten. Dieses Design wird von der Marketing-Abteilung erstellt.

Der Sendungsbericht soll über das Produktionssystem ePS aufrufbar sein und nicht mehr im Dokumentationssystem von erfal vorliegen.

### 3. Ergänzungen und Anmerkungen

Der Mechanismus, dass alle Pakete bei der Zustellung gescannt werden müssen, womit kein Paket auf dem Auto vergessen werden kann, wird von den Fahrern umgangen. Dies stellt das Qualitätsmanagement vor die Herausforderung, dass kein zweifelsfreier Nachweis über die Zustellung der bestellten Ware vorliegt. Da dies ein Fehler in der Bedienung der Software auf den MDE-Geräten darstellt und eine Weiterentwicklung dieser Software über die Firma GHT nicht gewollt ist, wird eine organisatorische Richtlinie geschaffen. Die Richtlinie ist als Arbeitsanweisung zu verstehen und muss seitens externer Logistik bzw. der Fahrer befolgt werden. Diese wird seitens Qualitätsmanagement ausgearbeitet und anschließend an einer zentralen Stelle veröffentlicht werden.

## 4. Abnahme Gesprächsprotokoll

| | |
|---|---|
| Datum | 05.05.2022 |
| Gesprächsteilnehmer | Florian Keßler (Leiter IT)<br>Manuela Hirth (Leiterin Qualitätsmanagement)<br>Lukas Schönherr (Mitarbeiter Qualitätsmanagement) |

Unterschrift Manuela Hirth

Unterschrift Lukas Schönherr

Unterschrift Florian Keßler

# Anhang 6 Arbeitsanweisung elektronische Auslieferung

**Arbeitsanweisung**

## Ablauf elektronische Auslieferung

AA 5162

Revision: 0

### 1. Zweck und Geltungsbereich

Die Anweisung gilt für die Auslieferungsfahrer und ist zwingend einzuhalten.

### 2. Mögliche Situationen bei Auslieferung

| Fall 1: Ware wird an Empfänger übergeben | Fall 2: Ware wird an Dritten übergeben | Fall 3: Ware wird beim Empfänger abgestellt | Fall 4: Ware kann nicht zugestellt werden |
|---|---|---|---|
| - Fahrer entlädt das Fahrzeug mit allen Packstücken des Kunden<br>- Fahrer scannt alle Packstücke außerhalb des Fahrzeugs<br>- Fahrer lässt Kunden unterschreiben/setzt eigenhändig einen Haken im Unterschriftfeld*<br>- Fahrer fragt nach Namen und trägt den Namen im Namensfeld mittels Tastatur ein<br>- Fahrer schließt Zustellung ab | - Fahrer entlädt das Fahrzeug mit allen Packstücken des Kunden<br>- Fahrer scannt alle Packstücke außerhalb des Fahrzeugs<br>- Fahrer lässt Dritten unterschreiben/setzt eigenhändig einen Haken im Unterschriftfeld<br>- Fahrer fragt nach dem Namen und trägt den Namen mit Zusatzinfo (z. B. Hausnummer/Geschäft + Gewerk) an welchem Ort Packstücke abgegeben wurden im Namensfeld mittels Tastatur ein*<br>- Fahrer schließt Zustellung ab | - Fahrer entlädt das Fahrzeug mit allen Packstücken des Kunden<br>- Fahrer scannt alle Packstücke außerhalb des Fahrzeugs<br>- Fahrer setzt Freihand einen Haken im Unterschriftenfeld<br>- Fahrer trägt im Namensfeld den eindeutig und aussagekräftig beschriebenen Ablageort ein (Garage/Terasse/...)<br>- Fahrer schließt Zustellung ab | - Fahrer setzt Status auf unzustellbar und schließt die Zustellung ab |

* in erster Linie soll eine Unterschrift verlangt werden, ist das nicht möglich, genügt der Haken im Unterschriftenfeld

## Selbstständigkeitserklärung gem. § 13 Absatz 5 MPO

Hiermit versichere ich, Florian Keßler, dass ich die vorliegende Masterarbeit mit dem Titel

Migration und Erweiterung eines Systems zur Unterstützung der Warenauslieferung

selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Chemnitz, den 05.07.2022

05.07.2022

X Florian Keßler

Florian Keßler

Signiert von: 194d626c-2e59-4371-a9da-4ec3fbc24dad