Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Kim Beier

Realisierung eines Software-Leitstandes am Beispiel des HAW-Logistics-Systems

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Kim Beier

Realisierung eines Software-Leitstandes am Beispiel des HAW-Logistics-Systems

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

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

Betreuender Prüfer: Prof. Dr. Stefan Sarstedt
Zweitgutachter: Prof. Dr. Olaf Zukunft

Eingereicht am: 27. April 2012

Kim Beier

Thema der Arbeit
Realisierung eines Software-Leitstandes am Beispiel des HAW-Logistics-Systems

Stichworte
Monitoring, Leitstand, Framework, .NET, C#, Softwaremetrik, Windows Presentation Foundation (WPF), Extensible Application Markup Language (XAML), HAW-Logistics-System (HLS), Quasar

Kurzzusammenfassung
Durch die Überwachung einer Anwendung durch einen Leitstand werden statistische Daten und Fehler ermittelt. Der Leitstand wird durch eine Benutzeroberfläche mit dem Komponentendiagramm des überwachten System realisiert. Die ermittelten statischen Daten werden analysiert, ausgewertet und dargestellt. Diese geben im Weiteren Aufschluss über die Anwendung. Die ermittelten Fehler helfen den Benutzer, die Fehler schneller im überwachten System zu lokalisieren und zu beheben.

Kim Beier

Title of the paper
Implementation of a software control center on the example of the HAW-Logistics-Systems

Keywords
Monitoring, Control Center, Framework, .NET, C#, Software Metrics, Windows Presentation Foundation (WPF), Extensible Application Markup Language (XAML), HAW-Logistics-System (HLS), Quasar

Abstract
By monitoring an application by a control station, data, and statistical errors are determined. The control center is implemented through a graphical user interface with the component diagram of the monitored system. The static data are analyzed, evaluated and presented. These give additional information on the application. The identified errors will help the user to quickly locate and fix the fault in the monitored system.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

ASP	Active Server Pages
AWK	Anwendungskern
COCOMO	Constructive Cost Model
FEU	Forty-foot Equivalent Unit
GQM	Goal-Question-Metric
HLS	HAW-Logistics-System
HTML	Hypertext Markup Language
JMX	Java Management Extensions
JVM	Java Virtual Machine
LOC	Lines of Code
MQ	Message Queuing
SQL	Structured Query Language
STA	Single-threaded Apatment
SVG	Scaleable Vector Graphics
TEU	Twenty-foot Equivalent Unit
WPF	Windows Presentation Foundation
XAML	Extensible Application Markup Language
XML	Extensible Markup Language

1. Einführung

1.1. Motivation

Die Arbeit ist in den Bereich Überwachung, systematische Erfassung(Protokollierung), Beobachtung eines Systems einzuordnen. Meist nimmt ein Ablauf bzw. Prozess in einer Anwendung nicht den gewünschten Verlauf. Die Funktion des Monitorings schafft Abhilfe. Das Monitoring beobachtet den Verlauf eines Ablaufs bzw. Prozesses und analysiert, bei nicht gewünschten Verlauf, die möglichen Ursachen. Das Konzept des Monitorings wird schon seit Jahren angewendet, z.B. in Kraftwerken. In einem Kraftwerk wird in einem extra Raum, welcher meist auch Kontrollraum genannt wird, die Funktion des Monitorings betrieben in Form eines Leitstandes. In diesem sogenannten Kontrollraum laufen alle Informationen zusammen, die grafisch z.B. auf einer Leinwand dargestellt werden. Durch diesen Leitstand können alle Aktivitäten im gesamten Kraftwerk überwacht werden. Bei einem nicht routinierten Ablauf einer Aktivität im Kraftwerk greift das Personal, welches im Kontrollraum sitzt, ein oder schickt Personal an die betroffene Stelle im Kraftwerk. Durch den Eingriff werden Probleme schnell in Kraftwerk gelöst. Durch den Leitstand ist die Sicherheit im Kraftwerk drastisch gestiegen, denn alle Aktivitäten im Kraftwerk können mit diesem Leitstand überwacht werden.
Ein anderer Bereich des Monitorings ist im Bereich des Netzwerks angesiedelt. Der Netzwerkverkehr eines Computers kann durch ein spezielles Programm komplett überwacht werden. Dieses Programm zeigt alle Aktivitäten, im Bereich des Netzwerksverkehrs, an. Durch Auswertungen kann der Benutzer analysieren, ob unberechtigt Verbindungen ins Netzwerk aufgebaut wurden sind und kann eingreifen, wenn es der Fall war.
Aber welche Möglichkeiten bietet das Monitoring im Bereich von Anwendungen? Kann überhaupt eine Anwendung überwacht werden? Wenn ja, was kann überwacht werden und wie?

1.2. Ziel der Arbeit

Das Ziel dieser Arbeit ist, eine Anwendung kontinuierlich zu beobachten. Dies erfolgt durch eine grafische Benutzeroberfläche, die das Komponentendiagramm der überwachten Anwen-

dung zeigt. In regelmäßigen Abständen wird die Benutzeroberfläche aktualisiert und zeigt den letzten aktuellen Zustand der Anwendung. Die Benutzeroberfläche zeigt statistische Werte, aber auch mögliche Ausfälle, die schnell, Dank Benutzeroberfläche, lokalisiert werden können. Die Informationsvermittlung über den Zustand der Anwendung erfolgt durch den Nachrichtenaustausch über eine Message Queue. Bei Aktivität schickt die Anwendung Nachrichten an die Message Queue heraus. Der Leitsand wertet die Daten aus, und stellt die Daten auf der Benutzeroberfläche dar.

1.3. Aufbau der Arbeit

Die Arbeit gliedert sich in acht Kapiteln.
Das erste Kapitel gibt eine Einführung in das Thema der Arbeit und das Ziel, welches mit der Arbeit erreicht wurde.
Das zweite Kapitel gibt einen kleinen Einblick in die Grundlagen, die ansatzweise in die Arbeit eingeflossen sind, u.a. Framework, Leitstand, Monitoring, Softwaremetrik, Windows Presentation Foundation (WPF) und Extensible Application Markup Language (XAML).
Das darauffolgende Kapitel gibt einen Einblick in das Fallbeispiel HAW-Logistics-System (HLS). Beschrieben wird der Ablauf des Systems, die Architektur, die Bestandteile der Architektur, die technische Komponenten, die technische und logische Komponente, die technische Infrastruktur des Systems, die Dialoge und die Entwicklungsumgebung.
Im vierten Kapitel werden die Anforderungen an den Leitstand beschrieben. Das fachliche Datenmodell und die fachliche Architektur mit Schnittstellen zu den Nachbarsystemen wird vorgestellt. Außerdem werden die Anwendungsfälle beschrieben, die später im Implementierungskapitel realisiert werden. Und zu guter Letzt werden die Dialoge gezeigt, mit denen später das System bedient werden sollen.
Das fünfte Kapitel zeigt die Architektur des Systems, den Ablauf, einige Entwurfsmuster und die Komponentenaussensichten- und innensichten.
Im sechsten Kapitel werden die Resultate der Implementierung vorgestellt, u.a der Anwendungskern, die Anwendungskernfassade, die Anbindung zum Apache Active MQ Server und die Benutzeroberfläche. Außerdem werden aufgetretene Probleme kurz erläutert.
Im Testkapitel werden die Tests, in Einheiten und im Ganzen, erläutert.
Das letzte Kapitel erläutert Probleme, Fazit und Erweiterungsmöglichkeiten der Arbeit.

2. Grundlagen

2.1. Framework

Bei Entwicklung ähnlicher Anwendungen sollten nicht immer einzelne Klassenbibliotheken genutzt werden, sondern es sollte eine Lösung verwendet werden, wo das Rahmengerüst schon steht, das sogenannte Framework (vgl. IEEE Xplore (2011)). Bei einem Framework werden nicht einzelne Klassen wiederverwendet, sondern die komplette Architektur. Das Framework gibt den Kontrollfluss für eine Anwendung vor, es muss nur der anwendungsspezifische Teil in das Framework integriert werden, im Gegensatz zu einer Klassenbibliothek, dort muss der Entwickler den gesamten Kontrollfluss selber festlegen. In einem Framework werden die sogenannten Hot Spots (siehe Glossar) erweitert bzw. ergänzt. Es gibt zwei verschiedene Frameworks, White-Box und Black-Box, eine detaillierte Beschreibung erfolgt im nächstem Abschnitt (vgl. (Ludewig und Lichter, 2010), S. 446).

2.1.1 Definition

Framework wird wie folgt definiert, laut Wikipedia:

> „Ein Framework (englisch für „Rahmenstruktur“oder „Fachwerk“) ist ein Programmiergerüst, das in der Softwaretechnik, insbesondere im Rahmen der objektorientierten Softwareentwicklung sowie bei komponentenbasierten Entwicklungsansätzen, verwendet wird. Im allgemeineren Sinne und nicht als dezidiertes Softwareprodukt verstanden, bezeichnet man mit Framework auch einen Ordnungsrahmen.“ (Wikipedia, 2011b)

2.1.2 White- und Black-Box

In Abbildung 2.1 ist ein **Black-Box-Framework** zu sehen. Black-Box-Frameworks sind meist sehr ausgereifte Frameworks. Die meisten Hot Spots sind bei diesem Framework-Typ lokalisiert und das Spezifizieren erfolgt komponentenbasiert. Es werden Komponenten an das Framework übergeben und dort wird die Spezifizierung durchgeführt, anders als beim White-Box-Framework. Deswegen ist es nicht notwendig die innere Struktur des Frameworks zu

kennen.

Abbildung 2.1 zeigt ein **White-Box-Framework**. Bei einem White-Box-Framework muss man die innere Struktur des Frameworks kennen, um spezifizieren zu können. Man spezifiziert durch Vererbung. Am Anfang fallen alle Frameworks in diese Kategorie, da oft noch keine stabilen Hot Spots vorhanden sind. Erst im Laufe der Zeit stellt sich heraus, an welchen Punkten das Framework angepasst werden muss, um die unterschiedlichen Anwendungsfälle abbilden zu können (vgl. Doberenz und Gewinnus (2010), S. 256).

Abbildung 2.1.: Black-Box-Framework und White-Box-Framework

2.1.3 .NET Framework

Microsoft hat im Jahre 2002 die .NET-Technologie herausgebracht. Die Anforderungen in der Softwareentwicklung stiegen in den letzten Jahren und deshalb hat Microsoft die .NET Technologie entwickelt, damit z.B. die Verteilbarkeit auf mehrere Schichten, Sicherheit und die Skalierbarkeit einer Anwendung gelöst werden können. Das .NET Kürzel steht für eine gemeinsame Plattform für viele Programmiersprachen, wie z.B. C++, C#, Basic, F# und weitere. Das .NET-Framework ist Teil der .NET-Technologie, nämlich die Infrastruktur von der gesamten .NET-Plattform, in der Windows- und Web-Anwendungen erstellt werden können. Folgende Komponenten enthält das .NET-Framework: .NET-Klassenbibliothek, Active Server Pages (ASP), Windows Forms, WPF und weitere. Auf WPF wird detaillierter im Kapitel 2.5 eingegangen. Als diese Komponenten werden als Framework bezeichnet, sie können mit spezifischen Code erweitert werden (vgl. Grechenig u. a. (2010), S. 67).

2.2. Leitstand

Im folgenden Abschnitt wird der Begriff Leitstand weiter erläutert und Ansätze präsentiert.

2.2.1 Definition

Wikipedia definiert den Begriff Leitstand wie folgt:

> „Ein Leitstand (auch Leitwarte, Schaltwarte oder Messwarte genannt) ist ein technische Einrichtung, die den Menschen bei der Leitung eines Prozesses unterstützt. Er ist Teil eines Leitsystems.“ (Wikipedia, 2011d)

2.2.2 Ansätze

Verschiedenste Ansätze sind vorhanden im Bereich Leitstand. In den folgenden beiden Abschnitten werden die beiden Ansätze Leitstand vom Kraftwerk und vom Betriebsleitzentrale erklärt.

2.2.2.1 Kohlekraftwerk

Es gibt schon verschiedene Ansätze eines Leitstandes, wie z.B. im Kraftwerk. Im Leitstand eines Kraftwerkes laufen Messwerte zusammen, wie Anlagen- und verfahrenstechnische Informationen, die an unterschiedlichsten Stellen im Kraftwerk gemessen wurden sind. Beispielweise können folgende Informationen in einem Kohlekraftwerk gemessen werden:

- Dampfkessel
  - Temperatur und Druck des Dampfes
  - Dampfmenge
- Generator
  - Temperaturen
- Dampfturbine
  - Drehzahl
  - Dampfdurchsatz
  - Temperatur des Abdampfes
  - Ein- und Austrittstemperatur des Kühlwassers
- Kühlturm
  - Außentemperatur
  - Kühlwassermenge

Die Messwerte werden im Leitstand angezeigt und ausgewertet. Der Leitstand ist ein geschlossener Raum mit Messgeräten zur Anzeige der Betriebszustände der einzelnen Kohlekraftwerkskomponenten. Das Kraftwerkspersonal kann mit Schaltern und anderen Steuerorganen in den Betriebsablauf eingreifen. Die Eingriffe erfolgen durch digitale Datenübertragung an die

Hilfsantriebe, aufgrund der großen Entfernung der Messstellen, und es bewirkt zum Beispiel das Öffnen oder Schließen einer Armatur oder eine Veränderung der Menge des Brennstoffes (vgl. Wikipedia (2011c)).

Abbildung 2.2.: Leitstand in einem Kraftwerk

2.2.2.2 Zug-Leitzentrale

In der Leitzentrale laufen die gesamte Koordination und Disposition zusammen von allen Zugfahrten, die dort geregelt und überwacht werden. Die gesamten Prozesse werden gesammelt, erfasst, koordiniert und gesteuert. Prozesse können u.a. folgende sein: Zugbewegungen und Gleisbewegungen, diese Prozesse können abgefragt und angezeigt werden. Die Abfrage kann netzweit, landesweit oder bundesweit erfolgen. So kann zu jeder Zeit ermittelt werden, wo sich ein Zug befindet oder ob irgendwelche Unregelmäßigkeiten, z.B. Verspätungen oder sonstige Konflikte, entstehen. Konflikte werden dadurch ermittelt, dass die gespeicherten Fahrplandaten (Soll-Werte) mit den übermittelten Zugdaten (Ist-Werte) verglichen werden. Durch weitere Berechnungen können sogar weitere Folgen berechnet werden, wie z.B. Zugüberholungen, Zugkreuzungen oder Anschlusszüge. Durch die Berechnungen können frühzeitig Maßnahmen eingeleitet werden, um weiter Folgen zu verhindern. Maßnahmen wären z.B. Bereitstellung eines Ersatzzuges, Freihalten von Überholungs-/ Kreuzungsgleisen oder Zugumleitungen (vgl. Langewiesche (2003)).

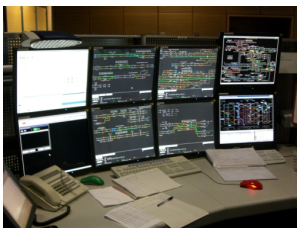

Abbildung 2.3.: Leitstand in einer Betriebszentrale

2.3. Monitoring

Der folgende Abschnitt geht auf das Thema Monitoring ein, auf die Definitionen, auf Arten des Monitorings, auf Werkzeuge und auf vorhandene Ansätze.

2.3.1 Definition

Monitoring wird wie folgt definiert, laut Wikipedia:

> „Monitoring ist ein Überbegriff für alle Arten der unmittelbaren systematischen Erfassung (Protokollierung), Beobachtung oder Überwachung eines Vorgangs oder Prozesses mittels technischer Hilfsmittel oder anderer Beobachtungssysteme. Dabei ist die wiederholende Durchführung ein zentrales Element der jeweiligen Untersuchungsprogramme, um anhand von Ergebnisvergleichen Schlussfolgerungen ziehen zu können." (Wikipedia, 2011e)

Das Webster Wörterbuch definiert Monitoring als:

> „to watch, keep track of, or check usually for a special purpose." (Webster, 2011)

Eine weitere Definition zu Monitoring aus dem Lexikon der Universität Rostock wird wie folgt definiert:

> „Monitoring ist ein Überbegriff für alle Arten der systematischen Erfassung, Beobachtung oder Überwachung eines Vorgangs oder Prozesses mittels technischer Hilfsmittel oder anderer Beobachtungssysteme. Ein Monitoring ist eine in die Zukunft gerichtete Langzeitbeobachtung. Ein Monitoringsystem ermöglicht zum Teil auch Eingriffe bzw. Steuerung der betreffenden Prozesse, sofern sich abzeichnet, dass der Prozess nicht den gewünschten Verlauf nimmt." (Universität Rostock, 2008)

2.3.2 Art des Monitorings

Im folgenden wird darauf eingegangen, welche Arten von Monitoring es gibt.

2.3.2.1 Hardwaremonitoring

Die Leistung der Hardware wird mittels Sensoren gemessen, es werden physikalisch messbare Signale empfangen, wie z.B. von einer CPU.

2.3.2.2 Softwaremonitoring

Im Abschnitt 2.3 wird auf das Thema Softwaremonitoring weiter eingegangen.

2.3.3 Werkzeug

Verschiedenste Werkzeuge sind vorhanden, um Überwachungen zu betreiben, zur Laufzeit oder zur Entwicklungszeit und die statische Struktur.

2.3.3.1 Dynamische Analyse

Die dynamische Analyse befasst sich mit dem laufenden System. Die dynamische Analyse ist ein Werkzeug, welches vier Werkzeuge in diesem Bereich beinhaltet, die Monitore, Logger, Profiler und der Debugger.
Monitore sind in Betriebssystemen, Datenbanken, Plattensystemen, Netzwerken oder virtuellen Maschinen integriert. Der Monitor im Betriebssystem gibt Information über die CPU-Auslastung, Prozesse und den belegten Speicherplatz. Der Monitor in der Datenbank gibt dagegen Information über erfolgte Structured Query Language (SQL)-Abfragen, die Ausführungsgeschwindigkeit und Deadlocks. Der Monitor im Plattensystem gibt Auskunft über Speicherzugriffe, Wartezeiten und die Verwendung von Caches. Monitore in Netzwerken geben Information über Latenzzeiten und Übertragungsraten. Der Monitor in der virtuellen Maschine kann z.B. unter anderem Java Virtual Machine (JVM) sein. Die JVM stellt unter anderem folgende Schnittstelle bereit: Java Management Extensions (JMX). Der Monitor liefert folgende Informationen über die virtuelle Maschine: was der Garbage Collector macht, wie viele Objekte die JVM erzeugt und freigibt und den Speicherbedarf der Objekte.
Logger zeichnen während der Laufzeit auf und schreiben diese Informationen in ein Protokoll, den Umfang des Protokolls lässt sich über das Log-Level steuern, bei niedrigem Level, enthält das Protokoll wenig Informationen und bei hohem Level enthält das Protokoll viel Information, was aber die Laufzeit während des Betriebes deutlich verlangsamt.
Profiler und **Debugger** kommen nur in der Entwicklung und im Test vor, da der Profiler und Debugger das Systemverhalten beeinflussen. Beobachten in der Regel nur ein Prozess.
(vgl. Weigend u. a. (2011), S. 486)

2.3.3.2 Statische Analyse

Auf Basis der Quelle wird die statische Struktur des Systems untersucht. Dabei werden Kennzahlen zur Komplexität und Hinweise auf unerwünschte und unzulässigen Abhängigkeiten ermittelt. Diese Analyse ist wichtig für den Softwareentwurf (vgl. Weigend u. a. (2011), S. 487).

2.3.4 Ansätze

Einige Ansätze im Bereich Monitoring sind schon vorhanden, auf die weiter in den folgenden beiden Abschnitten eingegangen wird.

2.3.4.1 Nagios

Nagios ist ein Überwachungssystem, was IT-Infrastruktur Probleme identifiziert und löst, bevor es zu ernsthaften Problemen wird. Nagios ist skalierbar und flexibel, angepasst auf jede Unternehmensorganisation, um Ausfälle zu vermeiden. Das Werkzeug erkennt und behebt Probleme und mindert sogar zukünftige Probleme, bevor sie sich beim Endanwender oder Kunden auswirken können. Folgende Eigenschaften zeichnet Nagios aus (vgl. Nagios (2011)):

- Monitoring
  Die IT-Infrastruktur in folgenden Bereichen überwachen: System-Metriken, Netzwerkprotokolle, Anwendungen, Dienste, Server und Netzwerkinfrastruktur
- Alarmierung
  Es werden Warnungen versendet, wenn in der IT-Infrastruktur Probleme auftreten, die Warnungen können über E-Mail, SMS oder benutzerdefinierten Skript erfolgen.
- Reaktion
  Mitarbeiter werden benachrichtigt und können sofort auf Ausfälle reagieren und mit den Untersuchungen der Sicherheitswarnungen beginnen.
- Berichtserstattung
  Auflistung der Rekorde über Ausfälle, Ereignisse, Benachrichtigungen und Warnungen für eine spätere Überprüfung.
- Wartung
  Durch geplante Ausfallzeiten werden die Warnmeldungen während der Wartung im Wartungsfenster und Aktualisierungsfenster verhindert.
- Planung
  Durch Diagramme und Berichte über die Kapazitätsplanung und den Trend ist es möglich notwendige Infrastruktur-Aktualisierungen zu erkennen, bevor Störungen auftreten können.

2.3.4.2 Kieker

Kieker ist ein Monitoring und Analyse Framework für Java Anwendungen, .NET und Visual Basic 6 befinden sich derzeit in Entwicklung. Kieker ist dafür da, um kontinuierlich die Produktionssysteme zu überwachen. Der Monitoring Teil ist für Programm Instrumentierung, Datenerfassung und Protokollierung verantwortlich und der Analyse Teil ist für das Lesen, Analysieren und Visualisieren von Messdaten verantwortlich. Erhaltende Messdaten aus dem Framework, werden von folgende Bereichen verwendet (vgl. Ehmke u. a. (2011), S. 3,4):

- Performance-Evaluation (z.B. Engpass-Erkennung),
- Anpassung (z.B. Kapazitätsmanagement),
- Applikation Level Fehlererkennung und Diagnose,
- Simulation (z.B. Arbeitsbelastung, Messung, Protokollierung und Analyse)
- Software Wartung, Reverse Engineering und Modernisierung
- Service Level Management

2.4. Softwaremetrik

Im folgenden Abschnitt wird der Begriff Softwaremetrik erklärt und den Nutzen von Softwaremetriken, sowie die Anforderungen an Metriken und den GQM Ansatz, nicht bestehende Metriken gezielt selber zu definieren.

2.4.1 Definition

Laut Wikipedia wird Softwaremetrik folgendermaßen definiert:

> „Eine Softwaremetrik, oder kurz Metrik, ist eine (meist mathematische) Funktion, die eine Eigenschaft von Software in einen Zahlenwert, auch Maßzahl genannt, abbildet. Hierdurch werden formale Vergleichs- und Bewertungsmöglichkeiten geschaffen. “ (Wikipedia, 2011f)

Eine weitere Definition nach IEEE Standard 1061 lautet:

> „A methodology for establishing quality requirements and identifying, implementing, analyzing and validating the process and product software quality metrics is defined. The methodology spans the entire software life cycle.“ (IEEE-Sandard 1061, 1998)

2.4.2 Nutzen

Die Softwaremetriken haben folgende Ziele (vgl. Ludewig und Lichter (2010), S. 296):

- Qualität von Produkten und Prozessen bewerten
  Das Produkt bzw. den Prozess auf bestimmte Kriterien messen und dann eine Entscheidung treffen, ob die Messergebnisse der geforderten Qualität entsprechen.
- Erfahrungen aufbauen
  Durch mehrere Messvorgänge kann ein Wissen aufgebaut werden und dadurch können Erfahrungen aufgebaut werden.
- Prognosen
  Durch das Messen durch bestimmte Kriterien an Produkten bzw. Prozessen kann nach und nach eine Prognose aufgestellt werden.
- Entscheidungsfindung
  Durch das immer wiederkehrende Messen werden Kennzahlen gewonnen und es können Veränderungen erkannt werden und es kann entschieden werden, was getan werden kann, um das Problem in den Griff zu bekommen.

2.4.3 Anforderungen

Die Softwaremetriken liefern einen Messwert, der bestimmten Anforderungen entsprechen sollte. Folgende Anforderungen werden definiert für Softwaremetriken (vgl. Ludewig und Lichter (2010), S. 299):

- differenziert
  Verschiedene Probanden sollten verschiedene Bewertungen liefern, die auch ersichtlich gemacht werden sollte. Wenn alle Probanden den selben Wert liefern, dann ist die Metrik nicht differenzierbar, z.B. Programmlänge in Lines of Code (LOC).
- vergleichbar
  Alle Bewertungen sollten miteinander vergleichbar sein. Dazu muss eine Skala definiert werden, auf der Abstände und Verhältnisse ersichtlich sind. Die Vergleichbarkeit schließt die Exaktheit ein. Es gibt keinen ungefähren Wert, sondern nur einen exakten Wert, z.B. zyklomatische Komplexität.
- reproduzierbar
  Der Proband sollte bei jeder Durchführung, unter gleichen Bedingungen, die gleiche Bewertung wiedergeben, z.B. Speicherbedarf.

- verfügbar
  Die Bewertungen sollten jeder Zeit zugreifbar sein, wenn man sie braucht, z.B. Zahl der Entwickler.
- relevant
  Jede Bewertung sollte eine praktische Bedeutung haben, um den Nutzen daraus zu ziehen, z.B. zu erwartende Entwicklungskosten.
- rentabel
  Der Aufwand bei der Erhebung einer Metrik sollte nicht die Relevanz der Metrik übersteigen. Ist die Metrik nicht relevant, dann ist sie auch nicht rentabel, z.B. Zahl der entdeckten Fehler im Code.
- plausibel
  Aus einer Metrik wird interpretiert, diese Interpretation sollte plausibel sein, dass bedeutet, dass sie den Beobachtungen entsprechen sollte, die gemacht wurden z.B. Kostenschätzung nach Constructive Cost Model (COCOMO).

2.4.4 Goal-Question-Metric-Ansatz

Es gibt schon definierte Metriken und es gibt Metriken, die man neu definieren muss, da es sie nicht in dieser Form gibt. Dafür gibt es den Ansatz Goal-Question-Metric von Basili und Weiss, der im Jahre 1984 entwickelt wurde (vgl. Ludewig und Lichter (2010), S. 321). Wie der Name schon sagt, besteht Goal-Question-Metric aus drei Schritten, die folgendermaßen lauten:

1 Definiere die relevanten Ziele eines Projektes bzw. Organisation

2 Leite Fragen aus jedem Ziel ab, die geklärt werden müssen, um zu überprüfen, ob das Ziel erreicht wurde.

3 Leite Metriken aus jeder Fragen ab, die die Antwort auf die Frage geben.

Ein kleines Beispiel von van Solingen und Berghout im Jahre 1999 gibt einen Einblick in den Goal-Question-Metric-Ansatz.
Zuerst muss das Ziel definiert werden, das könnte wie folgt lauten:

Z Erhöhung der Wirksamkeit von Code-Inspektion, dass heißt ohne höheren Aufwand mehr Mängel entdecken

Aus diesem Ziel müssen jetzt Fragen abgeleitet werden, die wie folgt lauten könnten:

F1 Wie effektiv sind die Inspektionen?

F2 Welche Merkmale muss ein Prüfling (in diesem Fall, das Dokument) haben, damit eine Inspektion möglichst erfolgreich ist?

F3 Welchen Qualitätskriterien soll ein Prüfling nach einer Inspektion genügen?

Die definierten Fragen sind noch zu komplex, um daraus Metriken abzuleiten, deswegen müssen die Fragen noch weiter vereinfacht werden. Das könnte für die erste Frage folgendermaßen aussehen:

F1.1 Wie viel Aufwand pro Woche erfordern die Inspektionen?

F1.2 Wie viele Fehler werden in einer Inspektion entdeckt?

F1.3 Wie viele Fehler pro Fehlerklasse (leicht, mittel, schwer) werden entdeckt?

F1.4 Wie viel Aufwand ist erforderlich, um einen Prüfling einer gewissen Komplexität zu inspizieren?

F1.5 Wie viel Aufwand ist erforderlich, um einen Prüfling einer gewissen Größe zu inspizieren?

Daraus lassen sich folgende Metriken definieren, die die Antwort auf die Frage geben:

M1 Anzahl der entdeckten Fehler pro Inspektion

M2 Anzahl der Inspektionen pro Woche

M3 Aufwand pro Inspektion in Stunden

M4 Anzahl der entdeckten Fehler pro Fehlerklasse pro Inspektion

M5 Komplexität des Prüflings (z.B. als zyklomatische Komplexität)

M6 Größe des Prüflings in LOC

Die Abbildung 2.4 zeigt, wie die Metriken den Fragen zugeordnet sind. Damit hat man nun die Metriken definiert und nun kann man mit den Metriken den Prozess bzw. das Produkt messen und dadurch ein Wissen oder eine Datensammlung aufbauen.

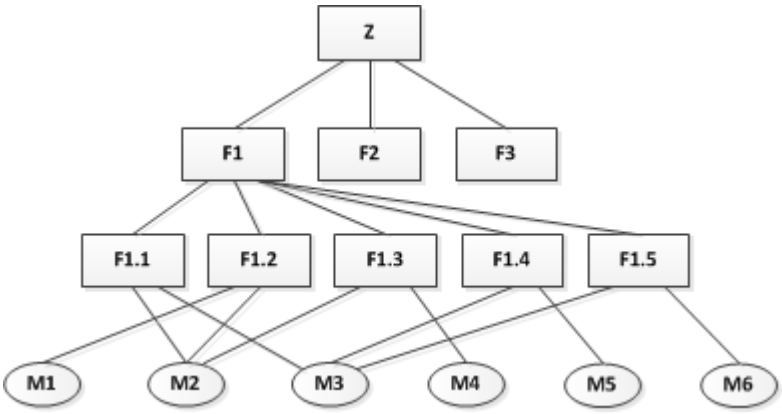

Abbildung 2.4.: Beispiel zum Goal-Question-Metric-Ansatz

2.5. WPF

WPF ist ein Programmiermodell, womit sich Webbrowser- und Windows-Anwendungen entwickeln lassen. Dabei werden die Geschäftslogik und die Präsentation getrennt. Die Präsentation wird üblicherweise mit der Auszeichnungssprache XAML (siehe Abschnitt 2.6) beschrieben. WPF ist ein Teil des .NET-Frameworks 3.0 und höher (vgl. Huber (2010)).

2.5.1 WPF-Windows-Anwendung

Die WPF-Windows-Anwendung kann in drei verschiedenen Varianten entwickelt werden. Folgende drei Varianten stehen zur Verfügung:

- eine reine Codeanwendung: nur C# ohne XAML
- eine reine, kompilierte XAML-Anwendung: nur XAML mit Inline-Code in C# (in XAML eingebettete C#-Code)
- eine Windows-Anwendung in XAML und C# (was auch üblicherweise von Visual Studio 2010 angelegt wird)

Im Folgenden werden die drei Varianten betrachtet.

2.5.1.1 Anwendung mit C# und XAML

Standardmäßig erstellt man eine WPF-Anwendung, die eine Applikation- und Window-Datei enthält, die jeweils als XAML-Datei hinterlegt ist, mit der dazugehörigen Codebehind-Datei.

Das bedeutet, dass man die Geschäftslogik von der Präsentation trennt. Die Codebehind-Datei enthält den selben Namen, wie die XAML-Datei, einschließlich mit der Dateiendung .cs. Zum Beispiel hat die Datei Window.xaml, die dazugehörige Codebehind-Datei Window.xaml.cs.
Die Window.xaml Datei enthält den XAML Code für die Darstellung der Benutzeroberfläche. Bei der Definition eines Events in der Benutzeroberfläche, z.B. beim Button-Element das Event Click, wird beim Button-Element in der XAML-Datei das Event hinzugefügt, dass kann dann wie folgt aussehen:

```
1 <Button Margin="5" Name="btn"
2         Click="HandleButtonClick"
3         Content="Wieviel Uhr ist es?" />
```

Listing 2.1: XAML Code Button Aktion

Zu dem Button-Element wurde der Event Handler HandleButtonClick hinzugefügt, was auch in der Codebehind-Datei Window definiert werden muss, denn dort folgt die Geschäftslogik des Button-Elements.
Die App.xaml Datei ist dafür da, um in XAML das Applikation-Objekt für die Anwendung zu definieren. Dort ist noch ein weiteres Attribut definiert, das StartupUri. Dieses Attribut gibt an, welche Datei die Instanz der Applikation beim Starten verwenden soll, z.B. kann es wie folgt aussehen:

```
1 StartupUri="MainWindow.xaml"
```

Listing 2.2: XAML Code StartupUri

In diesem Beispiel, wird auf die MainWindow.xaml Datei verwiesen, die beim Starten von der Instanz der Applikation aufgerufen werden soll. Logischerweise befindet sich in der Codebehind-Datei von der App.xaml, die Geschäftslogik.

2.5.1.2 Anwendung mit XAML und C# -Inline

Die WPF-Anwendung enthält reinen XAML Code. Sie enthält als Vaterelement nicht das Standardmäßige Window-Element, sondern ein Page-Element, da sie im Internet Explorer dargestellt wird. Die Codebehind-Dateien gehen nicht in die Kompilierung ein. Leider kann nicht alles in XAML Code dargestellt werden, deswegen ist es kein reiner XAML Code, Elemente, die nicht dargestellt werden können, werden durch C# -Inline eingebettet, das bedeutet, dass im XAML Code eingebetteter C# Code enthalten ist. C# -Inline wird folgendermaßen eingeleitet:

```
<x:Code>
  <![CDATA[
    void HandleButtonClick(object sender, RoutedEventArgs e)
    {
      MessageBox.Show(
      string.Format("Es ist {0:HH:mm:ss} Uhr.", DateTime.Now));
    }
  ]]>
</x:Code>
```

Listing 2.3: XAML Code C# -Inline Code

Das Beispiel zeigt C# -Inline Code, das mit dem x:Code-Element in die XAML Datei eingebettet wird. Das Schlüsselwort CDATA (Character Data) steht dafür, dass der C# Code im x:Code Bereich nicht vom XAML Parser geprüft werden soll, da der C# Code reservierte Extensible Markup Language (XML) Zeichen haben kann.
Es ist nicht zu empfehlen, C# -Inline Code zu verwenden, da dies sehr fehleranfällig ist.

2.5.1.3 Anwendung mit C#

Die WPF-Anwendung besteht nur aus reinem C# Code. Die Präsentation und die Geschäftslogik wird komplett in C# Code geschrieben. Es gibt eine Main-Klasse, die von Window erbt. Die Main-Klasse enthält eine Main-Methode in der ein Applikation-Objekt erzeugt wird, welches mit der Run-Methode gestartet wird. Bei dieser Variante muss darauf geachtet werden, dass man über die Main-Methode der Main-Klasse das Attribut STAThread definiert, damit die Anwendung bzw. der Haupt-Thread in einem Single-threaded Apatment (STA) gestartet wird. Das muss definiert werden, damit die Anwendung nur genau von einem Thread zur Laufzeit verwendet wird.

2.5.2 Namensraum

Es müssen einige Klassen im Namensraum des Projektes definiert werden, die meisten Klassen liegen im Namensraum System.Window, wie z.B. Controls, Media, Markup und viele andere.

2.5.3 Assemblies

Beim Anlegen einer WPF-Anwendung werden standardmäßig folgende Assemblies referenziert: PresentationCore.dll, PresentationFramework.dll, WindowsBase.dll und weitere.

2.5.4 Grafikdarstellung

WPF stellt die Grafiken in der Anwendung vektorbasiert dar. Dadurch ist die Anwendung skalierbar, sie wird nicht pixelig dargestellt. In den Versionen davor, wurden die Grafiken rasterbasiert dargestellt. Die Abbildung 2.5 zeigt eine Originallinie und die Darstellung der Linie, raster- und vektorbasiert.

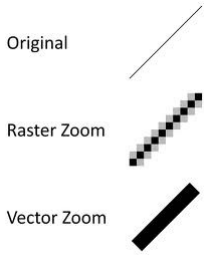

Abbildung 2.5.: Beispiel zum WPF - Raster- und Vektorbasiert

2.6. XAML

XAML ist eine XML deklarative Programmiersprache, mit der das Design der Anwendung implementiert werden kann. Das bedeutet, dass definiert wird, was gemacht werden soll, aber nicht wie es gemacht werden soll. XAML stellt eine eigenständige Sprache dar und vereinfacht das Erstellen von User Interface für eine .NET Framework-Anwendung. XAML wird üblicherweise dazu verwendet, um Fenster, Dialogfelder, Seiten, Benutzersteuerelemente und viele andere grafische Elemente darzustellen. In ASP.NET wird dieser Aufbau schon verwendet, dort wird das Layout mit Hypertext Markup Language (HTML) und die Logik in C# implementiert (vgl. Stropek und Huber (2008)).
Der Aufbau einer XAML-Datei ist hierarchisch gegliedert, das heißt, dass jedes Element nur ein Elternelement besitzt und das jedes Element eine unbegrenzte Anzahl an Kinderelemente besitzen kann, nur bei einigen Elementen ist die Anzahl der Kinderelemente beschränkt, z.B. besitzt die Scrollbar kein einziges Kindelement.

2.6.1 Wurzelelement

Wie jede XML-Datei fängt sie mit einem Wurzel-Tag an. Bei einer XAML-Datei fängt man z.B. mit dem Vaterelement <Window> an, damit wird ein Fenster erstellt und endet auch wieder mit dem Vater-Tag. Das wie folgt aussieht:

```
<Window [Optionen]>
</Window>
```

Listing 2.4: XAML Code Anfang einer Datei

In den Optionen werden weitere Eigenschaften angegeben, wie z.B. der Klassenname, die Namensräume, der Titel und gegebenenfalls ein Icon, wo in den weiteren Unterkapiteln darauf eingegangen wird. Eine Vater-Wurzel kann auch unter anderem eine Page(Seite) sein, es gibt des Weiteren noch weitere Vater-Wurzeln.
In das Vaterelement kann man weitere Kinderelemente einfügen. Das Vaterelement muss folgende Attribute enthalten: xmlns und xmlns:x, damit der Parser weiß, welche Namensräume der Parser verwenden soll.

2.6.2 Klassenname

Der Klassenname einer XAML-Datei kann wie folgt definiert werden:

```
x:Class="Visualisierung.MainWindow"
```

Listing 2.5: XAML Code Klassenname

Die Definition x:Class stellt die Verbindung zwischen dem Window-Element in der XAML-Datei und der Codebehind-Datei Window.xaml.cs definierten Window-Klasse her und legt den Namen der Klasse fest, die der Compiler verwenden soll.

2.6.3 XAML-Namensraum

Ein XAML-Namensraum ist eine Erweiterung des XML-Namensraums. Es wird auf folgende Namensräume verwiesen:

```
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

Listing 2.6: XAML Code Namensräume

Der erste Namensraum verweist auf den XML Namensraum für WPF und ordnet den Namensraum als Standard zu, dass bedeutet, das alle Klassen in diesem Namensraum direkt ohne Präfix in XAML verwendet werden können, da die WPF-Klassen am häufigsten in der Datei verwendet werden. Dem zweite Namensraum wird ein separaten Namensraum zugeordnet. Da ein zweiter Namensraum definiert wurde, muss der zweite Namensraum nach dem Attribut, der Namensraum, der als Standard festgelegt wurde, ein Präfix folgen. Damit der Zugriff auf

den XAML-Namensraum geregelt ist, muss für jeden weiteren Zugriff auf den Namensraum, immer das definierte Präfix gefolgt von der Klasse definiert werden, wie z.B.

```
1 <x:Array />
```

Listing 2.7: XAML Codebeispiel Array

2.6.4 Title und Icon

Des weiteren können noch folgende Eigenschaften eingetragen werden, um die Anwendung noch zu erweitern, z.B. kann ein Titel für die Anwendung vergeben werden, der wie folgt lauten kann:

```
1 Title="Visualisierung"
```

Listing 2.8: XAML Code Titel

Um die Anwendung noch weiter zu verschönern, kann man ein kleines Icon, ob selbst gebastelt oder ein diverses Bild, einfügen, dass wie folgt definiert wird:

```
1 Icon="Bilder/Icon.ico
```

Listing 2.9: XAML Code Icon

Das Bild muss im gleichen Verzeichnis liegen, wie die .xaml Datei und es muss die Datei Endung .ico haben.

2.6.5 Controls

Kindelemente werden aus der Gruppe von Controls, den Panels definiert. Das können folgende Panels sein: Canvas, Grid, TabPanel und viele weitere Panels. In den folgenden Abschnitten werden die beiden Panels Grid und Canvas weiter betrachtet.

2.6.5.1 Grid

Durch das Schlüsselwort <Grid> wird ein Kindelement eingeleitet, wie z.B. das Kindelement zur Bestimmung von der Höhe und Breite der Anwendung. Die Größe ist in Pixel angegeben. Das kann dann wie folgt aussehen:

```
1 <Grid Height="625" Width="912">
```

Listing 2.10: XAML Code Grid

Durch ein weiteres Schlüsselwort aus dem Grid dem ColumnDefinition können Spalten definiert werden, wie z.B.

```
<Grid.ColumnDefinition>
        <ColumnDefinition Width="912*" />
        <ColumnDefinition Width="0*" />
</Grid.ColumnDefinitions>
```

Listing 2.11: XAML Code Grid Spaltendefinition

Die Spalten definiert man durch die Breite und die Zeilen werden durch die Höhe definiert. Im aufgeführten Beispiel werden zwei Spalte definiert. Die Breite ist im ersten Fall 912 Pixel, der Stern bedeutet, dass der gesamt verfügbare Platz zwischen diesen beiden Elementen aufgeteilt wird. Die Definition wird mit dem gleichem Schlüsselwort beendet, womit das Kindelement eingeleitet wurde, es kommt bloß ein kleiner Zusatz dazu, der Slash(/).

2.6.5.2 Canvas

Ein weiteres Kindelement ist z.B. das Canvas. Damit die absolute Positionierung bzw. Darstellung von Grafiken erreicht werden kann , definiert man das Steuerelement Canvas. Canvas kann beliebig viele Objekte enthalten, wie z.B. ein Rectangle (Rechteck). Die Positionierung der Elemente in Canvas erfolgt durch eine absolute Position, die erreicht werden kann, durch die Attached Properties: Top(Oben), Left(Links), Bottom(Unten) und Right(Rechts). Die Größe des Canvas ist abhängig vom Elternelement, dass bedeutet, dass das Canvas nicht größer sein kann, als das Vaterelement. Die Größe des Canvas muss explizit angegeben werden, ansonsten ist das Canvas nicht sichtbar. Im Folgenden ein kleines Beispiel, wie so ein Canvas aussehen kann.

```
<Canvas Width="792.468" Height="448.94">
        <Rectangle Width="97.948" Height="12.7973" Canvas.Left="
            634.407" Canvas.Top="333.624" Stretch="Fill" Fill="#
            FFFFFFFF"/>
</Canvas>
```

Listing 2.12: XAML Code Canvas Rectangle

Das Canvas, im angegebenen Beispiel, hat eine Höhe von 792.468 Pixeln und eine Breite von 448.94 Pixeln. In dem Canvas wird ein Objekt definiert, ein Rechteck, das die Höhe von 97.948 Pixeln und die Breite von 12.7973 hat. Das Rechteck wird im Canvas an der Position Links bei 634.407 Pixeln und Oben bei 333.624 Pixeln positioniert. Des Weiteren wird das Rechteck

durch die Option Stretch=Fill gestreckt, um den Layoutbereich auszufüllen, damit wird das Seitenverhältnis nicht eingehalten und das Rechteck trägt die Farbe weiß, was durch die Angabe Fill="#FFFFFFFF" definiert wird. Das Kindelement wird wieder mit dem Schlüsselwort Canvas und dem Slash beendet.

Ein weiteres Element, das in Canvas eingefügt werden kann, ist das Objekt Path(Linie). Ein kleines Beispiel, wie so ein Path definiert werden kann.

```
1 <Path Width="0.32" Height="20.1227" Canvas.Left="303.7" Canvas.
    Top="420.416" Stretch="Fill" StrokeThickness="0.32"
    StrokeLineJoin="Round" Stroke="#FF000000" Data="M
    303.86,420.576 L 303.86,440.379"/>
```

Listing 2.13: XAML Code Canvas Path

Der Path Data wurde in der Path Markup Syntax definiert, dass bedeutet, das die Syntax fast vollständig der Syntax in Scaleable Vector Graphics (SVG) entspricht, die zum Definieren von Path verwendet wird, dass sieht man ganz gut an dem kleinem Beispiel. Das Beispiel definiert eine Linie mit einer Breite von 0.32 Pixeln und einer Höhe von 20.1227 Pixeln. Die Linie wird im Canvas an der Position Links bei 303.7 Pixeln und Oben bei 420.416 Pixeln angeordnet. Die Linie wird gestreckt, das gibt das Attribut Fill bei Stretch (MSDN, 2011c) an. Die Randstärke wird durch die Eigenschaft StrokeThickness (MSDN, 2011b) angegeben, in diesem Fall 0.32. Das Attribut StrokeLineJoin=Round (MSDN, 2011a) gibt an, welche Linienverbindung genommen werden soll, in diesem Fall wird eine runde Liniendarstellung dargestellt. Und zu guter Letzt wird die Linie in der Farbe schwarz dargestellt. Die Eigenschaft Data weißt die Geometrie zum Path zu, dass bedeutet das einige Kommandos für die Linie definiert werden, wie z.B. M, wie Move, damit wird die Startposition festgelegt, L, wie Linie, damit wird eine Linie von der Startposition zur Zielposition gezeichnet. Also das bedeutet für das Beispiel das die Linie an der Position 303.86,420.576 Pixeln anfängt und an der Position 303.86,440.379 endet.

3. Fallbeispiel „HAW-Logistics-System“

Das HLS ist ein System das die Planung der Transporte und die Abrechnung übernimmt. Das HLS erhält Aufträge vom Kunden, daraufhin erfolgt die Transportplanung. Da das HLS keinen eigenen Fuhrpark hat, beauftragt das HLS externe Transportdienstleister, die im Laufe der Transportplanung entsprechende Unterbeauftragungen erhalten. Das HLS arbeitet global, deshalb kommen folgende Verkehrswege für den Transport in Frage: Land, See und Luft. (Vgl. (Stefan Sarstedt, 2010a))

3.1. Ablauf des HAW-Logistics-System

In Abbildung 3.1 ist der Ablauf des HLS zu sehen. Das System empfängt diverse Sendunganfragen von diversen Kunden, z.B.

- Sendungsanfrage A mit
  - Saft, Mixer und Tomaten in einer bestimmten Menge
- Sendungsanfrage B mit
  - Autoreifen und Salat in einer bestimmten Menge

Die Sendungsanfragen, die im System eintreffen, haben ein Start und ein Ziel. In diesem Fall haben die Sendungsanfrage A und B das selbe Start und Ziel, die Ware soll von Hamburg nach Frankfurt transportiert werden. Nun geschieht folgendes, das HLS sucht alle Sendungsanfragen durch, die das selbe Start und Ziel haben und etwa zur selben Zeit am Ziel eintreffen sollten, packt dann die Sendungspositionen passend in Container, die gesamten Sendungsanfragen kommen oft nicht in den selben Container und deswegen werden vereinzelt Sendungspositionen in andere Container gepackt, damit die Kapazität in den Containern voll ausgelastet wird. In diesem Fall kommt die Sendungsposition Salat von der Sendungsanfrage B in die Frachteinheit A und die Sendungsposition Mixer von der Sendungsanfrage A in die Frachteinheit B. Nachdem das Packen der Container erfolgt ist, werden die Container zu einer Sendung zusammengefasst, da die Container das selbe Start und Ziel haben. Zu jeder Sendung gehört

ein Frachtbrief, der wird nun vom System erstellt. Jede Sendung hat Sendungsverfolgungsereignisse, damit das Unternehmen oder auch der Kunde die Sendung nachverfolgen kann. Mehrere Frachtführer, die die Sendung transportieren, müssen einem Transportplan befolgen, der diverse Frachtaufträge beinhaltet, die verschiedene Frachtführer ausführen. Nachdem die Sendung das Ziel erreicht hat, erstellt das System eine Kundenfrachtabrechnung, die an den Kunden geht. Nach der Erstellung der Kundenfrachtabrechnung bekommen diverse Frachtführer eine Gutschrift über den Transport der Sendung.

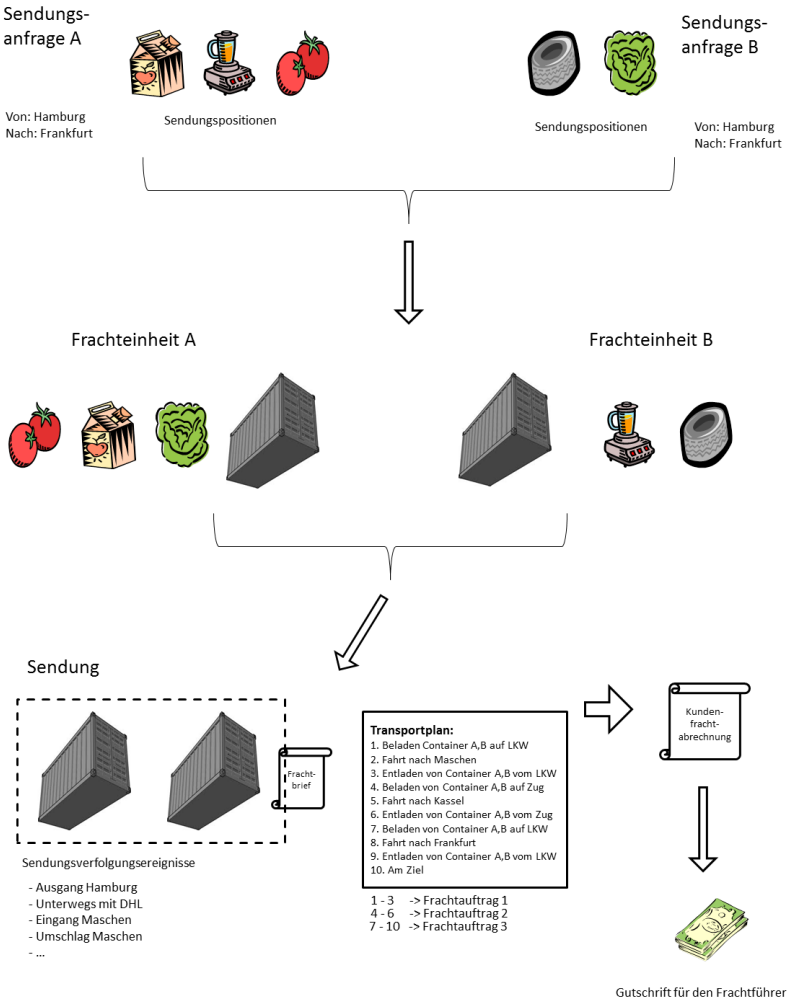

Abbildung 3.1.: HLS - Ablauf im HAW-Logistics-System

3.2. Softwarearchitektur - Quasar

Das HLS wurde nach der Standardarchitektur Quasar entwickelt, indem die Anwendung von der Technik getrennt wird. Die Softwarearchitektur beschreibt Komponenten und Schnittstellen, aus denen das HLS besteht und deren Zusammenspiel. Die Standardarchitektur Quasar (entnommen aus (Stefan Sarstedt, 2010d), S. 19) besteht aus drei Architekturen: A-Architektur, T-Architektur und TI-Architektur. (Vgl. (Siedersleben, 2004), S. 145-163)
Die **A-Architektur** oder auch Architektur des Anwendungskerns genannt, beschreibt die Bestandteile des Anwendungskerns. Der Anwendungskern besteht aus Anwendungskomponenten, kurz auch A-Komponenten genannt. Im Abschnitt 3.3 wird auf die Bestandteile der A-Architektur eingegangen.
Die **T-Architektur** beschreibt die technische Architektur des HLS, die den Umgang mit der Technik sicherstellt. Die T-Architektur enthält alle Komponenten, die von der A-Architektur unabhängig sind. Sie verbindet die A-Architektur mit der TI-Architektur. Auf die Bestandteile der T-Architektur wird im Abschnitt 3.5 eingegangen.
Die **TI-Architektur** beschreibt die technische Infrastruktur des HLS. Die Bestandteile der TI-Architektur sind konkrete physikalischen Geräte. Auf die Bestandteile der TI-Architektur wird im Abschnitt 3.7 eingegangen.

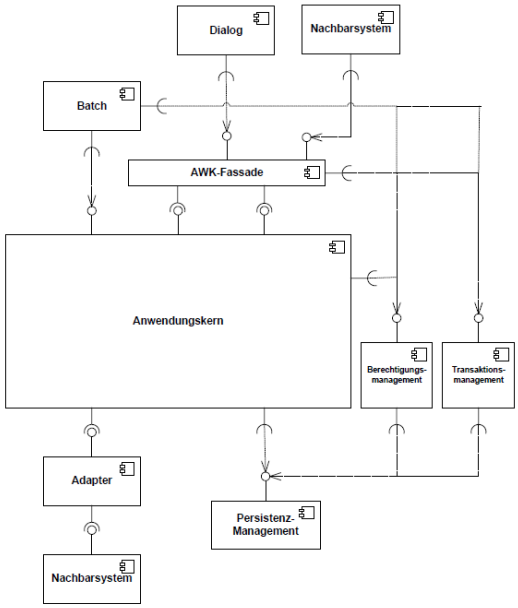

Abbildung 3.2.: HLS - Softwarearchitektur Quasar

3.3. A-Komponenten

Folgende Abbildung 3.3 zeigt die A-Architektur mit ihren A-Komponenten. Im Weiteren wird darauf eingegangen, was in den folgenden Anwendungskomponenten enthalten ist und welche Funktion sie haben und welche Annahmen und Abgrenzungen für das System getroffen wurden sind.

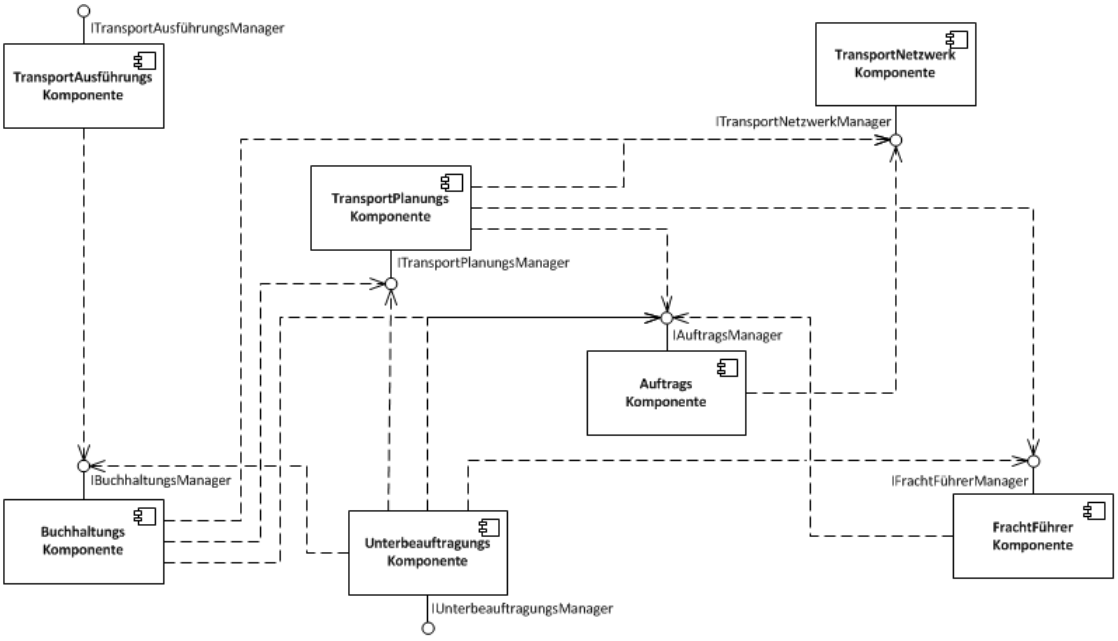

Abbildung 3.3.: HLS - Komponentendiagramm

3.3.1 Auftragskomponente

Die Auftragskomponente enthält folgende Klassen Geschäftspartner, Sendungsanfrage und Sendungsposition.
Im System müssen Geschäftspartner definiert sein, damit die Transportplanung überhaupt stattfinden kann. Ein Geschäftspartner kann ein Auftraggeber, Frachtführer oder Empfänger sein. Der Geschäftspartner ist durch folgende Attribute definiert:

- Identifizierungsnummer, die sich folgendermaßen zusammensetzt: GP-<Art des Geschäftspartners>-n, die Art des Geschäftspartners kann folgende Werte annehmen:
  - AG für Auftragsgeber,

  - FRA für Frachtführer und
  - EMP für Empfänger

  Der Platzhalter n stellt eine aufsteigende ganzzahlige Zahl dar.
- Name,
- Vorname,
- Telefonnummer, in der Form +<Länderkennzahl><Netzkennzahl><Rufnummer>,
- E-Mail-Adresse,
- Adresse, die sich aus folgenden Attributen zusammensetzt:
  - Straße,
  - Hausnummer,
  - Postleitzahl,
  - Ort,
  - Land,
  - Ländercode (DIN 3166-1 (siehe Glossar)),
  - geografische Daten (DD° MM’ SS.SS”)

Die Sendungsanfrage wird vom System unterstützt und stellt den Kundenauftrag zum Transport ein oder mehrerer Waren dar, der durch das HLS organisiert werden soll. Eine Sendungsanfrage wird wie folgt definiert:

- eindeutige Identifizierungsnummer in der Form HLS-SNA-n, n stellt eine natürliche fortlaufende Zahl dar,
- einen Wunschtermin für die Abfahrt, bestehend aus einem Datum und einem Zeitfenster von vier Stunden,
- Angaben zum Zahlungsweg (IBAN und SWIFT-BIC)

Eine Sendungsanfrage hat einen Abgangs- und Zielort, die durch Lokationen repräsentiert werden und einen Auftraggeber und Empfänger. Jede Sendungsanfrage hat eine Kundenfrachtabrechnung, die auch die Kundenrechnung enthält. Eine Sendungsanfrage besteht aus ein oder mehreren Sendungspositionen, die beliebige Waren darstellen. Eine Sendungsposition wird definiert durch:

- einer Sequenznummer,
- eine Warenbeschreibung,
- eine Stückzahl,

- ein Bruttogewicht,
- Bruttovolumen

Der Auftraggeber erhält durch E-Mail oder per Post eine Auftragsbestätigung.

3.3.2 Buchhaltungskomponente

Die Buchhaltungskomponente enthält folgende Klassen Rechnung und Gutschrift.
Nach einer gemeinsam mit dem Kunden erstellten Sendungsanfrage soll das System einen Transportplan erstellen, natürlich auch gleichzeitig für andere Sendungsanfragen, und die Kosten des Transports berechnen. Beim Befahren einer Transportbeziehung werden durch das System, die dort hinterlegte Distanz und mittlere Geschwindigkeit des Transportmittels als Grundlage für die Kostenberechnung verwendet. Für jedes Transportmittel sind im System Kosten hinterlegt, siehe Abschnitt 3.3 Transportnetzwerkkomponente unter Transportmittel. Die Kosten eines Transport auf einer Transportbeziehung berechnet sich wie folgt:

TKpTb = FK + (EK * K) + (DK * D) + (MK * AdtF)

TKpTb = Transportkosten pro Transportbeziehung
FK = Fixkosten
EK = Entfernungskosten
K = Kosten
DK = Dauerkosten
D = Dauer
MK = Mengenkosten
AdtF = Anzahl der transportierten Frachteinheiten

Die Kosten des Transportplans ergibt sich aus der Summe der Kosten für die einzelnen Transportbeziehungen, dass dann wie folgt aussieht:

TKG = $\sum$ TKpTb

TKG = Transportkosten Gesamt

Für die Tätigkeit der HAW-Logistics erhebt diese einen Zuschlag auf die Gesamttransportkosten, der im System variabel gestaltet ist. Als Basis sollen vom Auftragnehmer zunächst 15% angenommen werden.
Beim Erreichen des Ziels und bei der Sendungsverfolgungsereignisart „Das Erreichen des

Zielortes“erfolgt die Erstellung einer Rechnung durch das System für den jeweiligen Auftraggeber. Jede Rechnung hat folgende Bestandteile:

- eine eindeutige Rechnungsnummer, in Form von R-n, n stellt eine fortlaufende natürlich Zahl dar und beginnt bei 1,
- die Sendungsnummer,
- den Transportweg,
- die transportierten Waren (die Frachteinheiten und die Angaben zu den Sendungspositionen),
- den Rechnungsbetrag

Der Rechnungsbetrag für den jeweiligen Auftraggeber wird wie folgt berechnet:

RBpA = (TKG / AdtF) * AAtF

RBpA = Rechnungsbetrag pro Auftraggeber
AAtF = Anzahl der für diesen Auftraggeber transportierten Frachteinheiten

Die Rechnung wird durch das System im PDF-Format erstellt und nach Erstellung an den Auftraggeber per E-Mail übermittelt.
Nachdem der beauftragte Frachtführer seine Transportaktivitäten innerhalb des Transportplans erfolgreich durchgeführt hat, überweist das System dem Frachtführer die angefallenen Transportkosten. Diese Überweisung der Gutschriften an die Frachtführer erfolgt durch das System um 3:00 Uhr nachts, welches die angeschlossenen Transporte des vorherigen Tages berücksichtigt. Das System verschickt die Gutschriften per E-Mail. Der Aufbau der E-Mails für die Gutschriften ist wie folgt definiert:

- Betreff: Überweisung,
- Inhalt:
  - Überweisung von HLS an <Name des Frachtführers>,
  - Adresse: <Adresse des Frachtführers>,
  - Verwendungszweck: Gutschrift für Frachtauftrag <Frachtauftragsnummer> von <Quelle< nach <Ziel>,
  - Betrag: <Betrag in €>

Jede Gutschrift wird durch eine eindeutige Nummer gekennzeichnet, die wie folgt aussieht: G-n, n der der Platzhalter für eine fortlaufende natürliche Zahl, die bei 1 beginnt. Die E-Mail-Adresse der Bank für die Gutschriften wird im System variabel gestaltet.

3.3.3 Frachtführerkomponente

Die Frachtführerkomponente enthält folgende Klassen Frachtführer und Fahrzeug.
Fahrzeuge sind bewegliche Ressourcen, die Waren von einer Quelllokation zu einer Ziellokation transportieren. Jedes Fahrzeug gehört zu einem Transportmittel. Ein Fahrzeug wird wie folgt definiert:

- einer eindeutigen Identifizierungsnummer in der Form „FZG-n“, n stellt eine fortlaufende natürliche Zahl dar,
- einem Ort

Ein Frachtführer ist ein Geschäftspartner, der wie folgt im Abschnitt 3.3 Auftragskomponente definiert wird. Im Abschnitt 3.3 Unterbeauftragungskomponente wird die Dienstleistung gegenüber des HLS erklärt. Die Vorausbuchung durch das HLS und die Benutzung eines Transportmittels durch den Frachtführer ist die Voraussetzung den Vertrag zu erfüllen, dies wird im Abschnitt 3.3 detaillierter erklärt. Der Frachtführer ist auch verpflichtet bei der Ausführung des Transports Sendungsverfolgungsereignisse an das System zu übermitteln, welches wie folgt definiert wird, wird im Abschnitt 3.3 Transportausführungskomponente erklärt.

3.3.4 Unterbeauftragungskomponente

Die Unterbeauftragungskomponente enthält folgende Klassen Frachtauftrag und Frachtbrief.
Ein Frachtführer ist dafür verantwortlich, ein oder mehrere Transportaktivitäten in Auftrag von HAW-Logistics durchzuführen. Für eine Teilmenge von Transportaktivitäten erfolgt automatisch eine Unterbeauftragung jeweils eines Frachtführers. Dieser ist dann beispielsweise für das Beladen, Entladen und den Transport der Frachteinheiten mittels eines LKWs auf einer bestimmten Strecke verantwortlich. Für den Folgeabschnitt erfolgt die Beauftragung eines anderen Frachtführers.
Diese Unterbeauftragung erfolgt in Form eines Frachtauftrags für den jeweiligen Frachtführer. Frachtaufträge werden im System erstellt und elektronisch mit Hilfe des Message-Queueing-Systems an den Frachtführer übermittelt. Ein Frachtauftrag wird durch folgende Attribute definiert:

- einer eindeutigen Identifizierungsnummer, in der Form „<Sendungsnummer>/FRAUFT-n“, Sendungsnummer repräsentiert die Nummer der zu transportierenden Sendung und n stellt eine fortlaufende natürliche Zahl dar
- einem Kontrollkästchen, ob Transportplan bestätigt wurde,

- dem Bestätigungszeitpunkt

Eine Unterbeauftragung muss unmittelbar nach der erstmaligen Freigabe des Transportplans erfolgen. Der Eingang eines Frachtauftrages beim Frachtführer muss über das Message-Queueing-System elektronisch bestätigt werden.
Jeder Frachtführer führt einen Frachtbrief mit sich. Dieser Frachtbrief wird vom System erstellt und parallel zum Frachtauftrag, per E-Mail an den Frachtführer übermittelt. Ein Frachtbrief stellt ein Beförderungsdokument für den Frachtvertrag nach dem Handelsgesetzbuch (§ 407) dar, welches der Frachtführer mit den Gütern mit sich führt. Ein Frachtbrief enthält folgende Angaben:

- Ort und Tag der Ausstellung,
- Name und Anschrift des Absenders,
- Name und Anschrift des Frachtführers,
- Name und Anschrift des Empfängers,
- Stelle und Tag der Übernahme des Gutes,
- die allgemein anerkannte Bezeichnung der Waren,
- das Rohgewicht oder die anders angegebene Menge des Gutes,
- die vereinbarte Fracht und die bis zur Ablieferung anfallenden Kosten, sowie einen Vermerk über die Frachtzahlung

3.3.5 Transportausführungskomponente

Die Transportausführungskomponente enthält folgende Klassen Sendungsverfolgungsereignis und Transportaktivitätsausführungsstatus.
Auf dem Transportweg können verschiedene Sendungsverfolgungsereignisse auftreten, dadurch ist der aktuelle Status der Sendung durch das System nachvollziehbar. Das Sendungsverfolgungsereignis setzt sich aus folgenden Attributen zusammen:

- einer Sendungsnummer,
- der Ort,
- ein Zeitstempel,
- ein Textfeld für Nachrichteninhalte,
- einer Art, die folgende Werte annehmen kann:
  - Der Eingang in eine Lokation
  - Der Ausgang aus einer Lokation

  - Der Umschlag innerhalb einer Lokation
  - Das Laden einer Frachteinheit
  - Das Entladen einer Frachteinheit
  - Regelmäßige oder unregelmäßige Meldungen während des Transports mit geografischen Koordinaten
  - Das Erreichen des Zielortes

Sendungsverfolgungsereignisse werden über das Message-Queueing-System mit externen Partnern (Frachtführern und Lokationen) ausgetauscht.
Der Status einer Transportaktivität kann verschiedene Werte annehmen, die wie folgt definiert sind:

- „Bereit zur Ausführung“: Die Transportaktivität ist bereit zur Ausführung
- „In Ausführung“: Die Transportaktivität wird gerade durchgeführt
- „Ausführung abgeschlossen“: Die Transportaktivität wurde abgeschlossen

3.3.6 Transportnetzwerkkomponente

Die Transportnetzwerkkomponente enthält folgende Klassen Lokation, Transportbeziehung, Transportmittelbenutzung und Transportmittel.
Durch das HLS soll ein Transportnetzwerk verwaltet werden. Das Transportnetzwerk besteht aus Lokationen, die Orte darstellen. Eine Lokation wird durch die Adresse repräsentiert. Lokationen haben eine:

- eindeutige Identifizierungsnummer, die sich aus folgenden Bestandteilen zusammensetzt:
  - Ländercode der Lokation,
  - einem Slash(/),
  - einer fortlaufenden ganzzahligen aufsteigenden Nummer
- einen Ort

Zwischen zwei Lokationen kann eine Transportbeziehung bestehen, einer Quelllokation und einer Ziellokation, in der Form „TB-<StartLokationsId>-<ZielLokationsId>“, wobei <StartLokationsId> und <ZielLokationsId> durch die Lokationsdefinition repräsentiert wird. Zwischen der Transportbeziehung besteht eine Entfernung, die durch eine Distanz in der Klasse Transportbeziehung repräsentiert wird. Auf dieser Transportbeziehung wurde ein Transportmittel zugeordnet. Wenn keine Transportbeziehungen zwischen Lokationen bestehen, werden diese

auch nicht im Planungsprozess berücksichtigt.
Ein Transportmittel setzt sich wie folgt zusammen:

- eine eindeutige Nummer, in der Form TM-n, n stellt eine ganzzahlige aufsteigende Nummer dar
- einem Verkehrszweig bzw. Verkehrsweg, See, Luft oder Land
- einer Beschreibung,
- ein Eigenschaftsfeld für die textuelle Eingabe von weiteren Eigenschaften,
- eine Geschwindigkeit (niedrige, mittlere oder hohe), wichtig für die Berechnung der Kosten und der Dauer,
- einer Kapazität in Forty-foot Equivalent Unit (FEU) oder Twenty-foot Equivalent Unit (TEU),
- Kosten, wie
  - Fixkosten pro Fahrt,
  - Entfernungskosten pro km,
  - Dauerkosten pro min,
  - Mengenkosten pro FEU/TEU

Ein Transportmittel repräsentiert eine Fahrzeugklasse.
Ein Transportmittel kann auf mehrere Transportbeziehungen eingesetzt werden. Durch die Verwendung eines Transportmittels auf einer Transportbeziehung wird ein

- Gültigkeitszeitraum, von und bis,
- eine Distanz,
- eine Dauer

bestimmt. Die Dauer setzt sich aus

- Distanz,
- mittlerer Geschwindigkeit des Transportmittels

zusammen. Ein Transportmittel kann von mehreren Frachtführern verwendet werden.

3.3.7 Transportplanungskomponente

Die Transportplanungskomponente enthält folgende Klassen Route, Kapazitätsbenutzung, Sendung, Transportplan, Transportaktivität, Frachteinheit und Kapazität.
Eine Route wird definiert durch Gesamtkosten, Gesamtdistanz in km und Gesamtdauer in min.
Die Kapazitätsbenutzung setzt sich wie folgt zusammen:

- einer Startzeit,
- Kosten,
- einer Distanz in km,
- einer Dauer in min

und ist gekoppelt an einer Start- und Ziellokation, einer Route, an Frachteinheiten und an eine Kapazität.
Das Ergebnis des Planungsprozesses ist der Transportplan, welcher Transportaktivitäten enthält. Aus einem Planungsprozess können mehrere Transportpläne generiert werden. Der Auftraggeber wählt einen Transportplan aus. Der Preis hängt von der Schnelligkeit des Transports ab. Die Kapazitäten bzw. Ressourcen werden schon im Voraus bei den Frachtführern gebucht, die Dienstleistungen werden in Verträgen festgehalten. Eine Transportplanung kann nur erfolgen, wenn die Kapazitäten im Voraus gebucht wurden sind. Der Transportplan wird um 2:00 Uhr nachts durch das System erstellt und Berücksichtigt alle Sendungsanfragen aller Auftraggeber des vorherigen Tages. Ein Transportplan gilt für ein oder mehrere Frachteinheiten. Jedem Transportplan wird ein Status zugeordnet, dieser sieht wie folgt aus:

- „In Planung“: Der Transportplan befindet sich in Planung
- „Freigegeben“: Der Transportplan ist für die Ausführung freigegeben
- „Gesperrt“: Die Ausführung des Transportplans ist gesperrt
- „Ausgeführt“: Der Transport wurde abgeschlossen

Der Transportplan muss manuell durch den Disponenten im System freigegeben werden und dann kann die Ausführung des Transports beginnen bzw. fortgesetzt werden. Ein Transportplan kann gesperrt werden, wenn der Transportplan noch nicht ausgeführt wurde, d.h. noch nicht beendet wurde, das Sperren erfolgt durch den Disponenten. Durch ein erneutes Freigeben kann der Transportplan entsperrt werden.
Eine Frachteinheit ist eine Zusammenstellung von Waren, die einen Transportbedarf darstellt, die durch die gesamte Transportkette transportiert werden muss. Frachteinheiten werden im Transportplanungsprozess aus ein oder mehreren Sendungsanfragen zusammengestellt, um die Waren für den Transport geeignet zu bündeln. Ein Frachteinheit setzt sich aus folgenden Bestandteilen zusammen:

- einer eindeutigen Bezeichnung, die sich wie folgt zusammensetzt:
  - „FE-<Nummer>“, wobei <Nummer> eine fortlaufende natürliche Zahl ist und bei 1 beginnt, z.B. „FE-14“
- einem Bruttogewicht,

- einem Nettogewicht,
- einem Typ TEU oder FEU

Eine Frachteinheit ist im System ein TEU oder FEU, ein 20- oder 40-Fuß-Standardcontainer. Eine Frachteinheit kann verschiedene Waren von verschiedenen Auftraggebern beinhalten, wenn sie das selbe Start und Ziel besitzen. Das System bildet beim Transportplanungsprozess so wenig Frachteinheiten wie möglich, unter der Verwendung eines geeigneten Algorithmus. Ein Transportplan besteht aus Transportaktivitäten. Transportaktivitäten definieren den Einsatz von Transportkapazitäten zur Durchführung der Transportbedarfe. Eine Transportaktivität wird wie folgt definiert:

- einer Sequenznummer,
- einer Art, die folgende Werte annehmen kann:
  - „Laden“: Eine Frachteinheit wird auf eine Transportkapazität geladen
  - „Transport“: Null oder mehrere Frachteinheiten werden durch eine Transportkapazität transportiert
  - „Entladen“: Eine Frachteinheit wird von einer Transportkapazität entladen
- einem Startzeitpunkt,
- einem Endzeitpunkt

Eine Transportaktivität kann sich auf mehrere Frachteinheiten beziehen, z.B. kann eine Transportaktivität der Art Laden mehrere Frachteinheiten laden. Die Sequenznummer gibt eine Reihenfolge der Transportaktivitäten vor. Jedoch schränkt dies nicht die mögliche parallele Ausführung ein, z.B. paralleler Ladevorgänge verschiedener Frachteinheiten. Bei der Transportaktivität der Art „Transport“ist eine Transportbeziehung zugeordnet, auf der der Transport der Frachteinheit erfolgen soll bzw. erfolgt ist. Bei der Transportaktivität der Arten „Laden“und „Entladen “sind einer Lokation zugeordnet, an der die Transportaktivität ausgeführt wird.
Bei der Planung der Transportaktivitäten eines Transportplans müssen die beschriebenen Kapazitätseinschränkungen berücksichtigt werden. Ein Transport kann nur stattfinden, wenn ein Fahrzeug, d.h. ein konkretes Transportmittel, mit der entsprechenden Kapazität, d.h. mögliche Anzahl zu transportierender Container, Fahrzeuge am richtigen Ort, etc., im Transportzeitraum verfügbar ist. Die Verfügbarkeit von Fahrzeugen wird durch die Vorausbuchung bei dem Transportdienstleister sichergestellt. Jede Kapazität ist durch:

- eine eindeutige Nummer der Form „KAP-FZG-n“, n ist eine natürliche Zahl,
- einem Zeitpunkt, wann die Kapazität verfügbar ist,

- einem Status, der folgende Werte annehmen kann:
  - „Frei“: Die Kapazität ist verfügbar und frei.
  - „Reserviert“: Die Kapazität ist verfügbar, aber schon für eine andere Fracht reserviert.
  - „Belegt“: Die Kapazität ist nicht mehr verfügbar.

identifiziert.
Frachteinheiten, für die ein gemeinsamer Transportplan erstellt wurde, werden in einer Sendung gemeinsam transportiert. Eine Sendung bündelt somit mehrere Frachteinheiten mit dem selben Abgangs- und Zielort. Jede Sendung ist durch eine eindeutige Sendungsnummer identifiziert, deren Aufbau folgendermaßen aussieht: „HLS-SND-<Nummer>“, wobei <Nummer> eine fortlaufende natürliche Zahl ist und bei 1 beginnt, z.B. „HLS-SND-10“.

3.3.8 Annahmen

Folgende Annahmen werden bzw. werden nicht im System realisiert:

1 Flugzeuge werden im System derzeit nicht eingesetzt.
2 Das System führt nur eine Transportplanung von festen und unverderblichen Waren durch.
3 Eine Frachteinheit kann waren von verschiedenen Auftraggebern bündeln, wenn sie denselben Start- und Zielort besitzen.
4 Es werden keine weiteren Kosten angenommen, außer von der Transportaktivitätsart „Transport“

3.3.9 Ausgrenzungen

Für dieses System ist nur eine Abgrenzung definiert, dass die Prüfung des Bezahleingangs ausgeschlossen ist.

3.4. A-Komponenteninnensicht und -außensicht

Das Innenleben einer A-Komponenten im Anwendungskern kann sehr komplex sein. Im HLS sind alle A-Komponenten gleich strukturiert. Jede Innensicht einer A-Komponente wird mit folgende Bestandteilen definiert (vgl. (Stefan Sarstedt, 2010b), Kapitel 3, S. 152):

- Anwendungsfall,

- Verwalter,
- Entitätstyp,
- Datentyp

Die **Anwendungsfall**klasse enthält entitätsübergreifende Anwendungsfalllogik, was unter anderem Berechnungen, Aktionen und Delegationen an andere Anwendungsfälle und Komponenten beinhaltet. Die **Verwalter**klasse kümmert sich um das Lesen, Suchen, Aktualisieren und Löschen in den Entitätstypen. Die **Entitätstypen** repräsentieren die Entitäten, die entitätsspezifische Logik enthält und die **Datentypen** repräsentieren die Datentypen, die datenspezifische Logik enthält.
Die Innensichten der einzelnen A-Komponenten des HLS sind im Anhang A zu finden. Die Außensicht ist die Sicht von Außen, die von Außen als Black-Box betrachtet wird, nur die Schnittstellen der Komponente sind zu sehen. Die Schnittstellen der Komponenten sind in den Bildern der Innensichten im Anhang A zu sehen.

3.5. T-Komponenten

T-Komponenten stellen einfache Schnittstellen zur Verfügung, die von der A-Komponente verwendet wird. Aus folgende T-Komponenten besteht die T-Architektur:

- Anwendungskern (AWK)-Fassade
  Die AWK-Fassade hält den Anwendungskern frei von Technik. Die Fassade stellt Schnittstellen für Zugriffe zur Verfügung, steuert die Transaktionen und Konvertiert Datentypen,
- Batch
  Externe Anwendung, die zeitgesteuert, jede Nacht die Transportplanung ausführt. Greift direkt auf den Anwendungskern zu ,
- Dialog
  Greift über die AWK-Fassade auf den Anwendungskern zu. Dialoge werden durch das Spring Framework realisiert,
- Berechtigungskomponente
  Im HLS nicht vorhanden,
- Transaktionsmanagement
  Verwaltete die Transaktionen im HLS,
- Adapter
  Stellt das Bindeglied zwischen A-Komponenten und T-Komponenten dar,

- Nachbarsystem
  Der Adapter greift auf das Nachbarsystem zu.
  Folgende Nachbarsysteme wurden im HLS realisiert:
  - Datenbank - SQL Server 2008, SQLite (lokale Datenbank),
  - ActiveMQ - MessageQueue Server,
  - log4Net - Logging von Anwendungsmeldungen,
  - NHibernate - Objekte in einer relationalen Datenbank zu speichern und wieder aus den Datensätzen Objekte zu erzeugen,
  - GMap (Google Maps) - Kartenmaterial für die Anwendung, zur Realisierung der Transportbeziehungen zwischen Lokationen,
  - itextsharp - PDF Generierung von Gutschriften für die Bank,
  - Moq - Mock Objekte bzw. Dummys,
  - Spring - Benutzeroberfläche, für die Erstellung von Dialogen,
  - Autofac - verwaltet die Abhängigkeiten zwischen den Klassen, Änderungen können leicht in die Anwendung eingepflegt werden, auch wenn HLS größer und komplexer wird.
- Persistenzmanagement
  Bietet die Funktionalität an, beliebige Entitäten zu persistieren und zu laden. Greift auf die Datenbank SQL Server 2008 zu. Für die Entwicklung kann, zu Testzwecken, die lokale Datenbank SQLite verwendet werden.

3.6. R-Komponenten

R-Komponenten sind Komponenten in denen die Fachlichkeit und die Technik vermischt wurden sind. Dies ist manchmal notwendig und in diesem Fall auch zulässig. Folgende R-Komponenten sind im HLS enthalten: MQS und Planungskonnektor.
Der MessageQueue Server ist dafür zuständig MessageQueue Nachrichten aus Entitäten zu generieren und der Planungskonnektor ist dafür zuständig eine Schnittstelle für entfernte Aufrufe darzustellen, die die Transportplanung über eine Batch starten kann.

3.7. TI-Architektur

In der Abbildung 3.4 sind alle Komponenten zu sehen, welche vom HLS verwendet werden. Folgende Komponenten sind zu sehen (vgl. Stefan Sarstedt (2010c), S. 17):

- Mail2Bank, generiert die Mail für die Bank, in der die Gutschrift angewiesen wird und schickt die Mail an die Bank,
- MS SQL Server 2008, die Datenbank, die alle Daten vom HLS hält,
- Apache ActiveMQ 5.4 Server, schickt Nachrichten raus an Frachtführer und Lokationen, die die Transportplanung betreffen

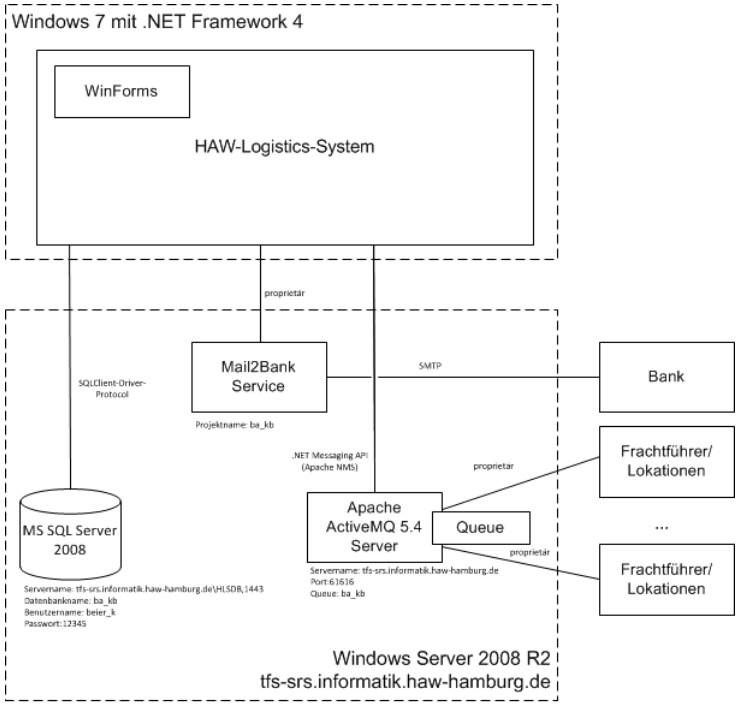

Abbildung 3.4.: HLS - TI-Architektur

3.8. Dialoge

Im Anhang A sind Bilder zu den Dialogen zu finden, die im HLS verwendet werden.

3.9. Software-Entwicklungs-Umgebungs-Architektur

Die Abbildung 3.5 zeigt die Software-Entwicklungs-Umgebungs-Architektur, in der das HLS entwickelt wurde. Im Folgendem wird darauf eingegangen, welche technischen Komponenten für die Software-Entwicklungs-Umgebungs-Architektur getroffen wurden sind (vgl. Stefan Sarstedt (2010c), S. 9).

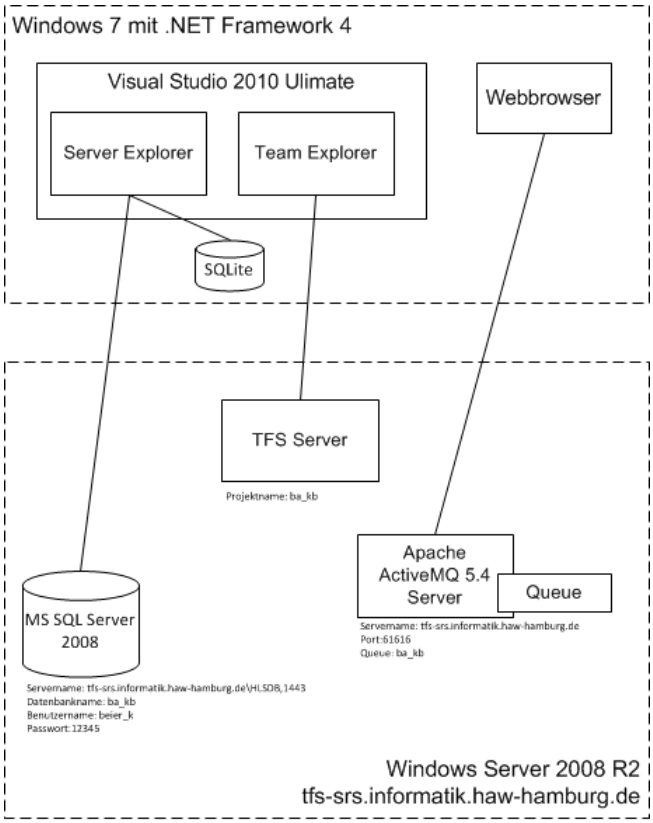

Abbildung 3.5.: HLS - Software-Entwicklungs-Umgebung-Architektur

Folgende technischen Komponenten enthält die Software-Entwicklungs-Umgebungs-Architektur

- Das Message-Queueing-System wird durch das Produkt Apache ActiveMQ 5.4 realisiert.
- Die Datenbank, die im HLS zum Einsatz kommt, ist der Microsoft SQL Server 2008.
- Die Entwicklungsumgebung in der das HLS entwickelt wurde, ist Microsoft Team Foundation Server 2010.
- Folgendes Framework wurde verwendet, dass Microsoft .NET Framework 4.0.
- Das HLS wurde in der Programmiersprache C# entwickelt.

4. Analyse

4.1. Anforderungen

Leitstand

A1 Der Leitstand muss folgende Bestandteile beinhalten

- Messung,
- Datenextrahieren,
- Interpretation,
- Darstellung

Messung

A2 Folgende Daten werden in Komponenten und Konnektoren gemessen:

- Fehler, Anzahl und Art des Fehlers mit Fehlerbeschreibung,
- Anzahl Aufrufhäufigkeit,
- Anzahl Datenbankzugriffe,
- Anzahl MessageQueue Nachrichten,
- Datendurchsatz,
- Durchlaufzeit

Die Anforderungen, was gemessen werden soll, entstanden durch das Anwenden des GQM-Ansatzes (siehe Anhang B).

A3 Die Messdaten werden in Nachrichten versendet.

A4 Der Datenaustausch erfolgt über den Leitstand und ein Messaging-System.

A5 Nachrichten haben folgendes Format:

- Version des Nachrichtenformats im Nachrichtenkopf „M0.9“
- eine eindeutige Identifizierungsnummer ,
- Name, der folgende Werte annehmen kann: Fehler, Zeit oder Anzahl,

- Art, beinhaltet den Komponentenname oder den Konnektornamen (Quell- und Zielkomponentenname),
- Beschreibung je nach Namen, die Zeit, Anzahl oder bei Fehlern die Details

Datenextrahieren

A6 Nachdem die Nachrichten verschickt wurden, werden die Nachrichten empfangen und extrahiert.

A7 Nach der Extrahierung werden die Daten in eine Datenbank geschrieben, für die Wiederverwendung.

Interpretation

A8 Die Daten werden wie folgt interpretiert:

- Zeiten, es werden Einzelzeiten, Gesamtzeiten und durchschnittliche Zeiten gebildet,
- Anzahl, es werden Anzahl, Gesamtanzahl und durchschnittliche Anzahl gebildet,
- Fehler, es werden die Fehler gezählt und die Details interpretiert in eine benutzerfreundliche Fehlererklärung

A9 Die interpretierten Daten werden für die Darstellung aufbereitet.

A10 Die Diagrammdaten für die Darstellung werden erstellt.

Darstellung

A11 Eine Benutzeroberfläche zeigt den aktuellen Zustand des HLS an, welches das Komponentendiagramm darstellt.

A12 Durch das Aktivieren der Komponente, durch das Klicken der Maus, erscheint ein neues Dialogfenster, welches statische Daten des letzten Durchlaufes zeigt.

A13 Die Messgrößen, die das Dialogfenster Komponente beinhaltet, wird unter der Anforderung A2 definiert.

A14 Durch das Aktivieren des Konnektors, durch das Klicken der Maus, erscheint ein neues Dialogfenster, welches statische Daten des letzten Durchlaufes zeigt.

A15 Die Messgrößen, die das Dialogfenster Konnektor beinhaltet, wird unter der Anforderung A2 definiert.

A16 Durch das Aktivieren, durch einen Mausklick, des Buttons "Verlauf" im Dialogfenster einer Komponente oder eines Konnektors öffnet sich ein neues Dialogfenster,

welches ein Diagramm enthält und die statistischen Daten anzeigt aus allen Durchläufen.

Technische

A17 Die Oberfläche der Anwendung wird mit dem Programmiermodell Windows Presentation Foundation (WPF) entwickelt.

A18 Die Anwendung wird auf dem Betriebssystem Windows 7 entwickelt.

A19 Das System wird mit der Entwicklungsumgebung Visual Studio 2010 und mit der Programmiersprache C# entwickelt.

4.2. Fachliches Datenmodell

In dem Fachlichen Datenmodell, welches in der Abbildung 4.1 zu sehen ist, sind folgende Entitäten zu sehen: Leitstand, Messung, Nachrichten, Datenextrahieren, Interpretation, Fehler, Zeit und Anzahl. Der Leitstand enthält Messung, Datenextrahieren und Interpretation, welches im fachlichen Datenmodell dargestellt wird. Bei der Entität Messung wird ein Wert gemessen, folgende Werte können gemessen werden: Fehler, Zeit oder Anzahl. Nach der Messung wird eine Nachricht generiert, die eines der Werte enthält. Nachdem die Nachricht versendet wurde, werden die Daten aus den Nachrichten extrahiert und die extrahierten Daten interpretiert, welches die Vorbereitung für die Darstellung darstellt. Und zu aller Letzt werden die Daten grafisch auf der Benutzeroberfläche dargestellt. Eine detaillierte Beschreibung der Entitäten Fehler, Zeit und Anzahl sind im Anhang B zu finden.

4.3. Fachliche Architektur und Schnittstellen zu Nachbarsystemen

Die Fachliche Architektur, die in der Abbildung 4.2 zu finden ist, zeigt eine Vorstrukturierung des Leitstands. Der Leitstand besteht aus drei Komponenten Messung, Datenextrahieren und Interpretation. Der Leitstand hat zwei Schnittstellen zu zwei Nachbarsystemen, die Datenbank und der Apache Active MQ Server.
Die Datenbankschnittstelle hat folgende Aufgabe, die extrahierten Daten aus den ankommenden Nachrichten in die Datenbank schreiben bzw. bei Bedarf Daten aus der Datenbank lesen. Die Apache Active MQ Schnittstelle ist dafür da, Nachrichten von dem untersuchenden System, in diesem Fall das HAW-Logistics-System (HLS), zu empfangen.

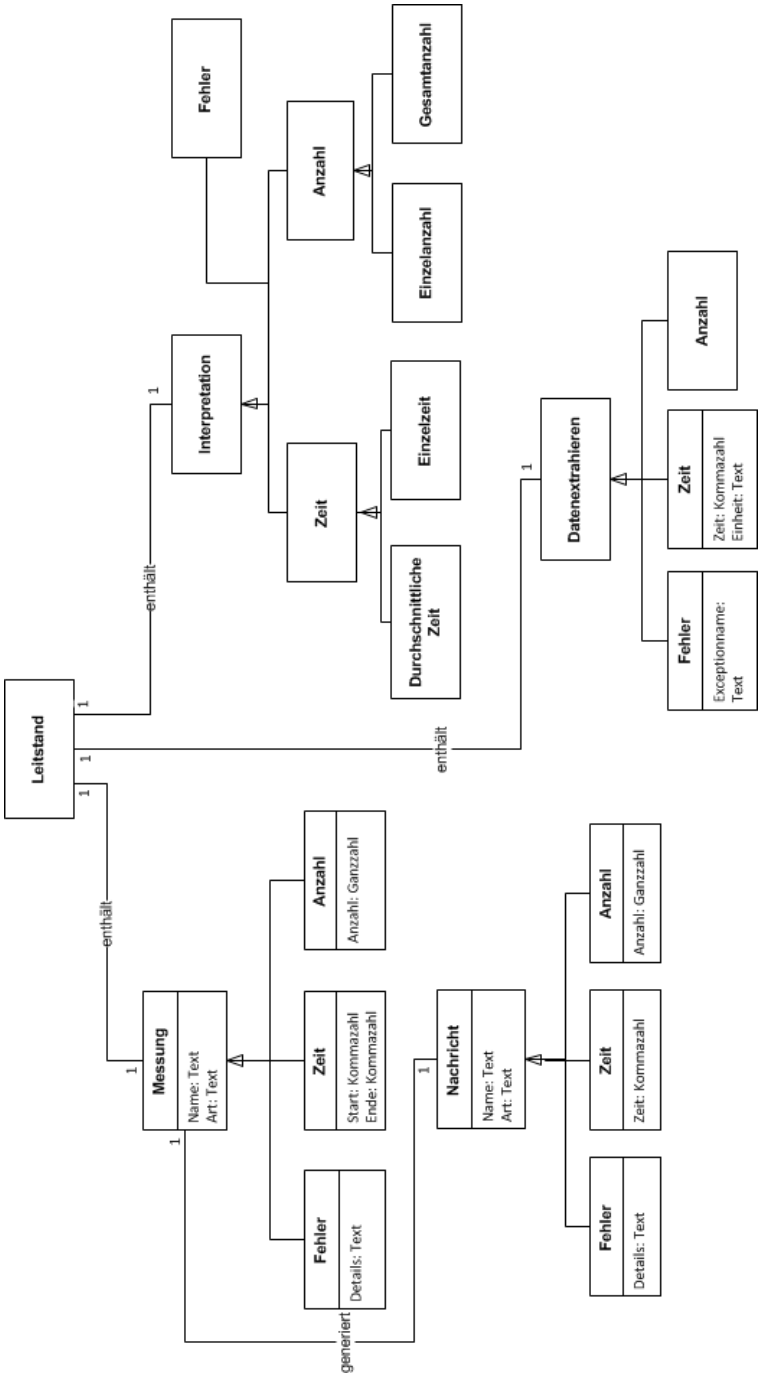

Abbildung 4.1.: Fachliches Datenmodell

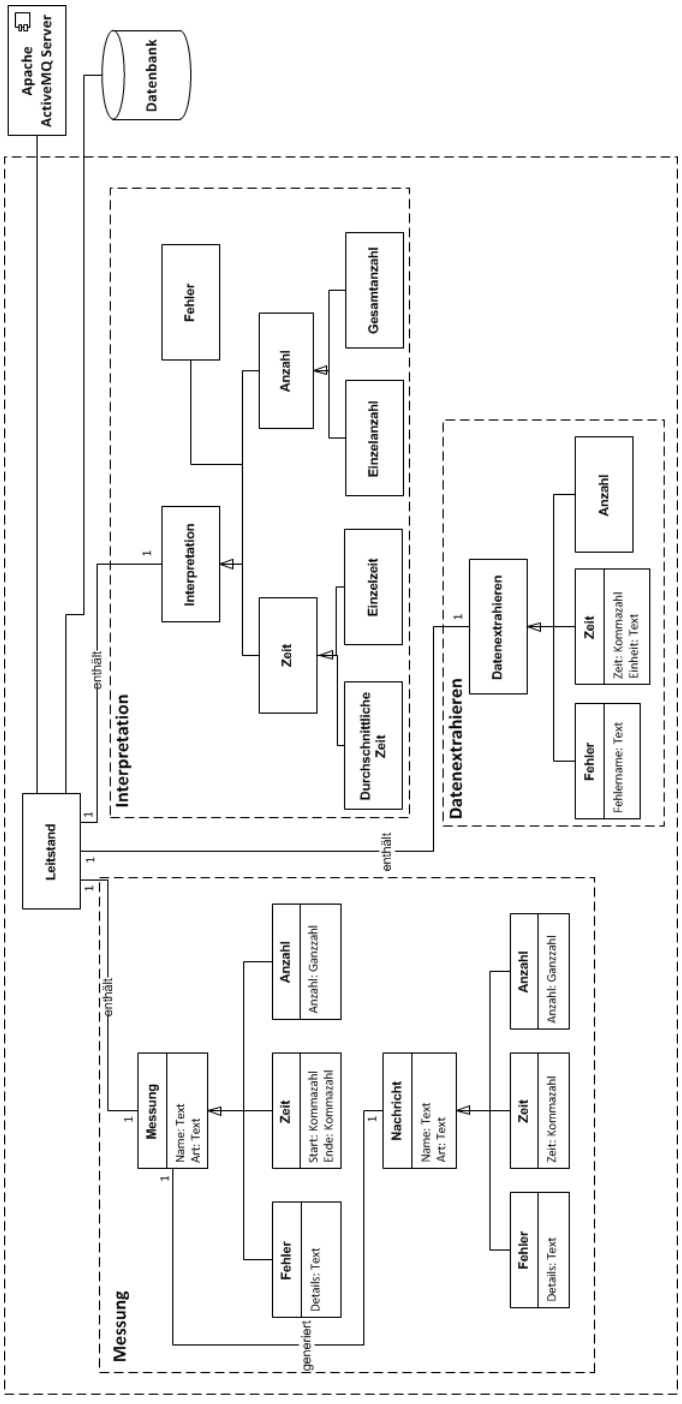

Abbildung 4.2.: Fachliche Architektur und deren Schnittstellen

Das folgende Listings 4.1 zeigt die Anbindungen zur Datenbank und zum Apache Active MQ Server.

```
<appSettings>
    <add key="DatabaseConnection" value="Server=tfs-srs.
        informatik.haw-hamburg.de\HLSDB,1443;Database=ba_kb;Uid
        =*****;Password=*****;Trusted_Connection=False" />
    <add key="MessagingServer" value="tfs-srs.informatik.haw-
        hamburg.de"/>
    <add key="MessagingPort" value="61616"/>
    <add key="MessagingQueue" value="ba_kb"/>
</appSettings>
```

Listing 4.1: Fachliche Architektur - Schnittstellen - Datenbank und Apache Active MQ Server

Die einzelnen Einträge sind in Key-Value-Paare gegliedert. Der Key gibt den Namen an und der Value gibt den Wert an, der benötigt wird, um eine Verbindung zur Datenbank bzw. zum Apache Active MQ Server aufzubauen. Der erste Eintrag ist wie folgt gegliedert: Der Key ist DatabaseConnection und der Value hat folgende Einträge: den Server tfs-srs.informatik.haw-hamburg.de und den Namen der Datenbank HLSDB und den dazu gehörigen Port 1443, den Datenbanknamen ba_kb, den Benutzernamen und das Passwort, die beiden Einträge werden benötigt, um sich bei der Datenbank zu authentifizieren und Trusted_Connection=False (vgl. (MSDN, 2012)) gibt an, dass der Benutzername im Zusammenhang mit dem Passwort bei der Anmeldung verwendet werden soll und nicht die Windows-Anmeldeinformation des aktuellen Benutzers.

4.4. Anwendungsfälle

In diesem Abschnitt werden die Anwendungsfälle "Das HAW-Logistics-System (HLS) messen", „Empfangende Nachrichten vom Apache Active MQ Server extrahieren", "Daten in der Datenbank interpretieren und Diagramme erstellen aus den Datenbankeinträgen" und „Interpretierte Daten und Diagramme auf der Benutzeroberfläche darstellen" beschrieben.

Das HAW-Logistics-System (HLS) messen

Der Anwendungsfall beschreibt die Abfolge, die geschieht beim Messen des HAW-Logistics-System (HLS) und das Versenden der daraus resultierenden Daten.

Titel	Das HAW-Logistics-System (HLS) messen

Akteur	Anwender
Ziel	Das HAW-Logistics-System (HLS) wurde gemessen
Auslöser	Der Anwender hat sich dazu entschieden das HAW-Logistics-System (HLS) zu überwachen
Vorbedingungen	Das HAW-Logistics-System (HLS) und der Leitstand wurden gestartet
Nachbedingungen	Das HAW-Logistics-System (HLS) wurde gemessen, Es wurden Nachrichten mit Messergebnissen erstellt, Die Nachrichten wurden an den Apache Active MQ Server versendet
Erfolgsszenario	1. Das HAW-Logistics-System (HLS) starten 2. Den Leitstand starten 3. Das System misst Daten, wie in Anforderung A2 beschrieben, in Konnektoren und Komponenten im HAW-Logistics-System (HLS) 4. Die Nachricht wird erstellt. Die Nachricht setzt sich wie folgt zusammen: Die Version des Nachrichtenformats, "M0.9", der Name, je nach Wert, Fehler, Zeit oder Anzahl, die Art, Komponentenname oder Konnektorname (Quell- und Zielkomponentenname) und einer Beschreibung, je nach Name einen Messwert bei Zeit, bei Anzahl eine Beschreibung oder Details der Fehlermeldung bei einem Fehler 5. Die Nachrichten werden über den Apache Active MQ Server versendet
Erweiterungen	-
Fehlerfälle	1.a Das HAW-Logistics-System (HLS) wurde schon gestartet 1.b Das HAW-Logistics-System (HLS) lässt sich nicht starten 2.a Der Leitstand wurde schon gestartet 2.b Der Leitstand lässt sich nicht starten

	3.a Beim Messen des HAW-Logistics-System (HLS) ist ein Fehler aufgetreten 4.a Es wurde keine Beschreibung ermittelt 5.a Es wurde keine Nachricht über den Apache Active MQ Server versendet
Häufigkeit	mindestens einmal beim Start des Leitstandes durch den Anwender und je nach Aktualisierung des Leitstandes, bis zu n mal die Stunde
Anforderungen	A1, A2-A5

Tabelle 4.1.: Anwendungsfälle - Messung

Empfangende Nachrichten vom Apache Active MQ Server extrahieren

Der Anwendungsfall beschreibt das Empfangen von Nachrichten vom Apache Active MQ Server, das Extrahieren von den einzelnen Nachrichten, sowie das Schreiben von den daraus resultierenden Daten in die Datenbank.

Titel	Empfangende Nachrichten vom Apache Active MQ Server extrahieren
Akteur	Anwender
Ziel	Daten extrahiert, vorbereitet für den nächsten Schritt, Interpretation
Auslöser	Nachrichten werden über den Apache Active MQ Server empfangen
Vorbedingungen	Anwendungsfall "Das HAW-Logistics-System (HLS) messen"
Nachbedingungen	Daten wurden aus den empfangenden Nachrichten extrahiert
Erfolgsszenario	1. Nachrichten werden vom Apache Active MQ Server empfangen 2. Nachrichten werden einzeln extrahiert 3. Nach der Extraktion werden die ermittelten Daten in die Datenbank geschrieben
Erweiterungen	-

Fehlerfälle	1.a Es werden keine Nachrichten über den Apache Active MQ Server empfangen 3.a. Die Datenbank ist nicht erreichbar
Häufigkeit	so oft, wie der Anwendungsfall "Das HAW-Logistics-System (HLS) messen" durchgeführt wird
Anforderungen	A6, A7

Tabelle 4.2.: Anwendungsfälle - Datenextrahieren

Daten in der Datenbank interpretieren und Diagramme erstellen aus den Datenbankeinträgen

Der Anwendungsfall beschreibt das Laden von Daten aus der Datenbank und die Statistikerhebung über die Daten, sowie die Diagrammerstellung über die Statistik.

Titel	Daten in der Datenbank interpretieren und Diagramme erstellen aus den Datenbankeinträgen
Akteur	Anwender
Ziel	Daten wurden interpretiert und es kann der nächste Schritt erfolgen, Darstellung der Daten auf der Benutzeroberfläche
Auslöser	In der Datenbank sind Daten verfügbar
Vorbedingungen	Anwendungsfälle "Das HAW-Logistics-System (HLS) messen" und „Empfangende Nachrichten vom Apache Active MQ Server extrahieren"
Nachbedingungen	Daten aus der Datenbank wurden entsprechend interpretiert
Erfolgsszenario	1. Daten aus der Datenbank abrufen 2. Je nach Name werden die Daten anders interpretiert 3. Nach der Interpretation der Daten fließen diese Daten in die Erstellung von Diagrammen ein, jedem Eintrag im Dialog wird ein Diagrammeintrag zugeteilt

Erweiterungen	2.a Bei Zeit werden die Daten als Einzelzeit, Gesamtzeit, alle Einträge aus der Datenbank und durchschnittliche Zeit, der Durchschnitt über alle Einträge in der Datenbank wird gebildet, interpretiert 2.b Bei der Anzahl werden (Einzel-)Anzahl, Gesamtanzahl, alle Einträge in der Datenbank und durchschnittliche Anzahl, der Durchschnitt über alle Einträge in der Datenbank gebildet, interpretiert 2.c Bei Fehler werden die Fehler gezählt und die Beschreibung in eine benutzerfreundliche Fehlererklärung interpretiert
Fehlerfälle	1.a Datenbank nicht erreichbar 2.a Bei Zeit ist kein Datenbankeintrag verfügbar 2.b Bei Anzahl ist kein Datenbankeintrag verfügbar 2.c Bei Fehler ist kein Datenbankeintrag verfügbar 3.a Aufgrund von nicht vorhandenen Datenbankeinträgen kann kein Diagramm erstellt werden
Häufigkeit	so oft, wie der Anwendungsfall „Empfangende Nachrichten vom Apache Active MQ Server extrahieren" durchgeführt wurde
Anforderungen	A8-A10

Tabelle 4.3.: Anwendungsfälle - Interpretation

Interpretierte Daten und Diagramme auf der Benutzeroberfläche darstellen

Nachdem alle Anwendungsfälle erfolgt sind, werden nun die interpretierten Daten und Diagramme auf der Benutzeroberfläche dargestellt.

Titel	Interpretierte Daten und Diagramme auf der Benutzeroberfläche darstellen
Akteur	Anwender
Ziel	Benutzeroberfläche wird vollständig mit Daten und Diagrammen dargestellt

Auslöser	Daten wurden interpretiert und Diagramme wurden erstellt
Vorbedingungen	Anwendungsfälle "Das HAW-Logistics-System (HLS) messen", „Empfangende Nachrichten vom Apache Active MQ Server extrahieren", "Daten in der Datenbank interpretieren und Diagramme erstellen aus den Datenbankeinträgen"
Nachbedingungen	Grafische Darstellung der Benutzeroberfläche mit Daten und Diagrammen
Erfolgsszenario	1. Daten und Diagramme werden an den richtigen Stellen auf der Benutzeroberfläche einsortiert bzw. dargestellt
Erweiterungen	-
Fehlerfälle	es liegen keine Daten und Diagramme vor
Häufigkeit	so oft, wie der Anwendungsfall "Daten in der Datenbank interpretieren und Diagramme erstellen aus den Datenbankeinträgen" vollzogen wurden ist
Anforderungen	A11-A16

Tabelle 4.4.: Anwendungsfälle - Darstellung

4.5. Dialog

Die Abbildung 4.3 zeigt das Hauptfenster der Benutzeroberfläche, die über das HLS wacht. Außerdem zeigt die Abbildung 4.4 die Auswertung der Komponente und Abbildung 4.5 die Auswertung des Konnektors. Die Komponente und Konnektor haben jeweils eine Statistik über alle empfangenden Daten, welches die Abbildungen 4.6 und 4.7 zeigen. Eine detaillierte Spezifikation der Dialoge wird in den folgenden Abschnitten erklärt.

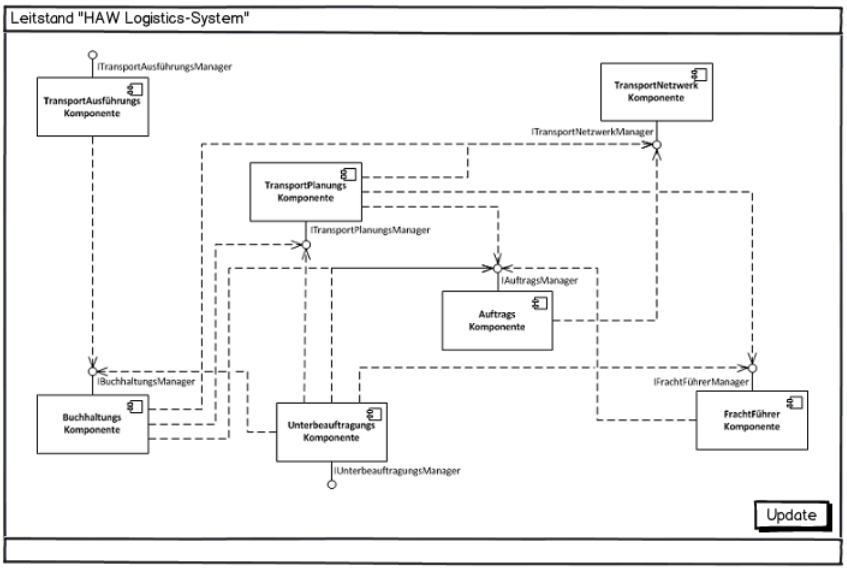

Abbildung 4.3.: Dialog - Hauptfenster

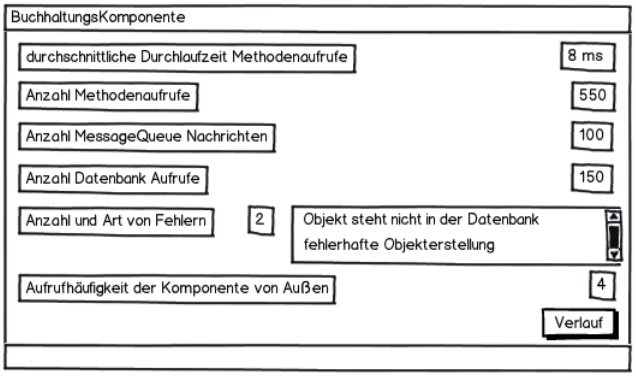

Abbildung 4.4.: Dialog - Komponentenansicht

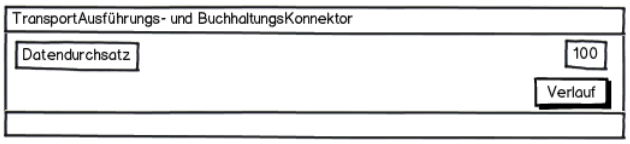

Abbildung 4.5.: Dialog - Konnektoransicht

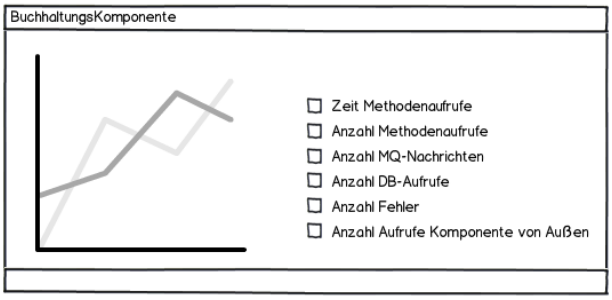

Abbildung 4.6.: Dialog - Komponentenansicht Statistik

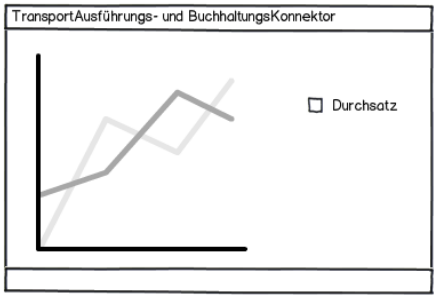

Abbildung 4.7.: Dialog - Konnektoransicht Statistik

4.5.1 Button/Bildabschnitte und deren Aktionen

Folgende Anstöße sind möglich, um bestimmte Aktionen auszulösen.

Anstoß	**Aktionen**
Druck des Buttons “Update"	Benutzeroberfläche erhält interpretierte Daten
Druck des Buttons “Verlauf"	Dialogfenster Verlauf einer Komponente bzw. Konnektor öffnet sich und es wird der Verlauf aller Durchläufe als Diagramm angezeigt
Druck einer Komponente	Dialogfenster der jeweiligen Komponente mit entsprechenden Messungen aus dem letzten Durchlauf wird angezeigt
Druck eines Konnektors	Dialogfenster des jeweiligen Konnektors mit entsprechenden Messungen aus dem letzten Durchlauf wird angezeigt

Tabelle 4.5.: Dialog - Anstoß und Aktionen aus Button/Bildabschnitten

4.5.2 Initialisierung

Beim Starten des Leitstandes erscheint das Hauptfenster mit dem Komponentendiagramm vom HAW-Logistics-System (HLS). Zu diesem Zeitpunkt wurden noch keine Daten angefordert, deshalb sind die Dialoge noch nicht mit Werten belegt. Durch Auswahl des Buttons “Update"werden die benötigten Daten angefordert und in die Dialoge eingebunden. Beim Klicken einer Komponente oder eines Konnektors erscheint jeweils ein weiteres Dialogfenster, welches Daten der Messung anzeigt, wenn das überwachte System läuft. Das gleiche geschieht auch beim Klicken des Buttons Verlauf, indem die Auswertungen stehen, wenn eine Messung durchgeführt wurde, erscheint ein Diagramm bzw. Diagramme.

5. Entwurf

5.1. Architektur

In der folgenden Abbildung 5.1 ist die Architektur des Systems zu sehen. Die Architektur wurde aus den Anforderungen, der Fachlichen Architektur mit Schnittstellen zu Nachbarsystemen und den Anwendungsfällen entwickelt. Im Abschnitt 4.3 wurde schon eine Vorstrukturierung des Leitstandes vorgenommen, welches nun mit in die Architektur einfließt. Daraus ergaben sich folgende Komponenten:

- Benutzeroberfläche
- Fassade
- HLS
- Leitstand
  - Messung
  - Datenextrahieren
  - Interpretation
- MQ Adapter
- Persistence Manager

5.1.1 Benutzeroberfläche

Stellt die ausgewerteten Daten nach der Interpretation, die von der Fassade kommen, in der Benutzeroberfläche dar.

5.1.2 Fassade

Die Fassade bietet die angebotenen Methoden dem HAW-Logistics-System (HLS) und der Benutzeroberfläche an. Diese delegiert die aufgerufenen Methoden an den Leitstand weiter und leitet angeforderten Daten wieder zurück an die Benutzeroberfläche bzw. an das HLS.

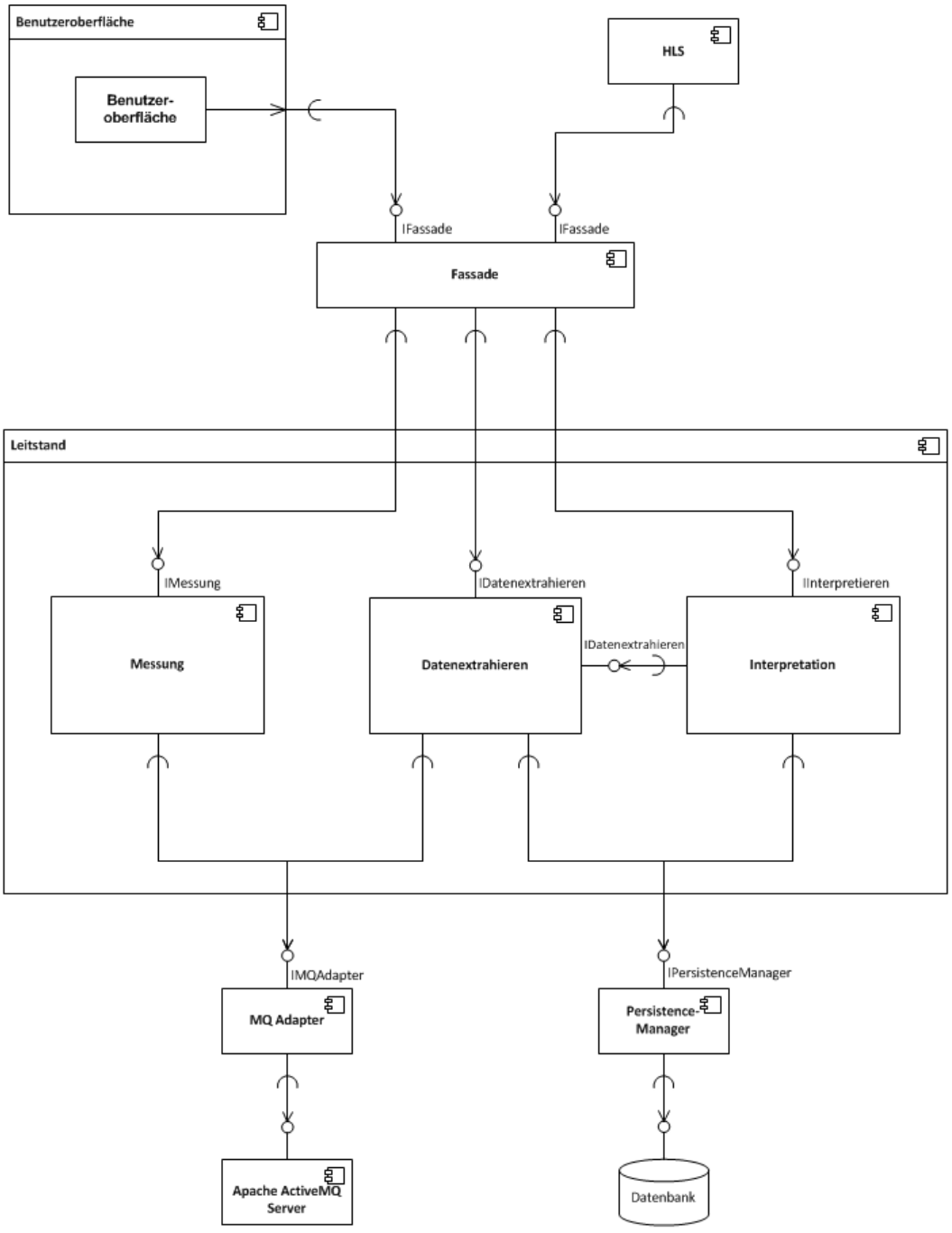

Abbildung 5.1.: Architektur

5.1.3 HLS

HAW-Logistics-System (HLS) ist das zu untersuchende System. Diese Komponente greift auf die Fassade zu und wendet die angebotenen Methoden an. Der Aufbau dieser Komponente wurde schon im Kapitel 3 detailliert erklärt.

5.1.4 Leitstand

Die Komponente Leitstand enthält drei Komponenten **Messung**, **Datenextrahieren** und **Interpretation**. Der Leitstand ist der Anwendungskern des Systems, der die geschäftliche Logik enthält. In den nächsten drei Abschnitten wird auf die einzelnen Komponenten des Leitstands eingegangen.

5.1.4.1 Messung

Die Komponente Messung ist ein Bestandteil des Anwendungskerns. Sie ist dafür zuständig Daten aus dem HAW-Logistics-System (HLS) zu messen und diese Daten in einer Nachricht über den Apache Active MQ Server zu versenden.

5.1.4.2 Datenextrahieren

Die Komponente Datenextrahieren ist ebenfalls ein Bestandteil des Anwendungskerns. Diese Komponente ist dafür zuständig, Nachrichten vom Apache Active MQ Server zu empfangen und diese Nachrichten zu extrahieren. Die Daten die bei der Datenextrahieren entstehen, werden in eine Datenbank geschrieben für die spätere Weiterverarbeitung bzw. Wiederverwendung.

5.1.4.3 Interpretation

Der letzte Bestandteil des Anwendungskerns ist die Komponente Interpretation, diese ist dafür zuständig die Daten aus der Datenbank zu laden und diese weiterzuverarbeiten. Die Weiterverarbeitung beruht auf der Statistikerhebung, die auf der Benutzeroberfläche dargestellt wird.

5.1.5 MQ Adapter

Der MQ Adapter ist ein weiterer Bestandteil der Architektur, der ist das Bindeglied zwischen Anwendungskern und dem Nachbarsystem Apache Active MQ Server. Im Abschnitt 5.3 werden detaillierte Angaben über den MQ Adapter gemacht.

5.1.6 Persistence Manager

Der Persistencemanager ist das Bindeglied zwischen Anwendungskern und Datenbank und erfolgt gleichermaßen, wie bei dem Nachbarsystem Apache Active MQ Server, über einen Adapter.

5.2. Ablauf des Leitstandes

Folgende Abbildung 5.2 zeigt den Ablauf des Leitstandes. Die Benutzeroberfläche startet die Anwendung. Der Aufruf wird an die Fassade des Leitstands weitergeleitet. Nachdem die Anwendung gestartet wurde, wird das HAW-Logistics-System (HLS) gestartet. Das HAW-Logistics-System (HLS) ruft die erforderliche Methode, für die Messung der Daten und das Versenden der Nachricht auf, die ebenfalls an die Fassade weitergeleitet wird. Die Fassade leitet die Methode an die Komponente Messung im Anwendungskern weiter. Die Komponente schickt die Nachricht über den MQ Adapter an den Apache Active MQ Server raus. Bei erfolgreicher Versendung der Nachricht erfolgt eine Bestätigung an die Komponente Messung und an das HAW-Logistics-System (HLS). Die Komponente Datenextrahieren fragt in bestimmten Abständen beim Apache Active MQ Server nach, ob Nachrichten anliegen, die empfangen und extrahieren werden können. Die Anfrage erfolgt über den MQ Adapter. Liegt eine Nachricht beim Apache Active MQ Server an, wird diese, über den MQ Adapter, an die Komponente Datenextrahieren, die sich im Anwendungskern befindet, übergeben. Nachdem die Komponente die Nachricht erhalten hat, werden die Daten aus der Nachricht extrahiert. Nach der Extrahierung werden die Daten über den Persistencemanager in die Datenbank geschrieben. Bei erfolgreicher Speicherung erfolgt eine Bestätigung. Wenn Daten in der Datenbank vorliegen, kommt die Komponente Interpretation nun zum Einsatz. Diese liest die Daten, über den Persistencemanager, aus der Datenbank, aus. Die Daten werden dann für die Statistik aufbereitet. Es erfolgt für die einzelnen Typen Zeit, Anzahl und Fehler eine Statistikerhebung, die später auf der Benutzeroberfläche präsentiert werden. Nach der Statistikerhebung erfolgt eine Rückmeldung der Komponente Interpretation an die Fassade. Die Fassade leitet die Statistikdaten an die Benutzeroberfläche weiter, die die Daten dann grafisch darstellt.

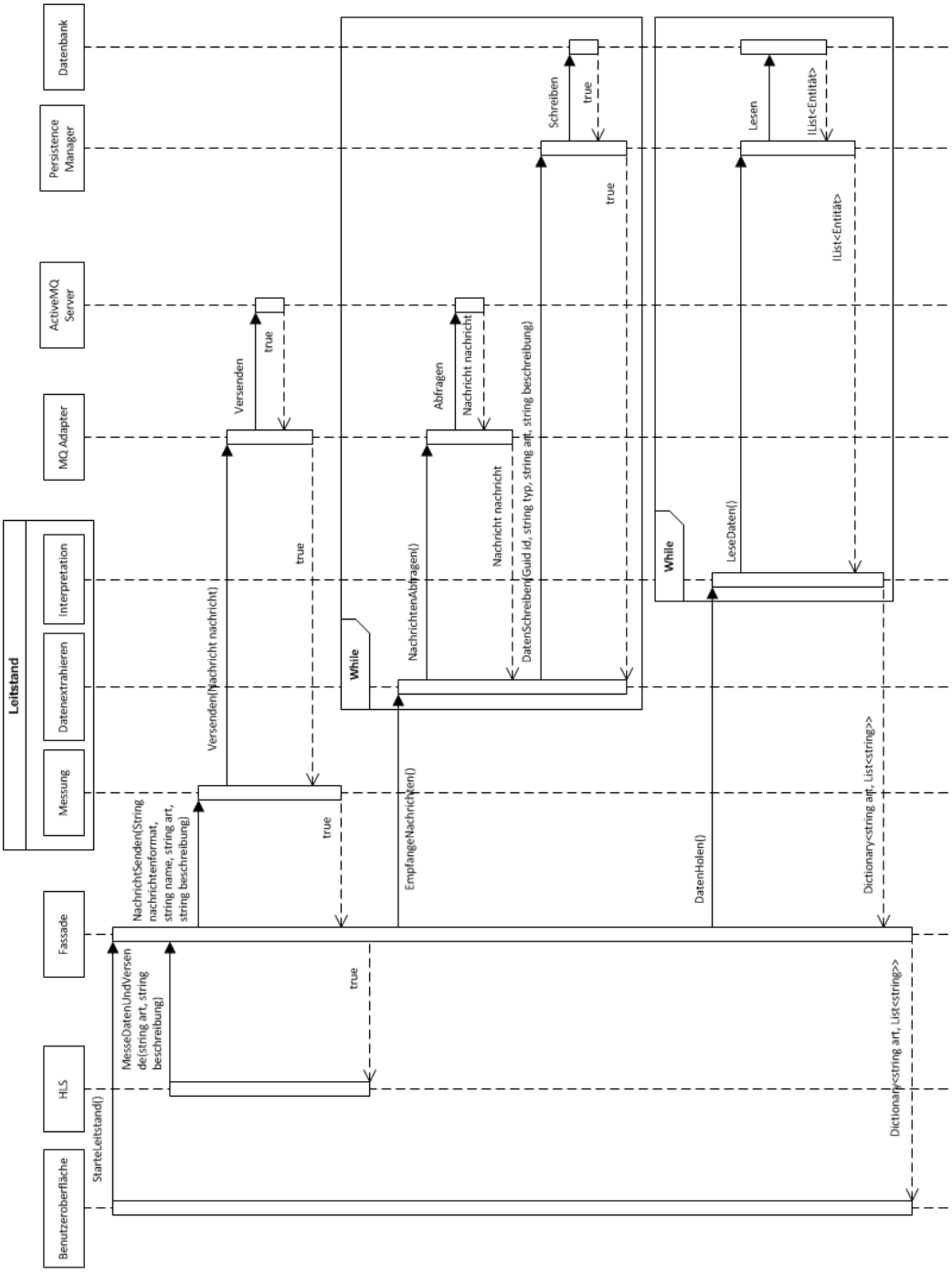

Abbildung 5.2.: Arbeitsablauf des Leitstands

5.3. Entwurfsmuster

In diesem Abschnitt werden auf die Entwurfsentscheidungen für die Architektur eingegangen, die auf (Freemann u. a., 2008) beruhen.

5.3.1 MQ Adapter

Der MQ Adapter konvertiert die inkompatible Schnittstelle vom Client in eine Schnittstelle, die der Client erwartet. Dadurch ermöglicht der Adapter die Zusammenarbeit zwischen Client und Nachbarsystem, in diesem Fall der Apache Active MQ Server und die Datenbank. Es wird immer ein Adapter für ein Nachbarsystem in Anspruch genommen. Der Client ist von der implementierten Schnittstelle entkoppelt. Der Vorteil dieser Entkopplung ist, dass der Adapter, wenn Änderungen an der Schnittstelle gemacht werden, diese kapselt. Durch die Änderungen an der Schnittstelle wird der Client nicht beeinträchtigt, auch dann nicht, wenn der Client mit einer anderen Schnittstelle zusammenarbeiten möchte. Der Persistence Manager in der Architektur repräsentiert den Adapter zur Datenbank. Dieser Adapter speichert und lädt die Entitäten in die Datenbank. Der Adapter wird von einem externen Dienstleister bereitgestellt, dadurch können keine weiteren Angaben gemacht werden. Um die Entitäten zu speichern und zu laden wird das Framework NHibernate verwendet.

5.3.2 Fassade

Eine Fassade bietet eine vereinheitliche Schnittstelle vom System an, die den Zugriff, durch die vereinfachte Schnittstelle, durch den Client auf das System ermöglicht. Der Vorteil einer Fassade ist, dass der Client nicht direkt irgendwelche Methoden aus dem Anwendungskern aufrufen kann. Die Fassade stellt Methoden bereit, die der Client nutzen kann bzw. soll. In diesem Fall stellt die Fassade vom Leitstand Methoden bereit, die aus der Komponente Messung, Datenextrahieren und Interpretation kommen, die das HAW-Logistics-System (HLS) bzw. die Benutzeroberfläche nutzen können. Außerdem leitet die Fassade Ergebnisse aus der Interpretation an die Benutzeroberfläche weiter, damit diese die Ergebnisse repräsentieren kann.

5.3.3 Observer

Das Observermuster definiert eine Eins-zu-viele Abhängigkeit, das bedeutet, dass die Beobachterobjekte von einem Subjektobjekt abhängig sind und darüber informiert werden, wenn sich der Zustand des Subjektobjekts ändert. Daraufhin können alle abhängigen Objekte ihren

Zustand aktualisieren. Der Observer wird dem dem Entwurfsmuster Dependency Properties realisiert. Dieses Entwurfsmuster stellt Windows Presentation Foundation (WPF) zur Verfügung und wird in der Benutzeroberfläche verwendet.

5.4. Komponentenaussensichten und -innensichten des Leitstands

Die Aussensicht zeigt die angebotenen und genutzten Schnittstellen einer Komponente. Die Innensicht zeigt die innere Struktur einer Komponente mit den Klassen Anwendungsfall (orange), Verwalter (grün), Entität (blau) und Datentyp (grau). Die Anwendungsfallklasse repräsentiert die Funktionalität der Komponente, aber kann auch weitere Anwendungsfälle und Komponenten enthalten. Der Verwalter ist für die Verwaltung der Entitäten zuständig, seine Hauptaufgaben sind Lesen, Suchen, Aktualisieren, Erstellen und Löschen. Die Entitäten enthalten entitätenspezifische Logik und die Datentypen enthalten datenspezifische Logik. Die Abbildungen der Aussen- und Innensichten der einzelnen Komponenten, die im Weiteren detaillierter erklärt werden, sind im Anhang C zu finden.

5.4.1 Messung

Die Aussensicht der Komponente Messung zeigt ihre angebotene Schnittstelle IMessung und die genutzte Schnittstelle IMQAdapter zum Apache Active MQ Server.
Die innere Struktur besteht aus drei Anwendungsfallklassen AnzahlAnwendungsfall, FehlerAnwendungsfall und ZeitAnwendungsfall.

5.4.2 Datenextrahieren

Die Komponente Datenextrahieren zeigt die DatenextrahierenAnwendungsfallklasse, die drei DatenextrahierenVerwalterklassen AnzahlVerwalter, FehlerVerwalter unf ZeitVerwalter und die drei Entitäten Zeit, Anzahl und Fehler.
Die Aussensicht der Komponente Datenextrahieren nutzt zwei Schnittstellen, zu einem die IMQAdapter zum Apache Active MQ Server und die andere Schnittstelle IPersistenceManager zur Datenbank. Außerdem stellt die Komponente noch die Schnittstelle IDatenextrahieren bereit.

5.4.3 Interpretation

Die Komponente Interpretation stellt wie die Komponente Messung und Datenextrahieren eine angebotene Schnittstelle IInterpretation zur Verfügung und nutzt die Schnittstelle IPersistenceManager zur Datenbank.
Die Innensicht besteht aus drei Anwendungsfallklassen AnzahlAnwendungsfall, FehlerAnwendungsfall und ZeitAnwendungsfall und außerdem aus drei Verwalterklassen AnzahlVerwalter, FehlerVerwalter und ZeitVerwalter.

6. Implementierung

Die nachfolgenden Abschnitte beschreiben die Realisierung des Leitstandes. Zu einem werden die Bestandteile des Anwendungskerns, die Anwendungskernfassade, die Benutzeroberfläche und die Apache Active Message Queuing (MQ) Anbindung zum Server detaillierter erläutert.

6.1. Anwendungskern

Der Anwendungskern besteht aus drei Komponenten **Messung**, **Datenextrahieren** und **Interpretation** und außerdem beinhaltet der Anwendungskern noch die Komponente **Nachrichten**. Im folgenden Abschnitt werden die Komponenten **Messung**, **Datenextrahieren**, **Interpretation** und die **Nachrichten** detaillierter erläutert.

6.1.1 Messung

Die Komponente **Messung** ist dafür zuständig, die gemessenen Daten im HAW-Logistics-System (HLS) an den Apache Active MQ Server zu versenden mit dem Komponentennamen bzw. Konnektornamen und einer detaillierten Beschreibung, die spezifisch auf Anzahl, Zeit oder Fehler zurück zuführen ist. Die Komponente besteht aus drei Anwendungsfällen *Anzahl-Anwendungsfall*, *FehlerAnwendungsfall* und *ZeitAnwendungsfall*, die jeweils das Versenden der Nachrichten, nach Nachrichtenart, tätig. Außerdem enthält die Komponente noch vier Interfaces, zu einem das Superinterface *IMessungManager*, welches die drei spezifischen Interfaces *IZeitManager*, *IAnzahlManager* und *IFehlerManager* enthält, die die Methode *MesseUndVersende-*<Nachrichtenart> anbietet.

6.1.2 Datenextrahieren

Die Komponente **Datenextrahieren** ist dafür zuständig die Nachrichten vom Apache Active MQ Server zu empfangen und diese entsprechend zu extrahieren bzw. aus den Nachrichten die richtigen Daten zu entnehmen. Danach werden die Daten in der Datenbank gespeichert. Die Komponente besteht aus den Entitäten *Anzahl*, *Zeit* und *Fehler*, die jeweils die Eigenschaften Identitätsnummer, Name, Art und Beschreibung beinhalten. Die Identitätsnummer wurde

mit dem Typ *Guid* realisiert, um die Eindeutigkeit der Identitätsnummer zu gewährleisten. Außerdem besteht die Komponente aus drei Verwaltern, den *AnzahlVerwalter*, *FehlerVerwalter* und *ZeitVerwalter*, die den Zugriff auf die Datenbank, über den Persistencemanager, regeln. Jeder Verwalter, je nach Entität, enthält die Methode *Erstelle*-<Entität>, die die Daten Identitätsnummer, Name, Art und Beschreibung in die Tabelle in der Datenbank speichert. Die Datenbank enthält drei Tabellen *Anzahl*, *Fehler* und *Zeit*. Der Anwendungsfall *DatenextrahierenAnwendungsfall* enthält die Methode *EmpfangeNachrichten*, die einen Empfänger erzeugt, der auf Nachrichten horcht und diese empfängt. Bei der Erzeugung des Empfängers wird dem Empfänger eine Behandlungsroutine mitgegeben, nach welchen Kriterien, nach dem Empfang einer Nachricht, die Nachricht behandelt werden soll. Jede Nachricht wird nach ihrer Nachrichtenart gefragt, so kann unterschieden werden, um welche Nachricht es sich handelt, damit die Daten später auch in der richtigen Tabelle in der Datenbank landen. Das Interface *IDatenextrahierenManager* stellt die Methode *EmpfangeNachrichten* nach Außen hin für den Zugriff bereit.

6.1.3 Interpretation

Die Komponente **Interpretation** entnehmt die Einträge aus der Datenbank und bringt sie in ein geeignetes Format, um diese Daten, der Anwendungskernfassade, bereit zustellen. Die Komponente besteht aus einem Superinterface *IInterpretationManager*, welches die drei untergeordneten Interfaces *IAnzahlManager*, *IFehlerManager* und *IZeitManager* beinhaltet. Die drei untergeordneten Interfaces bieten die Methoden *Gib*<Entität>*Liste* und *Gib*<Entität>*Einzel* an. Außerdem enthält die Komponente noch drei Verwalter, der *AnzahlVerwalter*, *FehlerVerwalter* und *ZeitVerwalter*, die Verwalter greifen über den Persistencemanager auf die Datenbank zu, um die Daten zu entnehmen. Die Anwendungsfälle *AnzahlAnwendungsfall*, *FehlerAnwendungsfall* und *ZeitAnwendungsfall* stellen die Logik der beiden Methoden *Gib*<Entität>*Liste* und *Gib*<Entität>*Einzel* bereit. Die Methoden *Gib*<Entität>*Liste* und *Gib*<Entität>*Einzel* geben jeweils ein Dictionary zurück, welches die Einträge aus der Datenbank, in einem geeigneten Format, beinhalten.

6.1.4 Nachrichten

Eine **Nachricht** besteht aus einem *NachrichtenKopf*, einer *Nachrichtenart* und die Nachricht an sich. Der *NachrichtenKopf* enthält folgende Key-Value-Paare:

- Monitoring-Nachricht, 1;
- Zeitstempel, „YYYY-MM-DD T HH:MM:SS";

- Version, M0.9,
- Nachrichtenart, <Nachrichtenart>

Die *Nachrichtenart* kann eine *AnzahlNachricht*, *ZeitNachricht* oder *FehlerNachricht* sein. Alle drei Nachrichtenarten sind gleich strukturiert, sie enthalten einen Namen, eine Art und eine Beschreibung. Die Nachrichtenarten sind auf die Entitäten *Anzahl*, *Zeit* und *Fehler* zurück zuführen. Die *Nachricht* strukturiert die Nachricht in Key-Value-Paare. Außerdem stellt die *Nachricht* einen Getter und einen Setter zur Verfügung, für das reinschreiben in das Dictionary und das wieder raus holen von Nachrichteninformationen. Außerdem werden die Handlungen für das Versenden bereitgestellt. Das Interface *INachrichtenFabrik* stellt die Methoden *Erzeuge-Empfaenger* und *ErzeugeSender* bereit, die Nachrichten über den eingetragenen Queuenamen versenden bzw. empfangen. Nachdem der Sender initialisiert wurde, können Nachrichten versendet werden, die Methode *SendeNachricht* stellt das Interface *ISender* bereit. Das Interface *IEmpfaenger* stellt nur die Methode *Stop* bereit, das Empfangen der Nachrichten übernimmt eine andere Komponente, die im Weiteren detailliert erklärt wird.

6.2. Anwendungskernfassade

Die **Anwendungskernfassade** stellt die Schnittstelle nach Außen hin dar. Die Fassade bietet folgende Methoden an:

- *MesseUndVersende*<Entität>,
- *EmpfangeNachrichten*,
- *Gib*<Entität>*Liste*,
- *Gib*<Entität>*Einzel*

Auf diese Methoden können nun andere Anwendungen zugreifen. Unter anderem greift die Benutzeroberfläche auf die Anwendungskernfassade zu. Das Interface *IAnwendungskernFassade* beinhaltet die drei Interfaces *IMessungManager*, *IInterpretationManager* und *IDatenextrahieren-Manager*, die jeweils die Methoden beinhalten, die in der Komponente Anwendungskernfassade zusammenlaufen.

6.3. Apache Active Message Queuing (MQ)

Die Komponente besteht aus vier Klassen, die *Factory*, *Listener*, *Reader* und *Writer*. Die Klasse *Factory* enthält die Methode *ErzeugeEmpfaenger* und *ErzeugeSender*. Bei jeder Initialisierung der Klasse *Factory* wird aus der Konfigurationsdatei *App.config*, die vordefinierten Daten *Server*

und *Port*, für die Anbindung an den Apache Active Message Queuing (MQ) Server, entnommen. Bei der Erzeugung eines Empfängers bzw. Senders wird der vordefinierte *Queuename* aus der Konfigurationsdatei *App.config* genommen. Außerdem wird bei der Erzeugung des Empfängers die Behandlungsroutine für empfangene Nachrichten mitgegeben, d.h. wie empfangene Nachrichten behandelt werden sollen. Die Klasse *Listener* definiert die Methode *OnMessage*, welche die empfangene Nachricht in ein Dictionary Format, mit Key-Value-Werten, bringt. Außerdem ist in der Klasse noch die Methode *FindeNachrichtenTypZuArt* definiert, die aus der Nachrichtenart, die in der *OnMessage* Methode extrahiert wurde, den Nachrichtentyp findet, die in der Komponente *Nachrichten* vorhanden sind. Der *Reader* definiert die Methode *Stop*, die den Empfänger stoppt Nachrichten weiter zu lesen. Der *Writer* definiert die Methode *SendeNachricht*, der die Nachrichten an die angegebene Queue sendet.

6.4. Benutzeroberfläche mit Windows Presentation Foundation (WPF)

Die Benutzeroberfläche wurde in C# und Extensible Application Markup Language (XAML) Code getrennt, damit besteht die Möglichkeit die Darstellung von der Logik zu trennen. In den folgenden beiden Abschnitten wird die Trennung verdeutlicht. Außerdem sind in den folgenden Abschnitten einige Bilder zu sehen, wie die Darstellung der Benutzeroberfläche der einzelnen Fenster realisiert wurde.

6.4.1 C#

Die Logik basiert auf reinen C# bzw. .NET Code. In den nächsten Abschnitten wird kurz darauf eingegangen, wie die Logik in die Darstellung integriert wurde.

6.4.1.1 Hauptfenster

Das **Hauptfenster** definiert für jede Komponente und Konnektor, bei Auswahl des Elements, auf der Benutzeroberfläche, eine Methode. Die Methode öffnet ein neues Fenster, in diesem Fall ein *Komponentenfenster* bzw. *Konnektorfenster*. Außerdem enthält das Hauptfenster einen Button *Update*, welcher die Daten von der Anwendungskernfassade anfordert, die auf der Benutzeroberfläche dargestellt werden sollen. Nach der Anforderung werden die Daten in ein geeignetes Format gebracht, damit die Weiterverarbeitung der Daten reibungslos verläuft. Wenn das Hauptfenster geschlossen wird, wird der Empfänger, der die Nachrichten empfängt, gestoppt.

6.4.1.2 Komponentenfenster

Wenn auf der Benutzeroberfläche Hauptfenster ein Element vom Typ Komponente ausgewählt wurde, öffnet sich das **Komponentenfenster**. Bei der Auswahl werden die entsprechenden Daten an das *Komponentenfenster* vom *Hauptfenster* übertragen. Diese Daten werden dann an die entsprechenden Felder im *Komponentenfenster* gebunden. Folgenden Daten werden auf der Benutzeroberfläche gebunden:

- durchschnittliche gemessene Zeit der Methoden der ausgewählten Komponente,
- Gesamtanzahl der Methodendurchläufe,
- Gesamtanzahl der versendeten Message Queuing (MQ) Nachrichten,
- Gesamtanzahl der Datenbankaufrufe,
- Gesamtanzahl der Fehler,
- Gesamtanzahl der Aufrufe von Außen

Es werden die letzten gemessenen Daten an die Benutzeroberfläche gebunden. Das *Komponentenfenster* enthält außerdem noch ein Button *Verlauf*, welcher ein Weiteres neues Fenster *Komponentenstatistikfenster* öffnet.

6.4.1.3 Konnektorfenster

Gleichermaßen erfolgt im **Konnektorfenster** das Gleiche wie im *Komponentenfenster*. Bei Auswahl eines Elements, des Typs Konnektors, erscheint ein neues Fenster vom Typ *Konnektorfenster*. Die entsprechenden Daten werden an das *Konnektorfenster*, vom Hauptfenster, übergeben. Danach werden die Daten an die entsprechenden Felder auf der Benutzeroberfläche gebunden. Folgenden Daten werden auf der Benutzeroberfläche gebunden:

- Gesamtanzahl des Datendurchsatz

Es werden die letzten gemessenen Daten an die Benutzeroberfläche gebunden. Wie das Komponentenfenster enthält auch das *Konnektorfenster* einen Button *Verlauf*, welches ein neues Fenster vom Typ *Konnektorstatistikfenster* öffnet.

6.4.1.4 Komponentenstatistikfenster

Das Fenster **Komponentenstatistikfenster** zeigt die gemessenen Daten vom HAW-Logistics-System (HLS) als Verlauf in einem Diagramm an. Bei Auswahl des Buttons *Verlauf* im Komponentenfenster werden die Diagrammdaten an das neue Fenster *Komponentenstatistikfenster* übergeben. Das *Komponentenstatistikfenster* bindet die übergeben Daten an die Benutzeroberfläche. Folgende Daten werden an die Benutzeroberfläche gebunden:

- durchschnittliche gemessene Zeit der Methoden der ausgewählten Komponente,
- Gesamtanzahl der Methodendurchläufe,
- Gesamtanzahl der versendeten Message Queuing (MQ) Nachrichten,
- Gesamtanzahl der Datenbankaufrufe,
- Gesamtanzahl der Fehler,
- Gesamtanzahl der Aufrufe von Außen

6.4.1.5 Konnektorstatistikfenster

Im **Konnektorstatistikfenster** passiert das Gleiche wie im Komponentenstatistikfenster. Beim Auswahl des Buttons *Verlauf* öffnet sich ein neues Fenster *Konnektorstatistikfenster* und es werden die entsprechenden Diagrammdaten übergeben. Diese Diagrammdaten werden an die Benutzeroberfläche gebunden. Folgenden Daten werden dabei gebunden:

- Gesamtanzahl des Datendurchsatz

6.4.2 XAML

Die Darstellung der Benutzeroberflächen wurde mit dem WPF-Designer gestaltet. Die Darstellung basiert auf reinen XAML Code. In den nächsten Abschnitten werden fünf Benutzeroberflächen vorgestellt. Das Hauptfenster stellt das Zentrum der Benutzeroberfläche dar, welches die anderen zwei Benutzeroberflächen Komponentenfenster und Konnektorfenster aufruft. Aber wiederum rufen die beiden Fenster das Statistikfenster, abhängig vom aufgerufenen Element, mit dem Diagramm auf.

6.4.2.1 Hauptfenster

Das dargestellte Komponentendiagramm basiert auf dem Komponentendiagramm des HAW-Logistics-System (HLS). Das Komponentendiagramm musste erst in eine Skalierbare Vektorgrafik transformiert werden, welches Microsoft Visio anbietet. Daraufhin wurde diese skalierbare Vektorgrafik mit Hilfe des Inkscape 0.48.2 Programms in ein PDF Format gebracht. Diese PDF Datei wurde umbenannt in ein AI (Adobe Illustratur) Format und nun konnte diese Datei mit dem Microsoft Expression Design Programm in eine XAML Datei exportiert werden. Die exportierte XAML Datei wurde in die Benutzeroberfläche des Hauptfenster integriert. Die Benutzeroberfläche des Hauptfenster wurde aus vielen Controls zusammengesetzt. Folgende Controls beinhaltet das Hauptfenster:

- Rectangle (Rechteck), repräsentiert die Komponente

- Textblock, Namensbezeichnung in den Komponenten und angebotenen Schnittstellen
- Path (Linie), wird für mehrere Darstellungen genutzt, u.a für die Pfeile, Kreise, gestrichelten und nicht gestrichelten Linien und Komponentensymbolen

Das Listing 6.1 zeigt einen kleinen Einblick in die XAML Datei der Benutzeroberfläche Hauptfenster. Der Einblick zeigt die definierten Controls innerhalb des Fensters.

```
1  <Rectangle Height="65" HorizontalAlignment="Left" Margin="
       216,140,0,0" Name="TransportPlanungsKomponente" Stroke="Black"
        VerticalAlignment="Top" Width="119" Canvas.Top="-8" Canvas.
       Left="29" MouseLeftButtonDown="
       TransportPlanungsKomponente_Click"/>
2
3  <TextBlock Height="17" HorizontalAlignment="Left" Margin="
       12,60,0,0" Name="TransportAusfuehrungsKomponente" Text="
       TransportAusführungs" VerticalAlignment="Top" Width="124"
       Canvas.Left="16" Canvas.Top="11" MouseLeftButtonDown="
       TransportAusfuehrungsKomponente_Click"/>
4
5  <Path x:Name="Linie" Width="0.32" Height="20.1213" Canvas.Left="
       501" Canvas.Top="247" Stretch="Fill" StrokeThickness="0.32"
       StrokeLineJoin="Round" Stroke="#FF000000" Data="F1 M
       472.603,274.751L 472.603,254.949"/>
6
7  <Path x:Name="Kreis" Width="8.71472" Height="8.72133" Canvas.Left
       ="497" Canvas.Top="239" Stretch="Fill" StrokeThickness="0.32"
       StrokeLineJoin="Round" Stroke="#FF000000" Data="F1 M
       468.405,250.751C ... 468.405,250.751 Z"/>
8
9  <Path x:Name="Pfeil" Width="11.3613" Height="7.68001" Canvas.Left
       ="506" Canvas.Top="239" Stretch="Fill" StrokeThickness="0.32"
       StrokeLineJoin="Round" Stroke="#FF000000" Data="F1 M
       487.717,254.439L ... 487.717,247.079"/>
10
11 <Path x:Name="FrachtFueherAuftragsKonnektor" Width="7.88" Height=
       "0.320007" Canvas.Left="216" Canvas.Top="349" Stretch="Fill"
       Fill="#FF000000" Data="F1 M 190.216,356.463L ...
       194.056,356.463 Z" MouseLeftButtonDown="
       FrachtFueherAuftragsKonnektor_Click"/>
```

Listing 6.1: XAML Code Benutzeroberfläche - Hauptfenster - Controls

Die Controls, die im Listing 6.1 zu sehen sind, wurden teilweise durch das Programm Microsoft Expression Design erstellt, u.a. Path, und die Controls Rectangle und Textblock bot der WPF Designer an. Das Grundlagenkapitel gibt Aufschluss über die Syntax von XAML.

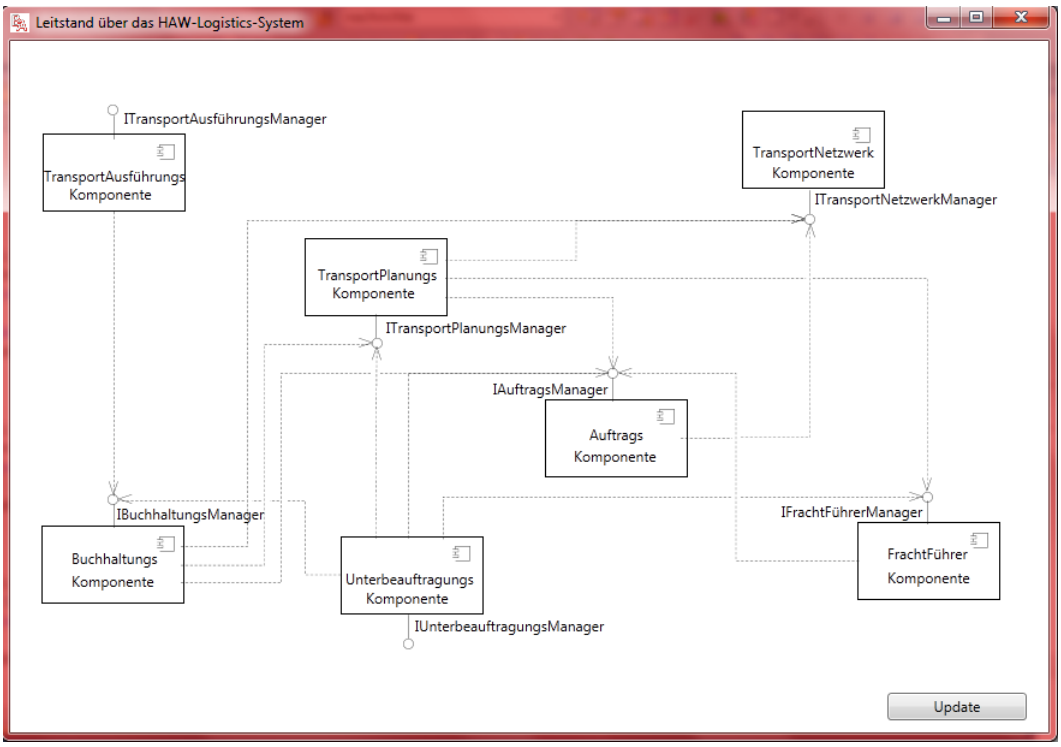

Abbildung 6.1.: Hauptfenster

6.4.2.2 Komponentenfenster

Das Komponentenfenster wurde einfach gehalten, es besteht aus dreizehn Textblöcken, sechs davon wurden als Namensbezeichnung verwendet, ein Textblock wurde für die Einheit verwendet und die anderen sechs werden für die interpretierten Daten verwendet, einem Scrollviewer, der die Art der Fehler enthält und einem Button über den das Statistikfenster, vom Typ Komponente, erreicht werden kann.

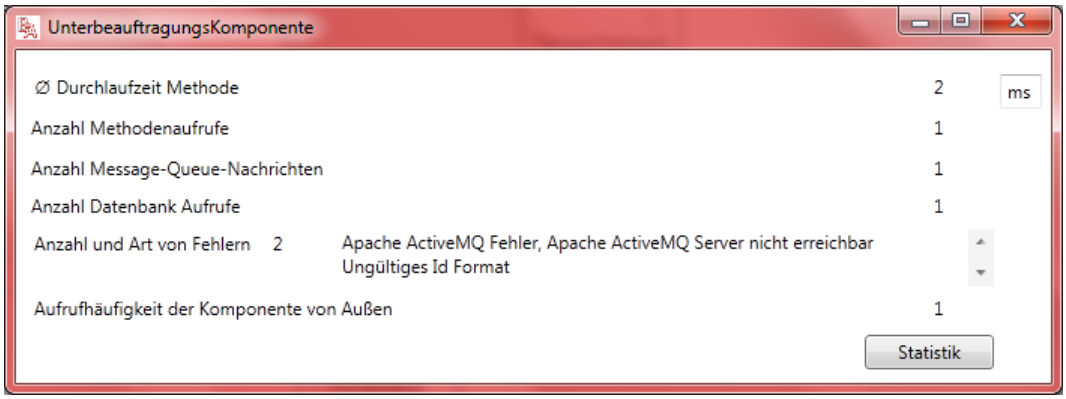

Abbildung 6.2.: Komponentenfenster

6.4.2.3 Konnektorfenster

Ebenso wurde das Konnektorfenster einfach gehalten, es besteht aus nur zwei Textblöcken, welches wiederum für die Namensbezeichnung ist und der andere Textblock für die interpretierten Daten, und einem Button, der das Statistikfenster, vom Typ Konnektor, anzeigt.

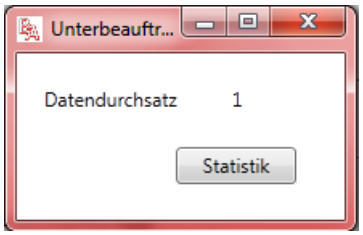

Abbildung 6.3.: Konnektorfenster

6.4.2.4 Statistikfenster

Das Statistikfenster vom Typ Komponente und Konnektor wurde mit dem WPF Toolkit 3.5 gestaltet. Dieses Toolkit bietet Controls für die Diagrammgestaltung an u.a. LineSeries. Das Control bietet die Möglichkeit die ausgewerteten Daten aus der Komponente Interpretation in einem Diagramm auf der Benutzeroberfläche darzustellen. Um die Charts auf der Benutzeroberfläche darzustellen, ist es notwendig die DLL *System.Windows.Controls.DataVisualization.Toolkit* einzubinden. Das Listing 6.4 zeigt die Einbindung der LineSeries im Control Chart und die Achsenbezeichnung.

```
1 <chartingToolkit:Chart Name="chart1" Title="Statistik">
```

```
2   <chartingToolkit:Chart.Series>
3      <chartingToolkit:LineSeries
4       Title="Zeit Methodenaufrufe" x:Name="ZeitMethode"
            ItemsSource="{Binding}" DependentValuePath="Value"
            IndependentValuePath="Key" IsSelectionEnabled="True"/>
5       ...
6   </chartingToolkit:Chart.Series>
7   <chartingToolkit:Chart.Axes>
8     <chartingToolkit:LinearAxis Orientation="X" Interval="1"
          Title="Durchläufe"/>
9     <chartingToolkit:LinearAxis Orientation="Y" Title="Anzahl/
          Zeit"/>
10  </chartingToolkit:Chart.Axes>
11 </chartingToolkit:Chart>
```

Listing 6.2: XAML Code Benutzeroberfläche - Statistikfenster

Das Listing 6.4 zeigt die Definition eines Diagramms mit mehreren Linien. Das Chart definiert die Grundstruktur im dem dann Diagrammtypen eingebunden werden können, z.B. LineSeries. Das Key-Value-Paar greift später auf die Logik in der .cs Datei zu. Außerdem kann noch die Achsenbezeichnung definiert werden im einem Chart, in diesem Fall trägt die X-Achse die Durchläufe und die Y-Achse die Zeit bzw. Anzahl der gemessenen Daten aus der Interpretation. Das Statistikfenster wurde für die beiden Elemente Komponente und Konnektor gleichermaßen gestaltet.

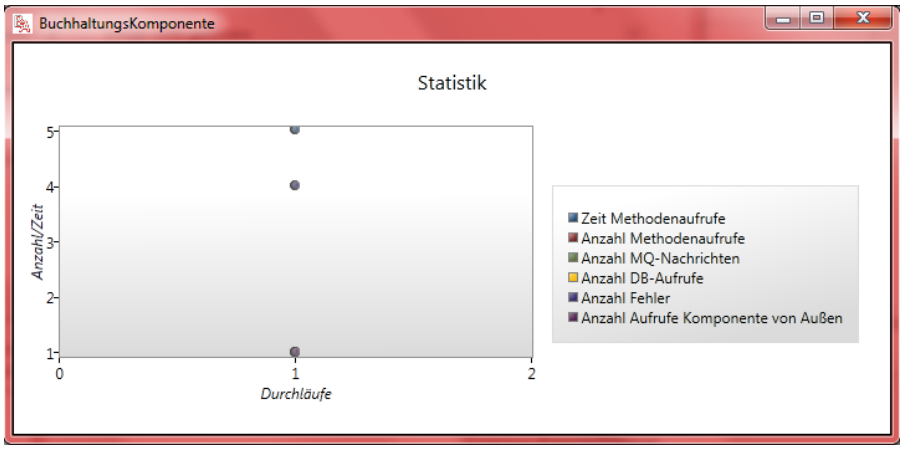

Abbildung 6.4.: Komponentenstatistikfenster

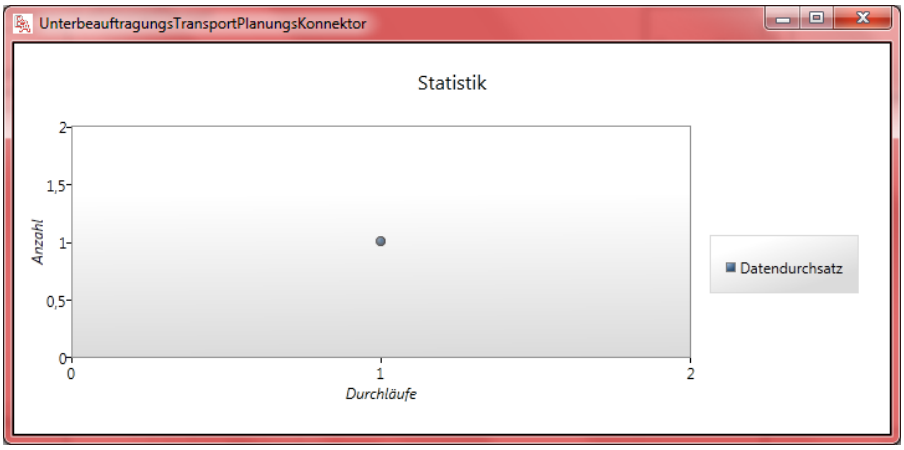

Abbildung 6.5.: Konnektorstatistikfenster

6.5. Integration ins HAW-Logistics-System (HLS)

Die Integration im HLS erfolgte durch die Einbindung der DLL Anwendungskernfassade. Durch die Einbindung war es möglich auf die drei Methoden MesseUndVersende<Entität> in der Anwendungskernfassade zuzugreifen.

6.6. Probleme

Einige Probleme sind bei der Realisierung des Leitstandes aufgetreten. Bei der Konvertierung des Komponentendiagramms in das XAML-Format wurden die Konnektoren, die ein oder mehrere Konnektoren geschnitten haben, getrennt, was später in der Implementierung zu Problemen führte. Durch diese Trennung war der Konnektor nicht mehr ein Konnektor, sondern der Konnektor bestand nun aus mehreren Teilen. Dieses Problem ließ sich aber relativ gut lösen. Durch das Schlüsselwort *DataContext* erhielt jeder Bestandteil des Konnektors, der aus mehreren Bestandteilen bestand, den selben Namen und damit die selbe Funktionalität. Durch die Konvertierung entstand noch ein weiteres Problem, die Konnektoren waren so fein gezeichnet, dass die Auswahl eines Konnektors durch den Benutzer erschwert war. Dieses Problem ließ sich nicht beheben.
Einige technische Probleme verzögerten die Weiterarbeitet in der Implementierung, u.a. die Verfügbarkeit des Apache Active Message Queuing (MQ) Server, SQL Datenbank und die Visual Studio 2010 Projektverwaltung. Die Verfügbarkeit des Apache Active Message Queuing (MQ) Server war oft nicht gewährleistet, welches die Anwendung beeinträchtigt hat, dieses Problem

lies sich nicht beheben. Außerdem war die Verfügbarkeit des SQL Servers auch eingeschränkt, aber dieses Problem lies sich mit einer lokalen Datenbank *SQLite* lösen. Der Unterschied zu einer lokalen Datenbank war kaum spürbar, eher ergab sich ein Vorteil daraus, die lokale Datenbank war von der Performanz besser als die Datenbank auf dem SQL Server. Große Probleme ergaben sich auch bei der Projektverwaltung von Visual Studio 2010, Assemblies wurden nicht gefunden, obwohl sie vorhanden waren oder Assemblies wurden nicht neu erstellt, obwohl sie verändert wurden sind. Diese Probleme wirken sich auch auf andere Projekte in der Projektmappe aus, welches einen zeitlichen Aufwand darstellte.

7. Test

Im Anwendungskern enthaltenen Komponenten *Messung*, *Datenextrahieren* und *Interpretation* wurden mit dem Testtool *UnitTesting* von Microsoft erstellt und getestet. In den folgenden beiden Abschnitte wird auf die einzelnen Unit-Tests und dem Systemtest eingegangen.

7.1. Unit-Test

Die Unit-Tests testen Einheiten, in diesem Fall Methoden, unabhängig von den anderen Komponenten. Die drei Komponenten Messung, Datenextrahieren und Interpretation im Anwendungskern wurden auf die Funktionalität der einzelnen Einheiten bzw. Methoden getestet.
In der Komponente Messung wurde die Erstellung der Nachricht, je nach Nachrichtenart, getestet, sowie die Erzeugung des Senders, der die Nachrichten an den Apache Active Message Queuing (MQ) Server versendet. Diese Testfälle werden im Anhang D unter den Testfällen VN-1, VN-2 und VN-3 beschrieben.
In der Komponente Datenextrahieren wurde die Funktionalität des Empfanges von Nachrichten vom Apache Active Message Queuing (MQ) Server getestet, wie der Testfall EN-1 im Anhang D zeigt, und die Erstellung von einem Empfänger. Nachdem eine Nachricht beim Empfänger eingetroffen ist und durch die Behandlungsroutine behandelt wurde, erfolgt die Erstellung einer Instanz einer Entität. Diese Erstellung ist abhängig von der empfangenen Nachricht, deren Nachrichtenart. Nachdem die Nachrichtenart der empfangenen Nachricht ermittelt wurde, konnte nun die Erstellung einer Instanz, abhängig von der Entität, erfolgen. Die Testfälle EA-1, EF-1 und EZ-1 sind im Anhang D zu finden. Bei der Erstellung wurden die ermittelten Daten, die bei der Behandlungsroutine ermittelt wurden, übergeben. Nach der Übergabe der Daten wurden die Daten an den Persistencemanager übergeben, der die Daten letztendlich in die Datenbank schreibt.
In der letzten Komponente Interpretation wurde die Funktionalität des Persistencemanager getestet. Es wurde getestet, ob die Einträge in der Datenbank mit den enthaltenen Daten in den versendeten Nachrichten übereinstimmen. Diese Daten wurden, je nach Entität, abgerufen. Nachdem die Daten eingetroffenen sind, wurde die Erstellung der Listen, die die Testfälle GL-1,

GL-2 und GL-3 wieder spiegeln, und Einzelwerte, die durch die Testfälle GE-1, GE-2 und GE-3 realisiert wurden, getestet, welches wichtig für die Benutzeroberfläche ist. Im Anhang D sind außerdem noch die Testfälle GA-1, GA-2 und GA-3 zu finden, die die Methoden im Verwalter testen.

7.2. Systemtest

Beim Systemtest wurde das Zusammenspiel zwischen den Komponenten im Anwendungskern und den Nachbarsystemen getestet. Es wurde getestet, ob die Komponente Messung ordnungsgemäß die Nachrichten über den Apache Active Message Queuing (MQ) Server versendet und diese Nachrichten durch die Komponente Datenextrahieren abgeholt bzw. empfangen werden können. Wenn Nachrichten empfangen wurden, wurden diese dem entsprechend extrahiert. Die extrahierten Daten wurden mit den versendeten Daten, aus der Komponente Messung, verglichen. Außerdem wurden die extrahierten Daten in der entsprechenden Tabelle, je nach Entität, in der Datenbank gespeichert. Die Komponente Interpretation holte sich alle Daten aus der jeweiligen Tabelle, aus der Datenbank. Diese Daten wurden mit den extrahierten Daten verglichen, welches das Resultat der Komponente Datenextrahieren war. Die Komponente Interpretation verarbeitete die Daten weiter und erstellte geeignete Strukturen, die später von der Benutzeroberfläche verarbeitet und dargestellt wurden. Beim Holen der Daten aus der Datenbank wurde die Anzahl der Einträge mit der Anzahl der Einträge, die gespeichert wurden, verglichen. Die Erstellung der Listen und Einzelwerte für die Benutzeroberfläche erfolgte mit den Einträgen aus der Datenbank, die wiederum gegen geprüft wurden.

8. Fazit

Die Möglichkeit den Anwendungskern (AWK), des Fallbeispiel HAW-Logistics-System (HLS), zu überwachen, wurde erfolgreich umgesetzt. Durch die Benutzeroberfläche des Leitstandes besteht die Möglichkeit, das Komponentendiagramm des HAW-Logistics-System (HLS) und die ausgewerteten Daten einzusehen. Das HAW-Logistics-System (HLS) schickt während des Betriebes Nachrichten an den Apache Active MQ Server. Diese Nachrichten werden vom Leitstand empfangen und ausgewertet, dies wird dann als Resultat auf der Benutzeroberfläche dargestellt.

8.1. Probleme

Bei der Realisierung des Leitstandes konnten oft Teile, die nicht Bestandteil der Anwendung waren, nicht getestet werden. Der Grund dafür war, dass hier über Systemgrenzen hinweg getestet werden musste, was aber beim Testen von Einheiten in einer Anwendung nicht realisiert werden kann. Ein Beispiel ist u.a., dass eine Nachricht versendet wurde, aber der Empfang der Nachricht bei dem Apache Active Message Queuing (MQ) Server nicht überprüft werden konnte. Es konnte nur der Empfang einer Nachricht durch die Anwendung getestet werden. War der Inhalt der versendeten Nachricht der selbe Inhalt der empfangenden Nachricht, so konnte angenommen werden, dass dieser Test erfolgreich abgeschlossen war.
Bei der Recherche nach Ansätzen im Bereich des Monitorings von Anwendungen wurden keine hilfreichen Ansätze gefunden, die in den Ansatz des Leitstands hätte einfließen können. Die Ansätze, die im Bereich Monitoring gefunden wurden, gehen eher in die Richtung Netzwerkverkehr, hierbei u.a. die Überwachung von einer bzw. mehrerer Schnittstellen.

8.2. Erweiterungsmöglichkeiten

Der Leitstand lässt sich in viele Richtungen weiterentwickeln. Zu einem besteht die Möglichkeit die Komponenten und Konnektoren zu animieren. Dies kann durch eigene gestaltete grafische Elemente realisiert werden. Wenn z.B. ein Datenbankaufruf erfolgt, erscheint auf der Benutzeroberfläche im Hauptfenster ein kleines Symbol einer Datenbank, dadurch wird der

Benutzer darüber informiert, dass gerade in der betroffenen Komponente ein Datenbankzugriff erfolgt ist.
Außerdem kann der Leitstand hingehend auf die Messkriterien erweitert werden, da sind dem Entwickler des Systems keine Grenze gesetzt und er kann seiner Kreativität freien Lauf lassen.
Eine weitere Möglichkeit den Leitstand zu erweitern, ist den direkten Zugriff auf das überwachte System, über die Benutzeroberfläche *Hauptfenster*. Der direkte Zugriff auf die betroffene Komponente im überwachten System erfolgt z.B. über ein weiteres Fenster in der Benutzeroberfläche, welches die Möglichkeit bietet, die Komponente neu zu starten, wenn sie droht auszufallen. Natürlich gibt es auch noch andere Möglichkeiten, die Bedrohungen im überwachten System zu managen.
Der Leitstand könnte hingehend auf die Benutzeroberfläche angepasst werden, dass die Möglichkeit besteht, das Komponentendiagramm nicht manuell einzubinden, sondern das Diagramm aus den zu überwachenden System auszulesen und in die Benutzeroberfläche einfließen zu lassen. Außerdem muss dem Leitstand die Struktur des überwachten Systems vermittelt werden, damit die Auswertungen auf der Benutzeroberfläche richtig dargestellt werden.

Anhang

A. HLS

A.1. Komponenteninnensicht und -außensicht

In den folgenden Bildern sind die Innen- und Außensicht der Komponenten zusehen. Bei der Außensicht werden nur die Schnittstellen der Komponente betrachtet. In der Innensicht werden die vier Klassen dargestellt: Anwendungfallklasse (orange), Verwalterklasse (grün), Entitätstyp (blau) und Datentyp (grau).

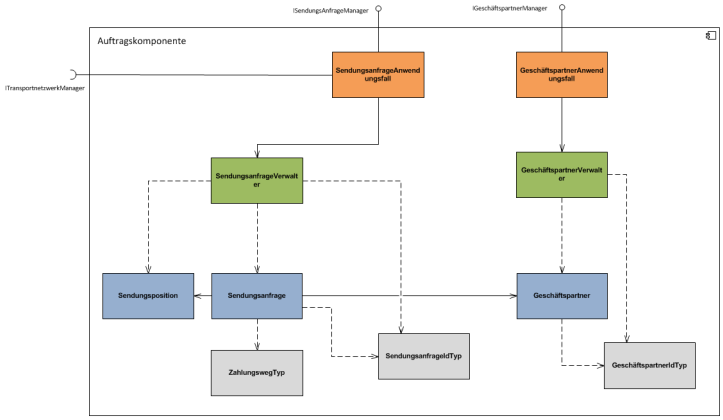

Abbildung 1.: HLS Komponenteninnensicht - Auftragskomponente

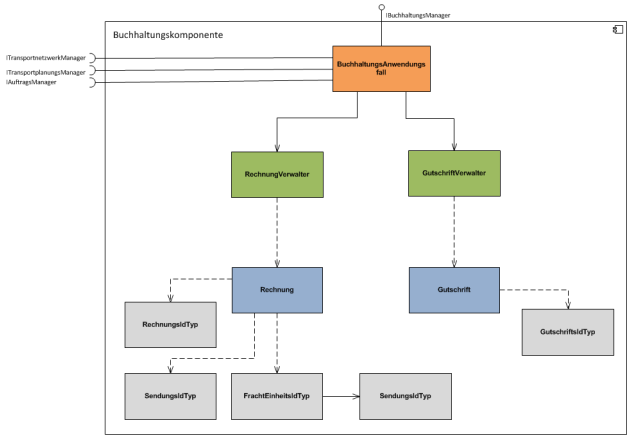

Abbildung 2.: HLS Komponenteninnensicht - Buchhaltungskomponente

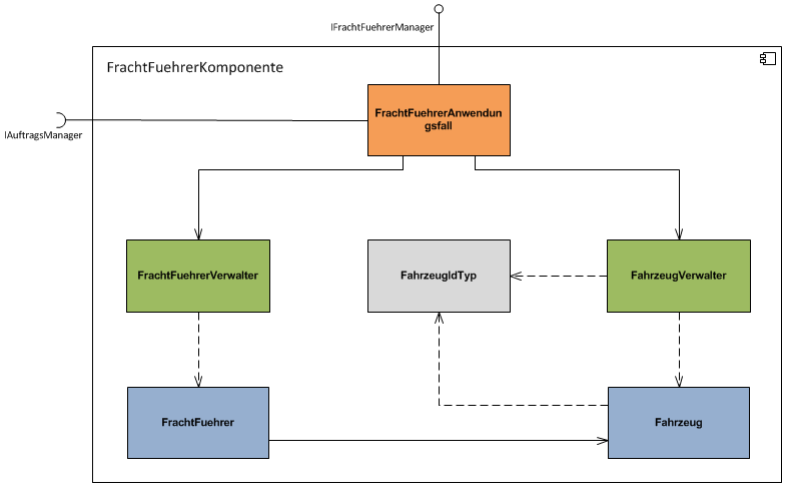

Abbildung 3.: HLS Komponenteninnensicht - Frachtführerkomponente

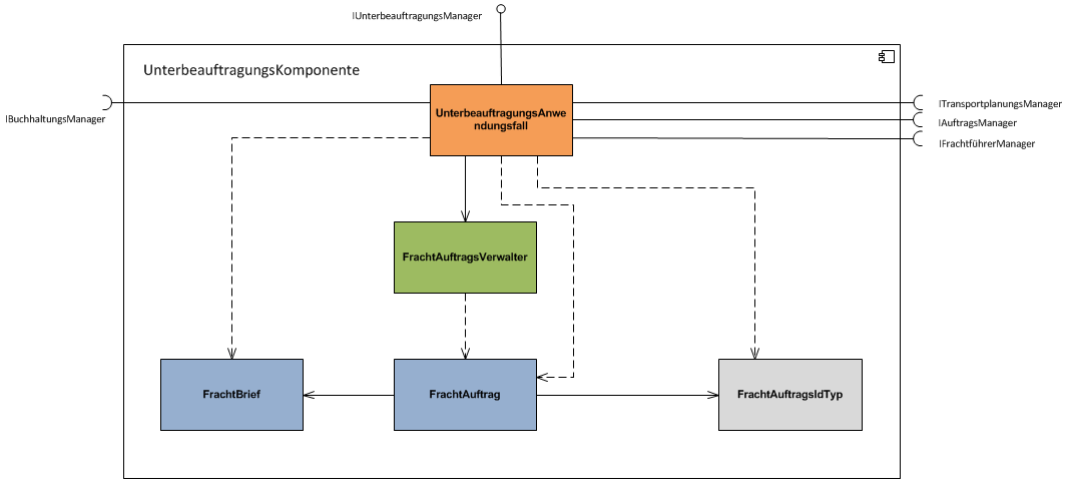

Abbildung 4.: HLS Komponenteninnensicht - Unterbeauftragungskomponente

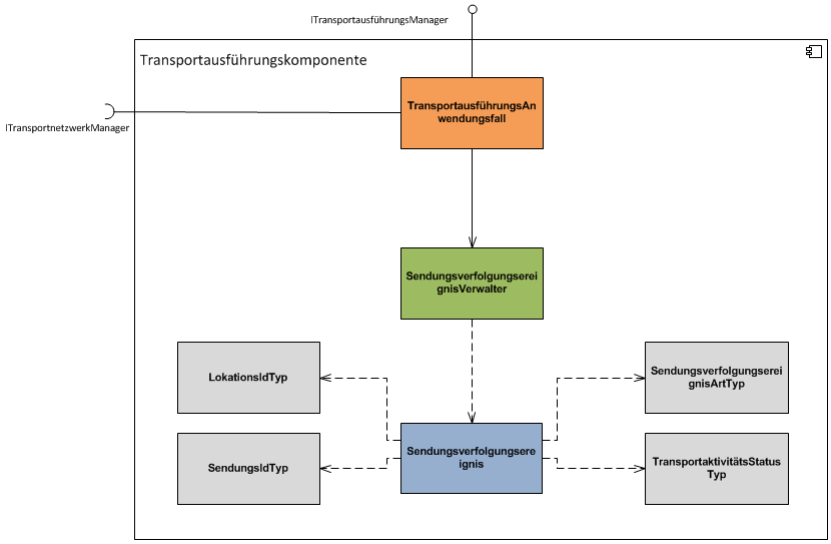

Abbildung 5.: HLS Komponenteninnensicht - Transportausführungskomponente

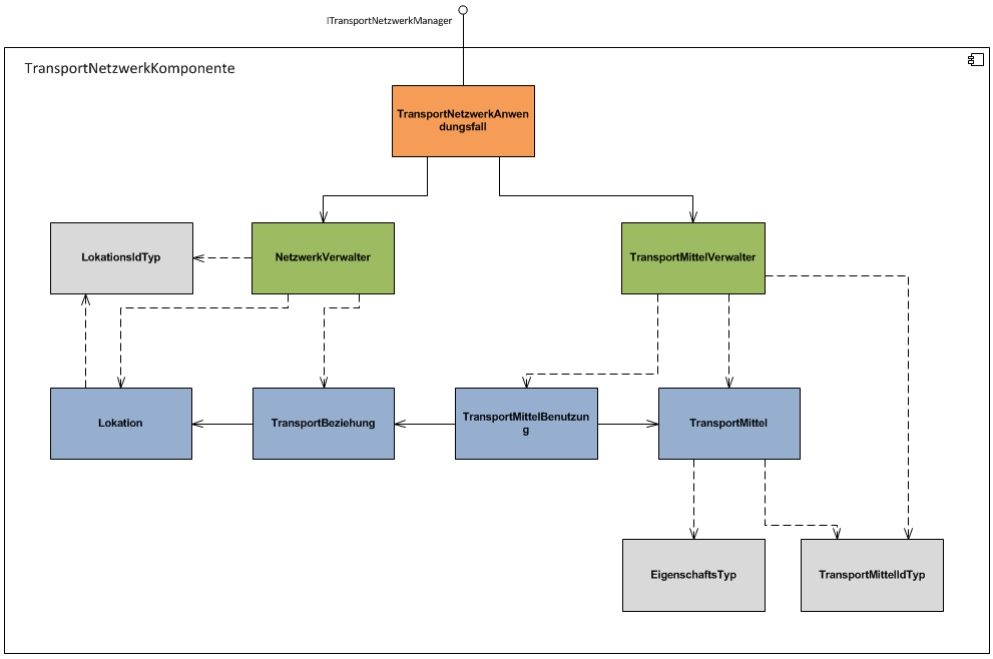

Abbildung 6.: HLS Komponenteninnensicht - Transportnetzwerkkomponente

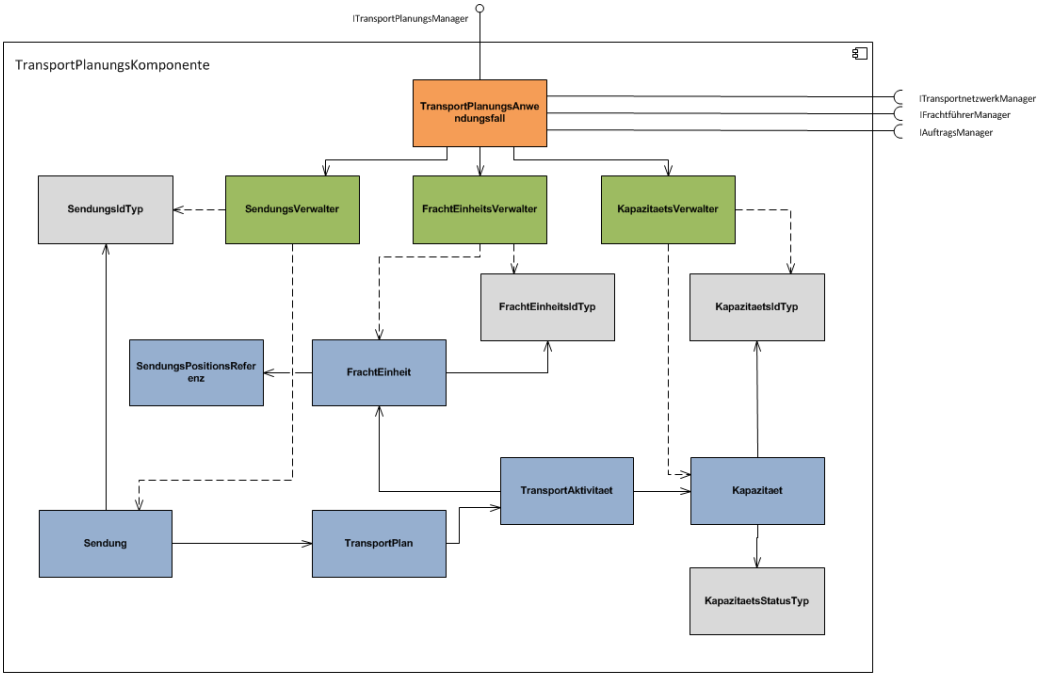

Abbildung 7.: HLS Komponenteninnensicht - Transportplanungskomponente

A.2. Dialoge

Die folgenden Abbildungen zeigen die Dialoge vom HLS.

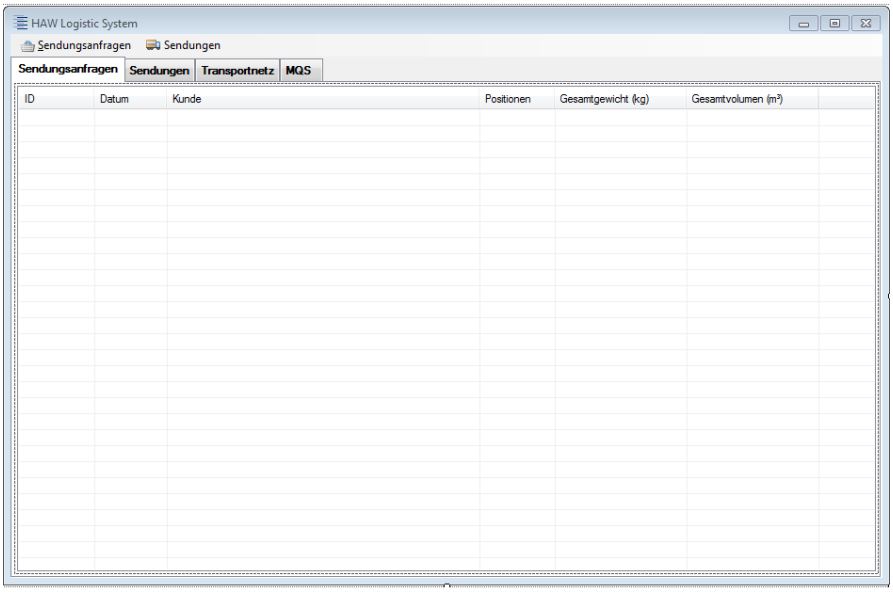

Abbildung 8.: HLS Dialog - Hauptfenster

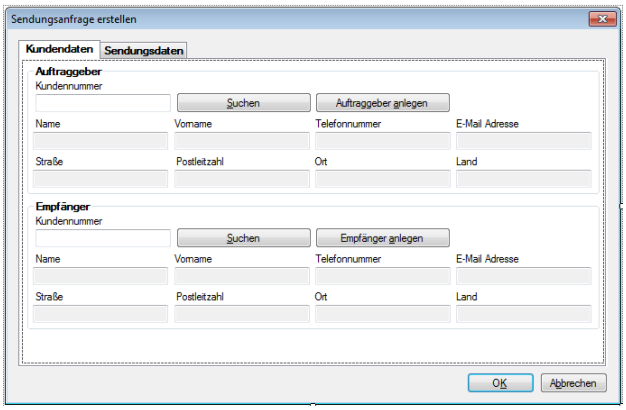

Abbildung 9.: HLS Dialog - Sendung hinzufügen

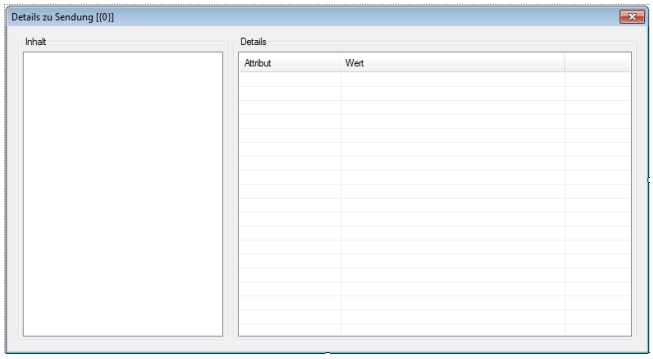

Abbildung 10.: HLS Dialog - Sendungsdetails

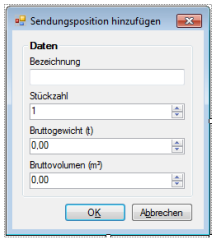

Abbildung 11.: HLS Dialog - Sendungsposition hinzufügen

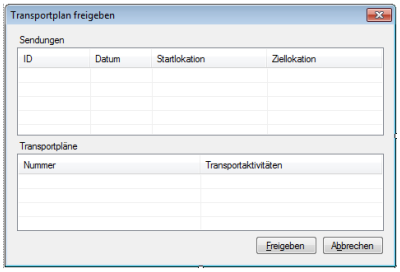

Abbildung 12.: HLS Dialog - Transportplan freigeben

B. Analyse

B.1. Anforderungen

Die folgende Tabelle zeigt das GQM-Verfahren und die daraus resultierenden Anforderungen für die Komponente Messung im Anwendungskern.

Ziel	Den AWK des HLS messen
Fragen	Was soll am HLS gemessen werden?
vereinfachte Fragen	Wie oft wird ein/e Komponente/Konnektor aufgerufen? Was für Fehler können auftreten? Wie oft wird ein Fehler im Konnektor bzw. in der Komponente geworfen? Wie oft wird auf die Datenbank zugegriffen von einer Komponente bzw. eines Konnektors aus? Wie viele Messsage Queue Nachrichten werden von einer Komponente bzw. eines Konnektors versendet? Wie viel Durchsatz hat ein Konnektor bzw. einne Komponente an Daten? Wie lange wird eine Komponente bzw. ein Konnektor durchlaufen?
Metriken	Fehler - Anzahl und Art des Fehlers mit Detailbeschreibung Anzahl Aufrufhäufigkeit Anzahl Datenbankzugriff Anzahl Message Queue Nachrichten Datendurchsatz Durchlaufzeit

Tabelle 1.: Anforderungen - GQM

B.2. Fachliches Datenmodell

B.2.1 Entitäten

Eine detaillierte Beschreibung der Entitäten zeigen die folgenden Tabellen:

Begriff	Fehler
Attribute	Id Name Art Details
Schlüssel	Die Identifizierungsnummer ist ein eindeutiger Schlüssel eines Fehlers
Beziehungen	**Nachricht**: Ein Fehler kann ein Nachrichtentyp sein. **Messung**: Am HLS werden Fehler gemessen. **Datenextrahieren**: Die Fehler werden aufbereitet. **Interpretation**: Die Fehler werden interpretiert. **Darstellung**: Die Fehler werden in der Benutzeroberfläche dargestellt.
Generalisierung	Ein Fehler ist eine Fehlermeldung
Spezialisierungen	Keine

Tabelle 2.: Fachliches Datenmodell - Entitäten - Fehler

Begriff	Zeit
Attribute	Id Name Art Zeit
Schlüssel	Die Idnetifizierungsnummer ist ein eindeutiger Schlüssel einer Zeit
Beziehungen	**Nachricht**: Eine Zeit kann ein Nachrichtentyp sein. **Messung**: Am HLS werden Zeiten gemessen mit einer Start- und Endzeit. **Datenextrahieren**: Die Zeiten werden aufbereitet. **Interpretation**: Die Zeiten werden interpretiert. **Darstellung**: Die Zeiten werden in der Benutzeroberfläche dargestellt.
Generalisierung	Eine Zeit ist ein zeitlicher Wert
Spezialisierungen	Durchschnittliche Zeit Einzelzeit

Tabelle 3.: Fachliches Datenmodell - Entitäten - Zeit

Begriff	Anzahl
Attribute	Id Name Art Anzahl
Schlüssel	Die Identifizierungsnummer ist ein eindeutiger Schlüssel einer Anzahl
Beziehungen	**Nachricht**: Eine Anzahl kann ein Nachrichtentyp sein. **Messung**: Am HLS werden Einheiten gezählt. **Datenextrahieren**: Die Anzahl wird aufbereitet. **Interpretation**: Die Anzahl wird interpretiert. **Darstellung**: Die Anzahl wird in der Benutzeroberfläche dargestellt.
Generalisierung	Eine Anzahl ist ein nummerischer Wert
Spezialisierungen	Einzelanzahl Gesamtanzahl

Tabelle 4.: Fachliches Datenmodell - Entitäten - Anzahl

C. Entwurf

C.1. Komponentenaussensichten und -innensichten

C.1.1 Messung

Im Folgenden ist die Aussensicht und die Innensicht der Komponente Messung zu sehen.

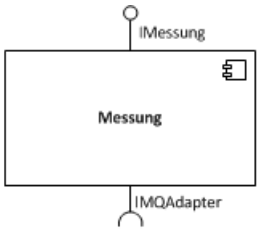

Abbildung 13.: Komponentenaussensicht Messung

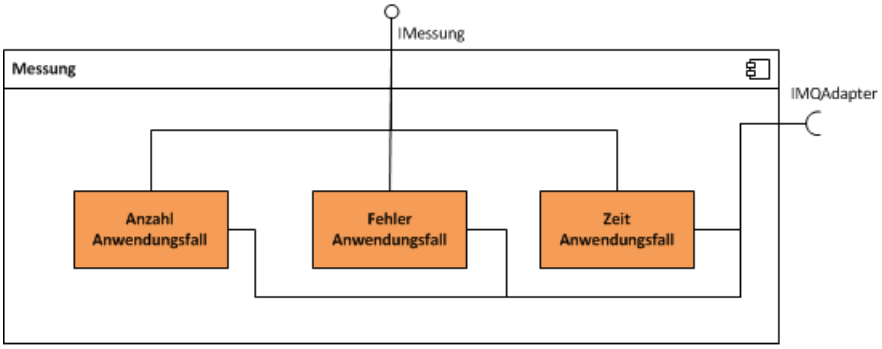

Abbildung 14.: Komponenteninnensicht Messung

C.1.2 Datenextrahieren

Im Folgenden ist die Aussensicht und die Innensicht der Komponente Datenextrahieren zu sehen.

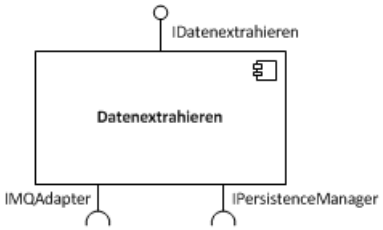

Abbildung 15.: Komponentenaussensicht Datenextrahieren

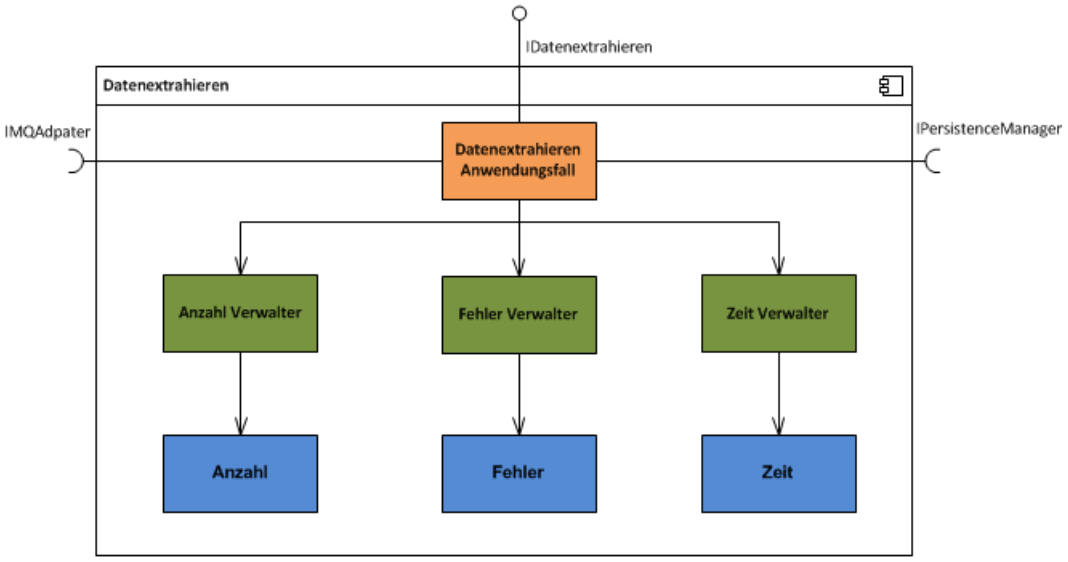

Abbildung 16.: Komponenteninnensicht Datenextrahieren

C.1.3 Interpretation

Im Folgenden ist die Aussensicht und die Innensicht der Komponente Interpretation zu sehen.

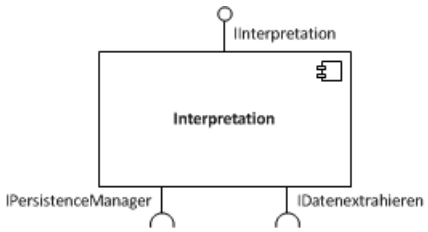

Abbildung 17.: Komponentenaussensicht Interpretation

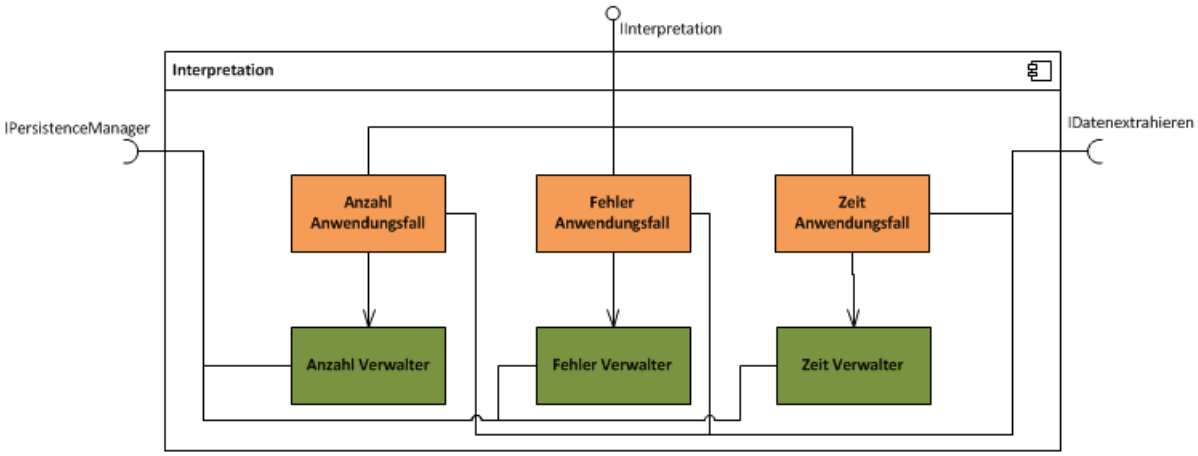

Abbildung 18.: Komponenteninnensicht Interpretation

D. Test

D.1. Unit-Tests

Name	Versende Nachricht vom Typ Anzahl
Beschreibung	Es wird eine neue Nachricht erstellt, die die gemessenen Daten enthält, diese Nachricht wird, nach der Erzeugung eines Senders, über den Apache Active Message Queuing (MQ) Server versendet.
Voraussetzungen	Apache Active Message Queuing (MQ) Server steht zur Verfügung, gemessene Daten stehen zu Verfügung
Eingabe	VersendeNachricht(“Anzahl", "BuchhaltungsKomponente", "DB-Aufruf")
Ausgabe	Nachricht wurde versendet
Verifizierung	Das Web Interface zeigt die versendeten Nachrichten an http://tfs-srs.informatik.haw-hamburg.de:8161/admin/ba_kb

Tabelle 5.: Testfall VN-1

Name	Versende Nachricht vom Typ Fehler
Beschreibung	Es wird eine neue Nachricht erstellt, die die gemessenen Daten enthält, diese Nachricht wird, nach der Erzeugung eines Senders, über den Apache Active Message Queuing (MQ) Server versendet.
Voraussetzungen	Apache Active Message Queuing (MQ) Server steht zur Verfügung, gemessene Daten stehen zu Verfügung
Eingabe	VersendeNachricht(“Fehler", "BuchhaltungsKomponente", "Datenbank nicht erreichbar")
Ausgabe	Nachricht wurde versendet
Verifizierung	Das Web Interface zeigt die versendeten Nachrichten an http://tfs-srs.informatik.haw-hamburg.de:8161/admin/ba_kb

Tabelle 6.: Testfall VN-2

Name	Versende Nachricht vom Typ Zeit
Beschreibung	Es wird eine neue Nachricht erstellt, die die gemessenen Daten enthält, diese Nachricht wird, nach der Erzeugung eines Senders, über den Apache Active Message Queuing (MQ) Server versendet.
Voraussetzungen	Apache Active Message Queuing (MQ) Server steht zur Verfügung, gemessene Daten stehen zu Verfügung
Eingabe	VersendeNachricht(“Zeit", "BuchhaltungsKomponente", 0.5)
Ausgabe	Nachricht wurde versendet
Verifizierung	Das Web Interface zeigt die versendeten Nachrichten an http://tfs-srs.informatik.haw-hamburg.de:8161/admin/ba_kb

Tabelle 7.: Testfall VN-3

Name	Empfange Nachrichten
Beschreibung	Nach der Erstellung eines Empfängers werden Nachrichten vom Typ Anzahl, Fehler und Zeit vom Apache Active Message Queuing (MQ) Server empfangen und nach speziellen Behandlungsroutinen behandelt.
Voraussetzungen	Apache Active Message Queuing (MQ) Server steht zur Verfügung, Nachrichten sind auf dem Apache Active Message Queuing (MQ) Server vorhanden
Eingabe	EmpfangeNachrichten()
Ausgabe	**Anzahl**: nachrichtenName = “Anzahl" nachrichtenArt = "BuchhaltungsKomponente" nachrichtenBeschreibung = "DB-Aufruf" **Fehler**: nachrichtenName = "Fehler" nachrichtenArt = "BuchhaltungsKomponente" nachrichtenBeschreibung = "Datenbank nicht erreichbar" **Zeit**: nachrichtenName = “Zeit" nachrichtenArt = "BuchhaltungsKomponente" nachrichtenBeschreibung = 0.5

Verifizierung	Der nachrichtenName, die nachrichtenArt und nachrichtenBeschreibung werden mit den erwarteten Werten verglichen

Tabelle 8.: Testfall EN-1

Name	Erstelle Anzahl
Beschreibung	Bei der Behandlungsroutine einer Nachricht werden nachrichtenName, nachrichtenArt und nachrichtenBeschreibung aus der Nachricht extrahiert, diese Daten werden über den Persistencemanager in der Datenbank gespeichert.
Voraussetzungen	Der Testfall EN-1 “Empfange Nachrichten"
Eingabe	ErstelleAnzahl(“Anzahl", "BuchhaltungsKomponente", "DB-Aufruf")
Ausgabe	nachrichtenName = “Anzahl" nachrichtenArt = "BuchhaltungsKomponente" nachrichtenBeschreibung = "DB-Aufruf"
Verifizierung	Der nachrichtenName, die nachrichtenArt und nachrichtenBeschreibung werden mit den erwarteten Werten verglichen

Tabelle 9.: Testfall EA-1

Name	Erstelle Fehler
Beschreibung	Bei der Behandlungsroutine einer Nachricht werden nachrichtenName, nachrichtenArt und nachrichtenBeschreibung aus der Nachricht extrahiert, diese Daten werden über den Persistencemanager in der Datenbank gespeichert.
Voraussetzungen	Der Testfall EN-1 “Empfange Nachrichten"
Eingabe	ErstelleAnzahl(“Fehler", "BuchhaltungsKomponente", "Datenbank nicht erreichbar")

Ausgabe	nachrichtenName = “Fehler" nachrichtenArt = "BuchhaltungsKomponente" nachrichtenBeschreibung = "Datenbank nicht erreichbar"
Verifizierung	Der nachrichtenName, die nachrichtenArt und nachrichtenBeschreibung werden mit den erwarteten Werten verglichen

Tabelle 10.: Testfall EF-1

Name	Erstelle Zeit
Beschreibung	Bei der Behandlungsroutine einer Nachricht werden nachrichtenName, nachrichtenArt und nachrichtenBeschreibung aus der Nachricht extrahiert, diese Daten werden über den Persistencemanager in der Datenbank gespeichert.
Voraussetzungen	Der Testfall EN-1 “Empfange Nachrichten"
Eingabe	ErstelleAnzahl(“Zeit", "BuchhaltungsKomponente", 0.5)
Ausgabe	nachrichtenName = “Zeit" nachrichtenArt = "BuchhaltungsKomponente" nachrichtenBeschreibung = 0.5
Verifizierung	Der nachrichtenName, die nachrichtenArt und nachrichtenBeschreibung werden mit den erwarteten Werten verglichen

Tabelle 11.: Testfall EZ-1

Name	Hole alle Daten vom Typ Anzahl
Beschreibung	Hole alle Daten aus der Datenbank, aus der Tabelle Anzahl
Voraussetzungen	Der Testfall EA-1 “Erstelle Anzahl"
Eingabe	GibAlleAnzahl()
Ausgabe	IList<Anzahl>

Verifizierung	Die Anzahl der Einträge in der Tabelle Anzahl wird mit der Anzahl der Listeneinträge verglichen

Tabelle 12.: Testfall GA-1

Name	Hole alle Daten vom Typ Fehler
Beschreibung	Hole alle Daten aus der Datenbank, aus der Tabelle Fehler
Voraussetzungen	Der Testfall EF-1 “Erstelle Fehler"
Eingabe	GibAlleFehler()
Ausgabe	IList<Fehler>
Verifizierung	Die Anzahl der Einträge in der Tabelle Fehler wird mit der Anzahl der Listeneinträge verglichen

Tabelle 13.: Testfall GA-2

Name	Hole alle Daten vom Typ Zeit
Beschreibung	Hole alle Daten aus der Datenbank, aus der Tabelle Zeit
Voraussetzungen	Der Testfall EZ-1 “Erstelle Zeit"
Eingabe	GibAlleZeit()
Ausgabe	IList<Zeit>
Verifizierung	Die Anzahl der Einträge in der Tabelle Zeit wird mit der Anzahl der Listeneinträge verglichen

Tabelle 14.: Testfall GA-3

Name	Gib Dictionary vom Typ Anzahl
Beschreibung	Konvertiert die Datenbankeinträge in ein spezielles Format für die Benutzeroberfläche
Voraussetzungen	Der Testfall GA-1 “Hole alle Daten vom Typ Anzahl"
Eingabe	GibAnzahlListe()
Ausgabe	IDictionary<string, List<string»

Verifizierung	Die Anzahl der Einträge in der Tabelle Anzahl wird mit der Anzahl der Dictionaryeinträge verglichen

Tabelle 15.: Testfall GL-1

Name	Gib Dictionary vom Typ Fehler
Beschreibung	Konvertiert die Datenbankeinträge in ein spezielles Format für die Benutzeroberfläche
Voraussetzungen	Der Testfall GA-2 “Hole alle Daten vom Typ Fehler"
Eingabe	GibFehlerListe()
Ausgabe	IDictionary<string, List<string»
Verifizierung	Die Anzahl der Einträge in der Tabelle Fehler wird mit der Anzahl der Dictionaryeinträge verglichen

Tabelle 16.: Testfall GL-2

Name	Gib Dictionary vom Typ Zeit
Beschreibung	Konvertiert die Datenbankeinträge in ein spezielles Format für die Benutzeroberfläche
Voraussetzungen	Der Testfall GA-3 “Hole alle Daten vom Typ Zeit"
Eingabe	GibZeitListe()
Ausgabe	IDictionary<string, List<double»
Verifizierung	Die Anzahl der Einträge in der Tabelle Zeit wird mit der Anzahl der Dictionaryeinträge verglichen

Tabelle 17.: Testfall GL-3

Name	Gib Dictionary mit Einzeleinträgen vom Typ Anzahl
Beschreibung	Konvertiert die Datenbankeinträge in ein spezielles Format, welches Einzeleinträge enthält, für die Benutzeroberfläche
Voraussetzungen	Der Testfall GA-1 “Hole alle Daten vom Typ Anzahl"

Eingabe	GibAnzahlAnzahl()
Ausgabe	IDictionary<string, Dictionary<string,int»
Verifizierung	Die Anzahl der Einträge in der Tabelle Anzahl wird mit der Anzahl der Dictionaryeinträge verglichen

Tabelle 18.: Testfall GE-1

Name	Gib Dictionary mit Einzeleinträgen vom Typ Fehler
Beschreibung	Konvertiert die Datenbankeinträge in ein spezielles Format, welches Einzeleinträge enthält, für die Benutzeroberfläche
Voraussetzungen	Der Testfall GA-2 “Hole alle Daten vom Typ Fehler"
Eingabe	GibFehlerAnzahl()
Ausgabe	IDictionary<string, int>
Verifizierung	Die Anzahl der Einträge in der Tabelle Fehler wird mit der Anzahl der Dictionaryeinträge verglichen

Tabelle 19.: Testfall GE-2

Name	Gib Dictionary mit Einzeleinträgen vom Typ Zeit
Beschreibung	Konvertiert die Datenbankeinträge in ein spezielles Format, welches Einzeleinträge enthält, für die Benutzeroberfläche
Voraussetzungen	Der Testfall GA-3 “Hole alle Daten vom Typ Zeit"
Eingabe	GibZeitAnzahl()
Ausgabe	IDictionary<string, double>
Verifizierung	Die Anzahl der Einträge in der Tabelle Zeit wird mit der Anzahl der Dictionaryeinträge verglichen

Tabelle 20.: Testfall GE-3

E. Inhalt der beigefügten CD

* Source Code des Fallbeispiels HAW-Logistics-System (HLS)
* Source Code des Leitstandes
* WPF Toolkit
* Arbeit in pdf Format

Glossar

$DIN3166-1$ Listet zweibuchstabige Länderkürzel auf. Die Länder werden mit Großbuchstaben gekennzeichnet. Z.B. für Deutschland steht das Länderkürzel DE., Seite 25

$HotSpot$ In einem Framework eine definierte Schnittstelle, die durch anwendungsspezifischen Code erweitert werden kann., Seite 3

Literaturverzeichnis

[Code Project - Chris Anderson 2007] Code Project - Chris Anderson: *Bild vom WPF - Raster- und Vektorbasierte Darstellung.* 2007. – URL http://www.codeproject.com/KB/books/essential_wpf/01fig03.jpg. – Online, aufgerufen am 5. Januar 2012

[Doberenz und Gewinnus 2010] Doberenz, Walter ; Gewinnus, Thomas: *Softwaretechnik - Mit Fallbeispielen aus realen Entwicklungsprojekten.* München : Cal Hanser Verlag, 2010. – ISBN 978-3-446-42118-9

[Ehmke u. a. 2011] Ehmke, Nils ; Hoorn, André van ; Jung, Reiner: *Kieker.* 2011. – URL http://kent.dl.sourceforge.net/project/kieker/kieker/kieker-1.4/kieker-1.4_userguide.pdf. – Online; aufgerufen am 3. Dezember 2011

[Freemann u. a. 2008] Freemann, Eric ; Freemann, Elisabeth ; Sierra, Kathy ; Bates, Bert ; Schulten, Lars ; Buchholz, Elke: *Entwurfsmuster von Kopf bis Fuß.* 4. korrigierter Nachdruck. Köln : O'Reilly Verlag GmbH & Co. KG, 2008. – Kapitel 7. – ISBN 978-3-89721-421-7

[Grechenig u. a. 2010] Grechenig, Thomas ; Bernhart, Mario ; Breiteneder, Roland ; Kappel, Karin: *Visual C# - Grundlagen und Profiwissen.* München : Pearson Studium, 2010. – ISBN 978-3-86894-007-7

[Huber 2010] Huber, Thomas C.: *Windows Presentation Foundation - Das umfassende Handbuch.* Bonn : Galileo Press, 2010. – Kapitel 1, 2. – ISBN 978-3-8362-1538-1

[IEEE-Sandard 1061 1998] IEEE-Sandard 1061: *Software Quality Metric.* 1998. – URL http://standards.ieee.org/findstds/standard/1061-1998.html. – Online; aufgerufen am 23. November 2011

[IEEE Xplore 2011] IEEE Xplore: *IEEE Standard 1471-2000 / ISO/IEC 42010.* 2011. – URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4278472. – Online, aufgerufen am 31. Dezember 2011

[Langewiesche 2003] Langewiesche, Thorsten: *Betriebszentralen / Leitzentralen.* 2003. – URL `http://www.rail-control.de/bz___rzue.htm`. – Online; aufgerufen am 31. Dezember 2011

[Ludewig und Lichter 2010] Ludewig, Jochen ; Lichter, Horst: *Software Engineering - Grundlagen, Menschen, Prozesse, Techniken.* 2., überarbeitete, aktualisierte und ergänzte Auflage. Heidelberg : dpunkt.verlag GmbH, 2010. – Kapitel 14, 16-19. – ISBN 978-3-89864-662-8

[MSDN 2011a] MSDN: *Shape.StrokeLineJoin-Eigenschaft.* 2011. – URL `http://msdn.microsoft.com/de-de/library/system.windows.shapes.shape.strokelinejoin.aspx`. – Online; aufgerufen am 13. November 2011

[MSDN 2011b] MSDN: *Shape.StrokeThickness-Eigenschaft.* 2011. – URL `http://msdn.microsoft.com/de-de/library/system.windows.shapes.shape.strokethickness.aspx`. – Online; aufgerufen am 13. November 2011

[MSDN 2011c] MSDN: *Stretch-Enumeration.* 2011. – URL `http://msdn.microsoft.com/de-de/library/system.windows.media.stretch.aspx`. – Online; aufgerufen am 13. November 2011

[MSDN 2012] MSDN: *SQLDriverConnect.* 2012. – URL `http://msdn.microsoft.com/en-us/library/aa177865%28v=sql.80%29.aspx`. – Online; aufgerufen am 3. März 2012

[Nagios 2011] Nagios: *Nagios.* 2011. – URL `http://www.nagios.org/about/overview`. – Online; aufgerufen am 3. Dezember 2011

[Siedersleben 2004] Siedersleben, Johannes: *Moderne Softwarearchitektur - Umsichtig planen, robust bauen mit Quasar.* korrigierter Nachdruck 2005. Heidelberg : dpunkt.verlag GmbH, 2004. – ISBN 3-89864-292-5

[Stefan Sarstedt 2010a] Stefan Sarstedt: *HLS Lastenheft WS 2010.* 2010

[Stefan Sarstedt 2010b] Stefan Sarstedt: *SE1 Foliensatz.* 2010. – Kapitel 2, 3, 5

[Stefan Sarstedt 2010c] Stefan Sarstedt: *Technik und Architektur.* 2010

[Stefan Sarstedt 2010d] Stefan Sarstedt: *Technik und Architektur - Quasar-Standardarchitektur.* 2010

[Stropek und Huber 2008] Stropek, Rainer ; Huber, Karin: *WPF und XAML Programmierhandbuch*. Siegen : entwickler.press, 2008. – Kapitel 2, 3. – ISBN 978-3-939084-60-0

[Uni Stuttgart 2009] Uni Stuttgart: *Bild von der Betriebszentrale*. 2009. – URL `http://www.uni-stuttgart.de/vwi/bilder/exkursionen/2009_BZ-Karlsruhe/CIMG1547.jpg`. – Online, aufgerufen am 31. Dezember 2011

[Universität Rostock 2008] Universität Rostock: *Monitoring*. 2008. – URL `http://www.geoinformatik.uni-rostock.de/einzel.asp?ID=1214287037`. – Online; aufgerufen am 02. Oktober 2011

[Webster 2011] Webster: *Monitoring*. 2011. – URL `http://www.merriam-webster.com/dictionary/monitoring`. – Online; aufgerufen am 02. Oktober 2011

[Weigend u. a. 2011] Weigend, Johannes ; Siedersleben, Johannes ; Adersberger, Josef: *Dynamische Analyse mit dem Software-EKG*. 2011. – URL `http://www.springerlink.com/content/k66t0p8725226631/`. – Online; aufgerufen am 3. Dezember 2011

[Wikipedia 2005] Wikipedia: *Bild vom Leitstand*. 2005. – URL `http://upload.wikimedia.org/wikipedia/commons/thumb/0/03/Leitstand_2.jpg/500px-Leitstand_2.jpg`. – Online, aufgerufen am 16. Oktober 2011

[Wikipedia 2011a] Wikipedia: *Extensible Application Markup Language*. 2011. – URL `http://de.wikipedia.org/wiki/Extensible_Application_Markup_Language`. – Online; aufgerufen am 11. November 2011

[Wikipedia 2011b] Wikipedia: *Framework*. 2011. – URL `http://de.wikipedia.org/wiki/Framework`. – Online; aufgerufen am 02. Oktober 2011

[Wikipedia 2011c] Wikipedia: *Kohlekraftwerk*. 2011. – URL `http://de.wikipedia.org/wiki/Kohlekraftwerk`. – Online, aufgerufen am 16. Oktober 2011

[Wikipedia 2011d] Wikipedia: *Leitstand*. 2011. – URL `http://de.wikipedia.org/wiki/Leitstand`. – Online; aufgerufen am 15. Oktober 2011

[Wikipedia 2011e] Wikipedia: *Monitoring*. 2011. – URL `http://de.wikipedia.org/wiki/Monitoring`. – Online; aufgerufen am 28. September 2011

[Wikipedia 2011f] Wikipedia: *Softwaremetrik*. 2011. – URL `http://de.wikipedia.org/wiki/Softwaremetrik`. – Online; aufgerufen am 20. November 2011

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit, laut Prüfungsordnung APSO-TI-BM nach §16(5), ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 27. April 2012

Ort, Datum

Unterschrift