

Thema

BACHELORARBEIT // PROJEKTARBEIT // STUDIENARBEIT

des Studienganges Studiengang

an der Dualen Hochschule Baden-Württemberg Name des Standortes

von

Klara Mustermann

Abgabedatum

Bearbeitungszeitraum

12 Wochen

Matrikelnummer, Kurs

Matrikelnummer, Kurskürzel

Dualer Partner

Firmenname, Stadt

Betreuer*in des Dualen Partners

Titel Vorname Nachname

Gutachter*in der Dualen Hochschule

Titel Vorname Nachname

*(Betreuer*in bzw. Gutachter*in ggfs. bei Projekt- und Studienarbeiten streichen)*

Inhaltsverzeichnis

1	ABBILDUNGSVERZEICHNIS	3
2	ABKÜRZUNGSVERZEICHNIS.....	4
	EHRENWÖRTLICHE ERKLÄRUNG.....	5
3	EINLEITUNG	7
3.1	EINSATZORT	7
3.2	SYSTEME.....	8
3.3	PROBLEMSTELLUNG	9
4	AUFGABENSTELLUNG.....	10
4.1	DOKUMENTATION	10
4.2	MIGRATION	11
5	VORÜBERLEGUNGEN.....	12
5.1	TROUBLE-TICKET-SYSTEME.....	12
5.2	IST-ANALYSE	15
5.3	DATENBANKIDE	17
5.4	MIGRATION	22
6	UMSETZUNG.....	26
6.1	VORHANDENE DOKUMENTATION	26
6.2	DATENBANKSCHEMA.....	26
6.3	MELDESYSTEMFUNKTIONALITÄTEN.....	28
6.4	GRAFISCHE OBERFLÄCHE	29
6.5	VB6-QUELLCODE.....	31
6.6	MIGRATION VON SYSTEMTEILEN	32
7	ZUSAMMENFASSUNG UND AUSBLICK.....	33
8	LITERATURVERZEICHNIS.....	34
9	GLOSSAR	37
10	ANHANG	38

1 **Abbildungsverzeichnis**

- Abbildung 1: Struktur eines TTS nach Veríssimo, P./Rodrigues, L. (2004), S. 568... 15
- Abbildung 2: Übersicht über die Datenbank-Modelle nach Hering, E./ Gutekunst, J./Dyllong, U. (2000), S. 183..... 18
- Abbildung 3: Meldungsdetails - alte Benutzeroberfläche30
- Abbildung 4: Meldungsdetails - neue Benutzeroberfläche31

2 Abkürzungsverzeichnis

DBMS ..*Datenbank-Managementsystem*
MVC.....*Model-View-Controller*
TIS.....*Technische Informationssysteme*

TTS.....*Trouble-Ticket-System*
UML*Unified Modeling Language*

3 Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Aus den benutzten Quellen, direkt oder indirekt, übernommene Gedanken habe ich als solche kenntlich gemacht.

Diese Arbeit wurde bisher in gleicher oder ähnlicher Form oder auszugsweise noch keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Ort, Datum

Unterschrift

Hinweis:

Dieser Bericht dient nur zur Übung im Seminar und darf nicht als Vorlage für den eigenen Praxisbericht verwendet werden.

gez.

Wolfgang Stark

4 Einleitung

Die IT-Branche zeichnet sich seit je her durch eine rasante Entwicklung und seine Schnelligkeit aus.

Gegenüber den schnellen Neuerungen der Technologien steht das Problem der ebenso schnellen Alterung von Systemen. Durch den ständigen Veränderungsprozess sind (v. a. Software-herstellende) Unternehmen gezwungen, sich in immer wieder neuen Gegebenheiten ein zu arbeiten, ältere (Software-) Lösungen anzupassen und eventuell um neue Möglichkeiten zu erweitern. In einigen Fällen kann es dazu kommen, dass es nötig ist ein gesamtes Programm abzuändern. Dies ist beispielsweise bei älteren Anwendungen der Fall, die in der Programmiersprache Visual Basic 6.0 (im Weiteren VB6 genannt) geschrieben sind. Für neuere Microsoft-Betriebssysteme ist der Extended Support für die Entwicklungsumgebung VB6 beendet und es wird „... dringend empfohlen die existierenden Anwendungen innerhalb des Vista-Support-Life-Cycles auf .NET Technologien umzustellen. Anstatt der nicht mehr unterstützten Visual Basic 6.0 Entwicklungsumgebung sollten Entwickler Visual Studio 2008 zusammen mit dem Interop-Forms-Toolkit und der unterstützten Visual Basic 6.0 Laufzeit nutzen.“¹

Da eine ganz neue Konzipierung der Anwendung unnötig (bspw. ist ein sinnvolles Datenbankschema bereits vorhanden) und sehr kostenintensiv ist, liegt die Lösung oftmals bei einer sogenannten Migration.

Im Rahmen dieser Arbeit wurden nötige Vorbereitungen und eine mögliche Umsetzung einer Migration eines kleinen Trouble-Ticket-Systems betrachtet.

4.1 Einsatzort

Das zu betrachtende Trouble-Ticket-System wird bei in der Firma..... eingesetzt, deren erstellte IT-Lösungen unter dem Begriff Product Lifecycle Management (PLM) fallen. Diese Anwendungen werden größtenteils für Sie dienen somit der Verwaltung von

¹ Microsoft Corporation (o. J.)

Daten über Automobile in allen Lebenszyklenstadien, von der Planung bis zur Entsorgung. Abteilungsintern werden diese Applikationen unter dem Begriff Technische Informationssysteme (TIS) zusammengefasst. Das System, mit dem sich diese Arbeit befasst, dient dem effektiven Management von Fehlermeldungen und Änderungswünschen, die die größten der zu TIS zählenden Anwendungen betreffen.

4.2 Systeme

Die größten und am meisten referenzierten Systeme der TIS sollen im Folgenden vorgestellt werden, um einen besseren Überblick des Anwendungsbereiches der zu betrachtenden Applikation zu geben. Jedes der unten aufgeführten Systeme wird in einem eigenen Projekt bearbeitet.

FiNAS

Das Fahrzeug und Versuchsteile Informations- und Admⁱnistrationssystem ist ein System für die Dokumentation und Ergebnisbewertung von Fahrzeug- sowie Fahrzeugobjekttests. Der gesamte Lebenszyklus eines Fahrzeugs oder Fahrzeugobjekts kann in FiNAS nachverfolgt werden.

SAFiR

Das System für Aggregate- und Fahrzeugentwⁱcklungsplaner ist ein System für den Planungsprozess von Fahrzeugen. In diesem System wird geplant, wie und wo Fahrzeugobjekte in ein Fahrzeug verbaut werden.

FEPLAS

Das **F**ahrzeug-**E**inplanungs-**S**ystem ist das kleinste der drei vorgestellten TIS und ist zuständig für die Einplanung und den Einsatz von Fahrzeuge. Dies gilt für Fahrzeuge, die sowohl innerhalb der Firma zu Erprobungszwecken eingeplant als auch extern anderen Fremdfirmen überlassen werden.

4.3 Problemstellung

Infolge der Komplexität von FiNAS, SAFiR und FEPLAS können viele Fehler in der Implementierung auftreten. Wegen des großen Anwendungsbereiches „Fahrzeugmanagement“, der vielen technischen Wandlungen unterliegt, existieren immer neue Veränderungs- und Erweiterungswünschen des Kunden.

Zum Management dieser unterschiedlichen Anforderungen wurde das in dieser Arbeit zu betrachtende System von unserer Firma entwickelt: Das Fehlermeldesystem (kürzer auch Meldesystem). Dieses System gibt dem Kunden die Möglichkeit, entdeckte Fehler schnell und direkt an das zuständige Projektteam zu melden und erleichtert den Projektteams die Fehlerbehebung.

Ursprünglich war das Meldesystem nur für FiNAS angelegt, später wurden aber in einer großen Umgestaltung SAFiR und FEPLAS mit eingebunden. Über die Jahre wurden an dem Meldesystem wie an jeder anderen beständig genutzten Applikation zahlreiche Änderungen vorgenommen. Derzeit wird das Meldesystem in der Version 4.1.18 genutzt, was auf die große Anzahl von funktionalen Änderungen hindeutet. Mit den durchgeführten Umgestaltungen wurde das System immer unübersichtlicher und vor allem bei der im Hintergrund stehenden Datenbank wurden Relationen und Benutzung der Tabellen undurchsichtig.

Des Weiteren liegt ein Problem beim Quellcode des Systems, der genauso undurchsichtig wie die Datenbank geworden ist und zudem in VB6 geschrieben wurde, bei dem Microsoft Corporation empfiehlt, auf eine modernere .NET Technologie umzustellen.²

² vgl. Microsoft Corporation (O. J.)

5 Aufgabenstellung

5.1 Dokumentation

Die Aufgabe für diesen Praxisberichts bestand zum einen Teil daraus, das Fehlermeldesystem in seinem aktuellen Zustand, das heißt mit seinen derzeitigen Prozessen und Funktionalitäten, zu dokumentieren. Es existierte bereits eine Systemdokumentation, welche aber nur sparsam mit den Veränderungen des Meldesystems gepflegt wurde und nur wenig über die gegenwärtige Situation aussagte.

Die zu erstellende Dokumentation sollte zukünftig neuen Projektteilnehmern die Arbeit mit dem Meldesystem veranschaulichen und erleichtern sowie den gesamten Prozess des Fehlermanagements übersichtlich darstellen. Zudem kann auf Grundlage einer zeitgemäßen Dokumentation leichter erkannt werden, welche Teile des Meldesystem-Sourcecodes überflüssig und welche Tabellen der Oracle-Datenbank unbenutzt sind bzw. welche sinnvollen Erweiterungs- und Optimierungsmöglichkeiten vorhanden sind.

Für die Dokumentation sollte eine Ist-Analyse des vorliegenden Systems durchgeführt werden, welche Aussagen über das derzeitige Layout, die vorhandenen Funktionalitäten und den technischen Hintergrund, das heißt das dahinterliegende Datenbankschema, des Fehlermeldesystems enthält.

Die Datenbankschema-Analyse erfolgte dabei auf einer Testdatenbank. Bei dieser werden wöchentlich die vorhandenen Datensätze der alltäglichen genutzten Datenbank synchronisiert. Hauptsächlich wird sie für Weiterentwicklungsteste und –anfänge genutzt.

Ziel dieser Aufgabe war somit das Erstellen einer Systemdokumentation. Damit verbunden war die Auseinandersetzung mit einem Trouble-Ticket-System und seinen Prozessen. Zudem erfolgte eine Einsicht in das Thema Datenbanken.

Die erstellte Dokumentation bildete eine Voraussetzung für die Durchführung des zweiten Teils der Aufgabe.

5.2 Migration

Der zweite Teil der Aufgabe bestand darin, einige erste Teile des veralteten Meldesystems zu migrieren. Wie dieser Begriff definiert wird, ist in Kapitel [7.4](#) beschrieben.

Die aktualisierte Dokumentation, die im ersten Teil der Aufgabe zu erstellt gewesen war, sollte dabei als Vorbereitung und Grundlage für die kommende Teilmigration dienen. Auch sollte die Migration an sich verständlich dokumentiert werden, sodass eine problemlose Fortsetzung der Migration möglich ist und mit der Zeit das gesamte Fehlermeldesystem vollständig migriert werden kann.

Eine Migration des Meldesystems wurde als nötig angesehen, da das System veraltet ist. Es sollte modernisiert und dabei strukturiert werden. Die Migration bietet die Möglichkeit, das Meldesystem geänderten Anforderungen anzupassen.

Zusätzlich gibt eine Migration die Möglichkeit, das System strukturiert zu implementiert und damit zukünftig leichter erweitern und ausbauen zu können. Bei der Migration kann darauf geachtet werden, dass Funktionalitäten so effektiv wie möglich vom System angeboten werden, sodass das Meldesystem ein schnelles Orientieren und einfaches Erstellen von Fehlermeldungen ermöglicht.

Ziel ist somit auf Grundlage der vorher durchgeführten Ist-Analyse eine teilweise Umsetzung und leichte Überarbeitung des Systems auszuführen, um eine größere Effizienz zu erreichen. Die Implementierung sollte mit Visual Studio in der .NET Programmiersprache C# durchgeführt werden, um sich in die im Projekt eingesetzte Programmiersprache einzuarbeiten. Bei der Implementierung sollten bei der Ist-Analyse gewonnene Erkenntnisse in einer verbesserten Benutzeroberfläche umgesetzt werden.

6 Vorüberlegungen

6.1 Trouble-Ticket-Systeme

Das System, welches Gegenstand dieser Arbeit ist, wird als Trouble-Ticket-System (abgekürzt TTS) bezeichnet.

Bei einem TTS handelt es sich um eine datenbankgestützte Software, die Anfragen von Benutzern (meist zu einer bestimmten anderen Anwendung) aufnimmt und deren strukturierte Bearbeitung unterstützt.³ Nach Verissimo kann ein TTS auch als System einer Sammlung von Unregelmäßigkeiten und möglichen Rückmeldungen angesehen werden. Dieses System erhält Anfragen (entweder vom Benutzer erstellt oder von einem System generiert), speichert und verarbeitet sie und bereitet anschließend eine entsprechende Rückmeldung vor. In einem TTS ist es möglich, Anfragen die für längere Zeit ungelöst bleiben nach zu verfolgen.⁴

Bezüglich der genauen Einordnung dieser Art von Systemen gibt es zwei, leicht unterschiedliche Auffassungen: Einerseits werden Trouble-Ticket-Systeme als synonym zu Helpdesk-Systemen angesehen. Andererseits werden TTS als eine Unterklasse oder sinnvolle Ergänzung von Helpdesk-Systemen bezeichnet. Die Ursache dieser unterschiedlichen Ansichten liegt darin begründet, dass Helpdesk-Mitarbeiter durch in der Regel durch ein TTS unterstützt werden.⁵

Das hier vorliegende System ist nicht mit einem speziellen Helpdesk-System verbunden und steht für sich allein.

Die prinzipielle Arbeitsweise von Trouble-Ticket-Systemen kann als Abbildung der „...Meldungsbearbeitung mit Laufzetteln per Software...“⁶ aufgefasst werden. Der physische Laufzettel wird in einem TTS zu einem digitalen Trouble Ticket oder kurz Ticket. (Im Projektumfeld dieser Arbeit wird zudem der Begriff Meldung verwendet.) Auf diesem werden Anfragen (und jegliche relevante Informationen zu diesen

³ Vgl. Gausemeier, J./Plass, C./Wenzelmann, C. (2009), S. 430

⁴ Vgl. Verissimo, P./Rodrigues, L. (2004), S. 567 f.

⁵ Vgl. Kühne, C./Pauer, D./Rautenstrauch, C. (2003), S. 886

⁶ Ebenda, S. 886

Anfragen) vermerkt und es bleibt im System vorhanden, bis es nach einer Abarbeitung der Anfrage geschlossen wird. Die für das Ticket gestellten Anfragen können dabei verschiedener Art sein: Beispielsweise können sie Störungsmeldungen oder Serviceanfragen, die Änderungen der Software betreffen, bilden.⁷ Ein zugeteilter Bearbeiter fügt dem Ticket durchgeführte Problemlösungsmaßnahmen hinzu und schließt das Ticket ab. Ist es dem Bearbeiter nicht möglich das bestehende Problem zu lösen, leitet er das Ticket an einen anderen (im besten Falle qualifizierteren) Bearbeiter weiter. Dieser Vorgang wird auch als Forwarding bezeichnet.⁸ Das Ticket kann aktuelle Informationen zur Problemlösung und vergangene Maßnahmen enthalten, sowie während seiner Bearbeitung klassifiziert (z. B. nach Ticketart und –priorität) werden.⁹

Kühne sieht die Hauptaufgabe eines Trouble-Ticket-Systems¹⁰ in der Unterstützung mehrere Experten bei der Zusammenarbeit zur Lösung einer bestimmten Anfrage. Dies ist besonders wichtig, da Probleme im Allgemeinen nacheinander von verschiedenen Experten abgearbeitet werden. Durch eine Dokumentation der Maßnahmen im Ticket wird ohne zusätzliche Kommunikation deutlich, welche Schritte ein vorheriger Bearbeiter bereits durchgeführt hat. Zu dieser Hauptaufgabe zählt demnach ebenso eine Entstehung effizienter Kommunikationsmechanismen.

Des Weiteren übernimmt in Trouble-Ticket-Systemen laut Veríssimo folgende Aufgaben:¹¹

- Protokollierung von Tickets (aus legalen und administrativen Gründen)
- Festlegung einer Bearbeitungsreihenfolge (und letztendliche Priorisierung)
- Klassifizierung von Tickets
- Bildung eines Archivs für häufige und wiederkehrende Probleme

Durch eine Abbildung und Protokollierung der Arbeitsabläufe zu einem Ticket, kann die Bearbeitungszeit reduziert werden und es entsteht eine transparente Kommunikation zwischen den Benutzern des TTS und den Managern bzw. Bearbeitern der

⁷ Vgl. Gausemeier, J./Plass, C./Wenzelmann, C. (2009), S. 430

⁸ Vgl. Kühne, C./Pauer, D./Rautenstrauch, C.(2003), S. 888

⁹Vgl. Veríssimo, P./Rodrigues, L. (2004), S. 568 f.

¹⁰ Vgl. Kühne, C./Pauer, D./Rautenstrauch, C.(2003), S. 86 f.

¹¹Vgl. Veríssimo, P./Rodrigues, L. (2004), S. 568

Tickets. Modernere Trouble-Ticket-Systeme beinhalten zusätzlich Tools für die Erstellung von Auswertungsstatistiken über erfasste Meldungen. Dadurch ist es möglich Konsequenzen aus Vorfällen zu ziehen und die Auswertungen können eine Basis für Verbesserungen des Service bilden.¹²

Trouble-Ticket-Systeme werden nicht nur für eine effizientere Bearbeitung von Tickets genutzt, sondern ebenso als strukturierte Speicherung der auftretenden Probleme eingesetzt. Anhand der Speicherung können Auswertungen, bspw. bezüglich besonderer Fehlerquellen und durchschnittlicher Bearbeitungszeit der Tickets, vorgenommen und als Ausgangspunkt für Verbesserungsmaßnahmen genutzt werden. Bei einer solchen Auswertung bietet die Klassifizierung von einzelnen Tickets eine große Unterstützung, wenn bestimmte Problembereiche betrachtet werden sollen.

Veríssimo strukturiert den Aufbau von TTS folgendermaßen:¹³

- In- und Output-Subsysteme: Diese dienen als Schnittstelle, bspw. zum Benutzer oder Ticketmanager.
- Filter- und Verarbeitungsmodule: Diese übernehmen die Hauptfunktionen des TTS.
- Regelbasis und Diagnosemodule: Diese automatisieren einige TTS-Funktionen, bspw. das Erstellen von Maßnahmen nach einem bestimmten Auslöser.
- Die in Abbildung 1 mit einer gestrichelten Linie gruppierten Komponenten stellen die Mindestanforderungen an ein TTS und somit die TTS-Basis dar.

¹² Vgl. Gausemeier, J./Plass, C./Wenzelmann, C. (2009), S. 430

¹³ Vgl. Veríssimo, P./Rodrigues, L. (2004), S. 568

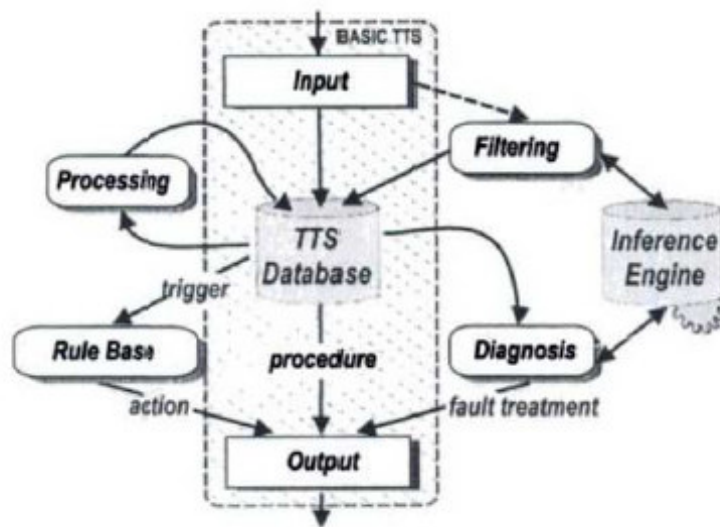


Abbildung 1: Struktur eines TTS nach Veríssimo, P./Rodrigues, L. (2004) S. 568

6.2 Ist-Analyse

Für die Erstellung einer aktuellen Systemdokumentation ist eine Analyse des zu betrachtenden Systems notwendig.

Eine Systemanalyse kann (beruhend auf der Definition von Göhner) folgendermaßen beschrieben werden:¹⁴

Den Ausgangspunkt bildet eine Anforderungsdefinition. In der Systemanalyse wird diese Anforderungsdefinition präzisiert, das System wird von seiner Umgebung abgegrenzt und es erfolgt eine detaillierte Beschreibung der Systemfunktionalität.

Die Systemanalyse befindet sich im Softwareentwicklungsprozess nach der Anforderungsdefinition und vor dem Softwareentwurf und der Implementierung.

Eine Systemanalyse stellt die Grundlage für die Entwicklung eines funktionierenden Anwendungssystems dar. In ihr werden der vorgegebene komplexe Aufgabenzusammenhang und die finanzielle, personelle sowie terminliche Vorgaben, unter dem das System erstellt werden soll, betrachtet.

¹⁴ Göhner, P. (2011), S. 217 ff.

Die diese Arbeit betreffende Aufgabe befasst sich lediglich mit einem Bruchteil der umfassenden Systemanalyse, der auch als Ist-Analyse bezeichnet werden kann.

Folgende Begriffserklärung der Ist-Analyse beruht auf den Erläuterungen im Gabler Wirtschaftslexikon:¹⁵

Die Ist-Analyse bildet einen Teil der Systemanalyse und kann aufgefasst werden als aktueller Zustand, der erhoben, aufbereitet und kritisch analysiert wird. Der aktuelle Zustand bezieht sich hierbei auf einen Problembereich, für welchen ein computergestütztes Informationssystem erstellt oder (falls bereits ein System existiert) verändert werden soll. Während dieser Arbeit wurde Ist-Analyse eines zu verändernden Systems erhoben.

Durch Ist-Analysen wird der Informationsbedarf eines Systems festgestellt und eine Anforderungsdefinition erstellt.

Zur Durchführung einer Ist-Analyse bedarf es einer klaren Abgrenzung des Systems, in der festgelegt wird, welche Sachverhalte zum System zählen und welche nicht im zu analysierenden Bereich enthalten sind. Zu einer Ist-Analyse zählen zudem eine Systemerhebung und eine Beschreibung des betrachteten Systems. Desweiteren bildet eine Fakten- und Schwachstellenanalyse, bspw. zur Aufstellung vorhandener und fehlender Funktionen, den letzten Bestandteil einer vollständigen Ist-Analyse.

Mehr im Detail betrachtet, deckt eine Ist-Analyse nach Hering folgende Bereiche der augenblicklichen Situation eines Systems ab (wobei immer eine Prüfung der Zuverlässigkeit der Informationen durchzuführen ist):¹⁶

- Aufgaben
- Tätigkeiten
- Abläufe
- Informationen und Kommunikation
- Mengengerüst und Datenvolumen
- Sachmittel (z. B. Maschinen, Geräte, Räume)

¹⁵ Müller-Stewens, G./Lackes, R./Siepermann, M. (o. J.)

¹⁶ Vgl. Hering, E./ Gutekunst, J./Dyllong, U. (2000), S. 20

- Arbeitsplätze (z. B. Arbeiten, Leistungen, Qualifikationen)
- Kosten

Eine Ist-Analyse kann allgemeiner bezeichnet werden als „systematische Untersuchung eines Systems hinsichtlich aller Komponenten und der Beziehungen zwischen diesen Komponenten“.¹⁷ Sie enthält Informationen zu der Aufgabe und den Hauptfunktionen eines Systems, sowie eine Abgrenzung der zum System gehörigen Komponenten. Zudem beschreibt eine Ist-Analyse, wie das betrachtete System zum aktuellen Zeitpunkt arbeitet und wie es für die Zukunft angelegt sein soll bzw., welchen Anforderungen es zukünftig genügen soll.¹⁸

Für eine Ist-Analyse ist demnach eine Analyse der Systemfunktionalitäten und der zugrundeliegenden Datenbank notwendig.

6.3 Datenbankide

Eine Datenbank bzw. ein Datenbanksystem setzt sich zusammen aus zwei Teilen. Der eine Teil ist die sogenannte Datenbasis. Sie enthält die Informationen der Datenbank. Der zweite Teil ist das Datenbank-Verwaltungssystem (kurz DBMS), welches für das Management der Dateien der Datenbasis verantwortlich ist (bspw. Oracle Database von Oracle Corporation). Durch das DBMS wird sichergestellt, dass Daten in der richtigen Form gespeichert oder auch verändert werden. Außerdem ist es für eine Zugriffsverwaltung verschiedener Benutzer zuständig.

Datenbanken besitzen unterschiedliche Strukturen und werden unterteilt in fünf Modellen: hierarchische, netzförmige, verteilte, objektorientierte und relationale Datenbanken.¹⁹

¹⁷ Böhm, R./Fuchs, E. (2002), S. 167

¹⁸ Vgl. Böhm, R./Fuchs, E. (2002), S. 167

¹⁹ Vgl. Hering, E./ Gutekunst, J./Dyllong, U. (2000), S. 170

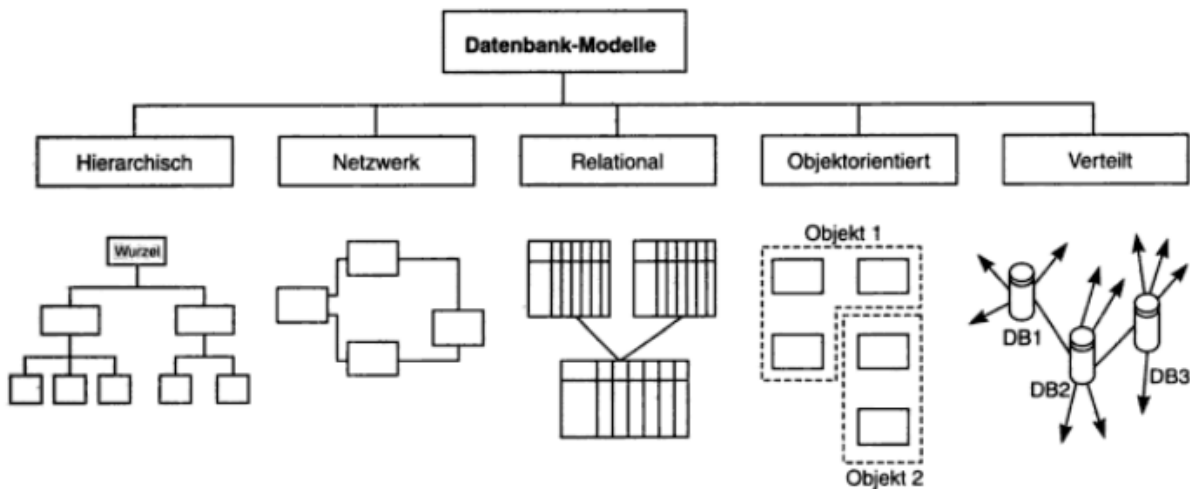


Abbildung 2: Übersicht über die Datenbank-Modelle nach Hering, E./ Gutekunst, J./Dyllong, U. (2000), S. 183

Der Unterschied zwischen diesen verschiedenen Ausprägungen liegt einerseits in der Darstellungsform der Daten und andererseits in den Beziehungen der Daten zueinander.²⁰

Das am frühesten eingesetzte Datenbankmodell ist das hierarchische, welches viele Redundanzen besitzt. In ihm existieren baumartig verknüpfte und hierarchisch gegliederte Knoten. Der Zugriff auf einen bestimmten Datensatz erfolgt abwärts über diese Knoten.²¹

Das nach dem hierarchischen Datenbankmodell entwickelte und Modell ist das netzförmige, welches weniger Redundanzen erhält. Die Verbindung der Daten in diesem Modell kann von mehreren Daten zu mehreren andern Daten erfolgen (m:m-verknüpft). Durch diese Struktur, muss für einen direkten Zugriff auf einen Datensatz der Pfad bekannt sein. Zudem neigt dieses Datenbankmodell dazu, unübersichtlich und schwer zu verwalten zu sein.²²

²⁰ Vgl. Ebenda, S. 183

²¹ Vgl. Fischer, P./Hofer, P. (2011), S. 208

²² Vgl. Ebenda, S. 209

Das verteilte Datenbankmodell besteht aus einzelnen, nach einem zentralen und logischen Modell entworfenen Datenbanken, die durch ein Netz miteinander verbunden sind. Meist werden alle Datenbanken von einem DBMS verwaltet.²³

Ein jüngeres Datenbankmodell ist das objektorientierte, welches nur langsam in der Praxis angenommen und umgesetzt wird. Dieses Modell findet vor allem bei umfassenden Datenbanken in der Client/Server-Umgebung Anwendung.²⁴ Datensätze werden als Objekte mit allen Objektinformationen gespeichert, wodurch es zu Redundanzen kommt. Jedes Objekt der Datenbank enthält Informationen (Attribute), Verweise (Beziehungen) auf andere Objekte und Operationen (Methoden), die das Verhalten der Objekte widerspiegeln.

Die relationalen Datenbankmodelle sind die derzeit populärsten. Ihre Struktur beruht auf zweidimensionalen Tabellen (Relationen). In diesen werden Datensätze (in diesem Zusammenhang als Tupel bezeichnet) als Tabellenzeile angezeigt. Im Tabellenkopf stehen die Attribute (Eigenschaften) eines jeden Tupels und eine Tabellenspalte wird auch als Domäne (Eigenschaftengruppe) bezeichnet.

Ein Tupel wird durch einen sogenannten Schlüssel eindeutig identifiziert. Ein Schlüssel wird definiert als „...minimale Menge von Attributen, deren Werte die

Tupel innerhalb der Relation eindeutig identifizieren“.²⁵ Kennlichgemacht wird ein Schlüssel in Modellen meistens, indem das jeweilige Attribut unterstrichen wird. Im größten Teil der Fälle, werden Attribute eigens für solch eine Schlüsselfunktion eingebaut. Gibt es mehr als ein mögliches Attribut, ein Tupel eindeutig zu bestimmen, wird einer dieser Kandidaten als Primärschlüssel gewählt.²⁶

Neben diesen Primärschlüsseln gibt es in Tabellen gegebenenfalls auch Attribute, die Fremdschlüssel darstellen. Fremdschlüssel werden genutzt, um Tupel aus andern

²³ Vgl. Ebenda, S. 210 f.

²⁴ Vgl. Ebenda, S. 209

²⁵ Kemper, A./Eickler, A. (2006), S. 74

²⁶ Vgl. Kemper, A./Eickler, A. (2006), S. 37

Tabellen zuzuordnen und dadurch komplexere Beziehungen zwischen den Tabellen zu modellieren.²⁷

Zugriffe auf Daten in einer relationalen Datenbank erfolgen durch Beschreibungen mit Operatoren, die Tupelmengen zurückliefern. Diese relationalen Operatoren stellen Verhältnisse zwischen unterschiedlichen Tabellen und deren Attributen her.²⁸

Relationale Datenbanken können redundanzfrei durch mehrere Normalisierungsschritte erstellt werden, dadurch wird die Datenbank kleiner und konsistenter gehalten.

Um festzustellen, wie die vorliegende Datenbank aufgebaut ist, muss definiert sein, wie Daten abgespeichert werden sollen. Eine Festlegung bzw. Aufstellung dieser Struktur der Datenobjekte wird Datenbankschema genannt. Dieses Schema trifft keine Aussagen über die individuellen Datensätze. Ein Datenbankschema kann als Sammlung von Metadaten (d. h. Daten über Daten) aufgefasst werden und da aus diesem Schema die Datenbank nicht implementiert werden soll, ist ein konzeptuelles (auch konzeptionelles) Datenmodell ausreichend.²⁹

Zur Darstellung einer Datenbank wird häufig ein ER-Modell verwendet. Wahlweise wird bei der Modellierung auch auf UML zurückgegriffen, da diese besonders objektorientierte Notation in einigen Fällen Vorteile bringt.

Grundsätzliche Unterschiede zwischen diesen populären Möglichkeiten, sind:

- a) die sehr gut strukturierte Präsentation (vor allem von den Tabellenattributen) im ER-Modell.
- b) die zusätzlich zugeordneten Operationen und Sichtbarkeiten in der UML.
- c) das Wegfallen eines Schlüssels in der UML, da immer eine eindeutige Objekt-ID existiert.³⁰

²⁷ Vgl. Ebenda, S. 73

²⁸ Vgl. Ebenda, S. 210

²⁹ Vgl. Ebenda, S. 22

³⁰ Vgl. Ebenda, S. 57 f.

Mit unterschiedlichen Sichtbarkeiten (auch Sichten oder Views) werden bestimmte Daten bestimmten Benutzern zugänglich oder auch unzugänglich gemacht.

Kemper beschreibt Sichten wie folgt:³¹

Views können als „virtuelle Relationen“ aufgefasst werden, welche nur einen definierten Teil des gesamten Modells zeigen. In diesem Zusammenhang sei virtuell in dem Sinne auszulegen, dass keine neuen Tabellen erstellt, sondern bei jeder Verwendung neu berechnet werden.

Zudem beschreibt er die Aufgabe von Sichten in drei Punkten:

- 1) Die Einsicht bestimmter Daten ist nur für bestimmte Benutzer möglich.
- 2) Sichten verdichten (aggregieren) Daten und anonymisieren sie somit.
- 3) Anfragen können mit Sichten vereinfacht werden, wenn sie „als eine Art Makro“ benutzt werden.

Eine Erleichterung der Datenverwaltung in Datenbanksystemen bieten Trigger. Sie bestehen aus einer benutzerdefinierten Prozedur, die vom DBMS automatisch gestartet wird, nachdem ein bestimmtes Ereignis eingetreten ist.³² Mit Triggern können beispielsweise Überprüfungen, Berechnungen und Änderungen von Datensätzen oder Attributen bewirkt werden.

Um bestimmte Anfragen oder Manipulationen an einer Datenbank vorzunehmen, wird die Anfragesprache SQL (für alle relationalen Datenbanken) verwendet. Bei dieser Sprache wird angegeben, welche Daten betrachtet und ausgegeben werden sollen.³³

Für diesen Zweck gibt es ausführende und unterstützende Software, beispielsweise der SQL Developer von Oracle. Dies ist eine kostenlose IDE, welche die Entwicklung und das Management von Oracle Database vereinfacht.

Zur Darstellung eines Datenbankschemas ist mit Microsoft Office Visio ein unterstützendes Programm zur konzeptionellen Datenmodellerstellung geboten. Die in Visio vorgegebene Notation ist IDEF1X. Diese Datenmodellierungsmethode ist

³¹ Vgl. Kemper, A./Eickler, A. (2006), S. 132 ff.

³² Vgl. Kemper, A./Eickler, A. (2006), S. 163

³³ Vgl. Kemper, A./Eickler, A. (2006), S. 107

besonders geeignet für relationale DB, da sie entitätenbasiert ist und wird zum ER-Modell gezählt.

6.4 Migration

Die Aufgabe beinhaltet eine Teilmigration an dem Meldesystem vorzunehmen.

Mit Migration wird, allgemein beschrieben, ein Prozess durchgeführt, der Anwendungen und/oder Daten aus einer vorhandenen Systemumgebung in eine neue umsetzt.³⁴

Bei diesem Prozess finden essentielle Erneuerungen der bestehenden Systemlandschaft oder einer erheblichen Komponente derselben statt. Eine Migration kann für Hard- sowie Softwaresysteme durchgeführt werden. Da sich die Aufgabenstellung einzig auf die Migration einer Anwendung bezieht, werden im Folgenden die Begriffe Migration und Softwaremigration synonym verwendet.³⁵

Oftmals ist bei einer Systemumstellung, wie sie bei der Migration vorgenommen wird, auch von einer Portierung anstatt von einer Migration die Rede.

Eine Differenzierung nimmt Ihringer in seinem Artikel „Migrationsmythen“ vor:

“Während der eine Ausdruck (Latein: *migrare*, „wandern“) betont, dass für den Plattformwechsel viele Schritte gegangen werden müssen, stellt der andere (Latein: *portare*, „tragen“) eher auf einen möglichst automatischen Ablauf ab.“³⁶

Oftmals werden beide aber auch synonym gebraucht, wenn es um den Sachverhalt geht, dass ein bestimmtes System umgesetzt werden soll.

Für eine automatisierte Übertragung sind besondere Programme entwickelt worden, die mindestens teilweise eine Migration durchführen.

Ist es jedoch nötig, die Programmiersprache des vorhandenen Systems zu wechseln, so der Anwendung spezielle Übersetzungstechniken, um diese Umsetzung möglich zu machen. Translatoren besitzen diese Übersetzungsfähigkeiten. Sie sind Tools, mit

³⁴ Vgl. Fischer, P./Hofer, P. (2011), S. 568

³⁵ Vgl. Die Beauftragte der Bundesregierung für Informationstechnik (2012), S. 6

³⁶ Ihringer, M. (2010)

welchen „...automatisiert Quellprogramme A in Zielprogramme B [konvertiert werden können]“.³⁷ Die praktische Eignung einer solchen Vorgehensweise wurde laut Erdmenger von mehreren unterschiedlichen Migrationsprojekten erfolgreich nachgewiesen.

Der Verlauf einer Migration kann in zwei unterschiedliche Ausprägungen erfolgen: Zum einen gibt es das Vorgehen einer schlagartigen (oder auch „Big-Bang“ genannten) Migration, zum anderen existiert das Modell einer etappenweisen (auch sanft genannten) Migration.

Die Beauftragte der Bundesregierung für Informationstechnik betitelt diese Methoden als Stichtagsumstellung bzw. schrittweise Migration und definiert diese Begriffe wie folgt:³⁸

Die Stichtagsumstellung erfolgt idealerweise an einem einzigen Tag oder zumindest sind der Beginn und das Ende der Umstellung auf das migrierte System mit keinem großen Abstand zueinander terminiert. Das Ziel einer Stichtagsumstellung liegt in einem übergangslosen Wechsel des (Teil-)Systems und es findet kein gleichzeitiger Betrieb von Alt- und Neusystem statt. Gegenüber einer klaren Umstellungsgrenze steht das erhöhte Risiko, dass bei der Umstellung Probleme auftreten, die Funktionalitäten erheblich beeinträchtigen.

Im Gegensatz zu diesem Vorgehen steht die schrittweise Migration eines Systems.

Sie verfolgt ein Prinzip der Aufteilung von komplexeren Zusammenhängen in einzelne, leichter zu überschauende Aufgaben bzw. Komponenten. Diese werden dann nacheinander migriert und in das bestehende System eingefügt.

Durch diese Herangehensweise wird das Risiko des Gesamtvorhabens, welches vor allem bei umfangreichen Systemen hoch ist, auf ein überschaubares Maß (nämlich auf das der einzelnen Komponente) reduziert. Dieser Ablauf erfordert einen längeren Umstellungszeitraum, welcher für manche ein Nachteil darstellt. Ist das Vorhaben sehr

³⁷ Erdmenger, U. u. a. (2008), S. 2

³⁸ Die Beauftragte der Bundesregierung für Informationstechnik (2012), S. 8

komplex, werden zusätzlich zur Aufteilung in kleine Aufgaben auch Migrationsphasen eingeteilt.

Die Migration des Meldesystems entspricht eher einer Stichtagsumstellung, da zuerst das gesamte System vollständig migriert werden soll.

Bei einer Migration mit Wechsel der Programmiersprache können weitere Optimierungen vorgenommen werden.

Bezüglich Quellcodestrukturierung bieten Architekturmuster eine gute Möglichkeit, spätere Änderung oder Erweiterung zu dem System problemlos hinzu zufügen. Der VB6-Quellcode ist unterteilt in drei Bereiche: Forms, Modules und Class Modules. Dies entspricht nicht einer effizienten MVC-Einteilung, welche für interaktive Systeme oftmals für eine klare Einordnung der unterschiedlichen Quellcodeteile genutzt wird.

Nach Aufgabenstellung soll die Zieltechnologie der Migration .NET von Microsoft bilden. Diese zählt zu den modernsten Technologien. Insbesondere ist die für diese Technologie entwickelte IDE Visual Studio sehr gut geeignet, um in kurzer Zeit verschiedenste Anwendungen zu implementieren. Dies können Webanwendungen, Desktopanwendungen oder auch Anbindungen an fremde Datenquellen sein. Die Implementierung ist zudem in vielen unterschiedlichen Programmiersprachen möglich und Plattform unabhängig.

Ein umfassender Bestandteil des .NET Frameworks, der für die Anbindung und den Umgang mit Datenbanken genutzt werden kann, wird als ADO.NET bezeichnet. Beispielsweise können mit der von ADO.NET bereitgestellten DataSet-Klasse komplette

Datenbanken dargestellt werden oder mit der bereitgestellten DataTable-Klasse Tabellen erstellt und manipuliert werden.

Die Programmiersprache C#, in die das Zielprogramm migriert werden soll ist eine „...systemeigene Sprache für die .NET Common Language Runtime“³⁹ und bietet daher die am besten geeigneten Implementierungsmöglichkeiten im gegensatz zu anderen

³⁹ Gunnerson, E. (2000)

möglichen Programmiersprachen wie beispielsweise C++. C# ist für robuste, langlebige Komponenten konzipiert worden, mit denen reale Lebenssituationen bewältigt werden können.⁴⁰

Mit den Komponenten, die Visual Studio zur Verfügung stellt, kann das Resultat einer Datenbankabfrage strukturiert dargestellt werden. Eine besondere Unterstützung bei der Datenpräsentation an den Anwendungsbenutzer, kann durch von Developer Express Inc. bereitgestellten Komponenten erhalten werden, die einen ansprechendes Erscheinungsbild und breitgefächerte Konfigurationsmöglichkeiten bieten. Diese Vielfalt der Einstellungsmöglichkeiten wird von den default-Komponenten der IDE nicht geliefert, wodurch sich DevExpress-Komponente besonders für hohe Individualisierungsansprüche eignet.

⁴⁰ Gunnerson, E. (2000)

7 Umsetzung

7.1 Vorhandene Dokumentation

Um eine aktuelle Dokumentation zu erstellen, wurde die bereits vorhandene Dokumentation betrachtet. Darauf aufbauend wurde versucht, durch entsprechende Anpassungen eine dem derzeitigen Stand entsprechende Variante zu erstellen. Bei der Bearbeitung dieser bereits existierenden Dokumentation wurde offensichtlich, dass ein sehr großer Anpassungsbedarf bestand. Dieser hatte den gleichen Aufwand wie eine Neuüberarbeitung, sodass eine komplette Neuerstellung der Systemdokumentation sinnvoller war und durchgeführt wurde.

Von der ehemaligen Dokumentation wurden nur die inhaltliche Gliederung und Bruchteile allgemeiner Beschreibungen mit in die neu erstellte Systemdokumentation übernommen. Die inhaltliche Gliederung entspricht einer projektinternen Vorlage, die mit diesem Vorgehen eingehalten wurde. Durch diese Konformität haben Projektmitglieder, die schon länger im Projekt sind oder sich öfter mit Dokumentationen aus diesem Projekt beschäftigt haben, ein bekanntes, übersichtliches Schema, in dem eine schnelle und effektive Orientierung möglich ist.

7.2 Datenbankschema

Das Datenbankschema wurde im Rahmen der Ist-Analyse erstellt, welche seinerseits die Basis für die zu erstellende Systemdokumentation bildet. Um das zugrundeliegende Datenbankschema zu verstehen und danach abbilden zu können, mussten einerseits Datenbankbegrifflichkeiten verstanden und angewandt und andererseits eine geeignete Darstellungsform für das Datenbankschema gewählt werden.

Mit Hilfe des SQL Developers wurde auf der vorliegenden, relationalen Oracle Datenbank untersucht, welche Relationen mit welchen Attributen die Meldesystemdatenbank beinhaltet. Bei der vorliegenden Datenbank gilt die Konvention, dass Attribute als Präfix die Relationsnummer in ihren Namen enthalten.

Ebenso wurden existierende Trigger und ihre Funktionen recherchiert und die vorhandenen Views analysiert. Zudem wurde aufgrund des analysierten Datenbankschemas nachvollzogen, wie Meldungen und Meldungsveränderungen in den unterschiedlichen Relationen und durch die verschiedenen Trigger abgebildet werden. Trigger dienen im Meldesystem beispielsweise für die Statistik Aktualisierung oder sie bezwecken eine automatische Eintragung des Änderungsdatums in verschiedenen Tabellen.

Ein hier auftretende Problem war, dass auf der Testdatenbank nur teilweise die aktuell verwendeten Trigger gefunden werden konnten, da sie durch eine wöchentliche Synchronisation der Daten nicht nötig waren und nur auf der alltäglich genutzten und aktiven Datenbank umgesetzt wurden. Somit enthält das erstellte Datenbankschema einige Trigger-Lücken, da die Testdatenbank diese Trigger nicht enthält.

Die bei der Analyse des Datenbankschemas erworbenen Erkenntnisse wurden in Microsoft Office Visio 2003 unter Zuhilfenahme der in Visio verfügbaren Datenbankmodellierungsvorlage zusammengefasst und grafisch aufbereitet dargestellt. Somit erfolgte die Visualisierung des Datenbankschemas in der IDEF1X-Notation, die ideal für die hier vorliegende relationale Datenbanken ist. Die einzelnen Relationen der Datenbank wurden für eine bessere Übersicht in unterschiedliche Gruppen eingeteilt, bspw. in eine Views-, Haupttabellen- und Statistikgruppe.

Ein weiteres Problem bei der Datenbankschemaerstellung war der durch das Alter der Anwendung entstandene Umstand, dass die Meldesystembenutzern (und teils auch die Administratoren) keinen Überblick mehr über die aktuell genutzten und aktuell ungenutzten Relationen der Testdatenbank hatten.

Mit zu dieser Schwierigkeit beigetragen hat, dass auf der Testdatenbank Testrelationen angelegt wurden, die bspw. für eine angefangene und wieder abgebrochene Erstellung einer Schnittstelle zu einem in der Abteilung verwendeten Quellcodeverwaltungssoftware genutzt werden sollten. Diese angelegten Testtabellen wurden in manchen Fällen nach Abbruch des Projektes nicht wieder gelöscht und es war ohne eine vorherige Analyse unklar, welchem Zweck diese Tabellen dienten.

Oftmals waren zeitnahe Änderungsdaten in den Relationen ein Hinweis für eine aktive und genutzte Tabelle, jedoch gab es nicht in allen Relationen ein solches Attribut.

Dies ist zum Beispiel bei einer Relation über alle möglichen Statureigenschaften einer im Meldesystem verfassten Fehlermeldung der Fall ist.

Die fertiggestellte Übersicht der genutzten Tabellen wurde der Systemdokumentation als erläuterndes Datenbankschema beigelegt und kann ebenfalls im Anhang A dieser Arbeit betrachtet werden.

7.3 Meldesystemfunktionalitäten

Teils gleichzeitig mit der Datenbankschemaanalyse, teils nachträglich wurden die vom Fehlermeldesystem angebotenen Funktionalitäten untersucht. Diese bilden eine weitere Komponente der Ist-Analyse für die zu erstellende Systemdokumentation und wurden seit dem Stand der letzten Systemdokumentation um einige Anforderungen erweitert.

Das Ermitteln der Meldesystemfunktionalitäten erfolgte durch Erstellen und Verändern einiger beispielhafter Meldungen. Teilweise basierten die Vorgehensweisen dabei auf den in der veralteten Systemdokumentation geschilderten Abläufen.

Das hier betrachtete Trouble-Ticket-System ist nicht so weit entwickelt, dass es Tickets automatisiert verarbeitet. Eine adäquate Rückmeldung wird ebenfalls nicht vorbereitet. Es existieren Schnittstellen zu anderen Systemen (z. B. Outlook), jedoch nicht direkt zum jeweiligen TIS, welches dann automatisiert Fehlermeldungen an das Meldesystem schicken könnte.

Beim Analysieren der Funktionalitäten musste den fünf verschiedenen Benutzerrollen, in die das Meldesystem die registrierten Benutzer einteilt, besondere Beachtung gewidmet werden. Grund dafür war, dass bestimmte Meldungsänderungen nur von einer bestimmten Benutzergruppe getätigt werden können. So verhält es sich beispielsweise mit dem Meldungsbearbeitungsschritt „in Built integrieren“, da diese Funktionalität nur der Benutzergruppe „Buildmaster“ zur Verfügung steht.

Die unterschiedlichen Berechtigungen jeder Benutzergruppe des Fehlermeldesystems gekoppelt mit den verschiedenen Status verkomplizieren den Ablauf einer Fehlermeldung, wenn er im Detail und mit vielen Sonderfällen betrachtet wird, und eine verbale Beschreibung wirkt dadurch sehr verwirrend und undurchsichtig. Aufgrund

dessen wurde entschieden, eine visuelle Aufbereitung des Meldungsablaufes zu erstellen, wodurch eine bessere Überschaubarkeit gewährleistet war.

Auf der Grundlage der Meldesystemfunktionalitäten und –prozesse wurde in Microsoft Office Visio 2003 eine Meldungslebenslaufübersicht erstellt, mit welcher der Ablauf der verschiedenen Meldungsstatus und die Rolle der einzelnen Benutzergruppen überschaubar dargestellt wurde.

Dieses Schema wurde der aktuellen Systemdokumentation als Übersicht des Meldungslebenslaufes beigefügt und kann ebenfalls im Anhang B dieser Arbeit betrachtet werden.

7.4 Grafische Oberfläche

Einen ersten Schritt für die Migration sollte die Umsetzung der Oberfläche bilden, wobei darauf geachtet wurde, Optimierungsmöglichkeiten wahrzunehmen und eine größere Konsistenz zu erlangen.

Um Verbesserungsmöglichkeiten zu erkennen, erfolgte eine genaue Analyse der derzeitigen Oberfläche und eine Rücksprache mit den Benutzern des Fehlermeldesystems, um Änderungswünsche und –anregungen von den Anwendern zu erhalten.

Nach diesen Untersuchungen wurde die grafische Oberfläche aufgeteilt in verschiedene Kategorien und Gruppen, um Redundanzen ersichtlich zu machen und bei der erneuten Umsetzung zu umgehen. Die gesamte Oberfläche sollte am Ende einfacher und übersichtlicher gestaltet sein.

In diesem Schritt wurde beispielsweise darauf geachtet, dass zukünftig Änderungen an einer bestimmten Meldungsinformation immer an genau einer festen Stelle des Meldesystems erfolgt und es nicht wie derzeit praktikierbar ist, auf unterschiedlichen Registerkarten der Meldungsverwaltung eine Meldungsinformation zu bearbeiten. Dies war beispielsweise der Fall bei dem Meldungsattribut „Meldungsversion“.

Zusätzlich wurden nach den Untersuchungen diverse Neuordnungen von Meldungsinformationen bzw. ihren Kategorien vorgenommen um danach eine effizientere Meldungserfassung und –einsicht zu ermöglichen. (siehe Abbildungen unten, grüne Markierung).

Um eine schnelle Übersicht des derzeitigen Meldungsstatus und mögliche, folgende Aktionen, die dem jeweiligen Benutzer zur Verfügung stehen, bereit zu stellen, wurde das im Punkt 8.3 angesprochene Meldungslebenslaufschemata mit in die Benutzeroberfläche integriert (Abbildung 4, orange Markierung). Diese Entscheidung hatte als Nebeneffekt, dass die grafische Oberfläche insgesamt durch dieses Schema ansprechender gestaltet war.

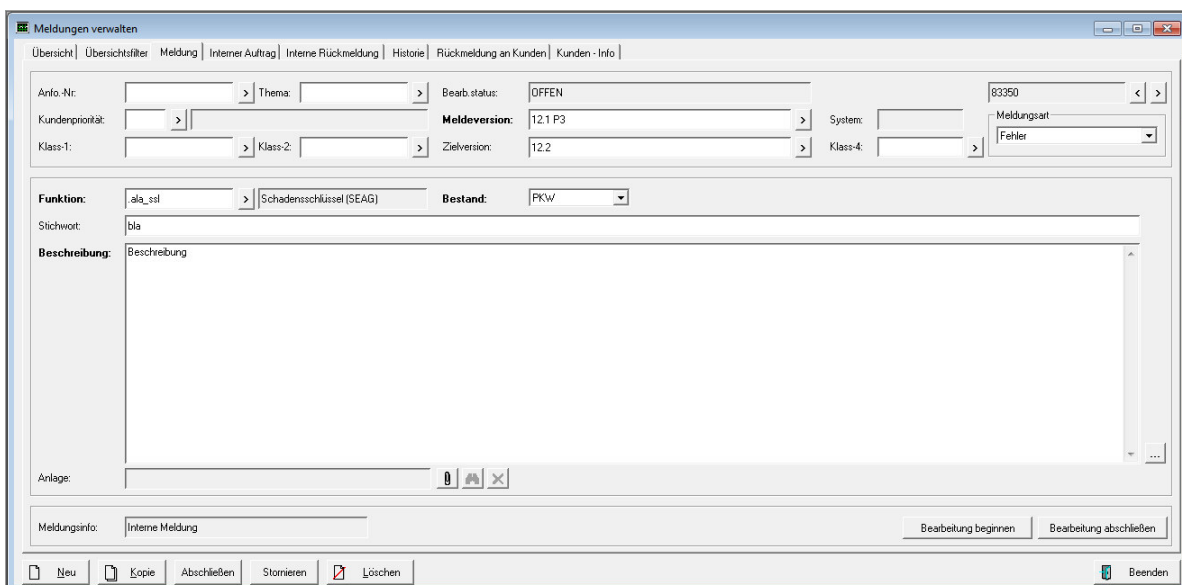


Abbildung 3: Meldungsdetails - alte Benutzeroberfläche

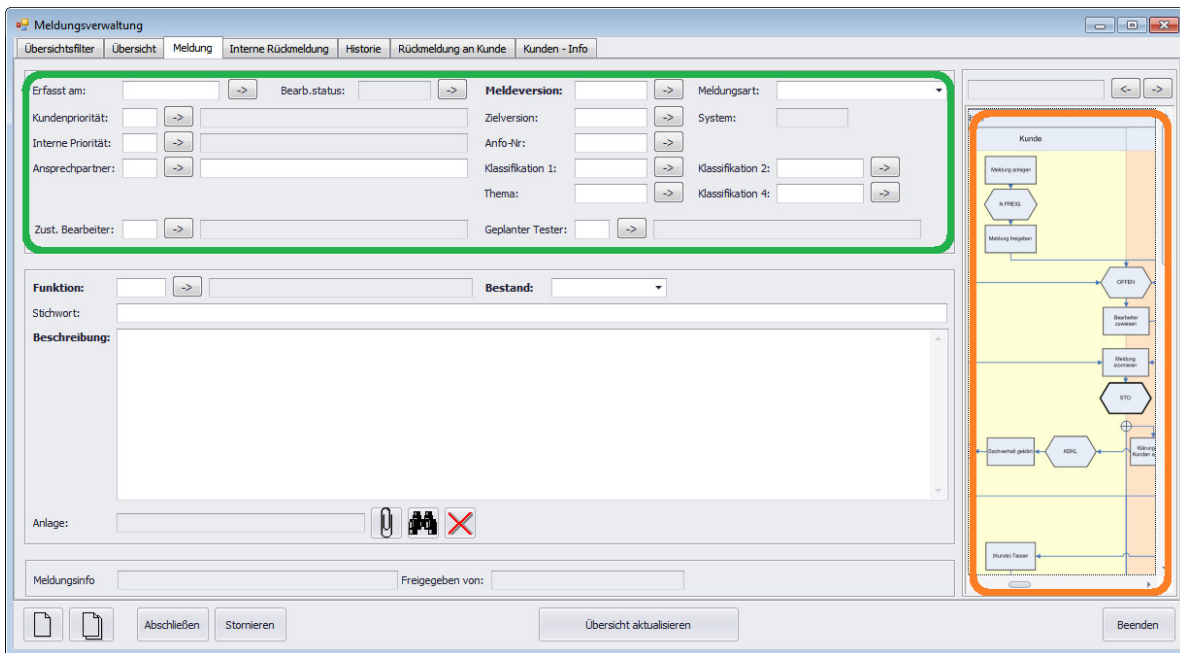


Abbildung 4: Meldungsdetails - neue Benutzeroberfläche

7.5 VB6-Quellcode

Zusätzlich zur Umsetzung der Oberfläche ist für eine vollständige Migration auch die Umsetzung der Systemfunktionalität nötig, damit das gesamte System nach vollendeter Migration immer noch seine Aufgaben erfüllt.

Für eine funktionale, korrekte Umsetzung wurde der vorhandene VB6-Quellcode als Ausgangspunkt genommen. Er sollte als Vorlage für eine mögliche Logik- und Funktionsimplementierung in C# dienen.

Dieser Vorgehensschritt führte jedoch nicht zum gewünschten Ergebnis, da es unmöglich war die Logik des bloßen Quellcodes mit allgemeinen Programmiersprachenkenntnissen zu verstehen. Es war angedacht beim vorliegenden Quellcode durch Debuggen Einsicht in die verwendete Logik zu erhalten. Dies gestaltete sich als nicht möglich, da einige der genutzten und relativ alten Controls (sogenannte DebisControls) bestimmte Software benötigten, um in der Entwicklungsumgebung korrekt angezeigt und ausgeführt zu werden.

Da projektintern unbekannt war, welche Software genau benötigt wurde und wo diese gelagert war, wurde beschlossen, den VB6-Quellcode nicht als Ausgangspunkt für eine funktionale Umsetzung zu benutzen.

7.6 Migration von Systemteilen

Auf Grundlage der erstellten Analyse wurde anschließend versucht, Teile der Funktionalität umzusetzen und zu migrieren. Dabei wurde Fokus auf die Implementierung der Filterfunktionalität der Meldungsverwaltung gelegt.

Ein Translator zur automatisierten oder auch teilautomatisierten Migration konnte nicht genutzt werden, da die bereits erwähnten DebisControls eine korrekte Arbeitsweise des Translators verhindern.

Der VB6-Quellcode ist nicht in MVC strukturiert, im Migrationsverlauf sollte der migrierte Code jedoch danach aufgebaut sein. Projektintern wird im Allgemeinen eine Gliederung in Business Logic, Data Model und GUI vorgenommen, die dem MVC entspricht.

Die Realisierung erfolgte in der Entwicklungsumgebung Visual Studio 2010 in Kombination mit von DevExpress (Trial-Version 11.2.) bereitgestellten Komponenten.

Bei der Umsetzung der Datenanbindung und Filterfunktionalität, wurden viele Implementierungsschritte durch ADO.NET vereinfacht und das Erstellen von SQL-Abfragen durch den Visual Studio „Konfigurations-Assistent für TableAdapter-Abfragen“ erleichtert. Der Versuch, komplexere Abfragen zu generieren, gelang nicht mit Hilfe des Konfigurations-Assistenten, der die Anweisungen nicht interpretieren und somit nicht in den Quellcode einfügen konnte. Bei einer manuellen Änderung der Suchanfrage im Quellcode wurde diese vom Compiler wieder auf den vorher vom Konfigurations-Assistenten definierten Wert zurückgesetzt.

Zum Ende dieser Arbeit war es möglich, sich Meldungen nach einzelnen Suchkriterien gefiltert anzeigen zu lassen.

8 Zusammenfassung und Ausblick

Die für diese Arbeit zu erfüllende Aufgabe bestand im ersten Teil aus der Ist-Analyse und einer daraus zu erstellenden Systemdokumentation des im eingesetzten Projekt verwendeten Trouble-Ticket-Systems. Der zweite Teil bestand daraus eine auf die im ersten Teil gewonnenen Erkenntnisse aufbauende Teilmigration durchzuführen, wobei der Fokus auf die grafische Oberfläche und Meldungsfilterungsfunktionalitäten gesetzt wurden.

Im Laufe der Durchführung wurde sich in die zugrundeliegende Datenbank eingearbeitet und es fand eine Auseinandersetzung mit den Fehlermeldungsprozessen sowie verschiedenen Funktionalitäten des aktuellen Meldesystems statt. Schließlich erfolgte eine vertiefte Analyse der Benutzeroberfläche.

Die Ergebnisse sind grafisch in einem Datenbankschema und einen Meldungslebenslaufschemata festgehalten worden. Zusätzlich wurde eine Systemdokumentation angefertigt und eine strukturiertere Oberfläche mit einfacher Filterfunktionalität implementiert.

Die erstellte Systemdokumentation soll zukünftig neuen Mitarbeitern die Arbeit mit dem Fehlermeldesystem erleichtern.. Außerdem soll sie als akkurate Darstellung des derzeitigen Meldesystems herangezogen werden können, wenn Fragen zur aktuellen Arbeitsweise des Meldesystems zu klären sind. Nicht zuletzt soll die Systemdokumentation auch eine Grundlage für zukünftige Änderungen des Fehlermeldesystems sein. Falls Änderungen vorgenommen werden, ist vorgesehen, dass diese in der Systemdokumentation vermerkt und eingearbeitet werden, damit die Dokumentation immer auf dem neuesten Stand ist.

Die teilmigrierte Benutzeroberfläche soll Ausgangspunkt für eine später durchgeführte vollständige Migration bilden, die auf jeden Fall durchgeführt werden muss auf Grund der veralteten Technologie und des unübersichtlich gewordenen Quellcodes. Möglicherweise kann die weitere Vorgehensweise eine Aufgabenstellung für eine andere Praxisarbeit bilden.

9 Literaturverzeichnis

Böhm, R./Fuchs, E. (2002) System-Entwicklung in der Wirtschaftsinformatik, Zürich
2002

Die Beauftragte der Bundesregierung für Informationstechnik (2012)

Migrationsleitfaden, Leitfaden für Migration von Software, 4. Version,
http://www.cio.bund.de/SharedDocs/Publikationen/DE/Architekturen-und-Standards/migrationsleitfaden_4_0_download.pdf?__blob=publicationFile, Berlin
2012, Einsichtnahme: 10.08.2012

Erdmenger, U. u. a. (2008) Methoden und Werkzeuge für die Software Migration,
http://www.proetcon.de/de/download/pec_article_Ini08.pdf, 2008, Einsichtnahme: 10.08.2012

Fischer, P./Hofer, P. (2011) Lexikon der Informatik, 15. Auflage, Berlin/Heidelberg
2011

Gausemeier, J./Plass, C./Wenzelmann, C. (2009)

Zukunftsorientierte Unternehmensgestaltung, Strategien, Geschäftsprozesse
und IT-Systeme für die Produktion von morgen,
München/Wien 2009

Göhner, P. (2011)

§5 Systemanalyse in: Softwaretechnik I (Vorlesungsunterlagen),
http://www.ias.uni-stuttgart.de/lehre/vorlesungen/st1/vorlesung/umdruck/st1_ws11_kapitel_05.pdf, 2011,
Einsichtnahme: 17.08.2012

Gunnerson, E. (2000) C#, Die neue Sprache für Microsofts .NET-Plattform,
<http://openbook.galileocomputing.de/csharp/intro.htm#t21>, 2000 Einsichtnahme: 10.08.2012

Hoffmann, D. W. (2008): Software-Qualität, Berlin/Heidelberg 2008

Hering, E./ Gutekunst, J./Dyllong, U. (2000)

Handbuch der praktischen und technischen Informatik, 2., Neubearb. und
erw. Aufl., Berlin/Heidelberg/New
York/Barcelona/Hongkong/London/Mailand/Paris/Singa
pur/Tokio 2000

Ihringer, M. (2010) Migrationsmythen, [http://entwickler-
magazin.de/zonen/magazine/psecom,id_8,online,2879,p,0.html](http://entwickler-magazin.de/zonen/magazine/psecom,id_8,online,2879,p,0.html), 2010, Einsichtnahmen:
10.08.2012

Kemper, A./Eickler, A. (2006) Datenbanksysteme, Eine Einführung, 6. Auflage,
Oldenburg 2006

Kühne, C./Pauer, D./Rautenstrauch, C.(2003)

Erstellung eines Helpdesk-Systems zur Bearbeitung von Anliegen
Studierender in: Wirtschaftsinformatik 2003, (Hrsg.:
Uhr, W.),Band 1, Heidelberg 2003, S. 881 – 935

Microsoft Corporation (o. J.): Erklärung zum Support von Visual Basic 6.0 unter
Windows® Vista™ und Windows® Server 2008,
<http://msdn.microsoft.com/de-de/vbrun/ms788708.aspx>,
o. J., Einsichtnahme: 13.03.2012

Müller-Stewens, G./Lackes, R./Siepermann, M. (o. J.)

Istanalyse in: Gabler Wirtschaftslexikon,
<http://wirtschaftslexikon.gabler.de/Archiv/56489/istanalyse-v7.html>, o. J., Einsichtnahme: 10.04.2012

Veríssimo, P./Rodrigues, L. (2004)

Distributed Systems for System Architects, o. O. 2004

10 Glossar

.NET	Software-Plattform bestehend aus der Common Language Runtime und einer Sammlung von Klassenbibliotheken, Programmierschnittstellen und Services
.NET Common Language Runtime	Laufzeitumgebung; hier werden die .NET-Programme ausgeführt
ADO.NET	Bezeichnung für .NET-Framework Klassen zum Umgang mit relationalen Datenbanken
Architekturmuster	dient zur grundlegende Organisation und Interaktion zwischen Komponenten einer Anwendung
DebisControls	eigenentwickelte Steuerelemente mit speziellen Funktionalitäten.
ER-Modell	Darstellung von Tabellen und ihre Beziehungen
Fahrzeugobjekt	erprobare Komponente eines Fahrzeugs, z. B. Motor
Helpdesk-System	Anwendung zum Verwalten von Kundenfragen
Laufzettel	Zettel an Werkstücken o. Ä., auf dem jeder Arbeitsgang eingetragen wird
MVC	Architekturmuster für interaktive Systeme
Oracle Database	DBMS von Oracle
Redundanz	unnötige Dopplung einer Information
Translator	genutzt für eine Migration in eine andere Programmiersprache
UML	standardisierte, grafische Modellierungssprache

