Abschlussprüfung Winter 2014

Fachinformatikerin für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

Übertragungswertverarbeitung

Desktopanwendung zur Verarbeitung von Übertragungswerten

Abgabetermin: 03.12.2014

Prüfungsbewerberin:
Gerda Feldhaus
Astrup 77
49429 Visbek

Ausbildungsbetrieb:
Alte Oldenburger Krankenversicherung AG
Theodor-Heuss-Straße 96
49377 Vechta

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AO	Alte Oldenburger Krankenversicherung AG
API	Application Programming Interface
BaFin	Bundesanstalt für Finanzdienstleistungsaufsicht
BREPL	Beschlagworten Routen Erkennen Prüfen Leisten (Eigenentwicklung)
CSV	Comma Separated Value
EPK	Ereignisgesteuerte Prozesskette
ERM	Entity-Relationship-Model
ESB	Enterprise Service Bus
FiBu	Finanzbuchhaltung
GUI	Graphical User Interface
IS	Integration Server (Enterprise Service Bus der Software AG)
MVC	Model View Controller
ORM	Object-Relational Mapping
PAM	Professional Archive Manager (Dokumentenmanagementsystem)
Versis	Versicherungsinformationssystem (Bestandsführungssystem)
WDO	Wechseldatenobjekt
WPF	Windows Presentation Foundation (Grafik-Framework)
XAML	Extensible Application Markup Language

1 Einleitung

Die folgende Projektdokumentation schildert den Ablauf des IHK-Abschlussprojektes, welches die Autorin im Rahmen ihrer Ausbildung zur Fachinformatikerin Fachrichtung Anwendungsentwicklung durchgeführt hat. Ausbildungsbetrieb ist die Alte Oldenburger Krankenversicherung AG (AO), eine private Krankenversicherung mit Standort in Vechta. Zur Zeit beschäftigt die AO 228 Mitarbeiter.[1] Zu den Produkten des Unternehmens zählen nicht nur private Krankenvoll- und Pflegeversicherungen, sondern auch Zusatzversicherungen für privat und gesetzlich Versicherte.

1.1 Projektbeschreibung

Die AO erhält sogenannte Übertragungswerte, wenn eine versicherte Person von einem anderen privaten Versicherungsunternehmen zur AO wechselt. Bei umgekehrtem Wechsel zahlt die AO diese Werte an das neue Versicherungsunternehmen aus.

Bei den Übertragungswerten handelt es sich um einen Teil der angesparten Alterungsrückstellungen der versicherten Person. Dies sind bestimmte Beitragsanteile, die in jungen Jahren für den Versicherten auf einem separaten Konto angelegt werden und im Alter zur Beitragssenkung dienen sollen.

Die Übertragungswerteingänge und -ausgänge müssen vom Rechnungswesen als Nachweis in der Finanzbuchhaltung (FiBu) verbucht werden. Die dafür benötigten Daten werden momentan manuell von Sachbearbeitern der Antragsabteilung aus den entsprechenden Kontoauszügen und dem Bestandsführungssystem Versis ermittelt und in einer Excel-Tabelle erfasst und verwaltet. Außerdem muss der Sachbearbeiter einen Statuswechsel bei einem sogenannten Wechseldatenobjekt (WDO) vornehmen und sein Vorgehen im Dokumentenmanagementsystem PAM-Storage[2] dokumentieren.

In diesem Projekt soll der gerade beschriebene Prozess automatisiert werden.

1.2 Projektziel

Ziel des Projektes ist die Automatisierung der Übertragungswertverarbeitung und -verwaltung. Dazu soll die Excel-Tabelle durch eine eigenständige Desktopanwendung mit graphischer Benutzeroberfläche und eigener Datenbank abgelöst werden. Die Anwendung soll das Auslesen der Daten aus dem Kontoauszug automatisieren und zusätzlich eine Validierung der ermittelten Werte vornehmen. Außerdem soll das Setzen des WDO-Status und das Starten, Bearbeiten und Beenden von PAM-Vorgängen zur Dokumentation automatisch angestoßen werden. Alle ermittelten Daten sollen in der Datenbank historisiert abgelegt werden. Ferner wird eine Möglichkeit zum Erstellen, Exportieren und Drucken von Auswertungen benötigt. Durch die Automatisierung soll ein weitestgehend elektronischer Verarbeitungsfluss gewährleistet werden.

[1] Kennzahl zum Stichtag 20.11.2014, vgl. ALTE OLDENBURGER [2013, S. 4].
[2] Vgl. http://www.hs-soft.com/de/dokumentenmanagement.aspx.

1.3 Projektumfeld

Auftraggeber des Projektes sind die Fachabteilung Rechnungswesen und spezielle Sachbearbeiter der Antragsabteilung der AO.

Zu den Aufgaben des Rechnungswesens gehören die Beitragsverbuchung, die Darstellung der Zahlungsein- und -ausgänge sowie der Bilanz, Gewinn- und Verlustkonten, die Vorbereitungen für den Jahresabschluss und das Versenden von Nachweisen (Meldewesen), z.B. an die Bundesanstalt für Finanzdienstleistungsaufsicht (BaFin). Auch das Erfassen von Kapitalanlagenbewegungen in der FiBu zählt zu dessen Tätigkeiten. Aus diesem Grund ist das Rechnungswesen auch für die Verbuchung der Übertragungswerte in der FiBu zuständig.

Die Mitarbeiter der Antragsabteilung prüfen Neuverträge und sind für die Aufnahme dieser Verträge in Versis zuständig. Aus diesem Grund haben sie ein fundiertes Wissen in Bezug auf den Umgang mit diesem System und können somit vergleichsweise schnell die für die Übertragungswertverarbeitung benötigten Daten ermitteln.

Um den Anforderungen beider Abteilungen gerecht zu werden, ist eine intensive und regelmäßige Kommunikation und Rücksprache mit diesen beiden Parteien zwingend erforderlich.

1.4 Projektbegründung

Die Hauptschwachstelle des momentanen Übertragungswertverarbeitungsprozesses ist das hohe Maß an manueller Arbeit, durch die der elektronische Datenverarbeitungsfluss ständig unterbrochen wird. Einige Beispiele hierfür wären das Auslesen des Kontoauszugs, das Füllen der Excel-Datei mit den ermittelten Daten, das Setzen des WDO-Status und das Erstellen von Buchungshilfen auf Grundlage der in der Tabelle erfassten Werte. Hierbei kann es schnell zu Flüchtigkeitsfehlern kommen, die dann wiederum zu Folgefehlern führen. Außerdem könnte es sogar passieren, dass einzelne Prozessabschnitte komplett vergessen werden.

Ein weiteres Problem ist, dass die Excel-Tabelle, in der die Daten, die für die Verarbeitung relevant sind, erfasst werden, durch die stetig wachsende Größe ihre Übersichtlichkeit verliert. Zudem ist sie nicht vollständig, da die Sachbearbeiter der Antragsabteilung in ihr nur die Übertragungswerteinund -ausgänge dokumentieren. Andere Kontoumsätze, wie z.B. die Kontoführungsgebühren, werden in diese Tabelle nicht mit aufgenommen. Dies hat den Nachteil, dass das Rechnungswesen diese Umsätze noch einmal speziell beachten muss. Ein weiteres Problem ergibt sich, wenn beide Abteilungen beziehungsweise mehrere Personen gleichzeitig auf die Excel-Datei zugreifen wollen, da sie sich dann gegenseitig blockieren.

Aufgrund dieser Probleme und der dadurch steigenden Fehleranfälligkeit hat sich die AO dazu entschieden, die Entwicklung einer Desktopanwendung in Auftrag zu geben, durch die der gesamte Prozess automatisiert werden soll.

1.5 Projektschnittstellen

Damit die Daten, die für den Übertragungswertverarbeitungsprozess relevant sind, vollständig ermittelt und validiert werden können und der Status des WDOs automatisch gesetzt werden kann, muss die Desktopanwendung mit Versis interagieren können. Diese Anforderung soll mit Hilfe von Webservices, welche ein Enterprise Service Bus (ESB) zur Verfügung stellt, umgesetzt werden. Bei dem ESB handelt es sich um ein Softwareprodukt, das sich um die Integration der verschiedenen Schnittstellen kümmert und die dafür benötigten Services zur Verfügung stellt. Bei der AO wird hierfür der sogenannte Integration Server (IS)[3], ein Produkt der Software AG, eingesetzt.

Um die sogenannten PAM-Vorgänge automatisch zu starten oder zu beenden, ist außerdem eine Interaktion mit dem PAM-Application Programming Interface (API) erforderlich. Das Erstellen und Routen von Vorgängen erfolgt bei der AO nicht direkt über den Aufruf der PAM-API, sondern wird über eine intern entwickelte Software namens BREPL gesteuert. Diese erhält die verschiedenen Eingangskanäle in Form von Fabriken[4]. Aus diesem Grund muss eine Übertragungswert-Fabrik für BREPL implementiert werden, um automatisch Vorgänge für die Dokumentation der Übertragungswertverarbeitung zu erzeugen.

1.6 Projektabgrenzung

Da der Projektumfang beschränkt ist, soll das Bereitstellen der Webservices über den ESB nicht Bestandteil des Abschlussprojektes sein.

2 Projektplanung

2.1 Projektphasen

Für die Umsetzung des Projektes standen der Autorin 70 Stunden zur Verfügung. Diese wurden vor Projektbeginn auf verschiedene Phasen verteilt, die während der Softwareentwicklung durchlaufen werden. Eine grobe Zeitplanung sowie die Hauptphasen lassen sich der Tabelle 1: Grobe Zeitplanung entnehmen. Außerdem können die einzelnen Hauptphasen noch in kleinere Unterpunkte zerlegt werden. Eine detaillierte Übersicht dieser Phasen befindet sich im Anhang A.1: Detaillierte Zeitplanung auf S. i.

Projektphase	Geplante Zeit
Analysephase	8 h
Entwurfsphase	11 h
Implementierungsphase	40 h
Abnahme und Einführung	1 h
Erstellen der Dokumentation	10 h
Gesamt	**70 h**

Tabelle 1: Grobe Zeitplanung

[3] Vgl. http://www.softwareag.com/de/products/az/webmethods/default.asp
[4] Fabrik bezieht sich in diesem Fall auf das Design-Pattern *Fabrik*.

2.2 Ressourcenplanung

In der Übersicht, welche sich im Anhang A.2: Verwendete Ressourcen auf S. ii befindet, sind alle Ressourcen aufgelistet, die für das Projekt eingesetzt wurden. Damit sind sowohl Hard- und Softwareressourcen als auch das Personal gemeint. Bei der Auswahl der verwendeten Software wurde darauf geachtet, dass diese kostenfrei (z.B. als Open Source) zur Verfügung steht oder die AO bereits Lizenzen für diese besitzt. Dadurch sollen anfallende Projektkosten möglichst gering gehalten werden.

2.3 Entwicklungsprozess

Bevor mit der Realisierung des Projektes begonnen werden konnte, musste sich die Autorin für einen geeigneten Entwicklungsprozess entscheiden. Dieser definiert die Vorgehensweise, nach der die Umsetzung erfolgen soll. Für das Abschlussprojekt wurde von der Autorin ein agiler Entwicklungsprozess gewählt. Hierbei soll in Anlehnung an das Vorgehensmodell Scrum gearbeitet werden. Das iterative Durchlaufen der Projektphasen sowie stetige Rücksprachen mit den Stakeholdern sind einige der Merkmale, die den agilen Entwicklungsprozess auszeichnen.[5]

Durch die relativ kurzen Iterationszyklen wird eine flexible Umsetzung der Anforderungen ermöglicht, sodass dem Fachbereich relativ zeitnah Resultate präsentiert werden können. Aufgrund dieser Tatsache wurde bei der Projektplanung in Abschnitt 2.1 (Projektphasen) für die Entwurfsphase vergleichsweise wenig Zeit eingeplant, da sich Teile dieser Phase erst im Laufe der Entwicklung der Software ergeben.[6]

Die stetige Kommunikation mit den Fachbereichen und das daraus erhaltene Feedback gewährleistet zudem eine bessere Reaktionsfähigkeit auf nachträgliche Änderungswünsche und fördert das Erzielen besserer Resultate. Nebenbei kann der Fachbereich somit bereits mit dem zu entwickelnden System vertraut gemacht werden. Dies hat wiederum den Vorteil, dass bei der Projektabnahme und -einführung Zeit gespart werden kann. Daher wurde für diese Phase in der Projektplanung ebenfalls relativ wenig Zeit einkalkuliert.[7]

Außerdem soll der agile Entwicklungsprozess durch die Praktik der testgetriebenen Entwicklung erweitert werden. Hierbei werden die Softwaretests bereits vor der Implementierung des eigentlichen Quellcodes erstellt. Da man sich bei diesem Vorgehen bereits im Voraus Gedanken über den Aufbau der einzelnen Module machen muss, haben die Tests für den Entwickler nicht nur eine verifizierende Funktion, sondern können auch als Unterstützung bei der Auswahl der Architektur dienen.[8]

Bei der testgetriebenen Entwicklung werden die folgenden drei Phasen iterativ durchlaufen:[9]

1. **Test schreiben, der fehlschlägt:** Zunächst wird ein Test definiert, der die korrekte Funktionalität einer neuen Funktion verifizieren soll. Dieser Test wird jedoch fehlschlagen, da diese Funktionalität noch nicht implementiert ist.

[5] Vgl. Bleek und Wolf [2008, S. 13f].
[6] Vgl. Pilcher [2007, S. 27].
[7] Vgl. Pilcher [2007, S. 108f].
[8] Vgl. Bleek und Wolf [2008, S. 88].
[9] Vgl. Bleek und Wolf [2008, S. 89].

2. **Produktivcode soweit implementieren, dass der Test erfolgreich durchläuft.**
3. **Refaktorisieren:** Der Quelltext kann nun umstrukturiert und verbessert werden, sodass er übersichtlicher und verständlicher ist. Durch die bereits definierten Tests wird sichergestellt, dass das Programmverhalten nicht versehentlich geändert wird.

In der Projektplanung wurde viel Zeit für die Implementierungsphase einkalkuliert, damit das Prinzip der testgetriebenen Entwicklung und das damit verbundene intensive Testen umgesetzt werden kann.

3 Analysephase

3.1 Ist-Analyse

Wie bereits im Abschnitt 1.1 (Projektbeschreibung) erwähnt wurde, ist das Rechnungswesen für die Verbuchung der Übertragungswertein- und -ausgänge zuständig. Die dafür benötigten Daten werden momentan von der Antragsabteilung aus den entsprechenden Kontoauszügen ausgelesen, vervollständigt und anschließend in Form einer fortlaufenden Excel-Tabelle an das Rechnungswesen übermittelt. Der zuständige Sachbearbeiter erhält den Kontoauszug (PDF) per Mail vom Rechnungswesen. Er muss neben der Datenermittlung auch noch den Abgleich und die Prüfung der eingegangenen Beträge vornehmen, den Status des WDOs in Versis korrekt setzen und einen Vorgang in PAM starten und beenden. Am Ende erstellt das Rechnungswesen aus der Excel-Tabelle Buchungshilfen für den entsprechenden Tag (ebenfalls in Form einer Excel-Tabelle) und führt dann mit Hilfe dieser Übersicht die Buchung in der FiBu durch. Außerdem werden der Kontoauszug und die Buchungshilfe vom Rechnungswesen zusätzlich ausgedruckt und zusammen abgeheftet.

Der gesamte Verarbeitungsprozess wurde zu Beginn des Projektes in Form einer EPK dargestellt, die das hohe Maß an manueller Arbeit noch einmal verdeutlichen soll. Ein Ausschnitt dieser Grafik befindet sich im Anhang A.3: EPK des Ist-Zustandes (Ausschnitt) auf S. iii.

3.2 Wirtschaftlichkeitsanalyse

Aufgrund der Probleme des momentanen Übertragungswertverarbeitungsprozesses, die in den Abschnitten 1.4 (Projektbegründung) und 3.1 (Ist-Analyse) geschildert und erläutert wurden, ist die Umsetzung des Projektes unbedingt erforderlich. Ob die Realisierung aber auch aus wirtschaftlichen Gesichtspunkten gerechtfertigt ist, soll in den folgenden Abschnitten geklärt werden.

3.2.1 „Make or Buy“-Entscheidung

Da es sich bei dem Projekt um eine sehr unternehmensspezifische Anforderung der AO handelt und verschiedenste Komponenten (Versis, PAM) mit der Anwendung agieren müssen, lässt sich auf dem Markt keine Lösung finden, die diese Wünsche erfüllen könnte. Daher soll das Projekt in Eigenentwicklung durchgeführt werden.

3.2.2 Projektkosten

Die Projektkosten, die während der Entwicklung des Projektes anfallen, sollen im Folgenden kalkuliert werden. Dafür müssen neben den Personalkosten, die durch die Realisierung des Projektes verursacht werden, auch noch die Aufwendungen für die Ressourcen (Hard- und Software, Büroarbeitsplatz etc.) berücksichtigt werden. Da die genauen Personalkosten nicht herausgegeben werden dürfen, wird die Kalkulation anhand von Stundensätzen[10] durchgeführt, die von der Personalabteilung festgelegt wurden. Der Stundensatz eines Auszubildenden beträgt demzufolge 10 €, der eines Mitarbeiters 25 €. Für die Ressourcennutzung wurde ein pauschaler Stundensatz von 15 € angenommen.

Die Kosten, die für die einzelnen Vorgänge des Projektes anfallen, sowie die gesamten Projektkosten lassen sich der Tabelle 2: Kostenaufstellung entnehmen.

Vorgang	Mitarbeiter	Zeit	Personal[11]	Ressourcen[12]	Gesamt
Entwicklungskosten	1 x Auszubildender	70 h	700,00 €	1.050,00 €	1.750,00 €
Fachgespräch	2 x Mitarbeiter	5 h	250,00 €	150,00 €	400,00 €
Code-Review	1 x Mitarbeiter	2 h	50,00 €	30,00 €	80,00 €
Abnahme	2 x Mitarbeiter	0,5 h	25,00 €	15,00 €	40,00 €
				Projektkosten gesamt	2.270,00 €

Tabelle 2: Kostenaufstellung

3.2.3 Amortisationsdauer

Im folgenden Abschnitt soll ermittelt werden, ab welchem Zeitpunkt sich die Entwicklung der Anwendung amortisiert hat. Anhand dieses Wertes kann dann beurteilt werden, ob die Umsetzung des Projektes aus wirtschaftlicher Sicht sinnvoll ist und sich auf Dauer Kostenvorteile ergeben. Die Amortisationsdauer wird berechnet, indem man die Anschaffungskosten durch die laufende Kostenersparnis dividiert, die durch das neue Produkt erzielt wird.

Durch die Automatisierung des Übertragungswertprozesses ließe sich Bearbeitungszeit einsparen. Dadurch würden sich die Kosten für das Personal reduzieren. Eine Übersicht über die konkrete Zeiteinsparung der einzelnen Vorgänge wird in der Tabelle 3: Zeiteinsparung dargestellt. Die für die Vorgänge angegebenen Zeiten wurden von den Fachbereichen (bei Verarbeitung der Übertragungswerte nach der alten Vorgehensweise) und von der Autorin (bei Verwendung der neuen Lösung) geschätzt.

Da die Verarbeitung des Kontoauszugs (Auslesen des Auszugs, Vervollständigung und Validierung der Daten mit Hilfe von VERSIS, Setzen des WDO-Status, Erfassung der Ermittelten Daten in einer Tabelle), das Erstellen und Bearbeiten der PAM-Vorgänge sowie das Erstellen von Buchungshilfen und Auswertungen automatisiert erfolgen soll, entfällt für diese Vorgänge die Bearbeitungszeit. Nur wenn Probleme bei der automatischen Verarbeitung eines Kontoauszugs auftreten, ist eine manuelle Nachbearbeitung nötigt. Für diesen Vorfall wurde eine Nachbearbeitungszeit von 2,5 Minuten eingeplant.

[10] Die aufgeführten Stundensätze setzen sich insbesondere aus dem Bruttostundenlohn und den Sozialaufwendungen des Arbeitgebers zusammen.
[11] Personalkosten pro Vorgang = Anzahl Mitarbeiter · Zeit · Stundensatz.
[12] Ressourcenbeitrag pro Vorgang = Anzahl Mitarbeiter · Zeit · 15 € (Ressourcenbeitrag pro Stunde).

Nachfolgend soll nun die Amortisationsdauer berechnet werden. Sie gibt an, ab welchem Zeitpunkt die Anschaffungskosten durch die Zeitersparnis beglichen werden können.

Vorgang	**Anzahl** pro Monat	**Zeit (alt)** pro Vorgang	**Zeit(neu)** pro Vorgang	**Einsparung** pro Monat[13]
Verarbeitung des Kontoauszugs	11	16 min	2,5 min	148,5 min
PAM-Vorgang erstellen und bearbeiten	44	1,5 min	0 min	66 min
Buchungshilfe erstellen	11	3 min	0 min	33 min
Auswertung erstellen	1	5 min	0 min	5 min
			Zeiteinsparung gesamt pro Monat	252,5 min

Tabelle 3: Zeiteinsparung

Berechnung der Amortisationsdauer:

$$252,5\frac{min}{Monat} \times 12\frac{Monate}{Jahr} = 3.030\frac{min}{Jahr} = 50,5\frac{h}{Jahr} \tag{1}$$

$$50,5\frac{h}{Jahr} \times (25\,€ + 15\,€)^{14} = 2.020,00\frac{€}{Jahr} \tag{2}$$

$$\frac{2.270{,}00\,€}{2.020,00\frac{€}{Jahr}} = 1,12 \text{ Jahre} \approx 1 \text{ Jahr } 7 \text{ Wochen} \tag{3}$$

Zusätzlich wurde die Amortisation graphisch dargestellt. Dadurch soll die Berechnung der Amortisationsdauer noch einmal bildlich veranschaulicht werden. In dem Diagramm wurden sowohl die variablen Kosten (pro Jahr) der alten als auch der neuen Lösung abgebildet. Außerdem wurden die Kosten, die für die Entwicklung der neuen Lösungen angefallen sind, miteinbezogen. Die Grafik befindet sich im Anhang A.6: Amortisation auf S. vi.

Anhand der Amortisationsrechnung ergibt sich für das Projekt eine Amortisationsdauer von 1 Jahr und 7 Wochen. Dies ist der Zeitraum, über den die neue Anwendung mindestens eingesetzt werden muss, damit sich Anschaffungskosten und Kosteneinsparung ausgleichen. Da das Unternehmen die neue Anwendung langfristig einsetzen möchte, kann das Projekt auch unter wirtschaftlichen Gesichtspunkten als sinnvoll eingestuft werden.

3.3 Nicht-monetäre Vorteile

Da die Ergebnisse der Wirtschaftlichkeitsanalyse die Realisierung des Projektes bereits ausreichend rechtfertigen, soll an dieser Stelle auf eine detaillierte Analyse nicht-monetärer Vorteile verzichtet werden. Nicht-monetäre Vorteile der neuen Anwendung wären aber z.B. die bessere Übersichtlichkeit, die durch die neue und klarere Strukturierung der Daten gewährleistet wird, die Möglichkeit zur Historisierung von Kontoauszügen und Kontoumsätzen sowie die Aufhebung der gegenseitigen Blockade durch die Ablösung der Excel-Tabelle.

[13] Einsparung pro Monat = Anzahl pro Monat · (Zeit (alt) pro Vorgang - Zeit (neu) pro Vorgang).
[14] Stundensatz für Mitarbeiter und dessen Ressourcennutzung.

3.4 Anwendungsfälle

Um eine grobe Übersicht über die Anwendungsfälle zu erhalten, die von dem umzusetzenden Programm abgedeckt werden sollen, wurde im Laufe der Analysephase ein Use-Case-Diagramm erstellt. Dieses Diagramm befindet sich im Anhang A.5: Use-Case-Diagramm auf S. v und bildet alle Funktionen ab, die aus Endanwendersicht benötigt werden.

3.5 Lastenheft/Fachkonzept

Am Ende der Analysephase wurde zusammen mit den beiden Fachabteilungen Rechnungswesen und Antragsabteilung ein Lastenheft erstellt. Dieses umfasst alle Anforderungen des Auftraggebers an die zu erstellende Anwendung. Ein Auszug aus dem Lastenheft befindet sich im Anhang A.4: Lastenheft (Auszug) auf S. iv.

4 Entwurfsphase

4.1 Zielplattform

Das Abschlussprojekt soll, wie bereits in Abschnitt 1.2 (Projektziel) erwähnt wurde, als eigenständige Desktopanwendung mit eigener Datenbank umgesetzt werden. Das Datenbanksystem Oracle soll hierbei zum Einsatz kommen, da sich dieses seit langem im Unternehmen bewährt hat. Außerdem gibt es genügend Mitarbeiter, die mit der Administration und Wartung dieses Systems vertraut sind, sodass sichergestellt wird, dass es sich immer auf dem aktuellsten Stand befindet.

Die Auswahl der Programmiersprache, mit der das Projekt realisiert werden soll, wurde anhand einer Nutzwertanalyse durchgeführt. Im Anhang A.7: Nutzwertanalyse zur Auswahl der Programmiersprache auf S. vi sind die einzelnen Kriterien mit ihren jeweiligen Gewichtungen und den Bewertungen für die verschiedenen Programmiersprachen aufgelistet. Sowohl für die Gewichtung als auch für die Bewertung werden Werte zwischen 1 und 5 verwendet.[15,16]. Um die Gewichtung in die Nutzwertanalyse einfließen zu lassen, wird diese mit der Bewertung multipliziert. Der Quotient aus der gewichteten Gesamtbewertung und der Summe der Gewichtungen ergibt den Nutzwert der jeweiligen Programmiersprache. Der geringste Wert, der hier erreicht werden könnte, wäre 1, der höchste wäre 5.

Da die Sprachen Natural, Java und C# bereits im Unternehmen etabliert sind und sich in Projekten als geeignet erwiesen haben, wurde die Auswahl auf diese drei Programmiersprachen beschränkt. Die meisten Kriterien, die in der Analyse betrachtet wurden, sind auch bereits in einer Nutzwertanalyse für das Unternehmen allgemein bewertet worden. Die Autorin hat die Gewichtung der einzelnen Kriterien auf die speziellen Anforderungen dieses Projektes abgestimmt, sodass die unternehmensinterne Nutzwertanalyse in gekürzter und angepasster Form wiederverwendet werden kann. Die Punkte „Oracle-Zugriff“, „Webservice konsumieren“, „Plattform: Windows“ und „GUI-Oberfläche“ wurden hierbei am höchsten gewichtet, da diese Kriterien für die Realisierung der Anwendung von großer Bedeutung sind.

[15] Gewichtung: 1 = unnötig, 2 = verzichtbar, 3 = wünschenswert, 4 = erforderlich, 5 = unbedingt erforderlich.
[16] Bewertung: 1 = mangelhaft, 2 = ausreichend, 3 = befriedigend, 4 = gut, 5 = sehr gut.

Aus der Nutzwertanalyse geht hervor, dass die Programmiersprache C# mit einem Nutzwert von 4,47 vor den anderen beiden Sprachen Java (mit 4,13) und Natural (mit 2,45) liegt. Daher ist C# unter Berücksichtigung dieser Gesichtspunkte am besten geeignet und soll für die Umsetzung des Projektes verwendet werden.

4.2 Architekturdesign

Das Projekt soll auf Basis des Model View Controller (MVC)-Architekturmusters umgesetzt werden. Demnach lässt sich jede Komponente einer Software einem der drei Bestandteile – Model, View oder Controller – dieses Musters zuordnen.[17] Jeder dieser drei Teile hat einen speziellen Aufgabenbereich, der von denen der anderen weitestgehend unabhängig ist. Das Model setzt sich hierbei aus den Daten und der entsprechenden Verarbeitungslogik zusammen, der View ist für die Präsentation bzw. Anzeige der Daten zuständig und über den Controller erfolgt die Steuerung der Anwendung. Er stellt das Bindeglied zwischen Model und View dar. Die lose Kopplung der einzelnen Komponenten erhöht die Wiederverwendbarkeit und Austauschbarkeit. Man könnte beispielsweise die Benutzeroberfläche austauschen, ohne das Model anpassen zu müssen. Außerdem können die einzelnen Komponenten durch die strikte Trennung einfacher getestet, gewartet und flexibel erweitert werden.[18] Aufgrund dieser Vorteile soll dieses Architekturmuster für die Realisierung des Projektes verwendet werden.

Zur Implementierung und Gestaltung der Benutzeroberfläche soll das Windows Presentation Foundation (WPF)-Framework genutzt werden. Dieses arbeitet mit zwei Komponenten, dem Application Markup und dem sogenannten Code-Behind. Mit Hilfe des Markups werden alle visuellen Aspekte wie beispielsweise die Größe oder Positionierung der Elemente, aus denen sich die Oberfläche zusammensetzt, definiert. Es liegt im sogenannten Extensible Application Markup Language (XAML)-Format vor. Die Logik, die sich hinter den einzelnen Elementen verbirgt, wird in einem dem Markup zugeordneten Codeabschnitt definiert und daher als Code-Behind bezeichnet. Sie legt fest, wie sich die Anwendung verhalten soll, wenn eines der Elemente durch Benutzerinteraktion ausgewählt bzw. verändert wird.[19] Möchte man die Komponenten des WPF-Frameworks nun noch denen des MVC-Musters zuordnen, so erfüllt das Markup die Rolle des Views und der Code-Behind stellt den Controller da. View und Controller des MVC-Musters werden somit in der Graphical User Interface (GUI)-Komponente der Anwendung vereint.

Um der Autorin eine einfache und weitestgehend automatische Interaktion mit der relationalen Oracle-Datenbank zu ermöglichen, soll das Entity-Framework verwendet werden. Dabei handelt es sich um ein Object-Relational Mapping (ORM)-Framework von Microsoft. Dieses Tool stellt Funktionalitäten bereit, mit denen sich die Objekte der Anwendung auf die Relationen der Datenbank abbilden lassen. Somit muss sich die Autorin nicht mehr um diese Konvertierung kümmern und kann sich stattdessen stärker auf die eigentliche Verarbeitungslogik der Anwendung konzentrieren.[20]

[17] Vgl. Freeman u. a. [2004, S. 529f].
[18] Vgl. Eilebrecht und Starke [2007, S. 67].
[19] Vgl. Microsoft Corporation [2014].
[20] Vgl. EntityFrameworkTutorial.net [2014].

4.3 Entwurf der Benutzeroberfläche

Um die neue Anwendung möglichst benutzerfreundlich bedienen zu können, soll eine klar strukturierte, einfache Benutzeroberfläche entwickelt werden. Mit Hilfe von Mockups wurde hierfür zunächst ein Prototyp der Oberfläche angefertigt. Dafür wurde die Software Balsamiq Mockups[21] verwendet. Damit die Benutzeroberfläche am Ende den Anforderungen und Vorstellungen des Fachbereichs entspricht, wurde dieser bei der Entwurfsphase intensiv mit einbezogen. In der Hauptansicht sollen auf Wunsch des Fachbereichs vier Reiter zu sehen sein: Verarbeitung ausstehend, Verarbeitung erfolgreich, Verarbeitung fehlerhaft und Suche. Innerhalb eines Reiters sollen die Daten mit Hilfe eines Accordion[22]-Controls strukturiert dargestellt und der jeweiligen Kategorie Übertragungswerteingang, -ausgang oder Sonstiges zugeteilt werden. Der Inhalt der Accordion-Elemente soll in Form einer Tabellenstruktur abgebildet werden. Mockups der Hauptansicht sowie des Importfensters für den Kontoauszugimport befinden sich im Anhang A.8: Oberflächenentwürfe auf S. vii.

4.4 Datenmodell

Um einen groben Überblick über alle für die Übertragungswertverarbeitung relevanten Daten zu erhalten, wurde zunächst schrittweise eine Analyse des Verarbeitungsablaufs durchgeführt. Dabei wurde hinterfragt, welche Daten gespeichert werden müssen und in welcher Beziehung sie zueinander stehen. Im Anhang A.9: Analyse des Prozessablaufs und Ermittlung der notwendigen Entitätstypen auf S. viii wird diese Analyse anhand eines Beispiels näher erläutert.

Auf Grundlage der herausgearbeiteten Entitätstypen wurde ein Entity-Relationship-Model (ERM) erstellt, welches die Typen und ihre Beziehungen graphisch veranschaulichen soll. Dieses befindet sich im Anhang A.10: Entity-Relationship-Model auf S. ix. Folgende Entitätstypen wurden herausgearbeitet und abgebildet: `Kontoauszug`, `Kontoumsatz`, `Erweiterter_Kontoumsatz`, `Korrekter_Kontoumsatz`, `Unternehmen`, `Produktkomponente`, `Status`, `Buchungstext` und `Mandant`.

Da für die Anwendung eine relationale Datenbank verwendet werden soll, wurde das konzeptionelle ERM in ein Tabellenmodell überführt. Das entsprechende Diagramm befindet sich im Anhang A.11: Tabellenmodell auf S. x.

4.5 Geschäftslogik

Durch die Vorgehensweise der testgetriebenen Entwicklung ergeben sich die zu implementierenden Klassen erst während der Implementierungsphase. Daher ist die Erstellung eines Klassendiagramms im Voraus noch nicht möglich. Die Komponenten, die für die Anwendung relevant sind, sind jedoch bereits gegeben. Daher wurde unter anderem ein Komponentendiagramm angefertigt, um die Geschäftslogik zu modellieren. Dieses Diagramm befindet sich im Anhang A.12: Komponentendiagramm auf S. xi. Es veranschaulicht die Abhängigkeiten der einzelnen Komponenten untereinander. Das `Domänenmodell` legt die Struktur der anwendungsinternen Daten fest. Mit Ausnahme der Datenbanksysteme und des Dokumentenmanagementsystems PAM wird diese von allen anderen Komponenten

[21] Vgl. http://balsamiq.com/products/mockups/.
[22] Vgl. https://www.nuget.org/packages/Accordion/.

der Anwendung verwendet. Die Schnittstellen zu den Datenbanksystemen werden von den Komponenten `Datenabgleich` und `Datenhaltung` genutzt, um einen Datenabgleich mit den Bestandsdaten aus Versis durchführen zu können bzw. das Persistieren der Daten in der eigenen Datenbank zu ermöglichen. Über die Komponente `Datenimport` können neue Daten in die Anwendung importiert werden. Die Schnittstelle `Verarbeitungsdokumentation` stellt das Bindeglied zwischen Anwendung und Dokumentenmanagementsystem PAM dar.

Möchte man die einzelnen Komponenten den in Abschnitt 4.2 (Architekturdesign) erwähnten Rollen des MVC-Musters zuordnen, so würden `Domaenenmodell`, `Datenabgleich`, `Datenhaltung`, `Datenimport` und `Verarbeitungsdokumentation` das Model abbilden. Die GUI-Komponente ist, wie bereits in Abschnitt 4.2 (Architekturdesign) beschrieben, für die visuelle Darstellung und die Reaktion auf Benutzerinteraktionen zuständig und vereint somit Controller und View des MVC-Musters. Die Komponente `Datenexport` soll das Exportieren von Auswertungen und Buchungshilfen im PDF-Format oder Excel-Format ermöglichen. Somit stellt sie eine Art Datenrepräsentation dar und lässt sich daher ebenfalls der View-Rolle des MVC-Musters zuordnen.

Zusätzlich zum Komponentendiagramm wurde eine Entscheidungstabelle erstellt. Diese soll die Fälle, die bei der Übertragungswerteingangsverarbeitung auftreten können, veranschaulichen. Sie befindet sich im Anhang A.13: Entscheidungstabelle auf S. xi und beschreibt grob die Bedingungen und Handlungsmaßnahmen, die beim Verarbeitungsprozess eintreffen können bzw. nötig sind.

4.6 Pflichtenheft

Am Ende der Entwurfsphase wurde ein Pflichtenheft erstellt. Dieses baut auf dem Lastenheft (vgl. dazu Abschnitt 3.5 (Lastenheft/Fachkonzept)) auf und beschreibt wie und womit die Autorin die Anforderungen des Fachbereiches umsetzen möchte. Es dient somit als Leitfaden für die Realisierung des Projektes. Ein Auszug aus dem Pflichtenheft befindet sich im Anhang A.14: Pflichtenheft (Auszug) auf S. xii.

5 Implementierungsphase

5.1 Iterationsplanung

Bevor mit der eigentlichen Implementierungsphase begonnen wurde, wurde zunächst ein Iterationsplan erstellt. In ihm wurden die einzelnen Iterationsschritte und deren Reihenfolge definiert. Innerhalb einer Iteration wurde jeweils eine bestimmte Funktionalität implementiert, die dann am Ende der entsprechenden Iteration dem Fachbereich präsentiert wurde. Diese Vorgehensweise entspricht dem in Abschnitt 2.3 (Entwicklungsprozess) beschriebenen Prinzip der agilen Softwareentwicklung. Der erstellte Iterationsplan befindet sich im Anhang A.15: Iterationsplan auf S. xiii. Die Autorin hat sich bei dessen Erstellung an den Komponenten des Komponentendiagramms orientiert.

5.2 Implementierung der Datenstrukturen

Auf Basis der Datenstruktur, die bereits im Abschnitt 4.4 (Datenmodell) für die zu erstellende Anwendung definiert worden ist, wurde die Implementierung der Datenbank durchgeführt. Zunächst wurde dafür vom Oracle-Datenbank-Administrator ein neues Schema mit dem Namen `UEWV` (Übertragungswertverwaltung) zu einer bereits bestehenden Oracle-Instanz hinzugefügt. Anschließend wurden die benötigten Tabellen manuell mit Hilfe von SQL-Statements von der Autorin angelegt. Dafür wurde das Tool Oracle SQL Developer benutzt. Als Grundlage für die Struktur diente das in der Entwurfsphase erstellte Tabellenmodell, welches im Anhang A.11: Tabellenmodell auf S. x abgebildet ist. Die erstellte Datenbank entspricht somit der in der Entwurfsphase definierten Struktur und erfüllt die nötigen Anforderungen.

5.3 Implementierung der Geschäftslogik

Da die Implementierung der Geschäftslogik den Kernbestandteil des gesamten Projektes darstellt, soll diese im folgenden Abschnitt genauer erläutert werden. Um die Implementierung möglichst komfortabel durchführen zu können, ist eine geeignete Entwicklungsumgebung von großer Bedeutung. Die Autorin hat dafür Visual Studio Professional 2013 verwendet.

Zu Beginn der Implementierung wurde das Grundgerüst der Anwendung erstellt. Dafür wurde zunächst eine neue Solution angelegt. Die Projektstruktur innerhalb der Solution wurde auf Basis der im Komponentendiagramm (siehe Anhang A.12: Komponentendiagramm auf S. xi) definierten Komponenten erstellt. Jede Komponente wurde durch ein separates Projekt abgebildet. Die Referenzen, die unter den einzelnen Projekten bestehen müssen, wurden ebenfalls aus dem Komponentendiagramm abgeleitet. Da die Implementierung testgetrieben erfolgen soll, wurde zusätzlich zu jedem Projekt ein entsprechendes Testprojekt angelegt, in dem die jeweilige Komponente getestet wird. Diese separaten Projekte haben den Vorteil, dass der Produktivcode von den Tests getrennt abgelegt werden kann. Nach der Fertigstellung des Grundgerüstes konnte mit der eigentlichen Implementierung begonnen werden. Ein Screenshot der Solution befindet sich im Anhang A.16: Screenshot des Grundgerüstes der Solution auf S. xiii.

Im ersten Iterationsschritt der Implementierungsphase hat die Autorin sich mit der Fertigstellung der Datenbank befasst. Um sich einen groben Überblick zu verschaffen, wie die Datenbankkomponente sinnvoll in die Architektur integriert werden kann, wurde von der Autorin zunächst ein konzeptionelles Klassendiagramm erstellt. Dieses befindet sich im Anhang A.17: Konzeptionelles Klassendiagramm auf S. xiv.

Im Laufe der Konzeptionsphase wurden folgende Klassen herausgearbeitet: `EntityFrameworkDatenbank` und `MockDatenbank`. Beide Klassen sollen ein Interface `IDatenbank` implementieren. Somit soll die Testbarkeit der Datenbankinteraktion gewährleistet werden. In den Tests sollen die Frameworks NUnit[23], welches zum Durchführen von Unit-Tests dient, und Moq[24], mit dem Pseudoklassen erstellen werden können, verwendet werden. In der echten GUI-Komponente arbeitet die Klasse

[23] Vgl. http://www.nunit.org/.
[24] Vgl. https://github.com/Moq/moq4.

`KontoauszugController` mit dem Interface, welches in diesem Fall durch die Klasse `EntityFrameworkDatenbank` implementiert wird. Da gegen das Interface implementiert wird, kann im Test die `MockDatenbank`[25] verwendet werden. Somit löst man sich von dem Zugriff auf die echte Datenbank.

Soll getestet werden, ob auch wirklich Daten in die Datenbank geschrieben wurden, so kann im Test auch die Klasse `EntityFrameworkDatenbank` verwendet werden. Diese greift immer auf die Datenbank zu, die im Connection-String in der `App.config` des jeweiligen Projektes referenziert wurde. Dadurch ist es möglich, in der echten GUI-Komponente auf die produktive Oracle-Datenbank zuzugreifen und im Test auf eine Testdatenbank, z.B. SQLite. Auf Basis des erstellten konzeptionellen Klassendiagramms wurde die Implementierung der Datenhaltungskomponente testgetrieben durchgeführt.

Die gesamte Implementierung der Geschäftslogik erfolgte nach dem bereits in Abschnitt 2.3 (Entwicklungsprozess) beschriebenen Zyklus. Als Beispiel soll an dieser Stelle die Implementierung des CSV-Imports herangezogen werden. Zunächst wurde hierbei immer ein Testfall geschrieben. Ein Auszug aus der Testklasse `CsvKontoauszugsquelleSollte` befindet sich im Anhang A.18: Listing der Tests für die Klasse `CsvKontoauszugsquelle` (Ausschnitte) auf S. xiv. Darauf folgte die Implementierung der getesteten Funktionalität. War diese erfolgreich abgeschlossen, wurde der nächste Testfall definiert. Nach dieser Vorgehensweise wurde die gesamte Logik der Klasse `CsvKontoauszugsquelle` Schritt für Schritt entwickelt. Ausschnitte aus dem Quellcode der Klasse `CsvKontoauszugsquelle` befinden sich im Anhang A.19: Listing der Klasse `CsvKontoauszugsquelle` auf S. xvii.

5.4 Implementierung der Benutzeroberfläche

Die Implementierung der Benutzeroberfläche wurde, wie bereits im Abschnitt 4.2 (Architekturdesign) geplant, mit Hilfe des WPF-Frameworks durchgeführt. Die Autorin hat das Markup, in dem das Design der Oberfläche definiert ist, auf Grundlage der in der Entwicklungsphase entworfenen Benutzeroberflächenmockups erstellt. Das Accordion-Control, welches nicht unter den Standardcontrols vorhanden war, wurde über das NuGet-Package WPFToolkit[26] bereitgestellt. Die Funktionalität, die sich hinter den einzelnen GUI-Elementen verbirgt, wurde über den Code-Behind festgelegt. In ihm wurden die Funktionen, die über die anderen Klassen der Anwendung zur Verfügung gestellt werden, den entsprechenden Elementen zugeordnet. Da die Funktionsweise dieser Methoden bereits in den einzelnen Projekten validiert wurde, erfolgte an dieser Stelle nur noch ein Integrationstest. Dieser stellt den korrekten Ablauf und Aufruf der einzelnen Funktionen sicher. Die gesamte Implementierungsphase konnte mit dem Fertigstellen der Benutzeroberfläche abgeschlossen werden. Ein Screenshot der erstellten GUI befindet sich im Anhang A.20: Screenshot der Anwendung auf S. xx.

6 Abnahme- und Einführungsphase

6.1 Abnahme durch den Fachbereich

Nachdem die gesamte Anwendung fertig gestellt war, konnte diese dem Fachbereich zur Endabnahme vorgelegt werden. Aufgrund der agilen Softwareentwicklungsmethode wurde den Fachbereichen nach

[25] Dabei handelt es sich um eine Pseudoklasse, die vom *Moq-Framework* erzeugt wird.
[26] Vgl. https://www.nuget.org/packages/WPFToolkit/.

jeder Iteration die aktuelle Version der Anwendung präsentiert. Dadurch waren sie bei der Endabnahme bereits mit der Oberfläche und der Funktionsweise des Programmes vertraut. Außerdem konnten Anregungen und Kritik der Fachbereiche durch die stetigen Rücksprachen schon frühzeitig während der Entwicklungsphase berücksichtigt werden. Dadurch ergaben sich bei der Endabnahme keine Probleme oder Hindernisse mehr, sodass der Einführung der Anwendung nichts mehr im Wege stand. Vor dem Deployment wurde zur Qualitätssicherung zusätzlich zur Abnahme durch den Fachbereich ein Code-Review durch einen anderen Entwickler durchgeführt.

6.2 Deployment und Einführung

Um den Zugriff auf die Anwendung für die Fachbereiche möglichst einfach zu gestalten, wurde die Anwendung zentral über ein bereits existierendes Netzlaufwerk, für das alle Beteiligten die nötigen Berechtigungen haben, bereitgestellt. Die Anwendung wurde als kompiliertes, ausführbares Programm auf diesem Laufwerk abgelegt. Außerdem wurde bei jedem Benutzer eine Desktopverknüpfung angelegt, um einen schnellen und einfachen Zugriff auf die Anwendung zu gewährleisten. Wie bereits im vorherigen Abschnitt erwähnt wurde, waren die Fachbereiche durch die stetige Rückkopplung schon mit der Anwendung vertraut. Daher waren zusätzliche Benutzerschulungen nicht erforderlich.

7 Dokumentation

Die Dokumentation der Übertragungswertverwaltung besteht aus drei Bestandteilen: der Projektdokumentation, dem Benutzerhandbuch und der Entwicklerdokumentation. In der Projektdokumentation beschreibt die Autorin die einzelnen Phasen, die während der Umsetzung des Projektes durchlaufen wurden.

Das Benutzerhandbuch enthält Informationen über den Aufbau und die Funktionsweise der Anwendung. Es soll den Fachbereichen als Anhaltspunkt für Nachfragen zur Verfügung stehen und zur Einarbeitung neuer Mitarbeiter in die Anwendung dienen. Ein Auszug aus dieser Dokumentation befindet sich im Anhang A.21: Auszug aus der Benutzerdokumentation auf S. xxi.

Bei der Entwicklerdokumentation handelt es sich um eine detaillierte Beschreibung der Klassen, die in der Anwendung verwendet werden. Außerdem werden auch deren Attribute und Methoden sowie die Abhängigkeiten der Klassen untereinander erläutert. Diese Dokumentation soll dem Entwickler als Übersicht und Nachschlagewerk dienen. Mit Hilfe des Tools Sandcastle[27] wurde diese Dokumentation automatisch generiert. Dazu wandelt das Tool die Kommentare im Programmcode in eine XML-Datei um, aus der schließlich eine HTML-Seite erzeugt wird. Ein Ausschnitt aus der Entwicklerdokumentation befindet sich im Anhang A.22: Entwicklerdokumentation auf S. xxii. Zusätzlich zu der Entwicklerdokumentation wurde für jede Komponente ein Klassendiagramm aus dem Quellcode generiert. Im Anhang A.23: Klassendiagramm des Domänenmodells auf S. xxiii befindet sich das Klassendiagramm für die Komponente `Domaenenmodell`.

[27] Vgl. https://shfb.codeplex.com/.

8 Fazit

8.1 Soll-/Ist-Vergleich

Bei einer rückblickenden Betrachtung des IHK-Abschlussprojektes, kann festgehalten werden, dass alle zuvor festgelegten Anforderungen gemäß dem Pflichtenheft erfüllt wurden. Der zu Beginn des Projektes im Abschnitt 2.1 (Projektphasen) erstellte Projektplan konnte eingehalten werden. In der Tabelle 4: Soll-/Ist-Vergleich wird die Zeit, die tatsächlich für die einzelnen Phasen benötigt wurde, der zuvor eingeplanten Zeit gegenübergestellt. Es ist zu erkennen, dass nur sehr geringfügig von der Zeitplanung abgewichen wurde. Die sich daraus ergebenen Differenzen konnten untereinander kompensiert werden, sodass das Projekt in dem von der IHK festgelegten Zeitrahmen von 70 Stunden umgesetzt werden konnte.

Projektphase	Soll	Ist	Differenz
Analysephase	8 h	8,5 h	+ 0,5 h
Entwurfsphase	11 h	10 h	- 1 h
Implementierungsphase	40 h	40,5 h	+ 0,5 h
Abnahme und Einführung	1 h	1 h	0 h
Erstellen der Dokumentation	10 h	10 h	0 h
Gesamt	**70 h**	**70 h**	**0 h**

Tabelle 4: Soll-/Ist-Vergleich

8.2 Lessons Learned

Im Zuge des Projektes konnte die Autorin wertvolle Erfahrungen bzgl. der Planung und Durchführung von Projekten sammeln. Dabei wurde besonders deutlich, von welch großer Bedeutung stetige Kommunikation untereinander und Rücksprachen mit den Fachbereichen für eine erfolgreiche Projektumsetzung sind. Außerdem konnten neue Erkenntnisse in Bezug auf das Einbinden und Nutzen von Frameworks gewonnen werden. Das Entity Framework erwies sich beispielsweise als sehr hilfreich, da es sich um das Mapping der Objekte und Relationen kümmert und die Autorin sich somit auf die wesentlichen Komponenten der Anwendung konzentrieren konnte. Abschließend kann man sagen, dass die Realisierung des Projektes nicht nur einen Mehrwert für die Fachbereiche bietet, sondern auch für die Autorin eine große Bereicherung war.

8.3 Ausblick

Obwohl alle im Lastenheft definierten Anforderungen realisiert werden konnten, können in Zukunft dennoch neue Anforderungen definiert bzw. Erweiterungsvorschläge entwickelt werden. Vom Rechnungswesen wurde beispielsweise bereits angefragt, ob es möglich wäre, auch die abschließende Verbuchung der Übertragungswerte in der FiBu über die Anwendung automatisch durchzuführen. Außerdem ist angedacht, die Anwendung in Zukunft auch in dem Geschwisterunternehmen einzusetzen. Aufgrund des im Abschnitt 4.2 (Architekturdesign) beschriebenen modularen Aufbaus des Projektes können solche Anpassungen bzw. Erweiterungen sehr einfach vorgenommen werden. Die Modularität der Anwendung ermöglicht somit eine gute Wartbarkeit und Erweiterbarkeit.

Literaturverzeichnis

ALTE OLDENBURGER 2013

ALTE OLDENBURGER: *Geschäftsbericht 2013*. http://www.alte-oldenburger.de/web/export/sites/aob/_resources/download_galerien/downloads_pdf/daten_und_fakten/Geschaeftsbericht_2013_ALTE_OLDENBURGER_Krankenversicherung_AG.pdf. Version: 2013

Bleek und Wolf 2008

Bleek, W.-G. ; Wolf, H.: *Agile Softwareentwicklung: Werte, Konzepte und Methoden*. dpunkt.verlag, 2008. – ISBN 978–3–89864–473–0

Eilebrecht und Starke 2007

Eilebrecht, K. ; Starke, G.: *Patterns kompakt – Entwurfsmuster für effektive Software-Entwicklung*. Spektrum Akademischer Verlag, 2007. – ISBN 978–3–8274–1591–2

EntityFrameworkTutorial.net 2014

EntityFrameworkTutorial.net: *What is Entity Framework?* http://www.entityframeworktutorial.net/what-is-entityframework.aspx. Version: 2014

Freeman u. a. 2004

Freeman, E. ; Freeman, El. ; Sierra, K. ; Bates, B.: *Head First Design Patterns*. O'Reilly Media, 2004. – ISBN 978–0–59600–712–6

Microsoft Corporation 2014

Microsoft Corporation: *Einführung in WPF*. http://msdn.microsoft.com/de-de/library/aa970268%28v=vs.110%29.aspx. Version: 2014

Pilcher 2007

Pilcher, R.: *Scrum – Agiles Projektmanagement erfolgreich einsetzen*. dpunkt.verlag, 2007. – ISBN 978–3–89864–478–5

A Anhang

A.1 Detaillierte Zeitplanung

Analysephase		**8 h**
1. Ist-Analyse durchführen (Fachgespräche, EPK, Use-Case-Diagramm)		4 h
2. Wirtschaftlichkeitsanalyse und Nutzwertanalyse durchführen		2 h
3. Unterstützung des Fachbereiches bei der Erstellung des Lastenheftes		2 h
Entwurfsphase		**11 h**
1. Zielplattform festlegen		1 h
2. Benutzeroberfläche entwerfen		1 h
3. Datenbank entwerfen (ERM, Tabellenmodell)		2 h
4. Planung der Geschäftslogik (Entscheidungstabelle, Komponentendiagramm)		4 h
5. Pflichtenheft erstellen		3 h
Implementierungsphase		**40 h**
1. Anlegen der Datenbank		1 h
2. Implementierung des Domänenmodells inkl. Tests		7 h
3. Implementierung der Geschäftslogik inkl. Tests		23 h
3.1. Implementierung Datenimport aus CSV-Dateien	2 h	
3.2. Implementierung der Datenpersistenz	4 h	
3.3. Implementierung der Verarbeitungslogik	8 h	
3.4. Implementierung der Versis-Logik (WDO-Status setzen, Datenlieferung)	5 h	
3.5. Anbindung der Schnittstellen (Webservices, Aufruf PAM-API)	1 h	
3.6. Implementierung Datenexport	3 h	
4. Implementierung der Benutzeroberfläche inkl. Tests		9 h
Abnahme und Einführung		**1 h**
1. Abnahme durch die Fachabteilung		0,5 h
2. Installation der Desktopanwendung		0,5 h
Erstellen der Dokumentationen		**10 h**
1. Erstellen der Projektdokumentation		8 h
2. Erstellen der Entwicklerdokumentation		1 h
3. Erstellen der Benutzerdokumentation		1 h
Gesamt		**70 h**

A.2 Verwendete Ressourcen

Hardware

- Büroarbeitsplatz mit Thin-Client

Software

- Windows 7 Enterprise mit Service Pack 1 – Betriebssystem
- Visual Studio Professional 2013 – Entwicklungsumgebung C#
- Oracle – Datenbanksystem
- Oracle SQL Developer – Verwaltungswerkzeug für Oracle-Datenbanken
- Enterprise Architect – Programm zum Erstellen verschiedener Modelle und Diagramme
- ARIS Express - Programm zum Erstellen von Diagrammen (EPK)
- git – Verteilte Versionsverwaltung
- MiKTeX – Distribution des Textsatzsystems TEX
- Eclipse Luna mit TeXlipse – Entwicklungsumgebung LATEX
- Balsamiq – Programm zur Erstellung von Mockups
- NUnit – Framework zur Durchführung von Unit-Tests
- Moq – Mocking-Framework zur Erstellung von Pseudoklassen

Personal

- Mitarbeiter des Rechnungswesens / der Antragsabteilung – Festlegung der Anforderungen und Abnahme des Projektes
- Entwicklerin – Umsetzung des Projektes
- Anwendungsentwickler – Review des Codes

A.3 EPK des Ist-Zustandes (Ausschnitt)

Der folgende Ausschnitt soll das hohe Maß an manueller Arbeit exemplarisch verdeutlichen.

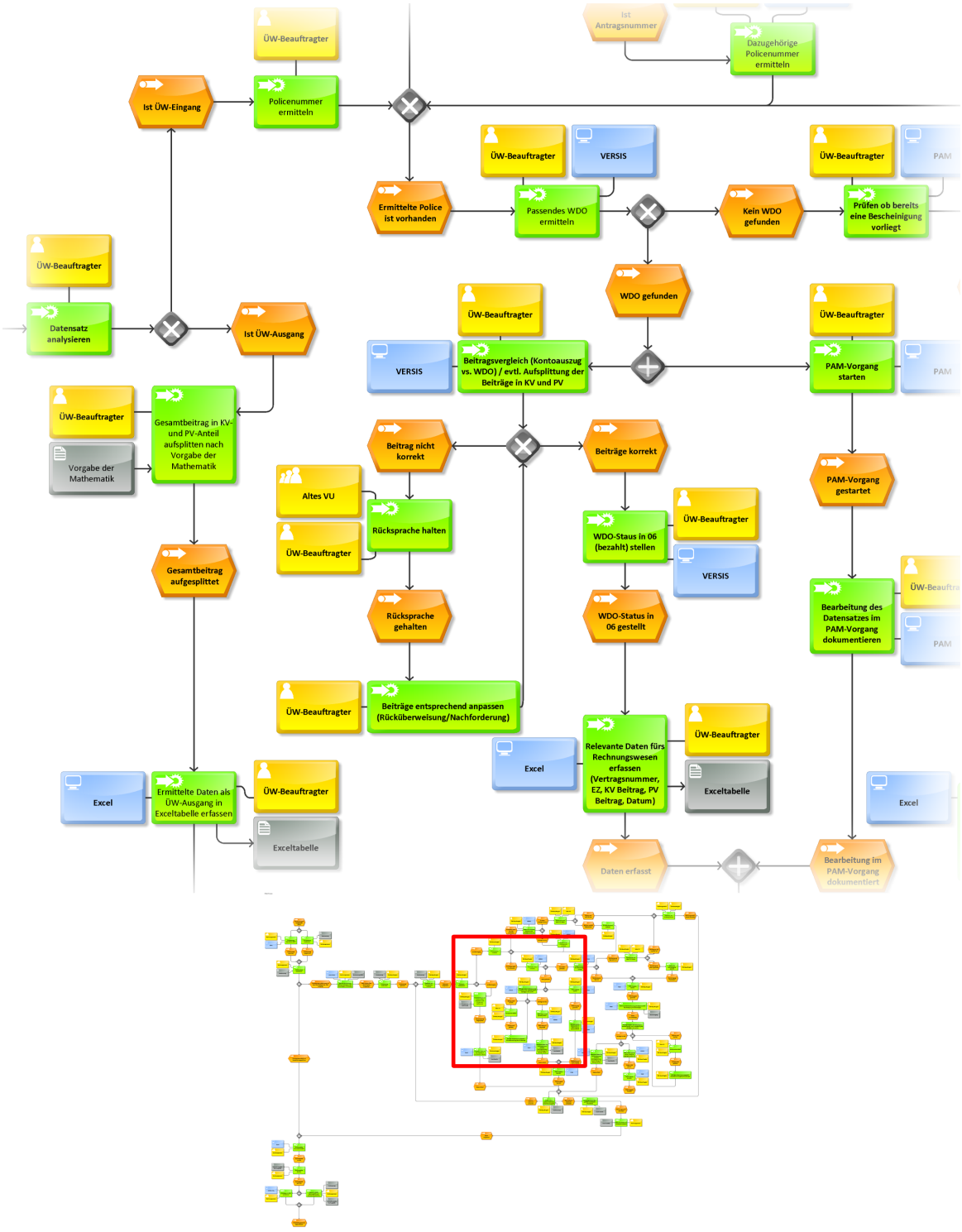


Abbildung 1: Ausschnitt aus der EPK

A.4 Lastenheft (Auszug)

Im folgenden Auszug aus dem Lastenheft werden die Anforderungen definiert, die die zu entwickelnde Anwendung erfüllen muss.

Anforderungen

Von der Anwendung müssen folgende Anforderungen erfüllt werden:

- Die Anwendung muss einen CSV-Import bereitstellen, mit dem die Übertragungswert-Kontoauszüge (CSV-Dateien) in das neue System importiert werden können.
- Die Anwendung muss die Daten aus dem Kontoauszug extrahieren, die für den Übertragungswertverarbeitungsprozess relevant sind. Dazu zählen: Kontoauszugsdatum, Anfangs- und Endsaldo, Kontoumsätze mit ihrem jeweiligen Verwendungszweck, Betrag und Buchungsdatum.
- Die Anwendung muss die verschiedenen Übertragungswertsätze automatisch einer der drei möglichen Kategorien (Übertragungswerteingang, Übertragungswertausgang, Sonstiges) zuordnen und dementsprechend verarbeiten.
- Die Anwendung muss eine Schnittstelle zum Bestandsführungssystem Versis anbieten, damit die aus dem Kontoauszug ermittelten Daten automatisch vervollständigt und validiert werden können und der Status des WDOs gesetzt werden kann.
- Die Anwendung muss die Kontoauszüge mit ihren Kontoumsätzen im Original und in der vervollständigten Form historisiert archivieren, damit sie jederzeit einsehbar sind.
- Die Anwendung muss eine Schnittstelle zum Dokumentenmanagementsystem PAM bereitstellen, damit die PAM-Vorgänge automatisch gestartet, bearbeitet und beendet werden können.
- Die Anwendung muss eine Übersicht über alle Übertragungswertsätze in Tabellenform anbieten, insbesondere der Sätze, die noch vom Rechnungswesen in der FiBu erfasst werden müssen.
- Die Anwendung muss die Möglichkeit bieten, automatisch eine Buchungshilfe für einen Kontoauszug zu generieren.
- Die Anwendung muss eine manuelle Nachbearbeitungmöglichkeit für den Sachbearbeiter zur Verfügung stellen, wenn ein Kontoauszug nicht automatisch verarbeitet werden kann.
- Die Anwendung muss eine Möglichkeit bereitstellen, automatisiert Auswertungen zu generieren (jährliche, monatliche Auswertungen).
- Die Anwendung muss eine Suchfunktion bereitstellen, damit nach einzelnen Übertragungswertsätzen gesucht werden kann.

[...]

A.5 Use-Case-Diagramm

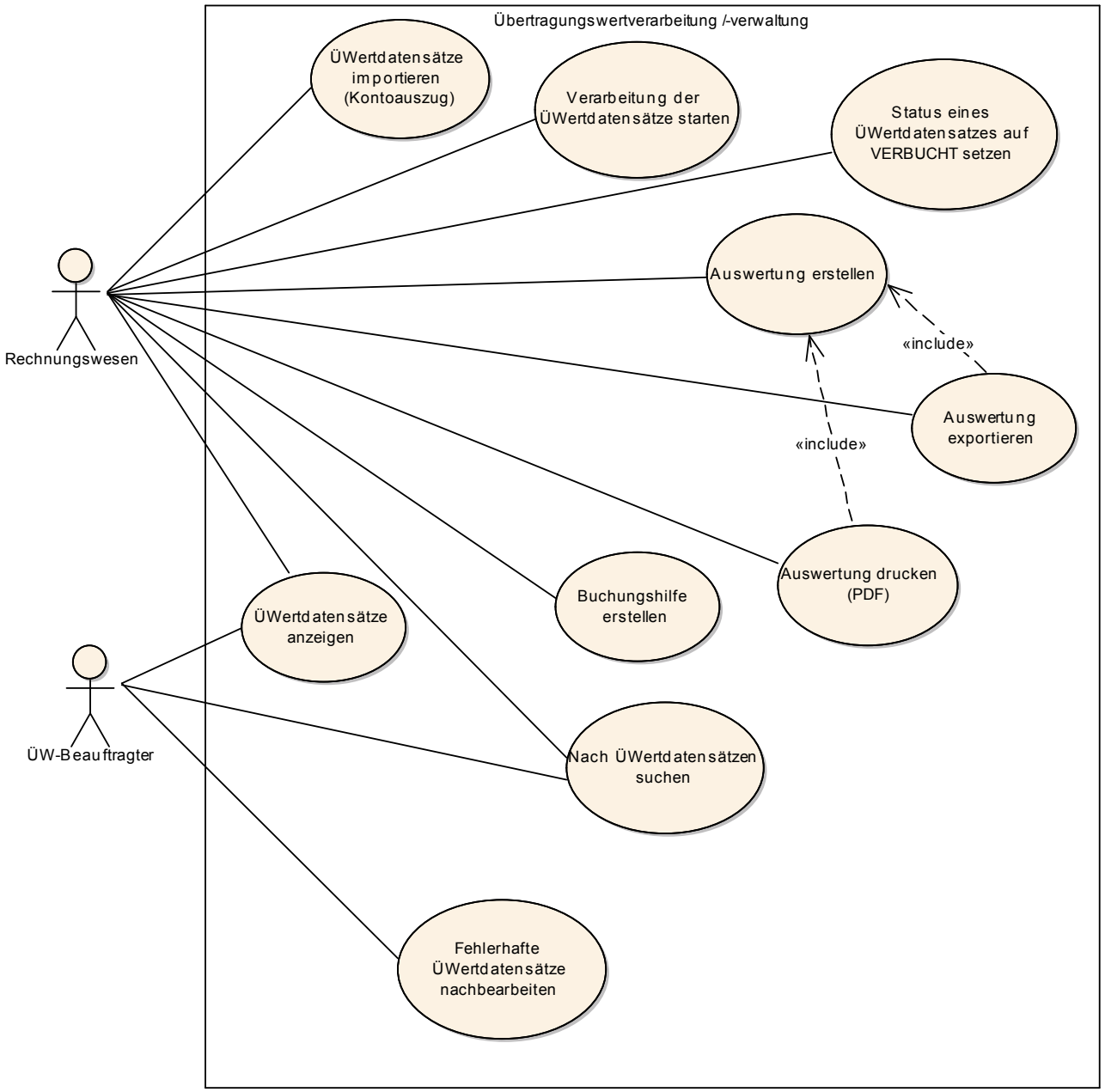


Abbildung 2: Use-Case-Diagramm

A.6 Amortisation

Der Schnittpunkt der beiden Geraden gibt den Zeitpunkt der Amortisation an.

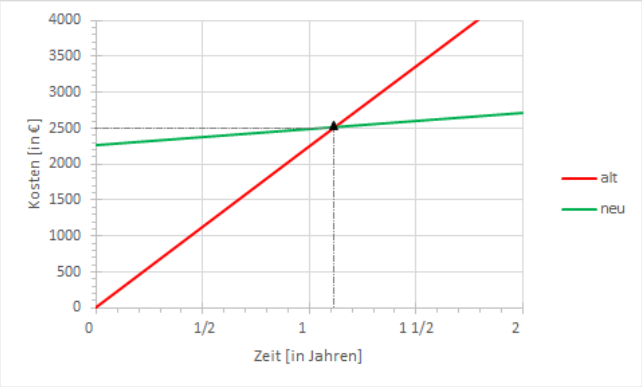


Abbildung 3: Graphische Darstellung der Amortisation

A.7 Nutzwertanalyse zur Auswahl der Programmiersprache

Eigenschaft	Gewichtung	Natural (bew.)	Java (bew.)	C# (bew.)	Natural (gew.)	Java (gew.)	C# (gew.)
Testbarkeit	4	2	5	5	8	20	20
Oracle-Zugriff	5	2	4	4	10	20	20
Webservices konsumieren	5	2	4	5	10	20	25
Plattform: Windows	5	3	4	5	15	20	25
Kenntnisstand (Entwickler-AO)	3	5	4	3	15	12	9
Entwicklungsumgebung	3	3	5	5	9	15	15
Kosten	4	2	5	4	8	20	16
Office-Integration	3	1	3	5	3	9	15
Umfang der verfügbaren Bibliotheken	3	1	5	4	3	15	12
Performance	3	5	4	4	15	12	12
Benutzeroberfläche	5	3	3	5	15	15	25
Reguläre Ausdrücke	4	1	4	4	4	16	16
Gesamt:	**47**				**115**	**194**	**210**
Nutzwert:					**2,45**	**4,13**	**4,47**

A.8 Oberflächenentwürfe

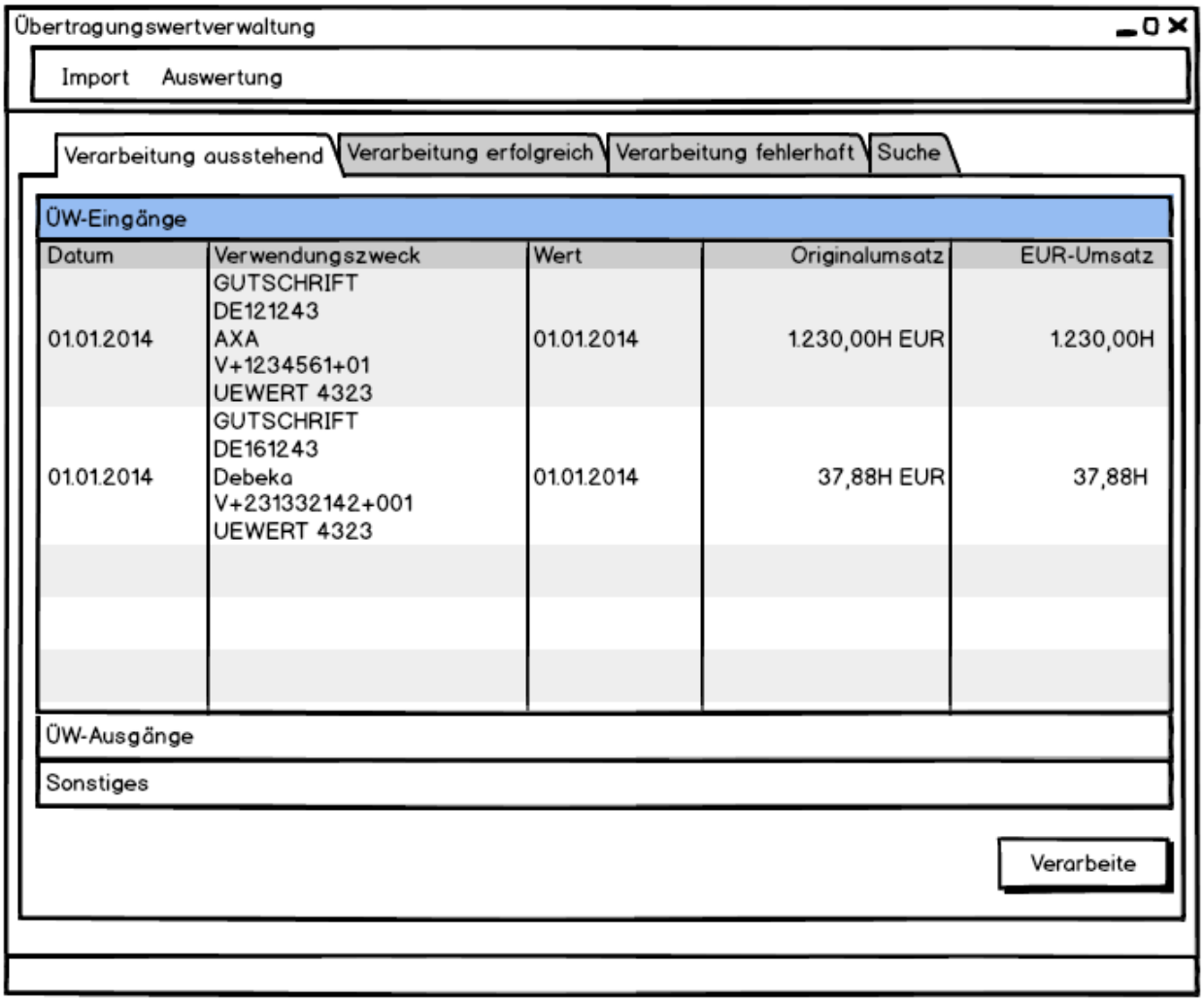


Abbildung 4: Hauptfenster

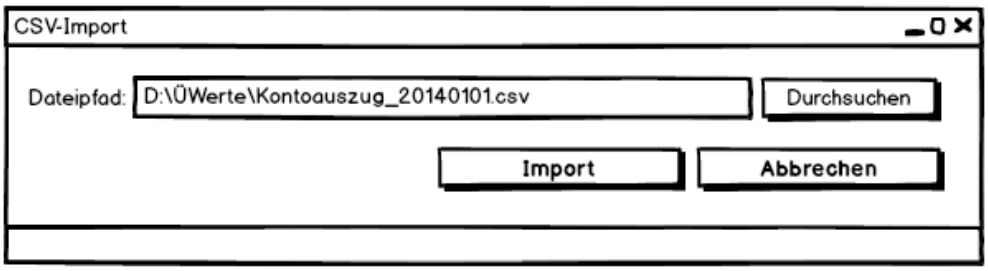


Abbildung 5: Importfenster

A.9 Analyse des Prozessablaufs und Ermittlung der notwendigen Entitätstypen

Das Rechnungswesen erhält einen Kontoauszug. Dieser kann einen oder mehrere Kontoumsätze enthalten. Um diese Daten abspeichern zu können, werden die Entitätstypen `Kontoauszug` und `Kontoumsatz` benötigt. Neben den Umsätzen können noch weitere Angaben aus dem Kontoauszug entnommen werden. Dazu zählen beispielsweise der Anfangssaldo und das Auszugsdatum. Diese werden dem `Kontoauszug` als Attribute zugeordnet. Ein Kontoumsatz bildet einen Buchungssatz ab und enthält den jeweiligen Betrag, das Buchungsdatum, einen Buchungstext sowie einen Verwendungszweck. Diese Daten werden dem `Kontoumsatz` als Attribute zugeordnet. Bei dem Kontoumsatz kann es sich entweder um einen Übertragungswerteingang bzw. -ausgang oder um Kontoführungsgebühren handeln. Handelt es sich um einen Ein- bzw. Ausgang, so sind in dem Verwendungszweck Daten enthalten, die für den weiteren Verarbeitungsprozess relevant sind.

Beispielinhalt eines Verwendungszwecks (Übertragungswerteingang):

EREF+INAR12345678SVWZ+UEWERT++P+4142+123456001+01++P+4044+453212345+

Aus dem Verwendungszweck müssen die Daten zunächst extrahiert und dann gespeichert werden. Dazu wird ein weiterer Entitätstyp benötigt. Dieser wird als `Erweiterter_Kontoumsatz` bezeichnet und enthält alle relevanten Daten, die aus dem Verwendungszweck ermittelt werden konnten. Dazu zählen beispielsweise die Vertragsnummer, die Personennummer sowie die Produktkomponente etc. Bei Übertragungswerteingängen kann es vorkommen, dass in dem Verwendungszweck nicht nur Daten für eine Produktkomponente enthalten sind, sondern für beide (Pflegeversicherung (PV) und Krankenversicherung (KV)). Daher können sich aus einem Kontoumsatz bis zu zwei `Erweiterter_Kontoumsatz` ergeben.

Bei dem vorliegenden Beispiel oben sind nur Daten zur Produktkomponente PV enthalten (gekennzeichnet durch das *P*). Daher ergibt sich aus dem Kontoauszug genau ein `Erweiterter_Kontoumsatz`. Dieser enthält neben der Produktkomponente beispielsweise auch die BaFin-Nummer des Unternehmens, zu dem der Versicherte wechselt (hier *4142*), sowie die Vertrags- und Personennummer (hier *123456001* und *01*) etc.

Da die Daten, die in `Erweiterter_Kontoumsatz` gespeichert werden, nicht immer vollständig sind bzw. auch fehlerhaft sein können, wird ein weiterer Entitätstyp benötigt. Dieser wird als `Korrekter_Kontoumsatz` bezeichnet. In ihm sind alle für die Verbuchung nötigen Daten enthalten. Daten, die in `Erweiterter_Kontoumsatz` evtl. noch fehlerhaft waren, befinden sich in `Korrekter_Kontoumsatz` in korrigierter Form. Außerdem kann dieser manuell um weitere nötige Angaben angereichert worden sein.

In dem Beispiel wurde während des weiteren Verarbeitungsprozesses festgestellt, dass es sich bei der extrahierten Vertragsnummer nicht um eine Vertrags- sondern um eine Antragsnummer handelt. Daher wurde die korrekte Vertragsnummer ermittelt und in `Korrekter_Kontoumsatz` zusammen mit den anderen validierten Daten hinterlegt. Mit Hilfe der genannten Entitätstypen kann nachvollzogen werden, wie sich die einzelnen Daten aus dem Kontoauszug ergeben haben.

A.10 Entity-Relationship-Model

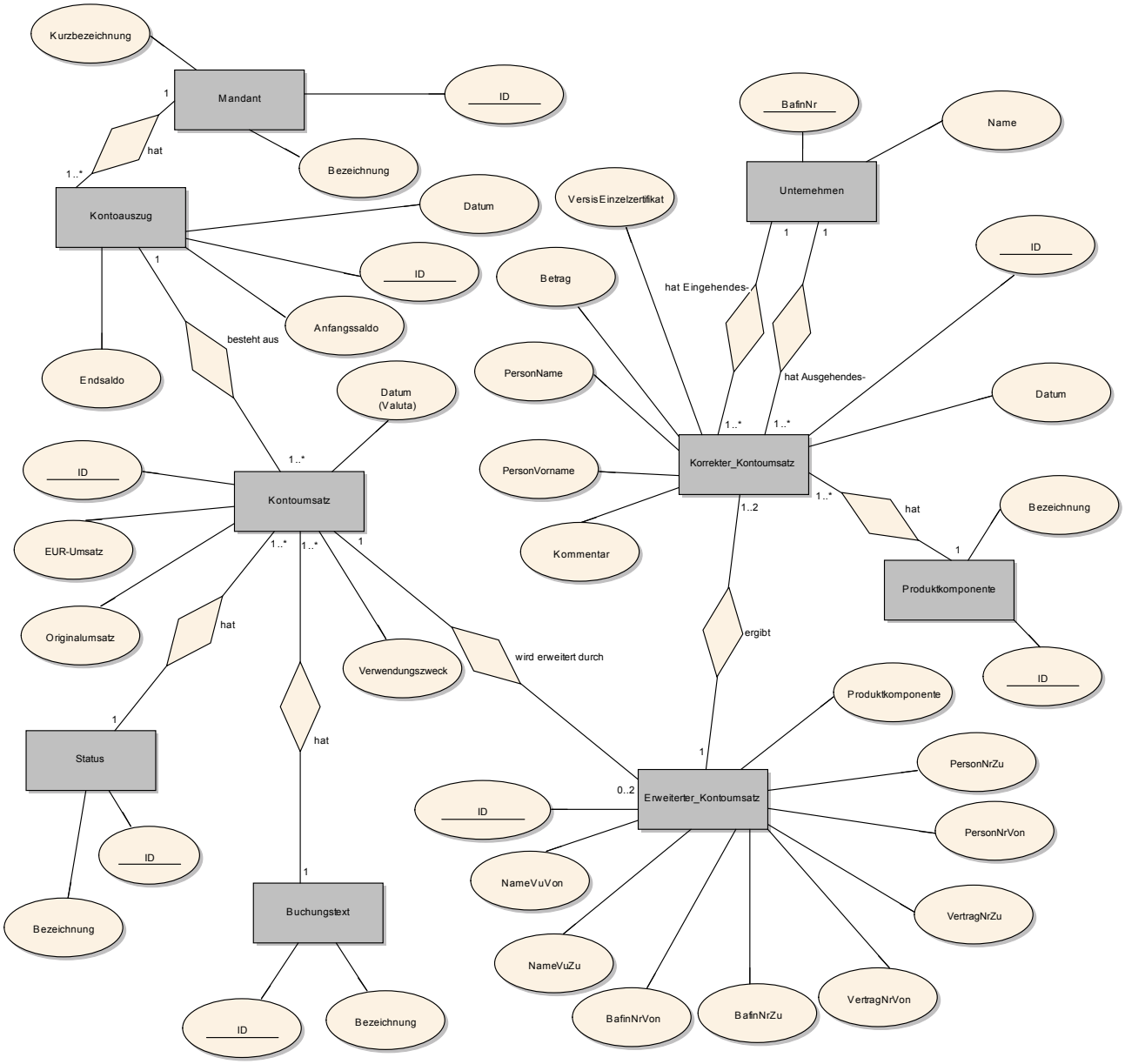


Abbildung 6: Entity-Relationship-Model

A.11 Tabellenmodell

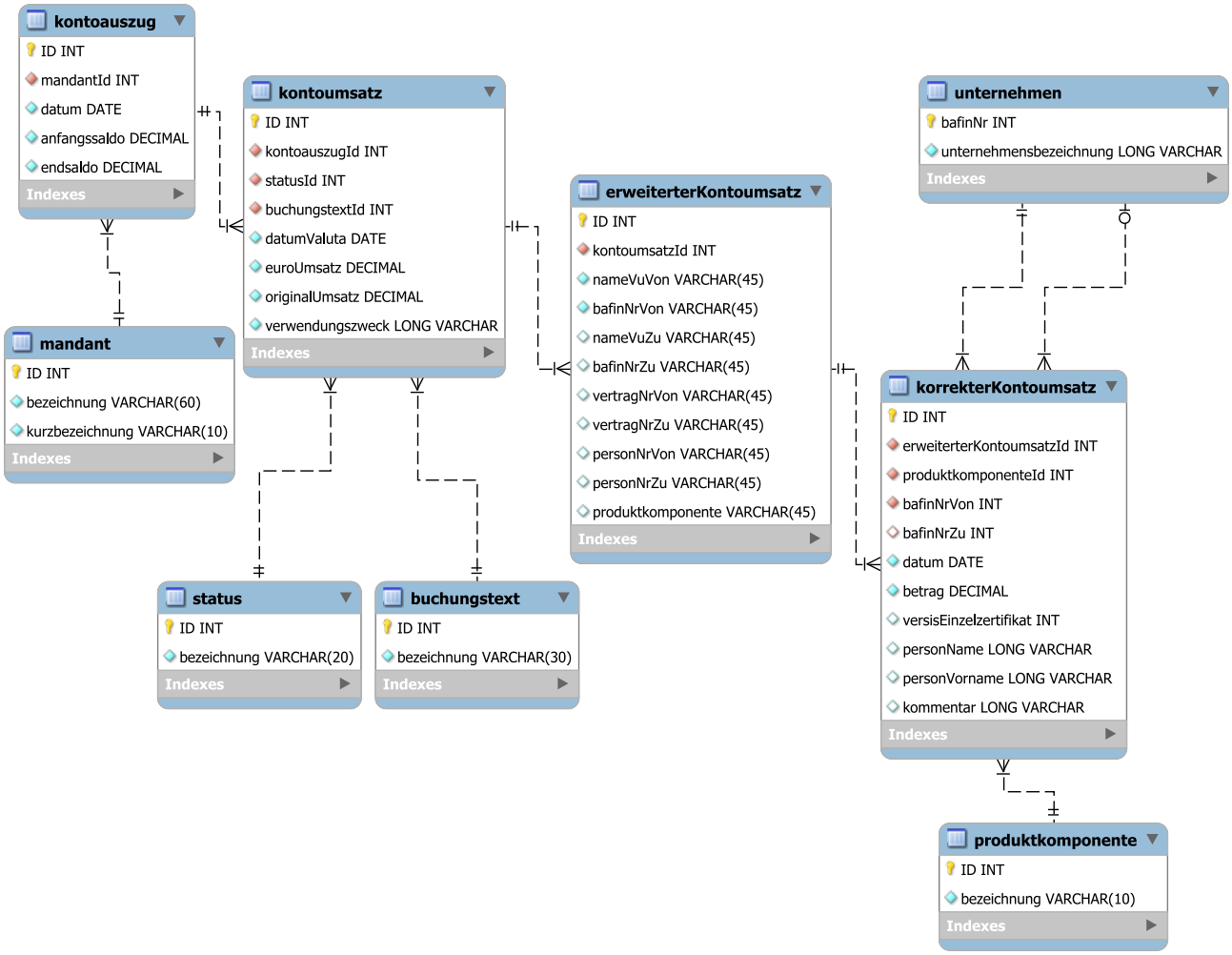


Abbildung 7: Tabellenmodell

A.12 Komponentendiagramm

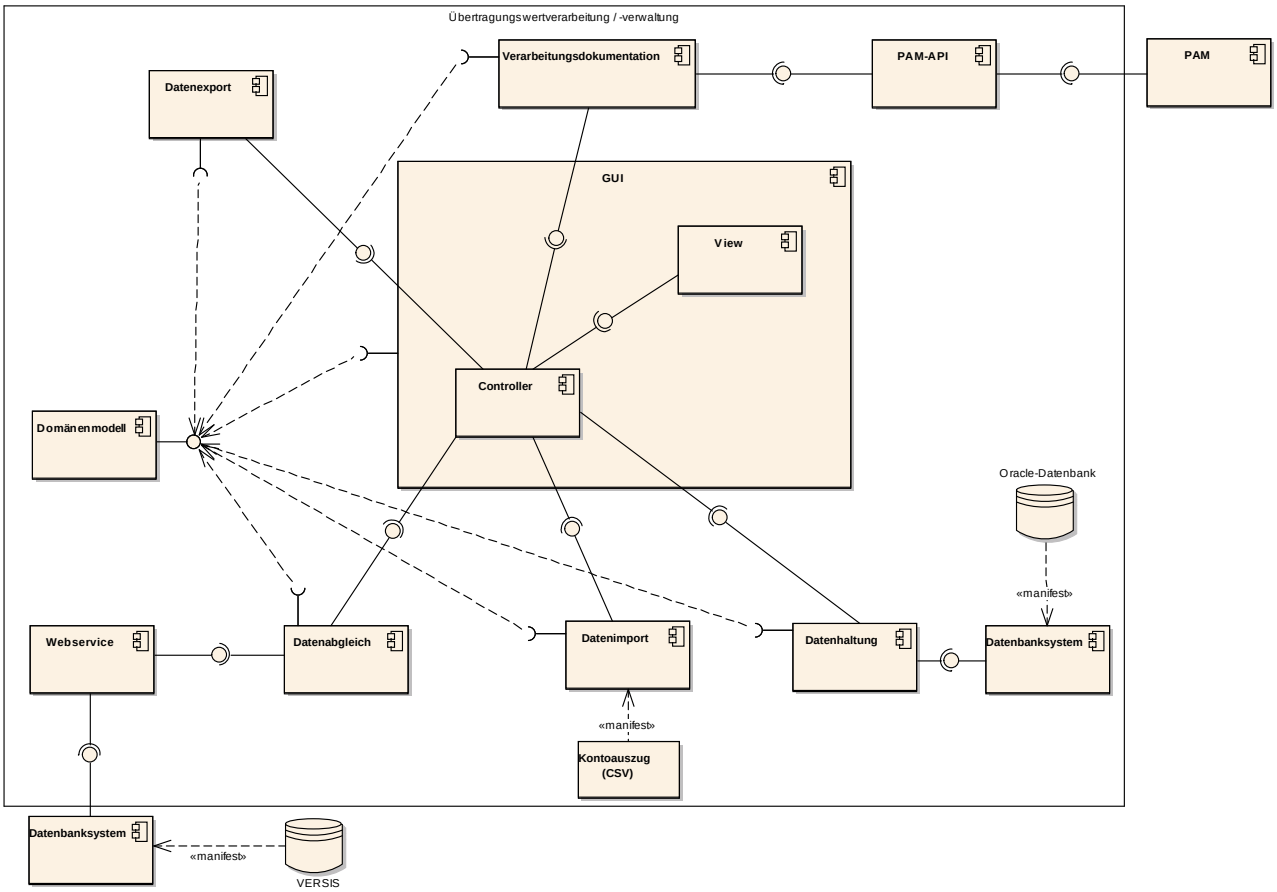


Abbildung 8: Komponentendiagramm

A.13 Entscheidungstabelle

ÜW-Verarbeitung	Fall 1	Fall 2	Fall 3	Fall 4
Bedingungen				
Einzelzertifikatnummer vorhanden	J	J	J	N
WDO vorhanden	J	J	N	-
Betrag korrekt	J	N	-	-
Aktionen				
Einzelzertifikatnummer ermitteln				X
Rücksprache mit Fachbereich			X	X
Rücksprache mit altem Versicherungsunternehmen		X	X	X
Vorgang in PAM starten	X	X	X	
Vorgang in PAM abschließen	X			
Status des WDOs auf 06 (bezahlt) setzen	X			
Korrekten Kontoumsatz in Datenbank schreiben	X			

A.14 Pflichtenheft (Auszug)

In folgendem Auszug aus dem Pflichtenheft wird die geplante Umsetzung der im Lastenheft definierten Anforderungen beschrieben:

Umsetzung der Anforderungen

- Der CSV-Import der Übertragungswertkontoauszüge soll direkt über einen Button innerhalb der Anwendung aufgerufen werden können, und mit Hilfe des NuGet-Packages `CsvHelper` in C# umgesetzt werden.
- Die erste Extraktion der Daten aus dem Kontoauszug soll ebenfalls mit Hilfe des NuGet-Packages erfolgen. Die genauere Auftrennung der Daten (Verwendungszweck) soll anhand von regulären Ausdrücken vollzogen werden.
- Die Zuordnung der einzelnen Kontoumsätze zu einer der drei Kategorien (Übertragungswerteingang, -ausgang oder Sonstiges) soll anhand eines dem Umsatz zugeordneten Status erfolgen.
- Die Schnittstelle zwischen dem Bestandsführungssystem Versis und der zu entwickelnden Anwendung soll durch einen Webservice realisiert werden. Dieser Webservice soll anhand der als Parameter übergebenen Vertragsnummer alle Daten ermitteln und zurückliefern, die für die Verbuchung der Übertragungswerteingänge relevant sind. Außerdem soll er den aus dem Kontoumsatz ermittelten Betrag mit dem in Versis hinterlegten Wert abgleichen sowie den Status des zugehörigen WDOs setzen.
- Die Daten sollen historisiert in einer Oracle-Datenbank abgelegt werden. Dafür stehen die Tabellen `Kontoauszug`, `Kontoumsatz`, `Erweiterter_Kontoumsatz` sowie `Korrekter_Kontoumsatz` zur Verfügung. Mit Hilfe einer in der Anwendung eingebauten Suchfunktion soll nach den gewünschten Daten gesucht werden können.
- Das objektrelationale Mapping soll mit Hilfe des Entity-Frameworks umgesetzt werden.
- Die Kommunikation zwischen dem Dokumentenmanagementsystem PAM und der Anwendung soll über die Implementierung einer `ÜbertragungswertFabrik` realisiert werden. Diese stellt einen weiteren Eingangskanal für BREPL dar. Über BREPL und den neuen Eingangskanal kann dann der Aufruf der PAM-API erfolgen.
- Die Daten sollen mit Hilfe des WPF-Controls `DataGrid` tabellarisch abgebildet werden. Um einen besseren Überblick darüber zu erhalten, welche Daten noch verarbeitet werden müssen und welche bereits verbucht werden können, soll die Anwendung folgende Reiter anbieten: Verarbeitung ausstehend, Verarbeitung fehlerhaft, Verarbeitung erfolgreich und Suche. Innerhalb dieser Reiter sollen die Daten nochmal in Unterkategorien aufgeteilt werden: Übertragungswerteingänge, Übertragungswertausgänge und Sonstiges. Diese Unterkategorien sollen mit Hilfe des WPF-Controls `Accordion` realisiert werden.

[...]

A.15 Iterationsplan

- Erstellung der Datenbank
- Erstellung des Domänenmodells
- Implementierung „CSV-Import“
- Implementierung „Datenpersistenz“ mit Hilfe des Entity Frameworks
- Implementierung „Versis-Logik“
- Implementierung „Übertragungswert-Fabrik“ für die Kommunikation mit PAM über BREPL
- Implementierung „Datenexport“
- Implementierung „Benutzeroberfläche“

A.16 Screenshot des Grundgerüstes der Solution

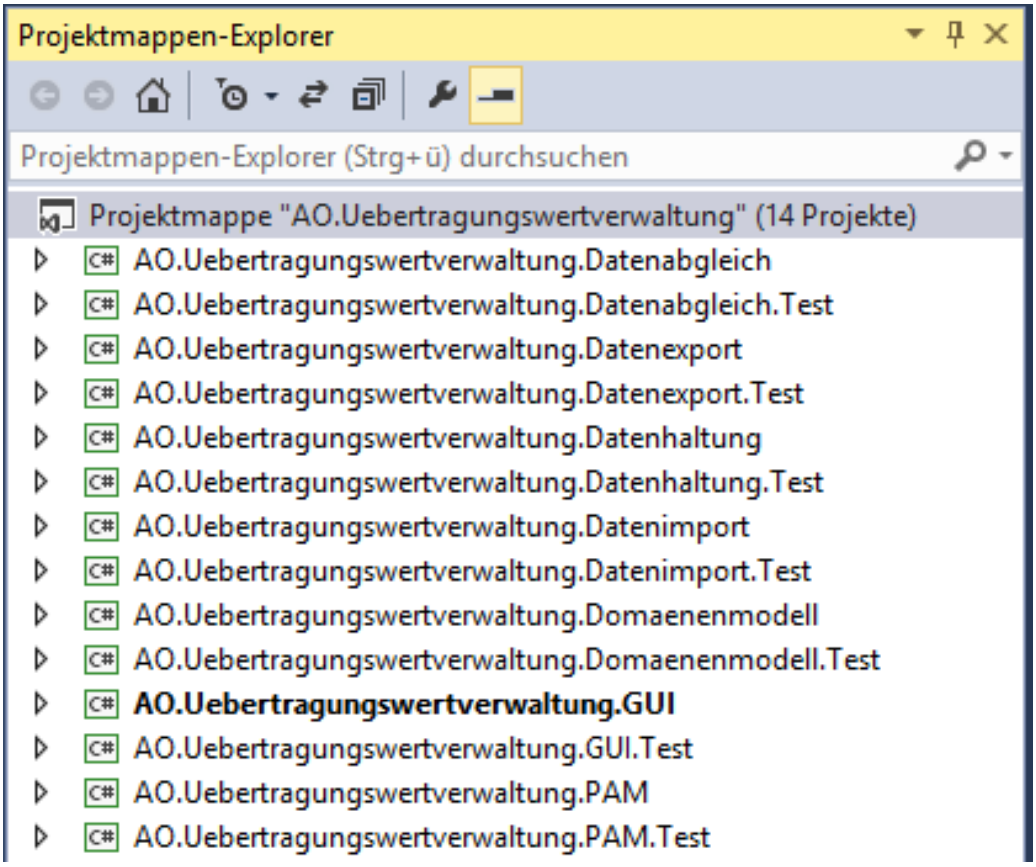
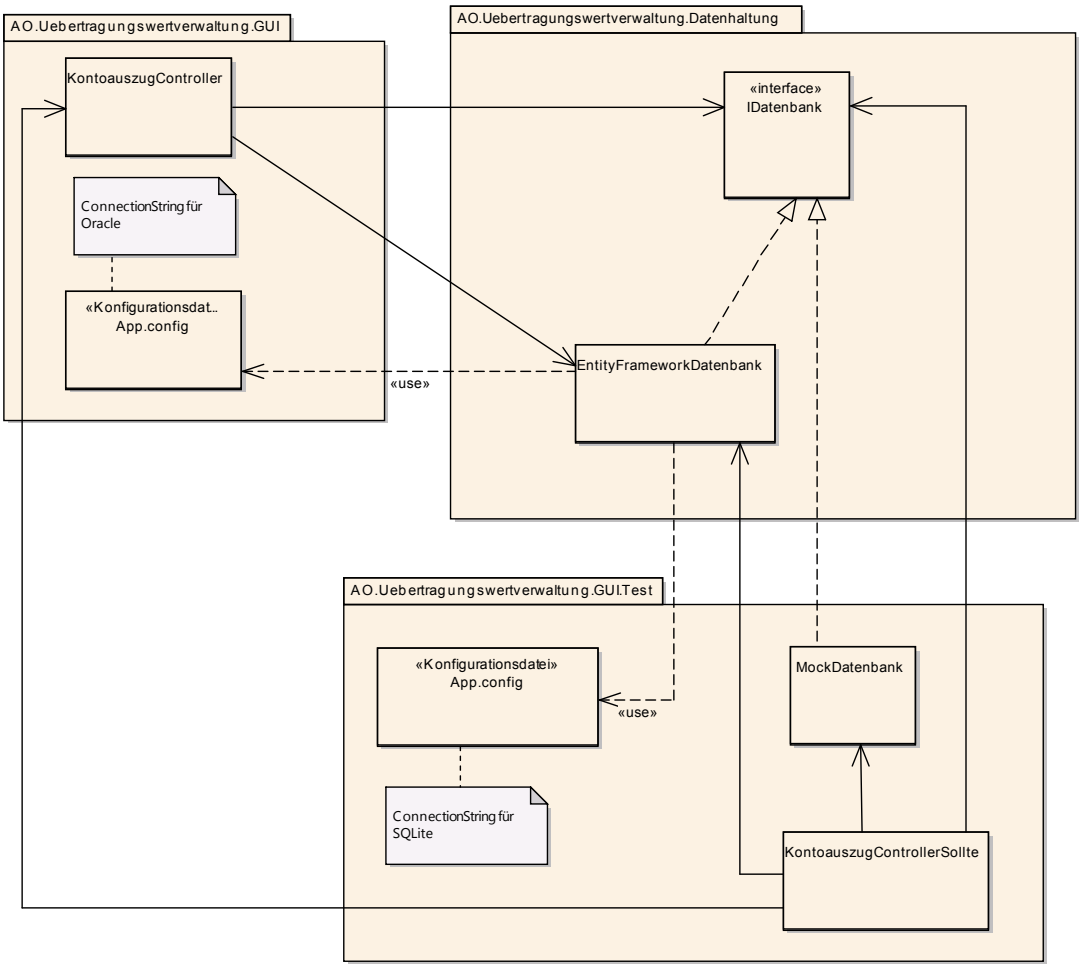


Abbildung 9: Screenshot der Solution

A.17 Konzeptionelles Klassendiagramm



A.18 Listing der Tests für die Klasse `CsvKontoauszugsquelle` (Ausschnitte)

```
1  [...]
2
3  [TestFixture]
4  public class CsvKontoauszugsquelleSollte
5  {
6    private const String FEHLERMELDUNG = "Fehler beim Importieren des Kontoauszugs: ";
7    private const Char ANFUEHRUNGSZEICHEN = '"';
8    private const String TRENNZEICHEN = ";";
9    private CsvKontoauszugsquelle sut;
10   private StringReader reader;
11
12   private CsvKontoauszugsquelle ErzeugeKontoauszugsquelle(String csvInhalt)
13   {
14     reader = new StringReader(csvInhalt);
15     return new CsvKontoauszugsquelle(reader, TRENNZEICHEN, ANFUEHRUNGSZEICHEN, Mandant.AO);
16   }
```

```
17   private void AssertKontoumsaetzeVorhanden(IEnumerable<Kontoumsatz> kontoumsaetze, Tuple<DateTime,
         decimal, decimal, String, StatusType, BuchungstextType>[] erwarteteKontoumsaetze)
18   {
19     for (int i = 0; i < kontoumsaetze.Count(); i++)
20     {
21       var aktKontoumsatz = kontoumsaetze.ElementAt(i);
22       var erwKontoumsatz = erwarteteKontoumsaetze[i];
23       var kontoumsatzNr = i + 1;
24
25       Assert.That(aktKontoumsatz.DatumValuta, Is.EqualTo(erwKontoumsatz.Item1), String.Format("Der
             Kontoumsatz {0} sollte als Valuta das Datum {1} haben.", kontoumsatzNr, erwKontoumsatz.Item1));
26       Assert.That(aktKontoumsatz.EuroUmsatz, Is.EqualTo(erwKontoumsatz.Item2), String.Format("Der Betrag des
             Kontoumsatzes {0} sollte {1} betragen.", kontoumsatzNr, erwKontoumsatz.Item2));
27       Assert.That(aktKontoumsatz.OriginalUmsatz, Is.EqualTo(erwKontoumsatz.Item3), String.Format("Der
             Originalbetrag des Kontoumsatzes {0} sollte {1} betragen.", kontoumsatzNr, erwKontoumsatz.Item3));
28       Assert.That(aktKontoumsatz.Verwendungszweck, Is.EqualTo(erwKontoumsatz.Item4), String.Format("Der
             Verwendungszweck des Kontoumsatzes {0} sollte {1} lauten.", kontoumsatzNr, erwKontoumsatz.Item4));
29       Assert.That(aktKontoumsatz.Status, Is.EqualTo(erwKontoumsatz.Item5), String.Format("Der Status des
             Kontoumsatzes {0} sollte {1} lauten.", kontoumsatzNr, erwKontoumsatz.Item5));
30       Assert.That(aktKontoumsatz.Buchungstext, Is.EqualTo(erwKontoumsatz.Item6), String.Format("Der
             Buchungstext des Kontoumsatzes {0} sollte {1} lauten.", kontoumsatzNr, erwKontoumsatz.Item6));
31     }
32   }
33 [...]
34
35   [Test]
36   public void ExceptionWerfenWennCsvDateiLeerIst()
37   {
38     sut = ErzeugeKontoauszugsquelle("");
39     AssertExceptionIsEqual(FEHLERMELDUNG + "Die Datei ist leer.");
40   }
41
42   [Test]
43   public void ExceptionWerfenWennAnzahlSpaltenInZeileZuGering()
44   {
45     sut = ErzeugeKontoauszugsquelle("123456;9321049000;1;\"\";\"\";\"01.07.2013\";\"45\"");
46     AssertExceptionIsEqual(FEHLERMELDUNG + "Fehler in Zeile 1: Anzahl der Spalten <7>"
47     + " entspricht nicht der festgelegten Anzahl an Spalten <53>.");
48   }
49
50 [...]
51
52   [Test]
53   public void ExceptionWerfenWennDatumSpalteNichtGeparstWerdenKann()
54   {
55     sut = ErzeugeKontoauszugsquelle("124;342343242;1;\"\";\"\";\"01.07pp.2014\"; [...] ");
56     AssertExceptionIsEqual(FEHLERMELDUNG + "Fehler in Zeile 1:"
57     + " Problem beim Ermitteln des Kontoumsatzes: Die Zeichenfolge wurde nicht als"
58     + " gültige DateTime erkannt. Ein unbekanntes Wort beginnt bei Index 5.");
59   }
```

```
60   [Test]
61   public void ExceptionWerfenWennBuchungstextSpalteNichtInEnumGeparstWerdenKann()
62   {
63     sut = ErzeugeKontoauszugsquelle("DEU123456;DE12345678901;4;\"HEIMER V.\";\"\";\"01.09.2014\";"
64     + "\"77\";12345623;\"\";\"11782,20\";\"01.09.2014\";\"GIBSNICHT\";\"166\";\"test\"; [...]");
65     AssertExceptionIsEqual(FEHLERMELDUNG + "Fehler in Zeile 1: Problem beim Ermitteln"
66     + " des Kontoumsatzes: Der angeforderte Wert \"GIBSNICHT\" konnte nicht gefunden werden.");
67   }
68
69   [Test]
70   public void AusCsvDateiMitEinemKontoumsatzKontauszugMitEinemKontumsatzErmitteln()
71   {
72     sut = ErzeugeKontoauszugsquelle("DEU123456;DE12345678901;4;\"HEIMER V.\";\"\";\"01.09.2014\";"
73     + "\"77\";12345623\"\";\"11782,20\";\"01.09.2014\";\"GUTSCHRIFT\";\"166\";\"test\";"
74     + "\"AG – 654321, BLM Ü−werte\";\"\";0;1234321;\"NONREF\";4;11782,20;\"EUR\";\"9249\";"
75     + "\"test \";43420,46;31655,06;0;11782,20;16,80;\" TRF\";\"\";\"test\";11782,20;\"EUR\";"
76     + "\"01.09.2014\";\"EREF+UEWERT++K+4148++445566\";\"\";\"\";\"\";\"\";\"\";"
77     + "\"0+01+.000012\";\"SVWZ+UEWERT++K+4148++445566\";\"0+01+ +P+4142++4434521"
78     + "01++\";\"K+\";\"\";\"\";\"\";"\";\"\";\"EUR\";\"test\";\"test\";43420,46");
79
80     var kontoauszug = sut.ErmittleKontoauszug();
81     var erwarteteKontoauszugEigenschafen = new Tuple<Mandant, DateTime, decimal, decimal, int>(Mandant.AO,
           new DateTime(2014, 9, 1), 31655.06m, 43420.46m, 1);
82     AssertKontoauzugEigenschaftenVorhanden(kontoauszug, erwarteteKontoauszugEigenschafen);
83
84     var erwarteteKontoumsaetze = new Tuple<DateTime, decimal, decimal, String, StatusType, BuchungstextType
           >[]
85     {
86       new Tuple<DateTime, decimal, decimal, String, StatusType, BuchungstextType>(new DateTime(2014, 9, 1),
             11782.20m, 11782.20m, "EREF+UEWERT++K+4148++4455660+01+.000012SVWZ+UEWERT+"
87           + "+K+4148++4455660+01+P+4142++4434521+01++K+", StatusType.VerarbeitungAusstehend,
               BuchungstextType.GUTSCHRIFT)
88     };
89
90     AssertKontoumsaetzeVorhanden(kontoauszug.Kontoumsatz, erwarteteKontoumsaetze);
91   }
92
93 [...]
```

Listing 1: CsvKontoauszugsquelleSollte.cs

Hinweis: Bei einigen Testfällen wurde der Eingabestring verkürzt dargestellt. Die entsprechenden Stellen wurden mit einem Auslassungszeichen ([...])gekennzeichnet. Aufgrund des eingeschränkten Umfangs dieser Dokumentation können nicht alle Tests der `CsvKontoauszugsquelle` aufgeführt werden.

A.19 Listing der Klasse `CsvKontoauszugsquelle`

```
1  [...]
2
3  public class CsvKontoauszugsquelle
4  {
5    private const int ANZAHL_SPALTEN = 53
6
7    private const int SPALTE_AUSZUGSDATUM = 5;
8    private const int SPALTE_BETRAG = 9;
9    private const int SPALTE_BUCHUNGSTEXT = 11;
10   private const int SPALTE_ORIGINALBETRAG = 20;
11   private const int SPALTE_ENDSALDO = 24;
12   private const int SPALTE_ANFANGSSALDO = 25;
13   private const int SPALTE_DATUM_VALUTA = 34;
14   private const int SPALTE_VWZ_ANFANG = 35;
15   private const int SPALTE_VWZ_ENDE = 48;
16
17   private CsvParser parser;
18   private CsvConfiguration konfiguration;
19
20   private Mandant mandant;
21   private DateTime auszugsdatum;
22
23   private decimal anfangssaldo;
24   private decimal endsaldo;
25
26   public CsvKontoauszugsquelle(TextReader reader, String trennzeichen, Char anfuehrungszeichen, Mandant mandant)
27   {
28     this.mandant = mandant;
29     konfiguration = new CsvConfiguration();
30     konfiguration.Delimiter = trennzeichen;
31     konfiguration.Quote = anfuehrungszeichen;
32     konfiguration.HasHeaderRecord = false;
33     parser = new CsvParser(reader, konfiguration);
34   }
35
36   public Kontoauszug ErmittleKontoauszug()
37   {
38     try
39     {
40       var kontoumsaetze = ErmittleKontoumsaetze();
41       return new Kontoauszug(mandant, auszugsdatum, anfangssaldo, endsaldo, kontoumsaetze);
42     }
43     catch(Exception ex)
44     {
45       throw new KontoauszugImportException("Fehler beim Importieren des Kontoauszugs: " + ex.Message, ex);
46     }
47   }
48
49
```

```
50  private ICollection<Kontoumsatz> ErmittleKontoumsaetze()
51  {
52      var kontoumsaetze = new List<Kontoumsatz>();
53      long aktuelleZeile = 0;
54
55      while(true)
56      {
57          try
58          {
59              var zeile = parser.Read();
60
61              if (zeile == null)
62              {
63                  break;
64              }
65
66              if(MussZeileVerarbeitetWerden(zeile))
67              {
68                  aktuelleZeile++;
69                  kontoumsaetze.Add(ErmittleKontoumsatz(zeile));
70              }
71          }
72          catch(Exception ex)
73          {
74              throw new KontoauszugImportException(String.Format("Fehler in Zeile {0}: {1}", aktuelleZeile, ex.
                    Message), ex);
75          }
76      }
77      if(aktuelleZeile <= 0)
78      {
79          throw new KontoauszugImportException("Die Datei ist leer.");
80      }
81      return kontoumsaetze;
82  }
83
84  private Kontoumsatz ErmittleKontoumsatz(String[] zeile)
85  {
86      if(zeile.Length != ANZAHL_SPALTEN)
87      {
88          throw new KontoauszugImportException(String.Format("Anzahl der Spalten <{0}>"
89          + "entspricht nicht der festgelegten Anzahl an Spalten <{1}>.", zeile.Length, ANZAHL_SPALTEN));
90      }
91      try
92      {
93          ErmittleKontoauszugEigenschaften(zeile);
94
95          var datumValuta = ErmittleDatum(zeile[SPALTE_DATUM_VALUTA]);
96          var betrag = ErmittleBetrag(zeile[SPALTE_BETRAG]);
97          var originalBetrag = ErmittleBetrag(zeile[SPALTE_ORIGINALBETRAG]);
98
```

```
99
100         var buchungstext = ErmittleBuchungstext(zeile[SPALTE_BUCHUNGSTEXT]);
101         var verwendungszweck = ErmittleVerwendungszweck(zeile);
102
103         StatusType status = StatusType.VerarbeitungAusstehend;
104
105         return new Kontoumsatz(datumValuta, betrag, originalBetrag, verwendungszweck, status, buchungstext);
106     }
107     catch(Exception ex)
108     {
109         throw new KontoauszugImportException("Problem beim Ermitteln des Kontoumsatzes: " + ex.Message);
110     }
111   }
112
113   private void ErmittleKontoauszugEigenschaften(String[] zeile)
114   {
115     auszugsdatum = ErmittleDatum(zeile[SPALTE_AUSZUGSDATUM]);
116
117     anfangssaldo = ErmittleBetrag(zeile[SPALTE_ANFANGSSALDO]);
118     endsaldo = ErmittleBetrag(zeile[SPALTE_ENDSALDO]);
119   }
120
121   private String ErmittleVerwendungszweck(String[] zeile)
122   {
123     return  zeile.Skip(SPALTE_VWZ_ANFANG)
124                  .Take(SPALTE_VWZ_ENDE - SPALTE_VWZ_ANFANG)
125                  .Aggregate((verwendungszweck, spaltenInhalt) => (verwendungszweck + spaltenInhalt))
126                  .Replace(" ", "");
127   }
128
129   private decimal ErmittleBetrag(String betrag)
130   {
131     return Convert.ToDecimal(betrag, new CultureInfo("de-DE"));
132   }
133
134   private DateTime ErmittleDatum(String datum)
135   {
136     return DateTime.Parse(datum);
137   }
138
139   private BuchungstextType ErmittleBuchungstext(String buchungstext)
140   {
141     return (BuchungstextType)Enum.Parse(typeof(BuchungstextType), buchungstext);
142   }
143
144   private bool MussZeileVerarbeitetWerden(String[] zeile)
145   {
146     return !IstZeileLeer(zeile);
147   }
148
```

```
149   private bool IstZeileLeer(String[] zeile)
150   {
151     return zeile.Aggregate((zeilenInhalt, spaltenInhalt) => (zeilenInhalt + spaltenInhalt)).Trim().Equals("");
152   }
153 }
154
155  [...]
```

Listing 2: CsvKontoauszugsquelle.cs

A.20 Screenshot der Anwendung

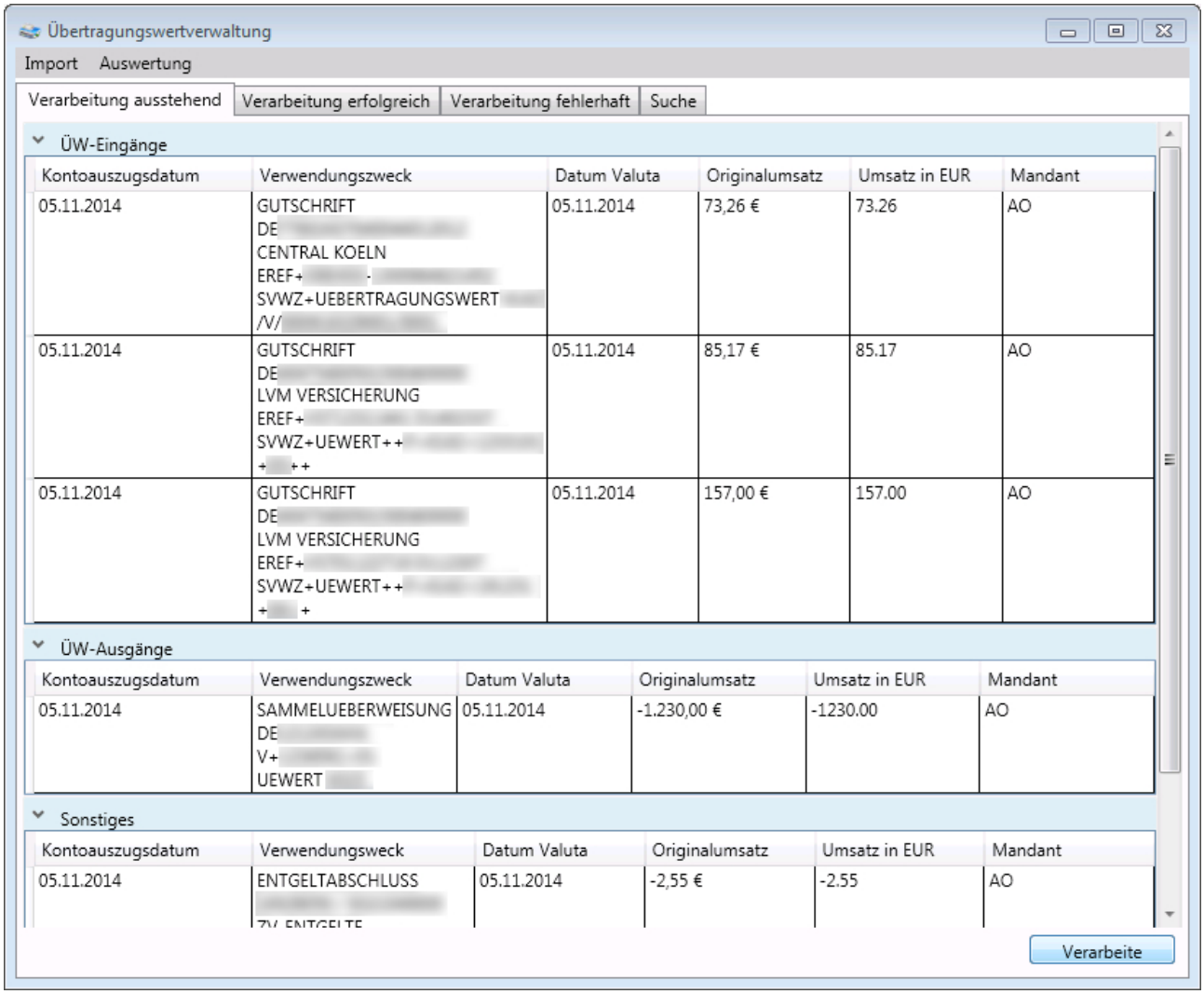


Abbildung 10: Screenshot der Anwendung

A.21 Auszug aus der Benutzerdokumentation

In dem folgendem Auszug aus dem Benutzerhandbuch wird Schritt für Schritt erläutert, wie ein neuer Kontoauszug in die Übertragungswertverwaltungsanwendung importiert werden kann.

Kontoauszug importieren

Um einen neuen Kontoauszug importieren zu können, muss zunächst das Importfenster geöffnet werden. Dies kann über den Punkt „Import“ der Menüleiste der Anwendung aufgerufen werden, indem man dort den Menüunterpunkt „Kontoauszug importieren“ auswählt (siehe Abbildung 11).

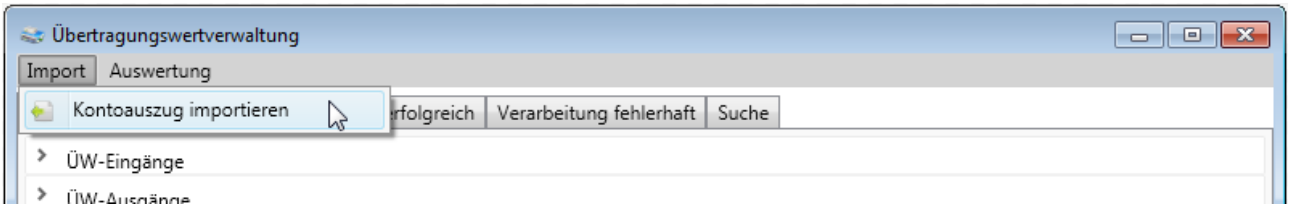


Abbildung 11: Importfunktion im Import-Menü des Hauptfensters

Nachdem sich das Importfenster geöffnet hat, kann die gewünschte CSV-Datei, in der die Daten des zu importierenden Kontoauszugs enthalten sind, ausgewählt werden. Dafür kann entweder die Durchsuchen-Funktion benutzt werden oder man kann den Pfad zur gewünschten Datei direkt in das dafür vorgesehene Textfeld eingegeben. Anschließend kann der Import des Kontoauszugs durch Betätigen des Buttons „Importieren“ gestartet werden (siehe Abbildung 12), über den Button „Abbrechen“ kann der Importvorgang abgebrochen werden.

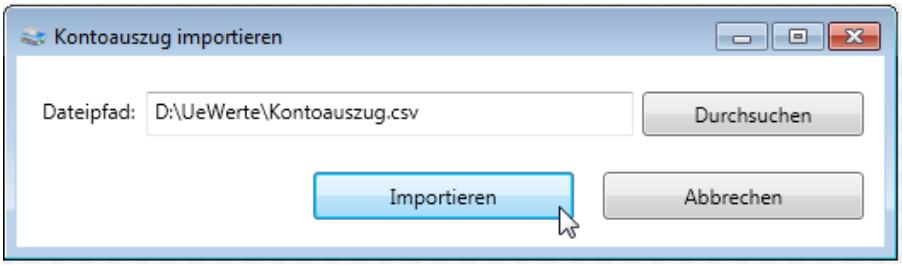
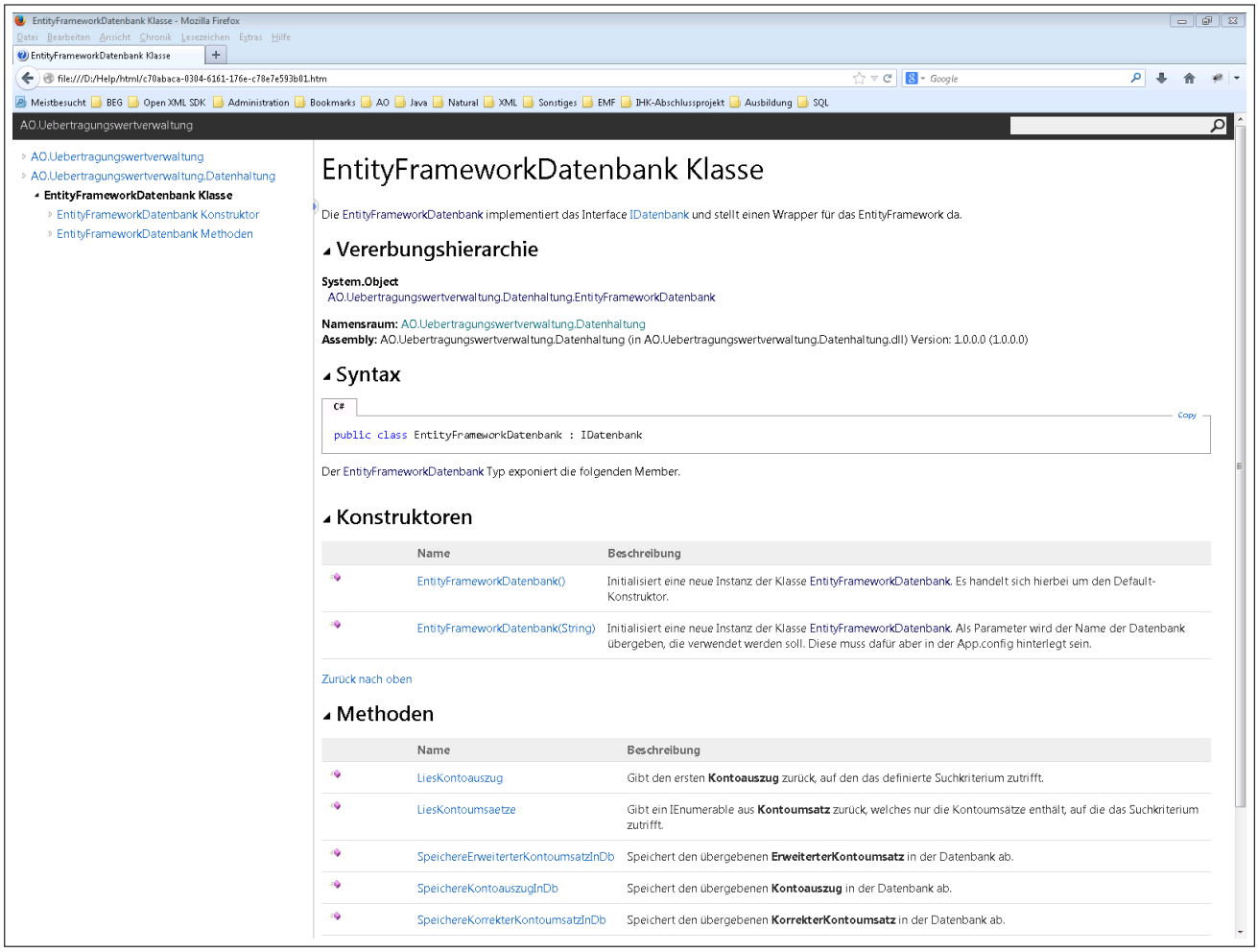


Abbildung 12: Fenster „Kontoauszug importieren“

[...]

A.22 Entwicklerdokumentation



A.23 Klassendiagramm des Domänenmodells

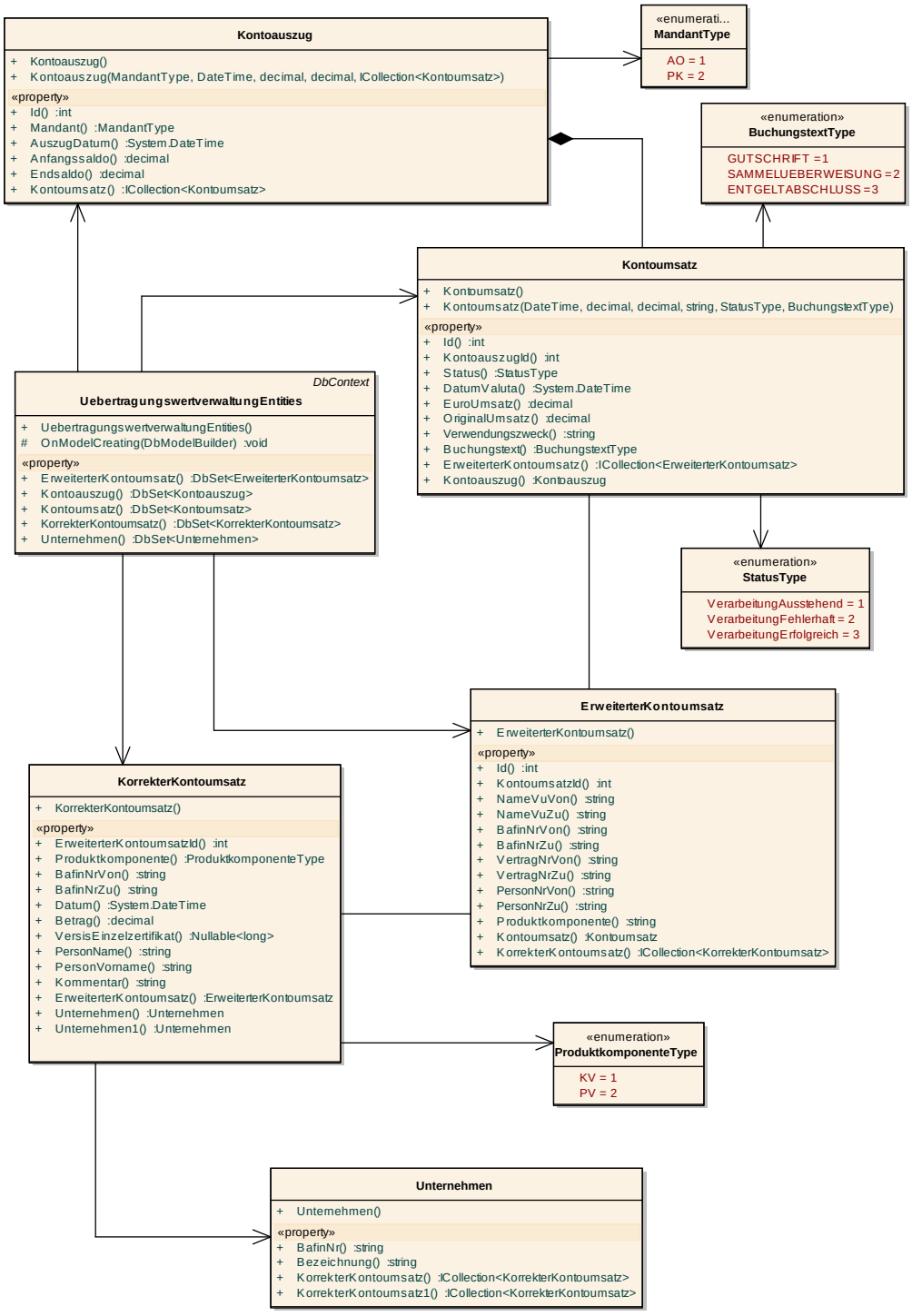


Abbildung 13: Klassendiagramm des Domänenmodells