Abschlussprüfung Sommer 2017

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

Globales Aktualisieren von Dokumenten

Computergestützte Betriebsprüfung - Abschluss & Dokumente

Abgabetermin: Berlin, den 02.06.2017

Prüfungsbewerber:
Guido Eckelt
Boddinstraße 30
12053 Berlin

Ausbildungsbetrieb:
Deutsche Rentenversicherung Bund
Ruhrstraße 2
10704 Berlin

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

DRV Bund	Deutsche Rentenversicherung Bund
CBP	Computergestützte Betriebsprüfung
CBP-AD	Computergestütze Betriebsprüfung - Abschluss & Dokumente
CBP-NB	Computergestütze Betriebsprüfung - Nachberechnung
VB	Visual Basic
DLL	Dynamic Link Library
ERM	Entity-Relationship-Modell
UML	Unified Modeling Language
GUI	Grafische Benutzeroberfläche
CI	Corporate Identity

1 Einleitung

1.1 Projektumfeld

Die Deutsche Rentenversicherung Bund (DRV Bund) ist ein bundesweit tätiger Träger der gesetzlichen Rentenversicherung in der Bundesrepublik Deutschland mit ca. 17.000 Mitarbeitern.
Zum Aufgabenfeld gehören:

- Bearbeitung von Rentenanträgen und Auszahlung von Renten
- Überprüfung von Sozialabgaben auf Richtigkeit
- Beratung zu gesetzlichen Pflichten und Rechten
- Bearbeitung von Rehabilitationsanträgen und Inspektion der Rehabilitationseinrichtungen

Für die Überprüfung von Sozialabgaben entwickelt die IT-Abteilung der DRV Bund verschiedene Anwendungen für alle Träger der Deutschen Rentenversicherung, um diesen Prozess zu vereinfachen.

Die Betriebsprüfung übernimmt die Nachprüfung der Sozialabgaben für bundesweit oder international tätige, in Deutschland gemeldete Betriebe. Die Betriebsprüfer müssen dafür viele Berechnungen durchführen und verschiedenste Dokumente, Anlagen und Schreiben(nachfolgend nur noch Dokumente genannt) erstellen.

Die Computergestütze Betriebsprüfung - Abschluss & Dokumente (CBP-AD) ist eine Desktopanwendung mit der jeweilig benötigte Dokumente für Betriebsprüfungen erzeugt werden können. Diese Dokumente basieren auf Datenquellen der Desktopanwendung Computergestütze Betriebsprüfung - Nachberechnung (CBP-NB).

1.2 Projektziel

Die Betriebsprüfer erstellen und bearbeiten Dokumente, die unter anderem mit Daten aus Berechnungen der Anwendung CBP-NB befüllt werden. Wenn Daten verändert werden, sind diese Dokumente in einem „asynchronen“ Zustand und müssen vor Weiterverwendung aktualisiert werden. Nähere Erläuterungen befinden sich im Abschnitt 3.2: Ist-Analyse.

Im Hauptmenü der CBP-AD soll ein neuer Menüeintrag bereitgestellt werden, dessen Kommando einen Aktualisierungsprozess anstößt, der alle Dokumente auf Asynchronität prüft und anschließend veraltete aktualisiert.

Für die verschiedenen Dokumenttypen gibt es zurzeit noch unterschiedliche Vorgehensweisen, wie die jeweiligen Dokumente neu erzeugt werden. Für die Funktionalität „Globales Aktualisieren“ sollen nun alle Dokumenttypen auf eine einheitliche Vorgehensweise umgebaut werden.

Erzeugungsstrukturen für einige Dokumenttypen berechnen ihren Fortschritt eigenständig und geben diesen in einer eigenen Oberfläche aus. Für diese soll eine Möglichkeit der Unterdrückung dieser Fortschrittsausgabe implementiert werden, damit der Aktualisierungsprozess „Globales Aktualisieren“ dies einheitlich für alle Dokumenttypen ausgeben kann.

1.3 Projektbegründung

Durch diese Erweiterung wird eine Vereinheitlichung der Dokumentenaktualisierung erreicht, die zugleich eine erhebliche Vereinfachung für den Anwender mit sich bringt.

1.4 Projektschnittstellen

Die Datenquellen werden über Schnittstellen in DLLs der CBP-NB der CBP-AD zur Verfügung gestellt.

Die Benutzer der Anwendung sind die Betriebsprüfer der Deutschen Rentenversicherung.

1.5 Projektabgrenzung

Dieses Projekt zur Erweiterung der CBP-AD ist unabhängig von der Entwicklung der CBP-NB, da nur bereits festgelegte Schnittstellen zum Datenaustausch benutzt werden und keine Änderung dieser notwendig sind.

2 Projektplanung

2.1 Projektphasen

Die Projektphase begann am 13.03.2017 und endete am 02.06.2016. Die tägliche Arbeitszeit betrug 7 Stunden 48 Minuten und zuzüglich 30 Minuten Mittagspause. Die Projektarbeit fand nicht durchgängig statt, da betriebsinterne Aufgaben und Ereignisse berücksichtigt werden mussten.

Projektphase	**Teilzeit**	**Gesamtzeit**
1. Analyse		5 h
1.1 Fachgespräch mit Mitarbeiter der IT-Abteilung	1 h	
1.2 Analyse des Ist-Zustands	3 h	
1.3 Wirtschaftlichkeitsanalyse	1 h	
2. Entwurf		5 h
2.1 Erstellen eines UML-Klassendiagramms der verwendeten Klassen	3 h	
2.2 Erstellen eines UML-Sequenzdiagramms des Hauptprozesses	2 h	
3. Implementierungsphase		45 h
3.1 Aktualisierungsprozess „Globales Aktualisieren“	25 h	
3.2 Umbau der Dokumentenerzeugung	10 h	
3.3 Fortschrittsausgabe vereinheitlichen	10 h	
4. Qualitätssicherung		5 h
4.1 Komponententests	3 h	
4.2 Abnahme der IT-Abteilung	2 h	
5. Dokumentation		10 h
5.1 Projektdokumentation	6 h	
5.2 Programmdokumentation	4 h	
Gesamt		**70 h**

Tabelle 1: Zeitplanung

Eine detailliertere Zeitplanung findet sich im Anhang A.1: Detaillierte Zeitplanung auf Seite i.

2.2 Ressourcenplanung

Hardware

- Büroarbeitsplatz mit Tisch, Stuhl, Stromanschlüsse
- Arbeitsmaschine 1 mit Windows7
- Arbeitsmaschine 2 mit Kartenleser und Zugang zum Entwicklungsnetz der DRV Bund

Software

- Visual Studio Professional 2013 + .NET-Framework (mindestens v2.0)
- MiKTeX - Distribution des Softwarepakets LaTeX
- TeXStudio - Entwicklungsumgebung für LaTeX
- microTool inStep - Projektverwaltungstool für Arbeitsmaschine 2

Personal

- Projektbetreuer zur Unterstützung

2.3 Entwicklungsprozess

Die ausgewählte Vorgehensweise ist das Wasserfall-Modell[1]. Es ist konventionell vorgesehen, dass alle Schritte im Wasserfall-Modell sequentiell zu bearbeiten sind, d. h. Schritte dürfen übersprungen werden. Nicht erfolgreicher Abschluss eines Schrittes bedeutet ein Neustart oder Abbruch des Projektes. In der IT-Branche wird jedoch meist mit dem erweiterten Wasserfall-Modell gearbeitet, welches Rücksprünge zum jeweils vorhergehenden Schritt erlaubt.

1. **Systemanforderungen:**
   Alle Anforderungen, die selbst nicht direkt das Software-Produkt betreffen, werden zunächst festgelegt. Dazu zählen:
   - Preis
   - Verfügbarkeit
   - Sicherheitsaspekte
   - Dokumentation
2. **Softwareanforderungen:**
   Alle Anforderung an die Software selber werden definiert. Jegliche Funktionen, Interaktionen und Besonderheiten werden konkretisiert, so dass sich aus den Systemanforderungen und Softwareanforderungen das Lastenheft ergibt.
3. **Analyse:**
   Anforderungen aus Lastenheft und Ist-Zustand der Situation werden analysiert, so dass diese in ein Pflichtenheft umformuliert werden können. Die Wirtschaftlichkeit eines Projektes wird ebenfalls hier geprüft.

[1] Wasserfall-Modell nach Dr. Winston W. Royce

4. **Entwurf:**
   Das Datenmodell, die Architektur und die Schnittstellen zu anderen Anwendungen werden herausgearbeitet. Zwischenergebnisse können sein:

   - Entity-Relationship-Modell (ERM)
   - UML-Diagramme (Klassendiagramm, Sequenzdiagramm, Anwendungsfalldiagramm usw.)
   - Mockups zur GUI
   - Schnittstellen-Verzeichnis

5. **Implementierung:**
   Umsetzung der Funktionalitäten nach Pflichtenheft und Entwurf in eine lauffähige Anwendung.

6. **Test/Qualitätssicherung:**
   Es wird nach der Implementierungsphase die Software auf Fehler, Schwachstellen und Unstimmigkeiten überprüft. Folgende Testmethoden sind Beispiele Qualitätssicherung in der IT-Branche:

   - Komponententests (Unit-Test)
   - Modultests
   - Systemtests
   - Integrationstests

7. **Inbetriebnahme:**
   Nach erfolgreichen Bestehen der Qualitätssicherung kann die Anwendung abgenommen werden und in Produktion gehen.

3 Analysephase

3.1 Auszug aus dem Fachkonzept

Folgender Auszug des Fachkonzeptes ist durch die IT-Abteilung erstellt worden.

1 Auftrag

Auszug aus Afo:

Die Nachberechnungsanlagen für alle Prüfgebiete(GSV, UV, KSA, WGS) sowie ggf. vorhandene Insolvenztabellen sollen über einen Befehl erzeugt und aktualisiert werden. Ferner sollen auch „Externe Dateien“ nach dem Auslösen der „Globalen Aktualisierung“ im Dokumentenbrowser angezeigt werden.

2 Fachliche Anforderungen

Bei Aufruf der Funktion ist zu prüfen, ob zu aktualisierende Anlagen existieren. Das ist der Fall bei:

- asynchronen Anlagen oder
- Anlagen (asynchron/synchron) ohne Berechnungen in CBP NB

Eine Aktualisierung von Anlagen ohne Berechnungen in CBP NB ist für einige Anlagen bisher nicht erfolgt und wird jetzt einheitlich für alle Anlagen durchgeführt. Unter folgenden Bedingungen war bisher keine Aktualisierung möglich:

- Es besteht die entsprechende Prüfrelevanz nicht mehr.
- Die Prüfart ist ungleich Sonder- oder Mischprüfung.
- Es liegen keine elektronischen Daten zur Prüfung vor.
- Ein Prüfabschluss ohne Feststellung (PaoF) wurde begonnen.

Sofern keine zu aktualisierenden Anlagen existieren, ist der Anwender wie folgt zu informieren:

„es liegen keine zu aktualisierenden Anlagen vor.
Der Ordner „Externe Dateien“ wurde aktualisiert“

es wird dann nur der Ordner „Externe Dateien“ aktualisiert.

2.1 Hauptmenü erweitern

Für den Aufruf der Funktion „Globales Aktualisieren“ sind zu ergänzen:

- Menüeintrag im Menü „Dokumente“
  - Text: „Globales Aktualisieren“
  - Position: nach „Bearbeiten...“
  - Shortcut: „STRG+R“

▪▪▪

Abbildung 1: Auszug aus dem Fachkonzept

3.2 Ist-Analyse

Die Betriebsprüfer müssen die Dokumente einzeln über den Dokumentenbaum aktualisieren. Dies kann sehr aufwendig sein, da manche Berechnungen zu Änderungen in mehreren Dokumenten führen. Das bedeutet man muss teilweise den kompletten Dokumentenbaum, siehe Anhang A.4: Screenshots der Anwendung auf Seite iii, durchgehen, um alle Dokumente aktualisieren zu können.

3.3 Wirtschaftlichkeitsanalyse

3.3.1 „Make or Buy“-Entscheidung

Da die Entwicklung der CBP-AD ein internes Projekt der DRV Bund und nur eine Funktionserweiterung ist, lässt sich kein fertiges Produkt finden, dass alle Anforderungen, vor allem fachliche, erfüllt. Daher wird dieses Projekt in Eigenentwicklung von der IT-Abteilung der DRV Bund umgesetzt.

3.3.2 Projektkosten

Die Kosten für die Durchführung des Projekts setzen sich sowohl aus Personal-, als auch aus Ressourcenkosten zusammen. Laut Tarifvertrag[2] verdient ein Auszubildender im dritten Lehrjahr pro Monat 1.014,02 € Brutto.

$$8\ \text{h/Tag} \cdot 220\ \text{Tage/Jahr} = 1.760\ \text{h/Jahr} \tag{1}$$

$$1.014{,}02\,€/\text{Monat} \cdot 12{,}9\ \text{Monate/Jahr} \approx 13.080{,}86\,€/\text{Jahr} \tag{2}$$

$$\frac{13.080{,}86\,€/\text{Jahr}}{1.760\ \text{h/Jahr}} \approx 7{,}43\,€/\text{h} \tag{3}$$

Es ergibt sich also ein Stundenlohn von 7,43 €. Die Durchführungszeit des Projekts beträgt 70 Stunden. Für die Nutzung von Ressourcen[3] wird ein pauschaler Stundensatz von 15 € angenommen. Für die anderen Mitarbeiter wird pauschal ein Stundenlohn von 25 € angenommen. Eine Aufstellung der Kosten befindet sich in Tabelle 2. Es ergibt sich daraus insgesamt 1.730,10 €.

Vorgang	Zeit	Kosten pro Stunde	Kosten
Entwicklungskosten	70 h	7,43 € + 15 € = 22,43 €	1.570,10 €
Fachgespräch	3 h	25 € + 15 € = 40 €	120,00 €
Abnahmetest	1 h	25 € + 15 € = 40 €	40,00 €
			1.730,10 €

Tabelle 2: Kostenaufstellung

[2] Tarifvertrag des öffentlichen Dienstes
[3] Räumlichkeiten, Arbeitsplatzrechner etc.

3.3.3 Amortisationsdauer

Nach Umfragen bei den Betrieberprüfern ist eine Zeiteinsparung von 10 Minuten pro Tag wahrscheinlich. Daraus ergibt sich für jeden der 4.000 Betriebsprüfer und 220 Arbeitstagen im Jahr eine gesamte Zeiteinsparung von

$$4.000 \cdot 220 \text{ Tage/Jahr} \cdot 10 \text{ min/Tag} = 8.800.000 \text{ min/Jahr} \approx 146.667 \text{ h/Jahr} \tag{4}$$

Dadurch ergibt sich eine jährliche Einsparung von

$$146.667\text{h} \cdot (25 + 15)\,€/\text{h} = 5.866.680\,€ \tag{5}$$

Die Amortisationsdauer beträgt also $\frac{1.730{,}10\,€}{5.866.680\,€/\text{Jahr}} \approx 0,000.3$ Jahre ≈ 3 Stunden.

3.4 Zwischenstand

Tabelle 3 zeigt den Zwischenstand nach der Analysephase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Fachgespräch mit Mitarbeiter der IT-Abteilung	1 h	1 h	
2. Analyse des Ist-Zustands	3 h	1 h	+2 h
2. Wirtschaftlichkeitsanalyse	1 h	1 h	

Tabelle 3: Zwischenstand nach der Analysephase

4 Entwurfsphase

4.1 Geschäftslogik

Das Kommando stellt die Schnittstelle zur Oberfläche zum Aufrufen der Funktionalität dar. Dieses instantiiert dann seinen Prozess, der dann wiederum seinen Ueberarbeiter instantiiert. Der Ueberarbeiter ist das zentrale Logikgerüst dieser Funktonalität. Er delegiert Prüfungs- und Aktualisierungsaufgaben an seine internen Bearbeiter, die dann je nach Aufgabe diese selbst implementieren oder an entsprechende Klasse weitergeben. Die Ausführung und Fortschrittsausgabe dieses Prozess ist in den Klassen JobgruppeAllgemein und JobAllgemein gekapselt.

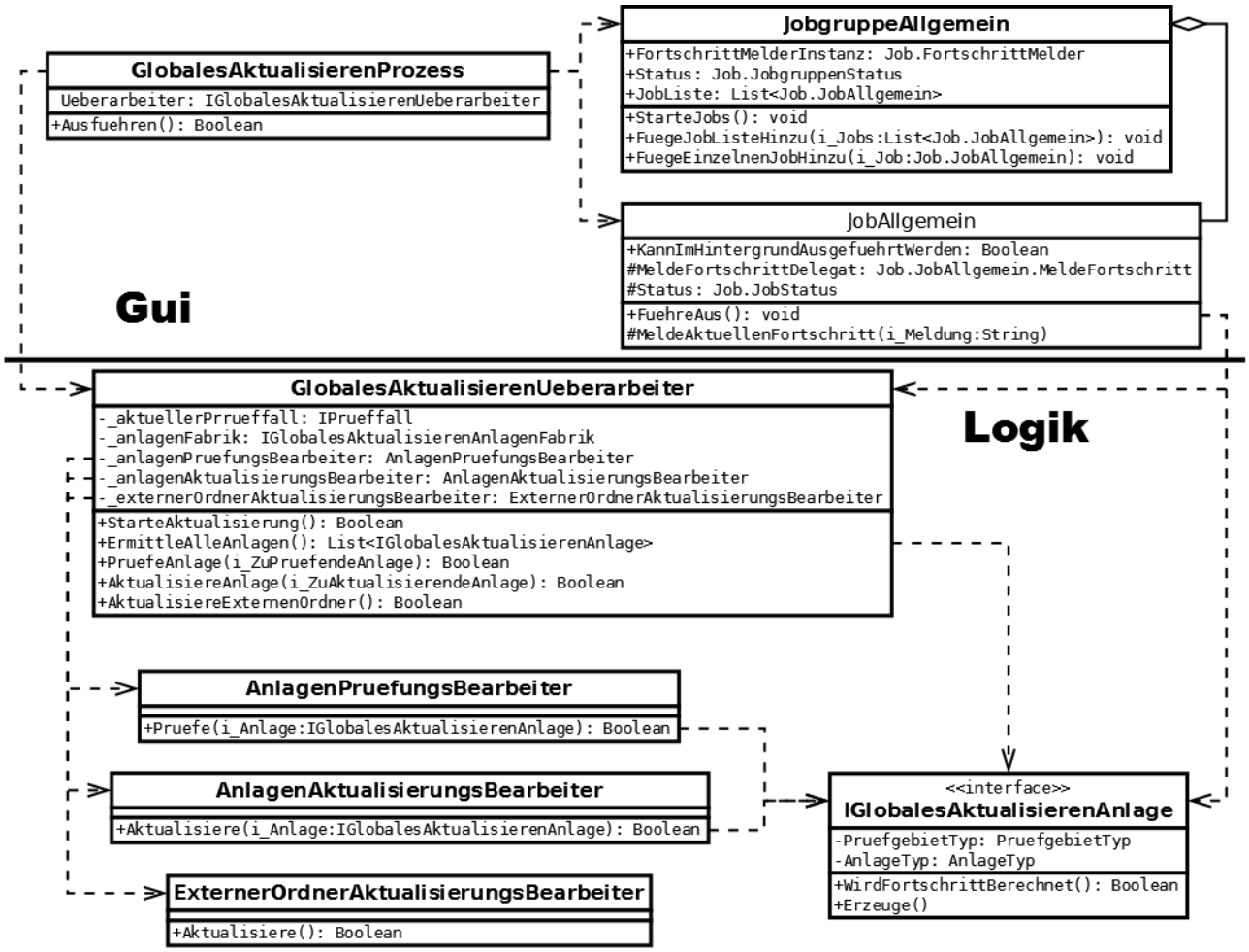

Abbildung 2: Ausschnitt des Klassendiagramms

Das vollständige Klassendiagramm kann im Anhang A.2: Klassendiagramm auf Seite ii eingesehen werden. Der Ablauf des Prozesses kann im Anhang A.3: Sequenzdiagramm auf Seite ii eingesehen werden.

4.2 Zielplattform

CBP-AD ist als Desktopanwendung in VB.NET, wie in 1.1: Projektumfeld erwähnt, auf Basis des .NET-Framework Version 2.0 implementiert. Sie wird als x86-Anwendung für Windows-7 entwickelt.

4.3 Benutzeroberfläche

Die GUI der CBP-AD ist, wie in 1.1: Projektumfeld erwähnt, als Windows-Forms-Oberfläche realisiert. Das Aussehen der Steuerelemente ist durch Corporate Identity (CI) vordefiniert. Für das Einbinden in des neuen Menüeintrag, wie in 1.2: Projektumfeld erwähnt, ist die Instanziierung eines Objekt der Klasse „GlobalesAktualisieren“ beim Starten des Programmes und ein Eintrag in der Konfigurationsdatei des Hauptmenüs notwendig. Die Konfigurationsdatei bestimmt die Position der einzelnen Menüeinträge im Hauptmenü. Für diese Funktionalität sind keine weiteren Änderung an der GUI nötig, da die Oberfläche zur Fortschrittsausgabe schon in bereits bestehenden Klassen implementiert wurde.

4.4 Datenmodell

Die Funktionalität „Globales Aktualisieren von Dokumenten“ beinhaltet keine Speicherung von Entitäten, daher ist kein neues Datenmodell erforderlich.

4.5 Maßnahmen zur Qualitätssicherung

Die Funktionalität „Globales Aktualisieren“ wird durch einen Komponententest auf korrekte Ausführung mit der Entwicklungsumgebung geprüft. Für die Einführung in die nächste Release-Version der Computergestütze Betriebsprüfung - Abschluss & Dokumente werden nochmal Verbundtests, bei denen alle Funktionalitäten in Verbindung mit den anderen Anwendungen der CBP geprüft werden, durch die Fachabteilung durchgeführt.

4.6 Zwischenstand

Tabelle 4 zeigt den Zwischenstand nach der Entwurfsphase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Erstellen eines UML-Klassendiagramms	3 h	3 h	
2. Erstellen eines UML-Sequenzdiagramms	2 h	2 h	

Tabelle 4: Zwischenstand nach der Entwurfsphase

5 Implementierungsphase

5.1 Implementierung der Geschäftslogik

Der Prüfungsalgorithmus zur Bestimmung der Asynchronität der Dokumente ist in der Klasse „AnlagenPruefungsBearbeiter“ implementiert. Jede Aktualisierungslogik ist in der Methode „Erzeuge“ der jeweiligen konkreten IGlobalesAktualisierenAnlage-Klasse implementiert.

5.2 Verwendete Entwurfsmuster

Für die Instanziierung der einzelnen konkreten Objekte der Schnittstelle „IGlobalesAktualisieren-Anlage“ verwendete ich das Fabrik-Entwurfsmuster, welches nach dem Schema in der Abbildung 3 angewendet wird.

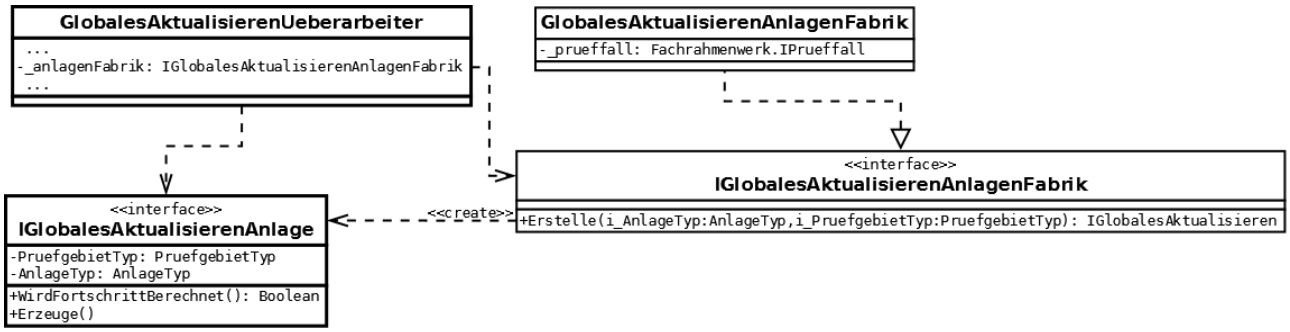

Abbildung 3: Klassendiagramm zur Verwendung des Farbik-Entwurfsmusters

5.3 Zwischenstand

Tabelle 5 zeigt den Zwischenstand nach der Implementierungsphase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Aktualisierungsprozess „Globales Aktualisieren“	25 h	25 h	
2. Umbau der Dokumentenerzeugung	10 h	10 h	
3. Fortschrittsausgabe vereinheitlichen	10 h	10 h	

Tabelle 5: Zwischenstand nach der Implementierungsphase

6 Abnahmephase

6.1 Komponententest

Der in 4.5 beschriebene Komponententest befindet sich im Anhang A.5: Testfälle auf Seite v. Die Ausführung dieses Tests führt zu folgendem Resultat.

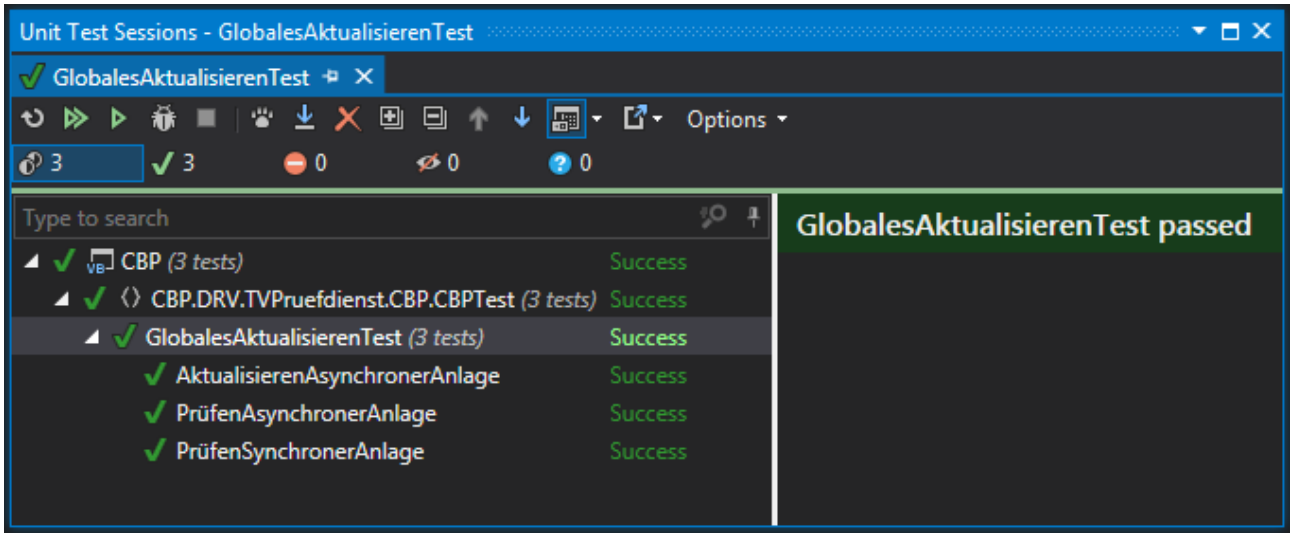

Abbildung 4: Ausführung des Komponententests

6.2 Abnahme der IT-Abteilung

Bei erfolgreichen Komponententests prüft ein zweiter Mitarbeiter mittels Code-Review, ob auffällige Stellen, Flüchtigkeitsfehler oder ähnliches vorliegen. Bei vorläufiger Abnahme ist die Erweiterung soweit zum nächsten sogenannten Verbundtest[4] in die Anwendung integriert zu werden. Dann erst kann komplett bestätigt werden, ob sich alle Funktionen der Anwendung weiterhin fehlerfrei ausführen lassen und sich an die Richtlinien zur Barrierefreiheit und Usability gehalten wird.

6.3 Zwischenstand

Tabelle 6 zeigt den Zwischenstand nach der Abnahmephase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Komponententests	3 h	3 h	
1. Abnahme der IT-Abteilung	2 h	2 h	

Tabelle 6: Zwischenstand nach der Abnahmephase

[4] System- und Benutzeroberflächentests aller Programme durch die Fachabteilung

7 Dokumentation

Die Entwicklerdokumentation zum Quellcode ist mittels VB-Doc realisiert.

Die Benutzer bekommen bei Neuerungen an der Anwendung einen Hinweis auf ein durch die Fachabteilung erstelltes Dokument, auf dem alle geänderten und neuen Funktionen verzeichnet sind und erläutert werden.

7.1 Zwischenstand

Tabelle 7 zeigt den Zwischenstand nach der Dokumentation.

Vorgang	Geplant	Tatsächlich	Differenz
1. Erstellen der Programmdokumentation	4 h	4 h	
2. Erstellen der Projektdokumentation	6 h	8 h	+2 h

Tabelle 7: Zwischenstand nach der Dokumentation

8 Fazit

8.1 Soll-/Ist-Vergleich

Wie in Tabelle 8 zu erkennen ist, konnte die Zeitplanung bis auf wenige Ausnahmen eingehalten werden. Die Analysephase brauchte weniger Zeit als geplant, weil die IT-Abteilung schon bereits den größten Teil der fachlichen Analyse vorbereitet hatte. Das Erstellen der Dokumentation hingegen brauchte etwas mehr Zeit als geplant, da die Verwendung von LateX zu einigen Komplikation geführt hatte.

Phase	Geplant	Tatsächlich	Differenz
Analysephase	5 h	3 h	-2 h
Entwurfsphase	5 h	5 h	
Implementierungsphase	45 h	45 h	
Qualitätssicherung	5 h	5 h	
Dokumentation	10 h	12 h	+2 h
Gesamt	70 h	70 h	

Tabelle 8: Soll-/Ist-Vergleich

8.2 Lessons Learned

Durch Projekte wie dieses wird einem erst bewusst, wie wichtig Anforderungen und Spezifikationen sind, da sie zum Teil die einzigen Anhaltspunkte zur Feststellung des Entwicklungsfortschritts sind.

8.3 Ausblick

Nach erfolgreichem Abschließen der Verbundtests können die Anwendungen in Produktion gehen.

Es sind keine Erweiterungen der Funktionalität geplant.

Literaturverzeichnis

Dr. Winston W. Royce

Dr. Winston W. Royce: *Managing the developement of large software systems*. https://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf, Abruf: 17.01.2017

A Anhang

A.1 Detaillierte Zeitplanung

Analysephase		**5 h**
1. Fachgespräch mit Mitarbeiter der IT-Abteilung	1 h	
2. Analyse des Ist-Zustands	3 h	
3. Wirtschaftlichkeitsanalyse	1 h	
Entwurfsphase		**5 h**
1. Erstellen eines UML-Klassendiagramms der verwendeten Klassen	3 h	
2. Erstellen eines UML-Sequenzdiagramms des Hauptprozesses	2 h	
Implementierungsphase		**45 h**
1. Aktualisierungsprozess „Globales Aktualisieren“	25 h	
1.1. Allgemeine Logik des Prozesses	10 h	
1.2. Verallgemeinertes Interface für Anlagen,Dokumenten etc.	5 h	
1.3. Aufrufen der jeweiligen Dokumentenerzeugungsprozesse	10 h	
2. Umbau der Dokumentenerzeugung	10 h	
3. Fortschrittsausgabe vereinheitlichen	10 h	
Qualitätssicherung		**5 h**
1. Komponententests	3 h	
2. Abnahme der IT-Abteilung	2 h	
Dokumentation		**10 h**
1. Projektdokumentation	6 h	
2. Programmdokumentation	4 h	
Gesamt		**70 h**

A.2 Klassendiagramm

Die grün umrandeten Klassen sind verwendete, bereits existierende Klassen der Code-Basis. Alle schwarz umrandeten Klassen sind durch dieses Projekt entstanden.

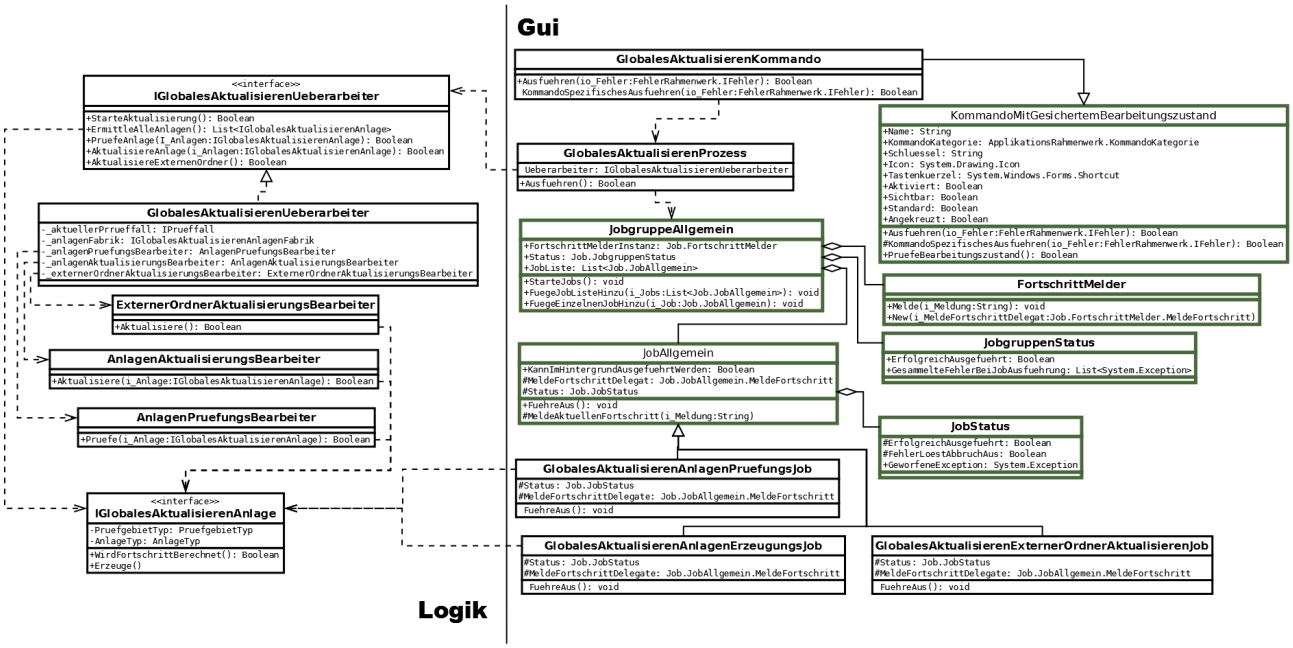

Abbildung 5: Vollständiges Klassendiagramm

A.3 Sequenzdiagramm

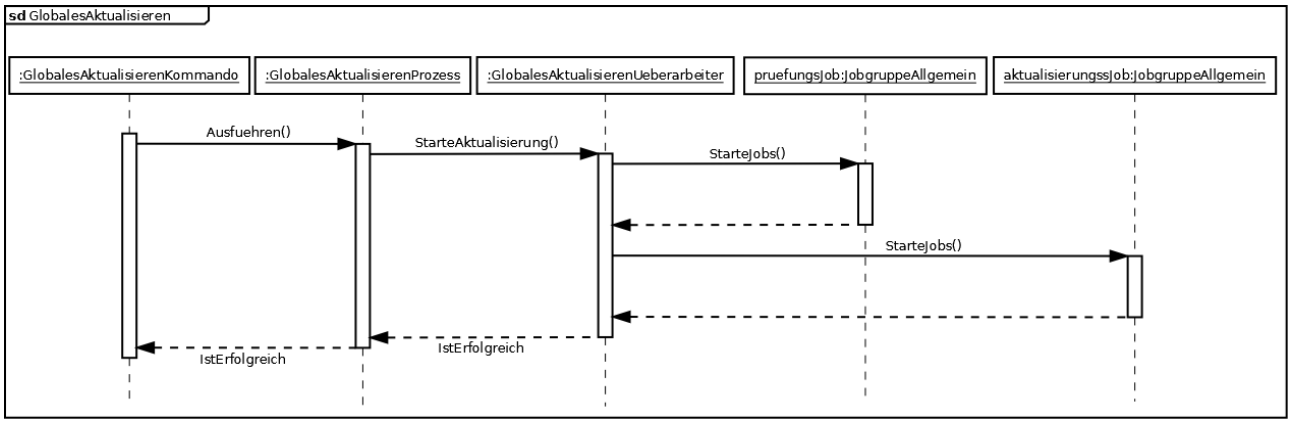

Abbildung 6: Sequenzdiagramm

A.4 Screenshots der Anwendung

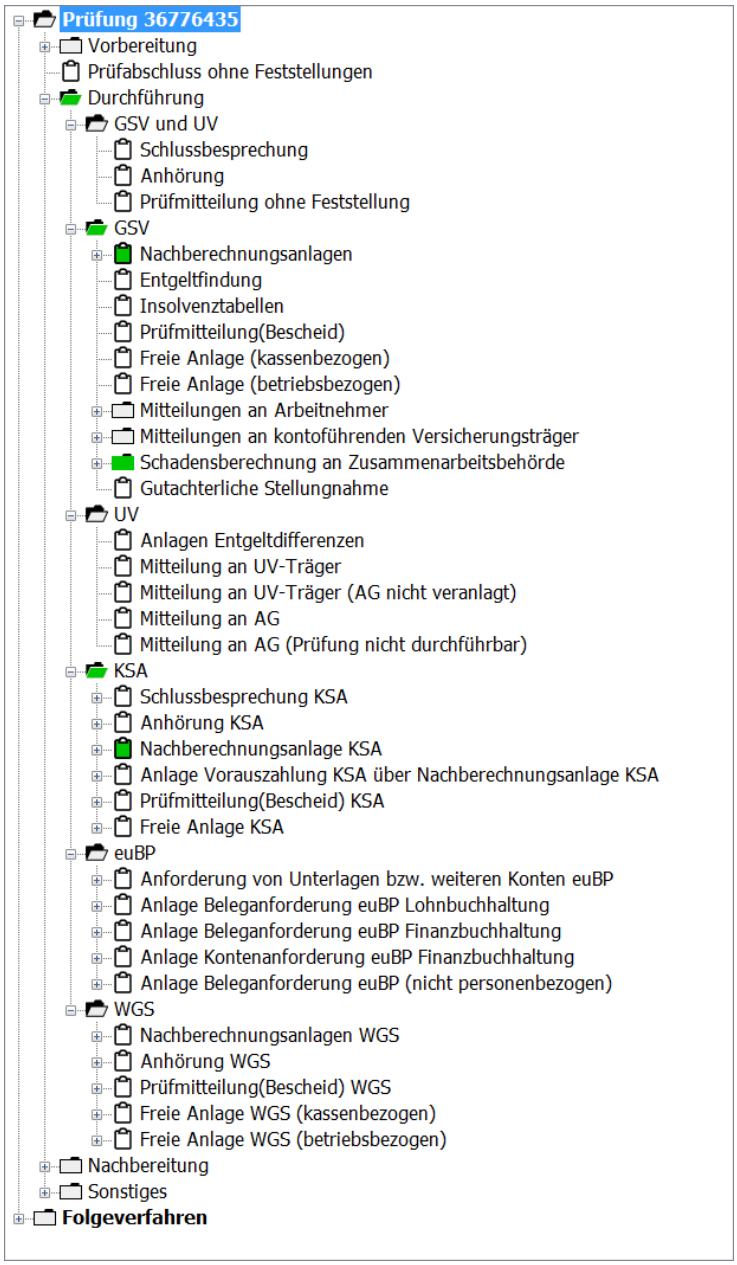

Abbildung 7: Dokumentenbaum mit den verschiedenen Dokumenttypen sortiert nach Prüfgebieten

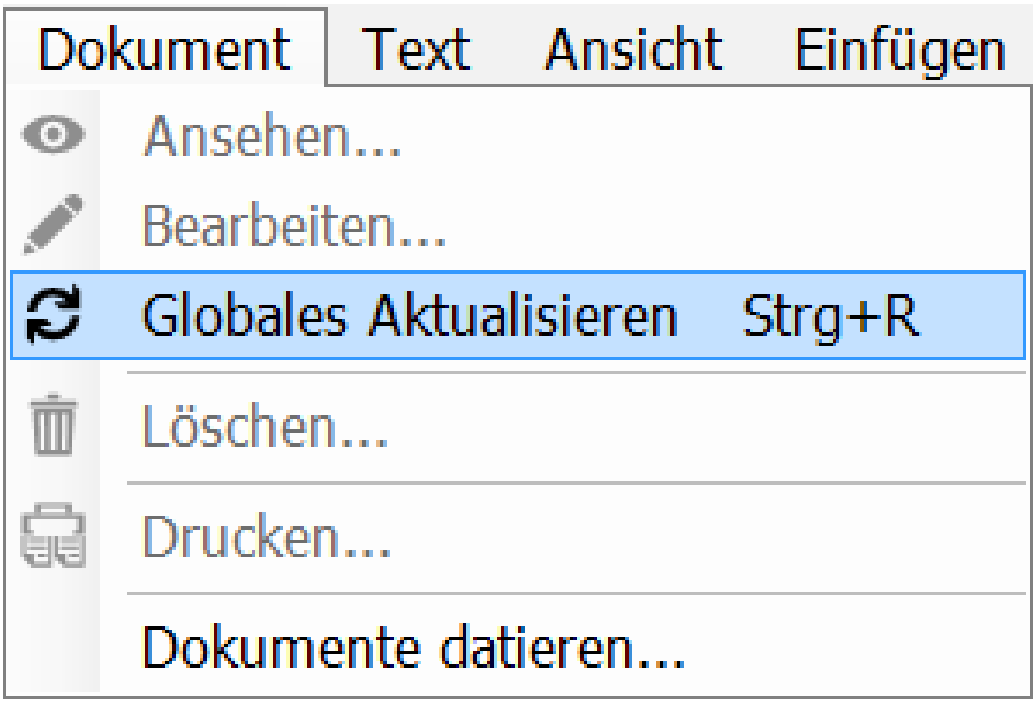

Abbildung 8: Menüeintrag zum Anstoß des Globalen Aktualisierens

A.5 Testfälle

```
Imports DRV.TVPruefdienst.CBP.CBPLogik.Allgemein.GlobalesAktualisieren
Imports DRV.TVPruefdienst.CBP.CBPLogik.Allgemein.GlobalesAktualisieren.Anlagen
Imports DRV.TVPruefdienst.CBP.CBPTest.GlobalesAktualisieren
Imports DRV.TVPruefdienst.FachRahmenwerk
Imports NUnit.Framework

Namespace CBPTest
    <TestFixture()> _
    Public Class GlobalesAktualisierenTest
#Region "Test Setup und TearDown"
        <SetUp()> _
        Public Sub SetUp()
            prueffallSynchroneAnlage = New CBPPrueffallSynchroneAnlageStub()
            prueffallAsynchroneAnlage = New CBPPrueffallAsynchroneAnlageStub()

        End Sub

        <TearDown()> _
        Public Sub TearDown()
            prueffallSynchroneAnlage = Nothing
            prueffallAsynchroneAnlage = Nothing
        End Sub
#End Region
#Region "Tests"
        <Test()> _
        Sub PrüfenSynchronerAnlage()
            'Arrange
            Dim ueberarbeiter As IGlobalesAktualisierenUeberarbeiter = New GlobalesAktualisierenUeberarbeiter(prueffallSynchroneAnlage)
            Dim synchroneAnlage As IGlobalesAktualisierenAnlage = New GlobalesAktualisierenGSVNachberechnungsAnlage(prueffallSynchroneAnlage)
            'Act
            Dim istAsynchron As Boolean = ueberarbeiter.PruefeAnlage(synchroneAnlage)
            'Assert
            Assert.IsFalse(istAsynchron, "Anlage ist synchron")
        End Sub

        <Test()> _
        Sub PrüfenAsynchronerAnlage()
            'Arrange
            Dim ueberarbeiter As IGlobalesAktualisierenUeberarbeiter = New GlobalesAktualisierenUeberarbeiter(prueffallAsynchroneAnlage)
            Dim asynchroneAnlage As IGlobalesAktualisierenAnlage = New GlobalesAktualisierenGSVNachberechnungsAnlage(prueffallAsynchroneAnlage)
            'Act
            Dim istAsynchron As Boolean = ueberarbeiter.PruefeAnlage(asynchroneAnlage)
            'Assert
            Assert.IsTrue(istAsynchron, "Anlage ist asynchron")
        End Sub

        <Test()> _
        Sub AktualisierenAsynchronerAnlage()
            'Arrange
            Dim ueberarbeiter As IGlobalesAktualisierenUeberarbeiter = New GlobalesAktualisierenUeberarbeiter(prueffallAsynchroneAnlage)
            Dim asynchroneAnlage As IGlobalesAktualisierenAnlage = New GlobalesAktualisierenGSVNachberechnungsAnlage(prueffallAsynchroneAnlage)
            'Act
            Dim istErfolgreich As Boolean = ueberarbeiter.AktualisiereAnlage(asynchroneAnlage)
            'Assert
            Assert.IsTrue(istErfolgreich, "Anlage ist aktualisiert")

        End Sub
#End Region
#Region "Attribute und Konstanten"
        Private prueffallSynchroneAnlage As IPrueffall
        Private prueffallAsynchroneAnlage As IPrueffall
#End Region
    End Class
End Namespace
```

Abbildung 9: Komponententest