Winterprüfung 2017	Fest, Fabian Fachinformatiker für Anwendungsentwicklung Prüflingsnummer: (183) 00120 Id.Nr.: 1403997

Fachinformatiker für Anwendungsentwicklung
Dokumentation zur betrieblichen Projektarbeit

ZRV

Erstellung eines zentralen Rechteverwaltungssystems

Ausbildungsbetrieb:	GSSD mbH Potsdamer Str. 90 14513 Teltow
Ansprechpartner:	Peter Altmann 03328303995

Eidesstattliche Erklärung: Hiermit erkläre ich Fabian Fest die hier vorliegende Arbeit (Dokumentation) selbstständig und nur unter Zuhilfenahme der aufgeführten Quellen angefertigt zu haben. Der aufgeführte zeitliche Rahmen wurde eingehalten.

Ort/Datum	Unterschrift

IHK Potsdam

Winterprüfung 2017

Ausbildungsberuf
Fachinformatiker/-in Anwendungsentwicklung

Prüfungsbezirk
Potsdam FI 1196-1 (AP T2V1)

Herr Fabian Fest
Identnummer: 1403997
E-Mail: fabian.fest@posteo.de, Telefon: 015234599303

Ausbildungsbetrieb: GSSD mbH
Projektbetreuer: Herr Peter Altmann
E-Mail: peteraltmann@gssd.de, Telefon: 03328303995

Thema der Projektarbeit
ZRV - Erstellung eines zentralen Rechteverwaltungssystems

1 Thema der Projektarbeit

ZRV - Erstellung eines zentralen Rechteverwaltungssystems

2 Geplanter Bearbeitungszeitraum

Beginn: 27.09.2017
Ende: 29.11.2017

3 Projektbeschreibung

Ziel ist die Erstellung eines Systems zur einheitlichen und zentralen

Verwaltung von Benutzerrechten bei Verwendung von Individualprogrammen sowie von Standardsoftware.

Aufgrund vieler verschiedener Programme, die bei diversen Kunden unter Windows und anderen Betriebssystemen eingesetzt werden, wird die Rechtevergabe durch die Systembetreuung mit jedem zusätzlichen Programm exponentiell aufwendiger, da jedes dieser Programme in der Regel eine individuelle Rechteverwaltung besitzt. Das führt zu dem Wunsch nach einer einheitlichen und programmübergreifenden Verwaltung von Rechten, um diese Systembetreuung wirtschaftlicher zu betreiben und damit übersichtlicher zu gestalten. Es soll eine Abhängigkeit der Rechte vom angemeldeten Benutzer, dem Rechte benötigenden Programm und dem zu bearbeitenden Objekt bestehen. Bewusst sollen die aufwendigen aber nicht so weit reichenden Möglichkeiten eines LDAP oder Active Directory Verzeichnisdienstes vermieden werden.

In der Projektarbeit soll ein Konzept bezüglich der oben genannten Anforderungen entwickelt werden, wie mit einer Standardbenutzeroberfläche (diese ist nicht Teil der Projektarbeit) für möglichst alle Softwaresysteme, die bei Kunden eingesetzt werden, die Benutzer und deren Rechte an Objekten zentral und flexibel verwaltet werden können.

Es ist zu erarbeiten, welche neuen Datenstrukturen sowie Module, Services, Client-Programme, Interfaces/Adapter für bereits bestehende Programme neu erstellt werden müssen und wie diese miteinander interagieren.

4 Projektumfeld

Das Projekt wird im Auftrag der GSSD durchgeführt. Zur Anwendung wird das fertig gestellte System in der Systemadministration der GSSD kommen, um die Rechte der beim Kunden eingesetzten Software effizient einzurichten und zu warten.

In einem späteren Schritt sollen Kunden selbst in der Lage sein, ihre Rechte zu verwalten.

5 Projektphasen mit Zeitplanung

1) Analyse Ist-/Soll (ges. 6h)
  - Analyse Ist-Zustand (3h)
  - Analyse Soll-Zustand (2h)
  - Analyse bereits bestehender Fremdsoftware, die diese Aufgabe übernehmen könnte (1h)
2) Erstellen eines Projektplans (ges. 15h)
  - Erstellen Liste Anforderunen und Ziele (10h)
  - Abschätzen des Projektaufwandes, Risikobetrachtung (3h)
  - Zeit- und Kostenplanung (2h)
3) Implementierung, Umsetzung des Konzeptes (42h)
4) Integration und Test (4h)
5) Einführung (2h)
6) Wirtschaftlichkeitsbetrachtung (1h)

6 Dokumentation zur Projektarbeit

1. Problembeschreibung
2. Lösungsmöglichkeiten
   2.1 AD
   2.2 weitere
   2.3 Eigenentwicklung
3. Lösungsweg
   3.1 gewählter Lösungsweg
4. Pflichtenheft

Anlagen werden diverse Grafiken enthalten.

7 Anlagen

keine

8 Präsentationsmittel

- Beamer
- Laptop

9 Hinweis!

Ich bestätige, dass der Projektantrag dem Ausbildungsbetrieb vorgelegt und vom Ausbildenden genehmigt wurde. Der Projektantrag enthält keine Betriebsgeheimnisse. Soweit diese für die Antragstellung notwendig sind, wurden nach Rücksprache mit dem Ausbildenden die entsprechenden Stellen unkenntlich gemacht.

Mit dem Absenden des Projektantrages bestätige ich weiterhin, dass der Antrag eigenständig von mir angefertigt wurde. Ferner sichere ich zu, dass im Projektantrag personenbezogene Daten (d. h. Daten über die eine Person identifizierbar oder bestimmbar ist) nur verwendet werden, wenn die betroffene Person hierin eingewilligt hat.

Bei meiner ersten Anmeldung im Online-Portal wurde ich darauf hingewiesen, dass meine Arbeit bei Täuschungshandlungen bzw. Ordnungsverstößen mit „null“ Punkten bewertet werden kann. Ich bin weiter darüber aufgeklärt worden, dass dies auch dann gilt, wenn festgestellt wird, dass meine Arbeit im Ganzen oder zu Teilen mit der eines anderen Prüfungsteilnehmers übereinstimmt. Es ist mir bewusst, dass Kontrollen durchgeführt werden.

10 Grund für „mit Auflage genehmigt“

Sehr geehrter Herr Fest,

Ihre Zeitplanung umfasst mehr als 70 Stunden. Für das Projekt stehen nach Prüfungsordnung nur 70 Stunden zur Verfügung. Das Projekt muss innerhalb von 70 Stunden bearbeitet und dokumentiert werden.

Zeitmitschrift der Projektarbeit

Dieses Formular ist mit der Projektdokumentation einzureichen!

Prüfungsteilnehmer/-in

Name: Fest

Vorname: Fabian

Login: 183 14 03 997
(Login des Online Portals)

Datum	Tätigkeit	Zeit in Stunden
02.10.2017	Anlegen Projektarbeit in Latex, Einleitung	3,5h
12.10.2017	Anlage Grobgliederung, stichpunktartig zu jedem Themengebiet	3h
16.10.2017	Einleitungsteil vollständig fertig gestellt	2h
18.10.2017	Weitere TODOs/Stichpunkte zur gesamten Projektstruktur	2h
20.10.2017	Projektplanungsphase	2h
25.10.2017	Analyse Ist / Soll	8h
26.10.2017	Analyse Kostenrechnung + Schnittstellen + Abgrenzung	3h
26.10.2017	Wirtschaftlichkeitsprognose	2h
27.10.2017	Komponenten Übersicht + Komponenten der ZRV	6h
01.11.2017	Pflichtenheft	7h
02.11.2017	Anhang + Lastenheft + Benutzerdokumentation	7h
03.11.2017	Überprüfung Pflichten gegen Lasten	3h
03.11.2017	Integration	3h
03.11.2017	Einführung	1h
13.11.2017	Überarbeitung Benutzerdokumentation, und der Komponenten	8h
14.11.2017	Prüfen der Projektdokumentation gegenüber der Handreichung Überprüfung Verweise, Inhalt auf Vollständigkeit	3h
14.11.2017	Erarbeitung Rechte-XML, sowie XSD	4h
15.11.2017	Deckblatt, Layout, Zeitmitschrift, Antrag in Projektarbeit integriert	2,5h
	Pufferzeit	
	Projektplanungsphase	
	Analysephase	
	Implementierungsphase	

Inhaltsverzeichnis

1 Einleitung

In der folgenden Projektdokumentation stellt der Autor den Projektablauf dar, der im Rahmen seiner Ausbildung zum Fachinformatiker für Anwendungsentwicklung durchgeführt wurde. Die Durchführung der Projektarbeit erfolgt bei der Gesellschaft für Systemtechnik, Softwareentwicklung und Datenverarbeitungsservice mbH (GSSD) in Teltow. Zur Zeit beschäftigt die GSSD 4 Mitarbeiter und betreut verschiedene kleine und mittlere Unternehmen als IT-Dienstleister. Des Weiteren ist die GSSD Partner verschiedener Warenwirtschaftssoftware und bietet in diesem Umfeld Anpassung und Individualisierung an. Zweites Standbein ist der Vertrieb von Hardware für Unternehmen und Privatkunden sowie deren Installation und Wartung.

1.1 Projektbeschreibung

Als IT-Dienstleister erhält die GSSD immer wieder Aufträge, bestehende Fremdsoftware zu erweitern. Zum Teil enthalten diese Programme ihre eigene Rechteverwaltung für Benutzer oder greifen auf bestehende Technologien, wie zum Beispiel AD zurück. Die Erweiterungen der GSSD dürfen meist nicht allen Benutzern zur Verfügung stehen und müssen aus diesem Grund ebenfalls auf eine Benutzer-Rechte-Verwaltung zugreifen.
Verschiedene Module der GSSD oder von anderen Unternehmen bringen aus historischen Gründen ihre eigene Rechteverwaltung mit sich, die meist parallel zu den Hauptprogrammen (z.B. Warenwirtschafts- oder Buchhaltungssoftware) betreut und gepflegt werden. Ein einfaches Benutzerrecht zu vergeben bedeutet enormen Aufwand und kann mitunter nur vom Programmierer, nicht aber vom Systemadministrator oder Kunden, durchgeführt werden.
Im Rahmen dieser Projektarbeit soll der beschriebene Prozess vereinheitlicht und vereinfacht werden.

1.2 Projektziel

Ziel ist die Erstellung eines Systems zur einheitlichen und zentralen Verwaltung von Benutzerrechten unter Verwendung von Individualprogrammen sowie von Standardsoftware. Aufgrund vieler verschiedener Programme, die bei diversen Kunden unter Windows und anderen Betriebssystemen eingesetzt werden, wird die Rechtevergabe durch die Systembetreuung mit jedem zusätzlichen Programm exponentiell aufwendiger, da jedes dieser Programme in der Regel eine individuelle Rechteverwaltung besitzt. Das führt zu dem Wunsch nach einer einheitlichen und programmübergreifenden Verwaltung von Rechten. Es wird angestrebt, den zeitlichen Aufwand und somit die Kosten bei der Systembetreuung zu reduzieren. Es soll eine Abhängigkeit der Rechte vom angemeldeten Benutzer, dem Rechte benötigenden Programm und dem zu bearbeitenden Objekt bestehen. Bewusst sollen die aufwendigen aber nicht so weit reichenden Möglichkeiten eines LDAP oder Active Directory Verzeichnisdienstes vermieden werden.
In der Projektarbeit soll ein Konzept bezüglich der oben genannten Anforderungen entwickelt werden, wie mit einer Standardbenutzeroberfläche (diese ist nicht Teil der Projektarbeit) für möglichst alle Softwaresysteme, die bei Kunden eingesetzt werden, die Benutzer und deren Rechte an Objekten zentral und flexibel verwaltet werden können. Es ist zu erarbeiten, welche neuen Datenstrukturen sowie Module, Services, Client-Programme, Interfaces/Adapter für bereits bestehende Programme neu erstellt werden müssen und wie diese miteinander interagieren.

In einem späteren Schritt sollen Kunden, bei denen Software der GSSD eingesetz wird, selbst in der Lage sein, ihre Rechte zu verwalten.

1.3 Projektumfeld

Das Projekt wird im Auftrag der GSSD durchgeführt. Zur Anwendung wird das fertig gestellte System in der firmeninternen Systemadministration kommen, um Benutzerrechte der beim Kunden eingesetzten Software effizient einzurichten und zu warten. Eine enge Zusammenarbeit mit der Systemadministration ist hierbei unumgänglich, da die Mitarbeiter umfangreiches Wissen im Umgang mit diesen Systemen haben und durch Arbeit mit anderen Rechteverwaltungssystemen bereits einen Workflow gewohnt sind, der in der Handhabung der Zentrale-Rechte-Verwaltung (ZRV) hilfreich sein kann.

1.4 Projektbegründung

Die Zeit, die zur Änderung der Rechte oder gar für die Anlage eines komplett neuen Benutzers benötigt wird, ist derzeit zu hoch. Des Weiteren werden die Benutzerrechte bzw. Rechtekonfigurationen der Software, je nach verwendeter Erweiterung der GSSD, unterschiedlich eingerichtet. Die Konfiguration der Benutzerrechte unterscheidet sich von Programm zu Programm in Workflow, Form, Art und Ort zur Speicherung der Rechte.
Aufgrund dessen hat sich die GSSD entschieden, ein Pflichtenheft erstellen zu lassen, das zur Analyse und Lösung der derzeitigen Probleme führt.

1.5 Projektschnittstellen

Betreut wird das Projekt durch den Geschäftsführer der GSSD Peter Altmann und dem Mitarbeiter Klaus-Dieter Knoll.

Für die zentrale Speicherung von Benutzern und deren Rechten wird eine Dateischnittstelle gewählt.

Zur Versionierung, Backup und Bereitstellung dieser Dateischnittstelle kommt freie Software zur Versionsverwaltung (git) zum Einsatz. Des Weiteren muss diverse Eigen- und Fremdsoftware an die ZRV angebunden werden. Um dies zu realisieren, wird ein Adaptermodul mit fester Schnittstelle definiert. Die Umsetzung der Komponenten und deren Schnittstellen erfolgt unter 4.10.

1.6 Projektabgrenzung

Nicht Bestandteil im Projektumfang ist die Entwicklung einer grafischen Benutzeroberfläche, diese wird vollständigkeitshalber angeführt, kann jedoch im zeitlichen Rahmen von 70 Stunden nicht planerisch berücksichtigt werden.

2 Projektplanung

2.1 Projektorganisation

Geleitet wird das Projekt durch den Autor. Ebenso erfolgt durch ihn die Ausarbeitung des Pflichtenheftes auf Grundlage des Lastenheftes Anhang A.1: Lastenheft auf Seite II, die daraus folgende Erarbeitung der einzelnen Komponenten und Prozessketten, sowie Aktivitäten im Gesamtprozess.

2.2 Projektphasen

Zur Durchführung des Projektes standen 70 Stunden zur Verfügung. Für die Ausarbeitung des Konzeptes wird die meiste Zeit eingeplant. Eine ausführliche Projektplanung ist im voraus nötig, allerdings kann auf Grund fehlender Erfahrung keine genaue Zeit geschätzt werden, es sind 15 Stunden angesetzt. Da der Autor zum Zeitpunkt der Durchführung dieser Projektarbeit mit den anzubindenden Komponenten und Problemen bei der täglichen Arbeit in der GSSD vertraut ist, wird ein Zeitaufwand von 6 Stunden für die Analysephase als realistisch eingeschätzt. Die weiteren Phasen werden als unkritisch eingestuft und sollten in der Ausarbeitung um maximal eine Stunde je Phase abweichen

Projektphasen Tabelle 1 zeigt die grobe Zeitplanung aus dem Projektantrag.

Projektphase	Geplante Zeit
Analyse Ist/Soll	6 h
Erstellung Projektplan	15 h
Implementierung, Umsetzung des Konzeptes	42 h
Integration und Test	4 h
Einführung	2 h
Wirtschaftlichkeitsbetrachtung	1 h
Gesamt	**70 h**

Tabelle 1: Zeitplanung grob

Dem Anhang A.2: Detaillierte Zeitplanung auf Seite III kann die detaillierte Zeiteinteilung entnommen werden.

Die Wirtschaftlichkeitsbetrachtung im Pflichtenheft ist unter 3.5 zu finden ist. Es wurde eine hypothetische Amortisationsrechnung durchgeführt, da die Umsetzung des Konzeptes der ZRV nicht Bestandteil der Projektarbeit und somit die reale Zeiteinsparung beim Supportprozess nicht direkt feststellbar ist.

2.3 Ressourcenplanung

Die essentiellen Ressourcen, die im Rahmen des Projektes benötigt werden, sind mit der Standardeinrichtung eines IT-Unternehmens abgedeckt. Alle Planungsschritte wurden im Büro der GSSD durchgeführt. Ein PC mit LaTeX, sowie WhiteStarUML in der freien Version GNU General Public License version 2.0 (GPLv2) (Lizenzbedingungen siehe Anhang) stand zur Verfügung.

3 Analyse

3.1 Ist-Analyse

Derzeit liegen viele eigene Produkte der GSSD mit jeweils individueller Rechteverwaltung vor. Ebenso betreut die GSSD diverse Kunden als IT-Dienstleitungsunternehmen und verwendet zu diesen Zwecken Fremdsoftware, Active-Directory von Microsoft übernimmt häufig deren Benutzerverwaltung.
In der Regel besteht die eingesetzte Software aus zwei Komponenten, dem Programm selbst, mit Benutzeroberfläche und einer Konfigurationseinheit. Die Konfigurationseinheit liegt in Form einer Datenbank oder Datei vor.
Benutzerrechte werden als unabhängige Objekte betrachtet, die einem Benutzer, einer Software sowie Funktionen darin zugeordnet werden können. Zum Beispiel gibt es ein Recht, um ein Programm zu starten. Der Programmierer dieser Software definiert darin verschiedene Zustände, wie z.B. eine Liste mit bestimmten Inhalt anzuzeigen. Jedoch ist es nicht jedem Benutzer erlaubt diesen Inhalt zu sehen, sondern es dürfen jeweils nur bestimmte Teile angezeigt werden. So ergeben sich beliebig viele und tiefe Rechtestrukturen.

3.1.1 Workflow

Folgender Workflow tritt regelmäßig bei der GSSD auf:

1. Anruf/E-Mail geht beim Support ein, ein Recht soll geändert werden.
    1.1. Meist soll ein neuer Benutzer die gleichen Rechte, wie ein bereits bestehender Benutzer erhalten.
2. Aufgrund der Komplexität, welches Programm durch welche Konfigurationseinheit gesteuert wird (unter anderem ob lokale, zentrale oder beide Konfigurationseinheiten vorliegen), muss der entsprechende Programmierer gefragt werden.
3. Mühsames Suchen und ändern der Konfiguration in einer Datenbank bzw. Datei folgt.
4. Rückmeldung an den Kunden.

Hierbei fallen Zeiten von bis zu einer Stunde an.

3.2 Soll-Analyse

Forderungen aus dem Anhang A.1: Lastenheft auf Seite II sind die zentrale Speicherung und Verwaltung von Benutzerrechten. Die Entscheidung welches Medium zur Speicherung der Daten verwendet wird entfällt, eine vom Menschen direkt interpretierbare Datei-Schnittstelle wird vorgegeben.
Die Zeit für den gesamten Prozess der Rechtevergabe, soll sich auf wenige Klicks und somit auf wenige Minuten beschränken. Um diese Ziele durchzusetzen, werden von der ZRV Konventionen eingeführt:

- Zentraler Ort der Rechtekonfiguration pro Kunde.
- Einheitlicher Aufbau der Konfigurationsdatei.
- Einheitlicher Zugriff auf die Konfigurationsdatei.

Mittels der ZRV soll langes Suchen von Benutzerrechten vermieden werden, alle Rechte befinden sich an einem zentralen Punkt. Eine Rückmeldung, durch den Systemadministrator beim Kunden, erfolgt nach wie vor.

3.3 Analyse möglicher Probleme

- Ein Problem stellt die Verfügbarkeit der Rechtedatei dar. Diese muss zu jeder Zeit gegeben sein und darf nicht durch andere Prozesse eingeschränkt werden.
- Die Integrität der Rechtedatei muss stets gegeben sein. Sollte diese nicht mehr gegeben sein, müssen Fall-Back Mechanismen greifen.
- Geschwindigkeit des Dateizugriffs, ist eventuell zu gering. Caching der Rechte pro Sitzung kann Abhilfe schaffen.
- Handling paralleler Zugriffe auf Rechtedatei bzw. zentrales git-Repository.
- Wer darf die Datei schreiben? Zugriff von Unbefugten verhindern.

3.4 Analyse Fremdsoftware/Komponenten

3.4.1 Active-Directory (AD)

AD ist ein Dienst innerhalb einer Windows Domain, bietet unter anderem eine leichte Verwaltung von Benutzern und deren Rechten. Die AD bietet in diesem Rahmen sehr tiefgreifende und verschachtelte Rechte für den Windows-Benutzer selber an. Als Softwareentwickler hat man die Möglichkeit über das Windows-Application Programming Interface (API) die AD anzusprechen und beispielsweise eigene Programme auf dieser Basis aufzubauen. Active Directory setzt zu dem immer einen Domain Controller (diese Rolle können sowohl Microsoft Windows Server, als auch in beschränkten Umfang Samba Server übernehmen) voraus.

3.4.2 Weitere Fremdsoftware

Eine bereits fertige Fremdsoftware auf dem Markt zu suchen ist nicht Sinn und Zweck des Projektes, dies würde zudem den zeitlichen Rahmen überschreiten und wurde daher nicht durchgeführt. Als einzige Fremdsoftware wird git benutzt. Git wird weitgehend von Programmierern als Versionierungstool und zum Absichern Programmcode verwendet.

3.4.3 Eigenentwicklung

Die Eigenentwicklung von Software hat den Vorteil, das genauestens auf die vorgegebene Problematik eingegangen und eine optimale Lösung für das zugrunde liegende Problem gefunden werden kann. Der Workflow bei Benutzung dieser Software kann im Voraus geplant und anschließend umgesetzt werden.

3.4.4 Projektkosten

Für die Projektarbeit sind insgesamt 70 Stunden veranschlagt. Bei einem Stundensatz von 16 € für einen Programmierer, zuzüglich 35 € Gemeinkosten, fallen Kosten in Höhe von 3570 € an.
Weitere Kosten die nicht im Rahmen der Projektarbeit anfallen, jedoch nicht vernachlässigt werden können, da sie im direkten Zusammenhang mit der ZRV stehen und bei der Ausimplementierung des Konzeptes anfallen:

- Erstellung des Adapter-Moduls und zrvadapter.exe zur Nutzung der ZRV.
- Anbindung an die ZRV, mittels des Adapter-Moduls, in der bestehenden Eigensoftware.
- grafische Benutzeroberfläche.

Da sowohl AD und andere Fremdsoftware an dieser Stelle keine Alternativen zu unseren speziellen Anforderungen bieten, muss unter den Gesichtspunkten des Anhangs A.1: Lastenheft auf Seite II eine eigene Rechteverwaltung entwickelt werden.

Kostenschätzung der Projektumsetzung Wird die Implementierung der nachfolgenden Komponenten von einem Programmierer der GSSD durchgeführt, wird ebenfalls mit dem oben genannten Stundensatz von 16 €, zzgl. 35 € Gemeinkosten kalkuliert.

Für die Entwicklung der zrvadapter.exe werden 4 Stunden veranschlagt (204 €). Es werden mit 8 h für die Entwicklung des Adapter-Moduls gerechnet (408 €). Das bereitgestellte Adapter-Modul, muss bei bestehender Eigensoftware implementiert werden, sofern die ZRV benutzt werden soll. Dafür wird ein Aufwand von ca. einem Arbeitstag pro Programm geschätzt (je 408 €).

Die für die Fremdsoftware zusätzlich benötigte adaptermodul.exe, muss für jede Fremdsoftware speziell angepasst werden. Pro Fremdsoftware-Modul wird mit einem Aufwand von mindestens zwei Arbeitstagen gerechnet (je 816 €).

Die im Rahmen der Projektarbeit erwähnte grafische Benutzeroberfläche, muss von Grund auf entwickelt werden. Für die Erstellung von ca. 80% der Funktionalität und damit dem erreichen des ersten Milestone, werden inklusive Tests ca. 10 Arbeitstage berechnet (4080 €). Sofern die bis dahin realisierte Funktionalität nicht als ausreichend bewertet wird, können in weiteren Schritten fehlende Funktionen ergänzt werden. Aufwandsschätzungen müssen daraufhin erneut stattfinden.

Für die abschließende Schulung sind einmalig 4 Stunden geplant (204 €).

Da das Projekt und auch alle davon abhängigen Prozesse keinem Kundenprojekt direkt zugeordnet werden können, müssen die Projektkosten mit zukünftigen Projekten verrechnet werden, sie fallen daher zunächst als variable Kosten im Unternehmen an.

3.5 Amortisation

Aufgrund der Eigenentwicklung fallen keine weiteren Lizenzgebühren zur Nutzung der Rechteverwaltung an. Das Projekt deckt allerdings nur die Planung ab. Die Implementierung ist nicht Bestandteil der Projektarbeit, die dafür anfallenden Kosten werden nicht in diesem Projekt berücksichtigt und aufgeführt.

Sofern eine Implementierung stattgefunden hat und neue Programme auf deren Grundlage aufbauen können, wird nicht die Dauer der Implementierung dieser Rechteverwaltung in neuen Programmen geringer, sondern die Zeit, die zur Wartung beim Kunden (speziell die Einrichtung der Benutzerrechte) benötigt wird. Eine Amortisationsrechnung kann daher nicht direkt stattfinden. Ausgehend davon, dass durch dieses Rechtesystem fast vollautomatisch Benutzerrechte vergeben werden können, wird für diesen Vorgang nur noch ein Bruchteil der ursprünglichen Zeit benötigt. Zu einem späteren Zeitpunkt, soll eine Benutzeroberfläche folgen (nicht Bestandteil des Projektes), mit der Kunden in der Lage sind, die teils komplexen Rechte selbstständig zu vergeben, um Zeit und Kosten zu sparen.

3.5.1 Vorteile der ZRV

Folgende Vorteile schafft die ZRV im Vergleich zum alten System:

- Sehr leichte Bedienung
- Einheitliche Benutzung durch einheitliches Schema und Konventionen
- Rechte können mit Editor bearbeitet werden → keine SQL Kenntnisse notwendig
- gleich bleibender Workflow für Systemadministrator → schneller in Bearbeitung
- Rechte liegen zentral → geringer Back-Up Aufwand
- Die Konfiguration von Fremdsoftware kann bei Verwendung der ZRV mit gewohntem Workflow angepasst werden
- gleichbleibende API durch Adapter-Modul
- kein Datenbankserver als Voraussetzung für Konfigurationsverwaltung → einfacheres System, ideal für Einzelplatzanwendungen

4 Pflichtenheft

4.1 Auftrag

Die GSSD mbH stellt dem Autor Fabian Fest den Auftrag, die momentan verwendeten Rechteverwaltungen diverser Eigensoftware und Fremdsoftware unter einer gemeinsamen Verwaltung zusammenzufassen.
Weitere Details können dem Anhang A.1: Lastenheft auf Seite II entnommen werden.

4.1.1 Ausgangslage

Im Ursprungszustand liegt keine einheitliche Rechteverwaltung vor. Jedes Programm bezieht die Benutzerrechte aus einer eigenen gewachsenen proprietären Rechteverwaltung.

4.2 Abgrenzung

Im zeitlichen Rahmen der Projektarbeit kann die grafische Benutzeroberfläche der ZRV nicht durch den Autor geplant werden. Einzig die Eingliederung in die Gesamtprozesse der ZRV wird betrachtet. Des Weiteren wird vom Autor keinerlei Programmcode erzeugt, die Umsetzung des Gesamtkonzeptes, in Form eines Pflichtenheftes, gilt es hier zu betrachten.

4.3 Entscheidung und Zielstellung

Es wird sehr schnell deutlich, dass zur einheitlichen Rechteverwaltung nur die Rechtedatei notwendig ist. Die eigentliche Schwierigkeit ist, die zentrale Bereitstellung dieser Rechtedatei sowie parallele Zugriffe zu ermöglichen. Ebenso muss diese Datei gegen Verlust gespeichert und auf vorherige Versionen zurück gegriffen werden können. Git als Versionierungstool beinhaltet diese Funktionen. Die AD bietet ähnliche Funktionalität, allerdings soll die Rechteverwaltung von keiner Domain abhängig sein, da Anwendungen der GSSD auch bei Einzelplatz-PC's, die nicht die Voraussetzung einer Domainverwaltung erfüllen, funktionieren sollen.

Daher entschied sich der Autor zur Entwicklung einer neuen Rechteverwaltung namens ZRV. Um Zeit und Kosten bei der Entwicklung des Konzepts, sowie der später erfolgenden Umsetzung in einem Programm, zu sparen, wird auf bestehende Funktionen von git zurück gegriffen.

Ziel der Projektarbeit ist es, die benötigten Komponenten darzustellen und deren Nutzen zu erläutern.

4.4 Schnittstellen

Die ZRV besteht aus drei Schichten.
Die erste Schicht stellt ein geteiltes git-Verzeichnis (4.10.1) dar . Git übernimmt die Speicherung sowie Versionierung der Rechtedatei und stellt diese zentral bereit.
Die zweite Schicht umfasst zrvadapter.exe (4.10.3) und das Adapter-Modul, eine Programmbibliothek (4.10.2). Diese Schicht beschäftigen sich weitestgehend mit der Bereitstellung der Rechtedatei bzw. Rechte selbst.
Dritte Schicht stellen die Eigen- bzw. Fremsoftware dar, die die ZRV nutzen. Eine grafische Darstellung der Komponenten erfolgt in Kapitel 4.10.

4.5 Generelles Verhalten bei Störungen

Bei Störungen in der ZRV ist kein Prozess vorgesehen, der zum automatischen Wiederaufsetzten des Systems führt. Manuell wird es im begrenzten Umfang möglich sein, solche Mechanismen anzustoßen, jedoch wird auf eine Entscheidung seitens der Software verzichtet. In solchen Fällen muss der Systemadministrator das Problem analysieren. Das Adapter-Modul garantiert die Bereitstellung valider Daten. Im Fehlerfall werden eine entsprechende Meldung, jedoch keine Benutzerdaten zurück gegeben.

4.6 Gewünschte Situationen, Verhalten bei Fehlbedienung und Störung

Der Normalzustand ist, dass beliebig viele Prozesse asynchron auf den Rechtespeicher mittels des Adapter-Moduls zugreifen können. Das git-Subsystem managt die asynchronen Zugriffe, sodass sich diese nicht gegenseitig konkurrieren. Wird die Konfigurationsdatei beschädigt (die interne XML-Struktur ist nicht mehr valide) und in das zentrale git-Repository[1] geladen, so muss das dem Systemadministrator gemeldet werden. Dieser hat die Möglichkeit die Rechtedatei entweder per Hand zu bearbeiten oder eine ältere Version aus dem zentralen git-Repository wieder herzustellen. Gleiches gilt auch, wenn Fehlentscheidungen bei der Rechtevergabe getroffen wurden. Die Versionierung wird an dieser Stelle durch git übernommen. Eine Anleitung zur Wiederherstellung einer älteren Version kann dem Benutzerhandbuch im Anhang B.3.1: Wiederherstellung einer älteren Version auf Seite XII entnommen werden.

4.7 Benutzerschulung

Vor Inbetriebnahme der ZRV bei der GSSD, findet im Bürogebäude der GSSD eine ausführliche Einführung statt. Es wird beispielhaft gezeigt, wie eine Rechtedatei angelegt wird und diese der ZRV bereitgestellt werden kann. Ebenso wird gezeigt, wie Änderungen an der Rechtedatei durchgeführt werden. Außerdem werden Fehlerfälle konstruiert und vorgeführt, sowie Lösungswege zum beheben dieser Fehler geboten. Zielgruppe der Schulung sind die Systemadministratoren der GSSD. Es sind insgesamt 4 Stunden eingeplant.

4.8 Dokumentation

Die Projektdokumentation befindet sich im Anhang B: Benutzerdokumentation auf Seite X und beschreibt die allgemeine Handhabung und Workflows der ZRV.

4.9 Migration

Die Migration der alten Rechteverwaltungen in die ZRV kann Schrittweise von statten gehen. Beide Systeme können parallel bestehen. Durch einen Programmierer der GSSD erfolgt die Integration des Adapter-Moduls in Eigensoftware. Für die Verwaltung von Fremdsoftware wird jeweils ein spezielles Modul erstellt. Ebenso muss das Konfigurationsverzeichnis vorbereitet und die Rechtedatei mit gewünschten Rechten eingerichtet werden.

4.10 Implementierungsphase

Im Folgenden werden die einzelnen Komponenten detaillierter beschrieben. Die Grafik im Anhang A.4: Komponenten-Diagramm ZRV auf Seite IV beschreibt die Übersicht aller möglichen Komponenten der ZRV.

[1] Server-Verzeichnis mit git-Verwaltung und der 'bare' Einstellung

4.10.1 Zentrales git-Repository

Bereit gestellt werden die Benutzerrechte in einem zentralen git-Repository[2]. Dieses git-Repository sollte auf einem PC oder Server eingerichtet werden, der täglich gegen Datenverlust gesichert wird. Zudem bietet git die Möglichkeit Versionsstände der Rechte inkrementell zu speicherm. Ein Zurücksetzten auf vorherige Stände ist somit gegeben. Die Besonderheit eines solchen zentralen, in Fachsprache shared git-Repository genannt, sind nicht direkt bearbeitbare Dateien, wie sie im lokalen git-Repository vorliegen. Alle Versionsstände der beinhalteten Dateien liegen in komprimierter Form, nach Speicherständen sortiert, vor. Dateiinhalte sind nur mit git auszulesen. Neben dem Kommandozeilenprogramm (`https://git-scm.com/`) gibt es diverse Programme, die darauf aufbauen und eine grafische Benutzeroberfläche bieten. Nachzulesen unter `https://git-scm.com/download/gui/win`.
Wird das shared git-Repository auf einem Windows Server abgelegt, so wird der Zugriff auf das Verzeichnis durch die Domain gesteuert.
Ebenso gut kann sich das shared git-Repository auf einem Linux Server befinden. In diesem Fall kann über Secure Shell (SSH), unter Angabe eines Benutzers (vorausgesetzt dieser hat die benötigten Rechte), auf das Verzeichnis per git zugegriffen werden.
Hinweise zur Benutzung befinden sich im Anhang B.2: Einrichtung des Konfigurationsverzeichnis auf Seite X.

4.10.2 Adapter-Modul

Unter einem Adapter-Modul der ZRV ist eine Programmbibliothek zu verstehen, um auf das lokale git-Repository zugreifen zu können. Das Adapter-Modul benutzt unter anderem die zrvadapter.exe (4.10.3), um eine lokale, aktuelle Version der Rechtedatei zu erhalten und stellt grundlegende Funktionen zum Auslesen dieser Rechte bereit. Das Modul wird als .net Bibliothek bereitgestellt. Die API kann dem Klassendiagramm (Anhang A.5: Klassendiagramm Adapter-Modul auf Seite IV) entnommen werden. Umsetzung des Adapter-Moduls ist nicht Inhalt dieser Projektarbeit.
Fremdsoftware benötigt ein gesondertes Modul (adaptermodul.exe). Dieses Modul muss vom Programmierer speziell an die Fremdsoftware angepasst werden. Der Funktionsumfang entspricht dem des Adapter-Moduls und wird zusätzlich um eine Schreibfunktion der Fremdsoftwarerechte in die lokale Rechtedatei erweitert. Gesteuert werden kann das Programm über die Kommandozeile. Die API wird durch den Entwickler vorgegeben.

4.10.3 Zrvadapter.exe

Bei der zrvadapter.exe handelt es sich um ein einfaches Frontend[3] für git, zugeschnitten auf die Bedürfnisse der ZRV. Die wesentliche Aufgabe der zrvadapter.exe ist das herunterladen und bereitstellen der Rechtekonfiguration aus dem zentralen git-Repository. Der genaue Funktionsumfang kann dem Anhang B.1: Zrvadapter.exe auf Seite X entnommen werden. Die Umsetzung des Programms ist nicht Bestandteil des Projektes.

[2] Ist ein Verzeichnis, das komprimierte Versionsstände beinhaltet
[3] zusätzliche Programmschicht, um einem Benutzer Zugriff auf ein Programm oder Prozess zu ermöglichen, meist auch mit einer Benutzeroberfläche gleich zu stellen

4.10.4 Clientprogramm

Im Sinne der ZRV versteht sich unter einem Clientprogramm entweder Eigen- oder Fremdsoftware. Je nach Benutzer müssen diese Programme unterschiedlichen Inhalt anzeigen und Funktionen ermöglichen. Diese Rechte werden von der ZRV durch ein zentrales git-Repository bereit gestellt. Die Clientprogramme benutzen die zrvadapter.exe, um eine lokale sowie aktuelle Version der Rechtdatei zu erhalten. Unter Zuhilfenahme des Adapter-Moduls, können die Benutzerrechte auf einfache Weise abgefragt werden. Fremdsoftware benötigt zur Benutzung der ZRV allerdings ein weiteres Modul. Des weiteren unterscheidet sich der Workflow (siehe Kapitel 4.10.5).

4.10.5 Umgang mit Fremdsoftware

Fremdprogramme können nur bedingt bis gar nicht durch die ZRV gesteuert werden. Voraussetzung ist eine Schnittstelle, an der die Konfiguration der Software beeinflusst werden kann. Unter anderem stellen solche Schnittstellen Dateien oder Datenbanken dar. Im Falle einer Datenbank ist der Zugriff auf entsprechende Tabellen zu überprüfen. Dateien lassen sich in der Regel beschreiben, sofern sie nicht bereits durch einen anderen Prozess geöffnet sind. Ferner ist zu überprüfen, ob dies lizenzrechtlich möglich ist. Eventuell kann dieses Recht im Nachhinein mit dem Hersteller der Fremdsoftware ausgehandelt werden. Zudem sollte überprüft werden, ob die Garantie, Wartungsverträge o.ä. der Fremdsoftware erlischt sofern Änderungen am System stattfinden.
Sind Änderungen an der Konfiguration möglich, wird mittels adaptermodul.exe (siehe Anhang A.4: Komponenten-Diagramm ZRV auf Seite IV) die proprietäre Rechtekonfiguration der Fremdsoftware gelesen und in die lokale Rechtedatei geschrieben. Diese Konfiguration kann mit der zrvadapter.exe ebenfalls zentral gespeichert werden. Allerdings sollte vor Änderung der Fremdsoftwarerechte die aktuelle Rechtekonfiguration aus der Fremdsoftware ausgelesen werden. So kann im Nachhinein auf diese Konfiguration zurück gegriffen werden.
Ist es mit der ZRV nicht möglich Benutzereinstellungen in Fremdsoftware zu ändern, wird anstelle der Rechtedatei eine Benutzeranleitung bereit gestellt, mit der der Systemadministrator die Fremdsoftware konfigurieren kann.

4.10.6 Grafische Benutzeroberfläche

Die grafische Benutzeroberfläche ist nur optional und wird nicht in den Prozessen der ZRV benötigt. Allerdings sorgt die Benutzeroberfläche für ein einfaches Abändern und Neuanlegen der Benutzerrechte und garantiert die Integrität der Rechtedatei.

4.11 Komponenteninteraktion

Die beiden nachfolgenden Aktivitätsdiagramme verdeutlichen das Zusammenspiel der drei Hauptkomponenten. Es wird Abbildung 1 der Ablauf gezeigt, wenn die Konfiguration durch den Administrator geändert wird.

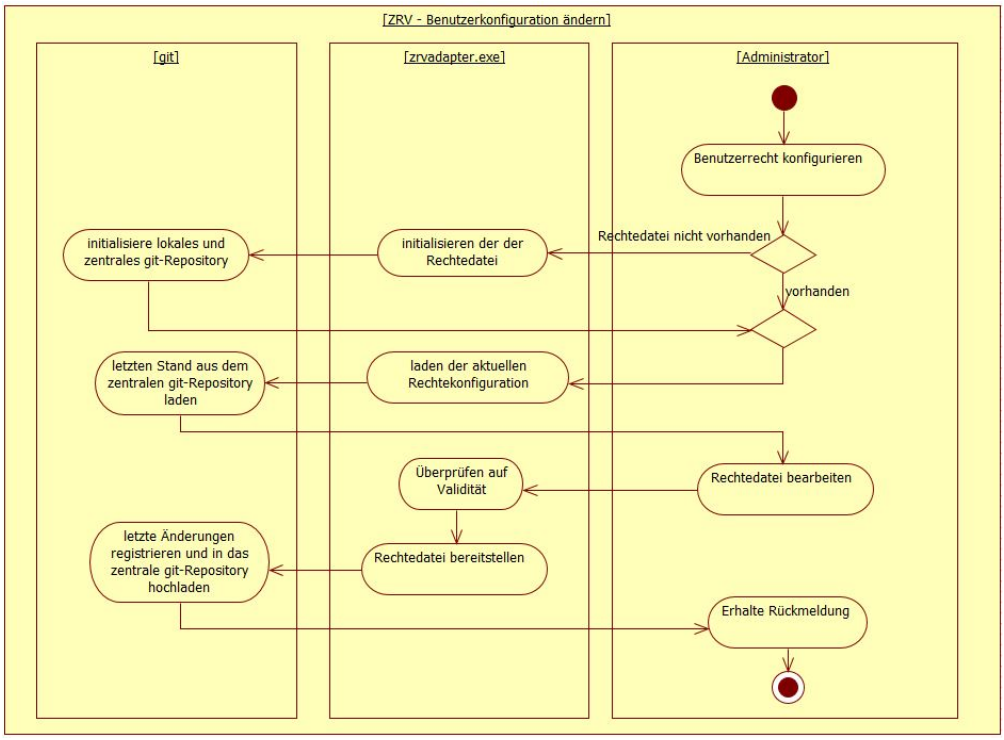

Abbildung 1: Activity-Diagramm Benutzerkonfiguration ändern.

Abbildung 2 zeigt den Programmfluss nach dem Starten einer Software, die ihre Rechte vom aktuellen Benutzer abruft.

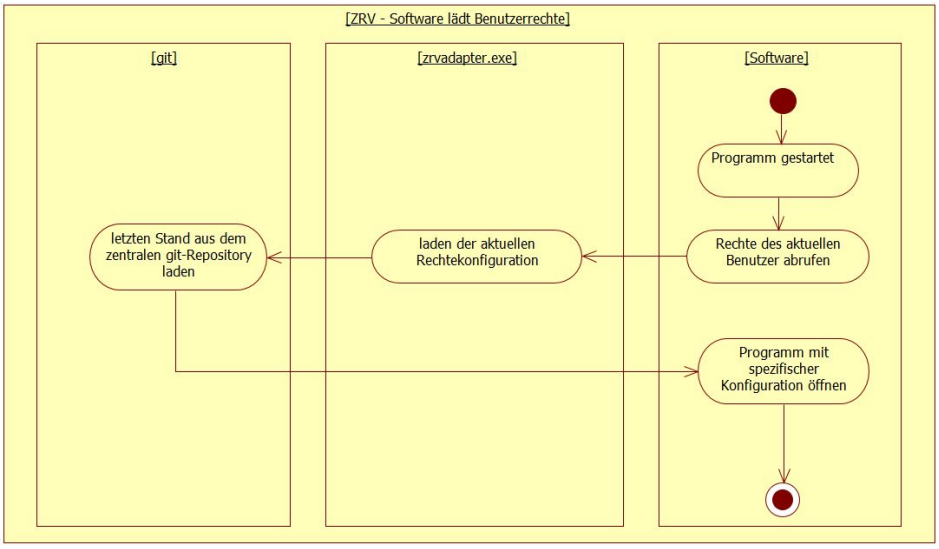

Abbildung 2: Activity-Diagramm Benutzerkonfiguration abrufen.

Die Unterteilung in drei Einheiten, hat den Vorteil, das ein Austausch der Komponenten durch ein beispielsweise neues Modul jederzeit möglich ist.

5 Integration und Test

Die ZRV basiert auf einer Datei, deren Inhalt einer XML-Struktur entspricht. Beim Einsatz im Echtbetrieb darf es zu keinerlei Fehlern kommen, daher ist von vorn herein die Integrität der Rechtedatei zu überprüfen. Die Überprüfung erfolgt mittels einer Beschreibungssprache für XML-Dateien (XSD).

Eine Anleitung dazu liegt Anhang B.3: Benutzerrecht bzw. Konfiguration ändern auf Seite XI vor.

6 Einführungsphase

Die Einführung findet in mehreren Schritten statt. Bevor die ZRV eingesetzt werden kann, ist dafür zu sorgen, dass alle benötigten Rechte in die Rechtedatei nach dem vorgegebenen Schema eingetragen werden. Daraufhin muss die Rechtedatei in das zentrale git-Repository geladen werden. Eine Anleitung dazu ist im Anhang B.2: Einrichtung des Konfigurationsverzeichnis auf Seite X zu finden.
Im zweiten Schritt muss das Adapter-Modul in allen Programmen, die an die ZRV angebunden werden sollen, implementiert werden. Wie das geschieht, obliegt jedem Programmierer selbst.
Daraufhin erfolgt ein Unit-Test, um das Adapter-Modul im Umfeld der Eigen- und Fremdsoftware auf Funktion zu überprüfen. Eventuelle Änderungsvorschläge können daraufhin eingereicht werden.
Wenn der Test erfolgreich war und eventuelle Änderungen am Adapter-Modul durchgeführt worden sind, findet im nächsten Schritt der Integrationstest statt. Hierbei muss ein Systemadministrator die Software der ZRV nutzen, um eine vorgegebene Konfiguration einzurichten. Ebenso ist es notwendig, dass alle möglichen Szenarien, die im täglichen Gebrauch auftreten können, zu durchlaufen. Eventuelle Änderungsvorschläge können daraufhin eingereicht werden.
Sofern Änderungen vorgenommen wurden, muss der komplette Testzyklus erneut durchlaufen werden. Das Projekt gilt als abgeschlossen, sofern alle Tests erfolgreich durchgeführt worden sind.

7 Fazit

Alle im Projektantrag genannten Anforderungen konnten erfüllt werden. Allerdings unterschied sich der reale Zeitplan von dem, wie er anfangs vermutet wurde. Tabelle 3 zeigt die Gegenüberstellung der geplanten und der Ist-Zeit. Wie angenommen nahm die Umsetzung des Konzepts die meiste Zeit in Anspruch. Die Analyse des Lastenheftes und die daraus folgende Ermittlung der Pflichten kosteten mehr Zeit als erwartet. Grund dafür war die hohe Anzahl an einzelnen Komponenten und der aktuelle sehr komplexe Prozess der Rechtevergabe. Die Projektplanungsphase wurde vom Aufwand, auf Grund der fehlenden Erfahrungen, weit aus umfangreicher eingeschätzt. Alle weiteren Phasen wurden vom zeitlichen Aufwand in etwa richtig geschätzt. Insgesamt konnten ca. 9 Stunden Arbeit keiner Projektphase direkt zugeordnet werden (Puffer). Diese Zeit wurde jedoch für die Kontrolle der Projektarbeit, zur Überprüfung der Rahmenbedingungen, wie Layout und Erstellung erforderlicher Anlagen, benötigt.

A Anhang

Abkürzungsverzeichnis

API	Application Programming Interface
AD	Active Directory
ZRV	Zentrale-Rechte-Verwaltung
GPLv2	GNU General Public License version 2.0
GSSD	Gesellschaft für Systemtechnik, Softwareentwicklung und Datenverarbeitungsservice mbH
git	freie Software zur Versionsverwaltung
SSH	Secure Shell
XSD	Beschreibungssprache für XML-Dateien
URI	Uniform Resource Identifier

Tabellenverzeichnis

Abbildungsverzeichnis

A.1 Lastenheft

Folgende Anforderungen muss die neue Rechte- und Konfigurationsverwaltung erfüllen:

1. einheitliches Schema bei Rechtevergabe
   1.1. hierarchische, beliebig tiefe Rechtestrukturen
   1.2. Rechte als unabhängige Objekte
   1.3. Möglichkeit Alias für Rechte anzugeben
   1.4. Layoutinformationen für Programme unterbringen
2. zentrale Verwaltung
3. Dateischnittstelle
   3.1. Rechte und Konfigurationen müssen mit dem Auge schnell erkennbar sein
   3.2. und durch Zuhilfenahme eines Texteditors muss die Konfiguration änderbar sein
4. Benutzer sollen sich mit dem Windowsbenutzer oder einem individuellen Benutzernamen authentifizieren können
5. jede Anwendung soll aus einem angegebenen Pfad optional eine Konfiguration laden können
6. Benutzerprofile müssen von anderen Benutzern kopiert werden können
7. Erstellung eines Benutzerhandbuches
8. Durchführung von Tests, um Funktionen sicher zu stellen

A.2 Detaillierte Zeitplanung

Analysephase		**6 h**
1. Analyse des Ist-Zustands	**2 h**	
2. Analyse des Soll-Zustands		
2.1 Analyse-Komponenten	**2 h**	
2.2 Schnittstellenanalyse	**1 h**	
2.3 Workflow-Analyse	**1 h**	
Projektplanungsphase		**15 h**
1. Erstellung einer Liste mit Anforderungen und Zielen	**11 h**	
2. Kosten und Aufwandsschätzung		
2.1 Abschätzung Projektaufwand und Kosten, Risikobetrachtung	**1 h**	
2.2 Zeit- und Kostenplanung	**3 h**	
Implementierungsphase		**42 h**
1. Schnittstellen	**8 h**	
2. Komponentenbeschreibung	**12 h**	
3. Komponenteninteraktion	**14 h**	
4. Erstellen der Dokumentation	**8 h**	
Integration und Test		**4 h**
Einführungsphase		**2 h**
Wirtschaftlichkeitsbetrachtung		**1 h**
Gesamt		**70 h**

Tabelle 2: Detaillierte Zeitplanung

A.3 Ist/Soll Zeitplanung

Projektphase	**Geplante Zeit (Soll)**	**(Ist)**	**Differenz**
Erstellung Projektplan	15 h	9	− 6 h
Analyse Ist/Soll	8 h	11	+ 5 h
Implementierung, Umsetzung des Konzepts	42 h	35	− 7 h
Integration und Test	4 h	3 h	− 1 h
Einführung	2 h	1 h	− 1 h
Wirtschaftlichkeitsbetrachtung	1 h	2 h	+ 1 h
Puffer	0 h	9 h	+ 9 h
Gesamt	**70 h**	**70 h**	

Tabelle 3: Soll/Ist-Vergleich benötigte Zeit je Phase

A.4 Komponenten-Diagramm ZRV

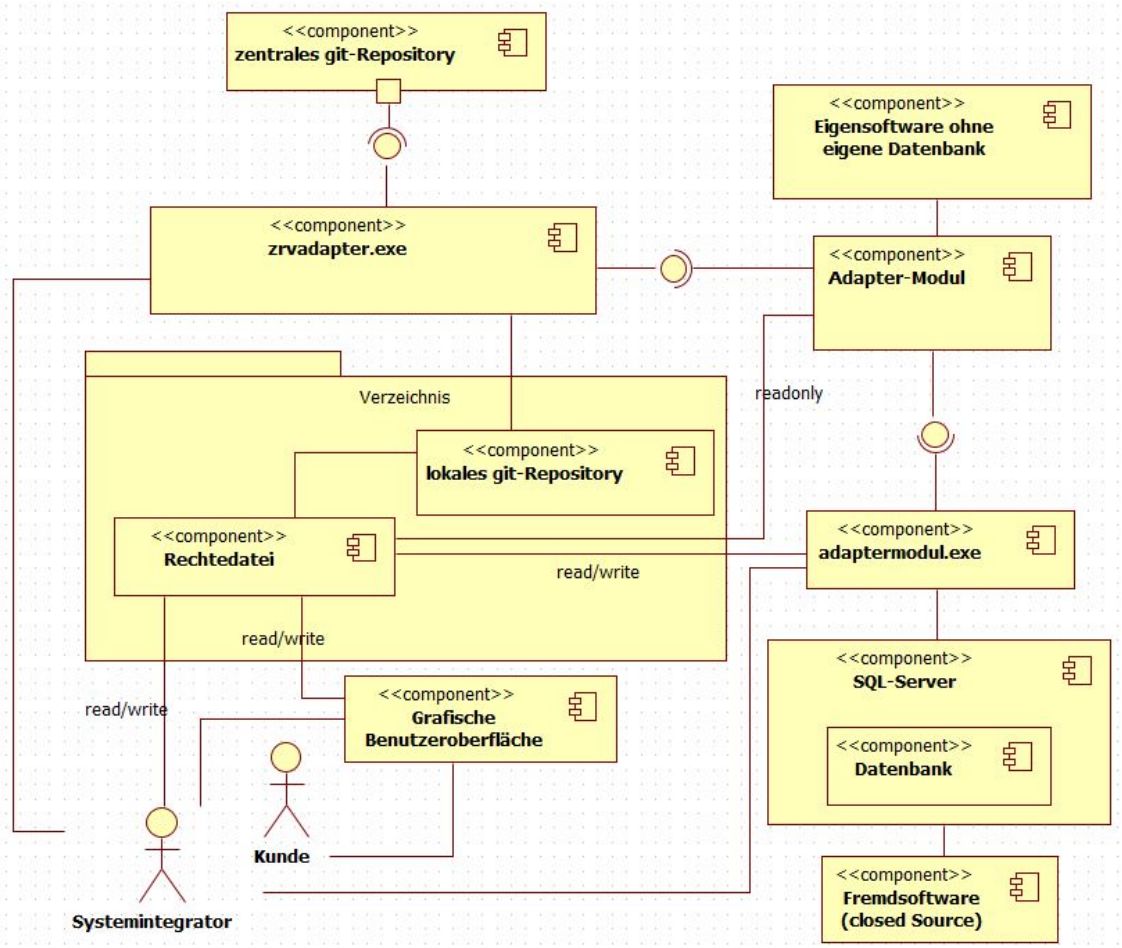

Abbildung 3: Übersicht aller Komponenten, die an der ZRV beteiligt sind

A.5 Klassendiagramm Adapter-Modul

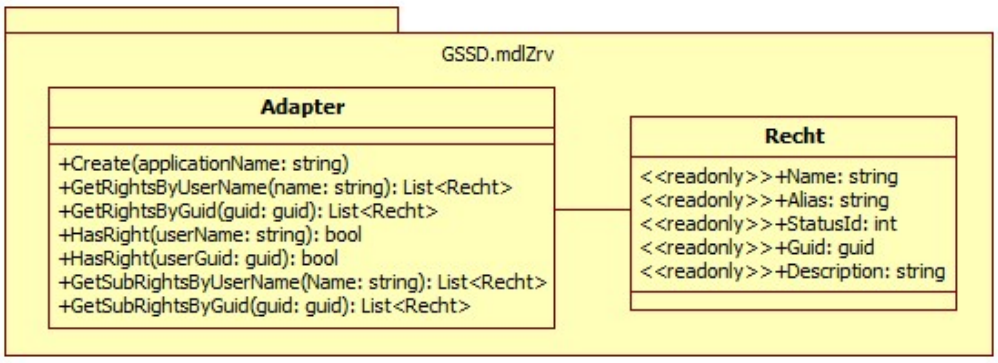

Abbildung 4: Klassendiagramm Adapter-Modul

A.6 Beispiel für eine Konfigurationsdatei

Aus Gründen der Übersichtlichkeit wurden sämtliche Attribute, bis auf den Namen, weggelassen. Die angebenen Namen reichen aus, um die entsprechenden Objekte eindeutig zu kennzeichnen. Aus der nachfolgenden XSD können zusätzliche Attribute entnommen werden. Zusätzlich zu den Namen helfen GUIDs Objekte zu identifizieren. Diese erhöhen die Redundanz, verringern jedoch das auftreten von Fehlern bzw. helfen bei der Wiederherstellung nach einem Fehler. Sofern neue Objekte angelegt und dafür GUIDs benötigt, kann z. B. auf die Webseite https://www.guidgen.com/ zurückgegriffen werden.

```
<ZRV remote="pfad/zum/zentralen/git-repository" Kunde="Testkunde">
    <liste_anwendung>
        <anwendung name="TestApp" createtime="11.11.2017 13:46:07" changetime="11.11.2017
            13:46:07" statusid="0" guid="" alias="AppStarter" description="" >
            <liste_benutzer>
                <benutzer name="template_user" />
                <benutzer name="admin" />
                <benutzer name="meier" />
            </liste_benutzer>
            <liste_rechte>
                <recht name="schreiben" />
                <recht name="lesen" />
                <recht name="aktualisieren" />
                <recht name="altdatenuebernahme" />
                <recht name="admintab" >
                    <recht name="X" />
                    <recht name="W" />
                    <recht name="R" />
                    <recht name="button" >
                        <recht name="delete" />
                        <recht name="4813" alias="save" />
                    </recht>
                </recht>
            </liste_rechte>
            <liste_benutzer_rechte>
                <benutzer_recht >
          <liste_benutzer>
            <benutzer name="admin" />
          </liste_benutzer>
                    <recht name="admintab" >
                        <recht name="button" >
                            <recht name="delete" />
                        </recht>
                    </recht>
                </benutzer_recht>
                <benutzer_recht name="stardard" >
                    <liste_benutzer>
```

```
                    <benutzer name="meier" />
                </liste_benutzer>
                <recht name="lesen" />
            </benutzer_recht>
        </liste_benutzer_rechte>
    </anwendung>
</liste_anwendung>
</ZRV>
```

A.7 XSD zum Prüfen der XML

Die folgende Darstellung entspricht der XSD, gegen die die Rechte-XML-Datei geprüft wird. Es können benötigte und optionale Attribute entnommen werden.

```
<?xml version="1.0" encoding="Windows-1252"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="
    http://www.w3.org/2001/XMLSchema" >
<!-- DECLARATION -->

  <!-- type GUID -->
  <xs:simpleType name="GUID">
    <xs:restriction base="xs:string">
      <xs:pattern value="\{[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
          fA-F0-9]{12}\}"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- type type (ENUM) -->
  <xs:simpleType name="type" final="restriction" >
    <xs:restriction base="xs:string">
      <xs:enumeration value="template" />
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="recht1">
    <xs:sequence>
      <xs:element name="recht" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element type="recht1" name="recht" maxOccurs="unbounded" />
          </xs:sequence>
          <xs:attribute name="name" type="xs:string" use="required" />
          <xs:attribute name="createtime" type="xs:date" use="required" />
          <xs:attribute name="changetime" type="xs:date" use="required" />
          <xs:attribute name="statusid" type="xs:unsignedByte" use="required" />
```

```
        <xs:attribute name="guid" type="GUID" use="required" />
        <xs:attribute name="alias" type="xs:string" use="optional" />
        <xs:attribute name="description" type="xs:string" use="optional" />
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required" />
  <xs:attribute name="createtime" type="xs:date" use="required" />
  <xs:attribute name="changetime" type="xs:date" use="required" />
  <xs:attribute name="statusid" type="xs:unsignedByte" use="required" />
  <xs:attribute name="guid" type="GUID" use="required" />
  <xs:attribute name="alias" type="xs:string" use="optional" />
  <xs:attribute name="description" type="xs:string" use="optional" />
</xs:complexType>


<xs:complexType name="benutzer_recht_recht">
  <xs:sequence>
    <xs:element type="benutzer_recht_recht" name="recht" minOccurs="0" maxOccurs="
        unbounded">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="optional" />
  <xs:attribute name="changetime" type="xs:string" use="required" />
  <xs:attribute name="statusid" type="xs:unsignedByte" use="optional" default="1" />
  <xs:attribute name="guid" type="xs:string" use="required" />
</xs:complexType>




<!-- DECLARATION END -->


<!-- BEGIN -->
<xs:element name="ZRV">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="liste_anwendung">
        <xs:complexType>
          <xs:sequence>
            <!-- type anwendung -->
            <xs:element name="anwendung" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
```

```
73                  <xs:element name="liste_benutzer" >
74                    <xs:complexType >
75                      <xs:sequence>
76                    <!-- type benutzer -->
77                        <xs:element name="benutzer" maxOccurs="unbounded">
78                          <xs:complexType>
79                            <xs:attribute name="name" type="xs:string" use="required" />
80                            <xs:attribute name="statusid" type="xs:unsignedByte" use="
                                  required" />
81                            <xs:attribute name="guid" type="GUID" use="required" />
82                            <xs:attribute name="type" type="type" use="optional" />
83                            <xs:attribute name="alias" type="xs:string" use="optional" />
84                            <xs:attribute name="description" type="xs:string" use="optional
                                  " />
85                            <xs:attribute name="windowuser" type="xs:string" use="required"
                                   />
86                            <xs:attribute name="loginuser" type="xs:string" use="optional"
                                  />
87                            <xs:attribute name="createtime" type="xs:date" use="required"
                                  />
88                            <xs:attribute name="changetime" type="xs:date" use="required"
                                  />
89                          </xs:complexType>
90                        </xs:element>
91                      </xs:sequence>
92                    </xs:complexType>
93                  </xs:element>
94                  <xs:element name="liste_rechte">
95                    <xs:complexType>
96                      <xs:sequence>
97                    <!-- type recht -->
98                        <xs:element type="recht1" name="recht" />
99                      </xs:sequence>
100                   </xs:complexType>
101                 </xs:element>
102                 <xs:element name="liste_benutzer_rechte">
103                   <xs:complexType >
104                     <xs:sequence>
105                   <!-- type benutzer recht -->
106                       <xs:element name="benutzer_recht">
107                         <xs:complexType>
108                           <xs:sequence>
109                             <xs:element name="liste_benutzer">
110                               <xs:complexType>
111                                 <xs:sequence>
```

```
112                                 <xs:element name="benutzer" maxOccurs="unbounded">
113                                   <xs:complexType>
114                                     <xs:attribute name="name" type="xs:string" use="
                                            required" />
115                                     <xs:attribute name="statusid" type="xs:unsignedByte"
                                            use="optional" default="1" />
116                                     <xs:attribute name="guid" type="GUID" use="required"
                                            />
117                                     <xs:attribute name="changetime" type="xs:date" use="
                                            required" />
118                                   </xs:complexType>
119                                 </xs:element>
120                               </xs:sequence>
121                             </xs:complexType>
122                           </xs:element>
123                           <xs:element maxOccurs="unbounded" name="recht" type="
                                benutzer_recht_recht">
124                           </xs:element>
125                         </xs:sequence>
126                       </xs:complexType>
127                     </xs:element>
128                   </xs:sequence>
129                 </xs:complexType>
130               </xs:element>
131             </xs:sequence>
132             <xs:attribute name="name" type="xs:string" use="required" />
133             <xs:attribute name="createtime" type="xs:string" use="required" />
134             <xs:attribute name="changetime" type="xs:string" use="required" />
135             <xs:attribute name="statusid" type="xs:unsignedByte" use="required" />
136             <xs:attribute name="guid" type="xs:string" use="required" />
137             <xs:attribute name="alias" type="xs:string" use="required" />
138             <xs:attribute name="description" type="xs:string" use="required" />
139             <xs:attribute name="configurationPath" type="xs:string" use="required" />
140             <xs:attribute name="path" type="xs:string" use="required" />
141           </xs:complexType>
142         </xs:element>
143       </xs:sequence>
144     </xs:complexType>
145   </xs:element>
146 </xs:sequence>
147 <xs:attribute name="remote" type="xs:string" use="required" />
148 <xs:attribute name="Kunde" type="xs:string" use="required" />
149 </xs:complexType>
150 </xs:element>
151
```

```
152 </xs:schema>
```

B Benutzerdokumentation

B.1 Zrvadapter.exe

Standard Einrichtung der Rechtedatei sowie des zentralen git-Repository im angegebenen Pfad. Alternativ kann eine bereits bestehende Rechtekonfiguration im aktuellen Verzeichnis verfügbar gemacht werden. Dafür wird der Parameter **-d** zusätzlich angegeben.

```
1 zrvadapter [--init|-i [--download|-d] --path|-p <uri/pfad/zum/zentralen/git-Repository
    >]
```

Mit dem folgenden Befehl wird die Rechtedatei hochgeladen und bereit gestellt. Es sollte vorher überprüft werden, ob Zugriff auf das Verzeichnis möglich ist.

```
1 zrvadapter [--upload|-u]
```

Herunterladen der aktuellsten Konfiguration.

```
1 zrvadapter [--download|-d]
```

Überprüfen der Rechtedatei gegen die XSD. Achtung die XSD wird durch die ZRV bereit gestellt und muss im gleichen Verzeichnis wie die Rechtedatei vorliegen.

```
1 zrvadapter [--check|-c]
```

B.2 Einrichtung des Konfigurationsverzeichnis

Zu beachten ist, das jederzeit Zugriff auf die zrvadapter.exe bestehen muss, um nachfolgende Schritte durchführen zu können.

B.2.1 Neues lokales Konfigurationsverzeichnis im aktuellen Verzeichnis anlegen

Unter Angabe des zentralen git-Repository wird im aktuellen Verzeichnis automatisch eine Standard Rechtedatei angelegt. Diese kann nach eigenen Wünschen angepasst werden. Eine Beispiel-Konfiguration im Anhang A.6: Beispiel für eine Konfigurationsdatei auf Seite V.

```
1 zrvadapter -i uri/pfad/zum/zentralen/git-Repository
```

Sofern Änderungen an der Rechtedatei vorgenommen wurden, werden diese der ZRV mit dem folgenden Befehl bereit gestellt.

```
1 zrvadapter -u
```

B.2.2 Lokales Konfigurationsverzeichnis aus bestehenden zentralen git-Repository erzeugen

Besteht bereits ein zentrales git-Repository und soll dies lokal bereitgestellt werden, wird mit folgendem Kommando die Konfigurationsdatei im aktuellen Verzeichnis anlegen. Zu beachten ist, dass der URI-Pfad keine Backslashes '\' enthalten darf, diese müssen durch Slashes '/' ersetzt werden.

```
zrvadapter -i -d uri/pfad/zum/zentralen/git-Repository
```

B.3 Benutzerrecht bzw. Konfiguration ändern

Im Folgenden wird der Workflow beschrieben, wenn der Systemadministrator der GSSD eine Änderungsanfrage von Benutzerrechten vom Kunden erhält.

1. Anruf/E-Mail geht beim Support ein, es soll ein Recht geändert werden.
   - Konfigurationsdatei muss mit dem aktuellsten Stand aus dem zentralen git-Repository abgeglichen werden

   ```
   cd uri\pfad\zum\zentralen\git-Repository
   zrvadapter.exe -d
   ```

2. Öffnen der Rechtedatei bzw. grafischen Benutzeroberfläche.
   - Entweder mit grafischen Texteditor benutzen
   - oder per Kommandozeile

   ```
   notepad.exe rechte.xml
   ```

3. Aus der Liste der Programme das entsprechende Suchen und im Unterzweig Konfiguration nach dem zugrundeliegenden Schema anpassen und speichern.
   - Prüfen der Rechtedatei gegen die XSD

   ```
   zrvadapter -c
   ```

   - Gibt es einen Fehler, muss dieser korrigiert werden oder mit B.3.1 fortfahren.
   - Im Anschluss wird die neue Version in das zentrale git-Repository geladen, um diese anderen Programmen bereit zu stellen

   ```
   zrvadapter.exe -u
   ```

4. Die Rückmeldung an den Kunden erfolgt.

B.3.1 Wiederherstellung einer älteren Version

Die Wiederherstellung einer alten Version, kann nur durch den Systemadministrator durchgeführt werden, hierbei sind Kenntnisse über git notwendig. Zuerst wird eine Konsole geöffnet und in das Verzeichnis mit der lokalen Version der Rechtedatei gewechselt sowie folgender Befehl zu Anzeigen der letzten Versionsstände eingegeben.

```
1 git log --pretty=format:"[%h] [%an] [%ai] | %s%d"
```

Der erste, in eckigen Klammern stehende, Ausdruck, ist ein Hashwert, mit dem diese Version identifiziert werden kann.
Letzte Änderungen können mit dem nachstehenden Befehl anzeigt werden.

```
1 git diff
```

Zeilen, die mit einem Plus anfangen, stehen für neue bzw. hinzugefügte Zeilen und ersetzen darüber stehende Zeilen, sofern diese mit einem Minus anfangen.

Um die Rechtedatei im Ganzen zu sehen, kann der nachfolgende Befehl eingegeben werden. Der sogenannt Hashwert (in diesem Beispiel 288935d) ist durch einen gültigen zu ersetzt.

```
1 git show 288935d:rechte.xml
```

Es kann der Inhalt kopiert oder alternativ (siehe nächsten Befehl) in eine Datei geschrieben werden. Achtung die temporäre Datei darf nicht "$rechte.xml$" benannt werden.

```
1 git show 288935d:rechte.xml > tmp.xml
```

Sobald Änderungen an der Rechtedatei durchgeführt worden sind, müssen diese gespeichert und bereitgestellt werden. Der zrvadapter.exe wird dazu benutzt.

```
1 zrvadapter -u
```