Entwicklung eines Programms zur Verwaltung und Dokumentation von Firewallpolicys nach 21CFR11

PROJEKTBERICHT
über das erste Praxisjahr

im Studiengang TIT09

an der DHBW Ravensburg
(Campus Friedrichshafen)

von

Florian Peschka

08.10.2010

Bearbeitungszeitraum	05.07.2010 – 01.10.2010
Matrikelnummer	6192194
Partnerunternehmen	TANNER AG Lindau
Betreuer im Partnerunternehmen	Christian Ostermeier

Erklärung

gemäß § 5 (2) der „Studien- und Prüfungsordnung DHBW Technik" vom 18. Mai 2009.

Hiermit erkläre ich, dass ich die vorliegende Arbeit mit dem Titel

Entwicklung eines Programms zur Verwaltung und Dokumentation von Firewallpolicys nach 21CFR11

selbständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt, keine anderen als die angegebenen Hilfsmittel benutzt und wörtliche sowie sinngemäße Zitate als solche gekennzeichnet habe.

Lindau, den 01.10.2010

Florian Peschka

Inhaltsverzeichnis

1 Einleitung

Firewalls sind ein enorm wichtiger Bestandteil von IT-Firmen und anderen Unternehmen, welche sich Computernetzwerke zu Nutze machen. Diese Firewalls schützen Benutzer vor Angriffen von anderen Servern, Netzwerken oder aus dem Internet selbst. Sie können aber auch die Zugriffe dieser Benutzer auf Server außerhalb ihres Netzwerkes einschränken, um den Netzwerkadministratoren eine Möglichkeit zu geben, die Zugriffe der Benutzer verwalten und nachvollziehen zu können.

Die Verwaltung und Zugriffsdokumentation der Firewall ist ein wichtiger Bestandteil moderner Netzwerkadministration. Damit dies auch für versierte Administratoren eine komfortable Arbeit ist, beschäftigt sich dieser Projektbericht mit der „Erstellung eines Programms zur Verwaltung und Dokumentation von Firewallpolicys nach 21CFR11".

Besonders im professionellen Bereich ist die Überwachung von so genannten „Policys" enorm wichtig. Diese definieren, welche Benutzer(gruppen) welche Art von Zugriffsrechten auf bestimmte, durch die Firewall geschützte Server und Netzwerke besitzen.

Diese Policys definieren sozusagen die Türen, durch welche Benutzer zu den Servern gelangen, welche sie anfragen. Benutzer und Server sind durch die Firewall voneinander getrennt. In der Firewall gespeichert sind die Policys, welche für jeden Benutzer und für jeden Server regeln, ob und wie die beiden miteinander kommunizieren dürfen.

Dabei kann eine Verbindung von beiden Seiten aus gestartet werden, d.h. entweder fragt der Benutzer eine Information auf dem Server an oder der Server sendet dem Benutzer Daten.

Das folgende Diagramm zeigt die Funktionsweise einer Policy beispielhaft. Hierbei ist zu beachten, dass die Verbindung wie bereits erwähnt auch vom Server ausgehen kann und nicht zwingend nur vom Benutzer.

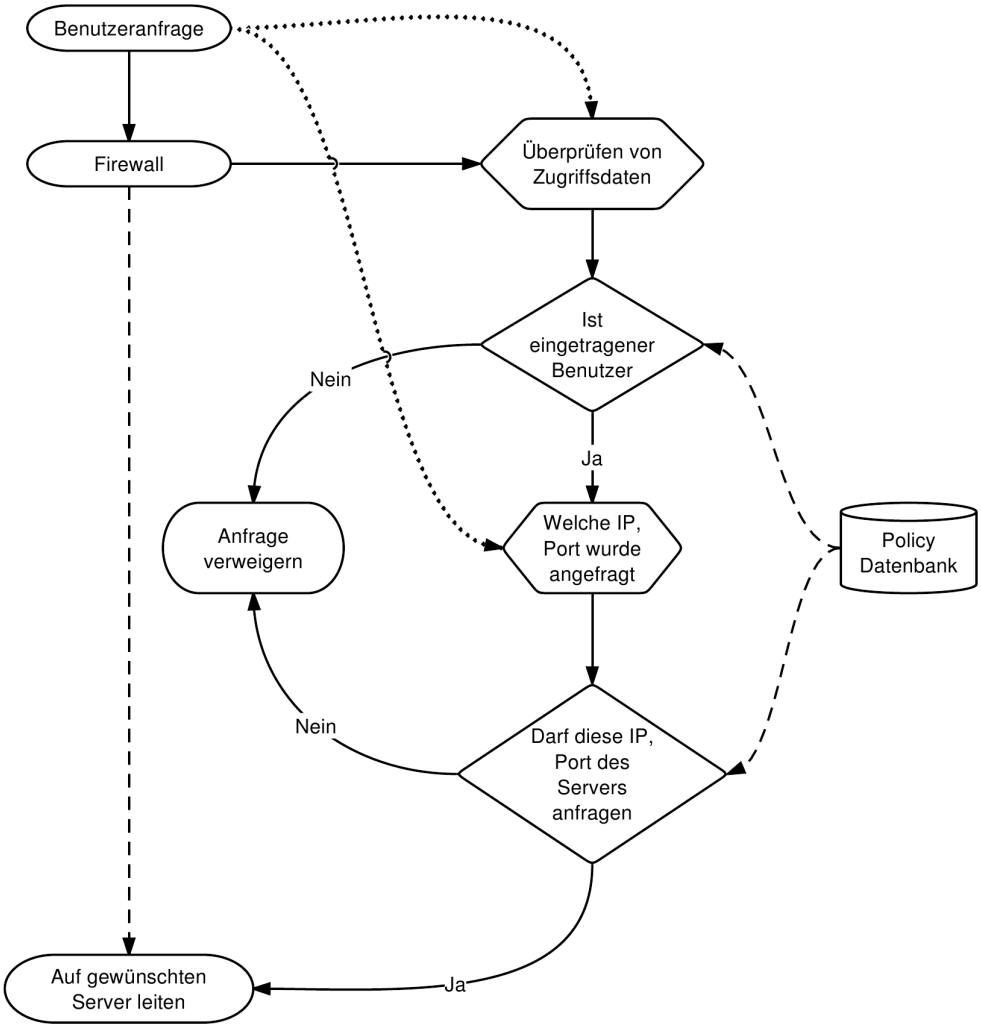

Abb. 1: Funktionsweise einer Policy

Um eine möglichst durchgängige und exakte Dokumentation dieser Policys sicherzustellen, ist das Ziel dieses Projekts, eine benutzerfreundliche Oberfläche zu erstellen, die den Administratoren und Zuständigen der Firewall eine einfache und intuitive Möglichkeit gibt, die Policys zu verwalten und deren Historie nachzuvollziehen.

Diese Dokumentation muss dem „21CFR11"[1] standhalten. Dabei handelt es sich um ein amerikanisches Gesetz, welches sich im „Code of Federal Regulations" befindet. Titel 21 dieses Gesetzes behandelt Regelungen für die amerikanische „Food and Drug Administration" – Teil 11 dieses Titels definiert „die Kriterien, nach denen elektronische Aufzeichnungen und Signaturen als vertrauenswürdig, verlässlich und gleichbedeutend zu Papieraufzeichnungen"[2] gelten.

[1] Gesamter Titel: Code of Federal Regulations Title 21 Chapter I Part 11
[2] Frei übersetzt von 21CFR11 Subpart A Sec 11.1 Sentence (a)

2 Zielsetzung

2.1 Allgemeines

Das Programm (getauft „FirewallDoc“) dient als Plattform für die Dokumentation der oben erklärten Policys. Es greift nicht direkt in die Firewall ein, sondern wird parallel dazu benutzt. Die Policys, welche in die Firewall eingetragen werden, müssen demnach in gleicher Form in dieses Programm eingepflegt werden, um die Dokumentation sicherzustellen.

Der wichtigste Aspekt dieser Dokumentation wird die Rückverfolgbarkeit sein, um auch im Nachhinein die Wirkungsweise der Policys zu einem bestimmten Datum nachvollziehen zu können.

Der Einsatz wird für die Firewall erfolgen, welche sich zwischen den Mitarbeitern des Kunden und dem CMS-System auf den Servern der TANNER AG befindet. Das Programm selbst wird auf einem Server installiert werden, auf welchem sowohl der Kunde als auch das TANNER-Team Zugriff erhält, da diese ebenfalls an der Firewallverwaltung beteiligt ist.

Beide Parteien müssen den Umgang mit dem Programm beherrschen und werden es aktiv verwenden. Dabei sind die Verantwortlichen bei TANNER das Support-Team, welches den Telefonsupport für den Kunden auch regulär leitet.

Der Kunde sowie das Support-Team müssen die Möglichkeit haben, auf eine umfangreiche Dokumentation sowohl der benutzerbezogenen Aspekte als auch der programmiertechnischen zuzugreifen, um gegebenenfalls Programm-erweiterungen oder –anpassungen zu entwickeln.

Das Programm dient auf lange Sicht dem Zweck, eine vollständige Dokumentation über alle Firewall-Polices zu gewährleisten, die dem „21CFR11“ gerecht wird.

Die Protokollierung der Firewallpolicys muss den in diesem Gesetz beschriebenen Richtlinien Rechnung tragen. Daher kann das momentan vom Kunden gepflegte, manuelle Eintragen von Policys nicht mehr weiter ausreichen, um die Richtlinien zu erfüllen.

2.2 Anforderungen

Die Anforderungen an das Programm sind größtenteils durch die Definitionen des „21CFR11" festgelegt. Anhand dieser Definitionen wurden die Anforderungen an das Programm erstellt und wurde versucht, die Regelungen des „21CFR11" so genau wie möglich einzuhalten.

Die folgende Liste zeigt die wichtigsten Regelungen des „21CFR11", die für dieses Projekt eine besondere Relevanz für die Umsetzung aufweisen[3].

- Elektronische Dokumente müssen „in einer exakten und vollständigen Form sowohl für Menschen lesbar als auch in einer elektronisch passenden Art und Weise kopierbar sein"[4].

- Das System soll „computererstellte und mit einem Zeitstempel versehene Aufzeichnungen der Tätigkeiten [bereitstellen, um] alle Tätigkeiten, die elektronische Aufzeichnungen erstellen, löschen oder modifizieren"[5], nachvollziehen zu können.

- Es muss ein System eingeführt werden, das „einen Benutzer anhand seiner einzigartigen elektronischen Signatur identifiziert [... Es muss sichergestellt sein, dass] keine zwei Personen die gleiche Kombination aus Identifikation und Passwort besitzen"[6]. Diese Personen müssen „identifiziert werden, um sicherzustellen, dass nur autorisierte Personen das System nutzen können"[7].

[3] Die komplette Liste ist dem Anhang zu entnehmen.
[4] Frei übersetzt von 21CFR11 Subpart A Section 11.1 Sentence (a)
[5] Frei übersetzt von 21CFR11 Subpart B Section 11.10 Sentence (e)
[6] Frei übersetzt von 21CFR11 Subpart C Section 11.100 Sentence (b)
[7] Frei übersetzt von 21CFR11 Subpart B Section 11.10 Sentence (g)

- Die Einführung von „Richtlinien, die Personen unter Verwendung ihrer elektronischen Signatur für bestimmte Tätigkeiten als verantwortlich kennzeichnen“[8].

Das Definieren der Anforderungen wurde in Abstimmung mit dem Auftraggeber und dem zuständigen Projektmanager durchgeführt. Hierzu ist ein internes Projektmanagementprogramm zum Einsatz gekommen, welches allen am Projekt beteiligten Personen die Möglichkeit bietet, Anforderungen, Unit-Tests und sonstige für die Durchführung eines Projektes wichtige Einzelheiten zentral zu verwalten und zu dokumentieren.

Bei den Anforderungen handelt es sich um eine kundenorientierte Zusammenstellung von Anforderungen, d.h. es werden keinerlei Angaben darüber gemacht, wie eine bestimmte Funktion umgesetzt werden soll oder mit welchem System das Programm erstellt werden soll. (Es sei denn, dieses System selbst wird als Anforderung erfasst.)

Der Auftraggeber beschreibt in normalen Worten nur die Funktionen, die das Programm seiner Meinung nach haben soll, ohne direkten Bezug auf technische Details zu nehmen. (Wiederum: Es sei denn, diese Details sind von Bedeutung für die Funktionalität des Programms.)

Für dieses Projekt wurden folgende Anforderungen erstellt, durch den Auftraggeber abgesegnet und dementsprechend festgelegt.

- **Administration: Datenbank**
  Die Datenbank, aus welcher das Programm alle nötigen Informationen zieht, muss eine Microsoft-Access-2000-Datenbankdatei sein, um die manuelle Einsicht der Daten auch ohne das Programm selbst zu ermöglichen oder sie manipulieren zu können.

[8] Frei übersetzt von 21CFR11 Subpart B Section 11.10 Sentence (j)

- **Administration: Aktionslog**
  Ein Aktionslog muss eingebunden werden, mit dem man alle Aktionen zwischen zwei wählbaren Daten nachvollziehen kann:
  - Wann wurde die Aktion ausgeführt
  - Wer hat die Aktion ausgeführt
  - Welche Aktion wurde ausgeführt
  - Welche Daten wurden der Aktion übergeben

- **Benutzer-Verwaltung: Benutzerdaten**
  Die Daten jedes Benutzers sind:
  - Name (für den Login, einzigartig)
  - Passwort (für den Login)
  - E-Mail (für eventuelle Kontaktaufnahme)
  - Berechtigungen für bestimmte Programmteile

- **Benutzer-Verwaltung: Berechtigungen**
  Jedem Benutzer müssen Berechtigungen gegeben oder verweigert werden können, die näher einschränken, welche Rechte er zur Bedienung des Programms hat. Die Berechtigungen eines Benutzers schränken seine Aktionsfreiheit innerhalb des Programms ein. Folgende Berechtigungen sind für bestimmte Aktionen festegelegt:
  - Zugriff auf Berichte
  - Zugriff auf Policy-Verwaltung
  - Zugriff auf Benutzer-Verwaltung
  - Zugriff auf Aktionslog

- **Benutzer-Verwaltung: Bearbeitung / Löschung**
  Benutzer müssen im Nachhinein bearbeitet und gelöscht werden können.

- **Policy-Verwaltung: Erstellung / Aktivierung / Deaktivierung**
  Bei jeder Aktion, welche die Policys betrifft, muss sichergestellt sein, dass die Änderungen dokumentiert werden, damit die Berichte nicht verfälscht werden.

D.h. nach einer Löschung darf die Policy nicht physikalisch gelöscht werden, sondern nur auf einen Status gesetzt werden, der erneute Änderungen verhindert. Damit soll gewährleistet werden, dass einmal aktivierte Policys in ihrer Form immer nachvollziehbar bleiben, auch nach ihrer Deaktivierung.

Bei der Erstellung kann der Benutzer mehrere Gruppen auswählen, die zu dieser Policy gehören. Dabei werden auch IPs angezeigt, die keiner Gruppe angehören.

- **Policy-Verwaltung: Policy-Eigenschaften**
  Eine Policy besteht aus folgenden Eigenschaften, die ihre Wirkungsweise definieren:
  - Datum (Angefragt)
    - An diesem Datum wurde die Policy von einem Verantwortlichen zur Freischaltung beantragt.
  - Datum (Aktiviert)
    - An diesem Datum wurde die Policy auf dem Firewallserver aktiviert.
  - Datum (Deaktiviert)
    - An diesem Datum wurde die Policy deaktiviert.
  - Auftraggeber
    - Der Mitarbeiter, der die Einrichtung der Policy beantragt hat.
  - Verantwortlicher
    - Der Mitarbeiter, der für die Einrichtung der Policy auf dem Firewallserver verantwortlich war / ist.
  - Client(s)
    - Die IP(s) oder IP-Ranges der Mitarbeiterrechner, welche durch diese Policy überwacht werden.
      Diese bestehen aus den Gruppen oder IPs der Policy-Verwaltung-Anforderung.
  - Client Port(s)
    - Die Ports oder Port-Ranges, auf welche die Clients ihre Anfragen schicken.

- Server Port(s)
  - Die Ports oder Port-Ranges der Server, auf welche die Anfragen der Clients geleitet werden.
- Server(s)
  - Die IP(s) oder IP-Ranges der Server hinter der Firewall, welche durch diese Policy überwacht werden.
- Richtung
  - Die Richtung, welche diese Policy überwacht (Client -> Server, Server -> Client oder beide).
- Kommentar
  - Ein kurzer Kommentar zur Funktion / zum Grund für diese Policy,

- **Policy-Verwaltung: IPs**
  IPs müssen einzeln verwaltbar sein, um ihnen Namen oder Kommentare geben zu können. Dabei handelt es sich ausschließlich um die IPs der Mitarbeiter auf Clientseite.

- **Policy-Verwaltung: Gruppen**
  Gruppen fassen mehrere IPs unter einem Namen zusammen. Sie bündeln damit mehrere Mitarbeiter in einer Clientgruppe, für welche Policys geltend gemacht werden.

- **Berichte: Vorschau im Programm**
  Die Vorschau soll die Policys anzeigen, welche später per PDF exportiert werden können. Dabei muss ebenfalls ein Datum ausgewählt werden, zu dem der Bericht erstellt wird.

- **Berichte: PDF-Export**
  Die Berichte müssen als PDF exportiert werden können, in welchem die gleichen Daten angezeigt werden, die in der Berichtvorschau ebenfalls sichtbar sind. Das Layout des PDFs wird an das Standard-dokumentenlayout der TANNER AG angelehnt.

Die Anforderungen wurden an den Auftraggeber und an das TANNER-Team weitergegeben und nach mehrmaligen Korrekturen in ihren oben vereinfacht gezeigten Formen freigegeben.

Die Anforderungen definieren den Funktionsumfang aus Sicht des Benutzers, respektive des Kunden. Um die Anforderungen in programmierbare Elemente umzuwandeln, folgt danach der Schritt der Komponentenerstellung.

Diese bilden die Schnittstelle zwischen den Anforderungen des Auftraggebers und den Aufgaben des Programmierers, der das Programm schlussendlich entwickeln muss.

3 Umsetzung

3.1 Komponenten

Um die Anforderungen aufzuteilen und effizient zu programmieren, wird das Programm in möglichst kleine Einheiten (sog. Komponenten) aufgeteilt, deren Gesamtheit das Programm ergibt.

Komponente	**Zugeordnete Anforderungen**
• Policy-Verwaltung	Policy-Verwaltung
○ Erstellung	Policy-Eigenschaften
○ Bearbeitung	...
○ Deaktivierung	...
○ Gruppen	...
○ IPs	...
• Benutzer-Verwaltung	Benutzer-Verwaltung
○ Login-System	Berechtigungen
○ Erstellung	Benutzerdaten
○ Bearbeitung	Bearbeitung
○ Löschung	Löschung

- Administration
  - Aktionslog — Aktionslog
  - Datenbank — Datenbank

- Berichte — Berichte
  - Ansicht im Programm — ...
  - PDF-Export — ...

Die Gesamtheit der Komponenten bilden das Programm – die Unterteilung erleichtert lediglich die Aufteilung der Aufgaben für das Entwicklerteam. So kann z.B. ein Entwickler die „Administration" übernehmen, während ein anderer die „Policy-Verwaltung" programmiert.

Komponenten haben eine äußere Ansicht und eine innere. Diese Ansicht beschreibt, wie die Anzeige der Komponente strukturiert ist, also z.B. dass die Benutzer in einer Liste angezeigt werden, die aus den Benutzernamen besteht.

Die Komponenten werden später gekapselt getestet. Da sie die Teile des Programms widerspiegeln, ist es möglich, diese Teile als Einheiten und getrennt von den anderen auf korrekte Funktionsweise hin zu testen.

3.2 Systemanforderungen

Die Systemanforderungen legen fest, wie die Komponenten aufgebaut werden und welchen Anforderungen (siehe 2.2) sie gerecht werden müssen. Dabei wird unter anderem der Ablauf von Prozessen, die Sicht des Benutzers und die Sicht des Programmierers beschrieben.

Jede Systemanforderung wird einer (oder mehreren) Komponenten zugeordnet, die diese Anforderung beinhaltet. Dies spiegelt im Grunde das daraus entwickelte Klassendiagramm des Programms.

Wichtig bei den Systemanforderungen ist, dass sie eine „Daseinsberechtigung" haben müssen. D.h., jede Systemanforderung muss durch eine Anforderung (siehe 2.2) abgedeckt sein und diese Anforderung erfüllen.

Dies verhindert, dass Programmteile erstellt werden, die den eigentlichen Anforderungen des Programms zuwider laufen oder gar nicht erst gewünscht sind. Die Kopplung der Anforderungen untereinander ermöglicht, dass einfach geprüft werden kann, ob alle Anforderungen erfüllt werden.

Wann eine Systemanforderung als erfüllt gilt, muss in der Systemanforderung festgelegt werden. Z.B. ist die Systemanforderung zur „Benutzererstellung" dann erfüllt, wenn der Benutzer korrekt in der Datenbank gespeichert wird und die Benutzerliste aktualisiert wird und dann den neu erstellten Benutzer anzeigt.

Die festgelegten Systemanforderungen inklusive deren Verbindung zu den Anforderungen und die Einordnung in die Komponenten des Programms werden hier dargestellt.

Komponente: Benutzer-Verwaltung

Systemanforderung: Login-System

Beim Start des Programms muss sich der Benutzer mit seinen Daten einloggen, um Zugriff auf die Funktionen zu haben. Je nachdem, welche Berechtigungen er hat, erhält er die einzelnen Module des Programms zur Ansicht und Verwaltung.

Wenn er keinen Zugriff auf ein bestimmtes Modul hat, wird es ausgeblendet, deaktiviert oder ist nicht funktional.

Systemanforderung: Erstellung

Beim Erstellen eines Benutzers müssen die erforderlichen Daten (Name, Passwort, E-Mail, Berechtigungen) in die Datenbank geschrieben werden.

Das Zuweisen von Berechtigungen und einer E-Mail-Adresse muss nicht zwingend erfolgen, es kann auch möglich sein, Benutzer ohne Rechte zu erstellen.

Alle anderen Daten müssen zwingend ausgefüllt werden, bevor der Benutzer erstellt werden kann.

Systemanforderung: Bearbeitung

Das Bearbeiten eines Benutzers ermöglicht die Änderung aller Daten dieses Benutzers. Wenn die Änderungen gespeichert werden, sind sie sofort in der Datenbank aktiv.

Ein Benutzer kann seinem eigenen Benutzereintrag keine neuen Rechte geben und keine alten Rechte verwehren. Er kann nur die Felder „Name", „Passwort" und „E-Mail" ändern.

Ändert ein Benutzer eines dieser Felder seines eigenen Benutzereintrags, wird er aus dem Programm ausgeloggt und muss sich mit den neuen Daten erneut einloggen.

Systemanforderung: Löschung

Das Löschen eines Benutzers muss zuerst bestätigt werden, um die versehentliche Löschung zu verhindern.

Ein Benutzer kann seinen eigenen Benutzereintrag nicht löschen.

Komponente: Policy-Verwaltung

Systemanforderung: Policy-Eigenschaften

Jede Policy wird durch die Gesamtheit ihrer Eigenschaften definiert. Diese sind wie bei den Anforderungen beschrieben einzubinden.

Eine Policy durchläuft immer einen festgelegten Prozess:

Erstellung -> Aktivierung -> Deaktivierung

Dieser Prozess muss eingehalten werden, d.h. man kann eine Policy nicht schon bei der Erstellung als aktiviert markieren oder eine noch nicht aktivierte Policy bereits als deaktiviert kennzeichnen.

So wird sichergestellt, dass die Historie jeder Policy eindeutig nachvollziehbar ist und auch im Nachhinein (beim Erstellen der Berichte) korrekte Daten ausgewertet werden.

Systemanforderung: Erstellung

Beim Erstellen einer Policy wird sie in den „Anfrage"-Status gesetzt. Alle Daten können geändert werden, solange sie in diesem Status ist.

Die Policy kann sowohl Gruppen als auch einzelne, noch nicht in Gruppen zusammengefasste IPs verwalten.

Systemanforderung: Aktivierung

Durch das Aktivieren einer Policy werden ihre Eigenschaften gespeichert und können im Nachhinein nicht mehr geändert werden.

Systemanforderung: Deaktivierung

Das Deaktivieren einer Policy ist gleichbedeutend mit der Löschung der Policy auf der Firewall. Sie ist nicht mehr aktiv und wird nur gespeichert, um rückwirkend über die Berichte nachvollziehen zu können, wann sie aktiv war.

Systemanforderung: Gruppen

Die Gruppen bündeln mehrere IPs unter einem Namen. Jede Gruppe besteht aus einer oder mehreren IPs und dem Gruppennamen. Dieser wird bei der Erstellung von Policys angezeigt.

Systemanforderung: IPs

Eine IP besteht aus ihrem Wert und einem Kommentar zu dieser IP. IPs können zu Gruppen zusammengefasst werden, müssen aber nicht. Wenn eine IP keiner Gruppe zugewiesen ist, wird sie bei der Erstellung einer Policy als eigenständige IP angezeigt.

Komponente: Administration

Systemanforderung: Datenbank

Die Datenbank, welche Benutzer, Policys und Aktionslog speichert, ist eine Microsoft-Access-2000-Datenbankdatei (*.mdb).

Dies soll die manuelle Anpassung von Daten ermöglichen, falls korrupte Datensätze oder anderweitige Probleme auftreten.

Systemanforderung: Aktionslog

Der Aktionslog zeigt die mitgeschriebenen Aktionen aller Benutzer an, welche innerhalb eines auswählbaren Zeitraumes ausgeführt wurden.

Dabei werden der Benutzer angezeigt, die Aktion, der Zeitpunkt sowie möglicherweise zusätzliche Felder, welche die Aktion genauer definieren.

Komponente: Berichte

Systemanforderung: Vorschau im Programm

Nach der Auswahl eines Datums wird eine Vorschau des Berichtes angezeigt, den der Benutzer dann als PDF exportieren kann.

Die Anzeige zeigt die gleichen Policys an, welche auch in der PDF-Datei gespeichert werden und ähnelt dem Layout der PDF-Datei.

Systemanforderung: PDF Export

Das Exportieren speichert alle Policys in einer PDF-Datei, die zu dem ausgewählten Datum aktiv waren und zeigt sie mit den Daten an, welche die Policys zu diesem Zeitpunkt hatten.

Das Layout des PDFs orientiert sich an der CI der TANNER AG um die Zuordnung zu vereinfachen.

Wenn alle Systemanforderungen erfasst wurden, werden die Komponenten programmiert. Dabei richtet sich der Programmierer ausschließlich nach den Angaben in den Systemanforderungen und nicht nach denen in den Anforderungen.

Dies trennt die Programmierung gänzlich vom Kunden, da dieser nur beschreiben muss, was das Programm tun soll, und sich der Programmierer nur danach richten muss, was die Systemanforderungen verlangen.

Das Bindeglied zwischen dem Programmierer und dem Kunden ist hier der Projektmanager. Er bespricht die Anforderungen mit dem Kunden und „übersetzt" diese in Form der Systemanforderungen für den Programmierer.

Um die Integrität der Komponenten im Programm und auch die Erfüllung der Systemanforderungen und der normalen Anforderungen sicherzustellen, müssen Komponenten auf die in den Anforderungen definierten Kriterien hin getestet werden.

3.3 Unit-Tests

Als Unit-Tests bezeichnet man Tests, welche auf einzelne Komponenten ausgeführt werden, um Programmfehler, nicht vorhergesehenes Verhalten und allgemeine Logikfehler frühzeitig zu erkennen.

Diese Tests werden gekapselt auf die Komponenten ausgeführt, was den Vorteil bringt, dass man die Fehler schneller erkennt und nicht erst dann, wenn das Programm bereits vollständig entwickelt und zusammengesetzt ist.

Die Struktur der Unit-Tests schreibt vor, welche Ergebnisse eine Komponente bei Eingabe von Daten liefern muss, welches Verhalten erwartet wird und (im Fehlerfall) welches Verhalten oder welche Daten die Komponente als Fehlermeldung ausgibt.

Die Unit-Tests werden sowohl mit korrekten Daten ausgeführt (um zu prüfen, ob die Komponente richtig arbeitet) als auch mit falschen (um zu prüfen, wie die Komponente mit fehlerhaften Daten umgeht).

Nachfolgend sind beispielhaft Unit-Tests aufgelistet, die zur Verifizierung der Komponenten in diesem Projekt erstellt wurden. Die Gesamtheit der Tests aufzuzeichnen würde den Rahmen dieses Berichts deutlich sprengen, jedoch sind alle hier nicht ausführlich aufgezeigten Unit-Tests am Ende der Liste stichwortartig beschrieben.

Komponente: Benutzerverwaltung

Beschreibung:

Erstellung eines Benutzers mit einem Namen, der noch nicht verwendet wird.

Verwendete Daten:

Name: „FloPes"

Passwort: „Test123"

E-Mail: „florian.peschka@tanner.de"

Rechte: [X] Zugriff auf Berichte

[X] Zugriff auf Policys

[X] Zugriff auf Benutzer

[X] Zugriff auf Aktionslog

Arbeitsablauf:

- Eingeben der Daten in die entsprechenden Felder
- Bestätigen des Passwortes im zweiten Passwortfeld
- Anklicken der Rechte
- Klicken auf „Speichern"

Erwartetes Verhalten:

Der Benutzer wird in die Datenbank geschrieben. Der Benutzer kann sich ab nun im Programm einloggen und nur die ihm zugewiesenen Programmteile sehen bzw. mit ihnen arbeiten.

Komponente: PDF-Export von Berichten

Beschreibung:

Exportieren eines Berichts für ein bestimmtes Datum als PDF-Datei.

Verwendete Daten:

Datum: 01.01.2010 12:00

(Und einige Testpolicys, welche zu diesem Datum aktiv waren)

Arbeitsablauf:

- Auswählen des Datums
- Klicken auf „Als PDF exportieren"
- Speichern der PDF-Datei
- Überprüfen des Inhalts der Datei

Erwartetes Verhalten:

Nach der Datumsauswahl wird die Vorschau des Berichtes aktualisiert und zeigt die in Frage kommenden Policys zu ihrem entsprechenden Stand an.

Es werden nur Policys angezeigt, die an dem gewählten Zeitpunkt aktiv waren.

Beim Klick auf „Exportieren" wird ein Fenster geöffnet, in welchem man auswählen kann, wohin der Bericht gespeichert werden soll. Der Dateiname wird automatisch generiert, kann aber geändert werden.

Die Policys, welche im PDF-Dokument gespeichert sind, entsprechen denen in der Vorschau des Programms.

Komponente: Aktionslog

Beschreibung:

Auswählen des Zeitraumes, aus dem die Aktionen der Benutzer angezeigt werden soll, jedoch mit umgekehrter Reihenfolge.

D.h. das Datum „Von" wird nach dem Datum „Bis" gewählt – dadurch entsteht ein logischer Fehler.

Verwendete Daten:

Von: 01.12.2010 12:00

Bis: 01.01.2010 12:00

Arbeitsablauf:

- Auswählen der beiden Daten in falscher Reihenfolge
- Erneuern der Aktionsloganzeige

Erwartetes Verhalten:

Die Anzeige zeigt keine Aktionen an, da der Zeitraum logisch einer negativen Zeitspanne entspricht, in welcher folgerichtig keine Aktionen stattgefunden haben können.

Weitere Unit-Tests sind hier zusammengefasst beschrieben.

- **Benutzer-Verwaltung**
  - Erstellen eines Benutzers mit einem Namen, der bereits verwendet wird
  - Erstellen eines Benutzers ohne Passwort
  - Erstellen eines Benutzers mit falschem Bestätigungspasswort
  - Bearbeiten eines Benutzers mit neuem Namen, der noch nicht verwendet ist
  - Bearbeiten eines Benutzers ohne Passwort
  - Bearbeiten eines Benutzers mit falschem Bestätigungspasswort
  - Bearbeiten eines Benutzers mit neuem Namen, der bereits verwendet wird
  - Löschen eines Benutzers

- **Policy-Verwaltung**
  - Erstellen einer Policy
  - Aktivieren einer Policy
  - Deaktivieren einer Policy
  - Erstellen einer neuen IP
  - Erstellen einer neuen IP mit falschen IP-Werten
  - Erstellen einer neuen Gruppe
  - Erstellen einer neuen Gruppe ohne zugewiesene IP
  - Erstellen einer neuen Gruppe, deren Name bereits verwendet wird
  - Bearbeitung einer deaktivierten Policy
  - Filtern der Policys nach Kriterien
  - Anzeigen einer Policy anhand ihrer ID

- **Berichte**
  - Vorschau im Programm aktualisieren

- **Administration**
  - Datenbankmanipulation außerhalb des Programms (Direkt an der Datenbank-Datei)

Alle Unit-Tests wurden überprüft und die Implementierung der Komponenten im Programm so angepasst, dass jeder Unit-Test zu dem erwarteten Verhalten führt.

Hierbei waren gerade im Bereich der Policy-Verwaltung enorme Änderungen am ursprünglichen Code nötig, um die Nachvollziehbarkeit bei den Berichten auch bei deaktivierten Policys im Nachhinein fehlerfrei zu gewährleisten.

3.4 Use-Cases

Use-Cases sind, wie die Übersetzung sagt, „Benutzungsfälle". D.h. sie beschreiben Szenarien, die bei der Benutzung des Programms auftreten können, oder Prozesse, die der Benutzer befolgen muss um bestimmte Aufgaben zu erledigen.

Bei den Use-Cases wird das Programm als „Black Box" gehandhabt, d.h. es ist von außen nicht ersichtlich, welche Funktionen, Klassen und Prüfungen das Programm (respektive die Komponente, für die der Use-Case geschrieben wurde) durchführt und zu welchen Ergebnissen sie führen.

Sie werden in diesem Sinne aus Benutzersicht geschrieben und versuchen, die intuitive Bedienung des Programms aus dessen Perspektive zu beschreiben und zu definieren, welchen Arbeitsabläufen er folgen muss, um seine Aufgaben zu erledigen.

Im Rahmen des Programms dieses Projektberichts werden hier einige Use-Cases beispielhaft aufgezeigt.

Komponente: Administration

Beschreibung:

Anzeigen von Aktionen der Benutzer innerhalb eines bestimmten Zeitraumes.

Vorheriger Zustand:

Das Feld mit den Aktionen ist leer oder durch vorherige Abfragen mit Aktionen gefüllt, die zu dem vorherig abgefragten Zeitraum passen.

Arbeitsablauf:

- Auswählen des Zeitraumes über die beiden Datumsfelder
- Klicken des „Anzeigen"-Buttons

Besonderheiten:

Wenn die Datumsfelder auf das gleiche Datum zeigen oder die Reihenfolge der Daten (Von -> Bis) vertauscht wurde (Das „Von"-Datum ist nach dem „Bis"-Datum), werden keine Aktionen angezeigt, da keine gefunden werden können.

Ergebniszustand:

Das Feld mit den Aktionen ist mit allen Aktionen gefüllt, die innerhalb des ausgewählten Zeitraumes von den Benutzern ausgeführt wurden.

Komponente: Benutzer-Verwaltung

Beschreibung:

Bearbeiten des eigenen Benutzerkontos zwecks Änderung des Passwortes.

Vorheriger Zustand:

Die Felder der Benutzer sind leer oder mit den Daten eines anderen, vorher ausgewählten Benutzers gefüllt.

Arbeitsablauf:

- Auswählen des eigenen Benutzernamens aus der Liste aller Benutzer
- (Die Felder werden mit den eigenen Daten gefüllt)
- Eintippen eines neuen Passwortes in das Passwortfeld sowie das Bestätigungsfeld
- Klicken des „Speichern"-Buttons

Besonderheiten:

Das Passwort muss in beiden Feldern identisch sein, andernfalls führt ein Speichern zu einer Fehlermeldung.

Ergebniszustand:

Das neue Passwort ist überprüft und in der Datenbank gespeichert. Der Benutzer wird automatisch ausgeloggt, das Hauptfenster geschlossen und das Loginfenster angezeigt.
Er muss sich mit dem neuen Passwort einloggen.

Komponente: Policy-Verwaltung

Beschreibung:

Deaktivieren einer aktivierten Policy.

Vorheriger Zustand:

Die Felder der Policy, welche ihre Eigenschaften definieren, sind ausgegraut und können nicht angeklickt oder bearbeitet werden.
Einzig die Checkbox neben dem Datumsfeld zur Deaktivierung ist bearbeitbar.

Arbeitsablauf:

- Doppelklick auf die Policy in der Policy-Übersicht
- (Die Policy wird geöffnet und ihre Eigenschaften angezeigt)
- Aktivieren der Checkbox neben dem Datumsfeld zur Deaktivierung
- (Das Datumsfeld zur Deaktivierung wird bearbeitbar)
- Eingabe des Datums der Deaktivierung der Policy
- Klicken des „Speichern"-Buttons

Ergebniszustand:

Die Policy wird als deaktiviert gespeichert zu dem angegebenen Datum. Die Eigenschaften der Policy werden geschlossen und die Anzeige aller Policys aktualisiert. Die soeben deaktivierte Policy wird als deaktiviert gekennzeichnet.

Komponente: PDF-Export

Beschreibung:

Erstellen einer PDF-Datei mit einem Policy-Bericht über ein bestimmtes Datum.

Vorheriger Zustand:

Das Feld der Berichtvorschau ist mit dem Bericht über das momentane Datum gefüllt.

Arbeitsablauf:

- Auswählen des Datums, über das der Bericht erstellt werden soll
- (Das Feld der Berichtvorschau wird automatisch aktualisiert und zeigt nun eine Vorschau des Berichtes zu dem ausgewählten Datum)
- Klicken des „Exportieren als PDF"-Buttons
- Auswählen des Speicherortes sowie des Dateinamens für den Bericht
- Klicken auf „Speichern"

Besonderheiten:

Wenn ein Datum ungünstig gewählt wird, kann es passieren, dass die PDF-Datei keine Policys enthält.

Ergebniszustand:

Die Datei wurde in dem angegebenen Verzeichnis gespeichert und kann von einem geeigneten PDF-Leseprogramm angesehen werden.

Weitere Use-Cases, deren Ausführung den Rahmen dieser Arbeit sprengen würde, sind hier stichwortartig aufgeführt.

- **Komponente: Administration**
  - Ändern beliebiger Daten in der Datenbankdatei anstatt im Programm

- **Komponente: Benutzer-Verwaltung**
  - Erstellen eines neuen Benutzers mit beliebigen Rechten
  - Löschen eines Benutzers
  - Ändern der Daten eines Benutzers

- **Komponente: Policy-Verwaltung**
  - Eintragen einer neuen Policy
  - Ändern der Daten einer bestehenden Policy
  - Aktivieren einer Policy
  - Suchen nach einer bestimmten Policy
  - Suchen von Policys nach bestimmten Kriterien
  - Erstellen von Gruppen
  - Erstellen von IPs

4 Programmierung

Für die Programmierung wurde C# auf .NET-3.5-Basis verwendet, um die Einbindung in die Windowsumgebung sowie das Verwenden einer Microsoft-Access-Datenbank zu ermöglichen.

Zur Entwicklung wurde Visual Studio 2008 verwendet, mit dem sich Debugprozesse einfach verwalten lassen und perfekte Integration der gewählten Programmiersprache bietet. Dadurch konnte die Effizienz gesteigert werden, mit der die Komponenten programmtechnisch realisiert wurden.

Das Programm wird als eigenständige Windows-Applikation entwickelt, mit einer grafischen Benutzeroberfläche, die auf dem Zielserver lauffähig ist.

Da das Programm für Endnutzer gedacht ist, wird eine umfangreiche Ausnahmenregelung eingebaut, die dem Nutzer leicht verständliche Fehlermeldungen anzeigt und keine kryptischen Stacktraces und Exceptions.

Die Umsetzung der Anforderungen wurde ständig überwacht und ist zu 100% erfüllt worden. Jede Anforderung ist durch eine Systemanforderung abgedeckt (s. 3.2), die ihrerseits in Komponenten zusammengefasst sind (s. 3.1).

Die Funktionen der Komponenten wurden durch Unit-Tests verifiziert (s. 3.3) und erfüllen demnach genau ihren Zweck, der rückwirkend gesehen durch die Anforderungen (s. 2.2) definiert wurden.

Während der Umsetzung der Systemanforderungen wurden die Anforderung stetig durch den Arbeitsablauf des Versionierungssystems auf dem aktuellen Stand gehalten. Sobald eine Systemanforderung durch eine fertig programmierte Komponente erfüllt wurde, konnten die Unit-Tests für diese Komponente durchgeführt werden.

Sollten sich bei den Unit-Tests Verhaltensweisen zeigen, die nicht dem erwarteten Verhalten entsprechen, wird die Komponente erneut in den Entwicklungsstand gesetzt und muss überarbeitet werden, um den Unit-Test zu erfüllen.

Dieser Prozess ermöglicht die ständige Anpassung und Weiterentwicklung des Programms und der Komponenten, bis sie den Anforderungen exakt gerecht werden.

Sobald die Unit-Tests erfolgreich abgeschlossen waren, konnten die Use-Cases für die getestete Komponente erstellt werden, die eine Art Dokumentation für Benutzer oder Administratoren darstellt.

Der hier beschriebene Ablauf ist in dem folgenden Diagramm schematisch dargestellt.

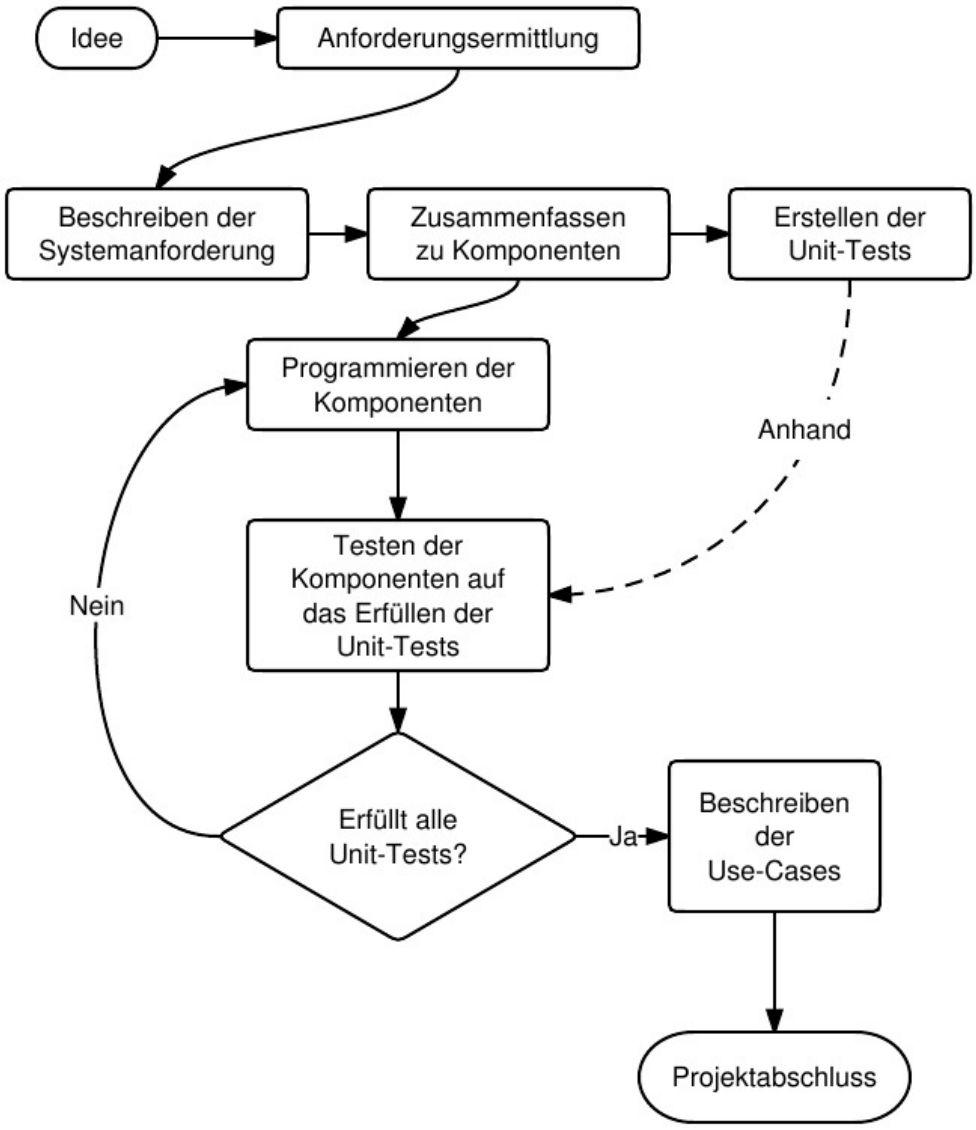

Abb. 2: Flussdiagramm der Projektentwicklung

Die Entscheidung zur Programmiersprache erwies sich als gut, da sie allen Anforderungen standhielt und die Umsetzung der Systemanforderungen nicht behinderte. Besonders durch die Datenbank im Access-Format war die ebenfalls von Microsoft stammende Programmiersprache erste Wahl für dieses Projekt.

Für spätere Änderungen am Programm wurden die Quellcode-Dateien in das Versionierungssystem eingepflegt. So kann die weitere Entwicklung am Programm überwacht und, wenn nötig, können alte Dateien wiederhergestellt werden.

Um das Verständnis des System für nicht eingearbeitete Programmierer zu erleichtern, wurden für die wichtigsten Abläufe Flussdiagramme erstellt, die den internen Programmablauf schematisch darstellen.

Das Öffnen des Programms ist aufgrund des Login-Systems komplex genug, um in einem Diagramm beschrieben zu werden.

Daraus werden die Abläufe ersichtlich, die sich im Inneren des Programms abspielen, wenn der Nutzer das Programm öffnet.

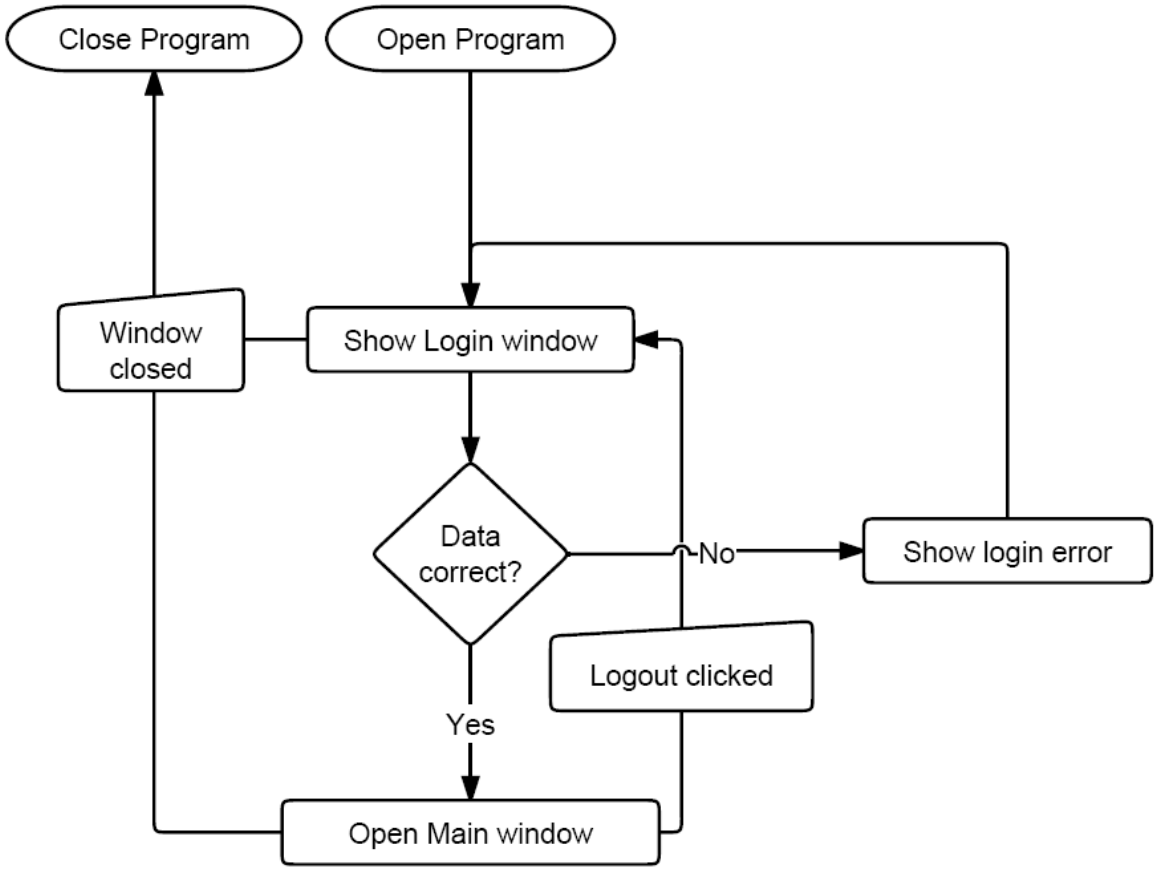

Abb. 3: Flussdiagramm des Öffnungsprozesses des Programms

Das Diagramm ist der Dokumentation für Programmierer angehängt, um ihnen die Weiterentwicklung des Programms zu erleichtern.

Der Ablauf des Öffnens einer Policy ist ebenfalls dokumentiert worden – bei diesem Diagramm geht es vor allem darum, die Logik des Aufrufs darzustellen, da diese vergleichsweise komplex ist.

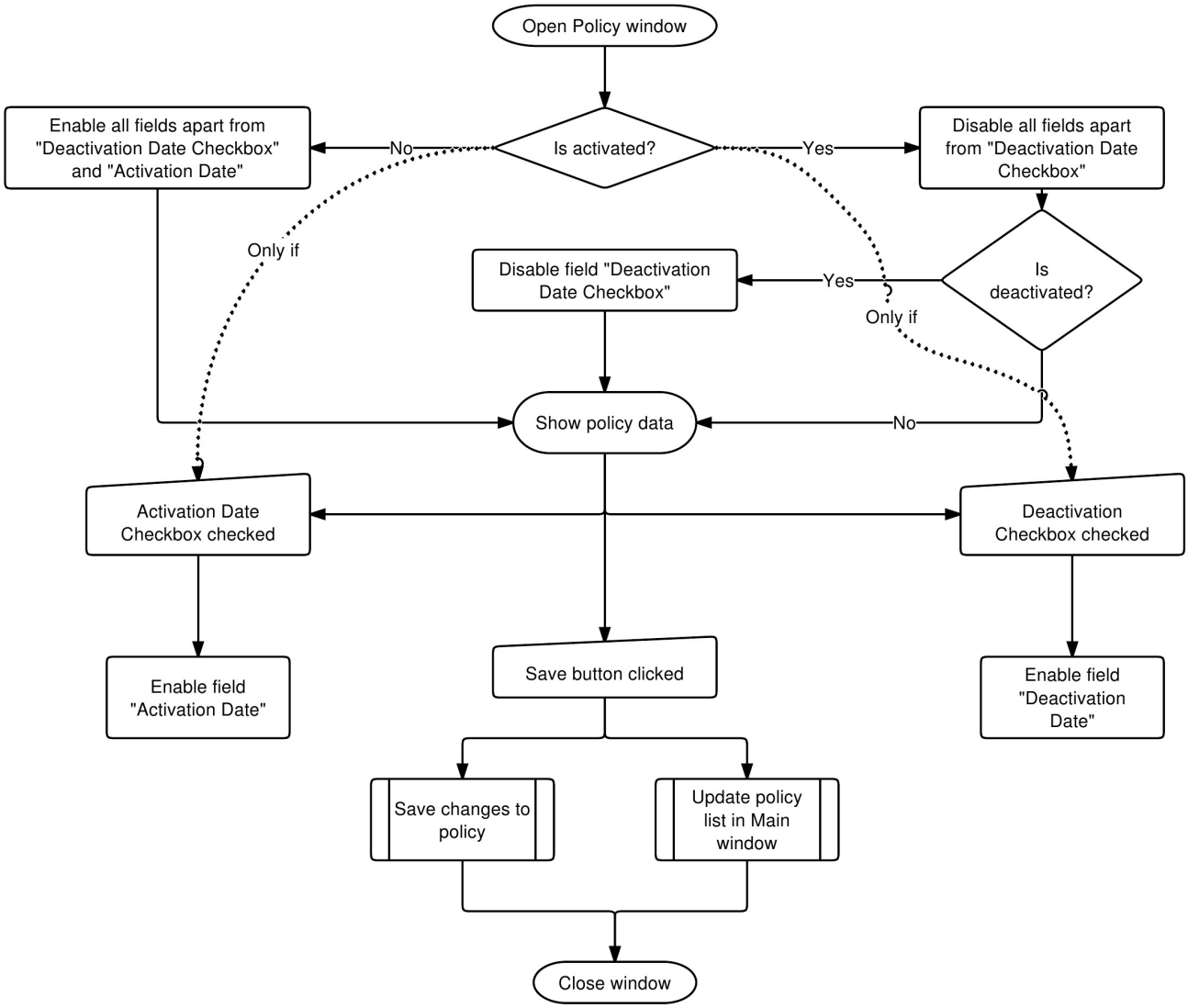

Abb. 4: Flussdiagramm des Öffnungsprozesses einer bereits bestehenden Policy

Die gepunktet gezeichneten Linien erzeugen eine Abhängigkeit der Komponenten, d.h. die Aktion, auf welche der Pfeil zeigt, kann nur ausgeführt werden, wenn die Bedingung, bei der er anfängt, in ihrer gezeigten Form erfüllt ist.

Die Diagramme wurden nach der Entwicklung fertig gestellt und sind daher versionsabhängig und könnten sich je nach Neuentwicklung ändern.

Um die Erweiterbarkeit des Programms zu gewährleisten, wurde auf eine komplett objektorientierte Programmierung gesetzt. Dies ermöglicht es, anderen Programmierern die Komponenten des Programms leicht zu erweitern oder zu verbessern.

Für die Objektorientierung wurden all jene Programmteile zu Klassen zusammengefasst, die nach dem Verständnis der Objektorientierung instanziiert werden können; Policys, Gruppen, IPs und Benutzer.

Hierbei musste jedoch mit einem Gesetz der Objektorientierung gebrochen werden, um die Integration in das Programm zu ermöglichen.

Jede der oben genannten Klassen definiert eine Eigenschaft „Id" für ihre Objekte. Dies entspricht der Id, welche dieses Objekt in der Datenbankebene hat, um die eindeutige Bindung an die Datenbank zu ermöglichen.

Da dies einen Bruch mit den Prinzipien der Objektorientierung darstellt, wird sie explizit dokumentiert, um weitere Entwickler am Programm nicht in die Irre zu führen.

Die folgenden Diagramme zeigen die beschriebenen Klassen. Hierbei gilt folgende Legende:

Symbol	**Bedeutung**
+	Öffentlich, für jeden innerhalb des Programms sichtbar
-	Privat, nur für das Objekt sichtbar
*	Statisch, für alle Objekte gleich

Von der Beschreibung der Funktionsweise der Eigenschaften und Methoden wurde abgesehen, da sie meistens sprechend sind und ihre Funktion sich daher in ihrem Namen widerspiegelt.

Die Übersicht dient vor allem der technischen Dokumentation des Programmes, um es Programmierern zu erleichtern, die Struktur der Klassen und Objekte nachvollziehen zu können.

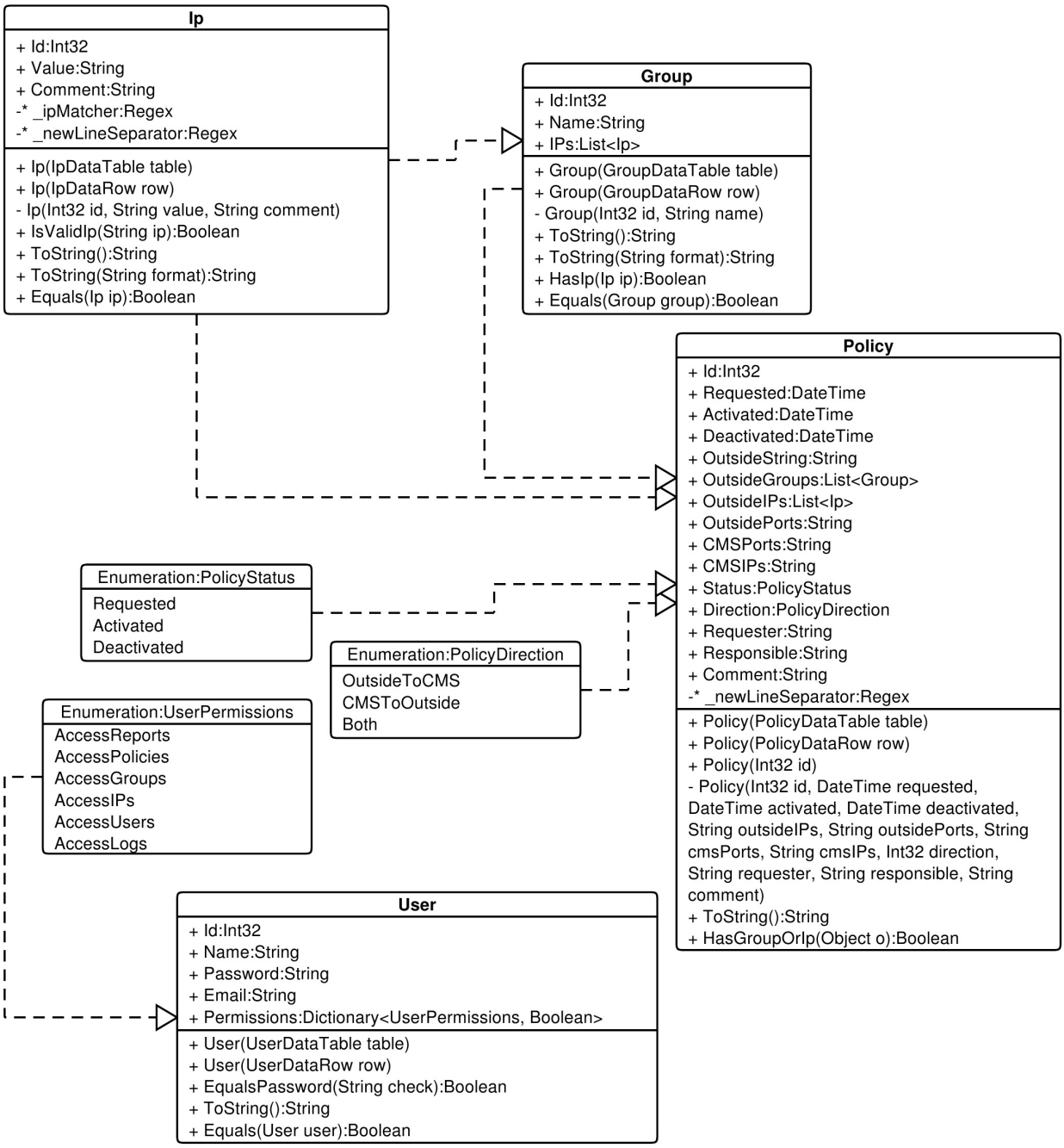

Abb. 5: Klassendiagramm der selbst erstellten Klassen

Die Linien kennzeichnen eine Abhängigkeit zwischen den Klassen. Wenn z.B. die Klasse *User* die Eigenschaft *Permissions* aufweist, muss die Enumeration *UserPermissions* vorhanden sein, damit die Eigenschaft korrekt dargestellt werden kann.

5 Projektabschluss

Nach dem technischen Abschluss des Projektes wird das Programm in den nächsten Schritten auf dem Produktivserver eingespielt und dort nochmals auf vollständige Funktionsfähigkeit hin getestet.

Es wurde eine benutzerorientierte Dokumentation erstellt, die grob die Funktionen des Programms umreißt und die Handhabung der Policys erläutert. (s. Anhang)

Da die Verwaltung der Policys bereits einige Monate vor diesem Projekt begonnen hat (damals noch mit einer Excel-Tabelle), müssen die bereits dokumentierten Policys nachgetragen werden, um eine lückenlose Dokumentation sicherzustellen.

Nachdem der Einrichtungsprozess abgeschlossen ist, muss eine kurze Schulung für alle Verantwortlichen erfolgen, um die korrekte Nutzung des Programms zu gewährleisten. Diese Schulung wird sowohl den Kunden als auch das TANNER-Team betreffen.

Nach dieser Schulung kann das Projekt im Sinne des Projektmanagements als abgeschlossen betrachtet werden, lediglich der laufende Support für Fragen oder Probleme ist und bleibt ein offener Posten, der vom leitenden Programmierer überwacht wird.

Sollten vom Kunden oder dem TANNER-Team Fragen zur Verwendung auftreten, werden diese durch den leitenden Entwickler beantwortet und es wird gegebenenfalls eine Nachschulung angesetzt.

6 Verzeichnisse

6.1 Literaturverzeichnis

[2-8] U.S. Department of Health & Human Services. Code of Federal Regulations. Revision of 2009/4/1. Title 21 Chapter I Part 11
(als Abschrift von der offiziellen Homepage des amerikanischen Gesundheitsministeriums, Zugriffsdatum: 26.09.2010 – Kopie s. Anhang)
http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?CFRPart=11&showFR=1

6.2 Abbildungsverzeichnis

7 Anhang

7.1 FirewallDoc User Manual

7.2 Code of Federal Regulations

Die hier folgende Kopie des „21CFR11" wurde der offiziellen Seite der „Food and Drug Administration" am 26.08.2010 entnommen.

Dies wurde nicht als Internet-Zitat definiert, da die Herkunft des Dokuments einwandfrei festgestellt werden kann und daher keine Gefahr der Veränderung oder Verfälschung des Inhalts zu erwarten ist.

http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?CFRPart=11&showFR=1
Zugriffsdatum: 26.08.2010 15:27