INSTITUT FÜR INFORMATIK
DER LUDWIG MAXIMILIAN UNIVERSITÄT MÜNCHEN

Diplomarbeit

Integration eines webbasierten Informations-Systems mit einem Hotline-Tool und einer Installationsdatenbank

Bearbeiter: Oliver Maul

Aufgabensteller: Prof. Dr. Johann Schlichter

Betreuer: Dipl.-Inform. Frank Schütz
Dipl.-Inform. Karl Ewald

INSTITUT FÜR INFORMATIK
DER LUDWIG MAXIMILIAN UNIVERSITÄT MÜNCHEN

Diplomarbeit

Integration eines webbasierten Informations-Systems mit einem Hotline-Tool und einer Installationsdatenbank

Bearbeiter: Oliver Maul

Aufgabensteller: Prof. Dr. Johann Schlichter

Betreuer: Dipl.-Inform. Frank Schütz
Dipl.-Inform. Karl Ewald

Abgabetermin: 14.09.2000

Hiermit versichere ich, daß ich die vorliegende Diplomarbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

....
Ort, Datum Unterschrift

Zusammenfassung

Der unkomplizierte Zugriff auf Informationen und die Transparenz der vorhandenen Dienstleistungen wird immer wichtiger als Faktor für Serviceorientierung und Kundenbindung. Bei der Firma IXOS Software AG besteht bereits ein webbasiertes Informationssystem, das IXOS Expert Service Center (ESC), über das Kunden auf produktspezifische Informationen aller Art zugreifen können.

Daneben arbeiten die IXOS-Angestellten in der Supportabteilung mit einem webbasierten „Problem Tracking System“ (PTS), in dem jeder Supportfall gepflegt und verwaltet wird. Die Daten über die Installationsdetails jeder Firma, z.B. eingesetzte Komponenten und Releases, werden in der „Customer Installation Database“ (CID) gespeichert und können ebenfalls über ein Web-Interface gepflegt werden.

Ziel ist nun, den Kunden einen direkten Zugriff auf ihre Supportfälle sowie auf ihre Daten in der Installationsdatenbank zu ermöglichen. Dieser Zugriff soll über das schon für Kunden geöffnete ESC – also über ein Web-Interface – erfolgen.

Aufgabe der Diplomarbeit ist es zunächst, in Zusammenarbeit mit der Supportabteilung der Firma IXOS eine Spezifikation für die gewünschte Integration der Systeme zu erstellen. Anhand dieser Spezifikation soll dann ein weitestgehend flexibles Berechtigungskonzept entwickelt werden, das in die Schnittstellen zu PTS und CID integriert wird.

Im Anschluß an die Definition der Schnittstellen sollen die Webseiten und Skripten für Kunden im ESC erstellt werden, über die Informationen zu Supportfällen modifiziert und Informationen zu eigenen Installationen abgerufen werden können. Bei der Modifikation von Daten durch den Kunden wird das PTS den Schwerpunkt bilden, weil hier der Informationsaustausch im Vordergrund steht.

Der Zugriff auf die verschiedenen Systeme soll von den Kunden als einheitliche Oberfläche wahrgenommen werden. Außerdem soll das neue System personalisiert sein und den Kunden über für seine Installation relevante Neuigkeiten informieren.

Inhaltsverzeichnis

Abbildungsverzeichnis

Glossar

CGI Common Gateway Interface. Eine Schnittstelle zum Datenaustausch zwischen Webserver und anderen Anwendungen.

CID Customer Installation Database. Die Installationsdatenbank der IXOS.

ESC Expert Service Center. Das Informationsportal für Kunden der IXOS.

HTML Hyper Text Markup Language. Beschreibungssprache für Dokumente.

HTTP Hyper Text Transfer Protocol

LDAP Lightweihgt Directory Access Protocol. Ein kompaktes Protokoll für den Zugriff auf Verzeichnisdienste.

KNW Know How Document Format. Ein Dokumentenformat, das im ESC vorkommt.

MKIS Marketing Information System. Eine Vertriebsdatenbank im SAP R/3.

NIS Network Information System

NFS Network File System

PDF Portable Document Format

Pool Gruppe von Support-Mitarbeitern, die eine organisatorische Einheit bilden.

PTS Problem Tracking System. Ein System zur Verwaltung der Supportfälle der IXOS.

RFC Remote Function Call. Es wird zur Kommunikation zwischen den Applikationsservern des SAP R/3 und anderen Kommunikationsservern eingesetzt.

SSL Secure Socket Layer. Ein Protokoll zur verschlüsselten Datenübertragung.

SQL Standard Query Language. Datendefinitions- und abfragesprache für relationale Datenbanken.

X.500 ITU-Norm eines verteilten Verzeichnisdienstes.

Kapitel 1

Einleitung

Unternehmen, deren Hauptaufgabe in der Erstellung von Software liegt, kommen heutzutage nicht mehr ohne einem leistungsstarken Kundenservice aus. Der Wettbewerb in der IT-Branche verlangt nach einem qualitativ hochwertigen Kundenservice. Kunden erwarten, daß sie nicht nur bis zum Kauf eines Produkts betreut werden sondern auch und besonders danach. Dies ist bei den immer komplexer werdenden Produkten auch unerläßlich. Allerdings darf der Aufwand für den Kundenservice nicht unterschätzt werden. Bei der IXOS Software AG sind rund 80 Mitarbeiter für den Support zuständig.

Die Effizienz des Kundensupports kann durch eine bessere Kommunikation zwischen den Kunden und den Supportmitarbeitern gesteigert werden, wodurch auch die Kundenzufriedenheit positiv beeinflußt wird. Die Effizienz nimmt zu, weil durch die bessere Kommunikation gezielter an Informationen gelangt werden kann. Durch die gesteigerte Effizienz können mehr Supportfälle in kürzere Zeit abgewickelt werden, was zu einer höheren Kundenzufriedenheit führt. Ziel dieser Arbeit ist es, die Kommunikationsmöglichkeiten zwischen Kunden und den Supportmitarbeitern zu verbessern.

Bei der IXOS geht es im Support um die Unterstützung ihrer Kunden beim Einsatz der von IXOS hergestellten und direkt vertriebenen Dokumentenarchivierungssoftware IXOS-ARCHIVE. Zur Verwaltung der Informationen zu einer Installation wird die „Customer Installation Database“ (CID) verwendet. Im CID finden sich unter anderem die Daten der eingesetzten Hardware, die Softwareversionen der einzelnen Komponenten sowie die technischen Ansprechpartner beim Kunden. Als Trouble Ticket System wurde das „Problem Tracking System“ (PTS) entwickelt. Es dient der Abwicklung von Supportfällen und enthält zu jedem Fall unter anderem eine Problembeschreibung, die Priorität, die bisher aufgewendete Zeit und das Protokoll der durchgeführten Aktivitäten zur Behebung des Problems.

Zu jeder Installation gibt es beim Kunden in der Regel mehrere technische Ansprechpartner, die sich bei auftretenden Problemen an den Support wenden können. Derzeit wenden sich die Ansprechpartner des Kunden mit einem Problem per Telefon oder Email an die

Hotline. Die Support-Mitarbeiter sind dadurch mit Routinetätigkeiten wie dem Erfassen von Daten belastet. Bei einem eingehenden Anruf an der Hotline muß festgestellt werden, welche Installation betroffen ist, wer der Ansprechpartner ist und ob ein existierender Supportfall betroffen ist. Ist ein existierender Fall betroffen, wird das Gespräch an den zuständigen Hotliner weitergegeben, falls er verfügbar ist. Bei einem neuen Fall, muß zuerst festgestellt werden, ob der Anrufer berechtigt ist, den Support in Anspruch zu nehmen. Danach wird ein neuer Supportfall angelegt und mit den Informationen (siehe auch [Bra99]) zum Problem gefüllt. Auch wenn ein Problem per Email an den Support gemeldet wird, erfolgt die Übernahme der enthaltenen Informationen teils manuell. Außerdem können bei schwierigen Supportfällen unter Umständen Bearbeitungszeiten von mehreren Wochen vorkommen. Dadurch verlieren Kunden leicht den Überblick über offene Fälle und deren Stand. Es kommt vor, daß Kunden Ihre Fallnummern nicht mehr wissen und dadurch zu einem Problem mehrere Fälle angelegt werden.

Bisher haben Kunden Zugriff auf das „Expert Service Center“ (ESC) mit Produktinformationen, Patches, technischen Hintergrundinformationen und aufbereiteten Problemlösungen. Diese Datenbank wird von den Support-Mitarbeitern gepflegt. Allerdings bleibt durch den hohen Aufwand an der Hotline wenig Zeit zur Pflege dieser Wissensdatenbank. Das und die fehlende Zeit zur Pflege der Wissensdatenbank sorgt für Unzufriedenheit bei Kunden und Mitarbeitern.

Durch die Öffnung der Support-Systeme für Kunden wäre zugleich eine Entlastung der Mitarbeiter und eine Erhöhung der Transparenz zu erreichen. Die Mitarbeiter werden nur mehr in geringem Maße mit Routinetätigkeiten aufgehalten und können sich dadurch mehr der inhaltlichen Lösung der Fälle und der Pflege der Wissensdatenbank zuwenden. Die Erfassung neuer und Mitwirkung zur Lösung bestehender Supportfälle durch den Kunden bezieht ihn stärker als bisher in den Supportprozeß ein. Dadurch wird es in einigen Fällen zügiger zu einer Lösung kommen. Zudem erhält der Kunde einen besseren Einblick in die Abwicklung seiner Fälle.

Außer Kunden sollen auch Partner der IXOS online Zugriff auf das PTS erhalten. Werden zukünftig die Vertriebswege auch auf indirekten Vertrieb durch Partner oder Reseller ausgeweitet, soll die Möglichkeit bestehen, daß diese und deren Kunden den Support von IXOS nutzen können. Das trägt einer geregelten und effizienten Übergabe von Supportfällen an die IXOS als Second Level Support bei und erlaubt einen Einblick in die Support-Abwicklung der Partner.

Da die Support-Systeme teilweise vertrauliche Informationen enthalten, ist es nicht möglich, den Zugriff für Kunden ohne Einschränkungen zu erlauben.

Es ergeben sich folgende Anforderungen für die Freigabe der Support-Systeme:

- Kunden dürfen nur Supportfälle anlegen, ändern und schließen, die eine ihrer Installationen betreffen.
- Kunden dürfen nur bestimmte Informationen in ihren Supportfällen sehen und

ändern.

- Kunden dürfen nur bestimmte Informationen ihrer Installationsdaten ändern.
- Nur bestimmte Benutzer dürfen Benutzerkonten administrieren.

Ziel der Diplomarbeit ist es, ein Konzept zu entwickeln, das die beiden bisher nur intern verfügbaren Support-Systeme PTS und CID den Kunden mit den genannten Einschränkungen bereitstellt. Im Anschluß soll auch damit begonnen werden, exemplarisch einen Teil des Konzepts umzusetzen.

Im ersten Schritt werden die bisher eingesetzten Systeme vorgestellt und auf ihre technische Realisierung eingegangen. Besonders die Verwendung der Systeme durch Mitarbeiter und Kunden, soweit bereits gegeben, wird analysiert.

Kapitel 2

Bestehende Systeme bei der IXOS

In diesem Kapitel sollen zuerst die im Support bestehenden Systeme, die für diese Diplomarbeit relevant sind, betrachtet werden. Dazu zählen neben dem Expert Service Center (ESC) das Problem Tracking System (PTS) und die Customer Installation Database (CID). Auch das Marketing Information System (MKIS), auf das aus den anderen Systemen verwiesen wird, wird kurz erläutert. Neben der Funktionalität werden die technischen Aspekte wie Datenmodell und -speicherung und Oberfläche der Systeme vorgestellt. Die Reihenfolge, in der die einzelnen Systeme hier behandelt werden, ist so gewählt, daß keine Verweise auf nachfolgende Kapitel vorkommen.

2.1 Marketing Information System (MKIS)

MKIS ist eine SAP R/3–Anwendung, die Marketing und Vertrieb unterstützt. Für die im folgenden vorgestellten Systeme ist die Verwaltung der Kundenstammdaten, die in jedem R/3 vorhanden ist, relevant.

Oberfläche

Das MKIS wird ausschließlich über das SAPGUI bedient. Das SAPGUI ist ein universeller Client für das SAP R/3. Universell bedeutet hier, daß der Client für alle gängigen Betriebssysteme verfügbar ist, für jedes Modul des R/3 verwendet werden kann. Er ist für die Kommunikation mit dem Dialogserver des R/3, die Darstellung der Dialoge und die Eingaben des Benutzers zuständig.

Authentifizierung und Autorisierung

Die Authentifizierung erfolgt über das SAPGUI mit einem Benutzernamen und einem Paßwort. Verifiziert wird der Login vom Application-Server. Die Autorisierung wird durch die R/3 Administratoren vorgenommen und durch den Application-Server gewährleistet. Der R/3 Administrator ordnet beim Erzeugen eines neuen Accounts Berechtigungsprofile zu. Berechtigungsprofile kommen auch im SapNet (siehe Kap. 3.1) vor. Sie enthalten Berechtigungen, welche bestimmen, was unter welchen Umständen erlaubt ist.

Zugriffsmöglichkeiten

Auf die Daten im MKIS kann über das RFC Protokoll (Remote Function Call) zugegriffen werden. Dazu werden im R/3 mit ABAP Funktionsbausteine programmiert, die für den Zugriff per RFC freigegeben sein müssen. Ein mit dem R/3 vorinstallierter Funktionsbaustein ist der `RFC_READ_TABLE`, der den Zugriff mit SQL auf alle freigegebenen Tabellen in der R/3 Datenbank zuläßt. Dieser Funktionsbaustein wird von CID, PTS und ESC verwendet, um auf Kundendaten und deren Ansprechpartner zugreifen zu können. Clients können mittels der RFC Bibliothek, die für jede gängige Plattform erhältlich ist, auf die Funktionsbausteine zugreifen.

Eine RFC-Verbindung basiert auf einer Sitzung und läuft folgendermaßen ab. Der Client muß sich zu Beginn einer Sitzung mit einem Account, dem durch ein Berechtigungsprofil eine Berechtigung für die Anmeldung über RFC zugewiesen ist, per RFC authentifizieren. Nach erfolgreicher Anmeldung ist die Sitzung aufgebaut und es können beliebig viele Funktionsaufrufe durchgeführt werden. Die Sitzung wird nach den Funktionsaufrufen explizit beendet.

Datenmodell

Das Datenmodell des MKIS besteht aus Tabellen im R/3 Data Dictionary. Die für das CID relevanten Tabellen sind `kna1`, `knvk` und `knvp` (siehe Abb. 2.1). Die Tabelle `kna1` enthält unter anderem die Kundennummer, die Anschrift sowie die Telefon– und Faxnummer der Firma. In der Tabelle `knvk` sind der Name, die Telefon– und Faxnummer sowie die Sprache der Ansprechpartner der Firmen gespeichert. Jeder Ansprechpartner hat eine Partnernummer und ist über die Kundennummer mit der Firma assoziiert. Die Tabelle `knvp` dient dazu Beziehungen zwischen einer Firma und einem anderen Objekt wie z.B. anderen Firmen oder fremden Ansprechpartnern abbilden zu können. `knvp` faßt aus technischen Gründen mehrere Join-Tabellen zusammen, die nach dem logischen Datenmodell einzelne Tabellen entsprechen würden. Sie sind in der Abbildung getrennt dargestellt. Das Feld Parnerfunktion bestimmt eindeutig, welche Beziehung dargestellt wird. Dementsprechend ist in jedem Datensatz nur ein Fremdschlüssel belegt.

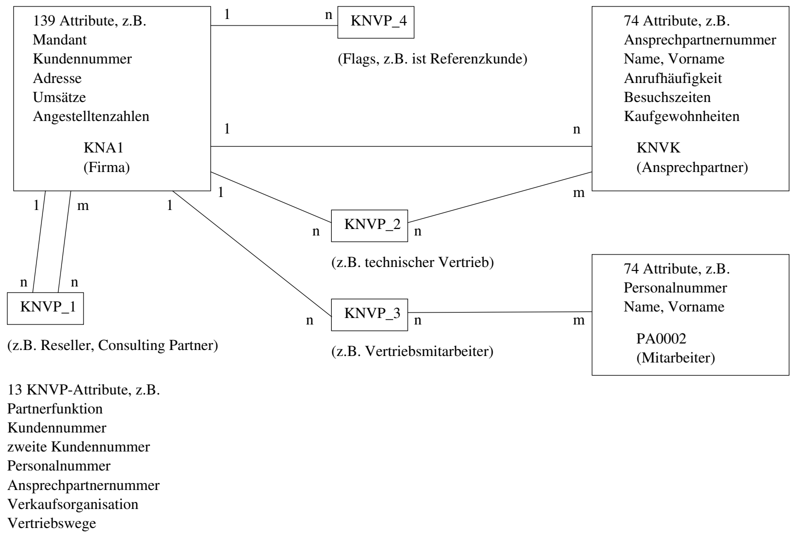

Abbildung 2.1: vereinfachtes MKIS-Datenmodell

2.2 Customer Installation Database (CID)

Die CID ist ein System mit Informationen zu den Installationen von IXOS-Produkten. Nach der Installation durch einen Support-Mitarbeiter werden hier die Informationen, die die Installation betreffen, gespeichert. Dazu gehört, ob es sich um eine Demoinstallation oder eine lizenzierte Installation handelt, das zugewiesene Supportzentrum, Informationen zu Lizenzen, Hardware– und Softwarekomponenten. Um redundante Speicherung von Daten zu vermeiden, wird auf die Stammdaten von Kunden und deren Ansprechpartnern ins MKIS verwiesen.

Das CID ist ein webbasiertes Informationssystem, das aus Perl–basierten CGI–Skripten und für den Zugriff auf das MKIS aus einem RFC-Client besteht. Die Daten werden in flachen Dateien und in einer separaten relationalen Datenbank bzw. im MKIS in einer R/3 Datenbank gehalten und sind zum überwiegenden Teil langlebige Daten (Stammdaten des Kunden).

Es ist mittelfristig geplant, die CID Daten als kundenspezifische Tabellen in das R/3 zu migrieren. Dadurch wäre der Zugriff auf die Daten einheitlich über die RFC Schnittstelle realisierbar. Aufwendige Joins zwischen Daten im MKIS und Daten im CID könnten dann im R/3 stattfinden.

Oberfläche

Die Schnittstelle zum Benutzer ist durch Webseiten realisiert. Nach der Authentifizierung mit einem Account (siehe nächsten Absatz) besteht auf der Einstiegsseite die Möglichkeit zur Suche nach einer Installation über die Installationsnummer oder eine Substringsuche im Ortsnamen und im internen Kurznamen der Firma. Zudem können neue Installationen erfaßt werden oder automatisch erzeugte Excel-Tabellen mit den CID-Daten heruntergeladen werden. Sowohl in der Weboberfläche als auch in den Excel-Tabellen werden die zugeordneten MKIS-Daten mit angezeigt.

Authentifizierung und Autorisierung

Die Anmeldung erfolgt über das HTTP-Protokoll des Apache-Webserver[1]. Die Skripten können den Benutzer einer Sitzung über die Umgebungsvariable REMOTE_USER ermitteln. Die Logins entsprechen den Unix-Logins und werden aus der NIS–passwd–Datenbank gewonnen. Da das CID derzeit nur im Intranet verfügbar ist und daher nicht mit Böswilligkeiten gerechnet werden muß, erhält jeder Benutzer die gleichen Rechte und darf Informationen zu Installationen ändern, neue Installationen eintragen und bestehende ansehen. Änderungen an den Daten werden aber im jedem Fall protokolliert. Die CID-Skripten selbst greifen über einen speziellen Account auf das MKIS zu.

Implementierung

Die Funktionalität ist hauptsächlich in dem Perl-Skript `cid.cgi` zusammengefaßt, das über Common Gateway Interface (CGI) mit dem Webserver kommuniziert. CGI definiert die Schnittstelle zwischen Webserver und externen Programmen. Die Schnittstelle besteht aus dem Environment und Standardeingabe und –ausgabe. Je nach Aufruf wird die Einstiegsseite aufgebaut, je nach Benutzergruppe verschiedene Editiersichten, eine Suche initiiert oder das Suchergebnis angezeigt.

Für die Informationen über den Kunden einer Installation ist ein Zugriff auf das MKIS notwendig. Dazu existiert das C-Programm `getmkis`, das mit `cid.cgi` über die Kommandozeilenparameter und die Standardausgabe kommuniziert. Zum MKIS wird über RFC auf den ABAP Funktionsbaustein RFC_READ_TABLE zugegriffen.

[1] http://www.apache.org/httpd.html

Datenmodell

Das Datenmodell (siehe Abb. 2.2) des CID teilt sich auf das MKIS und das CID auf. Die Daten zum Kunden, wie Anschrift, Telefon und Ansprechpartner sind im MKIS gespeichert. Im CID wird zu jeder Installation neben der MKIS-Kundennummer als Fremdschlüssel Daten zur Installation gespeichert. Dazu zählt das Produkt, der Lizenztyp, die Zeitzone und das zugewiesene Supportzentrum. Eine Installation entspricht einem Verzeichnis mit der Installationsnummer als Namen unter `/ixos/tssys/cid/cust/`. In einem solchen Verzeichnis befindet sich in der Datei `.status` der Datensatz zur Installation. Korrespondenz in Form von Email wird ebenfalls in diesem Verzeichnis gespeichert. Die meisten Felder des Datensatzes werden parallel in einer relationalen Datenbank gehalten.

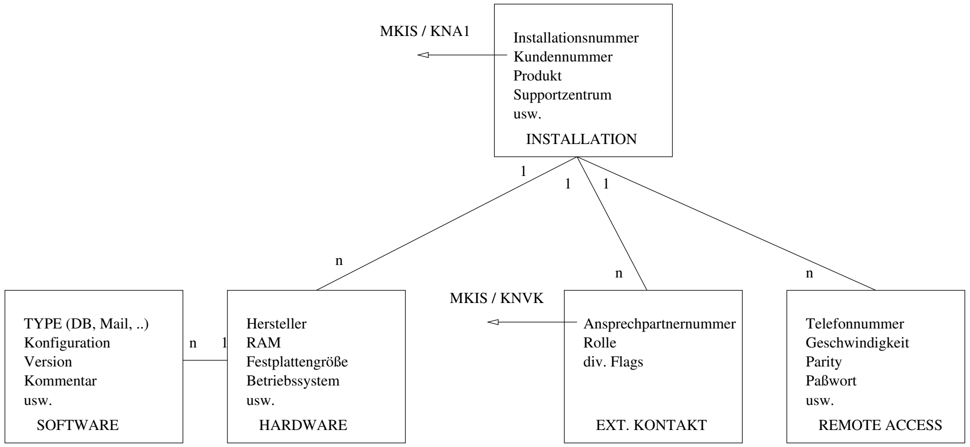

Abbildung 2.2: CID-Datenmodell (Auswahl)

2.3 Problem Tracking System (PTS)

Der hauptsächliche Zweck des PTS ist es, die Bearbeitung von Supportfällen zu unterstützen. Jeder Fall, die damit verbundenen Emails und Dateien, die Historie, der Status und der zugewiesene Verantwortliche wird darin verwaltet. Die Supportmitarbeiter sind in Pools eingeteilt. Aus jedem Pool gibt es einen Hotliner-on-Duty, der auch die Fälle von abwesenden Supportern angezeigt bekommt, falls diese eine Aktion seitens der IXOS erfordern. Jeder Pool kann dazu einen Kalender verwalten, wann welche Supporter als Hotliner-on-Duty eingeteilt sind.

Die Startseite des PTS zeigt die momentan anwesenden Hotliner, den Hotliner-on-Duty und vorallem eine Liste eingegangener Emails, die noch zu bearbeiten sind. Das PTS ist zwar jedem IXOS-Mitarbeiter über das Intranet zugänglich, wird aber hauptsächlich von Supportmitarbeitern genutzt. Weil der PTS–Webserver im Intranet liegt, haben externe Personen, also auch Kunden und Partner, keinen Zugriff darauf.

Oberfläche

Der Zugriff auf das PTS erfolgt wie beim CID ausschließlich über einen Web-Browser. Auf der Startseite erhält ein Supportmitarbeiter eine Übersicht seiner offenen Fälle in der Reihenfolge der Dringlichkeit.

Zudem besteht die Möglichkeit, nach Fällen zu suchen. Dabei kann die Suche nach vielen Kriterien (Kundenname, Produkt, Supportzentrum, Falltitel, Priorität, Status und Verantwortlicher) eingeschränkt werden. Natürlich ist auch der Sprung zu einem bestimmten Fall über die Fallnummer möglich.

Ein Fall besteht aus einer kurzen Beschreibung, dem Subject, sowie den Kundendaten und einer detaillierten Beschreibung des Falls. Die Entwicklung eines Falls wird in der Historie festgehalten. Jeder Lösungsvorschlag und jede Antwort des Kunden wird an die Historie angehängt und deren Bearbeitungsdauer erfaßt. Der Status eines Falls ändert sich von open-new bis closed-done mehrmals.

Mögliche Zustände eines PTS Supportfalls:

- open-new (Fall aufgenommen)
- open-todo (IXOS muß etwas tun)
- open-feedback (Kunde muß etwas testen oder weitere Informationen liefern)
- open-waitvers (Lösung durch in Entwicklung befindliche Produktversion)
- open-closewait (meldet sich Kunde nicht mehr, wird der Fall geschlossen)
- closed-postproc (Fall muß noch intern nachbearbeitet werden)
- closed-done (Fall ist geschlossen)

Die Priorität eines Falls läßt sich ebenfalls ändern und beeinflußt die Reihenfolge der vorgelegten Fälle.

Authentifizierung und Autorisierung

Zunächst erfordert der Zugriff auf die PTS-Funktionen einen Login wie bei der CID. Ist ein Benutzer nicht einem Pool von Support-Mitarbeitern zugeordnet, erhält er einen Gast-

Status, womit es nicht möglich ist, bestehende Supportfälle zu ändern oder neue Supportfälle anzulegen. Somit ist es allen IXOS-Mitarbeitern, erlaubt bestehende Supportfälle anzusehen. Eine weitergehende Sicherung ist für das PTS nicht notwendig, weil sich Server und Client im Intranet befinden und davon ausgegangen wird, daß von dort keine Bedrohung ausgehen wird.

Implementierung

Das PTS besteht aus einem Web-Server, Perl-Skripten und dem Filesystem mit den Daten. Da es sich bei PTS um ein System handelt, das Supportfälle von Kunden mit bestehenden Installationen lizenzierter Produkte von IXOS betrifft, spielen auch alle Daten rund um den Kunden und das Produkts bzw. dessen Installation beim Kunden eine Rolle. Deshalb ist außer den Daten im PTS auch ein Zugriff auf die Datenbestände der CID (siehe 2.2) und des MKIS (siehe 2.1) notwendig. Dazu werden Prozeduren der CID mitverwendet.

Das Perl-Skript `pts.cgi` ist für den Aufbau der Einstiegsseite mit der Liste der Hotliner, die gerade angemeldet sind, und der Liste der zu bearbeitenden Fälle zuständig.

Für die Suche in den Supportfällen wird `sup-search.cgi` verwendet, das Kunden findet, die Supportfälle besitzen, und deren offene oder alle Fälle anzeigt.

Die Bearbeitung eines Falls ist in `sup-case-nf.cgi` implementiert. Es baut die Webseiten mit den HTML-Formularen zum Ändern eines Falls auf und verarbeitet nach dem Submit auch die Eingaben. Je nach Berechtigung des Benutzers, die aus der Poolzugehörigkeit ermittelt wird, werden verschiedene Änderungsmöglichkeiten angeboten.

Datenmodell

Das PTS-Datenmodell ist im Filesystem als mehrstufige Verzeichnisstruktur realisiert. Das Wurzelverzeichnis ist hierbei `/ixos/tssys/pts/data/`. Darunter existiert für jedes unterstützte Produkt ein Unterverzeichnis mit dem Produktnamen als Verzeichnisnamen. Hierunter existieren Unterverzeichnisse zum Gruppieren von Tausenderblöcken von Supportfällen. Unter diesen Gruppenverzeichnissen liegen die Supportfälle als Verzeichnisse mit den Fallnummern als Verzeichnisnamen. In den Fallverzeichnissen liegen neben der `.status`-Datei auch Emails und Logdateien.

Beispiel:

Der Fall mit der Nummer 23678 für das Produkt ARCHIVE würde dann in dem Verzeichnis `/ixos/tssys/pts/data/ARCHIVE/023/23678/` zu finden sein.

Die Stammdaten eines Falls befinden sich in der Datei `.status` im Fallverzeichnis. Eine solche Datei enthält fortlaufend Feldnamen, Zeitpunkt und Benutzer der Änderung und den Wert des Felds. Abbildung 2.3 zeigt eine `.status`-Datei mit einem Feld namens Subject und dem Wert „Fragen zum NFS-Client“, das vom Benutzer matth am 8.1.1998 um 11:55 Uhr GMT geändert worden ist und ein Feld mit dem Namen Customer, das den Wert 90377 enthält und ebenso zuletzt am 8.1.1998 um 11:55 Uhr GMT vom Benutzer matth geändert worden ist.

```
Subject(19980108.1155,matth):
Fragen zum NFS-Client
.
Customer(19980108.1155,matth):
90377
.
```

Abbildung 2.3: Ausschnitt aus einer PTS `.status`-Datei

Wird der Wert eines Felds geändert, so bleibt der bisherige Wert in der `.status`-Datei erhalten und der neue Wert wird an das Ende der Datei angehängt. In Abbildung 2.4 ist eine `.status`-Datei zu sehen in der das Subject des Beispiels aus Abbildung 2.3 vom Benutzer oliver am 9.1.1998 um 13:31 Uhr auf „Fragen zum ARCHIVE NFS-Client“ geändert.

```
Subject(19980108.1155,matth):
Fragen zum NFS-Client
.
Customer(19980108.1155,matth):
90377
.
Subject(19980109.1331,oliver):
Fragen zum ARCHIVE NFS-Client
.
```

Abbildung 2.4: PTS `.status`-Datei mit geändertem Subject-Feld

Daraus ergibt sich, daß zur Ermittlung des aktuellen Werts eines Felds immer die ganze `.status`-Datei eingelesen werden muß. Ein Vorteil dieses Datenmodells ist es aber, daß alle Veränderungen an einem Fall nachvollziehbar sind. Das ist eine entscheidende Voraussetzung für ISO 9001. Außerdem ist der Mehraufwand beim Einlesen vertretbar, weil ein Fall und damit auch eine `.status`-Datei wegen der begrenzten Lebensdauer[2] durchschnittlich nur 10 KB groß wird[3].

Zusätzliche Informationen zu einem Fall, wie z.B. Logfiles und Emails, werden in eigenen

[2] vom Öffnen bis zum Schließen eines Falls vergingen in der Zeit von Dezember 1996 bis Juli 2000 durchschnittlich 17 Tage

[3] nach dem Schließen eines Falls wird der Zugriff auf Fall nur noch lesend erlaubt

Dateien im Fallverzeichnis abgespeichert. Bei der Anzeige eines Falls im Webbrowser werden diese Dateien als Links dargestellt, so daß sie per Mausklick geöffnet werden können.

anlegen eines neuen Falls

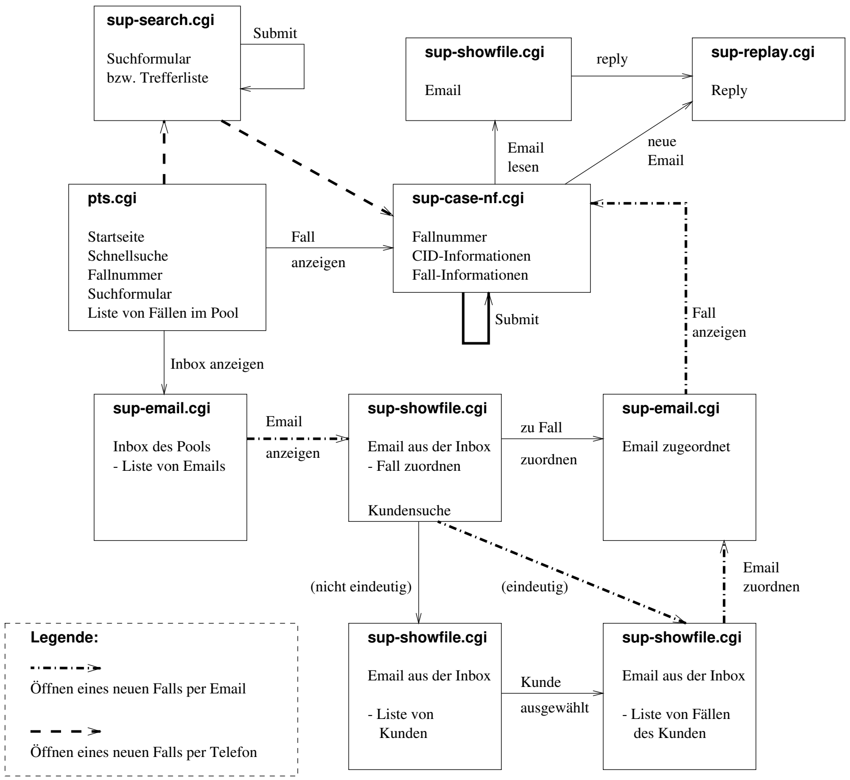

Abbildung 2.5: Arbeitsabläufe im PTS

Eingang per Hotline

Wenn ein Kunde bei der Hotline anruft, wird zunächst im PTS nach seinen offenen Fällen gesucht (`sup-search.cgi`). Handelt es sich um ein neues Problem, wird ein neuer Fall geöffnet (`sup-case-nf.cgi`). Der Fall wird dabei automatisch dem Kunden zugeordnet. Es müssen aber zumindest noch die Problembeschreibung und ein Verantwortlicher Support-Mitarbeiter eingetragen werden.

Eingang per Email

Meldet ein Kunde ein Problem per Email, landet diese Email zunächst in der Inbox des Pools. Jedem Pool ist eine Emailadresse zugeordnet, über die der Kunde den Support erreicht.

2.4 Expert Service Center (ESC)

Das ESC ist ein webbasiertes Informationssystem für Kunden und Mitarbeiter der IXOS, das auf einer SUN Enterprise 450 unter SunOS 5.6 mit Apache 1.3.6 läuft. Der Apache enthält die Module mod_ssl und mod_jserv. SSL wird verwendet, um die teilweise vertraulichen Daten bei der Übertragung vor dem Zugriff durch Dritte zu schützen. Das Modul mod_jserv dient als Connector zwischen dem Apache und der Servlet-Engine Tomcat. Das ESC enthält neben Produktinformationen und Problemlösungen auch Patches.

Technische Ansprechpartner von Kunden erhalten automatisch mit einem Produkt von IXOS einen Account für das ESC. Somit haben sie unmittelbar Zugang zur Dokumentation und den anderen Hilfen.

Oberfläche

Auf das ESC wird ausschließlich über Web-Browser (siehe Abb. 2.6) zugegriffen. Nachdem sich der Anwender erfolgreich angemeldet hat, gelangt er auf die Begrüßungsseite und kann durch die hierarchische Informationsstruktur browsen. Alternativ wird eine Suchmaschine [4] angeboten. Die Suche kann auf bestimmte Produkte, Produktversionen, Sprachen und Dokumenttypen wie PDF oder HTML eingeschränkt werden.
Zu jedem Account existiert ein Profil, das der jeweilige Benutzer selbst ändern kann. Im Profil steht neben dem Namen auch die Emailadresse, der Firmenname und weitere Optionen, die die Sicht auf das ESC betreffen.
Ist offensichtlich ein Problem noch nicht bekannt, besteht die Möglichkeit eine Frage zu stellen. Dazu wird ein Formular mit einer Kurzfassung und einer ausführlichen Beschreibung des Problems ausgefüllt und abgeschickt. Die Frage wird dann von Support-Mitarbeitern bearbeitet und die Lösung im ESC veröffentlicht. Der Fragesteller wird außerdem direkt über die Lösung benachrichtigt. Im Gegensatz zur Anfrage an der Hotline ist die Reaktionszeit auf ESC-Anfragen nicht garantiert.

[4] Verity, Document Navigator, http://www.verity.com

Abbildung 2.6: Ansicht des ESC

Authentifizierung und Autorisierung

Für den Zugriff auf das ESC wird ein Account benötigt, der für den Ansprechpartner des Kunden oder Partners angelegt wird. Dieser wird vom ESC-Team generiert und besteht aus einem Benutzernamen und einem zugehörigen Paßwort. Der Namespace der Benutzernamen ist unterteilt in

- cXXXXXX für Kunden, XXXXXX steht für die Partnernummer im MKIS
- pXXXXXX für Partner, XXXXXX steht für die Partnernummer im MKIS
- <name> für Mitarbeiter, <name> entspricht dem Unix-Login

Ein Account entspricht einem Unterverzeichnis unter `/big/esc/user`. In diesem Verzeichnis befindet sich die beiden Dateien `profile.cfg` und `lastaccess`.
`profile.cfg` enthält das Profil des Benutzers. Dazu zählen neben dem Namen, der Emailadresse und der Firma auch seine persönlichen Einstellungen des ESC, wie z.B. ob JavaScript oder Frames bevorzugt werden oder wie gross HTML TextAreas sein sollen. Die Datei besteht aus Schlüssel-Wert-Paaren.

```
*#!$realname: Oliver Maul
*#!$email: oliver.maul@munich.ixos.de
*#!$company: IXOS Software AG
*#!$linking: 1
*#!$areacol: 40
*#!$arearow: 8
*#!$userview: *esc_customers
```

Abbildung 2.7: Beispiel einer ESC `profile.cfg-Datei`

Die Datei `lastaccess` enthält als einzigen Wert den Zeitpunkt der letzten Anmeldung.

Die Paßwörter der Accounts werden getrennt von den Profilen in mehreren Paßwortdateien unter `/big/esc/etc/` abgelegt. Es gibt für die Gruppen Kunde, Partner und Mitarbeiter jeweils eine eigene Datei - `passwd_customers`, `passwd_partners` bzw. `passwd_intern` - , deren Zeilenformat <Login>:<Cryptstring> entspricht. `passwd_intern` wird dabei aus einer NIS-Map gewonnen. Für den Webserver werden die drei einzelnen Passwortdateien noch in eine einzige zusammengefaßt.

Der dem Benutzer bereitgestellte Funktionsumfang richtet sich nach seiner Gruppenzugehörigkeit. Im ESC existieren die vordefinierten Gruppen Kunde, Partner und Mitarbeiter. Die Zugehörigkeit der Accounts zu den Gruppen Kunde, Partner oder Mitarbeiter ergibt sich aus der Zugehörigkeit zu einer der passwd-Dateien. Zudem existieren besondere ESC-Gruppen, die z.B. festlegen, ob ein Benutzer das Recht besitzt, neue Accounts zu generieren oder ob er über neue Fragen benachrichtigt werden soll.

Im Gegensatz zur CID und dem PTS ist der Zugriff aus dem Internet über das https-Protokoll (TCP Port 443) freigeschaltet. Der Server steht jedoch im Intranet und hat dadurch Zugriff auf alle internen Ressourcen. Daraus ergeben sich hohe Sicherheitsanforderungen an alle aktiven Webinhalte, die auf dem Webserver ausgeführt werden, insbesondere für die in dieser Diplomarbeit zu entwickelnde Benutzerschnittstelle für die CID und das PTS.

Implementierung

Das ESC besteht aus einigen Perl-Skripten und den Verity Binaries, die unter `/big/esc/cgi-bin`, `/big/esc/lib/` bzw. `/big/esc/verity/_solaris/` abgelegt sind. Die Skripten erzeugen die HTML-Seiten, parsen Daten, die von Formularen stammen, speichern neue Dokumente und verschicken Emails z.B. an Moderatoren.
`index.cgi` enthält den äußeren Frameset und sorgt für die Aktualisierung der lastaccess-Datei des Benutzers.
Der Inhalt der einzelnen Frames wird durch weitere CGI-Skripten generiert. Der linke Frame, der den Navigationsbaum enthält, wird durch `esc-treenci.cgi` erzeugt. Je nachdem welchen Knoten der Benutzer im Navigationsbaum auswählt wird eines der weiteren Skripten aufgerufen, um den rechten Frame mit Inhalt zu füllen. Hier wird erkennbar, daß eine Erweiterung des ESC um neue Funktionalität leicht durch neue CGI-Skripten erreicht werden kann.

Datenmodell

Die Dokumente des ESC sind ausschließlich auf dem Filesystem unter `/big/escdata/` gespeichert. Jeder Ordner enthält eine `0.inf-Datei`, die Informationen zum jeweiligen Ordner enthält. Beispielsweise werden darin die Moderatoren, die Schlüsselwörter und die Berechtigungen festgelegt.

In jedem Ordner können Dokumente verschiedenen Typs abgelegt werden. Neben den üblichen Typen HTML , PDF , PPT , DOC oder XLS können auch standardisierte KNW-Dokumente, die die Strukturierung von Problemlösungen unterstützen, verwendet werden. Zu jedem Dokument wird zusätzlich eine inf-Datei (siehe Abb. 2.8) mit dem gleichen Namen und im gleichen Verzeichnis gespeichert. Diese inf-Datei hat das gleiche Format wie eine knw-Datei, mit dem Unterschied, daß sie ausschließlich die Strukturierungselemente enthält.

Durch die Schlüsselwörter wird der Inhalt des zugehörigen Dokuments beschrieben und die Zugriffsrechte der Gruppen festgelegt.

Für die effiziente Suche über alle Dokumenttypen wird durch den Document Navigator von Verity ein Index verwaltet. Bei der Suche werden die Dokumente mit einer Bewertung versehen, nach der die Antwortmenge sortiert werden kann, um relevante Informationen gezielt zu finden.

2.5 Zusammenspiel der Systeme

Die zuletzt vorgestellten Systeme MKIS, CID, PTS und ESC werden nun als ganzes betrachtet.

Die Basisdaten für alle Systeme liefert das MKIS. Es enthält Stammdaten, die den Kunden und dessen Ansprechpartner beschreiben. Dazu zählen unter anderem Adresse, Telefonnummer und Emailadresse. Direkt auf diese Daten baut das CID auf. Es enthält technische Daten zu Installationen bei den Kunden. Der Zugriff auf die Kundenstammdaten geschieht über die RFC-Schnittstelle des R/3. Das PTS baut wiederum auf die Installationen im CID auf. Es verwaltet die Supportfälle zu den Installationen im CID.

```
docid: 0913395482-570
doctype: meta
status: released
title: Wahl der Größe der Datenbank unter 3.0A
author: markusw
pages:
reply_to: robertm
approved_by: reginas
keywords:
approve_date: 19991019
creation_date: 19971218
modification_date:
expiry_date:
language: de
confidentiality: 0
hidden: 0
access: *esc_intern,
category: installation
platform: ALL
components: Database
arc_version_from: 3.0A
arc_version_till: 3.0A
r3_version_from: -
r3_version_till: -
```

Abbildung 2.8: Beispiel einer ESC `inf`-Datei

Als relativ unabhängig von MKIS, CID und PTS stellt sich das ESC dar. Der einzige Zusammenhang zum MKIS besteht in der Generierung von Kunden-Accounts für das ESC. Nach der Installation erhält ein Ansprechpartner beim Kunden, der im MKIS eingetragen und im CID einer Installation zugewiesen ist, einen Account. Dazu werden seine Daten wie Name, Emailadresse und Firmenname automatisch aus MKIS in das Benutzerprofil übernommen. Dadurch ist es dem Benutzer möglich seine Emailadresse zu ändern. Diese Änderung wirkt sich auch nur auf das Benutzerprofil im ESC aus, aber nicht auf das MKIS. Dadurch lassen sich solche Änderungen später noch nachvollziehen und evtl. wieder der Ausgangszustand herstellen. Außerdem ist es aus Sicherheitsgründen unerwünscht, Änderungen an den Kundenstammdaten durch externe Personen zuzulassen.

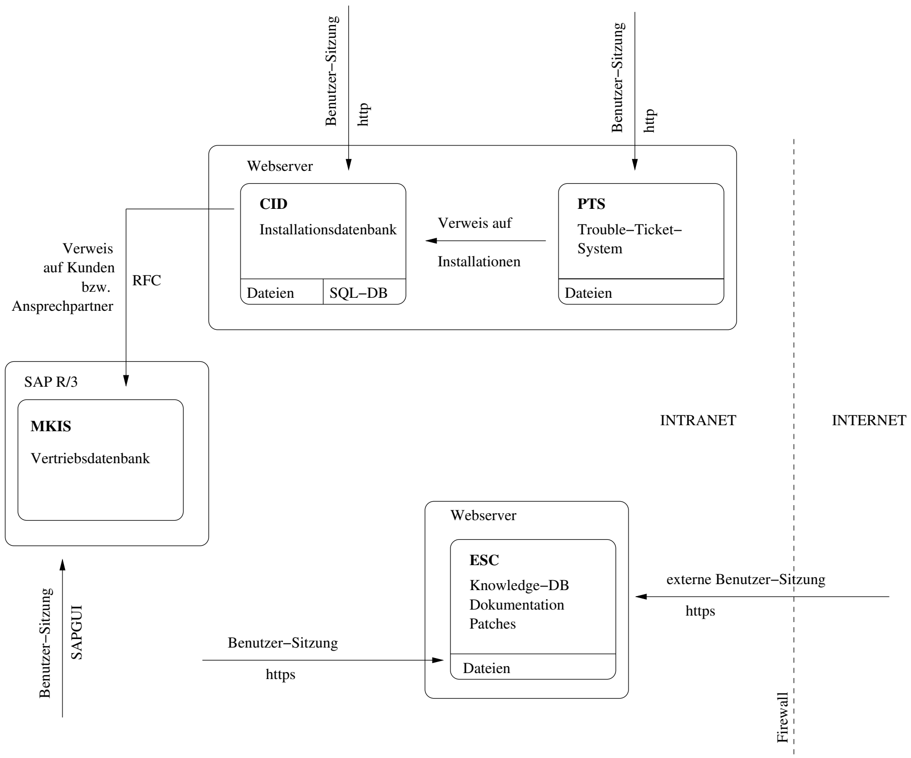

Abbildung 2.9: Zusammenspiel von MKIS, CID, PTS und ESC

Kapitel 3

Andere vergleichbare Systeme

Nachdem bisher die bei IXOS verwendeten Systeme MKIS, CID, PTS und ESC vorgestellt wurden, sollen nun alternative, bei anderen Firmen eingesetzte Online Support Systeme und ihre Eigenschaften vorgestellt werden. Dabei sollen Methoden zur Abwicklung von Kunden-Support über das WWW analysiert werden. Besonders die Benutzer- und Berechtigungsverwaltung spielt eine übergeordnete Rolle, weil diese in den Systemen PTS und CID bisher praktisch überhaupt nicht vorkommen.

3.1 SAPNet

Bei der SAP AG wird für den Kunden-Support das SAPNet betrieben. Das SAPNet ist über ein SAPGUI und über WWW (http://sapnet.sap-ag.de) zugreifbar. Da die Kunden von IXOS überwiegend auch R/3-Nutzer und damit SAP-Kunden sind, muß sich das zu schaffende IXOS-System daran messen lassen.

Benutzerverwaltung

SAP-Kunden können ihre Benutzerdaten und -berechtigungen weitgehend selbst verwalten. Die SAP richtet jedem Kunden einen Administrator ein, der über alle Berechtigungen (siehe Berechtigungskonzept) verfügt. Dieser kann beliebig viele Benutzer einrichten. Damit ist der Kunde auch in der Lage, seine Daten selbst zu bereinigen (Löschen von Benutzern). Die Pflege der Personendaten, insbesondere der Telefon- und Faxnummern und Email-Adressen, obliegt ebenfalls weitgehend den Kunden.

Berechtigungskonzept

Die Berechtigungen eines Benutzers legen fest, welche Funktionen der Benutzer im SAPNet ausführen darf. Manche Berechtigungen besitzen zusätzlich einen Geltungsbereich (Wert).

SAP unterscheidet gegenwärtig zwischen vier Berechtigungs-Typen. Die Vierteilung ergibt sich daraus, dass sich die Berechtigungen auf unterschiedliche Ebenen beziehen:

- Administrator-Berechtigung: Bezugspunkt dieser Berechtigung ist der einzelne Benutzer, der administriert werden darf. Die Zuordnung der Benutzer zu einem Administrator kann entweder durch namentliche Einzelzuordnung geschehen (starr), oder generisch durch Zuordnung der Kundennummer. In diesem Fall bezieht sich das Administrationsrecht auf alle jetzigen und zukünftigen Benutzer des Kunden, d.h., das Administratorrecht wird ständig aktualisiert.
- Benutzerbezogene Berechtigungen oder Ja-/Nein-Berechtigungen sind Rechte, die sich nur auf den Benutzer beziehen, d. h. Berechtigungen, die keinen zusätzlichen Bezug auf Kundenstrukturen haben, z.B. Hinweis suchen, Entwicklungsanträge erfassen.
- Berechtigungen auf Kundennummer-Ebene.
- Installationsrechte: Die meisten Berechtigungen beziehen sich auf Kunden-Installationen. So gibt es die Berechtigung, für eine bestimmte Installation eine Kundenmeldung zu erfassen.

Hier ein Auszug aus den konkreten Berechtigungen:

- **Administrator-Berechtigung**
  Berechtigung, anderen Benutzern die Berechtigungen zu erteilen oder zu nehmen, die der Administrator selbst besitzt.
- **Benutzerdaten verwalten**
  Diese Berechtigung umfasst folgendes:
  - Recht, neue Benutzer anzufordern
  - Recht, Benutzer zu löschen
  - Recht, Kontaktpersonen aus der Adressliste zu löschen
  - Recht, die Passwörter anderer Benutzer zurückzusetzen
  - Recht, die Kommunikationsdaten eines Benutzers zu ändern
- **Hinweise suchen**
  Berechtigung, im SAPNet - R/3 Frontend nach SAP-Hinweisen zu suchen.

- **Lizenzschlüssel anfordern**
  Berechtigung zum Anfordern von Lizenzschlüsseln bei SAP für neue Systeme oder Systeme mit geänderter Hardware-Plattform.

- **Produktionsdaten verwalten**
  Berechtigung zum Verwalten der technischen Daten der SAP-Systeme einer Firma.

- **SAP-Meldungen an SAP senden**
  Berechtigung, Problemmeldungen zur Bearbeitung an SAP zu senden.

- **SAP-Meldungen anlegen**
  Berechtigung, Problemmeldungen anzulegen.

- **SAP-Meldungen quittieren**
  Berechtigung, die von SAP vorgeschlagene Lösung zu einer SAP-Meldung zu quittieren und die Meldung damit zu schließen.

- **SAP-Meldungen wiedereröffnen**
  Berechtigung, eine SAP-Meldung erneut zu öffnen, wenn SAP bereits eine Lösung angeboten hat, diese sich aber als unzureichend herausgestellt hat.

Administratorkonzept

Ein Administrator ist ein Benutzer, der die Administrator-Berechtigung hat. Mit der Administrator-Berechtigung kann er alle Rechte, über die er selber verfügt, an andere Benutzer weitergeben bzw. diese löschen. Das bestehende Administratorkonzept ist sehr flexibel und lässt sich von den Kunden dazu nutzen, verschiedene Anforderungen und Hierarchien abzubilden.

Die Administrator-Berechtigung besteht aus drei Ebenen:

1. Welche Benutzer kann der Administrator administrieren?
   - Alle Benutzer, die ihm namentlich zugeordnet wurden (starr).
   - Alle jetzigen und zukünftigen Benutzer des Kunden oder - ein Sonderfall - des Customer Competence Center (CCC).
2. Welche Berechtigungen kann der Administrator vergeben?

   - Die Berechtigungen, die er selbst besitzt.
     Normalerweise haben Administratoren alle oder fast alle Berechtigungen. Ein Kunde kann aber auch Administratoren definieren, die nur wenige Berechtigungen vergeben können.
3. Für welche Werte kann der Administrator Berechtigungen vergeben?
   - Nur für die Werte, für die er selbst die entsprechende Berechtigung besitzt.
     Beispiel: Hat der Administrator die Berechtigung, SAP-Meldungen für die Kundennummer 4711 zu erfassen, dann kann er anderen Benutzern diese Berechtigung für alle Installationen erteilen, die zu Kundennummer 4711 gehören.

Das Administrations- und Berechtigungskonzept des SapNet ist sehr fein granuliert. Durch die Aufteilung von Berechtigungen in Berechtigungsobjekte und Wertebereiche lassen sich zudem recht einfach Administrationshierarchien abbilden. Das hat den Vorteil, daß sich zukünftige Veränderungen im Supportbereich leicht adaptieren lassen.
Ein Nachteil ist die Komplexität des Konzepts.

3.2 RedHat

Bei Red Hat, Inc. (http://www.redhat.com) besteht ein Online Support zur Unterstützung beim Einsatz von Red Hat Linux, Red Hat Secure Web Server und Open Source Software wie Apache oder Sendmail.

Benutzerverwaltung

Die Benutzerverwaltung besteht in einer einfachen Registrierung der Benutzer durch ein Webformular mit Informationen zur Person und einem Accountnamen und einem Paßwort. Die Registrierung steht jedem offen und ist mit einem Mailingliste gekoppelt, auf der jeden Monat Neuigkeiten über Red Hat verteilt werden.

Berechtigungskonzept

Nach der Registrierung ist der nächste Schritt das Anmelden von den Produktnummern zu den Produkten, die von Red Hat gekauft wurden. Die Produktnummern werden auf ihre Gültigkeit überprüft und erlauben es Red Hat, die in Anspruch genommen Supportleistungen auf lizenzierte Produkte einzuschränken. Durch die Zuordnung der Produktnummern zu einem Benutzer ist auch eine Mehrfachregistrierung von Produkten ausgeschlossen.

Öffnen eines neuen Supportfalls

Nach der Anmeldung mit einem gültigen Account, dem mindestens eine Produktnummer zugeordnet ist, erhält man Zugang zum Online Support System. Hier ist es zunächst möglich einen neuen Supportfall zu öffnen.

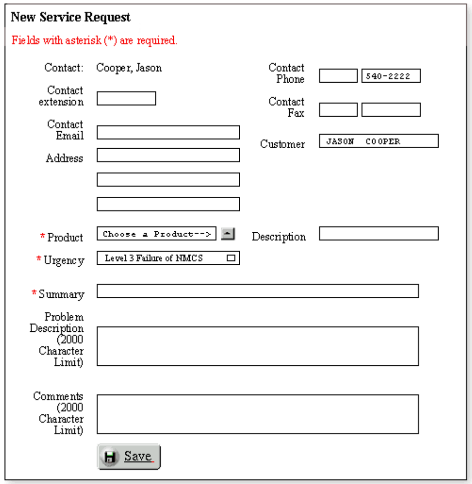

Abbildung 3.1: Anlegen eines neuen Supportfalls

Die Angaben zum Ansprechpartner und zu den zum Support zugelassenen Produkt sind bereits vorgegeben. Das Problem wird durch eine Zusammenfassung und eine Priorität beschrieben. Zusätzlich ist es möglich eine genauere Beschreibung und einen Kommentar anzugeben.

3.3 Oracle

Bei Oracle (http://www.oracle.com) besteht ein Online Support zur Unterstützung beim Einsatz von Oracle Datenbankprodukten. `MetaLink` erlaubt das Erstellen neuer TARs (Technical Assistance Request) und das Ansehen existierenden TARs.

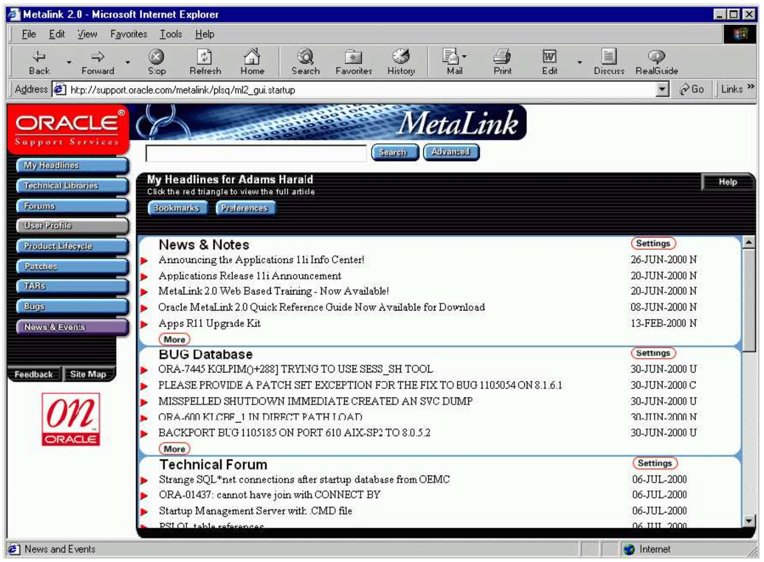

Abbildung 3.2: Oracle MetaLink

Benutzerverwaltung

Für einen Kunden wird ein Benutzer angelegt, der Administrationsrechte für andere Benutzer erhält. Dadurch können weitere Benutzer angelegt und verwaltet werden. Jeder Benutzer besitzt ein Profil in dem er seine persönlichen Daten wie Name und Adresse hinterlegt. Außerdem sind hier die Privilegien des Benutzers gespeichert.

Berechtigungskonzept

Das Berechtigungskonzept von MetaLink sieht folgende Privilegien vor:

Open, update and close TARs	Yes/No
Read TARs	Yes/No
Forum Access	Yes/No
Add Support Identifier	Yes/No
Remove Support Identifier	Yes/No
Configure „My Headlines“	Yes/No
Access Patch Download	Yes/No
Administer other users	Yes/No
Your administrator	<andere Benutzer>

Das Privileg zum Administrieren anderer Benutzer kann nicht an selbst angelegte Benutzer vererbt werden. Die Privilegien selbst angelegter Benutzer können nur durch die eingetragenen Administratoren dieses Benutzers geändert werden. Durch dieses Konzept ergibt sich eine zweistufige Hierarchie der Benutzer. Die von Oracle angelegten Benutzer mit Administrationsrecht und die von diesen angelegten Benutzer.

Verwalten von TARs

Ein Benutzer mit dem Privileg `Open, update and close TARs` kann neue TARs anlegen und bestehende ändern. Beim Anlegen wird zuerst eine Kurzbeschreibung abgefragt, anhand der eine Suche über die Knowledgebase gestartet wird. Die Treffer werden als mögliche Lösungsvorschläge ausgegeben. Führt kein gefundenes Dokument zur Lösung, wird im Anschluß ein Fragebogen angezeigt, der detailierte Fragen zum Problem enthält. Die Antworten werden im neuen TAR gespeichert.

Ein Benutzer mit dem Privileg `Read TARs` kann existierende TARs ansehen. Die TARs können nach Kriterien wie „letze Änderung am TAR“ oder „Status“ durchsucht werden. Die Trefferliste enthält die wichtigsten Informationen der Fälle. Es kann dann ein Fall ausgewählt werden, der in vollem Umfang angezeigt werden soll.

3.4 Bewertung der Systeme

Die zuletzt betrachteten fremden Online-Support-Systeme von SAP, RedHat und Oracle zeigen sehr unterschiedliche Ansätze was die Selbstverwaltung des Kunden betrifft.

Im SAPNet besteht für den Administrator des Kunden nicht nur die Möglichkeit neue Benutzer zu erstellen, sondern auch Verantwortung an diese zu delegieren und eine ganze Berechtigungshierarchie aufzubauen.

Bei RedHat besteht dagegen nur eine eingeschränkte Form der Verwaltung von Benutzern. Den angelegten Benutzern können keine Berechtigungen zugewiesen oder weggenommen

werden. Daraus ergibt sich ein sehr eingeschränkter Nutzungsbereich. Denn alle Benutzer einer Firma können alle Supportfälle der registrierten Produkte einsehen und ändern.

MetaLink von Oracle ermöglicht außer dem Anlegen neuer TARs und Bearbeiten bestehender TARs auch die Verwaltung von Privilegien der einzelnen Benutzer. Es ist möglich selbst neue Benutzer anzulegen und deren Privilegien festzulegen. Das Berechtigungskonzept hinter MetaLink ist allerdings nicht so flexibel wie das von SAPNet. Es können keine administrativen Privilegien vererbt werden. Und die Privilegien sind vorgegeben und können nicht parametrisiert werden. Im SAPNet können Berechtigungen in Abhängigkeit von Attributwerten eines Objekts, wie einem Supportfall, vergeben werden. So kann z.B. ein Benutzer das Recht erhalten nur Fälle mit niedriger Priorität anzulegen.

Für eine flexible Gestaltung der Berechtigungsverwaltung spricht, daß damit ein und dieselbe Benutzerschnittstelle für mehrere Benutzergruppen mit unterschiedlichen Privilegien verwendet werden kann. Denn abhängig von der Berechtigung des aktuellen Benutzers werden Funktionalitäten erlaubt oder verboten. So sind sogar Berechtigungen in Abhängigkeit von Laufzeitparametern wie z.B. eingegebene Formulardaten möglich.

Im Gegensatz dazu könnten für jede Gruppe von Benutzern eine eigene Benutzerschnittstelle implementiert werden. Kommt eine neue Gruppe hinzu, müßten neue Webseiten und -formulare implementiert werden, die genau die erforderlichen Funktionen gestatten. Hierbei kann aber nicht auf Laufzeitparameter reagiert werden. Dadurch ist das Konzept weniger mächtig.

Kapitel 4

Analyse

Nachdem bis jetzt bestehende Systeme vorgestellt worden sind und zuletzt andere Online-Support-Systeme betrachtet wurden, wird nun die Schnittstelle zum Kunden analysiert und Anwendungsfälle bestimmt. Da PTS und CID bisher ausschließlich im Intranet zur Verfügung standen, waren sie zwangsläufig den Mitarbeitern der IXOS vorbehalten. Bei der ursprünglichen Entwicklung stand noch garnicht nicht fest, daß sie außer von den Mitarbeitern auch von anderen Benutzern verwendet werden sollen.

Zu Beginn wird das zu entwickelnde System als „Black Box“ betrachtet und Anwendungsfälle aufgezeigt. Diese Anwendungsfälle werden dann näher betrachtet und analysiert. Zuletzt werden die Sicherheitsanforderungen des Systems spezifiziert.

4.1 Vorgehen

Zur Analyse der Anforderungen werde ich mich an OMT [RB91] und den Use-Cases von OOSE [J$^+$92] orientieren.

Die Schritte zur objektorientierten Analyse sind die folgenden:

- Use-Case Analyse (Beschreibung der Anforderungen)
- Entwicklung eines Objektmodells
- Entwicklung des dynamischen Modells
- Validieren, Überarbeiten und Erweitern der Modelle

4.2 PTS

Durch die fehlende direkte Zugriffsmöglichkeit der Kunden auf das PTS werden alle Vorgänge über andere Wege, hauptsächlich Telefon und Fax, abgewickelt.

Im PTS bestehen zur Zeit hauptsächlich die folgenden Anwendungsfälle:

- Anlegen neuer Fälle
- Ansehen bestehender offener und geschlossener Fälle
- Bearbeiten offener Fälle
- Schließen offener Fälle

Aktor in all diesen Fällen ist ein Support-Mitarbeiter von IXOS.

Ziel der Diplomarbeit ist es unter anderem, die Zugriffswege auf das PTS so zu erweitern, daß Kunden als Aktor in den genannten Anwendungsfälle auftreten können. Durch diese Erweiterung entstehen einige neue Anforderungen an die Berechtigungsprüfung des PTS. Das neue Anwendungsfallmodell ist in Abb. 4.1 dargestellt.

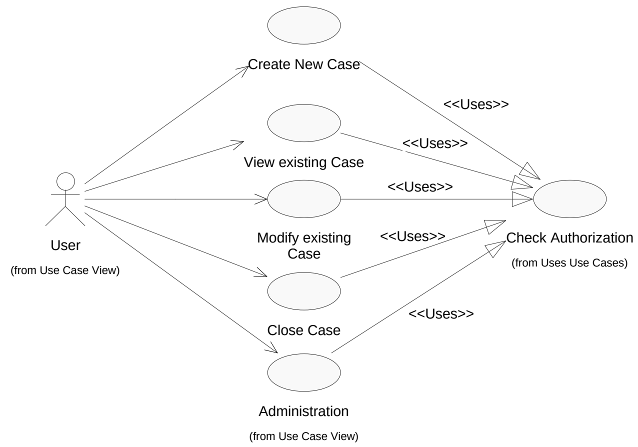

Abbildung 4.1: PTS Anwendungsfälle

Hauptsächlich kam der Anwendungsfall `Check Authorization` hinzu, der von allen anderen Fällen verwendet wird. `Check Authorization` ist dafür verantwortlich, die Berechtigung des Aktors zu überprüfen. Aus verständlichen Gründen ist es nicht vertretbar, daß ein

Kunde beliebig Supportfälle ändern oder ansehen darf. Deshalb wird im nächsten Schritt genauer auf die einzelnen Anwendungsfälle eingegangen und bestimmt, was genau darunter zu verstehen ist und welche Sicht die Kunden jeweils auf das PTS erhalten sollen.

4.2.1 Anlegen neuer Fälle

Tritt ein Problem mit einem Produkt auf, steht als erstes das ESC zur Verfügung, um festzustellen ob bereits eine Lösung bekannt ist. Hier sind Lösungshinweise, Patches und dergleichen hinterlegt. Findet der Kunde jedoch keine Hinweise, soll er die Möglichkeit haben, einen Supportfall zu öffnen.

Neuen Fall eröffnen

Handelt es sich um ein neues Problem, füllt er ein HTML-Formular aus, das neben der Kunden- und Installationsnummer eine Kurzfassung und eine ausführliche Beschreibung des Problems abfragt. Es soll möglichst einfach eine möglichst genaue Fehlerbeschreibung erfaßt werden können. Deshalb ist es nicht sinnvoll, vom Kunden zu verlangen, alle Felder (siehe Abb. 4.2) ausfüllen zu müssen. Zum Teil wird er die Bedeutung einiger Felder garnicht kennen oder sie mit sinnlosen Werten füllen.

- Kurzbeschreibung
- Installationsnummer
- Telefonnummer des Ansprechpartners beim Kunden (in Zukunft MKIS-Partnernummer)
- Emailadresse des Ansprechpartners beim Kunden (in Zukunft MKIS-Partnernummer)
- detaillierte Beschreibung
- Priorität des Falls
- Betroffene Komponente
- Problem-Klasse
- momentaner Status
- zugesagte Antwortzeit
- Verantwortlicher Support-Mitarbeiter
- Planed-Activity (vom Hotliner als nächstes geplante Lösungsschritte)
- Für den Fall aufgewendete Zeit
- Lösungsvorschlag
- Änderungsgrund für Änderungen des Status oder der Antwortzeit
- bereits durchgeführte Aktivitäten

Abbildung 4.2: PTS Informationen zu einem Fall

Der Vorgang zum Erstellen eines Supportfalls soll für den Kunden so einfach wie möglich werden, um eine hohe Akzeptanz zu erreichen. Dazu sollen Informationen, die automatisch ermittelt werden können, wie z.B. die Emailadresse des Ansprechpartners, als Standardwerte vorgeschlagen werden. Neue Fälle dürfen nur für Installationen des Kunden angelegt werden. Dazu wird die Installation, der ein Fall zugeordnet wird, vor dem Erfassen des Falls vom Benutzer aus einer Liste ausgewählt. Die zugesagte Antwortzeit, Lösungsvorschlag, Änderungsgrund und aufgewandte Zeit sollen vor dem Kunden komplett versteckt werden, weil sie nur für den Support-Mitarbeiter von Interesse sind. Die beiden Felder, Kurzbeschreibung und Beschreibung, sollen vom Kunden ausgefüllt werden. Dazu sollen ihm Hilfen bereitgestellt werden, um die Qualität der Problembeschreibung zu verbessern.

Folgefall eröffnen

Entsteht ein Problem, nachdem ein vorhergehendes gelöst wurde, und es besteht ein Zusammenhang zwischen beiden, soll es möglich sein, einen „Folgefall“ zu erzeugen. Dazu sollen relevante Informationen aus dem alten Fall übernommen werden können. Dabei sollen beim Anlegen des neuen Falls alle Informationen des alten Falls als Standardwerte vorgeschlagen werden, aber noch veränderbar sein.

4.2.2 Ansehen bestehender offener und geschlossener Fälle

Das Ansehen existierender PTS-Fälle ist sicher die interessanteste Funktion für den Kunden. Er will meist wissen, ob sich bereits jemand um sein Problem kümmert oder ob von ihm eine Reaktion erwartet wird. Dazu wird ihm zuerst eine Liste seiner Fälle geboten, wodurch er eine erste Übersicht erhält. Hier kann er dann einen Fall auswählen. Übersteigt die Anzahl seiner Fälle eine bestimmte Grenze, soll er die Möglichkeit zur Suche haben.

In bereits geschlossenen Fällen können Lösungsvorschläge nachgeschlagen werden, die möglicherweise verhindern, daß ein neuer Fall mit dem gleichen Inhalt eröffnet wird.

Fälle, die einer Installation zugeordnet sind, für die der Kunde keine Berechtigung hat, dürfen garnicht oder nur anonymisiert angezeigt werden. Denn PTS-Fälle enthalten unter Umständen vertrauliche Informationen wie Paßwörter und Beispieldaten, die Geschäftsgeheimnisse darstellen können (z.B. in Logfiles). Weil eine Anonymisierung von nicht formell spezifizierten Daten wie sie in Supportfällen vorkommen nur sehr schwer möglich ist, sollen hier nur PTS-Fälle, die einer Installation des Kunden zugeordnet sind, angezeigt werden können.

IXOS hat sich dagegen entschieden, vergangene Fälle direkt als Lösungsdatenbank zu verwenden. Lösungen für andere Kunden gehen unter Umständen von unterschiedlichen Voraussetzungen aus, die vom Kunden schwer zu durchschauen sind. Stattdessen werden Erkenntnisse, die aus Supportfällen entstanden sind, regelmäßig in das ESC übertragen.

Dort können sie aufgrund ihrer Aufbereitung (Anonym, Verständlichkeit) auch Kunden zugänglich gemacht werden. Dennoch dient das Ansehen vergangener eigener Fälle auch der Nachvollziehbarkeit der Supportdienstleistung. Aber eine Lösungssuche steht nicht im Vordergrund.

4.2.3 Ändern bestehender offener Fälle

Ein Fall ist bis zum Schließen ständig Änderungen unterworfen. Einige dieser Änderungen werden durch den Kunden ausgelöst, andere durch den Lösungsprozess innerhalb des Supports. Der Kunde wird immer dann eingreifen, wenn von ihm explizit eine Aktion verlangt wird. Dies wird durch den Status `open-feedback` angezeigt. Andere Änderungen wird der Kunde von sich aus vornehmen wollen. Dazu gehört z.B. wenn er die im Fall angegebene Emailadresse ändern oder den Fall vertagen will.

- anhängen von Dateien (Logfiles)
- anhängen von Kommentaren
- Eskalation veranlassen
- Priorität ändern
- vertagen

Abbildung 4.3: mögliche Änderungen durch Kunden an einem PTS Fall

Die unmittelbare Änderung der Priorität soll nur im Rahmen des Supportvertrags des Kunden möglich sein. Die Erhöhung der Priorität über diese Grenze hinaus wird auf Wunsch durch den verantwortlichen Supportmitarbeiter des Falls durchgeführt. So ist eine faire Behandlung aller Kunden und die Einhaltung der je nach Priorität engeren Antwortzeiten sichergestellt.

Die Eskalation eines Falls soll ebenfalls nur durch einen Mitarbeiter auf Wunsch des Kunden ausgelöst werden.

Der Kunde soll die Möglichkeit haben, einen Fall selbst zu vertagen, wenn er ihn nicht mehr für besonders dringend hält oder wegen Urlaub oder dringenderen Problemen verhindert ist.

Kommentare an einen offenen Fall anhängen

Ein Supportfall wird während seiner Bearbeitung um Kommentare erweitert. Solche Kommentare können weitere Details zum Problem oder aber Lösungsschritte enthalten. So kann z.B. der Supportmitarbeiter eine Notiz hinterlegen was er zuletzt gemacht hat, damit kein anderer Mitarbeiter, der den Fall in Zukunft bearbeitet, das gleiche ein zweites mal durchführt. Aber auch zur Kommunikation zwischen Mitarbeitern von First- und Second-Level-Support oder der Entwicklung können Notizen eingesetzt werden.

Bei den Kommentaren gibt es noch ein wichtiges Detail. Notizen werden in Zukunft nicht nur für die Kommunikation zwischen Mitarbeitern des Supports und der Entwicklungsabteilung verwendet sondern auch zwischen Support und Kunde. Derzeit lassen sich Kommentare nicht unterscheiden. Das heißt, daß der Kunde alle sehen würde. Um die interne Kommunikation weiterhin in den Kommentaren dokumentieren zu können, ohne daß diese für den Kunden einsehbar sind, muß in Zukunft zwischen internen und externen Kommentaren unterschieden werden können.

Interner Kommentar

Wird ein interner Kommentar erstellt, soll diese wie bisher behandelt werden und mit dem Fall dauerhaft abgespeichert werden. Diese wird dem Kunden nicht angezeigt werden.

Ein interner Kommentar kann durch Support-Mitarbeiter von IXOS erstellt werden. Das ist notwendig, um die interne Kommunikation zu ermöglichen. Partner werden ebenfalls interne Kommentare erstellen und einsehen können. Möglicherweise ist aber eine weitere Unterteilung notwendig, um die Kommunikation zwischen Partner und IXOS von einer rein internen Kommunikation unterscheiden zu können.

Externer Kommentar

Ein als extern markierter Kommentar wird ebenfalls dauerhaft mit dem Fall abgespeichert und auf Wunsch zusätzlich per Email an den im Fall angegebenen Ansprechpartner des Kunden geschickt. Dadurch erfährt der Kunde, daß sich etwas an dem Fall geändert hat und zugleich ist die Kommunikation im Fall dokumentiert. Ein externer Kommentar kann durch Support-Mitarbeiter von IXOS und durch den Kunden erfaßt werden.

Dateien an einen Fall anhängen

Neben den Notizen sind Logdateien ein weiterer wichtiger Bestandteil von Supportfällen. Aus ihnen können wichtige Hinweise gewonnen werden, die helfen, das Problem einzukreisen und zu beheben. Bisher wurden Dateien größtenteils per Email an den Support geschickt. Aber auch per Fax oder FTP (File Transfer Protocol) kamen schon Logdateien. Der Kunde soll zukünftig die Möglichkeit haben, Dateien bis zu einer bestimmten Größe über ein Webformular an einen Fall anzuhängen.

4.2.4 Fall schließen

Hält ein Kunde das Problem, das hinter einem seiner Fälle steht, für gelöst, soll er die Möglichkeit haben, den Fall als erledigt zu markieren. Dadurch wird es weniger offene

Fälle geben, die zwar gelöst sind, aber wegen fehlender Bestätigung des Kunden noch nicht geschlossen worden sind.

4.2.5 Administration

Um kleinere Verwaltungstätigkeiten dem Kunden zu überlassen und damit die Hotline zu entlasten, soll der Kunde die Berechtigungen seiner einzelnen Ansprechpartner im neuen System selbst verwalten können. Dazu soll es einen besonderen Ansprechpartner mit diesem Recht geben. Dieser kann den anderen entweder auch das Administrationsrecht geben oder ihnen beispielsweise Installationen zuweisen, deren Fälle dann durch sie abgewickelt werden können.

Berechtigungen verwalten

Benutzer mit Administrationsrecht sollen die Berechtigungen der ihnen zugeordneten Benutzer verwalten können. So kann bestimmten Benutzern beispielsweise nur das Recht zum Anzeigen von Fällen gegeben werden.

Installationen verwalten

Benutzer mit Administrationsrecht sollen auch anderen, ihnen zugeordneten Benutzern Installationen zuweisen können. Die Menge der Installationen, die anderen Benutzern zugeordnet werden kann, besteht aus den Installationen, denen der Administrator zugeordnet ist. Dadurch ist es dem Kunden möglich, für jede seiner Installationen einen eigenen Ansprechpartner zu benennen.

Eine Sonderrolle spielen hier die Customer-Competence-Center (CCC). Dies sind Kunden, die innerhalb ihres Konzerns für die einzelnen Installationen zuständig und Ansprechpartner für die Administratoren sind. Diese Beziehung ist in der CID durch das Feld `ExtSupport` realisiert. In den Installations-Einträgen des Konzerns in der CID steht im `ExtSupport`-Feld eine Referenz auf den CCC-Eintrag, der ebenfalls in der CID eingetragen ist.

4.3 CID

Durch die fehlende direkte Zugriffsmöglichkeit der Kunden auf die CID kann es zu nicht aktuellen Daten kommen, wenn sich an Installationen etwas ändert ohne das IXOS davon erfährt.

In der CID bestehen zur Zeit hauptsächlich die folgenden Anwendungsfälle:

- Anzeigen von Installationsdaten
- Ändern von Installationsdaten
- Anlegen neuer Installationen

Aktor in all diesen Fällen ist ein Support-Mitarbeiter von IXOS.

Ziel der Diplomarbeit ist es unter anderem, die Zugriffswege auf die CID so zu erweitern, daß wie im PTS Kunden als Aktor in den genannten Anwendungsfällen auftreten können. Eine Einschränkung oder Unterteilung wird der Anwendungsfälle wird unumgänglich sein. Besonders das Anlegen von neuen Installationen soll, wenn überhaupt, nur durch ausgewählte Partner- oder CCC-Kunden möglich sein. Durch diese Erweiterung entstehen einige neue Anforderungen an die Berechtigungsprüfung des CID. Das neue Anwendungsfallmodell ist in Abb. 4.4 dargestellt.

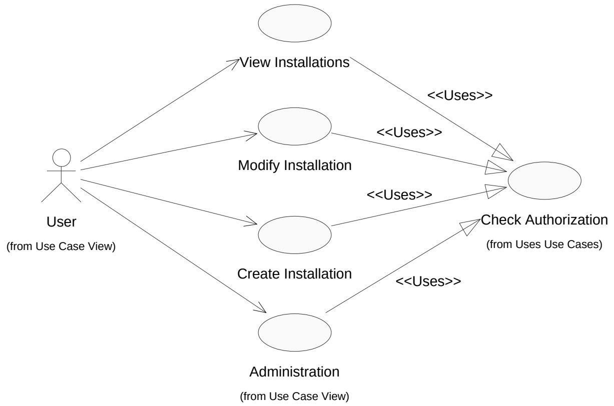

Abbildung 4.4: CID Use-Case Modell

Hauptsächlich kam der Anwendungsfall `Check Authorization` hinzu, der von allen anderen Fällen verwendet wird. `Check Authorization` ist dafür verantwortlich, die Berechtigung des Aktors zu überprüfen. Aus verständlichen Gründen ist es nicht vertretbar, daß ein Kunde beliebig Installationen ändern oder ansehen darf. Deshalb wird im nächsten Schritt genauer auf die einzelnen Anwendungsfälle eingegangen und bestimmt, was genau darunter zu verstehen ist und welche Sicht die Kunden jeweils auf die CID erhalten sollen.

4.3.1 Anzeigen von Installationsdaten

Der Kunde soll die Möglichkeit haben, die über ihn gespeicherten Stammdaten zu sehen. Dazu zählen die im MKIS gespeicherten Marketinginformationen, also Adresse, Telefon, Fax, Email, und die in der CID vorhandenen Informationen zu seinen Installationen. In der CID sind auch besonders schützenswerte Informationen zu Remote-Access-Zugängen für Wartungsarbeiten und Lizenzdaten gespeichert. Deshalb müssen die Berechtigungen hier besonders restriktiv sein.

4.3.2 Ändern von Installationsdaten

Außer dem reinen Ansehen der Daten von Installationen ist das Ändern ein wichtiger Punkt. Hier ist eine Unterscheidung der Felder genauso wichtig wie im PTS. Bei einigen Feldern wird eine Änderung nur indirekt durch eine zusätzliche Prüfung durch einen Support-Mitarbeiter möglich sein.

4.3.3 Erstellen neuer Installationen

Dies ist ein angedachter Anwendungsfall und soll hier nicht näher behandelt werden. Partner der IXOS sollen die Möglichkeit haben, die Daten zu ihren neuen Installationen im CID einzutragen. Allerdings wird Ihnen nicht erlaubt sein, die Marketinginformationen im MKIS einzutragen. D.h. vor dem Eintrag der ersten Installation durch den Partner muß durch IXOS der Kunde im MKIS erfaßt werden.

4.4 Sicherheitsanforderungen

Da es sich bei allen Daten im PTS und CID um schützenswerte Kundendaten handelt, spielt die Sicherheit eine große Rolle, wenn es darum geht, den Zugriff aus dem öffentlichen Internet heraus zuzulassen. Die Sicherheit einer Webanwendung ergibt sich aus folgenden Punkten [Bun97].

- der Sicherheit des WWW-Servers
- der Sicherheit des WWW-Clients und
- der Sicherheit der Kommunikationsverbindung zwischen beiden.

Zur Sicherheit des WWW-Servers zählt [Bun97]

1. Konzeptionierung des WWW-Servers
    - Festlegung der Sicherheitsziele

    - Anpassung der Netzstruktur
    - Grundlegende Voraussetzungen
    - Organisatorische Regelungen
2. Implementation des WWW-Servers
    - Umsetzung der IT-Grundschutz-Maßnahmen für den WWW-Rechner
    - Nutzung sicherer Kommunikationsverbindungen (Einsatz von S-HTTP, Verwendung von SSL)
    - Java, ActiveX (Schutz vor aktiven Inhalten)
3. Betrieb des WWW-Servers
    - regelmäßige Kontrolle
    - Anpassung an Änderungen und Tests
    - Zugriffsschutz auf WWW-Dateien
    - Protokollierung am WWW-Server
    - Notfallvorsorge für den WWW-Server
    - Datensicherung

Auf die Konzeptionierung des WWW-Servers soll hier nicht näher eingegangen werden. Da der Server bereits seit einiger Zeit in Betrieb ist, ist dieser Punkt bereits zu einem früheren Zeitpunkt durchgeführt worden. Auf die Implementierung des Servers, speziell die Kommunikation, wird im nächsten Abschnitt näher eingegangen.

Aktiver Inhalt wird angeboten und muß vor der Freigabe auf Sicherheitslücken überprüft werden. Dazu zählen vom Browser des Benutzers gelieferte Eingaben, die nicht auf Plausibilität überprüft werden.

Der Betrieb des Servers wird dahingehend sicher gestaltet, daß die Logdateien aufgehoben und deren Inhalte auf bestimmte auffällige Muster überprüft werden, der Datenbestand täglich gesichert wird und die Erreichbarkeit laufend kontrolliert wird. Zudem wird der Zugriff auf die Daten so weit wie möglich eingeschränkt. Darauf wird gleich noch näher eingegangen.

4.4.1 Kommunikation

Die Kommunikation zwischen Server und Client wird durch SSL (Secure Socket Layer) [HK95] gesichert. Dadurch wird die Authentizität des Servers sicher gestellt und der zwischen Server und Client übertragene Datenstrom verschlüsselt. Dadurch wird die

Vollständigkeit und Korrektheit der übertragenen Daten überprüft. SSL wird von allen modernen Browsern unterstützt und bietet sich daher zum Schutz der Kommunikation an.

4.4.2 Zugriffsschutz

Der Schutz der Daten vor unberechtigtem Zugriff wird durch Paßwörter gewährleistet. Das setzt voraus, daß die zugelassenen Benutzer sorgfältig mit den Paßwörtern umgehen müssen, sie also z.B. nicht weitergeben, sicher verwahren, regelmäßig wechseln und gut auswählen [Bun97].

4.4.3 Berechtigungen

Nicht jeder Benutzer soll die gleichen Rechte haben. Ein Kunde soll nur die für ihn bestimmten Informationen sehen. Aber ein Support-Mitarbeiter muß auf alle Daten Zugriff haben. Die Berechtigungen müssen also so flexibel sein, dass verschiedene Rollen oder Tätigkeiten damit abgebildet werden können, wobei auch zukünftige Erweiterungen der Funktionalität abgedeckt werden müssen. Kommt z.B. eine Funktion für Umfragen hinzu, so soll es möglich sein, die Berechtigungen dazu gezielt den entsprechenden Personen zuzuteilen.

Besitzt ein Benutzer für eine Funktion nicht die notwendige Berechtigung, dann soll diese auch nicht zur Auswahl stehen. Dadurch wird verhindert, daß der Benutzer versucht, die Funktion auszuführen und erst danach abgewiesen wird. Dieses Verhalten würde nur zur Unzufriedenheit beitragen und möglicherweise auch noch Supportanfragen provozieren. Die Berechtigung zu einer Funktion soll in Abhängigkeit von Feldinhalten möglich sein, z.B. eine Berechtigung zum Anzeigen von offenen PTS-Fällen mit hoher Priorität. Daraus ergibt sich, daß die Berechtigungen parametrisierbar sein müssen.

Beispiel einer Berechtigung names PTS_CASE.SHOW zum Anzeigen aller PTS-Fälle, die zur Installation mit der Nummer 5382 gehören, unabhängig von deren Fallnummer, Status und Priorität:

Berechtigung	PTS_CASE.SHOW	Supportfälle der Installation 5382 anzeigen
	Aktivität	Anzeigen
	Kundennummer	5382
	Fallnummer	„alle“
	Status	„alle“
	Priorität	„alle“

Kapitel 5

Sicherheitsmodelle

Nachdem in der Analyse festgestellt wurde, daß auf ein flexibles Berechtigungskonzept nicht verzichtet werden kann, sollen nun die Sicherheitsanforderungen in der Theorie näher betrachtet werden. Dazu wird das „Orange Book“ als Kriterienkatalog und das „Generalized Framework for Access Control“ als Model herangezogen.

In diesem Kapitel ist öfter die Rede von Objekten und Subjekten. Ein Subjekt entspricht hier einem Aktor, der eine Aktion an einem Objekt durchführen will. Das Subjekt ist daher immer der aktive Teil, der etwas durchführt, und das Objekt der passive Teil, mit dem etwas durchgeführt wird.

5.1 Ansatz der TCSEC („Orange Book“)

Bei der Entwicklung eines Berechtigungskonzepts gibt es nach den „Trusted Computer Security Evaluation Criteria“ des US-amerikanischen Verteidigungsministeriums [oD85] sechs grundlegende Anforderungen, die hier kurz erläutert werden sollen. Außer einer Sicherheitspolitik (Policy) ist deren Überwachung (Accountability) notwendig. Zudem muß es Komponenten geben, die die Policy und die Überwachung umsetzen (Assurance).

5.1.1 Sicherheitspolitik

SECURITY POLICY Es muß eine explizite und wohl-definierte Sicherheitspolitik geben.

Ausgehend von identifizierten Subjekten und Objekten muß es eine Menge von Regeln geben, die dazu verwendet werden, um zu entscheiden, ob einem Subjekt der Zugriff auf ein Objekt gewährt werden kann. Es muß eine vorgeschriebene Sicherheitspolitik geben, die durch Zugriffsregeln effektiv implementiert wird. Diese Zugriffsregeln umfassen Anforderungen wie z.B. Keine Person, der der notwendige Personalberechtigungsstatus fehlt, soll Zugriff auf klassifizierte Informationen erhalten. Zusätzlich werden diskrete Berechtigungen benötigt, um sicher zu gehen, daß nur bestimmte Benutzer oder Gruppen von Benutzern Zugriff auf Daten erhalten.

MARKING Berechtigungen müssen mit den Objekten gespeichert werden. Um den Zugriff auf Informationen gemäß den Regeln einer vorgeschriebenen Sicherheitspolitik zu kontrollieren, muß es möglich sein jedes Objekt mit einer Markierung zu versehen. Diese Markierung muß die Sensitivität des Objekts zuverlässig wiedergeben.

5.1.2 Identifizierung

IDENTIFICATION

Einzelne Subjekte müssen identifiziert werden können. Jeder Zugriff auf die Informationen muß basierend auf der Identität des Subjekts und dessen Berechtigungen kontrolliert werden. Diese Identifizierungen und Autorisierungsinformationen müssen vom System sicher behandelt werden und mit jedem aktiven Element, das eine sicherheitsrelevante Aktion ausführt, assoziiert werden. Zur Identifikation der Benutzer wird bei Websystemen in der Regel ein Benutzername und ein Paßwort verlangt. Wurde der Benutzer erfolgreich identifiziert, wird dessen Identität während der Sitzung in seinem Browser gecacht.

ACCOUNTABILITY

Auditinformationen müssen selektierbar gespeichert und geschützt werden, damit Aktionen, die die Sicherheit betreffen, zurückverfolgt werden können. Ein vertrauenswürdiges System muß die Möglichkeit haben, sicherheitsrelevante Ereignisse in einem Auditlog zu speichern. Das Einschränken der Auditevents, die aufgezeichnet werden sollen, ist notwendig, um den Aufwand gering zu halten und eine effektive Analyse zu ermöglichen. Auditdaten müssen vor Veränderungen und unautorisierter Zerstörung geschützt sein, um Sicherheitsverletzungen erkennen und nachvollziehen zu können. Bei allen Websystemen gibt es naturgemäß einen Webserver, über den alle ein- und ausgehenden Daten laufen. Daher können bereits im Webserver Auditinformationen gesammelt werden. Die Selektion geschieht hierbei durch Anpassung der Konfiguration. Darüberhinaus besteht die Möglichkeit in der Webanwendung andwendungsspezifische Informationen zu speichern.

5.1.3 Verläßlichkeit

ASSURANCE

Das System muß Mechanismen besitzen, die die Einhaltung der vier bisher genannten Anforderungen gewährleisten. Um die Einhaltung der vier Anforderungen Policy, Marking,

Identification und Accountability zu gewährleisten, muß es entsprechende Komponenten im System geben. Diese Komponenten sind typischerweise im System eingebettet und sind robust gebaut.

CONTINUOUS PROTECTION

Die vertrauenswürdigen Komponenten, die die Anforderungen durchsetzen, müssen fortwährend gegen Veränderungen und Angriffe geschützt werden. Kein System kann als wirklich sicher gelten, wenn die Hardware und Software, die die Sicherheitspolitik durchsetzt, selbst unautorisierter Modifikation ausgesetzt sind. Der Betreiber eines Websystem muß dafür sorgen, daß nur autorisierte Personen Zugriff zur Hardware haben und die Software nicht von außen verändert werden kann.

5.2 Discretionary Access Control (DAC)

Diskrete Zugriffskontrolle wird bereits in den TCSEC definiert als eine Methode der Zugriffskontrolle auf Basis von Benutzeridentitäten und Gruppen, denen Objekte zugeordnet werden. Da der Besitzer eines Objektes den Zugriff für alle andere Benutzer verwaltet, ist die Durchsetzung der jeweiligen Sicherheitspolitik vollständig vom Objekt-Besitzer abhängig.

Die diskrete Zugriffskontrolle hat einige prinzipbedingte Nachteile in der Sicherheit. Das Hauptproblem ergibt sich aus dem Besitzer-Prinzip, das jedem Benutzer die volle Verfügungsgewalt über die von ihm erstellten oder ihm anvertrauten Objekte gibt. Selbst unter Annahme der vollen Vertrauenswürdigkeit aller Benutzer im Rahmen der jeweils verwalteten Objekte kann durch Bedienungsfehler, fehlerhafte oder bösartige Programme, z. B. ein Trojanisches Pferd, enormer Schaden, wie die Vernichtung oder Verfälschung wichtiger Daten, angerichtet werden.

Die Sonderrolle des Verwalter-Zugangs ohne Zugriffsbeschränkung, der viele wichtige Aufgaben als einziger wahrnehmen darf und deswegen oft verwendet werden muß, erhöht das beschriebene Risiko noch einmal beträchtlich.

5.3 Mandatory Access Control (MAC)

Als mandatorisch wird ein Sicherheitsmodell bezeichnet, in dem eine Sicherheitspolitik ohne Einfluß der einzelnen Benutzer verbindlich durchgesetzt wird. Alle Subjekte und Objekte bekommen ihre Zugriffsattribute vom System oder einer zentralen Instanz zugewiesen, die durch spezielle Benutzerkonten repräsentiert wird.

5.4 Allgemeines Modell

Das Generalized Framework for Access Control wurde zum ersten Mal 1990 auf der 13. Nationalen Computer-Sicherheits-Konferenz vorgestellt [Ott97]. Es wurde entwickelt wegen der großen Zahl von unflexiblen Sicherheitsmodellen und Implementationen, die zwar viele, aber bei weitem nicht alle Anforderungen erfüllen konnten. Außerdem stiegen mit zunehmender Komplexität der Betriebssysteme der Aufwand, Änderungen der Sicherheitspolitik in die Systeme hineinzuprogrammieren, und die Wahrscheinlichkeit von sicherheitsrelevanen Fehlern.

Benötigt wurde deshalb eine Möglichkeit, Sicherheitsmechanismen mit wenig Aufwand flexibel an die jeweiligen Bedürfnisse anpassen zu können. Anstelle von Kodierung sollte Konfiguration die Anpassung ermöglichen. Durch gezielte Kombination von für Teilbereiche anerkannt sicheren Modellen war sogar ein Zuwachs an Sicherheit erreichbar.

Durch sinnvolle Gliederung sollten außerdem mögliche Fehlerquellen besser lokalisiert und vermieden werden können.

5.4.1 Komponenten

Das Generalized Framework for Access Control teilt Zugriffskontrolle allgemein in die vier Komponenten Zugriffskontrollinformation, Kontext, Autoritäten und Regeln. Zugriffskontrolle ist dabei die Anwendung der Regeln auf Basis der von den Autoritäten festgelegten Zugriffskontrollinformationen und des jeweiligen Kontextes.

Die Zugriffskontrolle ist bei jedem Zugriff eines Subjekts auf ein Objekt, Zugriffskontrollinformationen oder den Kontext durchzuführen.

Zugriffskontrollinformationen (Access Control Information, ACI)

Zugriffskontrollinformationen sind stets an Subjekte oder Objekte gebunden. Sie umfassen alle diejenigen ihrer Daten, die von den Regeln zur Entscheidungsfindung verwendet werden. Subjekt-spezifische ACI beinhalten z.B. Identifizierungs- und Authentisierungsdaten, Sicherheitsklassifikation, Aufgaben und Rollen im System und vieles mehr. Beispiele für Objekt-spezifische ACI sind Identifizierung, Sicherheitslevel, Erzeuger, zugelassene Verarbeitungs-Programme und individuelle Zugriffskontrollisten (Access Control Lists, ACL), die eine subjektbezogene Zugriffsverwaltung erlauben.

Kontext (Access Control Context, ACC)

Als Kontext werden alle Informationen bezeichnet, die zwar von den Regeln verwendet werden, aber weder Subjekten noch Objekten zuzuordnen sind. Zum Kontext gehören

Systemdaten, wie Zeit, Status etc., aber auch Objekt-Definitionen, die nur im Rahmen des Sicherheitssystems benötigt werden, wie Benutzergruppen, Aufgaben oder Rollen.

Regeln (Access Control Rules, ACR)

Die Regeln definieren die Politik der Zugriffskontrolle. Im Gegensatz zur gängigen Verwendung des Begriffs umfassen die Regeln im GFAC nicht die Systemverwaltungsfunktionen des vertrauenswürdigen Systemkerns (Trusted Computing Base, TCB), sondern lediglich die durchzusetzenden Entscheidungsvorschriften. Daraus ergibt sich eine Zweiteilung der Systemfunktionen in einen politikabhängigen Bereich, der Zugriffentscheidungen anhand der Regeln durchführt, und einen politikunabhängigen Bereich, der für die Verwaltung des Systems zuständig ist. Diese klare Trennung der Aufgaben ermöglicht, durch einfache Regelanpassung die Sicherheitspolitik zu ändern, ohne den politikunabhängigen Bereich zu modifizieren. Der Zugriff eines Subjektes auf ein Objekt im Modus M wird nur dann gestattet, wenn die Zugriffskontrollinformationen von Subjekt und Objekt sowie der aktuelle Kontext die Regeln erfüllen. Mehrere zusammengehörige Regeln können zu einem Regelsatz zusammengefaßt werden. Beispiele für einfache Regeln sind Zeitbeschränkungen oder das simple-security-property des Bell-La Padula-Modells, Beispiele für Regelsätze sind Umsetzungen des Bell-La Padula-Modells oder des Clark-Wilson-Integritätsmodells.

Bei Erstellung und Anwendung der Regeln sind folgende Punkte zu beachten:

- Kombination von Regeln: Sobald mehrere Regeln anwendbar sind, muß eine eindeutige logische Verknüpfung oder eine Präzedenz dieser Regeln definiert sein.
- Erzeugung von Objekten und Vererbung: Für die Erzeugung neuer Subjekte und Objekte sind Regeln notwendig, die die zu vergebenden ACI durch Standardwerte oder Vererbung festlegen.
- Kombination und Konfiguration von Regelsätzen: Obwohl beliebige Regeln durch die Autoritäten definiert werden können, sind in der Praxis zertifizierte Regelsätze nur unter Verlust der Zertifizierung veränderbar. Im GFAC können solche Regelsätze durch bedarfsabhängige Kombination und Konfiguration ohne Sicherheitsbeschränkung an individuelle Anforderungen angepaßt werden.

Autoritäten (Access Control Authorities, ACA)

Autoritäten sind spezielle Regeln, die gezielt den Zugriff auf alle vier Komponenten des GFAC festlegen. Diese Regeln bilden den zentralen Punkt für die Wirksamkeit eines Zugriffskontrollsystems. Durch die Autorisierung von Autoritäten ergeben sich mehrstufige Autoritätsbäume, die als Blätter entweder extern oder mit politikabhängigen Verfahren festgelegte oberste Instanzen der Autorität enthalten.

5.5 Ansatz von L. LaPadula

LaPadula spezifizierte ansatzweise die Umsetzung des Generalized Framework for Access Control in einem Unix-System [LaP95]. Dazu verwendete er die Sicherheitsmodelle klassische mandatorische Zugriffskontrolle (Bell-LaPadula) nach System V/MLS (MAC), Clark-Wilson-Integritäts-Modell (CWI), funktionale Kontrolle (FC) und Änderung der Sicherheitsinformationen (SIM) in teilweise vereinfachter Form.

5.5.1 Komponenten

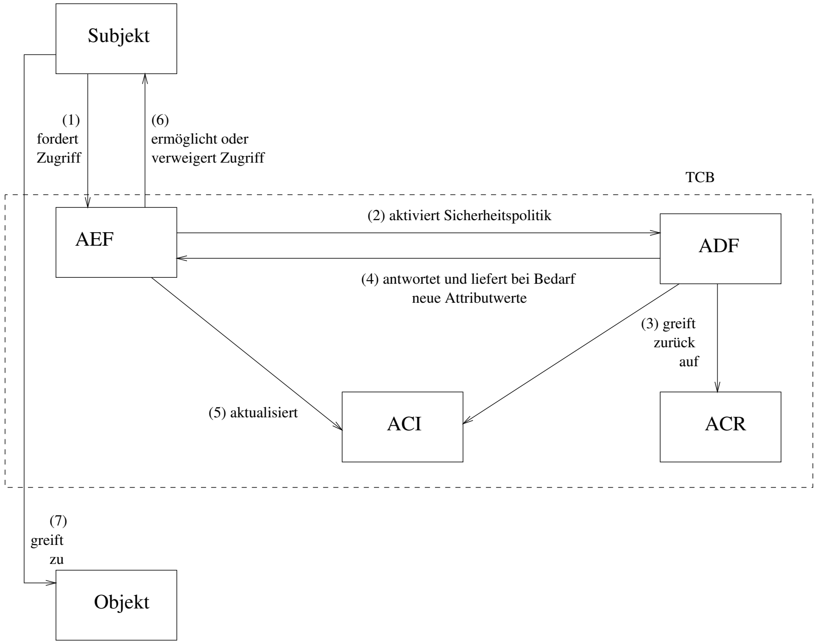

Abbildung 5.1: Komponenten des Generalized Framework for Access Control nach La Padula

La Padula bezeichnet den politikabhängigen, die Regelsätze enthaltenden Bereich des vertrauenswürdigen Systemkerns als Entscheidungskomponente (Access Control Decision Facility, ADF). Den Teil des politikunabhängigen Bereichs, der bei jeder sicherheitsrelevanten Systemanfrage die Entscheidung veranlaßt und durchsetzt, nennt er entsprechend die Durchsetzungskomponente (Access Control Enforcement Facility, AEF).

Das logische Zusammenspiel der funktionalen Systemkomponenten findet sich in Abbildung 5.1. Diese beginnt mit einem Systemaufruf an die Trusted Computing Base (TCB), den Systemkern. Dort wird innerhalb des Durchsetzungscodes der zugehörigen Bearbeitungsroutine eine Anforderung an die Entscheidungskomponente generiert. Diese Anfrage durchläuft alle aktiven Regelsätze, die Gesamtentscheidung wird zusammengesetzt und mit neuen Attributwerten an die Durchsetzung zurückgegeben. Je nach Ergebnis veranlaßt diese entweder die entsprechende Attributsetzung und ermöglicht den Zugriff oder liefert eine Fehlermeldung an den aufrufenden Prozeß.

5.6 Role Based Access Control (RBAC)

Das RBAC-Modell wurde am National Institute of Standards and Technology untersucht.

RBAC versucht einen Kompromiß zwischen MAC und DAC zu bilden. Nach [FK92] ist MAC für militärische Anwendungen geeignet, während angenommen wird, daß DAC für die Industrie ausreichend sei. In [FK92] wird gezeigt, daß DAC als die hauptsächliche Zugriffskontrollmethode in vielen komerziellen Anwendungen ungeeignet ist.

Mit RBAC werden Zugriffsentscheidungen anhand der Rollen, die ein Subjekt in einer Organisation hat, getroffen. Die Benutzer nehmen zugewiesene Rollen ein, z.B. Doktor, Krankenschwester oder Manager. Die Rollen sollten nahe an der Organisationsstruktur liegen.

Die Zugriffsrechte werden in den Rollen gruppiert, und die Verwendung von Ressourcen ist auf die Personen eingeschränkt, die für die entsprechenden Rolle autorisiert sind.

Dadurch, daß Rollen entsprechend der Organisation definiert werden und Personen diesen Rollen zugeteilt werden, wird gewährleistet, daß ein Benutzer immer nur die notwendigen Berechtigungen besitzt.

Rollen können neben Zugriffsrechten auch andere Rollen enthalten. Dadurch wird die Vererbung von Zugriffsrechten ermöglicht.

Die Beziehungen von Rollen untereinander sowie zwischen Rollen und Personen können Kardinalitäten besitzen. Dadurch kann z.B. verhindert werden, daß mehr als eine Person die Rolle des Bankfilialleiters einnimmt.

Zudem können zwischen den Rollen Abhängigkeiten bestehen, so daß eine Rolle eine Operation erlaubt, eine andere Rolle aber diese Operation explizit verbietet. Damit eine Person nicht diese beiden Rollen zugewiesen bekommt, können solche Abhängigkeiten definiert werden.

5.7 Bewertung

Das GFAC ist ein allgemeines Modell und ist deshalb ein guter Ansatzpunkt für ein neu zu entwickelndes Berechtigungskonzept.

La Padula spezifizierte eine mögliche Umsetzung des GFAC in einem Unixsystem und zeigt damit die Praxistauglichkeit des GFAC.

RBAC scheint einige neue Aspekte in die Zugriffskontrolle zu bringen, aber sie betreffen alle die Phase der Administration, also der Verwaltung der Zugriffsrechte [Bar97]. Deshalb werden diese Punkte zwar im Design des Berechtigungskonzepts einfließen, aber das RBAC-Modell wird sonst nicht weiter verfolgt.

Kapitel 6

Design

Nachdem im vorletzten Kapitel die Anforderungen analysiert und zuletzt bekannte Sicherheitsmodelle betrachtet wurden, werden nun die Komponenten des neuen Systems bestimmt und entwickelt. Zuerst werden einige Rahmenbedingungen festgelegt. Im Anschluß daran wird Bottom-Up vorgegangen. Das Berechtigungskonzept ist eine zentrale Komponente und wird deshalb als erste behandelt. Danach soll noch auf die Schnittstellen zum PTS und CID sowie die Integration in das ESC eingegangen werden.

6.1 Rahmenbedingungen

Das neue System ist eine neue Komponente eines bereits bestehenden Gesamtsystems. Daher muß es sich an bestimmte Bedingungen halten, die durch das bestehende System vorgegeben sind. Allerdings gibt es einen Spielraum, in dem Verbesserungen vorgenommen werden können. Dazu zählt beispielsweise der erste Punkt, die Objektorientierte Modellierung.

6.1.1 Objektorientiertes Design

Die bestehenden Systeme PTS und CID sind trotz Ihres Umfangs nicht objektorientiert entworfen worden. Um die Wiederverwendbarkeit der zu entwickelnden Komponenten zu fördern und das System besser strukturieren zu können, wird objektorientiertes Design verwendet. Das bedeutet, daß der erste Schritt des Entwurfs darin bestehen wird, aus dem Use-Case-Modell relevante Objektklassen zu identifizieren.

6.1.2 Design Pattern

„Das Entwerfen von objektorientierter Software ist schwer, und das Entwerfen von wiederverwendbarer objekteorientierter Software ist noch viel schwerer.“ [G+95]

Der schwierigste Punkt beim objektorientierten Entwurf ist bereits das Zerlegen des Systems in passende Objekte. Dabei spielen viele Faktoren eine Rolle. Bei einer zu geringen Granularität ist die Wartbarkeit nicht gegeben, bei einer zu großen wird die Komplexität sehr hoch.

Um die Wartbarkeit der in dieser Diplomarbeit entworfenen Software zu maximieren, wird beim Entwurf des Klassenmodells auf Design Pattern zurückgegriffen. Sie helfen, wiederkehrende ähnliche (d.h. eine gemeinsame Struktur habende) Probleme anhand von Lösungsmustern zu lösen.

Ein Design Pattern besteht aus einem Namen, einer Problembeschreibung, einer Lösungsbeschreibung und den Konsequenzen (Kosten/Nutzen).

Es gibt mehrere Design Pattern Kataloge, z.B. den, der nach [G+95] folgende Kategorien enthält.

- `Creational Pattern` umfassen Muster, die das Erzeugen und Verwalten von Objekten abdecken.
- `Structural Pattern` umfassen Muster zum Gliedern des Objektmodells eines Systems.
- `Behavioral Pattern` umfassen Muster, die des Verhalten von Objekten behandeln.

6.1.3 Perl

Als Programmiersprache ist Perl (http://www.perl.org) praktisch vorgegeben. Die bestehenden Systeme ESC, PTS und CID sind ausschließlich in Perl geschrieben. Perl hat den Vorteil, daß Änderungen schnell umgesetzt werden können und es im Comprehensive Perl Archive Network (CPAN, http://www.cpan.org) eine Vielzahl von Modulen gibt, mit denen die Funktionalität von Perl erweitert werden kann.

Hier noch einige Features, die Perl bietet [Mon]:

- Perl besitzt einige gute Features anderer Sprachen wie C, awk, sed, sh und BASIC.
- Perl unterstützt Datenbanken wie Oracle, Sybase, Postgres und viele andere durch die abstrakte Datenbankschnittstelle DBI
- Perl unterstützt Unicode
- Perl unterstützt prozedurales und objektorientiertes Programmieren

- Perl kann durch XS oder SWIG externe C/C++ Bibliotheken verwenden
- Perl ist wegen der einfachen Textmanipulation und der schnellen Entwicklung eine populäre Programmiersprache für das Web.

Die Kernbestandteile wie Datenhaltung und Anwendungslogik von CID und PTS sind ausschließlich in der Sprache Perl implementiert. Aus Gründen des Aufwands und der Konsistenz sollen diese Funktionen wieder verwendet werden. Deshalb ist die Entscheidung der Programmiersprache für das neue System auf Perl gefallen. Die Integration des nicht objektorientierten Codes von PTS bzw. CID in das neue objektorientierte System stellt kein Problem dar. Perl kennt die Begriffe Objekte, Methoden und Attribute erst seit kurzer Zeit und war deshalb angehalten, diese Konzepte möglichst reibungslos einzuführen, um Eingriffe an bestehendem Code zu minimieren.

6.2 Berechtigungskonzept

Der Zugriff auf PTS und CID soll gegenüber dem heutigen Stand einem größeren Benutzerkreis ermöglicht werden. Dabei gibt es recht unterschiedliche Anforderungen an die zur Verfügung gestellten Informationen und Funktionen. Ein Kunde wird sicher nur Support-Fälle sehen und bearbeiten dürfen, die seine Installationen betreffen. Ein Partner hingegen soll alle Installationen einsehen dürfen, die er bei Kunden, die er betreut, vorgenommen hat. Ein Support-Mitarbeiter wiederum soll natürlich alle Fälle einsehen dürfen. Diese Anforderungen können sich zukünftig aber ändern. Es können z.B. neue Akteure hinzukommen oder die Berechtigungen der Akteure erweitert werden, weil sich die Anforderungen ändern. Das neue System soll daher möglichst so flexibel sein, daß neue Anforderungen an die Berechtigungen ausschließlich durch Anpassung der Konfiguration realisiert werden können.

Es gibt mindestens zwei Komponenten bei einem Konzept für die Zugriffskontrolle. Der Benutzer muß eine Identität im System erhalten, was als Authentifizierung bezeichnet wird. Außerdem muß es möglich sein, die Identität eines Benutzers mit Rechten für die angebotenen Dienste in Verbindung zu bringen. [GM98]

Wie in 2.4 dargestellt, wurden bereits Kennungen für Mitarbeiter, Kunden und Partner vergeben. Da diese auf der Basis von Einzelpersonen (keine Firmen- oder Sammelaccounts) vergeben wurden, eignen sie sich als Grundlage für das im folgenden dargestellte Berechtigungskonzept. Auch aus Aufwandsgründen wird daher daran festgehalten.

Das hier entworfene Berechtigungskonzept lehnt sich an das im SAP R/3 gefundene Konzept (siehe Kap. 3.1) an. Es basiert auf Rollen, die Berechtigungen und andere Rollen enthalten können. Durch die starke Unterteilung ist es sehr flexibel für spätere Erweiterungen und vor allem auch für die Berechtigungsverwaltung.

Hier wird nun zuerst das statische Objektmodell für das Berechtigungskonzept entwickelt. Dazu gehört, die Klassen sowie deren Assoziationen und Attribute zu ermitteln (siehe Abb. 6.1).

Ausgehend vom Benutzer soll diesem eine oder mehrere Rollen zugeteilt werden können, die festlegen, was und in welchem Umfang er darf. Diese Rollen lassen sich in einzelne Tätigkeiten (Berechtigungen) zerlegen. Eine Berechtigung bestimmt was in welchem Umfang erlaubt ist.

Um die Anzahl der Berechtigungen und Berechtigungsobjekte strukturieren zu können, gibt es Berechtigungsklassen.

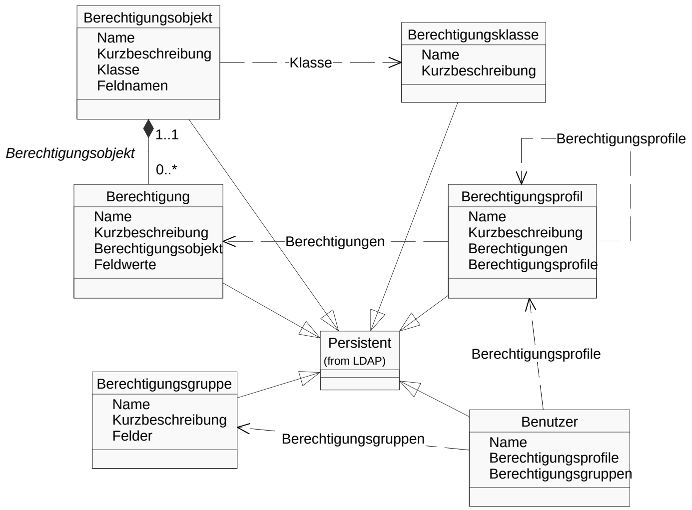

Abbildung 6.1: Objektmodell der Berechtigungen

Nun soll ausgehend vom Anwender Top-Down die Struktur des Berechtigungskonzepts erläutert werden.

6.2.1 Benutzerstammsatz

Der Benutzer ist nach dem GFAC das Subjekt, das auf ein Objekt im System zugreift. Um den Anwender authentifizieren zu können, muß dem System ein Name, ein Identifizierungsmerkmal bekannt sein. Im ESC wird zur Authentifizierung die Kombination aus dem

Namen und einem Paßwort verwendet. Zur Autorisierung des Anwenders werden ihm Profile zugeordnet. Der Benutzerstammsatz dient zur Zusammenfassung von Benutzergruppen und Berechtigungsprofilen, die einem Benutzer zugeordnet sind. Hier können zukünftig auch weitere Eigenschaften abgelegt werden, die spezifisch für einen Benutzer sind. Dazu könnten z.B. Standardwerte für Eingabe- und Auswahlfelder zählen, um das Ausfüllen von Formularen zu erleichtern.

Attributname	Attributwert	Beschreibung
Benutzerstammsatz	c122453	Kennung von Herrn Maier
Benutzergruppe	Kunde	Alle Installationen des Kunden (siehe Abb. 6.6)
Berechtigungsprofile	Kunde	Berechtigungen zum Verwalten von CID und PTS (siehe Abb. 6.3)

Abbildung 6.2: Beispiel eines Benutzerstammsatzes für Herrn Maier von der Beispiel AG

6.2.2 Berechtigungsprofil

Der Benutzerstammsatz alleine enthält noch keine Berechtigungen. Diese werden im Berechtigungsprofil gruppiert. Dadurch können mit Berechtigungsprofilen Rollen definiert werden. Es enthält alle Berechtigungen (Ausprägungen von Berechtigungsobjekten), die zur Ausübung einer Tätigkeit notwendig sind. So enthält z.B. das Berechtigungsprofil in Abb. 6.3 alle Berechtigungen, die für das anlegen, anzeigen und ändern von PTS-Fällen benötigt werden, aber keine Berechtigungen für die Administration, die z.B. mit ADMIN beginnen könnten.

Attributname	Attributwert	Beschreibung
Berechtigungsprofil	Kunde	Standardprofil für Kunden
Berechtigungen	PTS_CASE.EDIT	Anzeigen, Ändern, Anlegen von Supportfällen

Abbildung 6.3: Beispiel eines Berechtigungsprofils für Kunden

Ein oder mehrere Berechtigungsprofil werden einem Benutzerstammsatz zugewiesen. Dadurch erhält der Benutzer die in diesen Berechtigungsprofilen enthaltenen Berechtigungen.

6.2.3 Berechtigung

Berechtigungen erlauben den Zugriff auf ein Objekt in Abhängigkeit von dessen Attributen. Welche Attribute zur Entscheidung herangezogen werden können, wird im zugehörigen Berechtigungsobjekt definiert.

Als Wert der Attribute können Einzelwerte, Listen von Werten `A,B`, Wertebereiche `A-B` und Kombinationen `A,B-D` daraus angegeben werden. Außerdem gibt es als Wildcard den Stern *. Er steht für alle zulässigen Werte. Folgendes Beispiel (Abb. 6.4) zeigt eine Berechtigung bzw. eine Ausprägung des obigen Berechtigungsobjekts PTS_CASE.

	Attributname	Attributwert
Berechtigung	PTS_CASE.EDIT_INST5382	
Berechtigungsfelder	ACTVT	Anzeigen, Ändern, Anlegen
	INSTNO	5382
	CASENO	*
	STATUS	*
	PRIORITY	*

Abbildung 6.4: Beispiel einer Berechtigung

Diese Berechtigung erlaubt das Anzeigen und Ändern von allen Fällen, die der Installation 5382 zugeordnet sind, sowie das Anlegen von neuen Fällen zu dieser Installation.

6.2.4 Berechtigungsobjekt

Ein Berechtigungsobjekt besteht aus ein oder mehreren Berechtigungsfeldern (Attributen), die in Kombination überprüft werden können. Ein Berechtigungsobjekt ist per Definition einem Objekt nach GFAC zugeordnet, weil es die Zugriffsschutzmöglichkeiten eines Objekts definiert. Durch die Instanzierung mittels einer Berechtigung und dessen Zuordnung zu einem Benutzer durch ein Berechtigungsprofil wird die Verbindung zwischen Objekt und Subjekt erreicht.

Es sind Prüfungen der Art „Darf der Benutzer den Supportfall 5689 mit der Priorität High ansehen?“ möglich. Berechtigungsobjekte werden vom Programmierer definiert und dokumentiert. Die Definition geschieht durch Eintragen des Berechtigungsobjekts und dessen Attribute samt ihrer Wertebereiche in eine Datenbank, die im Kapitel 7 konkretisiert wird.

In Abb. 6.5 ist das Berechtigungsobjekt `PTS_CASE` angegeben. Es ist definiert aus den Feldern ACTVT, INSTNO, CASENO, STATUS und PRIORITY.

	Attributname	Attributwert
Berechtigungsobjekt	Name	PTS_CASE
Berechtigungsfelder	ACTVT	Anlegen, Ändern, Anzeigen
	INSTNO	Zahl
	CASENO	Zahl
	STATUS	open-*, closed-*
	PRIORITY	low, medium, high

Abbildung 6.5: Beispiel eines Berechtigungsobjekts

6.2.5 Berechtigungsfeld

Ein Berechtigungsfeld ist die kleinste Einheit, gegen die geprüft werden soll. Der Programmierer legt fest, welches Feld zur Überprüfung einer Berechtigung herangezogen werden soll, indem er ein Berechtigungsobjekt definiert, das das Feld enthält.

Ein Beispiel für ein Berechtigungsfeld ist die Nummer eines Supportfalls. Dadurch ließe sich der Zugriff auf bestimmte Fälle anhand ihrer Nummer einschränken.

Im Unterschied zum R/3 wird hier keine eigene Klasse für Berechtigungsfelder verwendet. Der Wertebereich der Berechtigungsfelder ist wird im jeweiligen Berechtigungsobjekt definiert, in dem das Berechtigungsfeld vorkommt.

6.2.6 Benutzergruppe

Benutzergruppen weichen die bisherige Strukturierung etwas auf. Sie können die durch Benutzerprofile erlangten Rechte eines Benutzers wieder einschränken.

Gibt es z.B. einen Benutzer, der nur auf bestimmte Installationen Zugriff haben soll, wäre es ohne Benutzergruppen notwendig, für ihn ein neues Benutzerprofil mit neuen Berechtigungen zu erstellen. Das neue Benutzerprofil würde sich von den bestehenden nur durch einen anderen Wert im Feld Installationsnummer unterscheiden.

Durch die Benutzergruppe ist es nun möglich, bestehende Benutzerprofile wieder zu verwenden. Dazu wird der Geltungsbereich eines bestehenden Benutzerprofils unterschiedlich eingeschränkt. Eine besondere Rolle spielt hierbei die Installationsnummer. Durch sie können Installationen und Supportfälle einem Kunden zugewiesen werden. Deshalb wird eine Benutzergruppe im Moment als einziges Feld die Installationsnummer besitzen.

Um ein Benutzerprofil wiederzuverwenden und seinen Geltungsbereich einzuschränken wird als Wert für die Installationsnummer `*` eingetragen und der Bereich der erlaubten Installationen durch die Benutzergruppe eingeschränkt. Die Beispielgruppe (Abb. 6.6) enthält den Platzhalter `cid` für das Feld `INSTNO`. Dieser Platzhalter steht für alle Installationsnummern denen der jeweilige Benutzer im CID als Ansprechpartner zugeordnet ist. Dies hat den zusätzlichen Vorteil, daß die Zuständigkeit nur an einer Stelle, hier dem CID, gepflegt werden muß. Deshalb ist für die Administration der PTS-Benutzer durch den Kunden eine Editiermöglichkeit seiner CID-Daten notwendig.

Benutzergruppe	Kunde	Standardgruppe für Kunden
INSTNO	cid	Alle Installationen des Kunden

Abbildung 6.6: Beispiel einer Benutzergruppe für Kunden

Ansonsten können für die Installationsnummer auch Einzelwerte (`537`), eine Liste von

Einzelwerten (`348,537,590`), Wertebereiche (`537-545`) und Kombinationen aus diesen (`236,537,634-639`) angegeben werden.

6.3 Schnittstelle zum PTS und CID

Das PTS und CID ist nicht objektorientiert entwickelt worden. Das neue System soll aber nicht die Funktionalität der beiden bestehenden Systeme erneut implementieren. Deshalb wird eine Schnittstelle zwischen den beiden Modellen notwendig.

Im folgenden wird der Kommunikationsablauf zwischen den Objekten und dem PTS bzw. CID erläutert.

6.3.1 PTS

Anlegen eines neuen Falls

Beim Anlegen eines neuen Falls prüft `CaseForm` die Berechtigung des angemeldeten Benutzers. Danach werden dem Benutzer Fragen zum neuen Fall gestellt. Dazu gehört unter anderem die betroffene Installation, eine Problembeschreibung. Die betroffene Installation wird aus einer Liste ausgewählt. Diese Liste generiert `CaseForm` aus der CID und den dem Benutzer zugewiesenen Benutzergruppen. Die Antworten werden von `CaseForm` entgegengenommen, verarbeitet und daraus ein neuer Fall generiert. `Case` schreibt seinen Zustand daraufhin in die Datei `.status` im Verzeichnis des neuen Falls, indem es auf Funktionen des PTS zurückgreift. Danach wird von `CaseForm` eine Bestätigungsemail, die die Fallnummer und die Beschreibung des Falls enthält, an den Benutzer geschickt.

Der Benutzer erhält zuletzt noch eine Bestätigung von `CaseForm` in seinem Browser.

Anzeigen von Fällen

Der Benutzer bestimmt die Kriterien der anzuzeigenden Supportfälle. Dazu gehört Installation und Status. `CaseForm` erhält diese Informationen und erstellt daraus eine Liste von bestehenden Fällen, die den Kriterien entsprechen. Dabei wird für jeden Fall die Berechtigung des Benutzers überprüft. Der Zugriff auf die Supportfälle geschieht über Funktionen des PTS.

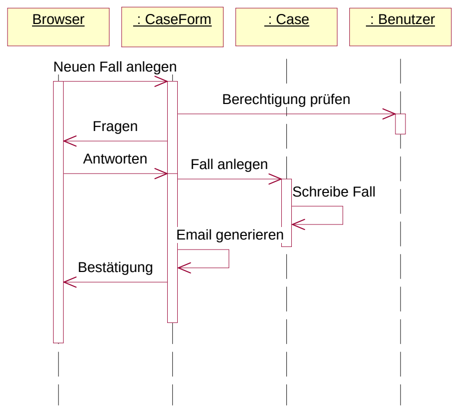

Abbildung 6.7: Anlegen von PTS-Fällen

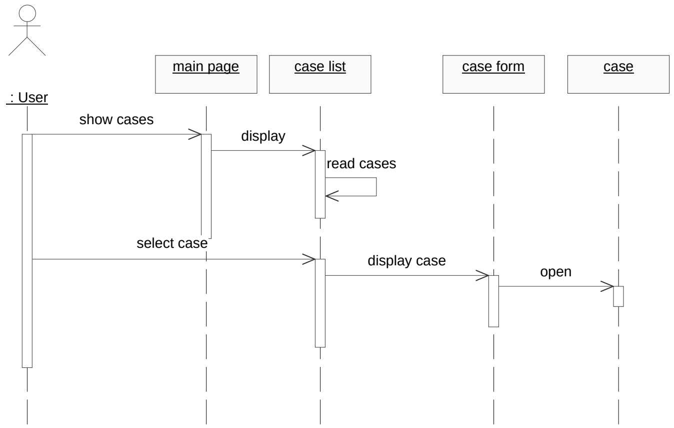

Abbildung 6.8: Anzeigen von PTS-Fällen

6.4 Integration in das ESC

Wie im Kapitel „bestehende Systeme“ bereits dargestellt, ist das ESC wie das PTS und die CID ausschließlich in Perl programmiert und modular entworfen. Eine Integration des neuen Systems in das ESC ist deshalb ohne großen Aufwand möglich.

Ein Zugriff auf Daten des ESC ist nicht notwendig, weil nur PTS-, CID- und MKIS-Daten notwendig sind. Es ist aber erforderlich, das Navigationsmenü (`esc-treenci.cgi`) um einen Link zum neuen System zu erweitern.

Die URL des Links ist im Hash `%esc_action` unter dem Key `casemgr` hinterlegt. Dieser Hash ist in der zentralen Konfigurationsdatei `esc-cfg.pl` definiert.

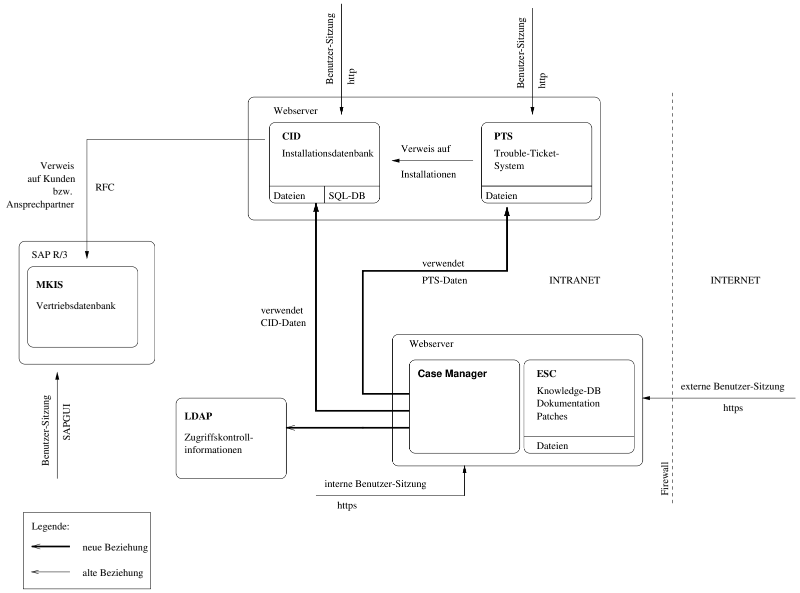

Abbildung 6.9: Zusammenspiel von MKIS, CID, PTS und ESC nach der Integration

Kapitel 7

Implementierung

In diesem Kapitel wird auf einige Implementierungsaspekte des neuen Systems eingegangen. Es geht darum, wie die im vorhergehenden Kapitel entworfenen Bausteine in die Praxis umgesetzt worden sind. Jedoch werden nicht alle Klassen komplett behandelt, sondern nur interessante Ausschnitte, die im engeren Zusammenhang mit dem Thema dieser Arbeit stehen. So werden die Datenhaltung und die Entscheidungskomponente des Berechtigungskonzepts im Mittelpunkt stehen. Zudem wird kurz der Zugriff auf die PTS-Daten skizziert.

7.1 Zugriffskontrolle

LaPadula hat die Komponenten des „Generalized Framework for Access Control“ in die Module Datenhaltung (ACI, ACC), Entscheidung (ADF, ACR) und Durchsetzung (AEF) gruppiert (siehe Abb. 7.1). Im folgenden wird detaillierter auf diese Module eingegangen.

7.1.1 Datenhaltung

Zur Speicherung der Berechtigungen, Berechtigungsobjekte, -profile und -gruppen, d.h. der Zugriffskontrollinformationen (Access Control Information, ACI) wird ein geeignetes Verfahren benötigt. Die einzelnen Objekte müssen dauerhaft gespeichert werden und schnell abgefragt werden können. Änderungen an den Objekten kommen nur dann vor, wenn sich an den Berechtigungen etwas ändert. Also z.B. wenn für eine neue Benutzergruppe ein eigenes Profil eingerichtet wird oder ein neuer Benutzer hinzukommt. Überwiegen werden die lesenden Zugriffe. Bei fast jedem Aufruf einer dynamischen Webseite des Systems werden Berechtigungen überprüft, was einen Zugriff auf die peristenten Daten erfordert.

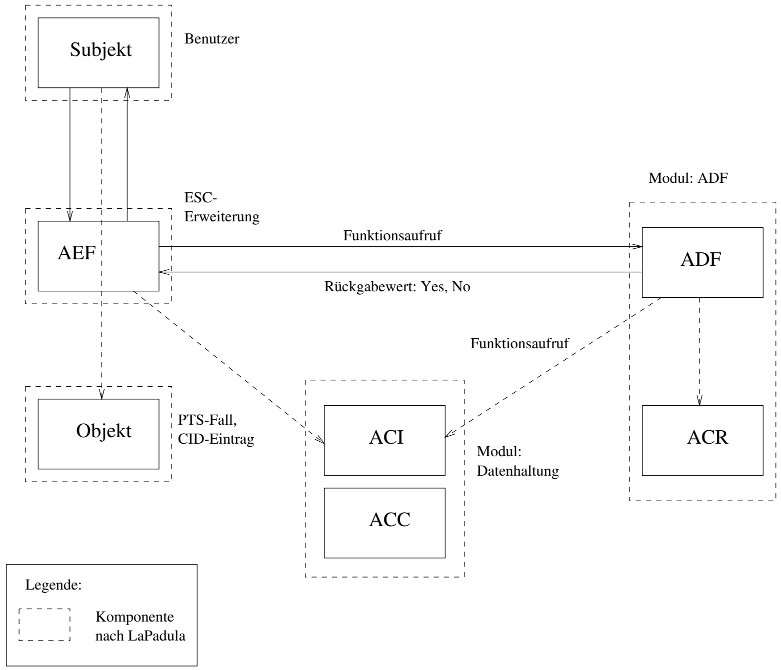

Abbildung 7.1: Zuordnung und Gruppierung der GFAC-Komponenten

Zur Diskussion bei der Speicherung stehen eine Datenbank, entweder relational oder objektorientiert, flache Dateien oder ein Verzeichnisdienst.

Flache Dateien bieten am wenigsten Komfort bei der Abfrage der Informationen, würden daher einen erheblichen Mehraufwand bei der Implementierung bedeuten und scheiden deswegen aus. Diese Methode fördert auch nicht die Modularisierung mittels einer Client-/Server-Architektur.

Datenbanken sind das andere Extrem und bieten eine Fülle an Unterstützung. Dazu gehören DDL (Data Definition Language) und DML (Data Manipulation Language), sowie Möglichkeiten der Indexierung zum schnelleren Auffinden von Informationen. Außerdem gewährleisten Datenbanken atomare Transaktionen. Diese Eigenschaft ist notwendig, wenn gleichzeitig zwei Prozesse schreibend auf die Daten zugreifen, was bei der hier behandelten Anwendung nicht der Fall ist. Die Administration wird zu jeder Zeit nur von einer Person gleichzeitig durchgeführt werden.

LDAP steht für Lightweight Directory Access Protocol [YHK95] und wurde eingeführt, um den Zugriff auf X.500-Verzeichnisse zu erleichtern, weil das X.500-Protokoll umfangreich und komplex ist. LDAP definiert aber ausschließelich das Zugriffsprotokoll und nicht den ganzen Verzeichnisdienst. Wenn von einem LDAP-Server gesprochen wird, handelt es sich um einen Verzeichnisdienst, der über LDAP erreichbar ist.

Es gibt vier große Unterschiede zwischen X.500 und LDAP, die LDAP eine große Verbreitung bescherten:

- LDAP läuft auf dem wesentlichen einfacheren Internetprotokoll TCP/IP.
- LDAP vereinfacht das X.500-Modell durch Verwendung einer search-Operation anstelle einer list- und einer read-Operation.
- LDAP verwendet ein einfacheres Encoding-Schema, das die meisten Einträge und Distinguished Names als Strings speichert [YHK95].
- LDAP verfolgt Referenzen auf andere Server selbständig. Dadurch wird der Client von solchen Tätigkeiten entlastet und wird kleiner. X.500 dagegen informiert den Client nur über die Referenz.

LDAP ist ein Internetstandard, der auch von Produkten großer Firmen wie Novell, Netscape, Microsoft und Sun unterstützt wird. Auch gibt es freie Open-Source LDAP-Server wie z.B. OpenLDAP (http://www.openldap.org). Durch das standardisierte LDAP-Protokoll ist es möglich, den LDAP-Server gegen einen anderen auszutauschen, ohne daß die Anwendung dadurch beeinflußt wird.

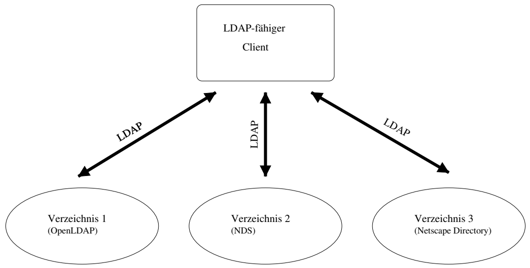

Abbildung 7.2: Zugriff auf unterschiedliche Verzeichnisse mit LDAP

Nun sollen die einzelnen Ansätze - Dateien, Datenbanken und Verzeichnisdienste - anhand von fünf Kriterien verglichen werden. Jedem Punkt wird eine Bewertung zwischen $--$ (sehr schlecht) und $++$ (sehr gut) gegeben. Als Kriterien wurden die Modularität, Komplexität des Clients, Skalierbarkeit, Anschaffungskosten und Betriebskosten gewählt.

Die Modularität zeigt, ob es möglich ist, die Speicherung von der Anwendung zu trennen. Bei Dateien ist das kaum möglich, außer es wird ein Netzwerkfilesystem wie z.B. NFS verwendet. Aber auch hierbei ist paralleler Zugriff nur bedingt möglich. Bei der Datenbank ist die Trennung von Client und Server zwar möglich, aber nur mit einem speziellen plattformabhängigen Datenbankclient oder einer Client-Bibliothek. Bei LDAP hingegen kann der Server ausgetauscht werden, ohne daß dies den Client in irgendeiner Form beeinflussen würde.

Die Komplexität des Clients sollte möglichst gering sein, um die Komplexität der Anwendung nicht zusätzlich durch die Speicherungsmethoden zu erhöhen. Bei Dateien hält sich die Komplexität in Grenzen, aber es müssen passende Methoden entwickelt werden, die bei einer Änderung der Datenstruktur angepaßt werden müssen. Zudem ist effizientes Durchsuchen von größeren Dateien nicht trivial. Bei einer Datenbank müssen die Anfragen an die Datenbank in der DDL/DML formuliert werden. Für Perl gibt es LDAP-Module, die es erlauben, sehr einfach persistente Objekte zu entwickeln. Persistente Attribute werden als solche markiert und werden durch einen Aufruf der save()-Methode im Verzeichnis gespeichert bzw. durch einen Aufruf der restore()-Methode wieder ausgelesen.

Bei der Skalierbarkeit schneiden die einfachen Dateien am schlechtesten ab, weil sie auf einem Filesystem liegen, daß nur durch schnellere bzw. größere Platten und schnellere Prozessoren skaliert werden kann. Bei einer Datenbank läßt sich aber schon durch geschickte Cachingalgorithmen und Indexstrukturen eine bessere Skalierbarkeit erreichen. LDAP erlaubt zusätzlich noch Referenzen, womit Teilbäume auf einen weiteren LDAP-Server ausgelagert werden können und somit eine Lastverteilung erreicht wird.

Zu den Anschaffungskosten zählt neben den Lizenzkosten auch der Zeitaufwand zum Installieren und Einrichten. Hier schneiden die Dateien am besten ab, weil ein Filesystem überall vorhanden ist. Datenbanken sind sehr teuer, wenn man nicht auf Freeware zurückgreifen will oder kann. Außerdem ist ein relativ hoher Aufwand notwendig, bis sie einsatzbereit sind. Es müssen Partitionen für Daten und Logs eingerichtet werden und Tablespaces angelegt werden. Dieser Mehraufwand ist sicher gerechtfertigt, wenn man alle Eigenschaften einer Datenbank ausnutzen wird. Bei LDAP ist mit der Installation der Software in der Regel alles erledigt. Es müssen keine Datenstrukturen angelegt werden, wenn man mit den vordefinierten Objektklassen auskommt.

Zu den Betriebskosten zählen die laufenden Kosten, wie z.B. Datensicherung und Änderung der Konfiguration. Bei Dateien ist die Änderung der Konfiguration sehr einfach, es müssen nur neue Dateien angelegt oder bestehende geändert werden. Bei einer Datenbank ist das nur durch eine Anfrage in der DDL oder DML möglich. Teilweise sind nachträgliche Änderungen nur sehr schwer möglich. LDAP kennt keine festen Datenstrukturen und ist dadurch sehr flexibel.

	Dateien	rel. SQL-Datenbank	LDAP-Server
Modularität	−−	+	++
Clientkomplexität	+	+	++
Skalierbarkeit	−−	+	++
Anschaffungskosten	++	−−	+
Betriebskosten	+	−−	+

Man sieht, für die hier geforderten Eigenschaften schneidet der Verzeichnisdienst LDAP am Besten ab. Die Wahl zur Speicherung der Konfiguration ist daher auf einen Verzeichnisdienst und speziell auf einen LDAP-Server gefallen. Als Server wird OpenLDAP (http://www.openldap.org) verwendet. OpenLDAP ist ein freier LDAP-Server mit verfügbaren Sourcen.

Für die persistente Speicherung von Perl-Objekten werden zudem folgende Perl-Module benötigt.

`NET::LDAP`	LDAP-Client
`Persistent::LDAP`	Klasse für persistente Objekte

Sie können auf CPAN (Comprehensive Perl Archive Network, http://www.cpan.org) gefunden werden.

Ein persistentes Objekt wird durch eine Klasse, die von Persistent::LDAP abgeleitet ist, erreicht. Diese Klasse enthält Methoden zum Initialisieren, Lesen und Speichern der Attribute, die den Zustand des Objekts ausmachen.

Hier ist als Beispiel die Klasse Person angegeben. Sie enthält die persistenten Attribute `uid, userpassword, givenname, sn, cn, mail, telephonenumber` sowie das spezielle Attribut `objectclass`, das im LDAP-Verzeichnis die Klasse festlegt, der das Objekt angehoert. Der

```
package Person;

use Persistent::LDAP;

[...]

sub initialize {
  my $this = shift;

  $this->add_attribute('uid',          'ID',         'String');
  $this->add_attribute('userpassword', 'Persistent', 'String');
  $this->add_attribute('objectclass',  'Persistent', 'String');

  [...]
}
```

Folgendes Perl-Skript verwendet die Klasse Person, um ein Objekt zu erzeugen und es bzw. seinen Zustand im LDAP-Verzeichnis zu speichern. Danach wird der Zustand des Objekts im Speicher zurückgesetzt, aber gleich anhand der Informationen im Verzeichnis wieder hergestellt. Nachdem das Paßwort geändert und die Änderung im Verzeichnis gespeichert wurde, wird das ganze Objekt im Verzeichnis gelöscht.

```
use Person;

my $person;

$person =
  new Person('ldaphost', 389, 'cn=Manager,dc=ixos,dc=de', 'password',
             'dc=ixos,dc=de');

# INSERT

$person->clear();
$person->uid('peter');
$person->userpassword('EaDxG3Ms1sDWc');
$person->objectclass('dcobject');
$person->insert();

# RESTORE

$person->clear();
$person->restore('peter');
$userpassword = $person->userpassword();

# UPDATE

$person->userpassword('XudLPtY3BEKLQ');
$person->update();

# DELETE

$person->delete('peter');
```

Nach der kurzen Einführung in die technischen Aspekte der Datenhaltung, sollen nun das Schema der Daten beschrieben werden.

Datensicherheit

Die Datenhaltung muß die Sicherheit und den Schutz der gespeicherten Daten gewährleisten.

Zur Sicherheit gehört der Schutz vor Datenverlust durch ein Backupkonzept und vor unberechtigten Veränderungen durch Authentifizierung und Autorisierung. Das Backup wird regelmäßig durchgeführt. Dazu ist es erforderlich, den schreibenden Zugriff zu unterbinden, um einen konsistenten Datenbestand sichern zu können. Der Schutz vor unberechtigten Veränderungen wird durch die LDAP-Zugriffskontrolle erreicht [Pro00]. Für den schreibenden Zugriff auf die Zugriffskontrollinformationen des Berechtigungskonzepts wurde ein spezieller Benutzer im LDAP-Verzeichnis definiert. Nur dieser Benutzer besitzt das Recht zum Schreiben in dem Verzeichnisbereich, der die Zugriffskontrollinformationen enthält.

Der Datenschutz spielt bei den hier behandelten Daten für die Zugriffskontrolle keine Rolle. Es sind keine personenbezogenen Daten wie z.B. Adressen oder Telefonnummern gespeichert.

Datenumfang

Zum Datenumfang der Datenhaltung gehören die Attribute der Berechtigungsobjekte, Berechtigungen, Berechtigungsprofile, Berechtigungsgruppen und der Benutzer. Die Daten des ACC (Access Control Context) werden zur Laufzeit bestimmt, da es sich hierbei um Informationen wie Uhrzeit oder Systemlast handelt.

Berechtigungsobjekt

Attribut	Mögliche Werte
Name	Bezeichner (z.B. Pts_Case)
Beschreibung	z.B. „Erforderlich für Bearbeitung von Supportällen“
Klasse	Bezeichner einer Berechtigungsklasse (z.B. PTS)
Felder	Liste der zu prüfenden Felder inkl. der Wertemengen

Berechtigung

Attribut	Mögliche Werte
Name	Bezeichner (z.B. Pts_Case.Show)
Beschreibung	z.B. „Ermöglicht das Ansehen von Supportfällen“
B.objekt	Bezeichner des Berechtigungsobjekts
Werte	Liste der Werte zu den Feldern des B.objekts

Berechtigungsprofil

Attribut	Mögliche Werte
Name	Bezeichner (z.B. Angestellter)
Beschreibung	z.B. „Kann alle Supportfälle bearbeiten“
Berechtigungen	Liste der enthaltenen Berechtigungen

Berechtigungsgruppe

Attribut	Mögliche Werte
Name	Bezeichner (z.B. ixos)
Beschreibung	z.B. „Kann auf alle Installationen zugreifen“
Instno	Liste von Installationsnummern oder „cid“

Benutzer

Attribut	Mögliche Werte
Name	Bezeichner (z.B. adam)
B.profile	Liste der zugewiesenen Berechtigungsprofile
B.gruppe	Die zugewiesenen Berechtigungsgruppe

7.1.2 Entscheidungskomponente (Access Control Decision Facility, ADF)

Die Entscheidungskomponente hat die Funktion, aufgrund entsprechender Anfragen der Durchsetzungskomponente zu entscheiden, ob ein Zugriff erlaubt oder verboten werden soll. Die Entscheidung wird dann der Durchsetzungskomponente mitgeteilt, die die Entscheidung umsetzen muß.

In dem hier entwickelten System entspricht die Entscheidungskomponente den Klassen Buser, Bgroup, Bprofile, Brecht und Bobj. Eine Entscheidungsfindung wird durch einen Aufruf der Methode `Buser::check_authority` bewirkt. Als Argumente sind ein Bezeichner eines Berechtigungsobjekts und ein Hash mit den erforderlichen Werten der zu prüfenden Berechtigung notwendig.

Soll zum Beispiel überprüft werden, ob der Benutzer `Adam` die Berechtigung hat, den den Supportfall mit der Fallnummer `4678` anzusehen, wäre ein Methodenaufruf folgender Art geeignet:

```
# Benutzerobjekt erzeugen
$user = &Buser->new('Adam');

# Berechtigung prüfen
$decision = $user->check_authority( 'PTS_CASE',
                                    ( 'CASENO' -> '4678',
                                      'ACTVT'  -> 'DISP'
                                    )
                                  );
```

Die Methode `Buser::check_authority` (siehe Anhang A) liefert bei einer positiven Entscheidung den Wahrheitswert `TRUE`, ansonsten `FALSE` zurück. Im der Abbildung 7.3 ist der Prozeß der Entscheidungsfindung dargestellt.

7.1.3 Durchsetzungskomponente (Access Control Enforcement Facility, AEF)

Die Durchsetzungskomponente erhält indirekt durch den Benutzer eine Anfrage zur Durchführung einer Funktion. Bevor die Funktion ausgeführt wird, fordert die Durchsetzungskomponente eine Entscheidung über die Gewährung oder die Ablehnung des Zugriffs.

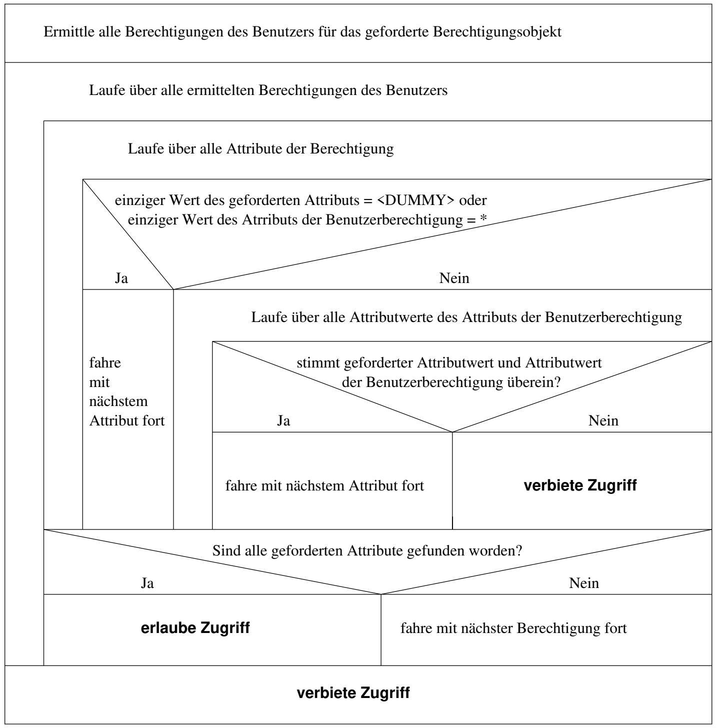

Abbildung 7.3: Ablauf der Entscheidungsfindung

Basierend auf der Antwort der Entscheidungskomponente muß die Durchsetzungskomponente auf die Anfrage des Benutzers reagieren.

In dem hier entwickelten System erfolgt die Durchsetzung der Entscheidungen in den einzelnen Methoden der Benutzerschnittstelle zum PTS. Diese sind in der Klasse Bpts gekapselt. Dazu gehören die Methoden _htmlsearch, _htmlnew und _htmlshow.

Hier soll als Beispiel die Methode _htmlsearch betrachtet werden. Sie besteht aus zwei verschachtelten Schleifen. Die äußere läuft über die Installationsnummern, die dem Benutzer in der Benutzergruppe zugeordnet sind. Die innere Schleife läuft über die Supportfälle der jeweiligen Installation. Für jeden Supportfall wird dabei die Entscheidungskomponente beauftragt, zu ermitteln, ob der Benutzer die Berechtigung hat, diesen Fall anzusehen. Fällt die Entscheidung positiv aus, wird der Fall angezeigt, ansonsten wird mit dem nächsten

Fall fortgefahren.

```
package Bpts;

[...]

sub _htmlsearch {
  print "<HEAD><TITLE>Case List</TITLE></HEAD>\n";

  foreach $instno (@instnos) {

    # Ermittle Fallnummern offener Fälle zur aktuellen Installation
    my $casenos = &PtsCaseList->new()->listopenno($instno);
    my $first=1;
    my $caseno;

    foreach $caseno (@$casenos) {

      # Erzeuge Objekt zu Supportfall
      my $case = $pcl->getcase($caseno);

      # Anfrage an Entscheidungskomponente:
      # Darf der aktuelle Benutzer diesen Supportfall ansehen?

      my $sy_subrc = Buser::ruser()->check_authority('pts_case', {
                     'ACTVT' => 'DISP',                   # Ansehen
                     'CUSTOMER' => $case->get_Customer,   # Installationsnummer
                     'CASENO' => $case->get_CaseNo,       # Fallnummer
                     'STATUS' => $case->get_Status,       # Status
                     'PRIORITY' => $case->get_Priority,   # Priorität
                                   });

      # Falls ja, zeige den Fall an

      if ($sy_subrc != 0) {
        [...]
      }
    }
  }
}
```

Die Reaktion auf die Entscheidung der Entscheidungskomponente ist damit dem Programmierer überlassen. Er ist für die passende Durchsetzung der Entscheidung verantwortlich. Genauso ist der Programmierer verantwortlich, geeignete Berechtigungsobjekte zu definieren und Berechtigungen an geeigneten Stellen mittels der Entscheidungskomponente zu überprüfen.

7.2 Beispiel

Soll eine neue Funktion unter Nutzung des Berechtigungskonzepts in das System aufgenommen werden, sind einige Schritte notwendig. Hier soll exemplarisch die einfache neue Funktion „Anfrage an Support schicken“ umgesetzt werden. Es soll einen Button auf der Startseite des Systems geben, durch den man auf eine Seite mit einem Webformular gelangt, das eine Email an den Support generiert.

Dazu muß man sich zuerst überlegen, von welchen Parametern diese Funktion abhängig sein soll. Als denkbarer Faktor kann man die Tageszeit oder ein Anfragenkontingent, das die Anzahl der Anfragen pro Zeiteinheit in einem gewissen Rahmen hält, hinzunehmen.

Es soll also folgende Berechtigungsfelder geben:

- Tageszeit - wann darf eine Anfrage an den Support geschickt werden
- Quota - wieviele Anfragen dürfen monatlich an den Support geschickt werden

Im nächsten Schritt werden Berechtigungsobjekte definiert. Dies sind Gruppen von Berechtigungsfeldern, die zusammen geprüft werden und alle erfüllt sein müssen. Hier wird es nur ein Berechtigungsobjekt geben, das das Generieren einer Anfrage schützen und die oben genannten Berechtigungsfelder enthalten wird. Das neue Berechtigungsobjekt wird ANFRAGE heißen und muß in das Verzeichnis (siehe Kap. 7.1.1) mit folgendem Befehl eingefügt werden.

```
ldapadd -D "cn=manager, dc=ixos, dc=de" -W
dn: dc=ANFRAGE,dc=bobj,dc=ts,dc=ixos,dc=de
desc: Anfrage an Support
dc: ANFRAGE
objectclass: dcobject
field: Tageszeit
field: Quota
^d
```

Dadurch ist das Berechtigungsobjekt definiert und dem System bekannt gemacht. Als nächstes können Berechtigungen erstellt werden. Dazu muß überlegt werden, wer, wann und wieviele Anfragen stellen darf. Hier soll angenommen werden, daß vorerst alle Benutzer in der Zeit von 8-16 Uhr Anfragen stellen dürfen und zwar 10 mal pro Monat. Dazu wird eine Berechtigung im Verzeichnis erstellt.

```
ldapadd -D "cn=manager, dc=ixos, dc=de" -W
dn: dc=ANFRAGE_Kunde,dc=brecht,dc=ts,dc=ixos,dc=de
desc: 10 Anfragen pro Monat, 8-16 Uhr
dc: ANFRAGE
objectclass: dcobject
bobj: ANFRAGE
bobjvalue: Tageszeit;8-16
bobjvalue: Quota;10
^d
```

Die Semantik der Werte wird durch den Programmierer der neuen Funktion festgelegt. Hier soll Tageszeit Stunden von 0 bis 23 enthalten, die die Stunden eines Tages bestimmen an denen Anfragen gestellt werden dürfen. Quota enthält die Anzahl von Anfragen, die pro Monat gestellt werden können.

Als nächstes wird die neue Berechtigung `ANFRAGE_Kunde` in das Berechtigungsprofil `Kunden` aufgenommen und ist somit allen Kunden zugeteilt, denen das Berechtigungsprofil zugeordnet ist.

Diese Operationen können auch über eine Web-Schnittstelle durchgeführt werden, um die Administration komfortabler zu gestalten.

Der Ausschnitt aus dem Code zum Aufbau der Startseite des Systems wird dann etwa folgendermaßen aussehen.

```
...
# Benutzerobjekt erzeugen (mit seinen Berechtigungen)
my $ruser = &Buser::ruser();
# Ermitteln der aktuellen Stunde des Tages
my $hour = (localtime(time))[2];

# Überprüfe die Berechtigung
# check_authority liefert 1, wenn die Berechtigung vorhanden ist,
# ansonsten 0.
my $sy_subrc = $ruser->check_authority(
                  # ANFRAGE-Berechtigungsobjekt
                  'ANFRAGE',
                 {
                  # Tageszeit-Feld
                  'Tageszeit' => $hour,
                  # Quota-Feld
                  'Quota' => $ruser->get_Anfrage_Quota();
                 });
if ($sy_subrc) {
   # Erzeuge Button für Anfrage
   ...
}
...
```

Der Ausschnitt aus dem Code zum Erzeugen einer Anfrage sieht dann etwa folgendermaßen aus. Hier wird mit `check_authority` geprüft, ob der Benutzer berechtigt ist, eine Anfrage zu stellen. Dazu wird als erster Parameter das zu prüfende Berechtigungsobjekt durch seinen Namen angegeben, hier `ANFRAGE`. Als zweiter Parameter folgt ein Hash mit den erforderlichen Berechtigungsfeldern und ihren Werten, hier `Tageszeit = $hour, Quota = $ruser->get_Anfrage_Quota()`. Die Methode `check_authority` des Benutzerobjekts liefert daraufhin eine 1 zurück, wenn es eine Berechtigung des Benutzers gefunden hat, die die geforderten Werte enthält, ansonsten eine 0. Danach wird entweder das Formular zur Eingabe der Anfrage aufgebaut oder die Eingabe an den Support weitergeleitet.

```
sub erzAnfrage {
   # Benutzerobjekt erzeugen (mit seinen Berechtigungen)
   my $ruser = &Buser::ruser();
   # Ermitteln der aktuellen Stunde des Tages
   my $hour = (localtime(time))[2];
   # undefiniert = erzeuge Webformular, 1 = schicke Anfrage ab
   my $step = $ARG{step};

   # Überprüfe die Berechtigung
   # check_authority liefert 1, wenn die Berechtigung vorhanden ist,
   # ansonsten 0.
   my $sy_subrc = $ruser->check_authority(
                  # ANFRAGE-Berechtigungsobjekt
```



```
                  'ANFRAGE',
                 {
                  # Tageszeit-Feld
                  'Tageszeit' => $hour,
                  # Quota-Feld
                  'Quota' => $ruser->get_Anfrage_Quota();
                 });
   if ($sy_subrc) {
      if (! $step) {
         # erzeuge Webformular
         ... <INPUT TYPE=HIDDEN NAME=step VALUE=1> ...
      } elsif ($step == 1) {
         # schicke Anfrage ab
      } else {
         # Fehler
      }
   } else {
      # keine Berechtigung, Fehlermeldung
   }
}
```

7.3 Zugriff auf PTS-Daten

Da die PTS-Skripten nicht objektorientiert implementiert sind, galt es eine Schnittstelle zu finden, die einen Übergang zum neuen objektorientierten System ermöglicht. Dazu wurden die Klassen PtsCase und PtsCaseList eingeführt. Die Klasse PtsCase kapselt einen PTS-Fall. Dazu zählen neben den Attributen auch die Methoden, die das Verhalten eines Falls ausmachen. Die Klasse PtsCaseList dient dazu, mit einer größere Menge von Fällen einfach umgehen zu können.

7.3.1 PtsCase

PtsCase erlaubt es über die Fallnummer einen Fall aus der PTS-Datenhaltung einzulesen und Änderungen am Fall wieder in die Datenhaltung zu schreiben. Dazu gibt es die Methoden `_create()` und `set_values()`. `_create()` besitzt als Parameter nur die Fallnummer. Als erstes wird die Funktion `readcase()` aus `sup-general.pl` aufgerufen. Diese findet durch die eindeutige Fallnummer das Fall-Verzeichnis und füllt hauptsächlich die Datenstrukturen `%cidx` und `%cont` mit den aktuellen Fall-Daten und auch den Änderungen in der Vergangenheit.

```
sub _create() {
  my $self = shift;
  my $caseno = shift;
  my $prod = '';
  &::readcase($prod, $caseno);
  $self->{'_Product'} = $prod;
}
```

Im Anschluß an den Aufruf von `readcase()` werden die Attribute des PtsCase-Objekts mit den Daten gefüllt, die sich bisher nur in den Hashs `%::cidx` und `%::cont` befinden und beim

Einlesen eines weiteren PTS-Falls überschrieben würden. Das Übertragen der aktuellen Falldaten in die Objekt-Attribute geschieht mittels der Funktionen `::currcont()` bzw. `::currcidx()` aus `sup-general.pl`.

```
foreach $k (@{$::casefields{$prod}}) {
  next if (grep($_ eq $k, @::fields_actilog));
  $self->{"_".$k} = &::currcont($k, '');
  $self->{"_timestamps"}->{$k} = &::currcidx($k);
}
```

Das Ändern eines PTS-Falls ist etwas aufwendiger, weil bestimmte Konsistenzbedingungen erfüllt sein müssen. Auf diese Bedingungen soll hier aber nicht näher eingegangen werden.

Nach der Konsistenzprüfung wird durch die Funktion `&::lock()` der Fall exklusiv vor Veränderungen geschützt und die Status-Datei zum Schreiben geöffnet. Danach wird jedes Attribut des PTS-Falls, das sich gegenüber dem gespeicherten Zustand verändert hat durch die Funktion `&::writecasefield()` in die Status-Datei geschrieben. Nachdem der Fall entsperrt worden ist, wird noch der Index aktualisiert.

```
sub setvalues {
  my $self = shift;
  my $stfile = $self->get_casedir();
  $stfile .= "/.status";

  [...]

  # save values
  &::lock("case.".$self->get_CaseNo());
  open(main::ST, ">>$stfile");

  foreach $k (keys %::ARG) {
    my $f=''; my $e='';
    my $v = $::ARG{$k};

    # filter unchanged fields
    next if ($v eq $::cont{$k});

    # filter fields that have been changed in the meantime
    $::cidx{$f} =~ /([^:]+):$/;
    my $olde = $1;
    next if ($e ne $olde && $e ne '' && not grep($_ eq $f, @::fields_actilog));

    next if (not grep($_ eq $f, @{$::casefields{$self->get_Product()}}));

    my $method = "set_".$f;
    $self->$method($v,&::currcidx($f));

    &::writecasefield($f, $k);
  }
  close ST;
  &::unlock("case.".$self->get_CaseNo());

  $::number = $self->get_CaseNo();
  $::prod = $self->get_Product();

  &::writeindex;
}
```


7.3.2 PtsCaseList

PtsCaseList dient der vereinfachten Handhabung der Menge der PTS-Fälle.

Beim Erzeugen eines PtsCaseList-Objekts durch die Methode `new` wird durch die Funktion `&::readsupindex()` der Index der offenen PTS-Fälle eingelesen. Dieser Index wird durchlaufen und jeder Eintrag, dessen Installationsnummer durch die Benutzergruppe des Benutzers abgedeckt ist, wird in das Attribut `idx` des PtsCaseList-Objekts übernommen. Der Hash `idxno` liefert den Array-Index zu einer namentlich bekannten Spalte im Index, der beim Erzeugen eines PtsCaseList-Objekts eingelesen wird.

```
sub new {
  my $that = shift;
  my $class = ref($that) || $that;
  my $case;
  $self = {
    $idx => [],
  };
  bless $self, $class;
  &::readsupindex();
  foreach $c (@::supidx) {
    push(@{$self->{'idx'}}, $c) if (Buser::ruser()->check_inst($c->[$::idxno{'Customer'}]));
  }
  undef @supidx;
  return $self;
}
```

Die Methode `listinstno` liefert eine Liste mit den Installationsnummern, zu denen PTS-Fälle existieren und die in der Benutzergruppe, die dem Benutzer zugewiesen ist, direkt oder indirekt durch den String „cid“ angeben sind.

```
sub listinstno {
  my $self = shift;
  my @il = ();
  foreach $c (@{$self->{'idx'}}) {
    my $ino = $c->[$::idxno{'Customer'}];
    push(@il, $ino) if (! grep(/$ino/, @il));
  }
  return \@il;
}
```

Die Methode `listopenno` liefert eine Liste mit Fallnummern von offenen Fällen, die einer Installation zugeordnet sind, die in der Liste von `listinstno` vorkommt.

```
sub listopenno {
  my $self = shift;
  my $instno = shift || 0;
  my @cl = ();
  foreach $c (@{$self->{'idx'}}) {
    my $cno = $c->[$::idxno{'caseno'}];
    my $ino = $c->[$::idxno{'Customer'}];
    next if ($c->[$::idxno{'Status'}] =~ /^closed/);
    push(@cl, $cno) if ((! $instno) || $instno == $ino);
  }
  return \@cl;
}
```

Die Methode `getcase` erzeugt ein PtsCase-Objekt und initialisiert dessen Attribute mit den bereits eingelesenen Indexwerten. Dadurch wird das Einlesen des PTS-Falls vom Filesystem gespart.

```
sub getcase {
  my $self = shift;
  my $caseno = shift;
  my $val;
  foreach $val (@{$self->{'idx'}}) {
    if ($val->[$::idxno{'caseno'}] == $caseno) {
      my $c = new PtsCase($caseno, $val);
      return $c;
    }
  }
}
```

7.4 Optimierungen

Nach der Implementierung der Entscheidungskomponenten und der Installation des OpenLDAP-Servers, standen erste Tests an. Als ein Schwachpunkt stellte sich die schlechte Performance des `Persistent::LDAP` Perl-Moduls heraus. Das erzeugen eines Objekts mit persistenten Attributen hat mehrere zehntel Sekunden benötigt. Bei 10 Berechtigungen mit den zugehörigen Berechtigungsobjekten ergibt sich schnelle eine Gesamtzeit von 10 bis 20 Sekunden, was einem Benutzer als Antwortzeit nicht zumutbar ist.

Daher wurde nach einer Alternative zu dem `Persistent::LDAP` Perl-Modul gesucht. Es fand sich das Modul `Tie::LDAP`, das einen Perl-Hash an ein LDAP-Verzeichnis binden kann.

Ein Hash wird durch Angabe von LDAP-Host, Benutzer, Paßwort und Base an ein LDAP-Verzeichnis gebunden.

```
tie %LDAP, 'Tie::LDAP', { base => 'o=IMASY, c=JP' };
```

Um auf einen Eintrag aus dem LDAP-Verzeichnis zugreifen zu können, muß der Distinguished Name (DN) angegeben werden.

```
$info = $LDAP{q{cn=tai, o=IMASY, c=JP}};
```

Ein Attribut eines LDAP-Eintrags kann mehrere Werte enthalten. Deswegen werden diese von `Tie::LDAP` als Listen verwaltet.

```
$user = $info->{username}->[0];
$mail = @{$info->{mailaddr}};
```

Änderungen von Attributwerten erfolgen durch einfache Wertzuweisungen. Wobei es sich hierbei wieder um Listen handelt.


```
$LDAP{q{cn=tai, o=IMASY, c=JP}} = {
  username => ['newname'],
  mailaddr => ['tai@imasy.or.jp', 'tyamada@tk.elec.waseda.ac.jp'],
};
```

Mit dem `Tie::LDAP` Modul ist eine deutliche Geschwindigkeitszunahme erreicht worden. Dies liegt unter anderem wahrscheinlich an der in C implementierten Netzwerkschicht, im Gegensatz zu `Persistent::LDAP`, bei der alles in Perl programmiert ist.

Kapitel 8

Abschließende Betrachtungen

Das Hauptziel dieser Arbeit war die Integration von PTS und CID in das ESC, um den Zugriff von Kunden und Partnern zu ermöglichen. Dabei waren besonders die Sicherheitsaspekte zu berücksichtigen. Denn bei PTS und CID handelt es sich um Systeme, die kundenbezogene Daten enthalten. Deshalb galt es die unterschiedlichen Benutzersichten mit dem Schutz der Daten in Einklang zu bringen.

Nach der Vorstellung der bestehenden Systeme PTS, CID und ESC sowie MKIS, wurden die für den Kunden zugänglichen Online-Supportsysteme von SAP, RedHat und Oracle betrachtet und bewertet. Vor allem wurde die Flexibilität der Systeme untersucht. Dazu gehörte, in wie weit die Benutzer weitere Benutzer anlegen, löschen und deren Rechte modifizieren können. Hier bietet das SAPNet die meisten Möglichkeiten. Die Flexibilität dieses Online-Supportsystems entstand durch die Verwendung des Berechtigungskonzepts, wie es im SAP R/3 vorkommt. Es besteht aus parametrisierten Berechtigungen, die in Profilen zu Rollen vereint werden. Solche Profile können auch andere Profile enthalten. Daher kennt das Berechtigungskonzept auch die Vererbung von Berechtigungen.

Die Administration eines solchen Berechtigungskonzepts muß unanfällig gegen Bedienungsfehler und flexibel zugleich sein. Um beides zu erreichen, wurden geläufige Sicherheitsmodelle wie das „Generalized Framework for Access Control“ (GFAC) und der amerikanische Kriterienkatalog TCSEC („Orange Book“) herangezogen. TCSEC enthält die Oberbegriffe Sicherheitspolitik, Überwachung und Durchsetzen der Sicherheitspolitik, um die Funktionalität von Sicherheitsmechanismen zu beurteilen.

Aus den Erkenntnissen der Untersuchung der Online-Supportsysteme von SAP, RedHat und Oracle sowie den Kriterien aus TCSEC wurde unter Einbeziehung des GFAC ein Berechtigungskonzept für das neue System entwickelt.

Die im GFAC definierten Zugriffskontroll- und Autorisierungsinformationen wurden in einer eigenen Komponente, dem LDAP-Server, zusammengefaßt. Zugriffe auf den LDAP-Server erfolgen ausschließlich über die Persistent::LDAP-Klasse, von der die Klassen des

Berechtigungskonzepts abgeleitet sind.

Zugriffe auf das PTS über das ESC sind nur über definierte Objekte möglich. Die Durchsetzung der Zugriffskontrolle geschieht in diesen Objekten an geeigneten Stellen. Dabei wird die Entscheidung durch die Entscheidungskomponenten des Berechtigungskonzepts getroffen. Parameter der Entscheidungsfunktion sind die Anforderung, die in abstrahierter Form die gewünschte Funktionalität beschreibt und der Benutzername. Die Entscheidungskomponente fordert die benötigten Attribute vom LDAP-Server an und trifft die Entscheidung.

Die Kodierung erfolgte in der Programmiersprache der bestehenden Systeme, Perl. Die Entscheidungskomponente und die Erweiterungen des PTS um die neuen Benutzerschnittstellen mit den Zugriffskontrollen wurden objektorientiert entworfen und implementiert.

Die Benutzerschnittstellen für die CID wurden im Rahmen dieser Diplomarbeit noch nicht implementiert, können aber nach dem gleichen Prinzip wie die beim PTS jederzeit nachgeholt werden.

Wegen der zum PTS hinzugekommen Benutzerschnittstellen und des neuen Berechtigungskonzepts ist es nun auch möglich, Kunden und Partnern den Zugriff auf ihre PTS-Fälle zu erlauben. Auch andere zusätzliche Rollen sind zukünftig schnell definier- und konfigurierbar.

Momentan existiert allerdings aufgrund des zusätzlichen Berechtigungskonzepts ein administrativer Zustatzaufwand bei der Einrichtung eines neuen Accounts. Einem neuen Account muß zusätzlich zu den bisher notwendigen Eigenschaften im PTS eine Rolle (Profil) im neuen Berechtigungskonzept zugeordnet werden, damit ein Zugriff über das ESC möglich ist. Dieser Vorgang ist aber automatisierbar.

Eine weiterer Nachteil ist momentan noch die Verwendung des LDAP-Servers als Datenhaltung der persistenten Objekte. Durch den bei jedem Zugriff auf das PTS notwendigen Zugriffskontrollen müssen auch die Zugriffskontrollinformationen jedesmal neu im LDAP-Server abgefragt werden. Der Zeitverlust dabei bewegt sich im Rahmen von wenigen Sekunden, ist aber auf dauer nicht akzeptierbar. Deswegen wurde ein Versuch unternommen, den Zugriff auf das LDAP-Verzeichnis anstatt durch die Persistent-Klasse mit dem Tie-Mechanismus zu realisieren. Dabei wurden deutliche Verbesserungen in der Antwortzeit erreicht.

Insgesamt sind die Ziele voll erreicht worden und eine Schnittstelle zum PTS mit hoher Sicherheitsfunktionalität entstanden. In der Fortführung dieses Projekts werden gezielt die bestehenden nur im Intranet verfügbaren Schnittstellen zur CID um die Sicherheitsfunktionalität erweitern, im Internet den Kunden und Partner zur Verfügung gestellt und die Geschwindigkeit der Datenhaltung verbessert.

Anhang A

Quelltext

Auf den folgenden Seiten findet sich der Quellcode ausgewählter Methoden des Berechtigungskonzepts.

A.1

```
package Buser;

[...]

sub check_authority {
  my $self = shift;
  my $bobjname = shift;
  my $value = shift;
  my $user = $self;

  # erstelle geforderte Berechtigung

  my $b = Brecht->new($bobjname, $value);

  # falls Benutzer eine andere Rolle eingenommen hat
  # und keine Zugriffsverwaltungsberechtigung überprüft
  # wird, verwende die eingenommene Rolle.

  if ($user->forceuserobj &&
      $bobjname ne 'user_aut' &&
      $bobjname ne 'user_pro' &&
      $bobjname ne 'user_set' &&
      $bobjname ne 'user_grp') {
    $user = $self->forceuserobj;
  }

  # leite die Berechtigungsüberprüfung weiter
  return ($b->check_authority($user));
}



package Brecht;

[...]

sub check_authority {
  my $self = shift;
  my $u = shift;  # zu überprüfender Benutzer
  my $match = 0;  # Ergebnis der Überprüfung

  # Berechtigungen des Benutzers vom Typ der zu überprüfenden Berechtigung
  my @ubrechts = $u->getbrecht($self->bobj);

  # erforderliche Werte der geforderten Berechtigung
  my $selfval = $self->brechtvalue;

  # Überprüfe jede Berechtigung des Benutzers
  for($ubrid=0; $ubrid<=$#ubrechts && !$match; $ubrid++) {
    my $ubrecht = $ubrechts[$ubrid];  # aktuelle Berechtigung des Benutzers
    my $uval = $ubrecht->brechtvalue;  # Werte der aktuellen Berechtigung des Benutzers
    $match=1;  # Initialisierung das Ergebnisses

    # Überprüfe jeden Wert der aktuellen Berechtigung
    foreach $i (keys %{$selfval}) {
      my $curselfval = $selfval->{$i};  # aktueller Wert der aktuellen Berechtigung

      # Extrahiere gültige Feldaktionen (DISP,EDIT,...) aus dem aktuellen Wert
      $uval->{$i} =~ /^([^;]+);?(.*)$/;  #
      my $curuval = $1;
      my $curuperm = $2;

      # Falls das Feld den Namen CUSTOMER hat und es sich um eine Berechtigung
      # vom Typ PTS_CASE handelt, wird der aktuelle Wert auf die dem
      # Benutzer zugewiesenen Installationen gesetzt.

      if ($i eq 'CUSTOMER' && $ubrecht->bobject->name eq 'pts_case') {
        $curuval = join(',',@{$u->iid()});
      }

      # Falls der aktuelle Wert der geforderten Berechtigung ein
      # einzelner Wert der Form $[A-Z]$ hat, handelt es sich um
      # eine Überprüfung der Berechtigung für ein einzelnes Feld.
      # Kommt dieser Wert ohne den beiden $ in der Liste der Feldberechtigung
      # des Benutzers vor, erlaube den Zugriff, ansonsten nicht.

      if (not $curselfval =~ /,/) {
        if ($curselfval =~ /^\$([A-Z]+)\$$/) {
          my $spbrfound = $1;
          if (grep(/$spbrfound/, split(/,/,$curuperm))) {
            $match = 0;
            last;
          } else {
            $match = 1;
            last;
          }
        }
      }

      # Soll ein Wert nicht überprüft werden, ist dieser Wert der geforderten
      # Berechtigung "<DUMMY>".
      # Ist der aktuelle Wert der aktuellen Benutzerberechtigung "*", muß
      # ebenfalls kein Wertevergleich durchgeführt werden.

      next if ($curselfval eq '<DUMMY>' ||
               grep(/^\s*\*\s*$/, split(/,/,$curuval)));


      # vergleiche die Werte der geforderten Berechtigung mit denen der
      # aktuellen Benutzerberechtigung.

      if ($curuval =~ /,/) {
        my @val = split(',', $curuval);
        my $j;
        for($j=0; $j<=$#val; $j++) {
          $val[$j] =~ s/^\s*(\S*)\s*$/$1/;
          last if ($curselfval eq $val[$j]);
        }
        $match = 0 if ($j > $#val);
      } else {
        $match=0 if ($curselfval ne $curuval);
      }

      # breche die Berechtigungsprüfung ab, sobald ein Wert nicht matched

      last if (!$match);
    }
  }

  return $match;
}
```

Literaturverzeichnis

[Bar97] John Barkley. Comparing simple role based access control models and access control list. 1997.

[Bra99] Carlheinz Braun. *ISO Document: Arbeitsanweisung 19 120 Hotline*. IXOS Software AG, Grasbrunn, 1999.

[Bun97] Bundesamt für Sicherheit in der Informationstechnik. *IT-Grundschutzhandbuch*. Bundesanzeiger-Verlag, 1997.

[Con99] Damian Conway. *Object Oriented Perl*. Manning, 1999.

[Cze98] Barbara Czegel. *Running an Effective Help Desk*. John Wiley & Sons, 1998.

[Ewa97] Karl Ewald. *General Overview and Notes on Handling Support Web Systems*. Grasbrunn, 1997.

[Ewa99] Karl Ewald. *Documentation for the Problem Tracking System*. IXOS Software AG, Grasbrunn, 1999.

[FK92] Ferraiolo and Kuhn. Role based access control. *15th National Computer Security Conference*, 1992.

[G$^+$95] E. Gamma et al. *Design Patterns*. Addison-Wesley, 1995.

[GM98] Ariel Glenn and David Millman. Access management of web-based services. *D-lib Magazine*, (9), 1998.

[HK95] Hickman and Kipp. *The SSL Protocol*. Netscape Communications Corp., 1995.

[J$^+$92] I. Jacobson et al. *Object-Oriented Software Engineering: A Use Case Driven Approach (ObjectOry)*. Addison-Wesley, 1992.

[LaP95] L. J. LaPadula. *Rule Set Modeling of a Trusted Computer System, Essay, in: Information Security: An Integrated Collection of Essays*. IEEE Computer Society Press, 1995.

[Mon] Perl Mongers. *http://www.perl.org*.

[oD85] Department of Defense. Trusted computer system evaluation criteria. 1985.

[Orw99] Jon Orwant. *Mastering Algorithms with Perl*. O'Reilly and Associates, Inc, 1999.

[Ott97] Amon Ott. *Rule Set Based Access Control as proposed in the 'Generalized Framework for Access Control' approach in Linux*. Universität Hamburg, 1997.

[Pro00] The OpenLDAP Project. Openldap 2.0 administrator's guide. 2000.

[RB91] J. Rumbaugh and M. Blaha. *Object-Oriented Modeling and Design*. Prentice Hall, 1991.

[sap98] *Berechtigungsprüfung - Eine kurze Einführung*. SAP AG, 1998.

[sap99] *SAP Library Release 4.6B*. SAP AG, 1999.

[sap00] *Die SAPNet-Konzernfunktionalität für Customer Competence Center*. SAP AG, 2000.

[Wal99] Egbert Wald. *Helpdesk-Management*. MITP, 1999.

[WS91] Larry Wall and Randal Schwartz. *Programming Perl*. O'Reilly and Associates, Inc, 1991.

[YHK95] W. Yeong, T. Howes, and S. Kille. Rfc 1777: Lightweight directory access protocol. *IETF*, 1995.