Seminararbeit
Rollenbasierte Zugriffskontrolle (RBAC)
Sommersemester 2007

von Alexander Gropp
Betreuerin Jutta Mülle

Institut für Programmstrukturen und Datenorganisation (IPD)
Universität Karlsruhe

Table of Contents

1 Motivation und Übersicht

Für die Rechtevergabe gibt es unterschiedliche Möglichkeiten. Die einfachste besteht in so genannten Access Control Lists (ACL's). Dabei wird für jedes Objekt eine Liste mitgeführt, welcher Benutzer welche Operationen auf dem Objekt ausführen darf. Man kann sich vorstellen, dass dies mit steigender Anzahl an Benutzern und Objekten recht schnell unübersichtlich wird und daher der Managementaufwand sehr hoch ist. An diesem Punkt setzt das Modell der rollenbasierten Zugriffskontrolle (kurz RBAC für role based access control) an. Die Idee ist, dass in der realen (Geschäfts-)Welt ein Benutzer innerhalb einer (Unternehmens-)Umgebung meist eine oder mehrere so genannte Rollen besitzt, z.B. Sekretärin, Abteilungsleiter, Administrator, Student, etc. . Daher ist es naheliegend dies auch in der eingesetzten Software zu modellieren, da sich so die auftretenden (Geschäfts-)prozesse besser auf die Software abbilden lassen und das Sicherheitsmanagement wesentlich vereinfachen, indem man die Rechte basierend auf der jeweiligen Rolle automatisiert vergibt. Der Wunsch nach Single-Sign-On Systemen, die aber gleichzeitig auch ein ausreichendes Maß an Sicherheit bieten, führt zu dem nachfolgenden Modell der rollenbasierten Zugriffskontrolle.
In den folgenden Abschnitten wird nun zuerst das Modell der Benutzergruppen vorgestellt und die Unterschiede zu einem rollenbasierten Zugriffsmodell erläutert, bevor der RBAC Standart betrachtet wird. Anschließend wird eine Erweiterung für diesen Standart gezeigt, sowie die Anwendungsmöglichkeiten bzw. derzeitige reale Anwendung für RBAC. Damit der Standart etwas anschaulicher wird, soll dessen Anwendung an einem Softwareprodukt gezeigt werden. Es handelt sich hierbei um eine Software für Bauunternehmen, die folgende im Betrieb vorkommende Bereiche EDV-mäßig abdeckt:

- Finanzbuchhaltung
- Angebotserstellung
- Lohn

2 Benutzergruppen

Um die Rechtevergabe mit ACL's zu vereinfachen, bildet man Benutzergruppen. Diese stellen im ursprünglichen Sinne nur eine Zusammenfassung mehrerer Benutzer dar, d.h. die Benutzer werden ohne Betrachtung der zu erhaltenden Rechte gruppiert. Man könnte z.B. alle Mitglieder der Abteilung Finanzbuchhaltung zusammenfassen. Meist haben die auf dieser Grundlage entstehenden Gruppen dann schon ein große Anzahl gemeinsamer Rechte. Dadurch ist es dann möglich, diese gemeinsamen Rechten in den ACL's nur für die Gruppe zu verwalten und nicht für den einzelnen Benutzer. Da die Anzahl der Gruppen im Normalfall geringer ist als die der Benutzer, erspart man sich dadurch Arbeit bei der Rechtevergabe. Berücksichtigt man bei der Erstellung der Gruppen zusätzlich die Betrachtung der jeweils zu erhaltenden Rechte, und gruppiert auf

dieser Basis, so bekommt die Benutzergruppe den Charakter einer Rolle. Mann kann dadurch nun die Zuordnung von Rechten zu Benutzern ausschließlich über das Hinzufügen bzw. Entfernen eines Benutzer zu bzw. von einer Gruppe realisieren (Siehe Abbildung 1). Es sei angemerkt, dass wir unter Recht immer die Kombination von Objekt und dafür zulässige Operation verstehen. Dies wird im nächsten Abschnitt aber nochmals genauer erläutert.

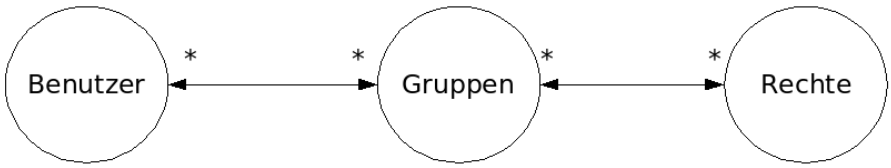

Fig. 1. Modell der Benutzergruppen

Einen großen Nachteil hat das Konzept allerdings. Ein Benutzer hat nämlich dann in jeder Situation die Rechte derjenigen Gruppe mit den meisten Rechten. Das bedeutet, dass er oftmals mehr Rechte besitzt, als er gerade tatsächlich braucht. Dies ist ein Sicherheitsrisiko und widerspricht klar dem least-privilege-Prinzip, das besagt, dass eine Benutzer nur mit den minimal notwendigen Rechten zur Durchführung seiner Aufgabe ausgestattet werden sollte. Ist man unter Windows z.B. Mitglied der Administratorgruppe, so besitzt man diese Rechte auch, wenn man nur zum Surfen im Internet angemeldet ist. Zwei wesentliche Gründe sprechen für die Einhaltung des least-privilege-Prinzips: zum einen schützt es vor Bediehnfehlern (vom Benutzer selbst) und zum anderen vor Angriffen von Außen, die dann mit den effektiven Rechten des Benutzers im System Schaden anrichten können (z.B. Trojaner).

In unserer Beispielanwendung gibt es derzeit 4 Mitarbeiter: Herr Schulz ist der Gruppe Finanzbuchhaltung zugeordnet, Frau Müller der Gruppe Angebotserstellung/Auftragserfassung, Frau Schmidt der Gruppe Lohn und Herr Maier der Gruppe der Systemverwalter. Entsprechend ihres Aufgabenbereichs erhält nun jede Gruppe die notwendigen Rechte. Die Gruppe Finanzbuchhaltung erhält das Recht Bilanzerstellung und Lohneinsicht, die Gruppe Angebotserstellung / Auftragserfassung das Recht Auftrag anlegen, die Lohngruppe das Recht Lohnabrechnung erstellen sowie Lohneinsicht und die Gruppe Systemverwalter erhält alle Rechte. Wird nun ein neuer Mitarbeiter eingestellt, so wird dieser Vorgang (Mitarbeiterakte anlegen, Lohnsteuerkarte, etc.) zweckmäßiger Weise von einem Benutzer der Gruppe Lohn durchgeführt, in der Beispielfirma also von Frau Schmidt. Allerdings scheitert Frau Schmidt mit dieser Aufgabe an dem Punkt,

an dem sie den neuen Mitarbeiter im System einer Gruppe zuordnen soll, da sie hierfür die Rechte des Systemverwalters benötigt. Man kann nun also entweder Frau Schmidt dieser Gruppe zuordnen, womit sie ständig alle Rechte besäße oder aber diesen Vorgang weiter an den Systemverwalter Herrn Maier delegieren, was zu Verzögerung im Ablauf der Personaleinstellung führen kann, bzw. erhöhten Kommunikationsaufwand bedeutet.
Wir werden nun im Folgenden sehen, welche Möglichkeit RBAC bietet, um das eben Beschriebene eleganter zu lösen.

3 Das RBAC Modell

Das Modell der Rollenbasierten Zugriffskontrolle (RBAC) wurde 1996 in [4] vorgestellt. 2003 wurde der Standard der ANSI [3] veröffentlicht.

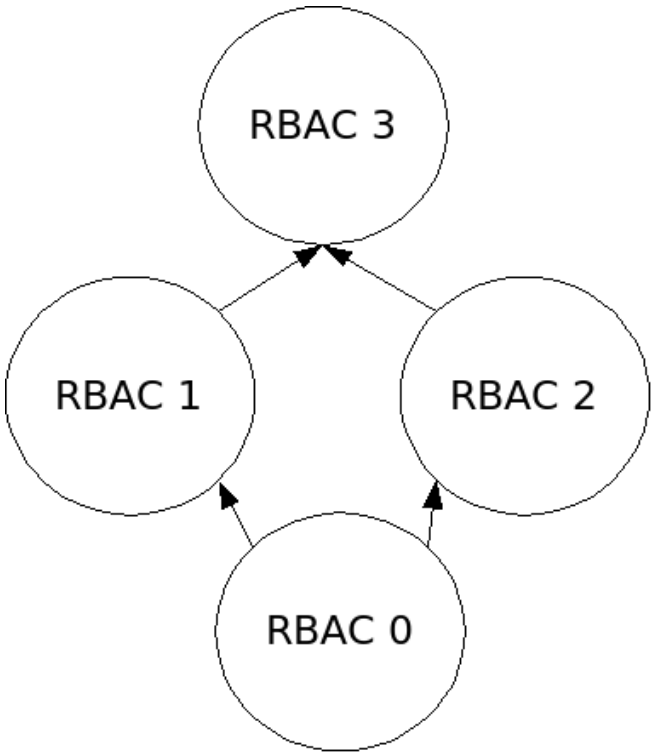

Fig. 2. RBAC Versionen

Das RBAC-Modell (siehe Abbildung 2) besteht aus den Versionen 0 bis 3. Die Version 0 wird als das Kernmodell bezeichnet und enthält die Grundle-

genden Funktionen für ein rollenbasiertes System, die alle anderen Versionen auch enthalten. Die Versionen 1 und 2 setzen jeweils auf dem Kernmodell auf, wobei die beiden Versionen nicht miteinander kompatibel sind. Version 1 bietet die Möglichkeit der hierarchischen Rollenerstellung, wohingegen RBAC 2 die Möglichkeit bietet, Einschränkungen für Rollen zu erstellen. Version 3 stellt die Vereinigung von RBAC 1 und 2 da.

3.1 Das Kernmodell (RBAC 0)

Das Kernmodell enthält folgenden Objekten:

- Benutzern
- Rollen
- Rechte
- Sessions

Unter einer Rolle versteht man eine Funktion innerhalb der Anwendung. Einer Rolle werden nun genau die Rechte zugeteilt, die sie zur Ausübung ihrer Funktionalität braucht. Die Rolle ersetzt also quasi die Benutzergruppen, wobei diese dann zwingend auf der Grundlage der gemeinsamen Rechte gebildet worden sein müssen, da eine Rolle immer die Gruppierung von Rechten und nicht von Benutzern darstellt. Jedem Benutzer wird nun auch entsprechend seiner Aufgabe eine oder mehrere Rollen zugeteilt, allerdings werden beim Login nicht automatisch alle zugeteilten Rollen aktiviert. Vielmehr wird eine Session erzeugt, die die benötigte Rolle(n) aktiviert. Die Zuordnung von Benutzern zu Rollen stellt also nur die maximal möglich Annahme von Rechten da. Die tatsächlich erhaltenen Rechte hängen von der (den) in der Session aktivierte(n) Rolle(n) ab. Zu beachten ist, dass ein Benutzer auch mehrere Sessions haben kann, allerdings kann jede Session nur einen Benutzer haben.

Formal wird RBAC 0 durch folgende Eigenschaften beschrieben:
Jede Zuordnung eines Benutzers zu einer Rolle wird durch ein Tupel bestehend aus dem Benutzer und der jeweiligen Rolle beschrieben:

$$Benutzerzuordnung \subseteq Benutzer \times Rollen$$

Jede Zuordnung von Rechten zu einer Rolle wird durch ein Tupel bestehend aus dem jeweiligen Recht und der Rolle beschrieben:

$$Rechtezuordnung \subseteq Rechte \times Rollen$$

Die Rechte bestehen aus der Potenzmenge der Objekte, die im System bestehen und der Operationen die ausgeführt werden können. Ein Recht ist also eine beliebige Kombination aus verfügbaren Operation und Objekten. Ein Recht für

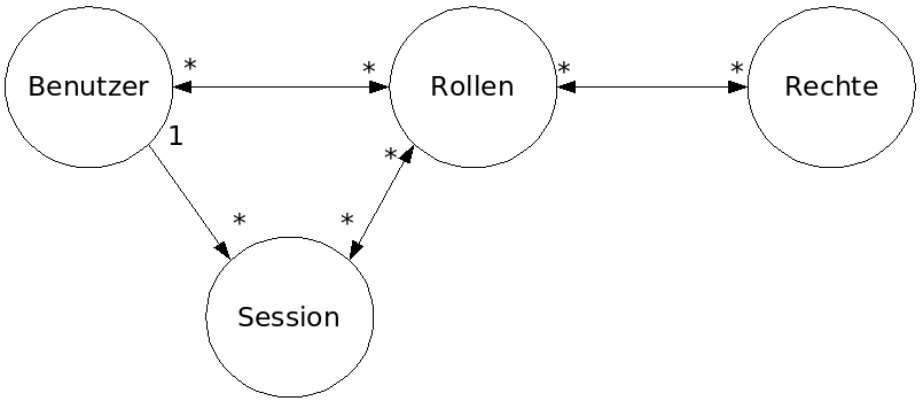

Fig. 3. RBAC 0

die Finanzbuchhaltung könnte z.B. heißen: (Schreiben und lesen, Bilanz):

$$Rechte = 2^{Objekte \times Operationen}$$

Eine Funktion, welche zu der Rolle die zugeordneten Benutzer liefert:

$$zugeordneteBenutzer : Rollen \rightarrow 2^{Benutzer}\{b \in B | (b, r) \in Benutzerzuordnung\}$$

Eine Funktion, welche zu der Rolle die zugeordneten Rechte liefert:

$$zugeordneteRechte : Rollen \rightarrow 2^{Rechte}\{re \in Re | (re, r) \in Benutzerzuordnung\}$$

Eine Funktion, welche zu einer Session den jeweiligen Benutzer liefert:

$$SessionBenutzer : Session \rightarrow Benutzer$$

Eine Funktion, welche zu einer Session die aktivierten Rollen liefert:

$$SessionRollen : Session \rightarrow 2^{Rollen}$$

Eine Funktion, welche zu einer Session die aktivierten Rechte liefert:

$$verfuegbareSessionRechte : Session \rightarrow 2^{Rechte}$$

In der Beispielanwendung soll nun Frau Schmidt aus der Lohnabteilung die Möglichkeit haben, einen neuen Mitarbeiter im System anzulegen, da sie auch die restlichen Formalitäten der Personaleinstellung abwickelt. Dadurch, dass die Gruppen in unserem Beispiel gleichzeitig zur Zusammenfassung von Benutzern auch eine Zusammenfassung von Rechten darstellen, können wir die Gruppen in unsere Rollen im RBAC Modell überführen. Frau Schmidt wird nun sowohl der Rolle Lohn als auch der Rolle Systemverwalter zugeordnet. Der Arbeitsablauf (Workflow) für die Einstellung eines neuen Mitarbeiters sieht nun folgender Maßen aus: Mitarbeiterakte anlegen (1) - Lohnsteuerklasse festlegen (2) - Mitarbeiter entsprechender Rolle (3) zuordnen. Für die Punkte 1 und 2 muss nur die Rolle Lohn aktiviert werden, für Punkt 3 wird die Rolle Systemverwalters aktiviert. Nach Beenden des Workflows 'Mitarbeitereinstellung' wird dann wieder nur die Rolle Lohn aktiviert. Somit ist es für Frau Schmidt nun möglich den kompletten Workflow abzuarbeiten ohne das least-privilege-Prinzip zu verletzen.

3.2 Hierarchie (RBAC 1)

Die erste Erweiterung stellt die Implementierung einer Rollenhierarchie dar.

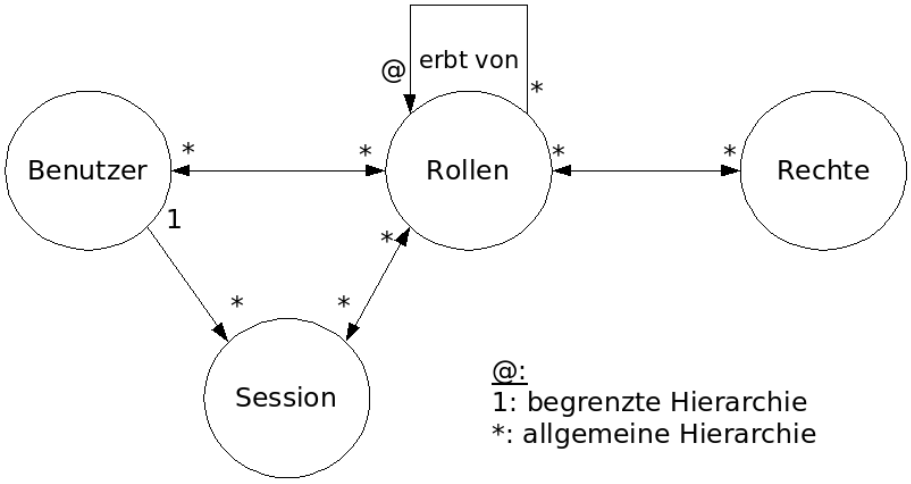

Fig. 4. RBAC 1

Dabei geht es um die Möglichkeit, dass übergeordnete Rollen alle Rechte einer darunterliegenden Rolle erben. Man kann sich dadurch eine Struktur erschaffen, die im wesentlichen die Firmenhierarchie abbildet. Besitzt eine Rolle 2 eine Teilmenge an Rechten einer anderen Rolle 1, so kann Rolle 1 alle Rechte von

Rolle 2 erben. Diese brauchen dann nicht noch einmal explizit zu Rolle 1 zugeordnet werden, sondern nur die Rechte, die die Rolle noch zusätzlich erhalten soll. Genau umgekehrt verhält es sich mit den Benutzern der Rollen. Alle Benutzer der übergeordneten Rolle sind dann auch automatisch Benutzer der darunter liegenden Rolle (Siehe Abbildung 5). Man kann sich 2 abstrakte Rollen definieren, nämlich eine maximale und eine minimale Rolle. Die maximale Rolle enthält dann alle Rechte, die minimale Rolle alle Benutzer.

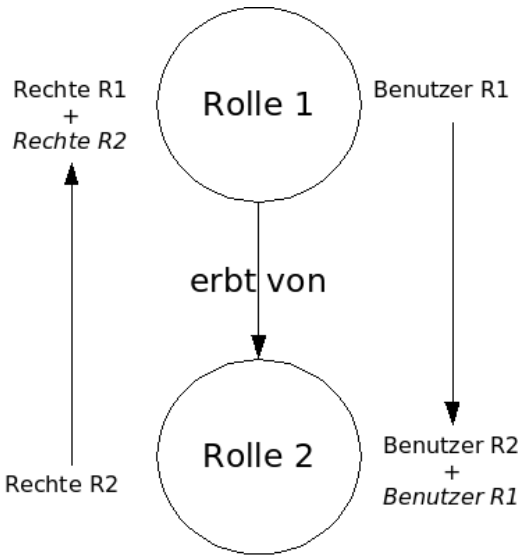

Fig. 5. Vererbungsregeln

Zu unterscheiden ist zwischen Mehrfachvererbung (allgemeine Hierarchie), d.h. dass eine Rolle von mehreren Rollen erben kann, und der Einfachvererbung (begrenzte Hierarchie), in der eine Rolle eben höchstens von einer Rolle erbt (Siehe Abbildung 6). Im Falle der Einfachvererbung ließe sich nicht eine einzige maximale Rolle bestimmen, sondern für jeden Ast im Baum würde eine maximale Rolle entstehen.

Die formale Beschreibung für RBAC 1 ergänzt die von RBAC 0 um folgendes: Eine Rollenhierarchie wird durch ein Tupel aus zwei Rollen beschrieben:

$$Rollenhierarchie \subseteq Rollen \times Rollen$$

Darauf wird eine Halbordnung $\geq$ definiert. Halbordnungen besitzen die Eigenschaften Reflexivität, Antisymmetrie und Transitivität. Das bedeutet:

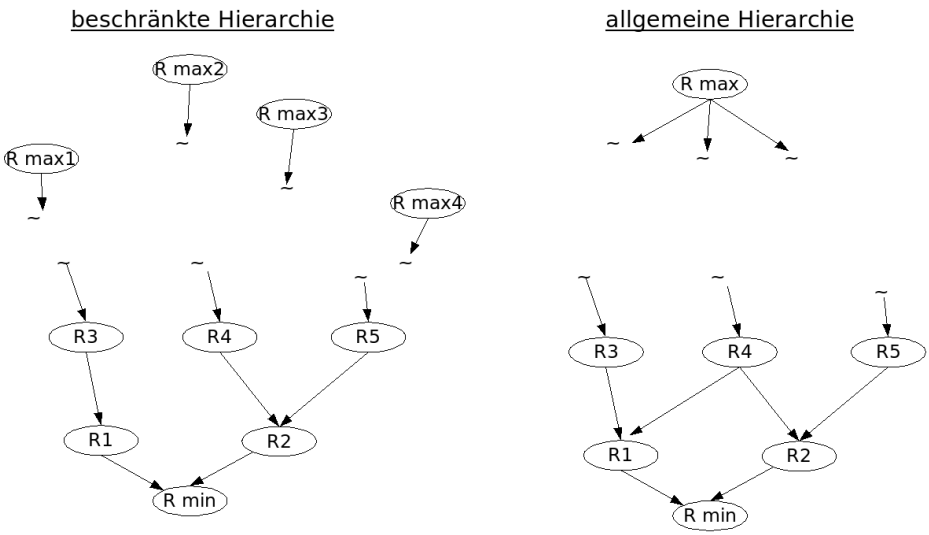

Fig. 6. Hierarchiebäume

- Eine Rolle kann von sich selber erben.
- Erbt Rolle A von Rolle B, so kann Rolle B nicht von A erben, es sei denn es gilt $A = B$.
- Erbt Rolle A von B und Rolle B von C, so erbt auch Rolle A von C.

Diese Funktion gibt nun zusätzlich zur Funktion zugeordneteBenutzer auch diejenigen Benutzer aus, die der Rolle durch Vererbung zugeordnet sind:

$$authorisierteBenutzer : Rollen \rightarrow 2^{Benutzer}\{b \in B | r' \geq r, (b, r) \in Benutzerzuordnung\}$$

Ebenso gibt diese Funktion nun auch diejenigen Rechte aus, die der Rolle durch Vererbung zugeordnet sind:

$$authorisierteRechte : Rollen \rightarrow 2^{Rechte}\{re \in Re | r' \geq r, (re, r) \in Rechtezuordnung\}$$

In der Beispielanwendung soll nun die zuvor neu eingestellte Mitarbeiterin Frau Schneider Lohnabrechnungen drucken und benötigt dazu das Recht zur Lohneinsicht. Durch die hierarchische Rollenverteilung hat man nun die Möglichkeit, eine Rolle Lohn zu erstellen, der dieses Recht zugeordnet wird. Dieser Rolle wird Frau Schmidt zugeordnet. Gleichzeit vererbt die Rolle aber ihr Recht weiter an die Rolle Abteilungsleiter Lohn, der zusätzlich das Recht zur Erstellung der

Lohnabrechnung zugeordnet wird, und an die Rolle Finanzbuchhaltung, die auch das Recht der Lohneinsicht benötigt.

3.3 Einschränkungen (RBAC 2)

Bei dieser Erweiterung des RBAC Kernmodells geht es um die Trennung von konfliktbehafteten Rollen. Darunter versteht man Aufgabenbereiche, die voneinander unabhängig sein sollten, wie z.B. Kassenführung / Kassenprüfung oder Produktion / Qualitätskontrolle. Hierbei unterscheidet man zwischen statischen und dynamischen Einschränkungen. Im statischen Fall verbietet man es generell, dass ein Benutzer zwei Konfliktbehafteten Rollen zugeordnet wird. Im dynamischen Fall hingegen lässt man die Zuordnung zu den Rollen zu, man verbietet aber die gleichzeitige Annahme der Rollen innerhalb einer Session (Siehe Abbildung 7).

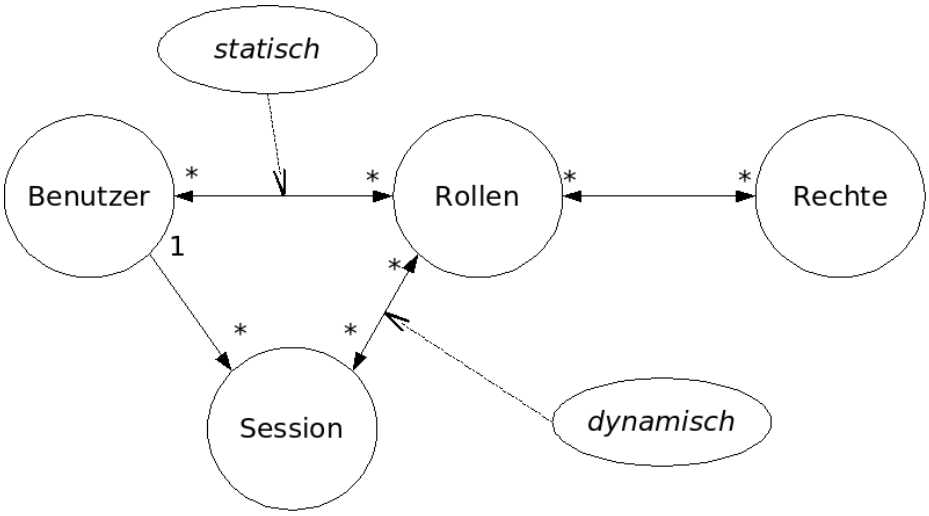

Fig. 7. RBAC 2

Formal lassen sich statische Einschränkungen durch folgendes Konstrukt beschreiben: Eine statische Einschränkung (im folgenden SSD von Static Separation of Duty genannt) wird durch ein Tupel bestehend aus einer Teilmenge der Rollen und einer natürlichen Zahl $n \geq 2$ beschrieben:

$$SSD \subseteq (2^{Rollen} \times N)$$

Kein Benutzer darf dann n oder mehr Rollen aus dieser Teilmenge zugeordnet

werden:

$$\forall (rs, n) \in SSD, \forall t \subseteq rs : |t| \geq n \Rightarrow \bigcap_{r \in t} zugeordneteBenutzer(r) = \emptyset$$

Im Fall der dynamischen Einschränkungen DSD (steht entsprechend für Dynamic Separation of Duty) wird die Schranke n nicht über die Benutzerzuordnung überprüft, sondern über die Anzahl der innerhalb einer Session aktivierten Rollen:

$$t \subseteq rs, t \subseteq SessionRollen(s) \Rightarrow |t| < n$$

In der Beispielanwendung sollen nun verhindert werden, dass Mitarbeiter der Finanzbuchhaltung die firmeninterne Bilanzprüfung (Rolle Bilanzprüfung mit dem Recht zur Bilanzeinsicht) durchführen. Dafür muss nun eine statische Einschränkungen definiert. Diese verbietet es zum einen, falls in Benutzer der Rolle Finanzbuchhaltung zugeordnet ist, auch der Rolle Bilanzprüfung zugeordnet zu werden und zum anderen, falls ein Benutzer der Rolle Bilanzprüfung zugeordnet ist auch noch der Rolle Finanzbuchhaltung zugeordnet zu werden. Eine dynamische Einschränkung würde in diesem Fall nicht reichen, da diese Einschränkung ja für die Benutzerzuordnung gelten soll und nicht nur innerhalb einer Session.

3.4 Vereinigung von RABC 1 und 2 (RABC 3)

RBAC 1 und 2 sind in der Grundform nicht kompatibel [4], da z.B. durch Mehrfachvererbung in RBAC 1 Beschränkungen aus RBAC 2 umgangen werden können. Erbt eine Rolle von zwei anderen Rollen, die durch Einschränkungen getrennt sind, so würden in dieser übergeordneten Rolle ja die konfliktbehafteten Rechte vereint und damit die Einschränkung außer Kraft gesetzt. Damit die beiden RBAC Versionen 1 und 2 vereint werden können, werden die Einschränkungen aus RBAC 2 nicht mehr über die RBAC 0 - Funktion zugeordneteBenutzer überprüft, sondern über die aus RBAC1 vorhandene Funktion authorisierteBenutzer. Zur Erinnerung; diese Funktion liefert auch diejenigen Benutzer zu einer Rolle, die ihr durch Vererbung zugeordnet sind. Der Schnitt darüber muss dann wieder leer sein.

RBAC 3 enthält keine zusätzlichen Funktionen gegenüber RBAC 1 und 2, stellt aber durch die Vereinigung der beiden Kernerweiterungen ein sehr mächtiges Konstrukt zur Benutzerverwaltung dar.

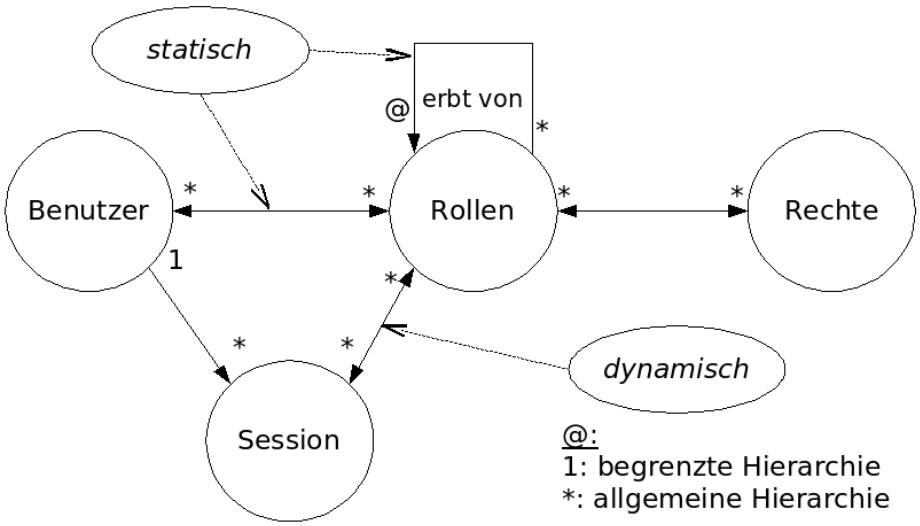

Fig. 8. RBAC 3

4 GTRBAC - Eine Erweiterung

Eine auf RBAC 3 basierende Erweiterung stellt Generalized Temporal Role Based Access Control da [2]. Diese Erweiterung bietet zusätzlich die Möglichkeit zeitliche Bedingungen zu formulieren. Eine solche Bedingung wird als Tupel (I,P,D,E) angegeben. Im folgenden sind die einzelnen Komponenten näher erläutert:

- I: gibt ein Intervall an, in dem das Tupel gültig ist
- P: ein sich (unendlich) wiederholender Zeitabschnitt (Periode) innerhalb des Intervalls I
- E: Ein Ereignis, bzw. eine Aktion, die ausgeführt wird
- D: Dauer, für die dieses Ereignis E gültig ist

Um dies nun etwas zu verdeutlichen, betrachten wir ein solches temporales Konstrukt am Fall der Beispielanwendung. Ein BWL Student arbeitet nebenher im Bereich der Finanzbuchhaltung. Damit dies aber nicht über die studentische Tätigkeit hinaus geht, werden gewisse Einschränkungen formuliert:

- I: [1.1.2007 - 31.12.2007]: Der Student ist während des Jahres 2007 bei der Firma beschäftigt
- P: Mo,Mi,Fr: Der Student kann/darf nur an diesen Wochentagen arbeiten

- D: 4h: Pro Tag darf der Student maximal 4 Stunden arbeiten
- E: Studentische Hilfskraft: Der Benutzer nimmt dann die Rolle Studentische Hilfskraft an

5 Anwendung für adaptive Workflowsysteme

Ein Bereich, in dem RBAC eingesetzt wird, sind so genannte adaptive Workflowsysteme, d.h. Workflowsysteme, die einer ständigen Anpassung unterliegen. Damit dies einfacher zu managen ist, unterteilt man das System in 3 Schichten:

- Planungs-/Ziel Ebene
- Schema Ebene
- Instanz Ebene

Man kann nun das RBAC Modell auf diese 3-Schichten Architektur abbilden. Man verwendet im Zusammenhang mit Workflows allerdings ein erweitertes Modell, genannt Task Based Authorization Controls [5]. Hierbei ist es unter anderem möglich, Rollen Tasks (Aufgaben) zuzuordnen und Einschränkungen abhängig von der jeweils ausgeführten Task zu definieren. Man stellt man die Komponenten wie in folgender Tabelle gegenüber.

Globale Einschränkungen	Planungs-/Ziel Ebene
Benutzer-Rollen Zuordnung	Schema Ebene
Rollen-Task/Rechte Zuordnung	Instanz Ebene

Gibt es nun im Workflow eine Änderung auf der höchsten Ebene, so müssen im schlimmsten Fall auf RBAC Seite die globalen Einschränkungen geändert werden. Natürlich können dadurch auch Änderungen in den beiden darunter liegenden Schichten notwendig sein. Erfolgt eine Änderung auf der Schema Ebene des Workflows, so hat dies maximal Auswirkungen auf die Benutzer-Rollen Zuordnung, die globale Einschränkungen sind dadurch nicht betroffen. Wird nun der Workflow auf Instanzebene geändert, so ist auf RBAC Seite lediglich eine Anpassung bei der Rollen-Task/Rechte Zuordnung nötig [1].
Als Beispiel betrachten wir den Ablauf bei Annahme eines Bauauftrages. Dazu wird die Beispielanwendung etwas verfeinert. Frau Müller soll der Rolle Abteilungsleiter Bauauftrag zugeordnet sein, die beiden Mitarbeiter Schuster und Schumann der Rolle Mitarbeiter Bauauftrag. Der Workflow ist in Abbildung 9 dargestellt. Initiiert wird er durch die Erfassung eines Auftrags. Dies geschieht durch einen Mitarbeiter. Im nächsten Schritt werden die Kosten kalkuliert und ein Angebot erstellt. Auch dieser Schritt kann von einem Mitarbeiter der Bauauftragsabteilung durchgeführt werden. Bevor das Angebot aber abgegeben wird, muss eine Prüfung

durch den Abteilungsleiter erfolgen (Schritt 3). Der letzte Schritt, Übermittlung des Angebots an den Kunden, kann dann wieder von einem Mitarbeiter durchgeführt werden. Nun kann der Fall eintreten, dass die Abteilungsleiterin Frau Müller in Urlaub ist und daher der normale Ablauf des Workflows nicht möglich ist, es sei denn, man wartet, bis Frau Müller wieder da ist. Möchte man das nicht, so gibt es für das Workflowmanagementsystem entweder die Möglichkeit den Workflow anders zu definieren, d.h. die Prüfung auszulassen oder aber man erlaubt einem Mitarbeiter die Prüfung des Angebots. Hier soll nun nach der zweiten Möglichkeit verfahren werden. Dazu muss dann der Mitarbeiterrolle das entsprechende Recht zur Prüfung des Angebots zugeordnet werden.
Das TBAC-Modell erlaubt darüber hinaus die Einschränkung zu definieren, dass Schritt 3, die Prüfung des Angebots, nur von einem Mitarbeiter durchgeführt werden kann, der nicht Schritt 2, die Erstellung, ausgeführt hat. Beim klassischen RBAC könnte man diese Trennung nur auf der Ebene der Benutzer-Rollen Zuordnung erreichen, da nur hier die Einschränkungen möglich sind. Man müsste also den Benutzern Schuster und Schumann ermöglichen, die Rolle des Abteilungsleiters anzunehmen. Dies wäre aber ein sehr viel stärkerer Eingriff in die Rechtekonzeption, als nur den Rollen die Task Angebotsprüfung, bzw. die entsprechenden Rechte dafür, zuzuordnen.

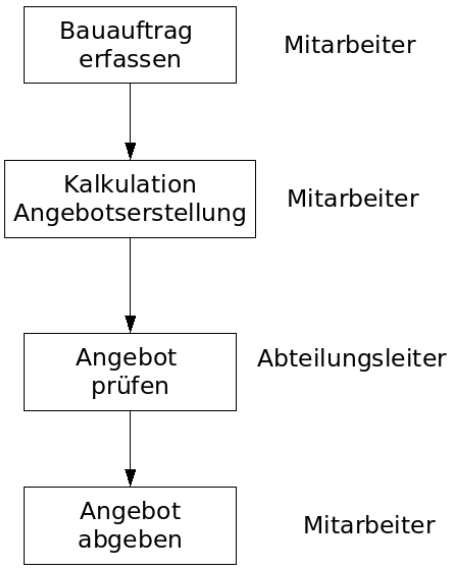

Fig. 9. Workflow für die Auftragserfassung

6 Realität

Mittlerweile gibt es auch Softwareprodukte, die den RBAC Standard implementiert haben. Im Nachfolgenden sind ein paar Beispiele bekannter Hersteller genannt, die das Konzept der Rollenbasierten Zugriffskontrolle verwenden.

6.1 IBM Websphere

Die auf J2EE basierende Produktlinie zur Anwendungsintegration verwendet RBAC. Die Benutzerdatenbank basiert auf LDAP. [6]

6.2 Oracle

Der Application Server von Oracle nutzt das Java RBAC Modul mit Namen Java Authentication and Authorization Service (JAAS). Dieses Modul stellt auch Rollenhierarchien zur Verfügung, also eine Implementierung von RBAC 1. [8]

6.3 Microsoft Windows Server 2003

In Windows Server 2003 wird eine neue Autorisierungsschnittstelle mit dem Namen **Autorisierungs-Manager** verwendet. Diese Schnittstelle verwendet eine rollenbasierte Anwendungsautorisierung. [7]

7 Fazit

RBAC stellt ein mächtiges Konzept da, eine große Anzahl von Benutzern zu verwalten und dabei dem Grundsatz Folge zu tragen, immer nur die minimal notwendigen Rechte zur Ausübung einer Aufgabe zu vergeben. Durch die Flexibilität der Rechtevergabe, die man durch das Rollenkonzept erreicht eignet sich RBAC gut als Authorisierungssystem für Systeme mit einem hohen Grad an Veränderung, wie z.B. adaptive Workflowsysteme. Allerdings werden hier meist Erweiterungen verwendet, die RBAC noch besser die Architektur von Workflowsystemem integrieren. Nicht so geeignet ist der Einsatz von RBAC aus meiner Sicht bei Systemem mit wenigen Benutzern, da hier der Aufwand der Rollendefinition wohl schnell größer werden kann, als der Nutzen den man bei

der Anwendung anschließend hat. Auch die vorangegangenen Beispiele lassen den Schluss zu, dass RBAC hauptsächlich in Systemen, die von sich aus schon einen sehr hohen Komplexitätsgrad haben, eine Vereinfachung bringt.

References

1. Nanjangud C. Narendra: Design of an Integrated Role-Based Access ontrol Infrastructure for Adaptive Workflow System, Journal of Computing and Information Technology - CIT 11,2003,4,293-308
2. Basit Shafiq, Arjmand Samuel and Halima Ghafoor: A GTRBAC Based System for Dynamic Workflow Compositition and Management, IEEE International Symposium on Object-Oriented Real-Time Distributed Computing
3. American National Standard for Information Technology: Role Based Access Control, Draft-4/4/2003, BSR INCITS 359
4. Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein and Charles E. Youman: Role-Based Access Control Models, IEEE Computer, Volume 29, Number 2, February 1996, page 38-47
5. Thomas R., and Sandhu R. S., Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management, IFIP WG11.3, 1997
6. http://www-1.ibm.com/support/docview.wss?uid=swg21219615
7. http://www.microsoft.com/austria/server/miis/bestandteile_loesung.mspx
8. http://download-west.oracle.com/docs/cd/B10464_05/web.904/b10325/jaas_int.htm#1006468