Institut für Rechnergestützte Ingenieursysteme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3476

Benutzerverwaltung und Sicherheitskonzepte im Geschäftsprozessmanagement

Johannes Kessel

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. D. Roller
Betreuer/in:	Dipl.-Inf. Felix Baumann
Beginn am:	11. April 2013
Beendet am:	11. Oktober 2013
CR-Nummer:	D.4.6, H.1.2, H.4.1

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Kurzfassung

Ein großer Teil der Geschäftsprozesse wird heutzutage in IT-gestützten Geschäftsprozessmanagementsystemen *(Business Process Management System (BPMS))* abgebildet. Diese Umsetzung ermöglicht erhöhte Effizienz, Flexibilität, Transparenz und eine bessere Qualität der Prozesse in Unternehmen.
Den geschäftlichen Möglichkeiten des Prozessmanagements stehen allerdings beträchtliche Gefahren bezüglich der Einhaltung von Sicherheitsvorschriften und gesetzlichen Bestimmungen gegenüber. Aus diesem Grund sind bei der automatisierten oder teilweise automatisierten Ausführung der Prozesse Mechanismen notwendig, um die Verletzungen der Sicherheitsrichtlinien zeitnah zu erfassen oder ganz auszuschließen.
Diese Arbeit behandelt im Wesentlichen das große Gebiet der bestehenden Benutzerverwaltungs- und Sicherheitsansätze im Geschäftsprozessmanagement. Im Rahmen dieser Arbeit wird zudem eine Lösung zur automatisierten Verteilung und Verwaltung von Benutzerrollen in bestehenden IT-Systemen untersucht sowie das Sicherheitskonzept des Geschäftsprozessmanagementsystems Activiti analysiert und erweitert.

Kapitel 1

Einleitung

1.1 Themenstellung der Arbeit

Die Autorisierung und die Zugriffskontrolle sind wichtige Bestandteile des Sicherheitsmanagements in Unternehmen. In Anwendungssystemen muss jeder Mitarbeiter bestimmte Berechtigungen besitzen, um seine Tätigkeit verrichten zu können. Betrachtet man das Unternehmen als Ganzes, verteilen sich Berechtigungen von der Geschäftsprozessebene bis hin zu deren technischer Ausprägung in den einzelnen Anwendungssystemen [Klarl 2011, S. 81]. Dabei ist die Administration der Zugriffsrechte in heterogenen, verteilten IT-Systemen großer Unternehmen kein einfacher Prozess. Dieser Administrationsprozess beinhaltet die Verwaltung einer großen Anzahl von Benutzern und deren Zugriffsrechten in der gesamten IT-Landschaft des Unternehmens.
Ein gängiges Zugriffsmodell, das vermehrt in Unternehmen, im Finanzsektor usw. Anwendung findet ist das rollenbasierte Berechtigungskonzept (RBAC) [Ferraiolo 2001]. Der Zugriff auf eine Ressource wird in dem Fall nur dann gewährt, wenn die jeweilige Person beim Zugriff auf ein Anwendungssystem einer speziellen Rolle angehört. Dabei muss darauf geachtet werden, dass jeder Benutzer nach dem Prinzip der geringst möglichen Privilegien *Least Privilege Prinzip* - so wenige Rechte wie möglich, so viele wie nötig, erhält [Ferraiolo und Kuhn 1992].
Obwohl das rollenbasierte Berechtigungskonzept einen vielversprechenden Ansatz für Unternehmen darstellt, ist die technische Umsetzung von diesem Berechtigungskonzept in den meist sehr komplexen IT-Landschaften vieler Unternehmen nicht trivial. In der Literatur existieren zudem verschiedene Auffassungen des Rollenbegriffs. Zum einen als ein Konstrukt für technische Berechtigung *(Systemrolle)*, zum anderen als Konstrukt eines Aufgabenfeldes innerhalb einer Organisation *(Geschäftsrolle)* [Klarl 2011, S. 81-87] [Wortmann und Winter 2007, S. 439-447].
Die Auffassung mit der *Systemrolle* betrachtet die eingesetzte IT-Landschaft und darin eingesetzte Zugriffsrechte. Hier stellt die *Systemrolle* eine Menge von Zugriffsrechten auf ein oder mehrere IT-Systeme dar.
Die andere Sicht mit *Geschäftsrollen* abstrahiert im Gegensatz dazu von den technischen Details und bezieht sich auf die Organisationsstruktur oder die Aufgabenbereiche innerhalb eines Unternehmens. Die *Geschäftsrollen* werden beispielsweise innerhalb der Modellierung von Geschäftsprozessen verwendet [Molitorisz 2008].
Die Komplexität des rollenbasierten Berechtigungskonzeptes in Unternehmen hängt in erster

Linie damit zusammen, dass die IT-Systeme in Unternehmen meist aus einer großen Anzahl von heterogenen Systemen besteht. Das führt zusätzlich zur unterschiedlichen Granularität der Zugriffsrechte in diesen Systemen. Aus der Unternehmens- oder Organisationssicht stellt sich das Problem so dar, dass man mit vielen unterschiedlichen Abteilungen und Geschäftsbereichen mit nicht ganz klaren und sich überschneidenden Strukturen zu tun hat [Wortmann und Winter 2007].
Diese Arbeit beschäftigt sich ausgiebig mit der Benutzerverwaltung und Sicherheit im Geschäftsprozessmanagement. Eine wichtige Fragestellung dabei ist die technische Umsetzung und Verwendung des rollenbasierten Berechtigungskonzeptes in Anbetracht der oben genannten Problematik. Danach wird eine Lösung zur automatisierten Verteilung und Verwaltung von Benutzerrollen untersucht. Dazu wurde eine Literaturrecherche betrieben und die bestehenden und möglichen Ansätze der Verwaltung von Rollen diskutiert. Im letzten Teil der Arbeit wird das Sicherheits- und Benutzerkonzept des Prozessmanagementsystems Activiti analysiert und erweitert.

1.2 Gliederung

Kapitel 1 – Einleitung umfasst eine kurze Einführung in das Themengebiet sowie die Aufgabenstellung der vorliegenden Arbeit.

Kapitel 2 – Grundlagen stellt ausführlich die theoretischen Grundlagen der Arbeit vor. Es werden verschiedene Autorisierungsstrategien, rollenbasierte Zugriffsmodelle sowie die grundlegenden Sicherheitsanforderungen im Geschäftsprozessmanagement beschrieben.

Kapitel 3 – Literaturübersicht erläutert die vorhandenen Ansätze zur Abbildung von Zugriffskontrol- und Sicherheitsinformationen während der Geschäftsprozessmodellierung. Danach wird die mögliche Umsetzung der automatischen Verwaltung und Verteilung von Benutzerrollen untersucht.

Kapitel 4 – Activiti als Workflow- und BPM-Plattform stellt eine Geschäftsprozessmanagement-Plattform namens Activiti vor. Nach der Untersuchung und Bewertung der Sicherheitsfunktionen wird das Sicherheitskonzept von Activiti erweitert. Anschließend wird eine Möglichekeit der Anbindung von Activiti an ein Verzeichnisdienst beschrieben.

Kapitel 5 – Zusammenfassung und Ausblick fasst die Ergebnisse der Arbeit zusammen und gibt einen Ausblick auf zukünftige Arbeiten in diesem Themengebiet.

Kapitel 2

Grundlagen

2.1 Kriterien zur Bewertung von Sicherheit in IT-Systemen

Um die Sicherheit informationstechnischer Systeme methodisch zu bewerten, wurden bereits nationale und internationale Bewertungskriterien entwickelt. Die Bewertungskriterien stellen ein Bewertungssystem zur Verfügung, das es erlaubt unterschiedliche Sicherheitssysteme mit einer ähnlichen Funktionalität zu vergleichen. Die Kriterien definieren dabei eine Art Vorgabe und dienen als Leitlinien für die Entwicklung der sichereren IT-Systeme.
Die ersten Kriterienkataloge dieser Art waren die *Trusted Computer System Evaluation Criteria (1983)* des amerikanischen Verteidigungsministeriums. Einige Jahre später führten Deutschland, Frankreich, die Niederlande und Großbritannien einen gemeinsamen Standard ein, welcher dann in einen internationalen *Common Criteria-Standard* aufgegangen ist [Eckert 2009].
Die Bewertung und die Evaluierung der Sicherheit informationstechnischer Systeme geschieht durch unabhängige Zertifizierungsstellen, welche einen amtlich bestätigten Nachweis (Zertifikat) ausstellen und dem Benutzer eines Produktes ein bestimmtes Maß an durch die Kriterien erfüllten Sicherheitsanforderungen attestieren. Der Nachweis, dass das Produkt die im Zertifikat definierte Sicherheitsleistung erfüllt, bietet auch dem Hersteller des Produktes bessere Chancen auf dem Markt.
Die Kriterien sind für die Bewertung der Sicherheitseigenschaften praktisch aller informationstechnischen Produkte, wie z. B. Betriebssysteme, Datenbanken, Firewalls, PC-Sicherheitsprodukte, usw. geeignet. Folgende Sicherheitskriterien oder Grundfunktionen der Sicherheitsarchitektur unterliegen einer getrennten Bewertung durch die Zertifizierungsstellen [Criteria 2012]:

- **FAU: Security Audit (Sicherheitsprotokollierung)**
  Die Sicherheitsprotokollierung befasst sich mit der Aufzeichnung und Analyse von Informationen im Betrieb eines Programms. Daraus ist z. B. ersichtlich, welche Benutzer für die Ausführung einer bestimmten Programmaktivität verantwortlich sind.
- **FCO: Communication (Kommunikation)**
  Diese Funktionsklasse stellt sicher, dass die Kommunikationspartner davon ausgehen können, dass die Sicherstellung der Identität (z. B. durch digitale Signaturen) der am Datenaustausch beteiligten Partner garantiert ist.

- ◇ **FCS: Cryptographic Support (kryptographische Unterstützung)**
  Diese Funktionalität befasst sich mit kryptografischen Komponenten des Programms, wie z. B. mit der verschlüsselten Übertragung der Daten.
- ◇ **FDP: User Data Protection (Schutz der Benutzerdaten)**
  Diese Funktionsklasse dient dem Schutz beim Import, Export und Lagerung der Benutzerdaten. Dazu gehören spezielle Sicherheitsvorschriften, Zugriffskontrollen oder eine Informationsflusskontrolle.
- ◇ **FIA: Identification and Authentication (Identifikation und Authentifizierung)**
  Hier steht die Einrichtung, Bestimmung und die Verifizierung der Benutzeridentität im Vordergrund. Dabei werden z. B. die Möglichkeiten der Zuordnung der Benutzer zu Gruppen oder Rollen untersucht.
- ◇ **FMT: Security Management (Sicherheitsmanagement)**
  Diese Funktionsklasse befasst sich mit der Festlegung unterschiedlicher Managementrollen, aber auch mit ihrer Interaktion wie: Trennung der Berechtigungen.
- ◇ **FPR: Privacy (Privatsphäre)**
  Behandelt die Anforderungen an die Privatsphäre, welche dem Schutz der Nutzer gegen Enthüllung und Missbrauch ihrer Identität dienen. Diese Klasse bietet zusammen mit den Klassen wie FAU, FTP und FDP die nötige Flexibilität um das gewünschte Verhalten im Bezug auf die Privatsphäre zu spezifizieren.
- ◇ **FPT: Protection (Schutz der Sicherheitsfunktionen)**
  Enthält die Anforderungen an die Integrität der Sicherheitsfunktionen, sowie an die Integrität von Daten der Sicherheitsfunktionen.
- ◇ **FRU: Resource Utilisation (Betriebsmittelnutzung)**
  Diese Funktionsklasse besteht aus drei Klassenkomponenten, welche z. B. den Schutz gegen Nichtverfügbarkeit bei Fehlern und die Betriebsmittelzuteilung garantieren.
- ◇ **FTP: Trusted Path/Channels (vertrauenswürdiger Pfad/Kanal)**
  Hierbei werden Anforderungen an einen vertrauenswürdigen Kommunikationspfad zwischen Benutzern und Sicherheitsfunktionen formuliert.

Das Ziel der Zertifizierung ist es dann die Sicherheitsleistung des IT-Produktes in Verbindung mit den abzuwehrenden Bedrohungen zu beschreiben und in einer Bewertung, die eine Angabe darüber macht, wie gut die Sicherheitsfunktionen sich diesen Bedrohungen widersetzen.

2.2 Die Anforderungen der Sicherheit im Geschäftsprozessmanagement

Die Anforderungen der Sicherheit im Geschäftsprozessmanagement vereinen die oben genannten Sicherheitseigenschaften mit speziellen regulatorischen Auflagen und organisationellen Richtlinien und bilden somit folgende Anforderungsklassen [Accorsi 2013]:

- **Autorisierung**
  Autorisierung regelt welche Benutzer, Programme oder Rollen in einem Prozess auf bestimmte Ressourcen zugreifen dürfen. In einem Unternehmensumfeld wird es meistens mit Hilfe von rollenbasierten Zugriffskontrollen ermöglicht. Nicht selten werden dabei die rollenbasierten Zugriffsbeschränkungen zusätzlich modifiziert und um *Authorisation Constraints* ergänzt [Bertino, Ferrari und Atluri 1999]. Die *Authorisation Constraints* können die Zugriffsrechte auf eine Ressource durch persönliche Attribute der Benutzer wie Alter, Beruf oder die Zugriffszeit zusätzlich beschränken.
- **Nutzungskontrolle**
  Die Nutzungskontrolle drückt bestimmte Regeln aus, um nach der Autorisierung die Nutzung eine Ressource zu regeln. Die Nutzung eine Ressource oder Aktivität ist z. B. nur in einem bestimmten Zeitintervall möglich oder es ist nur eine bestimmte, vordefinierte Anzahl der Zugriffe auf eine Ressource erlaubt.
- **Interessenkonflikt**
  Interessenskonflikte entstehen z. B. bei der unzulässigen Ausnutzung von Insiderwissen bei der Abwicklung von Banktransaktionen oder bei der gleichzeitigen Beratung unterschiedlicher Unternehmen. Die Anforderungen werden dabei anhand des *Chinese-Wall* Modells erfasst. Die Idee des Chinese-Wall Modells basiert darauf, dass die zukünftigen Zugriffsmöglichkeiten einer Person durch die Zugriffe in der Vergangenheit beschränkt werden. Die Zulässigkeit der Zugriffe auf bestimmte Ressourcen hängt somit von der Zugriffshistorie ab.
- **Funktionstrennung (Separation of Duty)**
  Funktionstrennung bedeutet, dass bestimmte Aktivitäten innerhalb eines Geschäftsprozesses nicht von einer Person oder Organisationseinheit durchgeführt werden dürfen. Die Abteilungen oder Personen, welche z. B. Finanzmittel erhalten oder Zugang zu Buchhaltungsdaten besitzen, sollten die Funktionstrennung beachten, um Betrug vorzubeugen [Botha und Eloff 2001].
- **Aufgabenbindung (Binding of Duty)**
  Mit der Aufgabenbindung ist das Gegenteil der Funktionstrennung gemeint. Manche Aufgaben oder Tätigkeiten im Geschäftsprozess sollen dabei nur von einer bestimmten Organisationseinheit oder Person durchgeführt werden. Das kann z. B. wegen den speziellen fachlichen Fähigkeiten oder aus Datenschutzgründen geschehen.

- ◇ **Isolation**
  Isolation bedeutet, dass zwischen Prozessen innerhalb oder außerhalb der Unternehmen keine Informations- oder Datenlecks entstehen dürfen um die Datenintegrität und Vertraulichkeit zu gewährleisten.

Diese Anforderungen können auf verschiedene Art und Weise während des Prozessablaufs überprüft werden. Auf die vielseitigen Implementierungsmöglichkeiten wird im weiteren Verlauf der Arbeit ausführlich eingegangen.

2.3 Sicherheitsmodelle

Ein hoher Qualitätsstandard bei der Entwicklung von sicheren Systemen ist nur dann erreichbar, wenn man von einer genauen, d.h. formalen Beschreibung der Sicherheitseigenschaften der zu entwickelten Anwendung ausgeht und ein entsprechendes Sicherheitsmodell anfertigt. Ein Modell beschreibt dabei die Eigenschaften eines realen Systems mit einer hinreichenden Abstraktion davon. Als nächstes werden die bekannten Sicherheitsmodelle im Hinblick auf die durch sie beschreibbaren Eigenschaften vorgestellt [Eckert 2009].

2.3.1 Bestandteile eines Sicherheitsmodells

Für jeden Zugriff innerhalb eines Anwendungssystems bedarf es einer Zugriffsentscheidung im Rahmen eines Autorisierungsvorganges. Bei der Modellierung der Zugriffskontrolle muss deshalb sichergestellt werden, dass *Subjekte* einen berechtigten Zugriff auf *Objekte* besitzen.

Subjekt
Als Subjekte werden aktive Einheiten (Benutzer, Programme, Prozesse) verstanden, welche auf bestimmte Ressourcen innerhalb eines Prozesses Zugriffe anfordern (Abb. 2.1). Eine grobgranulare Modellierung der Subjekte bedeutet, dass die Zugriffe der einzelnen Benutzer nicht explizit kontrolliert werden können. Das ist z. B. dann der Fall, wenn die Benutzer ausschließlich zu einer Gruppe oder Rolle zusammengefasst werden. Eine feingranulare Subjektmodellierung ermöglicht es dagegen benutzerspezifische Zugriffsrechte zu vergeben.

Objekt
Die zu schützenden Einheiten werden bei der Modellierung als Objekte (Dateien, Aktivitäten) bezeichnet. Dabei ist die Granularität der Objekte ebenfalls von großer Bedeutung. Die zu grobkörnige Festlegung der Objekte könnte eventuell viel zu umfassende Rechte auf sicherheitsrelevante Daten gewähren. Eine grobe Granularität kann auch zusätzlich dazu führen, dass zu viele Ressourcen in einem Objekt zusammengefasst und die einzelnen, kritischen Elemente im Nachhinein nicht ohne Weiteres oder mit großem Aufwand getrennt geschützt werden müssen.

Eine feingranulare Objektmodellierung ist z. B. bei der Umsetzung des „Kenntnis nur bei Bedarf“ - Prinzips *(need-to-know Principle)* erforderlich. Es ermöglicht die zu schützenden Objekte anwendungsspezifisch festzulegen und bestimmten Subjekten nur diejenigen Rechte zu gewähren, welche für die Ausführung der jeweiligen Aufgabe an einem Objekt nötig sind. Nachteil der feingranularen Modellierung ist allerdings der größere Verwaltungsaufwand bei der jeweiligen Rechtevergabe.

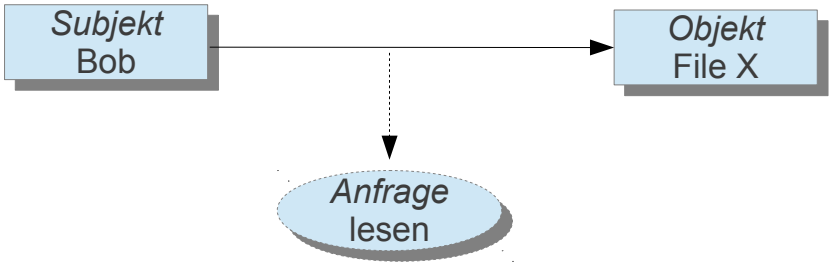

Abbildung 2.1: Subjekt-Objekt Interaktion

2.3.2 Zugriffsrechte und Zugriffsbeschränkungen

Die Zugriffsrechte können unterschiedliche Ausprägungen haben. Man spricht z. B. vom *universellen* und *objektspezifischen* Recht. Die Zugriffsrechte sind dann *universell*, wenn sie sich nicht auf ein bestimmtes Objekt, sondern auf allgemeine Systemoperationen beziehen. Bekannte universellen Rechte sind z. B. die *write-* und *read-* Rechte für Dateien in einem Betriebssystem [Eckert 2009].
Die *objektspezifischen* Rechte beziehen sich im Vergleich zu *universellen* Rechten auf ein bestimmtes Objekt. Die Vergabe von Rechten auf ein Objekt ist dabei abhängig von den spezifischen Aufgaben des Subjekts, welcher auf die Objekte zugreift.
Die Zugriffsbeschränkungen können ebenfalls unterschiedliche Ausprägungen haben. Man unterscheidet zwischen *einfachen* und *komplexen* Zugriffsbeschränkungen.
Eine Zugriffsbeschränkung ist *einfach*, wenn ein Subjekt auf ein Objekt ohne zusätzliche Nebenbedingungen (Constraints) zugreifen kann (Abb. 2.1). Die zusätzlichen Nebenbedingungen können sich z. B. auf die Zugriffszeit, persönlichen Attribute oder persönlichen Fähigkeiten des Subjekts beziehen.
Ein Beispiel für eine *komplexe* Zugriffsbeschränkung in (Abb. 2.2) zeigt eine Modellierung von einem e-Commerce-System in UML[1] *(Unified Modeling Language)* zum Verleih von Videofilmen. Dabei besitzt der Kunde die Rolle **Customer** (Kunde) und ein Attribut **age** (Alter). Der Kunde darf einen Film nur dann ausleihen, wenn sein Alter > (größer) oder = (gleich) der Altersbeschränkung des Filmes ist. Diese Zugriffsbeschränkung wird dabei als Bedingung formuliert [Priebe und Dobmeier 2005].

[1] `http://www.uml.org/`

Die meisten bekannten Betriebssysteme wie Windows oder Linux verfügen generell über einfache Zugriffsbeschränkungen. Die komplexen Zugriffsbeschränkungen werden meist unabhängig von den Betriebssystemen, anwendungsspezifisch realisiert [Eckert 2009].

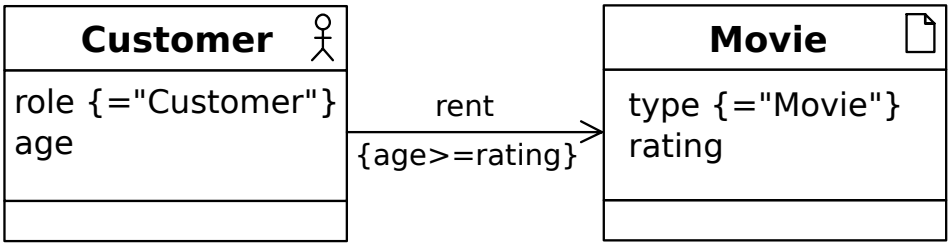

Abbildung 2.2: Subjekt-Objekt Interaktion mit Nebenbedingung [Priebe und Dobmeier 2005]

2.3.3 Autorisierungsstrategien

Autorisierung oder Zugriffskontrolle bedeutet die Kontrolle und Verwaltung von Zugriffsrechten [Rupprecht und Wortmann 2006]. Die Kontrolle von Zugriffsrechten ist als ein Vorgang der Vermittlung von Anfragen an Ressourcen eines Systems definiert und als Entscheidung, ob die Anfragen zugelassen oder abgelehnt werden. Im Laufe der Zeit wurden unterschiedliche Sicherheitsstrategien entwickelt und können in drei Zugriffskontrollstrategien eingeteilt werden [Fischer-Hübner 2001].

Benutzerbestimmbare Zugriffskontrolle (DAC)

Die benutzerbestimmbaren Zugriffskontrollstrategien (*Discretionary Access Control (DAC)*) basieren auf dem Eigentümer- oder Inhaber-Prinzip und haben ihren Ursprung in der akademischen und industriellen Forschung [Schmaltz 2005]. Für die Rechtevergabe auf Objekte ist nur der Eigentümer (*owner*) dieser Objekte verantwortlich. Der Eigentümer kann seine Nutzungsrechte auch an andere Subjekte im System übertragen. Die Tatsache, dass bei der Vergabe der Rechte die eventuellen Abhängigkeiten zwischen mehreren Objekten nicht beachtet werden, besteht die Gefahr, dass inkonsistente Zustände auftreten können. Eine inkonsistente Rechtevergabe liegt z. B. vor, wenn ein Subjekt durch eine implizite Berechtigung die Rechte erhält, welche er bei einer anderen, expliziten Berechtigung eines anderen Benutzers nicht besitzt [Eckert 2009]. Aus diesem Grund kann es zu einer teilweise unerwünschten Rechtevergabe kommen.

Die systembestimmte Zugriffsontrolle (MAC)

Die systembestimmte Zugriffskontrolle (*Mandatory Access Control (MAC)*) beschreibt eine globale, systemweite und auf Regeln basierende Zugriffskontrollstrategie. Jede Datei, bzw.

jedes Objekt erhält dabei ein *Security-Label*. Durch ein *Security-Label* werden zusätzliche, durch das System festgelegte Beschränkungen auf Objekte festgelegt.

Innerhalb der MAC kann DAC zum Einsatz kommen, wobei die MAC über die Regeln der DAC dominiert. In DAC kann ein Subjekt seine Rechte ohne Weiteres an ein anderes Subjekt weiterreichen. In MAC-Systemen ist es ohne einer zusätzlichen Erlaubnis durch das System nicht gestattet. Eine wichtige Komponente von MAC ist die systemweite Policy, welche die erlaubten Zugriffe zwischen den Objekten und Subjekten innerhalb des Systems regelt [Wortmann und Winter 2007].

Rollenbasierte Zugriffskontrolle (RBAC)

Im Bereich der Sicherheit von Geschäftsprozessen spielt die rollenbasierte Zugriffskontrolle eine wesentliche Rolle. Im Vergleich zu den oben genannten Modellen wie DAC und MAC, in welchen eher die Subjekte im Mittelpunkt stehen, beschäftigen sich die rollenbasierte Modelle hauptsächlich mit durchzuführenden Aufgaben (*Tasks*) innerhalb eines Unternehmens. Die Zugriffsrechte werden somit auf die Tätigkeiten oder Funktionen der Subjekte im Unternehmen zugeschnitten. Durch die Rollenzuweisung bekommen die Subjekte letztendlich die nötigen Rechte für die Ausführung ihrer Aufgaben [Ferraiolo und Kuhn 1992].

Einer Rolle können jederzeit mehrere Benutzer (*User*) und Berechtigungen (*Permissions*) zugewiesen werden (Abb. 2.3). Für jeden Benutzer bedeutet es, dass er alle vorhandenen Berechtigungen innerhalb dieser Rolle nutzen kann. Die Übertragung der Rechte durch die Nutzer ist jedoch im Vergleich zu MAC und DAC nicht möglich. Dies ergibt einen großen Vorteil bei der Umsetzung der Sicherheitspolitik, da die Berechtigungen nicht für jeden Benutzer getrennt verwaltet werden müssen. Dadurch wird auch der administrative Aufwand und die Fehlergefahr deutlich minimiert [Schmaltz 2005] [Eckert 2009].

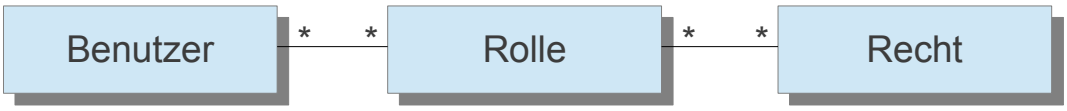

Abbildung 2.3: Rollenkonzept, angelehnt an [Wiesner 2008]

2.3.4 Gegenüberstellung der Autorisierungsstrategien

Die abschließende Bewertung der drei Zugriffskontrollstrategien zeigt, dass die Strategien sich zum Teil grundlegend unterscheiden (siehe Abb. 2.4) und abhängig vom Einsatzzweck in verschiedenen Gebieten Anwendung finden.

Die DAC-Kontrolle eignet sich z. B. für die benutzergesteuerte Kontrolle am besten. Die Benutzer können flexibel und feingranular die Zugriffsrechte auf Objekte definieren, anpassen oder weitergeben. Die Verwaltung der Zugangskontrolle mit *Access Control List (ACL)*, ist z. B. eine mögliche Umsetzung der DAC-Strategie.
Da es aber in DAC keine zentrale Kontrollinstanz zur Prüfung der Konsistenz der Rechtevergabe existiert, kann es zu inkonsistenten Rechtezuweisungen kommen. Aus diesem Grund ist DAC in sicherheitskritischen Systemlandschaften nicht anzutreffen.
In MAC wird die Vergabe und Weitergabe der Rechte im Vergleich zu DAC zusätzlich durch eine systemglobale Instanz kontrolliert. Die starre Zuweisung der Zugriffsregeln durch eine Instanz bietet aber wenig Flexibilität und wird aufgrund dessen hauptsächlich in sehr restriktiven Bereichen, wie z. B. dem Militär, verwendet.
RBAC verwendet Rollen als ein Konstrukt der Vereinigung von Benutzern und Benutzerberechtigungen. Die Rollen vereinfachen und verbessern dabei die Verwaltung und Steuerung der Zugriffskontrolle, da sie meistens die Organisationsstrukturen innerhalb eines Unternehmens abbilden und damit zur besseren Übersicht der Zugangsberechtigungen beitragen.
Die Tabelle (Tab. 2.1) fasst die wesentlichen Merkmale der drei oben genannten Zugriffskontrollstrategien zusammen.

Bewertungskriterium	**DAC**	**MAC**	**RBAC**
Dezentrale Rechteverwaltung	✓		
Flexibilität	✓		✓
Übersichtlichkeit			✓
Weitergabe der Rechte durch Besitzer	✓		
Sicherheitskritische Systeme		✓	✓
Einsatz in Unternehmen	✓		✓

Tabelle 2.1: Gegenüberstellung DAC, MAC, RBAC

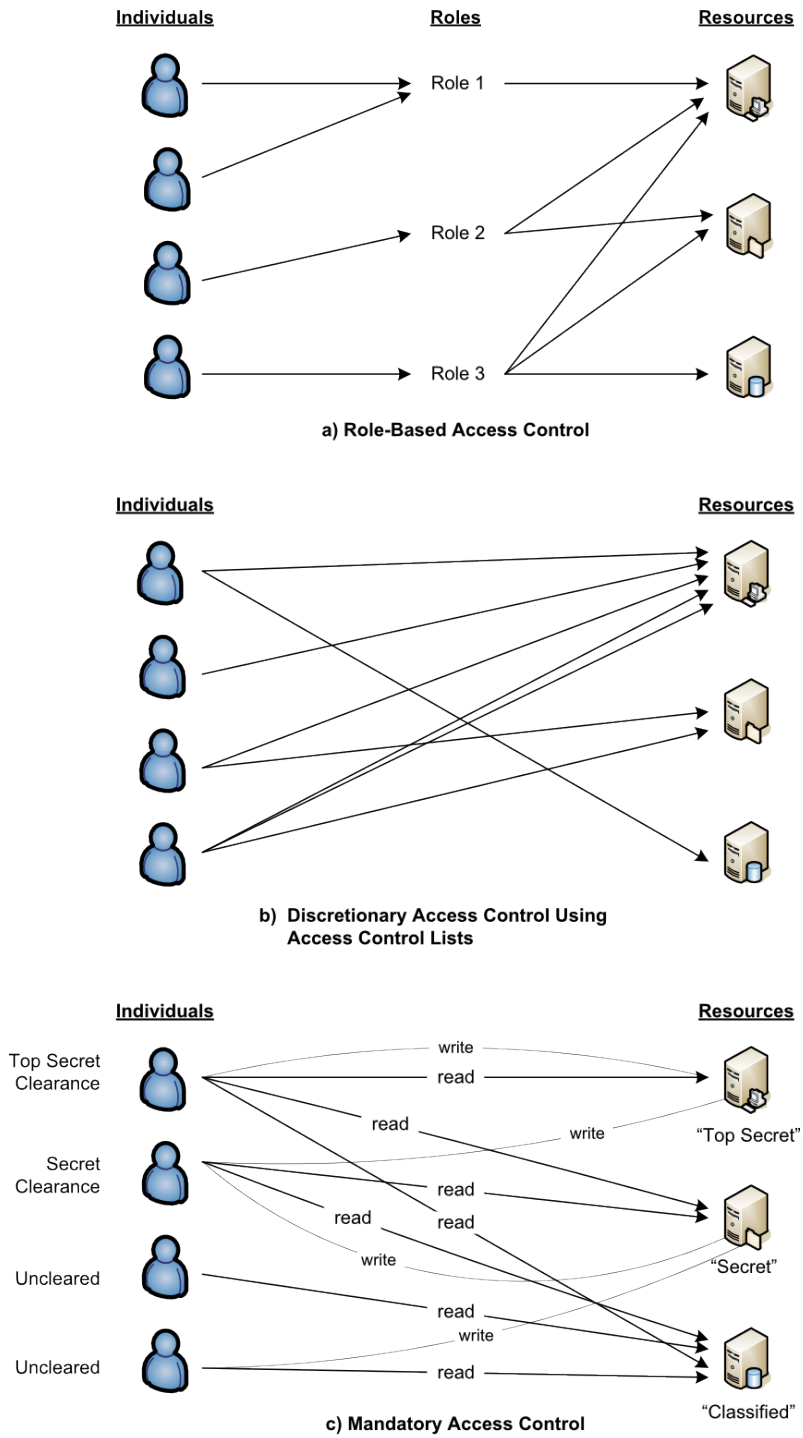

Abbildung 2.4: RBAC, DAC und MAC als Zugriffskontrollen [O'Connor und Loomis 2010, S. 31]

2.3.5 Unterscheidung zwischen Rollen- und Gruppenkonzept

Eine Gruppe stellt ähnlich wie die Rolle eine Zusammenfassung der Benutzer mit bestimmten Rechten dar und findet in DAC- und MAC-Systemen Anwendung [Valek 2010].

Im Vergleich zu RBAC-Systemen, in welchen der Rolle Rechte und dem Benutzer Rollen zugewiesen werden, kann man in einer Gruppe jedem Benutzer (Subjekt) die Rechte zusätzlich einzeln zuweisen (Abb. 2.5) [Sandhu 1996]. Bei der Prüfung der Rechte durch das System ist es dabei nicht relevant, ob die Rechte eines Subjekts direkt oder durch eine Gruppe zugewiesen wurden [Wiesner 2008].

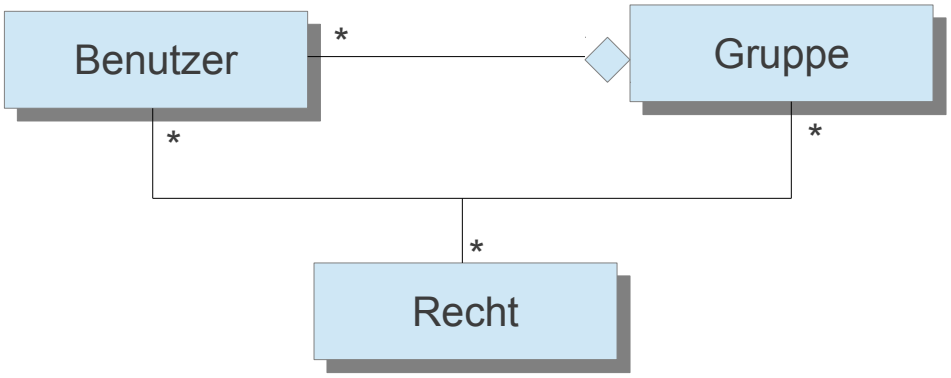

Abbildung 2.5: Gruppenkonzept, angelehnt an [Wiesner 2008]

Ein großer Nachteil des Gruppenkonzeptes besteht darin, dass die Rechte nicht bei der Gruppe direkt, sondern entweder auf der Seite der Objekte *ACL* oder auf der Seite der Subjekte liegen können. Die Gruppen können dann z. B. in ACL-Listen der Objekte vermerkt werden.
Bei einer Erhebung aller Rechte eines bestimmten Nutzers *(Subjekts)* im System, müssen somit die ACL-Listen aller *Objekte* traversiert werden (Abb. 2.6). Das stellt im Vergleich zu RBAC einen erhöhten Aufwand dar, da dort nur die Rollen mit diesem Benutzer gefunden werden müssen [Valek 2010].

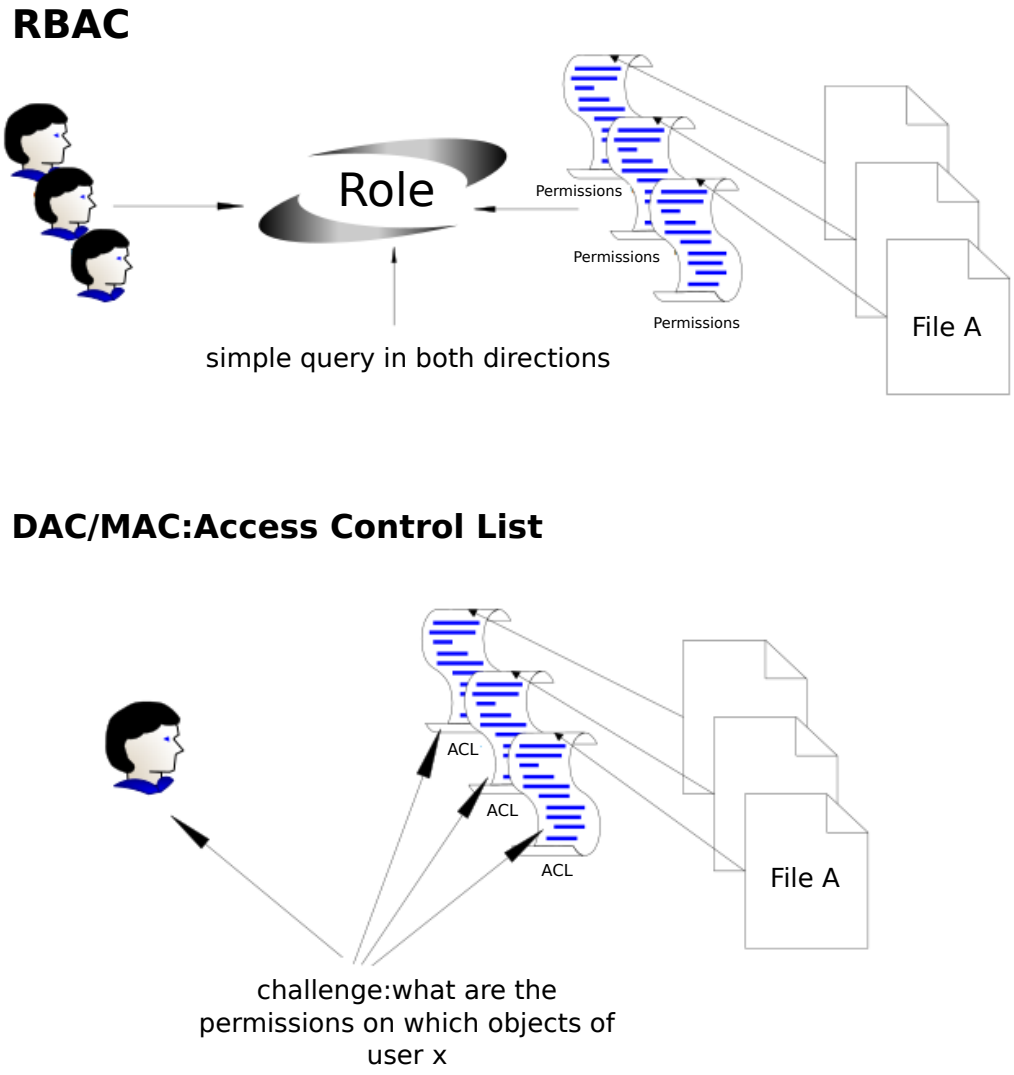

Abbildung 2.6: Berechnung der Berechtigungen in RBAC- und DAC-/MAC-Systemen [Valek 2010]

2.4 Rollenbasierte Zugriffsmodelle (RBAC)

In einer Konferenz über Computersicherheit in den USA aus dem Jahr 1992 veröffentlichten David Ferraiolo und Richard Kuhn vom NIST[2] *(National Institute of Standards and Technology)* einen Beitrag mit dem Titel „Role-Based Access Control“ [Ferraiolo und Kuhn 1992]. In diesem Beitrag beschreiben sie die wesentlichen Aspekte der Berechtigungsverteilung mit

[2] http://www.nist.gov

Hilfe von Rollen [Tsolkas und Schmidt 2010].
Einige Jahre später entwickelte Ravi Sandhu von der George Mason University ein Framework für die Zugriffskontrolle in informationstechnischen Strukturen.
Basierend auf dem Beitrag und dem Framework entstand im Jahr 2000[3] ein Entwurf für einen Standard, welcher sich auf die rollenbasierte Zugriffskontrolle bezieht.
In den nächsten Jahren, nach einer Weiterentwicklung wurde es vom ANSI[4] *(American National Standards Institute)* übernommen und am 19. Februar 2004 zum ANSI-Standard erhoben.
Die Implementationen des *RBAC*-Standards finden eine große Akzeptanz in der Industrie, Finanzbereich, Gesundheitswesen, oder auch im Militär und werden heutzutage in nahezu allen Enterprise Resource Planning (ERP)-Systemen, Contentmanagememt- oder Dokumentenmanagementsystemen eingesetzt [Eckert 2009].
IBM setzt RBAC beispielsweise in solchen Produkten, wie *Tivoli* oder *IBM Websphere* ein [Tsolkas und Schmidt 2010]. Im Application Server von Oracle wird das Java RBAC Modul mit dem Namen *Java Authentication and Authorisation Service* verwendet [Gropp 2007]. RBAC findet aber auch in vielen verschiedenen SAP-Lösungen des SAP-Konzerns Anwendung [Lehnert, Stelzner und John 2013].

Das RBAC-Modell besteht insgesamt aus 4 *(RBAC0-RBAC3)* Ausprägungen (Abb. 2.7), welche aufeinander aufbauen und sich gegenseitig ergänzen [Gropp 2007] [Information Technology 2004]. Nachfolgend werden diese Ausprägungen einzeln dargestellt.

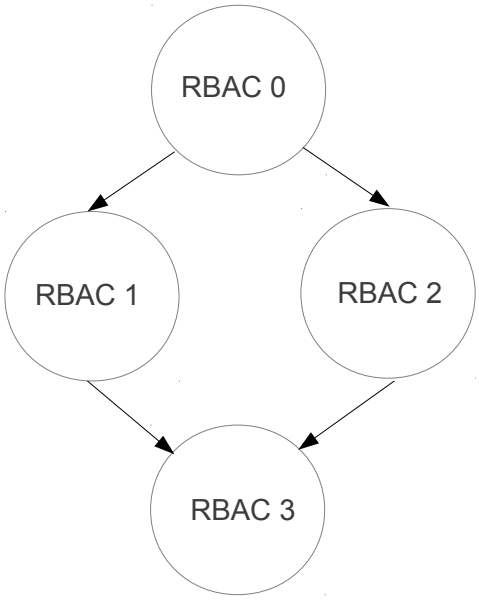

Abbildung 2.7: Ausprägungen des RBAC-Modells

[3] http://csrc.nist.gov/groups/SNS/rbac/faq.html#timeline
[4] http://www.ansi.org/

2.4.1 Das Kernmodell (RBAC0)

Das Kernmodell (RBAC0) (Abb. 2.8) bildet die Grundlage der restlichen RBAC-Ausprägungen *(RBAC1-RBAC3)* und setzt sich aus folgenden Komponenten zusammen [Tsolkas und Schmidt 2010] [Eckert 2009]:

- Benutzer *(User)*
  Ein Benutzer ist ein Subjekt, welcher Berechtigungen besitzt und sie auch nutzen kann.
- Rolle *(Role)*
  Eine Rolle ist ein Element, welchem Benutzer und die Berechtigungen zugewiesen werden.
- Benutzersitzung *(User Session)*
  Durch die Benutzersitzung kommt bei der Modellierung eine dynamische Komponente hinzu. Die *Session* erlaubt dem Subjekt die selektive Aktivierung oder Deaktivierung von Rollen.
- Objekt *(Object)*
  Hierbei sind Objekte (z. B. Verzeichnisse, Dateien) gemeint, welche durch die Berechtigungen geschützt sind.
- Operation *(Operation)*
  Mit der Operation wird die Art des Zugriffes auf einen Objekt beschrieben. Auf eine Datei kann man z. B. *lesend* oder *schreibend* zugreifen.
- Berechtigung *(Permission)*
  Bei einer Berechtigung handelt es sich um eine beliebige Kombination zwischen den *Operationen* und *Objekten*.

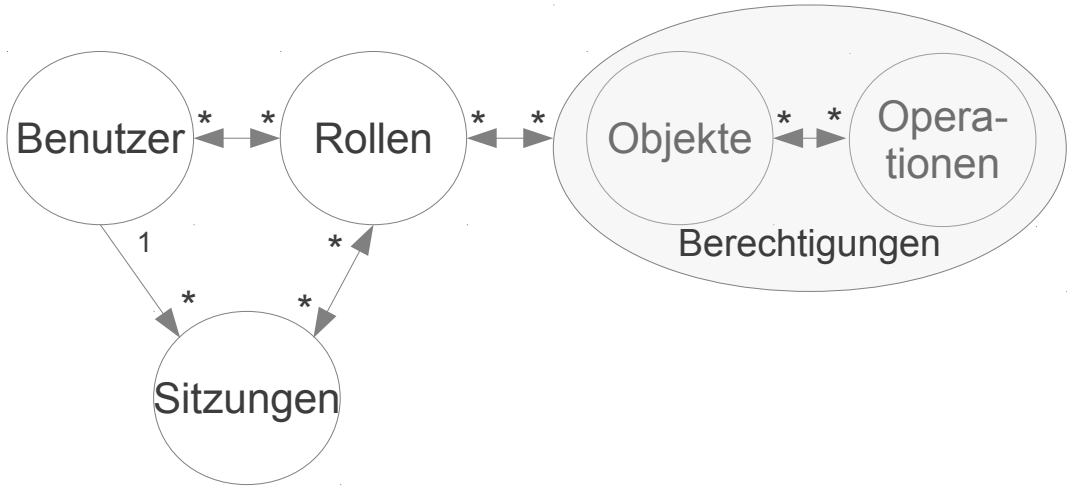

Abbildung 2.8: Das Kernmodell (RBAC0)

In diesem Modell eröffnen die Benutzer die Sitzungen (Sessions), wobei jeder Benutzer mehrere Sitzungen, aber jede Sitzung nur einen Benutzer haben kann. Jedem Benutzer werden dann abhängig von seinen Aufgaben eine oder mehrere Rollen zugeteilt. Diese Rollen können dann innerhalb dieser Sitzungen angewendet werden. Dadurch hängen die erhaltenen Rechte der Benutzer von den aktivierten Rollen innerhalb der jeweiligen Sitzung(en) ab [Information Technology 2004].
Erst durch das Konzept der Sitzungen kommt die Unterstützung des bereits bekannten *Principle of Least Privilege* zustande. Hierbei bekommt der Benutzer nur die Rollen zugewiesen, welche für die Ausführung seiner Aufgaben notwendig sind [Valek 2010]. Nachfolgend kommt eine formale Beschreibung des RBAC0-Modells.

Formale Beschreibung der Elemente und Relationen von $RBAC0$[5] [Molitorisz 2008] [Information Technology 2004]:

- Elemenete des Modells: *users*, *roles*, *operations*, *objects*, *sessions*, *permissions*
- Menge an Berechtigungen: $permissions = 2^{(Operations \times Objects)}$
- m-zu-n-Beziehung zwischen Benutzern und Rollen: UA $\subseteq users \times roles$
- m-zu-n-Beziehung zwischen Rollen und Berechtigungen: PA $\subseteq permissions \times roles$
- Abbildung von Berechtigungen auf Operationen; dient zur Rückgabe der Operationen für die definierte Berechtigung: Op(p:*permissions*) $\rightarrow \{op \subseteq operations\}$
- Abbildung von Berechtigungen auf Objekte; dient zur Rückgabe der Objekte, welche durch die *permission* belegt sind: Op(p:*permissions*) $\rightarrow \{ob \subseteq objects\}$
- Abbildung eines Benutzers auf dessen Sitzungen: $user\_sessions(u : user) \rightarrow 2^{session}$
- Abbildung einer Rolle auf die darin definierten Benutzer:
  $assigned\_users(r : roles) \rightarrow 2^{users}$,
  $assigned\_users(r) = \{u \in users \mid (u, r) \in UA\}$;
- Abbildung einer Rolle auf eine Teilmenge der Berechtigungen:
  $assigned\_permissions(r : roles) \rightarrow 2^{permissions}$,
  $assigned\_permissions(r) = \{p \in permissions \mid (p, r) \in PA\}$;
- Abbildung einer Sitzung auf die darin definierten Rollen:
  $session\_roles(s : session) \rightarrow 2^{roles}$,
  $session\_roles(s_i) \subseteq \{r \in roles \mid (session\_users(s_i), r) \in UA\}$;
- Abbildung eines Benutzers auf Berechtigungen für eine spezielle Sitzung:
  $avail\_session\_perms(s) = \{\bigcup_{r \in session\_roles(s)} assigned\_permissions(r)\}$;

[5] http://profsandhu.com/journals/tissec/ANSI+INCITS+359-2004.pdf

2.4.1.1 Funktionale Spezifikation

Im Kernmodell (RBAC0) bedarf es den administrativen Funktionen um einzelne Elemente des Modells zu erzeugen und zu verwalten.
Die Modellelemente *Object* und *Operation* werden meistens vom Anwendungsprogramm oder Betriebssystem vordefiniert und administriert. Aus diesem Grund beschränkt sich die Verwaltung im Modell auf die Elemente Benutzer und Rollen [Tsolkas und Schmidt 2010].
Die administrative Funktionen werden im Modell ebenfalls formal beschrieben und können plattformunabhängig in verschiedenen Systemumgebungen implementiert werden [Information Technology 2004].
Für die Elemente Benutzer und Rollen werden folgende administrative Funktionen benötigt [Tsolkas und Schmidt 2010]:

- *Add User* Anlegen eines neuen Benutzers.
- *Change User* Änderung der Attribute (z. B. Name) eines Benutzers.
- *Delete User* löscht einen vorhanden Benutzer.
- *Add Role* Anlegen einer neuen Rolle.
- *Change Role* Änderung der Rolle.
- *Delete Role* Löschung der Rolle.
- *AssignUser* ordnet einer Rolle einen Benutzer zu.
- *DeassignUser* Aufhebung der Zuordnung zwischen Rolle und Benutzer.
- *GrantPermission* erstellt eine Zuordnung zwischen Rolle und Berechtigung.
- *RevokePermission* Aufhebung der Zuordnung zwischen Rolle und Berechtigung .

Zur Verwaltung der dynamischen Benutzersitzung *(Session)* sind laut RBAC-Standard folgende Funktionen zu implementieren:

- *CreateSession* Damit kann eine Benutzersitzung eröffnet werden. In dieser Sitzung werden automatisch die speziellen Sitzungsrollen *(session_roles)* aktiviert.
- *AddActiveRole* Mit dieser Funktion können im Rahmen der Benutzersitzung zusätzliche Rollen aktiviert werden.
- *DropActiveRole* Deaktivierung der aktiven Rolle in der Benutzersitzung
- *CheckAccess* Diese Funktion prüft und gibt einen boolischen Wert zurück, ob ein Benutzer während der Sitzung eine bestimmte Operation auf einen Objekt ausführen darf.

Um den Überblick über die Elemente und die Zuordnungen zu behalten sind folgende Berichtsfunktionen nötig:

- *AssignedUsers* Rückgabe der Benutzer aus einer Rolle.

- *AssignedRoles* Rückgabe von Rollen mit einem bestimmten Benutzer.
- *RolePermissions* Rückgabe der Berechtigungen aus einer bestimmten Rolle.
- *UserPermissions* Rückgabe der Berechtigungen eines Benutzers in allen vorhandenen Rollen.
- *SessionRoles* Rückgabe der aktivierten Rollen in einer Benutzersitzung.
- *SessionPermissions* Rückgabe der Berechtigungen eines Benutzers in den Sitzungsrollen.
- *RoleOperationsOnObject* gibt zurück, welche Aktionen eine bestimmte Rolle an einem bestimmten Objekt ausführen darf.
- *UserOperationsOnObject* gibt zurück, welche Aktionen ein bestimmter User aufgrund seiner Rollenzuordnung an einem bestimmten Objekt ausführen darf.

2.4.2 Modell mit Hierarchien (RBAC1)

In RBAC1 wird die grundlegende Struktur des RBAC0-Modells beibehalten und durch die Implementierung einer Rollenhierarchie (Abb. 2.9) erweitert.

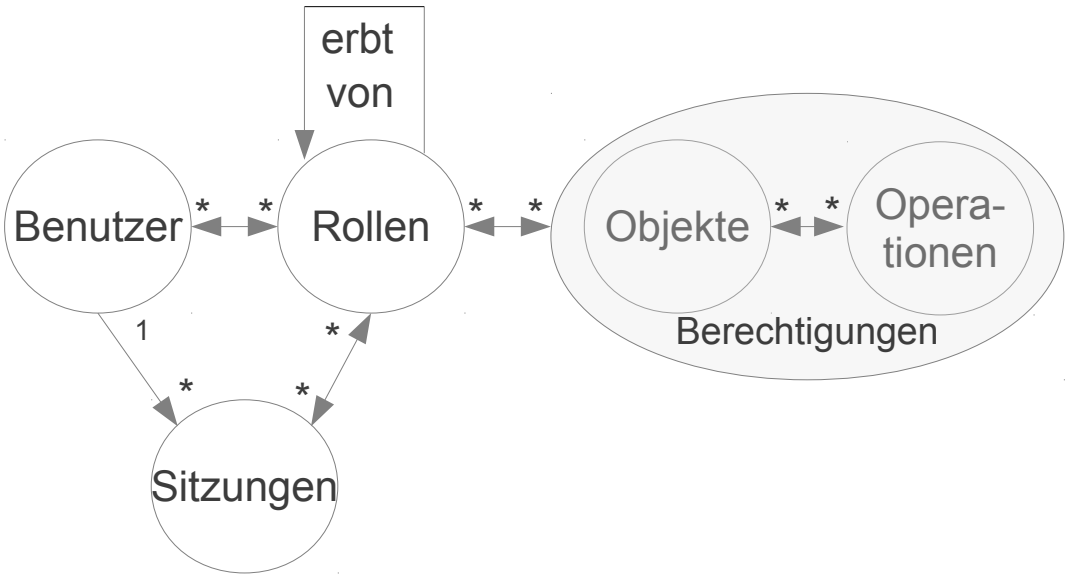

Abbildung 2.9: Hierarchisches (RBAC1)

Die Hierarchien beschreiben dabei eine Vererbungsrelation mit Hilfe von Rollen. Mit Hierarchien können z. B. die Verantwortlichkeiten in einem größeren Unternehmen oder Organisation abgebildet werden. Laut RBAC-Standard existieren zwei Möglichkeiten der Vererbung [Information Technology 2004]:

◇ *General Hierarchical RBAC*
Die Rollen können in dem Fall beliebige hierarchische Beziehungen eingehen, d. h. jede Rolle kann mehrere übergeordnete Rollen haben.

◇ *Limited Hierarchical RBAC*
Manche IT-Systeme unterstützen die beliebigen hierarchischen Beziehungen nicht. Aus diesem Grund besitzen die Rollen in dieser Vererbungsart eine einfache oder invertierte Baumstruktur. Im Beispiel aus dem NIST-Standard[6] (Abb. 2.10) erben *PL1* die Rechte von *PE1* und *QE1* und *PL2* die Rechte von *PE2* und *QE2*. Der *Director DIR* erbt die Rechte von *PL1* und *PL2*.

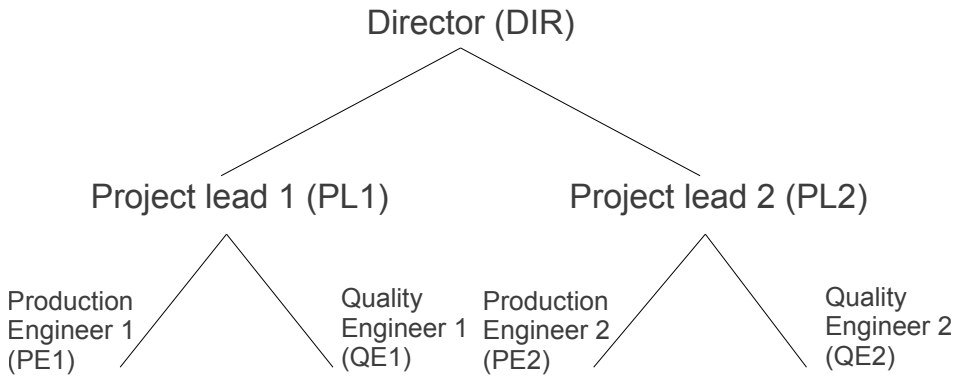

Abbildung 2.10: Rollenhierarchie als Baumstruktur aus dem NIST-Standard

Die Vererbung zwischen den Rollen verläuft dann so, dass die Benutzer der übergeordneten Rolle zu den Benutzern der untergeordneten Rolle *(top-down)* und die Rechte der untergeordneten Rolle zu den Rechten der übergeordneten Rolle *(bottom-up)* zugeteilt werden (Abb. 2.11).

Die formale Beschreibung des RBAC1-Modells basiert auf der formalen Beschreibung des RBAC0-Modells mit folgenden zusätzlichen Elementen und Relationen:

- Abbildung einer Rolle r auf eine Menge von Benutzern welche direkt in der Rolle, oder in der Rollenhierarchie enthalten sind:
  $Authorized\_users(r : roles) \rightarrow 2^{users}$,
  $Authorized\_users(r) = \{u \in users \mid r' \succeq r, (u, r') \in UA\}$;
- Abbildung einer Rolle r auf eine Menge von Berechtigungen direkt in dieser Rolle, oder durch die Berechtigungen entlang der Hierarchie:
  $Authorized\_permissions(r : roles) \rightarrow 2^{permissions}$,
  $Authorized\_permissions(r) = \{p \in permissions \mid r' \succeq r, (p, r') \in PA\}$;

[6] `http://csrc.nist.gov/staff/Kuhn/towards-std.pdf`

- Formale Beschreibung der Rollenhierarchie (RH) als partielle Ordnung auf Rollen. $RH \subseteq roles \times roles$,
  $R1 \preceq R2 \Rightarrow authorized\_permissions(R2) \subseteq authorized\_permissions(R1) \wedge authorized\_users(R1) \subseteq authorized\_users(R2)$;

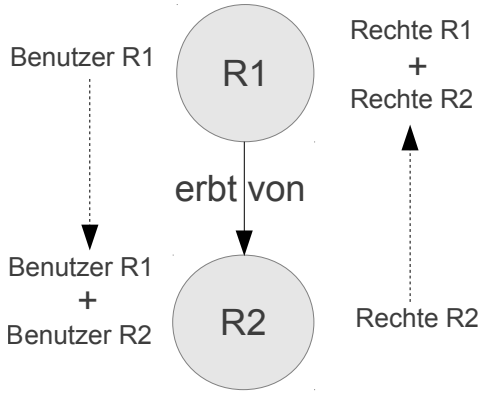

Abbildung 2.11: Vererbung zwischen zwei Rollen in RBAC1

2.4.2.1 Administrative Funktionen

In RBAC1 bedarf es der zusätzlichen administrativen Funktionen um die Rollenhierarchien zu verwalten:

- *AddInheritance* Diese Funktion erzeugt eine neue hierarchische Beziehung zwischen zwei Rollen. Diese zwei Rollen dürfen zu diesem Zeitpunkt in keiner hierarchischen Beziehung zu einander stehen [Tsolkas und Schmidt 2010].
- *DeleteInheritance* Diese Funktion löscht eine vorhandene hierarchische Beziehung.
- *AddAscendent* Hier wird eine neue Rolle erzeugt, welche hierarchisch über der angegebenen Rolle platziert wird.
- *AddDescendent* Erzeugt eine Rolle, welche hierarchisch unter der angegebenen Rolle platziert wird.

2.4.3 Modell mit Beschränkungen (RBAC2)

Das Modell mit Beschränkungen (RBAC2) erweitert das RBAC0 um die Funktionstrennung *(Separation of Duty)*, welche in der Praxis eine große Bedeutung besitzt [Botha und Eloff 2001]. Die Funktionstrennung schreibt vor, dass ein Nutzer nicht gleichzeitig zwei Rollen ausführen darf, die sich kritisch überschneiden [Information Technology 2004].
Einfache Beispiele dafür sind: Kassierer/Kassenprüfer, Antragsteller/Antrag-Genehmiger

oder Produktion/Qualitätskontrolle.
Der RBAC-Standard unterscheidet dabei zwei Arten der Funktionstrennung:
statische Funktionstrennung und *dynamische Funktionstrennung* (Abb. 2.12).

- **Statische Funktionstrennung *(Static Separation of Duty (SSOD))***
  Bei einer statischen Funktionstrennung findet die Überprüfung der Rollenkombinationen zum Zeitpunkt der Benutzerzuordnung zu einer Rolle statt. Das heißt, dass bei jeder Einteilung eines Benutzers zu einer Rolle eine Kontrolle stattfindet, ob es eine Beschränkung *(Constraint)* für diesen Benutzer im Bezug auf andere Rollen existiert.
  Der NIST-RBAC-Standard nennt zwei Möglichkeiten der technischen Umsetzung von SSOD: Entweder durch eine globale Liste mit den Rollen, die sich gegenseitig ausschließen, oder es existiert dafür für jede Rolle eine eigene Liste, die beim Anlegen eines Benutzers überprüft wird.

- **Dynamische Funktionstrennung *(Dynamic Separation of Duty (DSOD))***
  Bei der dynamischen Funktionstrennung werden die Rollen nicht zum Zeitpunkt des Anlegens, sonst erst während der Sitzung geprüft. Im Unterschied zur statischen Funktionstrennung, ist es bei der dynamischen ausdrücklich erlaubt, dass unvereinbare Rollen einem User zugeordnet werden.
  Ein Beispiel zur Verdeutlichung: Ein Anwender kann sich gleichzeitig in den Rollen *Kassierer* und *Kassenprüfer* befinden. Er darf aber nicht gleichzeitig in diesen zwei Rollen operieren, d. h. in einer Sitzung gleichzeitig *Kassierer* und *Kassenprüfer* sein.
  Im Vergleich zur statischen Funktionstrennung, in welcher sich die Anzahl potentiell zur Verfügung stehenden Rollen stark verringern kann, nimmt die dynamische Funktionstrennung die Beschränkung auf der Rollenteilmenge nur im aktuellem Kontext (Sitzung) vor.

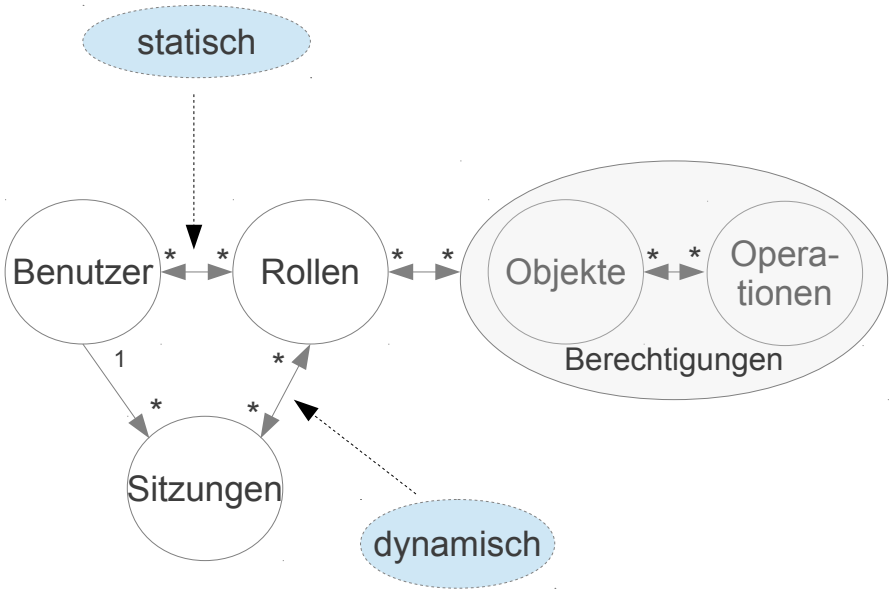

Abbildung 2.12: RBAC2 mit statischen und dynamischen Funktionstrennung

2.4.4 Vereinigung von RBAC1 und RBAC2 (RBAC3)

Das letzte Teilmodell (RBAC3) des RBAC-Standards vereinigt die beiden Teilmodelle RBAC1 und RBAC2. In der Grundform sind die beiden Teilmodelle nicht zu einander kompatibel, da die Rollenhierarchie aus RBAC1 die Beschränkungen (Constraints) auf Rollen aus RBAC2 außer Kraft setzen kann.
Aus diesem Grund muss bei der Vereinigung der beiden Teilmodelle beachtet werden, dass die SSOD- und DSOD-Definitionen durch Vererbung entlang der Rollenhierarchie unter Umständen angepasst werden müssen.
Durch die Vereinigung der beiden Teilmodelle (RBAC1 und RBAC2) stellt das RBAC3 letztendlich ein sehr mächtiges Modell zur Verwaltung von Rollen dar (Abb. 2.13).

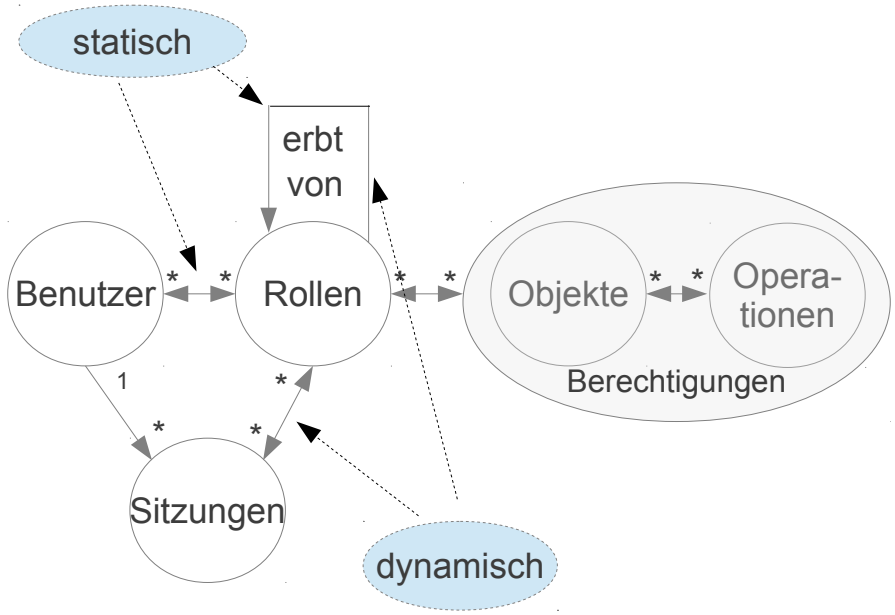

Abbildung 2.13: RBAC3 als Vereinigung von RBAC1 und RBAC2

2.5 Erweiterungen des RBAC-Modells

Aufgrund der Praxisrelevanz hat sich das RBAC-Modell als operativer De-facto-Standard zur Modellierung und technischer Umsetzung von Rollen etablieren können [O'Connor und Loomis 2010, $ES5 - ES6$].
Mit dem RBAC-Modell lassen sich Objekte anwendungsspezifisch modellieren und die Berechtigungen können dabei aufgaben- oder organisationsorientiert vergeben werden. Die Erfassung unzulässiger Abhängigkeiten der Rollen durch die Funktionstrennung, sowie die Möglichkeit der Modellierung der Rollenhierarchien bieten zusammen gute Voraussetzungen, um in Organisationsstrukturen der großen und hierarchisch aufgebauten Unternehmen

eingesetzt zu werden.
Um die Ausdrucksstärke des RBAC-Standards zu erhöhen, wurden mehrere Konzepte zur kontextbezogenen Zugriffskontrolle von Rollen in das Modell integriert. Durch die kontextbezogene Modellierung erhöht sich die Flexibilität der Zugriffskontrolle, da dadurch die Zugriffsbeschränkungen verfeinert werden können.
Nachfolgend werden mehrere Ansätze der kontextabhängigen Erweiterung des RBAC-Modells vorgestellt.

2.5.1 ABAC - Attributbasierte Zugriffskontrolle

Das Attribute-Based Access Controll (ABAC) ist ein Zugriffskontrollansatz, welches auf Basis von Benutzerattributen *(Credentials)* und Metadaten entwickelt wurde. Das Ziel von ABAC besteht darin, die starre Kopplung von Benutzern und Rollen sowie die Rollen und Berechtigungen aus dem RBAC-Modell flexibler zu gestalten. Dabei können die zugeteilten Rollen nach wie vor verwendet und durch Hinzunahme von zusätzlichen Attributen verfeinert werden.
Die Identität der Benutzer spielt im Vergleich zu RBAC eine sekundäre Rolle. Viel wichtiger ist die Sicherstellung, dass die Benutzer bestimmte Berechtigungen anhand ihrer Benutzerattributen besitzen [Priebe und Dobmeier 2005, S. 286-288].
Bei den Benutzerattributen kann es sich um allgemeine Eigenschaften wie z. B. die Unternehmensposition, erworbene Kenntnisse, Alter, aber auch dynamische Attribute wie aktueller Standort eines Benutzers handeln.
Auf der Seite der Objekte lassen sich die Inhalte der Dokumente mit Hilfe von Metadaten beschreiben, die als Attribute von Objekten herangezogen werden können.
Es existieren bereits mehrere Ausprägungen des ABAC-Ansatzes wie z. B. Shibboleth[7] und Permis[8]. Mit *eXtensible Access Control Markup Language (XACML)*[9], als Standard vom *OASIS*-Konsortium, können Regeln aufgestellt werden, welche ebenfalls auf Attributen von Objekten und Subjekten basieren und durch deren Auswertung die Zugriffe auf Ressourcen gesteuert werden können.
In der Abbildung 2.14 ist ein UML-Klassendiagramm zu sehen, welches als Referenzmodell für die attributsbasierte Zugriffskontrolle gilt. Subjekte und Objekte sind dabei durch die verschiedenen Attributsbedingungen (*SubjectQualifier* und *ObjectQualifier*) z. B. (Alter > 18) dargestellt.
Die eigentliche Autorisierung wird zwischen *SubjectDescriptor* und *ObjectDescriptor* definiert. Ein *SubjektDescriptor* beschreibt dabei eine Menge von Subjekten mit bestimmten Attributswerten. Das gleiche gilt dann auch für einen *ObjectDescriptor*.
Die Verwendung von Subjekt- und Objektattributen bei der Autorisierung bringt aber auch zusätzlichen Aufwand mit sich. Die Attribute müssen durchgehend syntaktisch und semantisch korrekt sein und dem aktuellen Stand der Subjekte und Objekte entsprechen [Klarl 2011, S. 60].

[7] `http://shibboleth.net/`
[8] `http://www.permis.org/`
[9] `http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf`

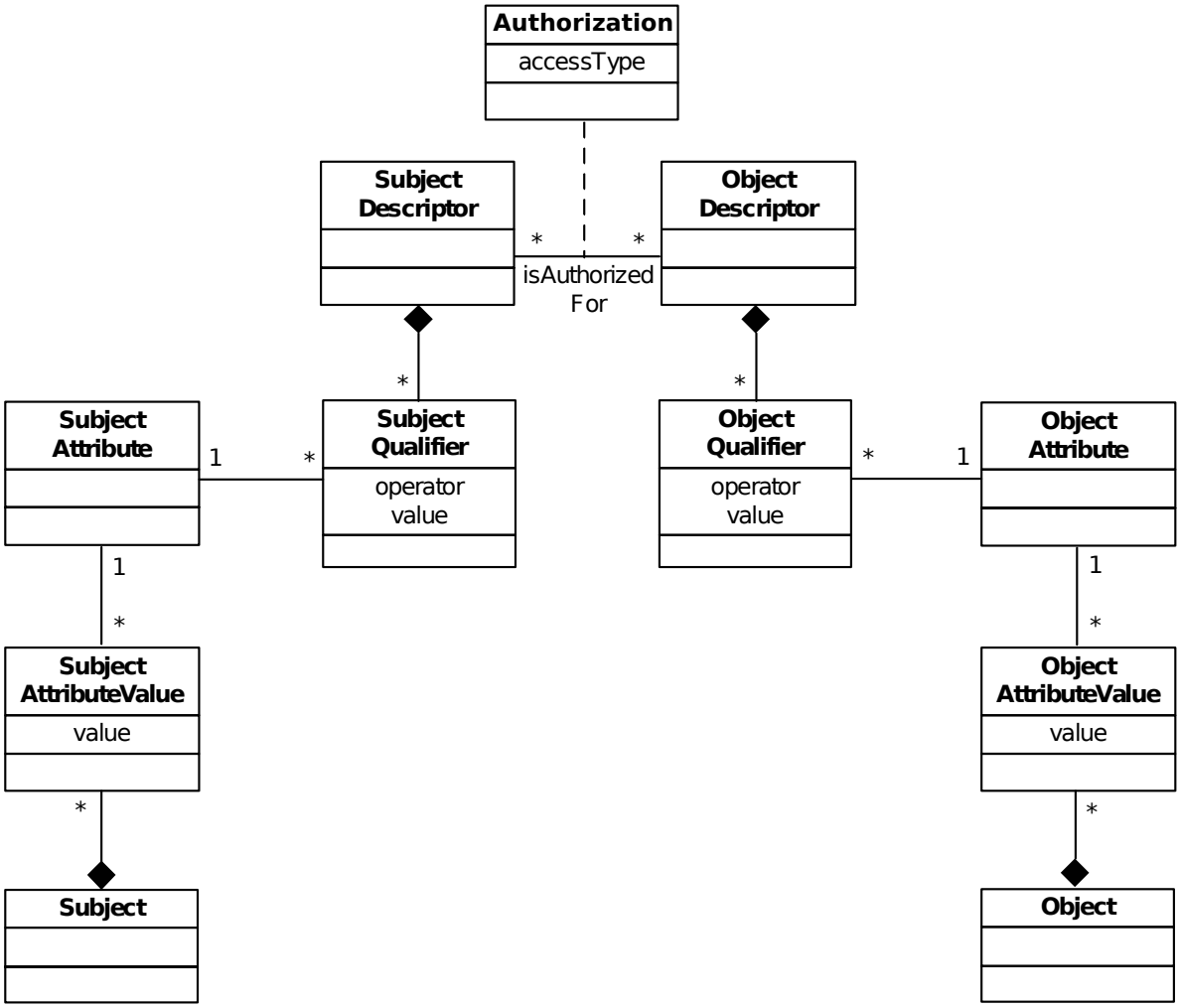

Abbildung 2.14: ABAC Grundmodell [Priebe und Dobmeier 2005, S. 287]

2.5.2 GTRBAC

In Unternehmen können Geschäftsabläufe eine zeitliche Begrenzung oder periodischen Ablauf haben. Das *Generalized Temporal RBAC-Modell (GTRBAC)* stellt eine temporale Ergänzung des rollenbasierten Zugriffsmodells-RBAC dar, um die zeitabhängige Vorgänge und Beschränkungen auf bestimmte, sicherheitskritische Ressourcen zu spezifizieren [Joshi und Bertino 2005].
Das Modell beschreibt im Einzelnen die zeitlichen *Constraints* auf einzelne Rollen, die Zuweisungen von Benutzern zu Rollen und die Zuweisungen von Berechtigungen zu Rollen. Darüber hinaus unterstützt das GTRBAC die zeitlichen *Constraints* in Bezug auf die bereits bekannten Rollenhierarchien und Funktionstrennung *(Separation of Duty)*. Die Ellipsen in der (Abb. 2.15) verdeutlichen zusammenfassend die Möglichkeiten der Formulierung von zeitlichen *Constraints* zwischen den einzelnen Elementen des RBAC-Modells.

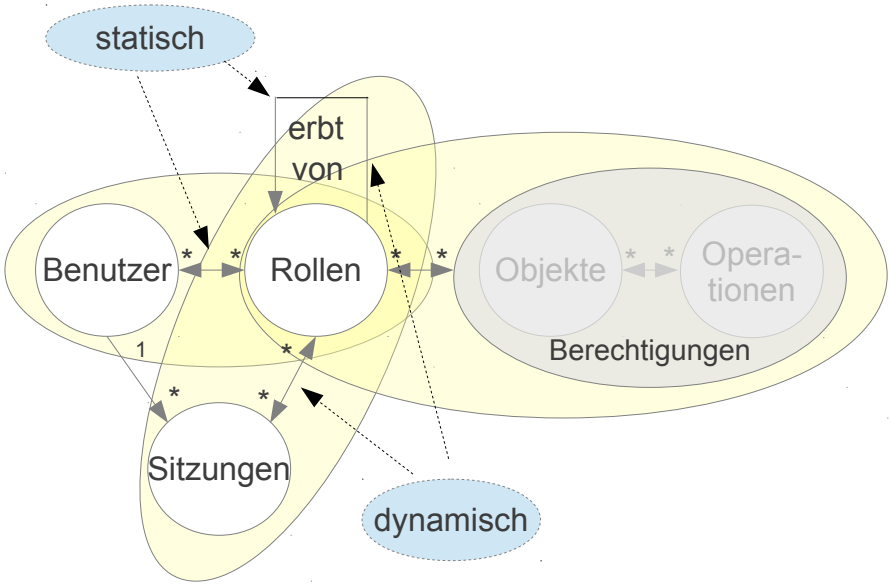

Abbildung 2.15: GTRBAC mit temporären Beschränkungen *(Constraints)*

Nachfolgend wird am Beispiel eines medizinischen Informationssystems eines Krankenhauses aufgezeigt, wie die zeitlichen Beschränkungen *(Constraints)* (Tab. 2.2) mit GTRBAC spezifiziert werden können [Joshi, Bertino und Shafiq 2003]. Die zeitlichen *Constraints* werden dabei als Tupel (I, P, D, E) angegeben:

- I: gibt ein Zeitintervall an, in dem das Tupel gültig ist
- P: beschreibt Zeitperiode innerhalb des Intervalls I
- D: Dauer der Gültigkeit eines Ereignisses oder einer Aktion
- E: definiert ein Ereignis bzw. Aktion, welche ausgeführt wird

Der periodische *Constraint* 1*a* aus (Tab. 2.2) spezifiziert die Aktivierungszeit (*DayTime, NightTime*) der Rollen *DayDoctor* und *Nightdoctor*.
Der periodische *Constraint* 1*b* erlaubt die Zuweisung des Benutzers *Adams* zur Rolle *DayDoctor* nur an bestimmten Wochentagen (*Monday, Wednesday, Friday*).
Die Zuweisung von *Carol* zur Rolle *DayDoctor* in 1*c*, ist nur zwischen 10:00 Uhr und 15:00 Uhr gestattet.
In 2*a* sind *Ami* und *Elizabeth* ohne jegliche Zeitbeschränkungen in die Rollen *NurseInTraining* und *DayNurse* eingeteilt.
Der *Constraint* **c1** in 2*b* ist ab Aktivierung 6 Stunden lang gültig. Während dieser Zeit darf ein Benutzer zur Rolle *NurseInTraining* nur für 2 Stunden zugewiesen werden.
In der dritten Spalte werden die Aktionen mit Hilfe von Triggern ausgeführt.
In 3*a* wird bei der Aktivierung der Rolle *DayNurse* die Beschränkung **c1** gestartet. Das bedeutet, dass die Rolle *NurseInTraining* in den nächsten 6 Stunden genutzt werden kann.
Der Constraint in 3*b* sagt aus, dass 10 Minuten, nach dem *Elizabeth* die Rolle *DayNurse* aktiviert hat, die Rolle *NurseInTraining* 2 Stunden lang genutzt werden kann.

In 3*c* wird nach der Aktivierung der Rolle *DayDoctor*, die Rolle *DayNurse* nach einer Wartezeit von 10 Minuten aktiviert.

1	a.	(***DayTime***, *enable* DayDoctor), (***NightTime***, *enable* NightDoctor);
	b.	((M, W, F), $assign_U$ Adams *to* DayDoctor);
	c.	([10am, 3 pm], $assign_U$ Carol *to* DayDoctor)
2	a.	($assign_U$ Ami *to* NurseInTraining); ($assign_U$ Elisabeth *to* DayNurse)
	b.	**c1** = (6 hours, 2 hours, *enable* NurseInTraining);
3	a.	(*enable* DayNurse $\rightarrow$ *enable* **c1**);
	b.	(*activate* DayNurse for Elizabeth $\rightarrow$ *enable* NurseInTraining $after$ 10 min);
	c.	(*enable* DayDoctor $\rightarrow$ *enable* DayNurse $after$ 10 min);

Tabelle 2.2: Temporäre Rollenbeschränkungen in einem Krankenhaus nach GTRBAC [Joshi, Bertino und Shafiq 2003]

In der Arbeit von *Rafae A. Bhatti* [Bhatti 2003] ist eine mögliche, technische Umsetzung des GTRBAC-Modells mit Hilfe von einem XML-basierten Regelwerk *(Policy)* vorgestellt. Im Rahmen der Arbeit wurde ein in Java geschriebenes Prototyp-Framework (X-GTRBAC) implementiert, welches die zeitlichen Beschränkungen *(Constraints)* bei der Zuweisung von Benutzern zu Rollen regelt.

2.5.3 ERBAC

IT-Umgebungen großer Unternehmen bestehen heutzutage aus verschiedenen Plattformen und Applikationen. Dazu gehören unterschiedliche Betriebssysteme (z. B. Windows, Linux), Datenbanken (z. B. Oracle[10], DB2[11]), Geschäftsprozessmanagementsysteme, aber auch große Anzahl an verschiedenen Geschäftsapplikationen wie z. B. SAP R/3[12] [Kern 2002, S. 5].
Ein typischer Benutzer in einem solchen Unternehmen muss einen Zugang zu allen diesen Systemen besitzen, die in den meisten Fällen durch eigene Sicherheitslösungen geschützt sind. Die Sicherheitskomponenten dieser Applikationen unterscheiden sich jedoch in der Praxis stark voneinander und sind somit nicht ohne Weiteres miteinander vereinbar. Die Benutzer und die Rollen müssen aus diesem Grund in allen Systemen getrennt erstellt und gepflegt werden [Molitorisz 2008, S. 32].
Aufgrund der unterschiedlichen Sicherheitslösungen der einzelnen Applikationen und den damit verbundenen Aufwand bei der Administration von Benutzern und Systemen bietet sich eine globale Lösung zur unternehmensweiten Zugriffskontrolle an. Diese Lösung wird mit Hilfe von unternehmensweiten Rollen oder Geschäftsrollen *(Enterprise Roles)* im ERBAC-Modell bewerkstelligt [Kern 2002, S. 5]. Die Geschäftsrollen vereinen die Berechtigungen aus mehreren IT-Systemen *(Target Systems)* im Unternehmen. Die Zuweisung

[10] `http://www.oracle.com/`
[11] `http://www-01.ibm.com/software/data/db2/`
[12] `http://www.sap.com/germany/index.epx`

der Benutzer zu Geschäftsrollen funktioniert analog zu den Standard-Rollen aus dem RBAC-Modell.
Die Abbildung 2.16 zeigt ein Beispiel einer Geschäftsrolle *(Enterprise Role)*, welche eine Gruppe in UNIX[13], eine Rolle in Oracle und eine Gruppe in Resource Access Control Facility (RACF)[14] in sich vereint.

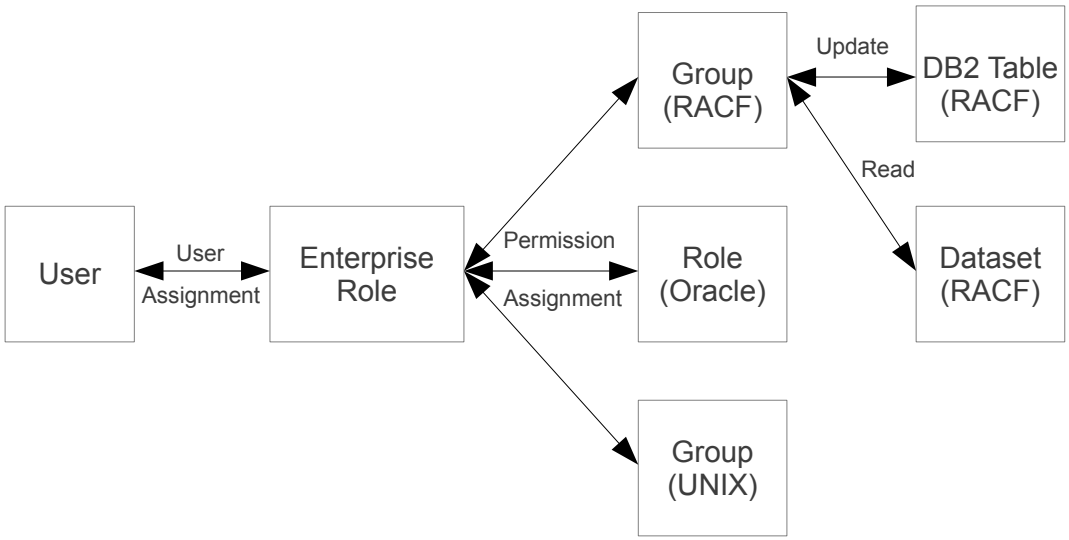

Abbildung 2.16: Beispiel einer Geschäftsrolle *(Enterprise Role)* [Kern 2002, S. 6]

Der wesentliche Unterschied zwischen RBAC- und ERBAC-Modell besteht zusätzlich darin, dass das Element-Sitzung *(Session)* in ERBAC-Modell nicht vorhanden ist. Eine Sitzung und somit auch die dynamische Funktionstrennung *(Dynamic Separation of Duty DSOD*) können aufgrund der autonom agierenden IT-Systemen in Geschäftsrollen nicht abgebildet werden. Durch den fehlenden direkten Bezug der Geschäftsrollen zu den technischen Systemen, ist es nicht einfach festzustellen, ob die Rollen sich gerade gegenseitig ausschließen. Eine dynamische Funktionstrennung kann aus diesem Grund nur direkt und innerhalb eines einzelnen IT-Systems implementiert werden [Kern 2002, S. 5] [Asprion 2013, S. 89-90].
Die Rollenhierarchien und die statische Funktionstrennung *(Static Separation of Duty)* werden vom ERBAC-Modell jedoch vollständig unterstützt. Die statische Funktionstrennung wird in Form von Regeln realisiert, welche bei der Zuweisung von Benutzern zu Rollen ausgewertet werden. Damit wird eine konsistente Rollenstruktur erreicht und die nicht erlaubten Einteilungen von Benutzern zu Rollen verhindert.
Bei der Zuweisung von Benutzern zu Geschäftsrollen werden die Berechtigungen mit den entsprechenden IT-Systemen im Unternehmen verknüpft. Dies kann auch zur Erzeugung eines neuen Benutzeraccounts in den Systemen führen. Eine Berechtigung in ERBAC-Modell entspricht dabei der Berechtigung des RBAC-Modells und kann jede Art der Autorisierung

[13] `http://www.unix.org/`
[14] `http://publibz.boulder.ibm.com/epubs/pdf/ich1a510.pdf`

darstellen, welche mit den einzelnen IT-Systemen formuliert werden kann (Abb. 2.17).
Die Zusammenführung der Berechtigungen in den Geschäftsrollen erfordert aber eine genaue Kenntnis der einzelnen IT-Systeme. Die Fachabteilung ist daher bei der Änderung oder Erstellung neuer Geschäftsrollen auf die Unterstützung des technischen Personals aus der IT-Abteilung angewiesen. Eine einwandfreie Kommunikation zwischen Fachabteilung und den Mitarbeitern der IT ist von daher zwingend notwendig, um einen optimalen Autorisierungsablauf im Unternehmen zu gewährleisten [Klarl 2011, S. 56].
Bei der Aufnahme vieler unterschiedlicher Berechtigungen in einer Geschäftsrolle darf außerdem das Konzept der minimalen Berechtigungsvergabe *(Least Privilege Concept)* nicht außer Acht gelassen werden. Die Beachtung des Konzeptes wird aber dadurch erschwert, da bei der Aufnahme einer Vielzahl von anwendungsspezifischen Berechtigungen der Verwaltungsaufwand und die Komplexität der Rollen deutlich ansteigen kann. Aus diesem Grund verwendet man in der Praxis *Identity Management (IdM)*-Werkzeuge, die zentral und übergreifend die Administration der Geschäftsrollen regeln und die Verwaltung von diesen erheblich erleichtern.

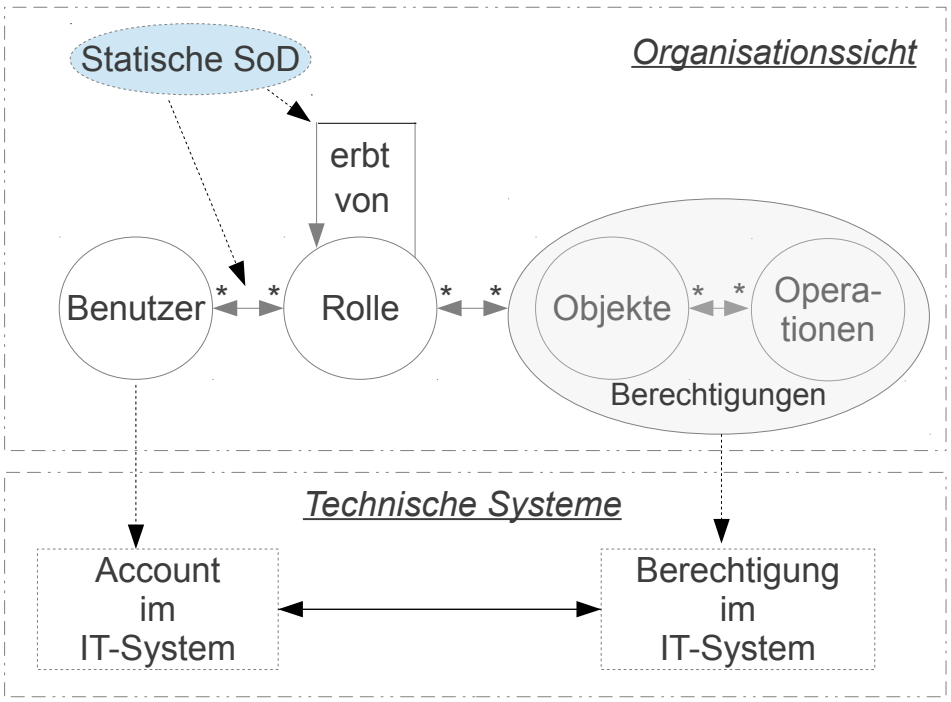

Abbildung 2.17: Enterprise RBAC-Modell (ERBAC), angelehnt an [Kern 2002, S. 7]

2.6 Zusammenfassung und Bewertung des rollenbasierten Berechtigungskonzeptes

Das Rollenkonzept kann man nach DAC- und MAC-Konzepten als die wichtigste Innovation im Autorisierungsmanagement betrachten [O'Connor und Loomis 2010, *ES*1] [Bertino und Sandhu 2005].
Anstatt die Rechte und Zugriffe auf Systeme direkt an die Benutzer in ACL (Access Control Lists) zu vergeben, funktioniert die Kontrolle des Zugangs im rollenbasierten System ausschließlich durch Rollen. Dies ermöglicht eine effektivere und effizientere Verwaltung von Berechtigungen im IT- und Geschäftsbereich der Unternehmen. Dadurch verbessert sich auch die Sicherheit und die Integrität der organisatorischen Systeme. Im Vergleich zu den anderen Ansätzen werden zudem die Kosten für GRC-Aktivitäten *(Governance, Risk Management, and Compliance – GRC)* deutlich gesenkt, da die Berechtigungen der Benutzer in Verbindung mit Rollen einer einfacheren Kontrolle durch IT-Revision bedürfen.
Eine im Auftrag von *National Institute of Standards and Technology*[15] im Jahr 2010 durchgeführte Studie zur ökonomischen Analyse der rollenbasierten Zugriffssysteme *(Economic Analysis of Role-Based Access Control)* in der IT-Industrie, kam zum Ergebnis, dass mehr als 80% der befragten Unternehmen (Anzahl der Mitarbeiter pro Unternehmen > 500) eine bessere Umsetzung der Sicherheitspolitik mit Verwendung von Rollen erreichen konnten [O'Connor und Loomis 2010, *ES*5]. Im Rahmen der Studie wurde auch folgendes festgehalten:

- Bei der Verwaltung von Rollen werden hauptsächlich proprietäre, anwendungsspezifische Rollenkonstrukte verwendet und keine eigenen konzipiert (78% der befragten Unternehmen).
- Annähernd die Hälfte (55 %) der befragten Unternehmen nutzen verschiedene *IdM*-Lösungen um die Berechtigungen aus verschiedenen IT-Systemen in Geschäftsrollen zu vereinen.
- Durch die spezifischen und teils unterschiedlichen Implementierungen des RBAC-Modells in verschiedenen Anwendungen sieht sich jedoch mehr als die Hälfte (55%) der Unternehmen mit Herausforderungen im Bezug auf die Übersichtlichkeit und Verwaltung der Anwendungssystemen konfrontiert. Zusätzlich lassen sich die Benutzer und die Berechtigungen in bestimmten Anwendungskombinationen oder in älteren IT-Systemen *(Legacy Systems)* nicht effektiv mit Rollen verwalten. Fast alle Unternehmen berichten deshalb von der hybriden Nutzung der Rollen mit ACL-Listen (Tab. 2.3).
  Die ersten fünf Einträge aus der (Tab. 2.3) zeigen diejenigen Informationssysteme, welche laut der durchgeführten Studie RBAC als primäres Zugriffskonzept verwenden. Dazu gehören unter anderem die Geschäftsprozessmanagementsysteme *(Business Prozess Management Systems)*, auf welche im Nachfolgenden genauer eingegangen wird.

[15] `http://www.nist.gov/`

System	ACLs	Roles	Rules	Attributes	Other
Human resource information systems	37%	56%	6%	2%	0%
Sales and customer relationship management systems	41%	52%	2%	4%	0%
Accounting and financial management systems	41%	50%	6%	2%	2%
Purchasing, order management, and logistics systems	41%	50%	7%	2%	0%
Business process management systems	42%	44%	7%	4%	2%
Enterprise database systems	43%	41%	10%	6%	0%
Electronic health record and health information systems	48%	34%	10%	7%	0%
Identity management systems	39%	34%	15%	7%	5%
Physical security services	50%	28%	9%	9%	4%
Directory services	49%	27%	10%	6%	8%
Network identity services Web services	53%	22%	14%	6%	4%
Web services	51%	20%	14%	6%	8%

Tabelle 2.3: Primäre Zugriffskontrolle in Informationssystemen, angelehnt an [O'Connor und Loomis 2010, *ES* – 7]

2.7 Geschäftsprozesse im Unternehmen

Der Wettbewerb und die Anforderungen des Marktes verleiten die Unternehmen schnell und flexibel auf die Veränderungen zu reagieren und gleichzeitig die Qualität der erzeugten Produkte zu sichern. Eine Möglichkeit diese Anforderungen zu erfüllen, ist das aus der Betriebswirtschaftslehre stammende Konzept des prozessorientierten Unternehmens. Die Aufgaben oder Aktivitäten eines Unternehmens werden dabei als Geschäftsprozesse beschrieben [Klarl 2011].
In der Literatur existieren dazu mehrere Begriffsdefinitionen:

*„Ein **Geschäftsprozess** definiert sich als ein am Kerngeschäft orientierter Arbeits-, Informations- und Entscheidungsprozess mit einem für den Unternehmenserfolg relevanten Ergebnis."* [Eiff und Ziegenbein 2003]

*„**Geschäftsprozesse** sind betriebliche Abläufe, die zur Leistungserstellung und Vermarktung eines Produktes oder einer Dienstleistung vollzogen werden müssen."* [Siegle 1994]

*„Ein **Geschäftsprozess** ist eine Folge von logisch zusammenhängenden Aktivitäten, der für das Unternehmen einen Beitrag zur Wertschöpfung leistet, einen definierten Anfang und ein definiertes Ende hat, typischerweise wiederholt durchgeführt wird und sich in der Regel am Kunden orientiert."* [Laudon 2010, S. 11]

Geschäftsprozesse werden in Kernprozesse, Steuerungsprozesse (Managementprozesse) und unterstützende Prozesse (Supportprozesse) unterteilt. [Koch 2011, S. 6-7]
Die Kernprozesse sind die wesentlichen Prozesse in Unternehmen und tragen direkt zur Wertschöpfung des Unternehmens bei.
Die Steuerungsprozesse (Managementprozesse) umfassen die Unternehmensplanung und Steuerung von Kernprozessen.
Die unterstützenden Prozesse sind die betrieblichen Prozesse, die den Kernprozess unterstützen und somit notwendig sind, selbst aber nicht direkt zur Wertschöpfung des Unternehmens

beitragen. Typische unterstützende Prozesse sind z. B. Personalwesen, Buchhaltung oder Instandhaltung.

2.7.1 Modellierung von Geschäftsprozessen

Im Rahmen des Geschäftsprozessmanagements *(BPM)* werden die Prozessabläufe in Unternehmen modelliert, gesteuert, überwacht und optimiert [Kocian 2011, S. 4].
Die Modellierung von Geschäftsprozessen wird zur Dokumentation, zum Verständnis der Prozessabläufe, der anschließenden Prozessanalyse sowie zum besseren Informationsaustausch zwischen Mitarbeitern verwendet.
Zur Modellierung von Prozessen haben sich im Laufe der Zeit mehrere grafische Modellierungssprachen wie z. B. *BPMN*[16], *Unified Modeling Language (UML)*, *Ereignisgesteuerte Prozesskette (EPK)* oder *Petri-Netze*[17] entwickelt.
Im Vergleich zu den anderen Modellierungssprachen, beschreibt die BPMN sowohl eine grafische Notation, als auch eine Ausführungssemantik für Geschäftsprozesse und besitzt dadurch großes Potenzial sich in technischer Umsetzung sowie in fachlicher Modellierung von Geschäftsprozessen als Standard durchzusetzen [Rücker, Freund und Henninger 2010, S. 8].
Die Ausführungssemantik beschreibt dabei wie die BPMN-Modelle durch die BPM-Systeme (Geschäftsprozessmanagementsysteme) zu interpretieren sind. Dadurch ist es möglich die BPMN-Modelle direkt in solchen BPM-Systemen auszuführen und die Umwandlung in andere Ausführungsformate wie z. B. *Business Process Execution Language (BPEL)*[18] ist nicht mehr nötig (siehe Kapitel 4 *Activiti*).

2.7.2 BPMN *(Business Process Model and Notation)*

BPMN ist eine grafische Notation, um Geschäftsprozesse darzustellen, die 2004 von der *Business Process Management Initiative (BPMI)* veröffentlicht wurde und zur Zeit als Standard bei der *Object Management Group (OMG)* angesiedelt ist [Allweyer 2009, S. 10].
Die aktuelle BPMN2.0 Version standardisiert ein XML-basiertes Format, in welchem die BPMN-Diagramme gespeichert und zwischen verschiedenen Modellierungs- und Ausführungswerkzeugen von Prozessmodellen ausgetauscht werden können [Allweyer 2009, S. 12].
Die BPMN2.0-Elemente werden insgesamt in vier Kategorien eingeteilt: Flussobjekte *(Flow Objects)*, verbindende Objekte *(Connecting Objects)*, Schwimmbahnen *(Swimlanes)* und Artefakte *(Artifacts)* (siehe Abb. 2.18), auf die nachfolgend genauer eingegangen wird.

Flussobjekte *(Flow Objects)*

Flussobjekte sind die eigentlichen Kernelemente und bilden die Knoten *(Activity, Gateway, Event)* in den Geschäftsprozessdiagrammen.

[16] `http://www.bpmn.org/`
[17] `http://public.beuth-hochschule.de/~grude/Petrinetze.pdf`
[18] `https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel`

Activity
Eine *Activity* (Aktivität) beschreibt eine zu erledigende Aufgabe *(Task)* in einem Geschäftsprozess und wird als Rechteck mit abgerundeten Ecken dargestellt. Komplexere (zusammengesetzte) Aktivitäten werden als *Subprocess* bezeichnet und unterscheiden sich durch ein „+“ Zeichen.

Gateway
Ein *Gateway* beschreibt einen Entscheidungspunkt, an welchem eine Verzweigung oder eine Zusammenführung von mehreren Sequenzflüssen modelliert wird. Es wird zwischen exklusivem Gateway (XOR), inklusivem Gateway (OR) und parallelem Gateway (AND) unterschieden.

Event (Ereignis)
Ein *Event* beschreibt ein Ereignis, welches in einem Geschäftsprozess auftreten kann. Die wichtigsten *Events* sind Start-, End- und Zwischenereignisse.

Verbindende Objekte *(Connecting Objects)*

Die Flussobjekte werden über verbindenden Objekte miteinander verbunden. Es existieren insgesamt drei Typen von verbindenden Objekten: Sequenzfluss, Nachrichtenfluss und Assoziation.

Sequenzfluss (Sequence Flow)
Der Sequenzfluss verbindet miteinander Aktivitäten, Gateways sowie Ereignisse und beschreibt die Ausführungsreihenfolge von diesen Elementen. Der Sequenzfluss ist durch einen durchgezogenen Pfeil symbolisiert.

Nachrichtenfluss (Message Flow)
Der Nachrichtenfluss ist durch einen gestrichelten Pfeil symbolisiert und wird für die Informationsweitergabe zwischen *Pools* verwendet.

Assoziation (Association)
Eine Assoziation ist mit einer gepunkteten Linie dargestellt. Mit Assoziation werden z. B. Daten, Beschreibungstexte und andere Artefakte mit Flussobjekten verbunden.

Pools* und *Lanes (Swimlanes)

Ein *Pool* stellt im Allgemeinen eine unabhängige Einheit (Unternehmen, Organisationseinheit) mit einem eigenen Geschäftsprozess dar und kann mit den anderen unabhängigen Einheiten durch den Nachrichtenfluss interagieren.

Ein *Pool* kann eine oder mehrere *Lanes* (Bahnen) enthalten. Eine *Lane* kann z. B. eine Abteilung oder Geschäftsrolle im Geschäftsprozess repräsentieren. Ein Nachrichtenfluss zwischen *Lanes* ist nicht erlaubt. Der Prozessfluss innerhalb der Lanes wird nur durch die Sequenzverbindungen bewerkstelligt.

Artefakte *(Artifacts)*

Mit Hilfe von Artefakten können die BPMN-Elemente durch einen zusätzlichen Kontext ergänzt werden. Sie dienen hauptsächlich dazu die Prozessabläufe verständlicher zu gestalten und haben keinen Einfluss auf den Prozessablauf. Als Artefakte gibt es Datenobjekt, Gruppierung und Anmerkung *(Annotation)*.

Datenobjekt (Data Object) Datenobjekte werden durch ein Dokumentensymbol dargestellt. Mit Hilfe von Datenobjekten kann man Ein- bzw. Ausgabeinformationen, die in einem Prozess vorkommen können darstellen.

Gruppierung (Group) Mit der Gruppierung kann man bestimmte Elemente des BPMN-Prozesses (z. B. zusammenhängende Aktivitäten) zur besseren Übersicht visuell zusammenfassen.

Anmerkung (Annotation) Die Anmerkungen dienen dazu, um bestimmte Elemente des BPMN-Prozesses mit zusätzlichen Informationen anzureichern und werden durch eine eckige (öffnende) Klammer dargestellt.

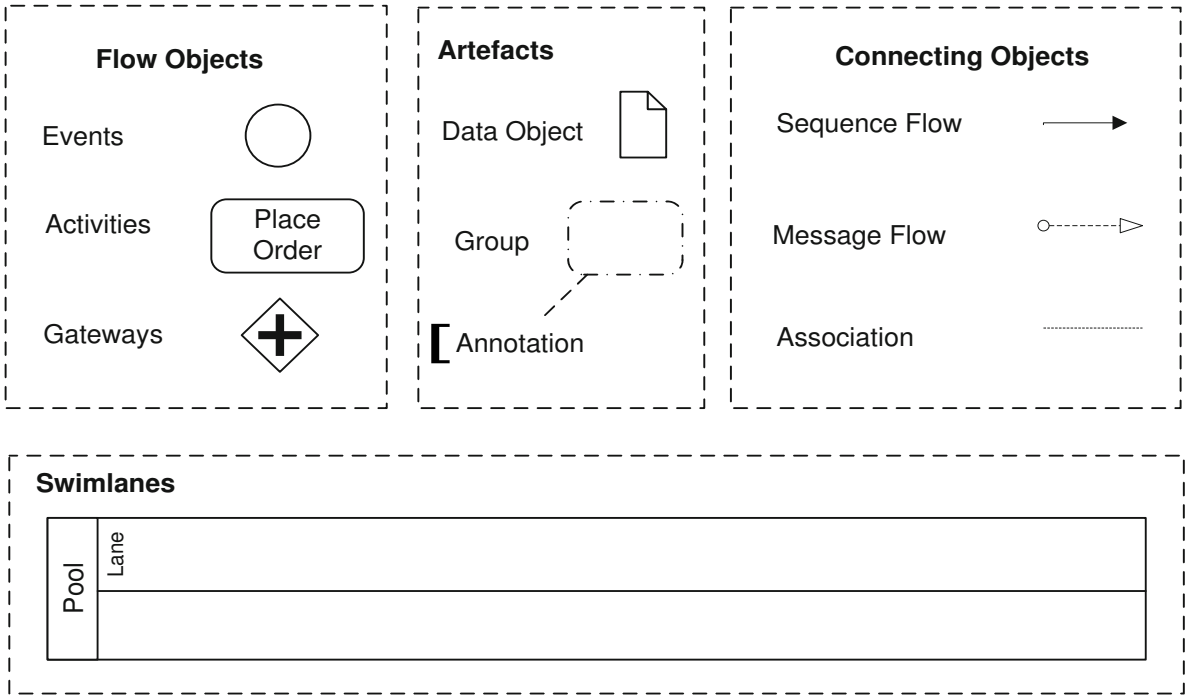

Abbildung 2.18: Elemente der BPMN, angelehnt an [Klarl 2011, S. 31]

2.7.3 Sicherheit in Geschäftsprozessen

Die IT-Sicherheit in einem Unternehmen kann aus verschiedenen Abstraktionsebenen bestehen. Eine Abstraktion beschreibt z. B. das *Enterprise Architecture Metamodel*. In diesem Modell wird die IT-Infrastruktur des Unternehmens in drei zusammenhängende, aufeinander aufbauende Schichten unterteilt [Winter und Sinz 2007], [Accorsi 2013].

- *Business-Schicht* beschreibt in erster Linie die Geschäftsprozesse, -management, -zielsetzungen sowie die organisatorische Struktur und einzuhaltende Richtlinien des Unternehmens.
- *Anwendungsschicht* definiert die Anwendungen und Dienste, welche für die Ausführung der Geschäftsprozesse nötig sind. Dazu gehört z. B. auch die Verwendung von *Service-Oriented Architecture (SOA)*[19]- oder *Cloud*[20]-Komponenten.
- *Infrastrukturschicht* umfasst die Hardware, Software und Kommunikationsdienste, welche die Nutzung von Anwendungen aus der Anwendungsschicht ermöglichen.

Bei der Einhaltung der Sicherheitsrichtlinien und Sicherheitsanforderungen im Unternehmen müssen alle diese drei Schichten berücksichtigt werden.
Die *Business-Schicht* ist allerdings ein relativ neues Forschungsgebiet in der IT-Sicherheit und wurde im Vergleich zu den anderen beiden Schichten (Anwendungsschicht und Infrastrukturschicht) bis jetzt wenig erforscht [Accorsi 2013].
Das Ziel dieses Forschungsgebietes, welches „Sicherheit in Prozessmanagement“ genannt wird, ist die Entwicklung und Integration der Sicherheits- und Kontrollmechanismen auf der Ebene der Prozessspezifikationen.
Der nachfolgende Kapitel setzt sich mit den vorhandenen Lösungsansätzen aus der wissenschaftlichen Literatur auseinander, welche sich mit den Sicherheitsaspekten in Prozessmanagement auf der Ebene der Prozessspezifikation beschäftigen.

[19] `http://www.opengroup.org/subjectareas/soa`
[20] `http://www.cloud.fraunhofer.de/`

Kapitel 3

Literaturübersicht

In der Literatur existieren verschiedene Ansätze, die Geschäftsprozessmodelle zur Abbildung von Zugriffskontroll- und Sicherheitsinformationen verwenden. Für die Geschäftsprozessmodellierung können z. B. solche grafische Modellierungssprachen wie BPMN, UML und EPK verwendet werden.
Es hat sich allerdings herausgestellt, dass viele Autoren in ihren Forschungsarbeiten für die Abbildung der Sicherheitskomponenten die BPMN (vgl. Abschnitt 2.7.2) als Modellierungsnotation bevorzugen [Rodríguez, Fernández-Medina und Piattini 2007], [Wolter und Meinel 2010], [Mülle, Stackelberg und Böhm 2011], [Klarl 2011]. Dies wird hauptsächlich damit begründet, dass die BPMN mittlerweile hohe Akzeptanz in der Industrie und somit in den Fachabteilungen der Unternehmen besitzt [Allweyer 2009, S. 9], [Klarl 2011, S. 100].
Die Fachabteilungen können beispielsweise die bereits vorhandenen BPMN-Modelle mit Zugriffskontrollinformationen anreichern. Dies hat den Vorteil, dass die bestehenden BPMN-Notationen weiterhin verwendet werden können und keine neuen, zusätzlichen Notationen für die Abbildung der Sicherheitseigenschaften in den bestehenden Geschäftsprozessmodellen eingeführt werden müssen.
Dieser Kapitel stellt mehrere Forschungsarbeiten vor, welche die Modellierung und Umsetzung der Sicherheitsanforderungen im Geschäftsprozessmanagement behandeln. Im Anschluss daran werden die möglichen und bestehenden Ansätze der automatischen Verteilung von Rollen in Geschäftsprozessmanagement untersucht.

An approach to capture authorisation requirements in business processes [Wolter und Meinel 2010]

In diesem Papier schlagen die Autoren eine Erweiterung der BPMN vor, um eine bessere Kollaboration zwischen den Sicherheitsexperten und der Fachabteilung in einer frühen Phase der Prozessmodellierung zu ermöglichen.
Die BPMN besitzt grundsätzlich keine speziellen Modellierungselemente um Zugriffskontrollinformationen darzustellen, bietet aber die Möglichkeit eigene Autorisierungsanforderungen zu definieren. Diese Autorisierungsanforderungen können im BPMN-Modell abgebildet werden, erweitern es auf eine standardisierte Weise und beeinflussen die grundlegende Semantik des Prozessmodells nicht.
Die Standard-BPMN-Spezifikation enthält bereits eine Möglichkeit die Benutzer- und Rollen durch Benutzeraktivitäten *(Activity)* und Bahnen *(Swimlane)* im Prozessmodell darzustellen. Die Autoren definieren jedoch zusätzlich verschiedene Arten von grafischen Zugriffskontrollinformationen aus dem RBAC-Modell (Rollenhierarchien, Funktionstrennung (siehe

Abschnitt 2.4.2 und Abschnitt 2.4.3)), aus dem ABAC-Modell *(Authorisation Rules)* (siehe Abschnitt 2.5.1) und einige Integritäts- und Vertraulichkeitsanforderungen.
Um rollenbasierte Zugriffsanforderungen zu spezifizieren, unterscheidet das Papier zwischen automatischen *(automatic)* und manuellen *(manual)* Aktivitäten, die von Menschen ausgeführt werden und zusätzlich durch ein Symbol gekennzeichnet sind (siehe Abb. 3.1 (a)). Die Aktivitäten können dann zu einer Rolle *(Swimlane)* hinzugefügt werden. Die *Swimlane* kann dabei durch das Hinzufügen von Text-Attributen *(Clerk)* zusätzlich bezeichnet werden (siehe Abb. 3.1 (b)).
Die Benutzer innerhalb einer Rolle *(Swimlane)* können direkt zu bestimmten Aktivitäten zugewiesen werden (siehe Abb. 3.1 (c) *(Phil)*). Die direkte Zuweisung überschreibt in dem Fall die Zuweisung zu der Rolle *(Clerk)*.
Um eine Rollenhierarchie zwischen zwei Rollen *(Swimlanes)* darzustellen, werden diese Rollen *(Swimlanes)* verschachtelt dargestellt (siehe Abb. 3.1 (d)). In dem Fall erbt die Rolle *Manager* die Rechte der Rolle *Clerk*. Um die Vererbung zwischen zwei verschachtelten Rollen zu verbieten, wird ein Anker-Zeichen eingeführt. Damit wird verdeutlicht, dass die Aktivität A *(Activity A)* von den Benutzern der Rolle *Manager* nicht ausgeführt werden darf (siehe Abb. 3.1 (e)).
Für den Fall, dass während des Prozesses die Benutzer einer bestimmten Rolle nicht zur Verfügung stehen oder falls andere unvorhergesehene Ereignisse auftreten, kann für die Zuweisung und Ausführung von Aktivitäten eine andere Rolle benannt werden (siehe Abb. 3.1 *(Admin)* (f)).

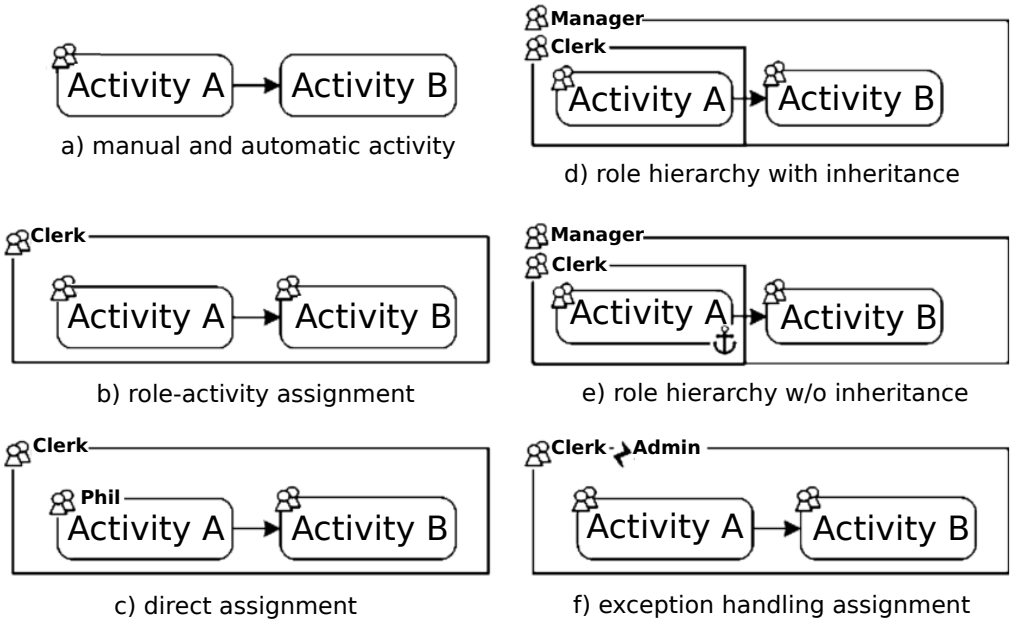

Abbildung 3.1: Rollenbasierte Erweiterungen des BPMN-Standards [Wolter und Meinel 2010]

Die Abbildung 3.2 zeigt die Zugriffsregeln, die als Textbedingungen an den Benutzeraktivitäten angebracht sind.
Die Bedingung zwischen den beiden Benutzeraktivitäten auf der linken Seite beschreibt die Funktionstrennung *(Separation of Duty)*. Dies bedeutet, dass die beiden Benutzeraktivitäten (A und B) nicht vom gleichen Benutzer ausgeführt werden dürfen.
Auf der rechten Seite ist eine attributbasierte Zugriffskontrolle abgebildet, in welcher Attribute aus dem Datenobjekt mit Informationen aus der Benutzeraktivität *(Activitiy A)* abgeglichen werden.

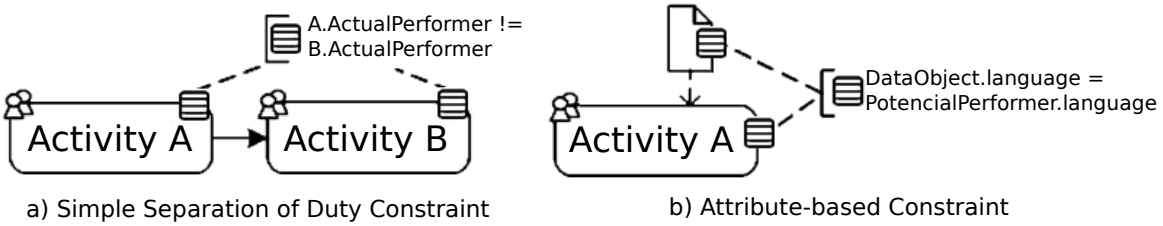

Abbildung 3.2: Modellierung der Funktionstrennung und attributbasierten Zugriffskontrolle [Wolter und Meinel 2010]

Im weiteren Verlauf des Papiers wird mit Hilfe von Petri-Netzen eine formale Semantik definiert, die das *Modelchecking* ermöglicht. *Modelchecking* wird bei Validierung und Verifikation des Modells verwendet.
Die Autoren weisen zum Schluß darauf hin, dass die vorgestellten Zugriffskontrollerweiterungen des BPMN-Modells in *Security Policies*[1] umgewandelt werden können und verweisen auf das Papier von [Wolter, Schaad und Meinel 2007], welches sich mit der Transformation in *Security Policies* beschäftigt. Die Einhaltung von diesen *Security Policies* kann dann während der Prozesslaufzeit von XACML-Komponenten *(z. B. Policy Enforcement Point (PEP), Policy Decision Point (PDP), Policy Information Point (PIP))*[2] kontrolliert werden.

A Security Language for BPMN Process Models [Mülle, Stackelberg und Böhm 2011]

Das Papier von [Mülle, Stackelberg und Böhm 2011] erweitert die BPMN ebenfalls um *Security Constraints*. Die *Security Constraints* werden als Textbeschreibungen (Annotationen) mit Hilfe von Artefakten im BPMN-Prozessmodell dargestellt. Das Ziel ist demnach solche Sicherheitsanforderungen wie Autorisierung, Authentifizierung, Audit, Vertraulichkeit und Datenintegrität als strukturierte Textannotationen im Prozessmodell deklarativ darzustellen. Die nachfolgende Tabelle 3.1 zeigt beispielsweise die Übersicht der vorgestellten Authorisierung-Constraints. Die Authorisierung-Constraints spezifizieren dabei die Rechte der Benutzer unter bestimmten Restriktionen. Mit dem Constraint «*Assignment: type = "Role" name = "$rolename"*» können z. B. zu einer Rolle (*"$rolename"* steht für eine beliebige

[1] `https://www.oasis-open.org/committees/download.php/2713/#xacml-top`
[2] `http://www.cs.wustl.edu/~jain/cse571-09/ftp/soa/#sec3.2`

Rollenbezeichnung) Aktivitäten, Gruppen von Aktivitäten, *Pools* und *Lanes* zugewiesen werden.

Auth. Constraint	Syntax
Role Assignment	«Assignment: type="Role" ... »
Assignment Mechanism	«Assignment: type="Mechanisms" ... »
User Assignment	«Assignment: type="User" ... »
Separation of Duty	«SoD: ... »
Binding of Duty	«BoD: ... »
Delegation	«Delegation: ... »
Adaptation	«Adaptation: ... »

Tabelle 3.1: Übersicht der Authorisierung-Constraints [Mülle, Stackelberg und Böhm 2011]

Nach der Erweiterung des BPMN-Modells mit *Security Constraints*, beschreiben die Autoren die Möglichkeit der Transformation. Das Ziel dieser Transformation ist es aus den annotierten *Security Constraints* spezifische Sicherheitskonfigurationen zu generieren, die während der Ausführung des Prozesses von den speziellen Sicherheitskomponenten des BPM-Systems verwendet werden (siehe Abb. 3.3).

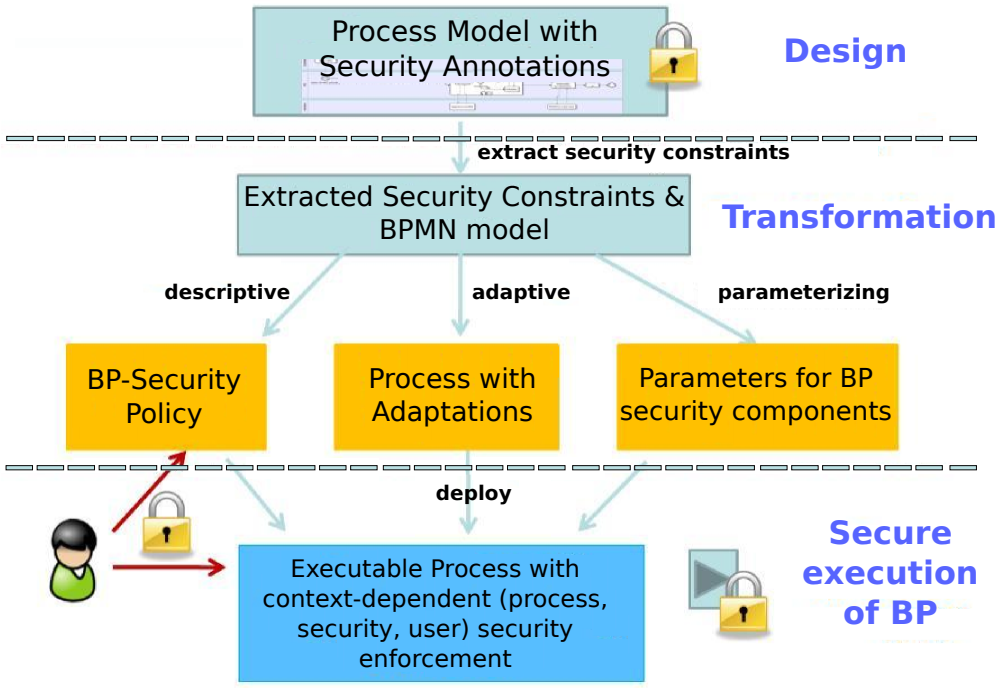

Abbildung 3.3: Drei Entwicklungsphasen eines sicheren Geschäftsprozesses [Mülle, Stackelberg und Böhm 2011]

Die Transformation von *Security Constraints* kann mindestens eine der folgenden Auswirkungen haben:

- ◇ Generierung oder Modifikation der *Security Policies* in XACML. Während der Prozessausführung überprüft dann der *PDP* die Konformität dieser *Security Policies* mit den Benutzerzugriffen in der laufenden Prozessinstanz.
- ◇ Anpassung und Erweiterung der Geschäftsprozesse mit speziellen Implementierungsbestandteilen wie z. B. «*Audit*». In dem Fall bedeutet es, dass bei der Ausführung solcher Aktivitäten, eine Protokollierung der Zugriffe durch externe *Logging*-Komponenten stattfindet.
- ◇ Parametrisierung der Aufrufe von speziellen Security-Komponenten *(PEP, PIP)*, von welchen man z. B. die Benutzerdaten oder vergangenheitsbezogene Prozessdaten (relevant für die Funktionstrennung) bekommt. Durch die «*confidentiality*»-Annotation kann man z. B. spezifizieren, dass die Übertragung der Daten zwischen den Komponenten verschlüsselt ablaufen soll.

A BPMN Extension for the Modeling of Security Requirements in Business Processes [Rodríguez, Fernández-Medina und Piattini 2007]

Eine weitere Arbeit, die sich mit Sicherheitsanforderungen während der Prozessmodellierung beschäftigt, ist [Rodríguez, Fernández-Medina und Piattini 2007]. Es wird ein BPMN-Metamodell präsentiert, welches das Standard-BPMN-Modell mit grafischen Sicherheitselementen erweitert. Die grafischen Sicherheitserweiterungen sollen vergleichbar mit den anderen vorgestellten Arbeiten, die Zusammenarbeit der Fachabteilungen und Sicherheitsexperten bei der Formulierung der Sicherheitsanforderungen unterstützen. Die eingeführten grafischen Sicherheitselemente sind ebenfalls mit den Sicherheitselementen der bereits vorgestellten Arbeiten vergleichbar, es fehlen allerdings Elemente für die Modellierung der Funktionstrennung *(Separation of Duty)*.
In [Rodríguez, Fernández-Medina und Piattini 2010] definieren die Autoren Transformationsregeln, die die modellierten *Security Constraints* aus [Rodríguez, Fernández-Medina und Piattini 2007] in die Implementierungsspezifikationen umwandeln.

3.1 Verwaltung und Verteilung von Benutzerrollen

Die durchgeführte Literaturrecherche hat gezeigt, dass die Autoren den vorhandenen BPMN-Standard mit den grafischen oder textuellen Annotationen erweitern, um die nicht vorhandenen Sicherheitsanforderungen zu spezifizieren.
Die erweiterten BPMN-Modelle werden anschließend mit Hilfe von Transformatoren und Transformationsregeln in *Security Policies* umgewandelt. Die Einhaltung dieser *Security Policies* während der Prozesslaufzeit übernehmen dann die XACML-Komponenten *(z. B. PEP, PDP, PIP)*.
Die vorgestellten Annotationen sind jedoch nicht standardisiert und besitzen somit keine

einheitlichen Elemente. Für jede dieser Annotationen, bedarf es deshalb eigener Transformatoren und den Aufbau eigener, spezifischer IT-Landschaften für die Einhaltung der *Security Policies*.
XACML gilt zwar als der Standard[3], mit dem sich Autorisierungsentscheidungen über die XACML-Komponenten, statt im Code einzelner Anwendungen steuern lassen, besitzt aber große Komplexität der Regelwerke mit der daraus resultierenden großen Anzahl an Zugriffsregeln. Die Integration und Umstellung auf XACML mit vorhandenen IT-Systemen ist somit mit erheblichem Aufwand verbunden. Aus diesem Grund setzt sich der Standard nur langsam durch und ist in der Industrie wenig verbreitet[4]. In kleineren IT-Umgebungen mit wenigen und einfach zu implementierenden Sicherheitsanforderungen wäre es somit sicherlich sinnvoller die Authorisierungsregeln im Code einzelner Anwendungen zu implementieren.
Für die Umsetzung der automatischen Verwaltung und Verteilung von Benutzerrollen, kommen somit zwei Möglichkeiten in Betracht:

- ◇ Modellierung der nötigen Anforderungen in BPMN-Modell, mit anschließender Transformation von diesen in *Security Policies* und dem Aufbau von XACML-Komponenten für die Umsetzung und Kontrolle zur Laufzeit.
- ◇ Die Implementierung der Anforderungen direkt in die Anwendungen (z. B. in die Geschäftsprozesssysteme).

Die bessere Lösung hängt letztendlich von den Anforderungen und von der Komplexität der jeweiligen Umsetzung ab.
In bestimmten Fällen können die bereits existierenden BPMN-Elemente die Anforderungen der Fachabteilungen zur automatischen Verteilung von Rollen erfüllen. Mit Hilfe von im Abschnitt 2.7.2 vorgestellten *Swimlanes* lassen sich Benutzer-Rollen innerhalb eines Unternehmens abbilden. Dazu besteht auch die Möglichkeit die einzelnen Bahnen *(Lanes)* zu verschachteln *(Nested Lanes)*, um z. B. die Rollenhierarchien darzustellen (siehe Abb. 3.4)[5]. In diesem Beispiel (Abb. 3.4) wechseln sich die Benutzerrollen während des Prozessflusses mit dem Verlauf der einzelnen Aktivitäten automatisch ab.

[3] http://www.cs.wustl.edu/~jain/cse571-09/ftp/soa/#sec3.4
[4] http://blogs.forrester.com/andras_cser/13-05-07-xacml_is_dead
[5] http://de.bpmn-community.org/tutorials/17/

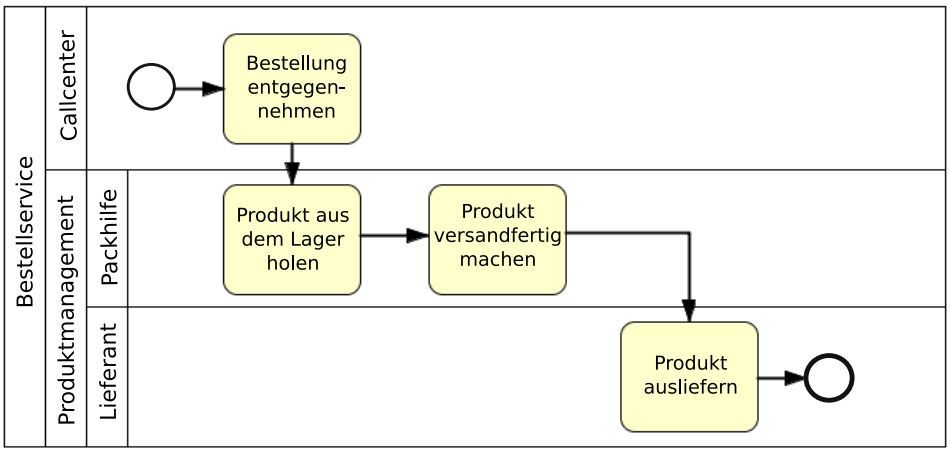

Abbildung 3.4: Umsetzung der Rollenhierarchien mit verschachtelten *Lanes* in BPMN

Eine weitere Möglichkeit automatische Verteilung von Rollen mit BPMN zu spezifizieren, ist die Verwendung von *Escalation-Event* (siehe Abb. 3.5)[6].

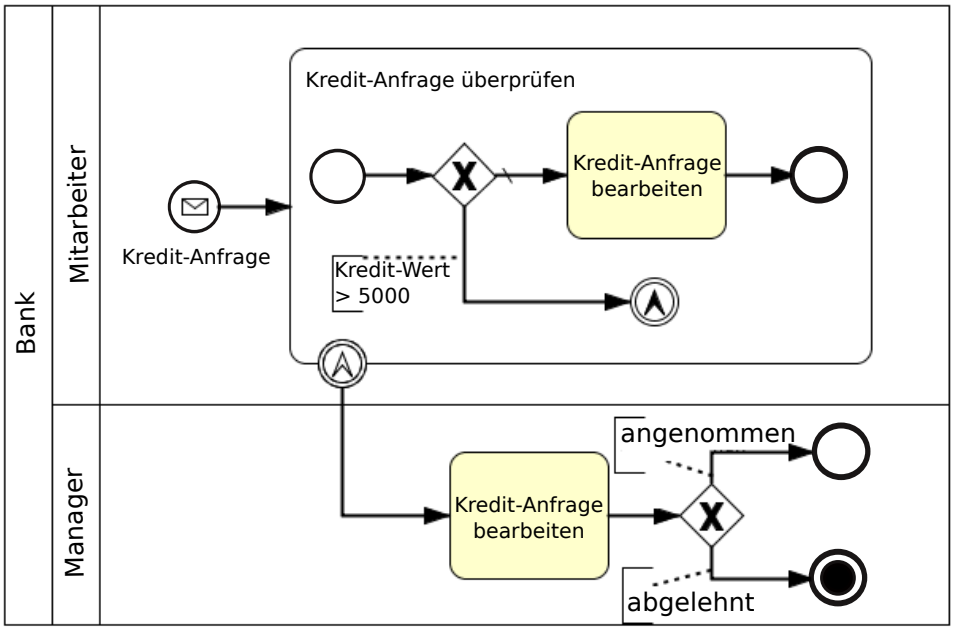

Abbildung 3.5: Beispiel mit der Eskalation zur nächsthöheren Verantwortungsstufe in BPMN

[6] http://de.bpmn-community.org/tutorials/31/

Das Beispiel in Abbildung 3.5 verdeutlicht einen Kreditvorgang in einer Bank. Der Mitarbeiter empfängt eine Kreditanfrage und kann sie nur dann bearbeiten, falls der Betrag der Anfrage 5000 nicht übersteigt. Bei einem höheren Betrag (Auslösung des *Escalation-Event*), wird der Manager involviert und entscheidet dann selbst über die Vergabe oder Ablehnung des Kredites. Mit *Escalation-Event* wird somit eine Möglichkeit der Eskalation zur nächsthöheren Verantwortungsstufe (Rolle) erreicht.
Eine andere Möglichkeit der Rollenverteilung wäre die Verwendung des *Timer Boundary Event*[7]. Ein *Timer Boundary Event* fungiert dabei als eine Stoppuhr und kann an Aktivitäten des BPMN-Modells angebracht werden. Nach dem eine solche Aktivität im Prozessfluss erreicht wird, startet die Stoppuhr automatisch und unterbricht die Ausführung dieser Aktivität nach einem festgesetzten Zeitintervall. Der Sequenzfluss wird aber abhängig von der Modellierung, von dem *Timer Boundary Event* weiter vorgesetzt.
Mit dem *Timer Boundary Event* wäre es z. B. möglich die Ausführung einer Aktivität nach einer bestimmten Zeit an eine andere Aktivität, welche sich in einer anderen *Lane* (Rolle) befindet, zu delegieren. Dies wäre z. B. in solchen Fällen sinnvoll, wenn für die Ausführung der Aktivität gerade kein Personal zur Verfügung steht, die Aktivität eine hohe Priorität besitzt und in einer bestimmten Zeit abgeschlossen werden muss (siehe. Abb. 3.6).

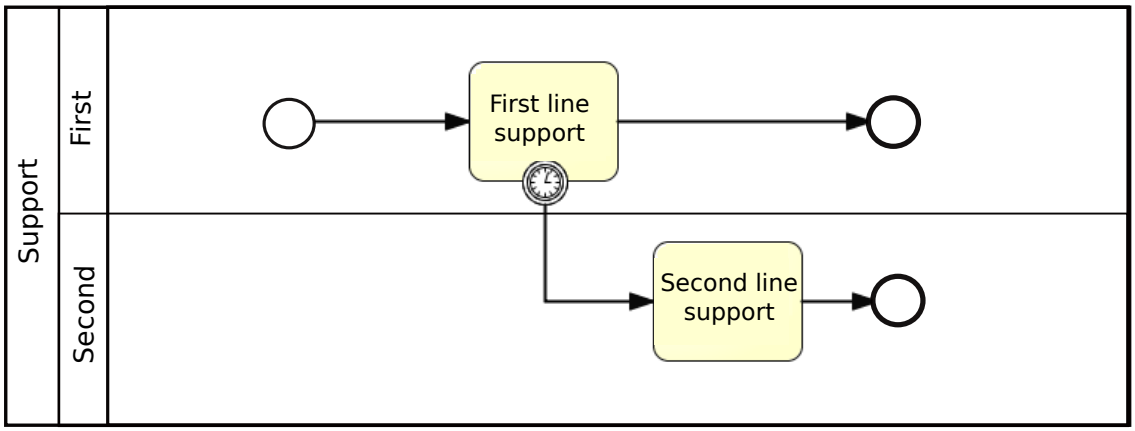

Abbildung 3.6: Beispiel einer Rollendelegation mit dem *Timer Boundary Event* in BPMN

[7] http://www.activiti.org/userguide/#bpmnTimerBoundaryEvent

Kapitel 4

Activiti als Workflow- und BPM-Plattform

In diesem Kapitel wird eine Workflow- und BPM-Plattform namens Activiti vorgestellt. Zuerst wird ein Überblick über den Funktionsumfang und die Kernkomponenten der Activiti-Plattform gegeben. Danach findet eine Untersuchung und Bewertung der Sicherheitsfunktionen des Systems statt. Im Anschluss daran wird das Sicherheitskonzept von Activiti mit der Möglichkeit der Umsetzung der Funktionstrennung *(Separation of Duty)*, Funktionsbindung *(Binding of Duty)* und der Einbindung der Plattform an ein LDAP-Server, untersucht und erweitert.

4.1 Activiti

Activiti[1] ist eine leichtgewichtige, ressourcenschonende und auf Java basierende Workflow- und BPM-Plattform, die sich auf die Modellierung und Ausführung von Geschäftsprozessen konzentriert [Rademakers 2012]. Die Plattform ist ein relativ neues Produkt auf dem Markt und basiert auf dem Know-How der zwei Hauptentwickler von jBPM (Workflow Engine von JBoss)- Tom Baeyens und Joram Barrez, welche 2010 mit der Umsetzung von Activiti bei Alfresco[2] starteten. Die erste produktiv nutzbare Version 5.0 wurde im gleichen Jahr fertiggestellt und im Rahmen der Apache-Lizenz zum freien Download[3] angeboten[4] . Activiti ist somit ein Open Source Projekt und wird mittlerweile von einer großen Community und einem Zusammenschluss von mehreren Unternehmen vorangetrieben[5].
Das Ziel von der Activiti-Gemeinschaft ist die Entwicklung einer schlanken Geschäftsprozessmanagementplattform, die den aktuellen BPMN2.0 Standard zur Modellierung von Geschäftsprozessen unterstützt. Die wichtigen Activiti-Features im Einzelnen sind[6]:

- Die Geschäftsprozessmodellierung findet in BPMN2.0 statt und soll dadurch zu einer besseren Zusammenarbeit von IT- und Fachabteilung beitragen.

[1] http://www.activiti.org/
[2] http://www.alfresco.com/
[3] http://www.activiti.org/download.html
[4] Aktuelle Version 5.12
[5] http://www.activiti.org/team.html
[6] http://www.camunda.com/wp-content/uploads/2010/12/Activit_Computerwoche_JF.pdf

- *Human-Workflow-Management* erlaubt die Steuerung und Automation von Prozessen, an denen Menschen beteiligt sind.
- Die prozessorientierte Anwendungsintegration *(Service Orchestration)* erlaubt effiziente Abarbeitung von Prozessvorgängen durch die Heranziehung von internen und externen IT-Systemen.
- Einfache Integration in eine Java-Landschaft, da die native BPMN2.0 Prozess-Engine selbst in Java geschrieben ist und über eine Java-API verfügt.
- Es lässt sich leicht mit dem SPRING-Framework[7] integrieren, ist leichtgewichtig und basiert auf einfachen Konzepten.
- Unkomplizierte Einsetzbarkeit in der Cloud (wurde im Rahmen der Arbeit auf einer *Java Platform as a Service (PaaS)*[8] getestet).
- Veröffentlichung unter Apache-Lizenz bietet eine Rechtssicherheit für die mögliche kommerzielle Nutzung von Activiti.

4.2 Kernkomponenten von Activiti

In diesem Abschnitt werden die Kernkomponenten von Activiti, wie Activiti Engine, Activiti Explorer, Activiti Modeler und Activiti Designer ausführlich dargestellt.

4.2.1 Activiti Engine

Das Herzstück der Activiti-Plattform bildet die auf Java-basierte Process-Engine, welche die BPMN2.0-Prozesse nativ durchläuft. Die Process-Engine bietet dabei die grundlegenden Funktionen um die BPMN2.0-Prozesse auszuführen wie z. B. die Erzeugung von *Workflow-Tasks*. Die Engine kann, wie schon oben erwähnt in jede Java Anwendung und jeden *Runtime Container* eingebettet werden.
Um die bessere Kollaboration zwischen der IT- und Fachabteilung zu ermöglichen und die Fachabteilung nicht mit vielen technischen Details zu überfordern, bietet die Activiti-Engine die sogenannten *event listeners* an. Die *event listeners* erlauben bei bestimmten, festgelegten Ereignissen im BPMN-Diagramm die Ausführung von *out-of-the-box* Aktionen (z. B. Ausführung von *Java-Code* oder Skripten). Die Entwickler aus der IT-Abteilung können somit Prozesse mit zusätzlichen, für die Fachabteilung im BPMN-Diagramm nicht sichtbaren, technischen Details, anreichern.
Eine andere interessante Eigenschaft bieten die benutzerdefinierten Aktivitäten *(custom activities)*. Activiti-Engine unterstützt in erster Linie den BPMN2.0-Standard, kann aber auch

[7] `http://www.springsource.org/spring-framework`
[8] `http://www.cloudbees.com/`

verschiedene, benutzerdefinierte Aktivitäten im Prozess durchlaufen. Die Fachabteilung hat dadurch die Möglichkeit spezifische, mit den benutzerdefinierten Aktivitäten angereicherte BPMN-Diagramme zu erstellen.

4.2.1.1 Prozess Engine API und Dienste *(Services)*

Über die Prozess Engine API *(Application Programming Interface)* verläuft die Interaktion zwischen Activiti und mehreren Diensten *(Services)*, welche verschiedene Workflow/BPM-Methoden beinhalten (Abb. 4.1).

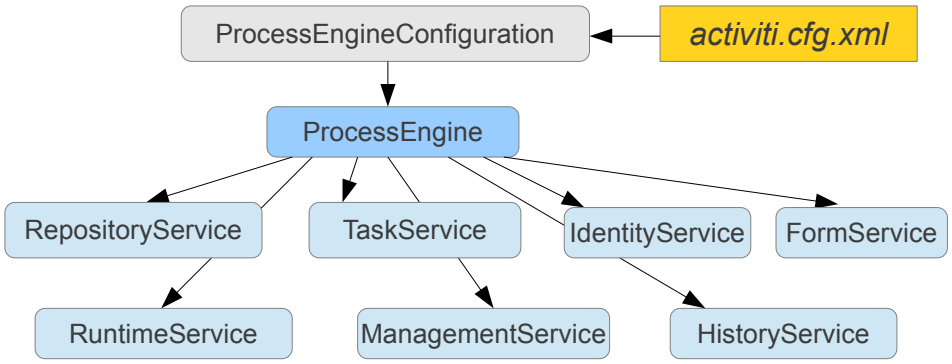

Abbildung 4.1: Prozess Engine API und Dienste *(Services)*

Zuerst scannt die *ProcessEngine-class* die vorhandene/en *activiti.cfg.xml*-Datei/en, in welchen z. B. die Anbindung an die Datenbank konfiguriert ist und startet anschließend die Prozess-Engine. Activiti unterstützt dabei mehrere Datenbank-Typen, wie H2[9], MySql[10], Oracle, Postgres[11], DB2 oder MSSQL[12] *(Microsoft SQL Server)*.
Während des Prozesses kann die Prozess-Engine dann auf mehrere Dienste *(Services)* (Abb. 4.1) zugreifen, welche im Nachfolgenden beschrieben werden.

- *RepositoryService* bietet die Verwaltung von Prozess-Definitionen und Prozess-*Deployments*. Eine Prozess-Definition entspricht dem Java-basierten Duplikat des BPMN2.0-Prozesses und enthält die Struktur und Verhaltensweise von jedem BPMN2.0-basierten Prozess-schritt.
  Ein Prozess-*Deployment* kann z. B. mehrere BPMN2.0-XML Dateien und andere, für den Prozess relevante Dateien enthalten. Während der Durchführung des Deployments, werden die Daten von der Engine geladen, inspiziert, geparst und anschließend in die Datenbank abgelegt. Ab diesem Zeitpunkt ist das *Deployment* dem System bekannt und die darin enthaltene Prozesse können dann vom *RuntimeService* gestartet werden.

[9] http://www.h2database.com/
[10] http://www.mysql.de/
[11] http://www.postgresql.org/
[12] http://www.microsoft.com/de-de/server/sql-server/2012/default.aspx

- *RuntimeService* startet die neuen Prozess-Instanzen von den Prozess-Definitionen. Von jeder Prozess-Definition können gleichzeitig mehrere Prozess-Instanzen existieren. *RuntimeService* verwaltet und speichert zudem die in der Prozess-Instanz enthaltenen Variablen, welche z. B. von *gateways* im Prozess verwendet werden können.
- *TaskService* beinhaltet Funktionen für die Erstellung, Anpassung und Ausführung von Aufgaben *(Tasks)* im Laufe des Prozesses.
- *IdentityService* bietet die Verwaltung von Benutzern und Gruppen in Activiti.
  Bei einer Zuweisung von Benutzern oder Gruppen zu einer Aufgabe *(Task)*, findet in Activiti keine interne Überprüfung statt, ob die Benutzer dem System bekannt sind oder nicht. Die eigentliche Prüfung findet dann erst während der Laufzeit bei der direkten Ausführung der Aufgabe *(Task)* statt. Dies erlaubt z. B. die Anbindung an verschiedene Verzeichnisdienste wie LDAP oder AD *(Active Directory)* von Microsoft.
- *FormService* ist ein optionaler Dienst und kann im Prozess die Verwendung von Formularen ermöglichen. Die Formulare werden in die BPMN2.0-Prozess-Definition/en mit eingebunden und können während des Prozesses verwendet werden.
- *HistoryService* bietet Speicher- , Verwaltungs- und Zugriffsmöglichkeiten auf die historischen (vergangenheitsbezogenen) Prozess-Daten. Im Laufe des Prozesses lassen sich z. B. die Dauer der Aufgaben *(Tasks)*, Verlauf der Benutzerzuweisung, Ablaufpfade der jeweiligen Prozessinstanzen, usw. protokollieren und abspeichern.
- *ManagementService* ermöglicht z. B. die Abfrage von Informationen und Metadaten über die in der Datenbank enthaltenen Tabellen.

4.2.2 Activiti Explorer

Activiti Explorer ist eine webbasierte Applikation zum Zugriff auf die Activiti Engine zur Laufzeit. Sie beinhaltet Taskmanagement, Informationen zu den Zuständen aktueller Instanzen, sowie das Reporting statistischer, historischer Daten.
Das Taskmanagement beinhaltet folgende Ansichts- und Anwendungsmöglichkeiten:

- ◇ Betrachtung der persönlichen Aufgabenliste *(Task List)*.
- ◇ Betrachtung der Aufgabenliste *(Task List)*, für die der aktueller Benutzer als ein Kandidat gilt.
- ◇ Eine neue Aufgabe *(Task)* erstellen (unabhängig vom aktiven Prozess).
- ◇ Eine Aufgabe abschließen, nachdem die Daten in ein Formular *(Task Form)* eingetragen werden.
- ◇ Delegieren von Aufgaben *(Tasks)* an verschiedene Benutzer.
- ◇ Ansicht der involvierten Benutzer zu einer bestimmten Aufgabe *(Task)*.
- ◇ Erstellung der Unteraufgaben *(Subtasks)* und Zuweisung von diesen an bestimmte Benutzer.

Zu den Zuständen aktueller Prozess-Instanzen gibt es folgende Ansichts- und Verwaltungsmöglichkeiten:

- ◇ Monitoring und aktueller Status des Activiti-Engines.
- ◇ Verwaltung der Prozess-*Deployments*.
- ◇ Verwaltung der Prozessdefinitionen.
- ◇ Inspektion der Datenbanktabellen.
- ◇ Ansicht der Log-Dateien.
- ◇ Anzeige der durchschnittlichen Transaktionslaufzeiten.

Alle Ausführungsschritte der Prozessinstanzen werden persistent in einer Datenbank abgespeichert. Von diesen historischen Ablaufinformationen können verschiedene Arten von Reports generiert werden. Die Reports sind in erster Linie für die analytische Datensicht interessant und können für die Zwecke der *Business Intelligence (BI)* (z. B. Anzahl der Verkäufe pro Region, Anzahl der zurückgewiesenen Bestellungen), oder zur Prüfung/Protokollierung *(Audit Logs)* der abgelaufenen Prozessinstanzen verwendet werden.

4.2.3 Activiti Modeler

Aktiviti Modeler ist eine Open-Source Version von KIS-BPM[13]. Es kann dazu verwendet werden standardisierte BPMN2.0-Modelle grafisch und webbasiert zu erstellen. Die BPMN2.0-Elemente können dabei ganz einfach durch die „Ziehen und Ablegen"-Methodik *(Drag and Drop)* von der Form-Repository *(Shape Repository)* (siehe Abb. 4.2 links) auf die Modellierungs-Fläche verschoben werden. Durch die relativ einfache, webbasierte Modellierung können damit beispielsweise die Fachabteilungen eines BPM-Projekts in die Prozessabstimmung einbezogen werden. Die Prozess-Dateien werden dann nach der Erstellung des BPMN2.0-Modells in einer dafür vorgesehenen Modell-Datenbank *(Database Model Repository)* hinterlegt.
Im Vergleich zur kommerziellen Version enthält es allerdings nur eine begrenzte Anzahl an Modellierungsmöglichkeiten[14]. Die Modellierung von *Pools* und *Lanes* ist z. B. mit der fehlenden Rollen-Unterstützung *(Role Based Security)* in der Open-Source-Version nicht möglich. Aus diesem Grund eignet sich Aktiviti Modeler in der Grundversion eher für kleinere Anpassungen des bestehenden BPMN2.0-Modells durch die Fachabteilungen, direkte Zuweisung *(Assignments)* von Benutzern zu den Aufgaben *(Tasks)*, oder z. B. für die Änderung der Formulareigenschaften *(Form Properties)* (siehe Abb. 4.2 rechts).

[13] `http://kisbpm.com/`
[14] `http://kisbpm.com/features.html`

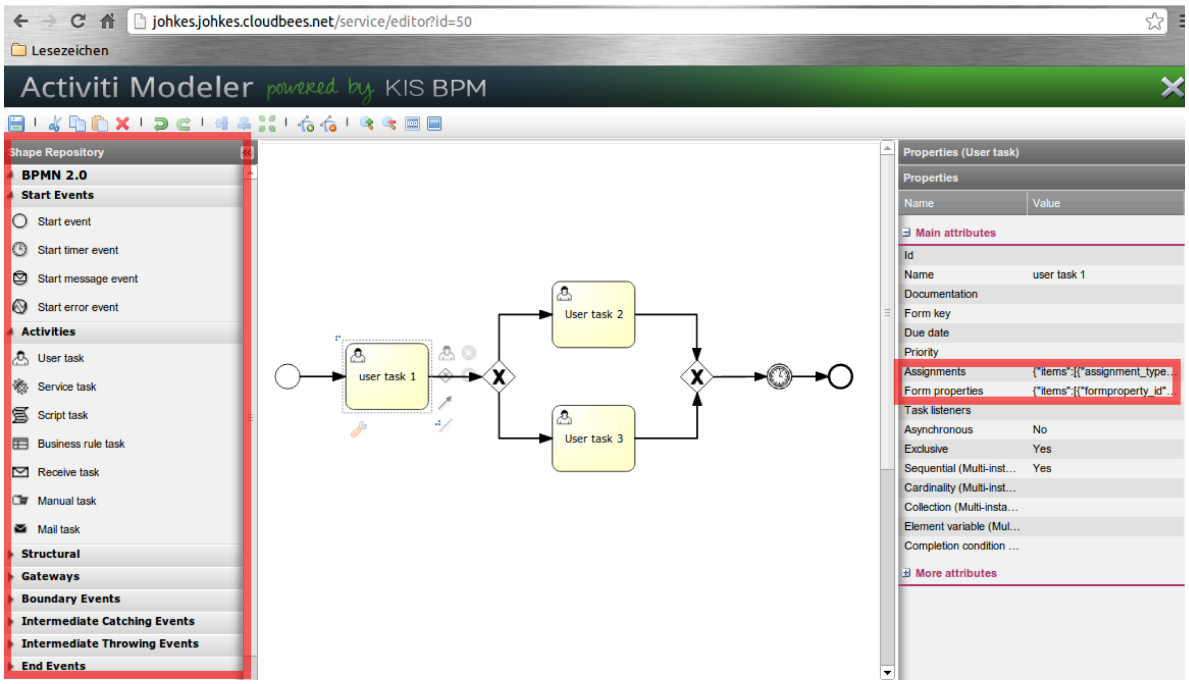

Abbildung 4.2: Activiti Modeler

4.2.4 Activiti Designer

Activiti Designer bietet eine erweiterte grafische Unterstützung und ist im Vergleich zu Activiti Modeler, hauptsächlich an die Entwickler gerichtet, da es eine BPMN-Modellierung direkt in der Eclipse[15]-Umgebung ermöglicht (siehe Abb. 4.3).
Activiti Designer kann in der Eclipse-Umgebung als Plugin[16] heruntergeladen werden und bietet zusätzlich zur grafischen Modellierung die Entwicklung, Test- und Installationsmöglichkeit *(Deployment)* von BPMN2.0-Prozessen. Es muss allerdings darauf geachtet werden, dass zur Zeit nicht jede Eclipse-Version unterstützt wird. Beispielsweise wird *Eclipse Helio* nicht unterstützt und zur Installation von Activiti Designer auf der Homepage von Activiti, die getestete *Eclipse Indigo*-Version vorgeschlagen[17]. Nach der Installation des Plugins in *Eclipse Indigo*, bietet das Activiti Designer folgende Features:

- Erstellung von Activiti-Projekten und BPMN2.0-Diagrammen.
- Bereits existierende BPMN2.0-Prozess-Dateien (BPMN2.0 XML) können vom Activiti Designer interpretiert und angezeigt werden. Dazu müssen die Dateien in das Activiti-Projekt hinzugefügt und mit dem Activiti-Diagram-Editor geöffnet werden. Für die Anzeige des Prozessdiagramms ist allerdings das Vorhandensein von BPMN DI *(BPMN*

[15] http://www.eclipse.org/
[16] http://activiti.org/designer/update/plugins/
[17] http://www.activiti.org/userguide/

Diagram Interchange) -Information in der Prozess-Datei zwingend erforderlich, da daraus die Diagramme aufgebaut werden.

- ◇ Unterstützung der testgetriebenen Entwicklung *(Test-Driven Development)* mit dem Junit[18]-Framework. Ein Unit-Test kann mit der Activiti-Konfiguration direkt in der eingebetteten In-Memory-H2-Datenbank ausgeführt werden.
- ◇ Die Erstellung des Activiti-Projektes als Maven[19]-Projekt (siehe Abb. 4.3 links). Die Entwicklung des Activiti-Projektes wird dadurch deutlich erleichtert, da damit viele Schritte wie Auflösung von Abhängigkeiten, Aufbau von standardisierten Verzeichnisstruktur, kompilieren und packen automatisiert werden können. Dazu Bedarf es allerdings der richtig-konfigurierten *pom.xml*-Datei im Hauptprojektordner. Wichtig ist dabei die richtige Referenzierung der Online-Maven-Repositorys, da von diesen *url*-Adressen die Dateien in das lokale Repository für die Verwendung kopiert werden.
- ◇ Unterstützung von *Pools* und *Lanes* (siehe Abb. 4.3 rechts). Mehrere *Pools* werden von Activiti als verschiedene Prozesse aufgefasst, aus diesem Grund wird von der Verwendung mehrerer *Pools* in einem Prozess abgeraten. In einem *Pool* können dann aber unbestimmte Anzahl von *Lanes* existieren.
- ◇ Unterstützung der Activiti-Erweiterungen, wie Mail-Task, Konfiguration von User-Tasks (siehe Abb. 4.3 unten), Script-Tasks, Task-Listeners oder Injektion der Konditionsbedingungen im Sequenz-Fluss *(Condition on Sequence Flow)* des Prozesses.
- ◇ Unterstützung von mehreren, ineinander verschachtelten Unterprozessen *(Embedded Sub Processes)*.

[18] `http://junit.org/`
[19] `http://maven.apache.org/`

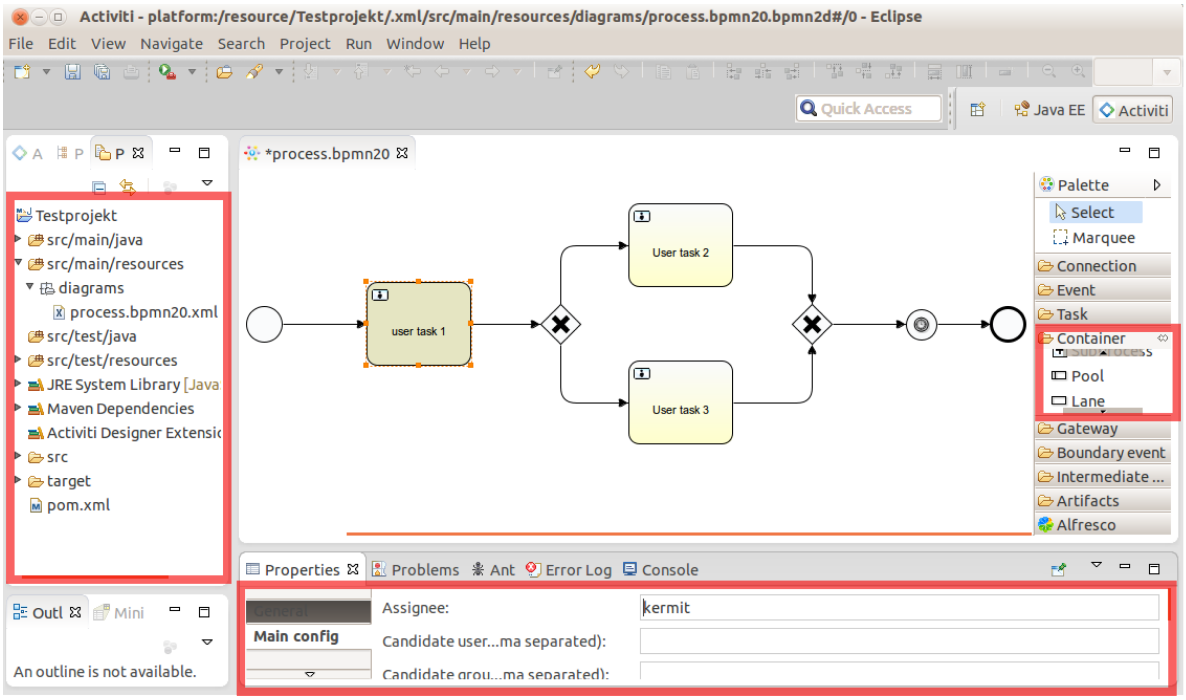

Abbildung 4.3: Activiti Designer

4.3 Sicherheitsfunktionen von Activiti

Zur Bewertung der funktionalen Sicherheit definieren die *Common Criteria (CC)* verschiedene Funktionsklassen, die im Kap. 2.1 vorgestellt wurden [Criteria 2012].
In diesem Abschnitt findet eine Untersuchung der Sicherheitsfunktionen von Activiti und die gleichzeitige Einordnung von diesen, in die von *(CC)* definierten Funktionsklassen statt.

4.3.1 Benutzerauthentifizierung

Unter Benutzerauthentifizierung versteht man die Verifizierung der Benutzeridentität.
In der *CC*-vorhandenen FIA-Klasse *(FIA_UAU (User Authentification))*, welche aus insgesamt sieben Untertypen besteht, werden die möglichen Authentifizierungsmechanismen anwendungsneutral beschrieben [Criteria 2012, S. 94].
Die Authentifizierung in Activiti entspricht dabei dem Untertyp-*FIA_UAU.2*. Dieser Untertyp legt fest, dass es vor jeglicher Aktion am System durch einen Benutzer, eine Authentifizierung stattfinden muss.
Weiterhin werden für diesen Authentifizierungstyp zwei mögliche Verwaltungsfunktionen vorgeschlagen: Verwaltung aller Authentifizierungsdaten durch einen *Administrator* und

Verwaltung der Authentifizierungsdaten durch den jeweiligen *Benutzer*. Beide Möglichkeiten der Authentifizierungsadministration sind in Activiti vorhanden.
Die Schwierigkeitsstufe und Länge des Passworts werden von Activiti allerdings nicht immer beachtet. Es ist z. B. möglich, bei einer Änderung der Zugangsdaten, ein einziges Zeichen (auch Leerzeichen) als Passwort festzulegen und sich damit zu authentifizieren. Bei einer Neuerstellung eines Benutzers in Activiti, sind aber fünf Zeichen als Passwort vorgeschrieben.

4.3.2 Benutzerautorisierung

Unter Autorisierung versteht man einen Vorgang der Rechtezuweisung. Die Rechte berechtigen den Benutzer anschließend eine bestimmte Aktion durchzuführen oder nicht.
In Activiti können einzelne Benutzer oder Gruppen von Benutzern zu den Aufgaben *(Tasks)* im Prozess zugewiesen werden. Die Gruppen tragen dabei einen Rollencharakter und bestehen aus zwei vordefinierten Gruppen-Typen: *security role* und *assignment*.
Die genaue Beschreibung und Funktionen dieser Gruppen-Typen ist auf der Activiti-Homepage[20] leider nicht vorhanden. Für die Ansicht und Verwaltung der Datenbanktabellen, Deployments, aktiven und ausgesetzten Prozessen, Benutzern und Gruppen in Activiti Explorer hat sich herausgestellt, dass man der *security-role* als Gruppen-Typ angehören muss. Zusätzlich muss der Name der Gruppe zwingend „admin“ sein, da sonst die Verwaltungsfunktionen in Activiti Explorer nicht mehr vorhanden sind.
Die Zuweisung zu der Gruppe des Gruppen-Typs *assignment* bedeutet, dass während des Ablaufs einer Prozessinstanz, nur die Mitglieder einer solchen Gruppe, bestimmte Aufgabe/en *(Task/s)* ausführen können. Diese Gruppe muss aber davor in der Prozess-Definition für diese Aufgabe/en *(Task/s)* als „candidate group“ deklariert worden sein.
In der *Common Criteria (CC)* entspricht die Unterklasse *(Security Management Roles (FMT_SMR))* der Gruppen-Funktionen von Activiti am besten [Criteria 2012, S. 116]. Die Unterklasse schlägt auch die Verwaltungsfunktion/en für die Gruppen-/Rollen- Verwaltung vor. Die Verwaltung der Gruppen ist in Activiti implementiert und nur den Benutzern aus der Gruppe des Gruppen-Typs *security-role* mit der Bezeichnung „admin“ gestattet.
Die Unterklasse *(Security audit review (FAU_SAR))* der Klasse *(Security audit (FAU))* definiert die Anforderungen an die Audit-Tools (z. B. die Möglichkeit der Ansicht der historischen Ablaufinformationen), welche die Benutzer bei der Überprüfung der historischen Daten unterstützen sollen.
Activiti bietet, wie bereits im Abschnitt 4.2.2 erwähnt, die Möglichkeit der Generierung von Reports, in welchen die historischen Ablaufinformationen der Prozess-Instanzen ersichtlich sind und entspricht dabei den Anforderungen der *(Security audit review (FAU_SAR))* [Criteria 2012, S. 37]. Aus den Ablaufinformationen der Prozessinstanzen ist dann auch ersichtlich, welche Benutzer oder Gruppen bestimmte Aufgaben *(Tasks)* ausgehführt haben.

[20] `http://www.activiti.org/userguide/`

4.3.3 Datenintegrität

Datenintegrität umfasst die Maßnahmen, die dafür Sorge tragen, dass die geschützten Daten während der Verarbeitung nicht beschädigt oder verändert werden können.
Activiti bietet keinen speziellen Datenintegritätsschutz während der Ausführung von Prozess-Instanzen, speichert aber beim Erreichen von Aufgaben *(Tasks)* die Prozessdaten persistent in einer Datenbank und geht dabei in einen Wartestatus *(Wait State)* über. Erst nach dem ein Benutzer die Ausführung der Aufgabe *(Task)* aktiv startet, werden die Daten aus der Datenbank wieder abgefragt.
Während der Wartezeit *(Wait State)* kann man aber davon ausgehen, dass die Datenintegrität durch die verwendete Datenbank gewährleistet ist, da die meisten relationalen Datenbanksysteme diesen Schutz enthalten.
Die *Common Criteria (CC)* definiert dafür eine Unterklasse *Stored data integrity (FDP_SDI)* der Klasse *User data protection (FDP)*, die Anforderungen an den Integritätsschutz der gespeicherten Benutzerdaten stellt und welche Activiti durch die Anbindung an eine relationale Datenbank, erfüllt [Criteria 2012, S. 81].

4.3.4 Zurücksetzen der einzelnen Verarbeitungsschritte *(Rollback)*

Activiti führt die Prozesse transaktionsbasiert aus, d. h. jede Transaktion (Prozessschritt) wird entweder fehlerfrei und vollständig oder gar nicht ausgeführt. Die fehlerhafte Transaktionen werden dabei abgebrochen und die bereits durchgeführte Änderungen werden in der Datenbank rückgängig gemacht *(Rollback)*.
Auf der Abbildung 4.4 ist ein Segment des BPMN-Prozesses zu sehen, auf dem die BPMN-Elemente: *UserTask*, *ServiceTask* und *TimerEvent* enthalten sind. Die Ausführung der *UserTask* (Bereitstellung der Lieferadresse) und die anschließende Validierung der Adresse durch die *ServiceTask*, gehören in diesem Beispiel zu einer Arbeitseinheit *(Unit of Work (1st TX))*. Dies bedeutet, dass in diesem Fall, bei einer fehlerhaften Ausführung der *ServiceTask* (2), ein automatischer *Rollback* zur *UserTask* (3) stattfindet. Die bereits durchgeführten, möglichen Änderungen werden dabei rückgängig gemacht und der Ausgangszustand der *UserTask* wird wiederhergestellt.
In der *Common Criteria (CC)* beschreibt die Unterklasse *(Rollback (FDP_ROL))* der Funktions-Klasse *(User Data Protection (FDP))* die Möglichkeit der Zurücksetzung einer bestimmten Anzahl an Arbeitsschritten *(FDP_ROL.1)* und entspricht dabei der Möglichkeit des *Rollbacks* in Activiti [Criteria 2012, S. 79].

4.3.5 Zusammenfassung der Sicherheitsfunktionen von Activiti

Abschließend lässt sich sagen, dass Activiti aus den in CC-definierten Sicherheitskriterien nur einige wenige Sicherheitsanforderungen erfüllt. Viele der existierenden Sicherheits-Anforderungsklassen werden von Activiti gar nicht, oder nur zum Teil durch bestimmte Unterklassen und Untertypen bedient.

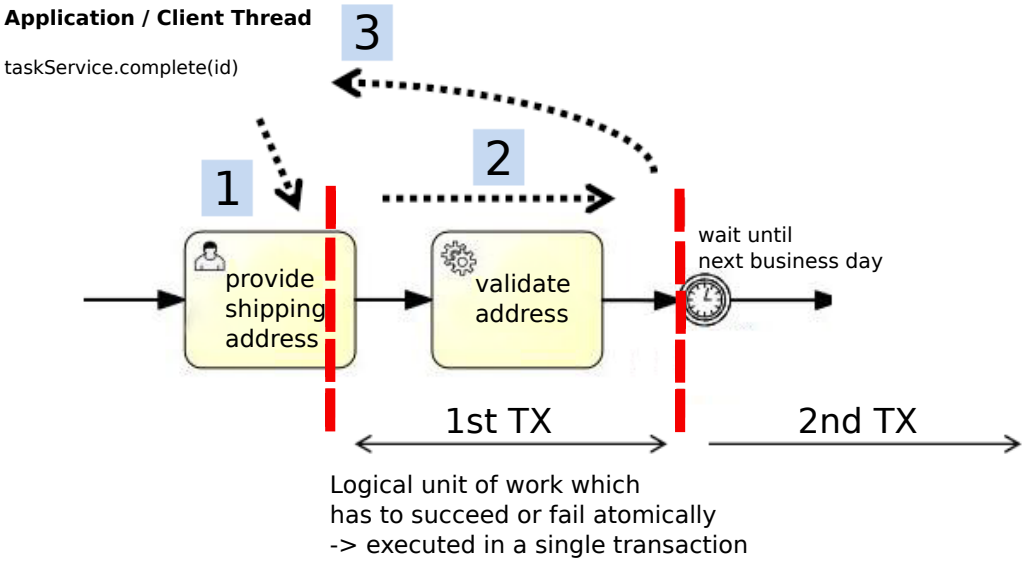

Abbildung 4.4: *Rollback* in Activiti

Die nachfolgende Tabelle 4.1 fasst die Ergebnisse der Bewertung der Sicherheitsfunktionen von Activiti anhand von CC zur besseren Übersicht, zusammen.

Anforderungsklasse aus CC	Unterklasse	Untertyp	Activiti
FAU: Security Audit	Security audit review (FAU_SAR)	Alle	✓
FCO: Communication	-	-	-
FCS: Cryptographic Support	-	-	-
FDP: User Data Protection	Stored data integrity (FDP_SDI)	Alle	✓
	Rollback (FDP_ROL)	FDP_ROL.1	✓
FIA: Identification and Authentication	User Authentification (FIA_UAU)	FIA_UAU.2	✓
FMT: Security Management	Security Management Roles (FMT_SMR)	Alle	✓
FPR: Privacy	-	-	-
FPT: Protection	-	-	-
FRU: Resource Utilisation	-	-	-
FTP: Trusted Path/Channels	-	-	-

Tabelle 4.1: Bewertung der Sicherheitsfunktionen von Activiti anhand von CC

4.4 Erweiterung des Sicherheitskonzepts von Activiti

Die internen Regeln oder *Compliance*-Richtlinien der Unternehmen können wechselseitigen Ausschluss von Rollen, durch die bereits im Abschnitt 2.4.3 vorgestellte Funktionstrennung *(Separation of Duty)*, erfordern. Dies ist z. B. dann der Fall, wenn bestimmte Aktivitäten innerhalb eines Geschäftsprozesses nicht von einer Person oder Organisationseinheit durchgeführt

werden dürfen.
Im BPMN2.0-Standard, welcher zur grafischen Modellierung und zum Ablauf von Geschäftsprozessen in Activiti verwendet wird, existiert keine explizite Möglichkeit der Modellierung von Sicherheitsaspekten [Mülle, Stackelberg und Böhm 2011, S. 2]. Dies bedeutet, dass der Standard keine speziellen BPMN2.0-Elemente zur Verfügung stellt, mit welchen man z. B. die Funktionstrennung *(Separation of Duty)* oder Funktionsbindung *(Binding of Duty)*, grafisch darstellen kann.
Im Folgenden wird die Anwendung beider Einschränkungen in Bezug auf Gruppen/Rollen in Activiti untersucht.

4.4.1 Statische Funktionstrennung

Bei der statischen Funktionstrennung muss bereits zum Zeitpunkt der Zuweisung zu einer Geschäftsrolle überprüft werden, ob dadurch eine Forderung zur Funktionstrennung verletzt wird. Ist dies nicht der Fall, können die Gruppen/Rollen gleichzeitig dem Benutzer zugewiesen werden.
Eine nicht zulässige Kombination wäre z. B. die gleichzeitige Zuweisung eines Benutzers in die beiden Gruppen/Rollen: „Genehmigung“ und „Auszahlung“ eines Kreditantrags. In diesem Beispiel muss bei der statischen Funktionstrennung darauf geachtet werden, dass der Schnitt der beiden Gruppen/Rollen leer bleibt: $\text{„}Genehmigung\text{“} \cap \text{„}Auszahlung\text{“} = \emptyset$,
oder allgemeiner ausgedrückt:
$\forall R_i, R_j \in Role, \forall s \in Subject : (s \in Member(R_i) \land s \in Member(R_j)) \Rightarrow (R_i, R_j) \notin SSOD$,
d. h. kein Benutzer *(Subjekt)* darf gleichzeitig in beiden Gruppen/Rollen existieren, um Anforderungen an $SSOD$ zu erfüllen.
Die manuelle Zuweisung von Benutzern zu Gruppen/Rollen muss aufgrund dessen in Activiti sorgfältig durchdacht sein. Bei einer nacheinander folgenden Ausführung von zwei kritischen Aufgaben *(Tasks)*, muss es deshalb für jede dieser Aufgaben *(Tasks)* zwingend eine eigene Gruppe/Rolle von Benutzern existieren.

4.4.2 Dynamische Funktionstrennung

Die Einschränkungen bei statischer Funktionstrennung werden zum Zeitpunkt der Zuweisung der jeweiligen Benutzer zu Gruppen/Rollen wirksam. Bei der dynamischen Funktionstrennung sind die Anforderungen an einen wechselseitigen Ausschluss von Gruppen/Rollen jedoch auf den Kontext bezogen, in dem die Ausübung der Gruppe/Rolle wirksam wird. Die Anforderung an die dynamische Funktionstrennung anhand des angeführten Beispiels mit dem Kreditantrag, besagt nicht zwangsläufig die strikte Trennung der beiden Gruppen/Rollen zur Genehmigung und Auszahlung eines Kredites. Die Anforderung verlangt eher eine Trennung der beiden Tätigkeiten in Bezug auf den selben Kontext, in unserem Fall auf den selben Kreditantrag. Ein Benutzer kann daher in beiden Gruppen/Rollen enthalten sein und verletzt die fachlichen Anforderungen an DSOD in unserem Beispiel nur, wenn er nach der *Genehmigung* des Kreditantrages, die *Auszahlung* denselben Kreditantrages freigibt.

Die Umsetzung der Kontextabhängigkeit in einem Anwendungssystem ist allerdings nur dann möglich, wenn die Möglichkeit der Speicherung und Auswertung der Tätigkeiten, sowie die Möglichkeit der Heranziehung der digitalen Identitäten, im jeweiligen Kontext gegeben ist [Klarl 2011].
Die Umsetzung der dynamischen Funktionstrennung ist in Activiti durch die Einrichtung des sogenannten *taskListeners* und durch die Verwendung des im Kapitel 4.2.1.1 vorgestellten *HistoryService*, möglich.

TaskListener

Ein *taskListener* wird in Activiti verwendet, um beim Auftreten eines bestimmten, aufgabenbezogenen Ereignisses einen benutzerdefinierten Java-Code auszuführen. Ein *taskListener* darf in der Prozess-Definition von Activiti nur als Kindelement einer *userTask* und zusätzlich als Kindelement des BPMN2.0 - *extensionElements* vorkommen, da er eine „Activiti-spezifische" Erweiterung des BPMN2.0 - Standards darstellt (siehe Listing 4.1).

Listing 4.1 Beispiel eines *taskListener* in Activiti

```
1 <userTask id="myTask" name="My Task" >
2   <extensionElements>
3     <activiti:taskListener event="assignment" class="org.activiti.
          MyTaskCreateListener" />
4   </extensionElements>
5 </userTask>
```

Ein *taskListener* kann folgende Attribute beinhalten:

- *event* beschreibt eine der drei Ereignismöglichkeiten, nach welchen der *taskListener* aufgerufen wird.
  - *create*: tritt auf, wenn eine Aufgabe *(Task)* erstellt ist und alle Aufgabeneigenschaften gesetzt sind.
  - *assignment*: tritt auf, bei einer Zuweisung eines Benutzers *(User)* zu einer Aufgabe *(Task)* (siehe Listing 4.1, Zeile 3).
  - *complete*: tritt auf, wenn die Aufgabe *(Task)* beendet wird.
- *class*: die Java-Klasse, welche beim Auftreten des *event*-Ereignisses aufgerufen wird (siehe Listing 4.1, Zeile 3).

Die eigentliche Logik der dynamischen Funktionstrennung muss letztendlich in der aufgerufenen Java-Klasse des *taskListener* implementiert und bei einer Zuweisung *(event = assignment)* der Aufgabe *(Task)* zu einer Person aufgerufen werden. Beim Aufruf der Java-Klasse wird mit Hilfe des *HistoryService* der Verlauf der Aufgabenzuweisungen während des Prozesses festgestellt. Damit kann bei einer Zuweisung eines Benutzers zu einer Aufgabe (durch den

Vergleich der Benutzer-IDs) überprüft werden, ob dieser Benutzer eine vorherige (kritische) Aufgabe *(Task)* ausgeführt hat. Versucht dann ein Benutzer zwei Aufgaben *(z. B. Genehmigung und Auszahlung)* nacheinander auszuführen, wird die Zuweisung dieses Benutzers zur Aufgabe *Auszahlung*, von Activiti verwehrt.
Nachfolgend wird die Vorgehensweise der Umsetzung der DSOD, mit dem bereits angeführten Beispiel mit der *Genehmigung* und *Auszahlung* eines Kredits (siehe Abb. 4.5), ausführlich beschrieben. Die Modellierung der möglichen Ablehnung der Genehmigung wird in diesem Beispiel, aus Übersichtlichkeitsgründen, weggelassen.

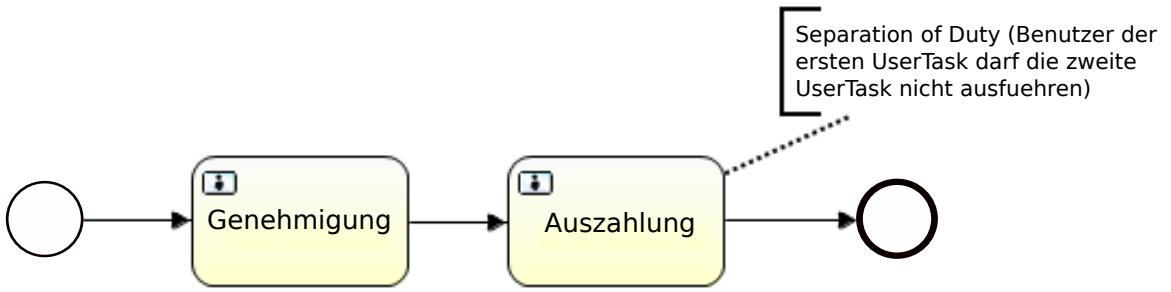

Abbildung 4.5: Funktionstrennung während der Ausführung von zwei Aufgaben *(Tasks)*

Die Abbildung 4.5 besteht aus insgesamt acht BPMN2.0-Elementen: zwei *userTasks*, einem *StartEvent*, einem *EndEvent* und drei *SequenceFlow*-Linien, die die Elemente miteinander verbinden. Die Text-*Annotation*, die durch eine punktierte Linie mit der *userTask-Auszahlung* verbunden ist, ist ebenfalls ein Teil des BPMN2.0-Standards und wird meistens zur zusätzlichen Beschreibung der Prozess-Elemente verwendet.
Bei der Umsetzung der DSOD-Anforderung muss in diesem Fall sichergestellt werden, dass die zweite *userTask-Auszahlung* einen *TaskListener* beinhaltet, welcher bei einer Zuweisung eines Benutzers zur Prüfung der Funktionstrennung eine Java-Klasse aufruft.
Der nachfolgende Listing 4.2 enthällt die Prozessdarstellung des Kreditantrags aus (Abb. 4.5) in XML[21], in welcher der integrierte *TaskListener* (Zeile 8) enthalten ist und welcher bei der Zuweisung eines Benutzers zur *userTask-Auszahlung*, die Java-Klasse *DSOD* aufruft *(class=„DSOD“)*.
Beim Aufruf der Java-Klasse - „DSOD“ findet ein Vergleich der Benutzer-IDs statt. Bei einer Übereinstimmung der Benutzer-IDs startet eine Ausnahmebehandlung von Activiti, in welcher der jeweilige Benutzer mit einer Textmeldung darauf aufmerksam gemacht wird, dass er die vorgenommene *Auszahlung* des Kredits nicht durchführen darf (siehe List. 4.3 und Abb. 4.6).

Die gleichzeitige Zuweisung der beiden Aufgaben (*Genehmigung* und *Auszahlung*) zur einer Rolle/Gruppe *(activiti:candidateGroups="kreditbearbeitung")* stellt in diesem Fall kein Problem dar (siehe List. 4.2, Zeile 4 und 6), da der Vergleich der Benutzer-IDs letztendlich sicherstellt,

[21] `http://www.w3.org/XML/`

Listing 4.2 Prozessdarstellung des Kreditantrags in XML

```
<process id="DSOD" isExecutable="true">
    <startEvent id="theStart"></startEvent>
    <sequenceFlow id="sequenceFlow-284a92d5-d8ba-4535-90d9-dd226cdec6ce"
        sourceRef="theStart" targetRef="Genehmigung"></sequenceFlow>
    <userTask id="Genehmigung" name="Genehmigung" activiti:candidateGroups
        ="kreditbearbeitung"></userTask>
    <sequenceFlow id="sequenceFlow-1ea2364a-c24a-402d-9fb4-b5337d957fb7"
        sourceRef="Genehmigung" targetRef="Auszahlung"></sequenceFlow>
    <userTask id="Auszahlung" name="Auszahlung" activiti:candidateGroups="
        kreditbearbeitung">
      <extensionElements>
        <activiti:taskListener event="assignment" class="DSOD">
        </activiti:taskListener>
      </extensionElements>
    </userTask>
    <sequenceFlow id="sequenceFlow-680f02fc-21a0-41b9-9a89-881a82f9e0a2"
        sourceRef="Auszahlung" targetRef="theEnd"></sequenceFlow>
    <endEvent id="theEnd"></endEvent>
    <textAnnotation id="textannotation1">
      <text>Separation of Duty (Benutzer der ersten UserTask darf die
          zweite UserTask nicht ausfuehren)</text>
    </textAnnotation>
    <association id="association1" sourceRef="secondTask" targetRef="
        textannotation1"></association>
  </process>
```

Listing 4.3 Auschnitt aus der Java-Klasse - „DSOD“

```
/* Abfrage und Zuweisung der Benutzer-ID fuer die Auszahlung */
String BenutzerAusz = delegateTask.getAssignee();
/* Abfrage und Zuweisung der Benutzer-ID fuer die Genehmigung */
String BenutzerGen = historicTask.getAssignee();

/* Vergleich der beiden Benutzer-IDs. Falls gleich -> Ausnahmebehandlung */
if ((BenutzerAusz.equals(BenutzerGen))) {
    throw new ActivitiException("Sie duerfen diese Aufgabe nicht ausfuehren
        , da Sie selbst die Genehmigung zur Auszahlung erteilt haben!");
```

dass die Anforderung an die DSOD erfüllt ist und die beiden Aufgaben im gleichen Kontext *(Kreditantrag)* nicht von einer einzigen Person ausgeführt werden können.

4.4.3 Funktionsbindung *(Binding of Duty)*

Mit der Aufgabenbindung *(Binding of Duty)* ist das Gegenteil der dynamischen Funktionstrennung *(dynamic Separation of Duty)* gemeint. Manche Aufgaben oder Tätigkeiten in

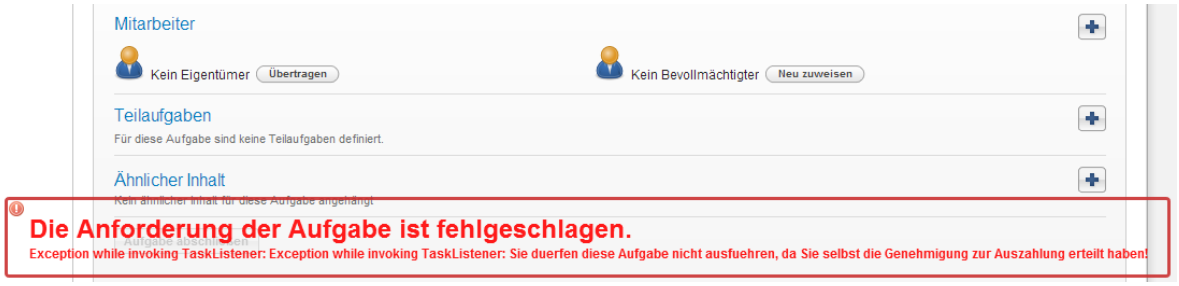

Abbildung 4.6: Die Anforderung der Aufgabe in Activiti Explorer wird verweigert

Geschäftsprozessen dürfen nur von einer bestimmten Person durchgeführt werden. Dies kann z. B. wegen den speziellen fachlichen Anforderungen oder aus Datenschutzgründen geschehen.
Die Umsetzung der Funktionsbindung in Activiti ist der Umsetzung der dynamischen Funktionstrennung sehr ähnlich. Der wesentliche Unterschied liegt nur in der Implementierung der vom *TaskListener* aufgerufenen Java-Klasse. In der Java-Klasse muss im Vergleich zur dynamischen Funktionstrennung sichergestellt werden, dass die Benutzer-IDs der beiden, nacheinander folgenden Aufgaben *(Tasks)*, identisch sind (siehe List. 4.4). Sind die Benutzer IDs verschieden, wird der Benutzer mit einer Textmeldung (vglb. Abb. 4.6) darauf aufmerksam gemacht, dass er wegen der Funktionsbindung der beiden Aufgaben die zweite Aufgabe nicht erledigen darf.

Listing 4.4 Auschnitt aus der Java-Klasse - „BOD“

```
1 /* Abfrage und Zuweisung der Benutzer-ID fuer die erste userTask */
2 String BenutzerErsteTask = delegateTask.getAssignee();
3 /* Abfrage und Zuweisung der Benutzer-ID fuer die nachfolgende userTask */
4 String BenutzerZweiteTask = historicTask.getAssignee();
5
6 /* Vergleich der beiden Benutzer-IDs. Falls unterschiedlich ->
      Ausnahmebehandlung */
7 if (!(BenutzerErsteTask.equals(BenutzerZweiteTask))) {
8     throw new ActivitiException("Sie duerfen diese Aufgabe nicht ausfuehren
          , da Sie damit die Funktionsbindung verletzen!");
```

4.5 Benutzerverwaltung in Activiti mittels LDAP

Zur Abfrage und Verwaltung der Benutzer verwendet *Activiti-Engine* als Standard-Einstellung eigene Tabellen in der vorkonfigurierten *In-Memory*-Datenbank[22] von Activiti. In heterogenen

[22] http://www.h2database.com/

IT-Umgebungen vieler Unternehmen existieren jedoch meistens eigene *Identity-Management*-Komponente, welche die Verwaltung der Benutzer durch zentrale Verzeichnisdienste wie LDAP- oder *Microsoft Active Directory (AD)* - Server betreiben.
Activiti stellt in der Version 5.12 keine konkrete Lösung der Anbindung an ein Verzeichnisdienst zur Verfügung, bietet jedoch durch die im Abschnitt 4.2.1.1 beschriebene Eigenschaft (Abfrage der Benutzer erst während der Laufzeit) eine Möglichkeit der Anbindung an ein von Activiti getrennt verwaltetes Benutzermanagementsystem. In diesem Abschnitt wird nach den LDAP-Grundlagen die Umsetzung der Anbindung von Activiti an ein solches Verzeichnisdienst untersucht und beschrieben.

4.5.1 LDAP *(Lightweight Directory Access Protocol)*

LDAP steht für *Lightweight Directory Access Protokol* und ist ein Anwendungsprotokoll aus der Netzwerktechnik [Widmer 2008]. LDAP erlaubt die Abfrage und die Modifikation der Daten eines Verzeichnisdienstes. Ein Verzeichnisdienst stellt dabei bestimmte Daten zur Verfügung, die hierarchisch aufgebaut sind und sich verteilt in einem Netzwerk befinden können. Ein Verzeichnisdienst ist in seiner Funktionsweise mit einer Datenbank vergleichbar, unterscheidet sich jedoch von den relationalen Datenbanken in folgenden Punkten [Plötner und Wendzel 2012]:

- ◇ Verzeichnisdienst ist grundsätzlich für Leseoperationen optimiert. Häufiges Einfügen und Aktualisieren von Verzeichniseinträgen kann unter Umständen aufwändig sein.
- ◇ Verzeichnisdienste stellen spezifische und erweiterte Suchfunktionen von Einträgen zur Verfügung.
- ◇ Die Datenstrukturen, welche als *Schema* bezeichnet werden, können abhängig von den lokalen Anforderungen erweitert werden.
- ◇ Die Nutzung der offenen Standards, wie RFC 4510[23] / RFC 4511[24], um das Zusammenspiel der Implementierungen zwischen den verschiedenen Herstellern zu ermöglichen.
- ◇ Die Nutzung von speziellen Replizierungstechniken zur verteilten Speicherung von Daten (z. B. Aufteilung nach organisatorischen Einheiten), welche eine effiziente Skalierung ermöglichen.

Häufige Anwendungsbereiche von Verzeichnisdiensten sind Authentifizierungs- und Autorisierungsvorgänge. Aufgrund der flexiblen Struktur können aber jede Art von Daten in einem Verzeichnisdienst abgebildet werden (z. B. *Domain Name System (DNS)*). Beispiele für Implementierung von Verzeichnisdiensten sind openLDAP[25], eDirectory[26], AD und

[23] http://tools.ietf.org/html/rfc4510
[24] http://tools.ietf.org/html/rfc4511
[25] http://www.openldap.org/
[26] https://www.netiq.com/products/edirectory/

ApacheDS[27].

4.5.2 Datenstruktur im LDAP-Verzeichnis

Ein Verzeichnisdienst besitzt eine hierarchische Datenstruktur zur Speicherung von Daten. Die Datenstruktur ist baumartig aufgebaut und jeder Baumknoten, sowie jedes Blatt stellen einen Verzeichniseintrag (LDAP-*Objekt*) innerhalb der Baumstruktur dar. Dieser Baum wird auch als *DIT* bezeichnet. Das Beispiel in Abb. 4.7 zeigt die Verzeichnisstruktur einer fiktiven Firma, welche aus zwei Domänenkomponenten *(domainComponent (dc))*, zwei Organisationseinheiten *(organizationalUnit (ou))* und vier allgemeinen Namen *(Common Name (cn))* besteht. Jeder Verzeichniseintrag (LDAP-*Objekt*) gehört normalerweise zu mehreren Klassen, welche verschiedenen Objekten gleichartige Attribute zuweisen. Ein LDAP-*Objekt* besitzt mehrere Attribute und wird eindeutig durch den vollständigen Pfad zum gewünschten Eintrag *(Distinguished Name (DN))* identifiziert. Der vollständige Name für den fiktiven Mitarbeiter *user_1* in der Abb. 4.7 lautet in dem Fall *cn=user_1,ou=people,dc=examplecorp,dc=com*. Einzelne Knoten im Pfad dieses Eintrages bezeichnet man als *Relative Distinguished Name (RDN)*. Der vollständige Name des *user_1* wird zusammengesetzt, indem dem RDN *cn=user_1* der *DN* des vorhergehenden Eintrags *ou=people,dc=examplecorp,dc=com* hinzugefügt wird.

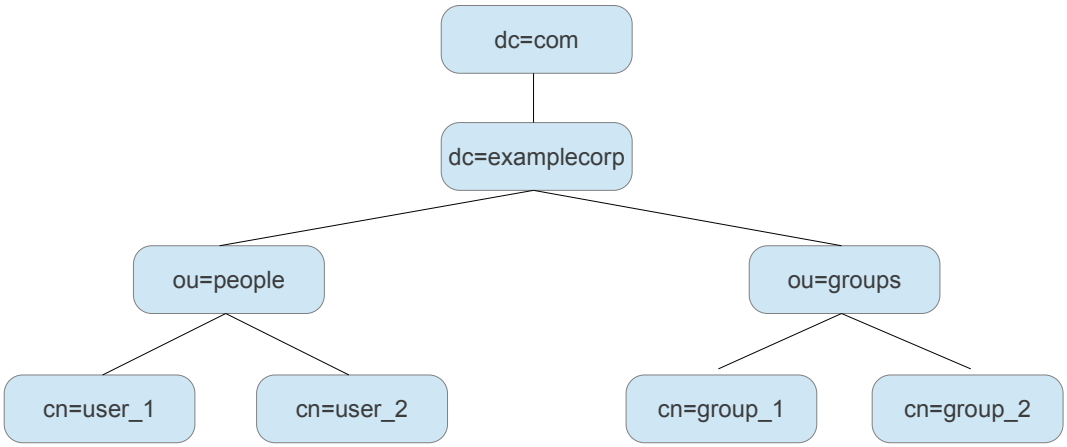

Abbildung 4.7: Verzeichnisinformationsbaum *(Directory Information Tree (DIT))*

4.5.3 Attribute

Verzeichniseinträge (LDAP-*Objekte*) bestehen aus einer Menge von Attributen. Die Attribute enthalten die eigentlichen Daten, die bezogen auf das betreffende LDAP-*Objekt* im

[27] `http://directory.apache.org/`

Verzeichnis gespeichert werden. Um auf die Eigenschaften von LDAP-Objekten zuzugreifen, muss man die Bezeichnungen der entsprechenden Attribute kennen. Ein Attribut besteht dabei aus einem Attribut-Typ und einem oder mehreren Attribut-Werten. Die wichtigen Attribut-Typen sind [Widmer 2008]:

OID *(Object Identifier)* ist eine globale, eindeutige Nummer zur Identifizierung von Attribut-Typen.

NAME Über den Namen des Attributs kann man auf den jeweiligen Attribut zugreifen.

DESCRIPTION Eine optionale Beschreibung des Attribut-Typs.

EQUALITY Erlaubt die Angabe einer Regel *(Matchingrule)*, welche eine Angabe darüber macht wann ein Suchfilter auf einen der Attribut-Werte passt. Mit *„caseIgnoreMatch“* ist es z. B. möglich die Klein- und Großschreibung bei der Abfrage des Attributs nicht zu beachten.

SINGLE-VALUE Gibt an, dass es nur ein einzelner Attribut-Wert pro Eintrag existieren darf. Bei keiner Angabe wird der Standardwert *MULTI-VALUE* übernommen, was bedeutet, dass das Attribut beliebig viele Werte enthalten darf.

Ein spezielles Attribut, welches in jedem LDAP-*Objekt* enthalten sein muss, ist das Attribut *objectClass*.

4.5.4 Objektklassen

Ein LDAP-*Objekt* wird im Baum-Verzeichnis durch einen Verzeichniseintrag dargestellt. Ein Verzeichniseintrag wird mit der Zuweisung zur Objektklassen erstellt, in dem man die Attribute mit den Attribut-Werten füllt. Ein Verzeichniseintrag entspricht somit einer „Instanz“ aus der objektorientierten Programmierung. Die Objektklassen, zur welchen der LDAP-*Objekt* gehört, legen letztendlich fest, welche Attribute in diesem LDAP-*Objekt* vorhanden sein müssen.
Eine Objektklasse kann aufgrund des Konzeptes aus der objektorientierten Programmierung als eine Subklasse einer anderen Objektklasse fungieren und erbt dann deren Attribute. Durch die Ableitung der Attribute müssen die bestehenden Objektdefinitionen somit nicht mehrfach neu erstellt werden. Die wichtigen Angaben bei einer Objektklasse sind:

OID *(Object Identifier)* ist eine globale, eindeutige Nummer zur Identifizierung der Objektklasse.

NAME Über den Namen des Objekts, kann man auf das Objekt zugreifen.

DESCRIPTION Eine optionale Beschreibung der Objektklasse.

MUST Angabe der Attribute, welche zwingend einen Wert enthalten müssen.

MAY Angabe der Attribute, welche einen Wert enthalten dürfen, aber nicht müssen.

4.5.5 LDAP-Filter

Mit einem LDAP-Filter können bestimmte Suchkriterien angegeben werden, nach welchen die LDAP-*Objekte* im LDAP-Verzeichnis durchsucht werden können. Mit LDAP-Filter können z. B. alle Benutzer in einer bestimmten *(organizationalUnit (ou))*, alle Gruppen einer Domäne, Benutzer anhand von bestimmten Kriterien wie E-Mail Adressen usw. durchsucht werden. Wird bei einer Suchabfrage kein Filter angegeben, wird ein Standardfilter *(objectClass=*)* verwendet, der auf jeden LDAP-*Objekt* im LDAP-Verzeichnis passt, da jeder LDAP-*Objekt* definitionsgemäß zu mindestens einer Objektklasse gehört.
Ein Filter besteht aus logischen Verknüpfungen von Ausdrücken. Ein Ausdruck besteht dabei aus einem Attributnamen und dem jeweiligen Suchmuster. Der Attributname und das Suchmuster werden durch einen Vergleichsoperator *(= Gleichheit, <= kleiner oder gleich, >= größer oder gleich, ∼= ähnlich)* miteinander verbunden und von einem Klammerpaar umschlossen. Das Suchmuster kann dabei aus einer Zeichenkette, einer Zahl oder einem booleschen Wert bestehen. Die Ausdrücke können durch die logischen Verknüpfungen (& „und“, | „oder“, ! „Negation“) ergänzt werden.

Beispiele

1. $(objectClass = inetOrgPerson)$
   Bei dieser LDAP-Suche werden diejenigen Einträge zurückgeliefert, welche die Objektklasse *inetOrgPerson* enthalten.
2. $(\&(objectClass = user)(cn = *Marketing*))$
   Diese LDAP-Suche liefert alle Einträge, welche als *objectClass=user* UND als *(Common Name (cn))* das Wort „Marketing“ enthalten.
3. $(\&(objectClass = group)(|(ou : dn := Stuttgart)(ou : dn := Berlin)))$
   Mit dieser LDAP-Suche werden diejenigen Gruppen aus dem LDAP-Verzeichnis zurückgeliefert, welche als *(organizationalUnit (ou))* ihrer *DN* - Stuttgart ODER Berlin enthalten.

4.5.6 LDAP Data Interchange Format (LDIF)

Die Daten, die in einem LDAP-Verzeichnis gespeichert sind, können mit einem speziellen, standardisierten Austauschformat *LDIF*[28] zwischen verschiedenen Verzeichnissen ausgetauscht werden. *LDIF* ist ein ASCII[29] basierendes Dateiformat und ist von Menschen einfach zu interpretieren. Es stellt die Daten aus einem Verzeichnis als Inhalt einer Reihe von Datensätzen dar, indem jeder Datensatz als ein Eintrag (LDAP-Objekt) dargestellt wird. Jeder Eintrag (LDAP-*Objekt*) im LDIF-Format ist durch eine Leerzeile von den anderen LDAP-Objekten getrennt und beginnt immer mit der eindeutigen DN, die sich in der ersten Zeile des jeweiligen LDAP-Objekts befindet. Das folgende Beispiel (List. 4.5) zeigt einen Eintrag

[28] `http://tools.ietf.org/html/rfc2849`
[29] `AmericanStandardCodeforInformationInterchange`

eines Demo-Benutzers (LDAP-*Objekt*) aus Activiti im LDIF-Format, nach dem er aus dem LDAP-Verzeichnis[30] exportiert wurde.

Listing 4.5 Demo-Benutzer aus Activiti im LDIF-Format

```
1 dn: uid=kermit,ou=users,ou=system
2 objectClass: organizationalPerson
3 objectClass: person
4 objectClass: inetOrgPerson
5 objectClass: top
6 cn: kermit theFrog
7 sn: kermit
8 uid: kermit
9 userPassword:: e1NTSEF9Q1RTDM3UjdDamVNNzBHdnV5UWFPQ2MxV2pBL2ZI
```

4.5.7 LDAP und Sicherheit

Die Sicherheit von Daten in einem Verzeichnis lässt sich über drei Teilsicherheitsaspekte beschreiben:

Zugang (Access) Für einen sicheren Zugang unterstützt LDAP ein Verschlüsselungsprotokoll *(Transport Layer Security (TLS))* zur sicheren Übertragung der Daten und sorgt in dem Fall für die verschlüsselte Kommunikation zwischen LDAP-Client und LDAP-Verzeichnis.

Authentifizierung Die aktuelle LDAP-Version[31] (v3) unterstützt das *Simple Authentication and Security Layer (SASL)*-Framework[32], mit dem man verschiedene Authentifizierungsmöglichkeiten wie z. B. *Challenge-Response Authentication Mechanism - Message Digest 5 (CRAM-MD5)* (Übertragung des Passworts nicht im Klartext), *ANONYMOUS* (Zugriffe sind ohne Authentifizierung gestattet), *One-Time-Password (OTP)* (Authentifizeirung mit dem einmaligen Passwort) verwenden kann.

Autorisierung Nach einer erfolgreichen Authentifizierung sind die Daten im LDAP-Verzeichnis mit Hilfe von ACL geschützt. Anhand von ACL entscheidet sich dann, welchen Zugriff ein Benutzer auf die Ressourcen des LDAP-Verzeichnisses bekommt.

4.5.8 Apache Directory Server und Apache Directory Studio

Als Verzeichnisdienst für die Anbindung an Activiti wurde auf eine freie, vollständig in Java realisierte und so auf praktisch jeder Hard- und Softwareplattform funktionierende

[30] http://directory.apache.org/apacheds/
[31] http://tools.ietf.org/html/rfc2251
[32] http://www.ietf.org/rfc/rfc4422.txt

LDAP-Lösung[33] *(Apache Directory Server (ApacheDS))* zurückgegriffen [Zörner 2009]. Das *Apache Directory Projekt*[34] stellt mehrere Installer bereit (für Windows, Linux, MacOS), mit welchen sich *ApacheDS* einfach installieren lässt. *ApacheDS* ist von der *OpenGroup*[35] zertifiziert, was die Konformität zur LDAP-Version (v3) garantiert.
Um auf die Struktur und die Einträge des Verzeichnisses *(ApacheDS)* zuzugreifen, wird im Folgenden *Apache Directory Studio*[36] verwendet. *Apache Directory Studio* ist ein Eclipse-Plugin, welches unter anderem das Lesen, Anlegen und Manipulieren von LDAP-Einträgen erlaubt. Es stehen dabei mehrere Tools zur Verfügung (Browser, Editor, Schema-Browser, LDIF-Editor), die als Plugin in der *Integrated Development Environment (IDE)* oder als *Rich Client Platform (RCP)*[37]-Anwendung betrieben werden können. *Apache Directory Studio* wurde speziell für *ApacheDS* entwickelt, kann aber auch zum Zugriff auf alle gängigen LDAP-Server verwendet werden.
Nach der Installation von *ApacheDS* und *Apache Directory Studio* kann zwischen den beiden Komponenten eine neue Verbindung angelegt werden. Dazu braucht man solche Angaben wie Hostname des *Apachde Directory Servers* und Port (Standard 10389).
Als Authentifizierungsmethode kann „anonym“ gewählt werden. Dazu sind die Identitätsangaben nicht nötig, d. h. die vorhandenen Informationen im LDAP-Server stehen ohne Authentifizierung zur Verfügung.
Der *ApacheDS* verwendet jedoch als Default-Einstellung einen Administrator (siehe Abb. 4.8) als Benutzer *(uid=admin,ou=system)* mit einem Passwort *(secret)*, mit welchem man sich authentifizieren kann (falls man sich nicht für die anonyme-Authentifizierungsmethode entscheidet) [Rademakers 2012] [Zörner 2009].
Nach der Authentifizierung sind die Einträge aus *ApacheDS* im LDAP-Browser des *Apache Directory Studio* ersichtlich. Jetzt können Benutzer und Gruppen mit Hilfe des LDAP-Browsers in *ApacheDS* hinzugefügt werden.
Um den jeweiligen Benutzer hinzuzufügen, wählt man mit der rechten Maustaste den Eintrag für die Benutzer *(ou=users)*, übernimmt aus der Liste der zur Verfügung stehenden Objektklassen die Klasse *„inetOrgPerson“*, die einen Benutzereintrag im LDAP-Server repräsentiert und vergibt anschließend den Benutzernamen. Die Abbildung 4.8 zeigt den bereits hinzugefügten Benutzer *„kermit“* mit der *dn: uid=kermit,ou=users,ou=system*. Der exportierte LDIF-Format von diesem Benutzer ist in Listing 4.5 zu sehen.
Die Erstellung der Gruppen in LDAP-Browser funktioniert ähnlich wie die Erstellung der neuen Benutzer. Dazu klickt man mit der rechten Maustaste den LDAP-Eintrag *„ou=groups“*, übernimmt aus der Liste der zur Verfügung stehenden Objektklassen die Klasse *„groupOfUniqueNames“*, was bedeutet, dass die Benutzer in dieser Gruppe nicht doppelt vorkommen können und vergibt anschließend den Namen für die Gruppe (siehe Abb. 4.8 *(sales)*). Um einen Benutzer zu dieser Gruppe *(sales)* hinzuzufügen, ist die Angabe des Benutzers mit der jeweiligen DN nötig (für Kermit *dn: uid=kermit,ou=users,ou=system*).

[33] `http://directory.apache.org/apacheds/`
[34] `http://directory.apache.org/apacheds/downloads.html`
[35] `http://www.opengroup.org/certification/directory-home.html`
[36] `http://directory.apache.org/studio/`
[37] `http://www.eclipse.org/home/categories/rcp.php`

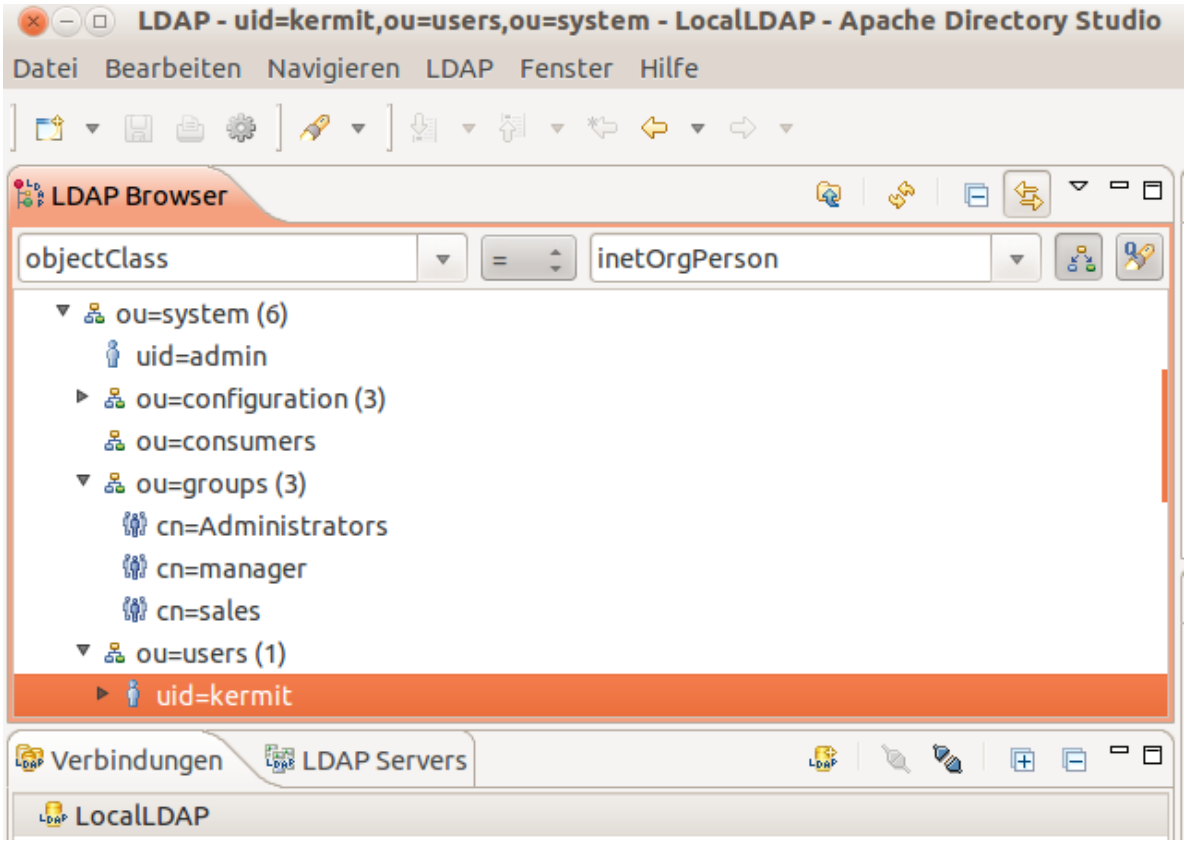

Abbildung 4.8: LDAP-Browser in *Apache Directory Studio*

4.5.9 Implementierung der Abfragelogik in Activiti

Die LDAP-Anbindung an Activiti hat grundsätzlich zwei Anwendungsfälle. Der erste Fall ist die Authentifizierung durch *IdentityService*. *IdentityService* wird von Activiti für die Benutzeranmeldung in Activiti Explorer verwendet.
Der zweite Anwendungsfall ist die Abfrage sowie die Auflösung von Benutzern und Gruppen durch *Activiti-Engine* während der Prozesslaufzeit.
In Activiti existieren zwei Java-Klassen *(UserManager und GroupManager)* die als Standard-Klassen die IdM-Lösung von Activiti implementieren und welche für die LDAP-Anbindung erweitert werden müssen.
Im Nachfolgenden wird schrittweise aufgezeigt, wie die Klasse *UserManager* von Activiti für die Authentifizierung und die Abfrage der Benutzer in ApacheDS erweitert werden kann.
Die *UserManager*-Klasse kommuniziert normalerweise mit der eigenen Activiti-Datenbank, in welcher die Benutzer und die Gruppen hinterlegt sind. Diese Klasse besitzt mehrere Methoden für die Erstellung, Entfernung und Aktualisierung von Benutzereinträgen in der eigenen Datenbank. Diese Methoden sind durch die zentralisierte Verwaltung der Benutzer mit LDAP-Server *(ApacheDS)* nicht mehr sinnvoll, da dies ab jetzt direkt durch einen LDAP-Administrator mit *Apache Directory Studio* erledigt wird. Aus diesem Grund muss in der Unterklasse *(Subclass)*, welche die *UserManager*-Klasse erweitert die Erstellung, Entfernung und Aktualisierung der neuen Benutzer in *Activiti-Explorer* durch eine Ausnahmebehandlung *(ActivitiException)* abgefangen werden (siehe List. 4.6).

Listing 4.6 Überschreibung der Standard-Methoden aus der *UserManager*-Klasse

```
1     @Override
2   public User createNewUser(String userId) {
3     throw new ActivitiException("Das Anlegen eines neuen Benutzerkontos ist
          nur durch einen LDAP-Administrator moeglich");
4   }
5     @Override
6   public void updateUser(User updatedUser) {
7     throw new ActivitiException("Die Aktualisierung der Benutzerdaten ist
        nur durch einen LDAP-Administrator moeglich");
8   }
9     @Override
10  public void deleteUser(String userId) {
11    throw new ActivitiException("Die Entfernung eines neuen Benutzerkontos
        ist nur durch einen LDAP-Administrator moeglich");
12  }
```

Als nächstes muss die Authentifizierungsmethode *(checkPassword)* angepasst werden, da die Authentifizierung jetzt nicht über die interne Datenbank, sondern über *ApacheDS* ablaufen soll. In dieser Methode, welche bei der Authentifizierung der Benutzer in *Activiti Explorer* aufgerufen wird, müssen die LDAP-Verbindungsparameter (*Server* und *Port* von *ApacheDS*) aufgerufen, sowie die Authentifizierungsdaten (*uid* und *password*) überprüft werden (siehe List. 4.9).

Listing 4.7 Anpassung der *checkPassword*-Methode aus der *UserManager*-Klasse

```
1     @Override
2   public Boolean checkPassword(String userId, String password) {
3         boolean credentialsValid = false;
4         LdapConnection connection = new LdapConnection(connectionParams.
              getLdapServer(),
5                   connectionParams.getLdapPort());
6       try {
7         BindResponse response = connection.bind("uid=" + userId + "," + "ou
              =users,ou=system", password);
8         if(response.getLdapResult().getResultCode() == ResultCodeEnum.
              SUCCESS) {
9             credentialsValid = true;
10        }
11    } catch (Exception E) {
12        throw new ActivitiException("Fehler bei der LDAP-Verbindung", E);
13    }
14        LDAPConnectionUtil.closeConnection(connection);
15
16      return credentialsValid;
17  }
```

Als nächster Schritt muss eine Java-Methode implementiert werden, die die Abfrage und Auflösung von Benutzern während der Prozesslaufzeit ermöglicht.
In dieser Methode muss die LDAP-Abfrage den Eingabeparametern der *UserQueryImpl*-Methode aus Activiti entsprechen. Man kann dann z. B. die Benutzer nach ihrer *uid (user identifier)* oder *sn (surname)* im LDAP-Verzeichnis durchsuchen (siehe List. 4.8). Die Abfrageparameter können aber abhängig von den vorhandenen Benutzer-Attributen im LDAP-Verzeichnis beliebig erweitert werden [Rademakers 2012].
Die jeweilige LDAP-Benutzerabfrage wird dann nach der Authentifizierung am LDAP-Server ausgeführt. Bei der Ausführung der Abfrage muss die Baumstruktur des LDAP-Verzeichnisses beachtet werden. Die Suche nach dem Benutzer „kermit“ aus der Abb. 4.8 muss z. B. im Abfrage-String den Pfad *ou=users, ou=system* enthalten.

Listing 4.8 Die LDAP-Suche über *uid (user identifier)* oder *sn (surname)*

```
1 if(StringUtils.isNotEmpty(userQuery.getId())) {
2             searchQuery.append("(uid=").append(userQuery.getId()).append(")
                  ");
3
4         } else if(StringUtils.isNotEmpty(userQuery.getLastName())) {
5             searchQuery.append("(sn=").append(userQuery.getLastName()).
                  append(")");
```

Als letzter Schritt müssen die LDAP-Konfigurationsparameter der *Activiti-Engine* bekannt gemacht werden. Dazu muss in der internen Datei von Activiti *„activiti-standalone-context.xml“* die eigentlichen Verbindungs- und Zugangsdaten *(ldapServer, ldapPort, ldapUser und ldapPassword)*, sowie der Pfad zur angepassten *UserManager*-Klasse eingetragen werden. Diese Verbindungs- und Zugangsdaten, werden während einer LDAP-Abfrage von der *„checkPassword“*-Methode (siehe List. 4.9) aufgerufen. Da die Abfrage der Zugangsdaten nicht anonym abläuft, findet die Authentifizierung bei der jeweiligen Benutzerabfrage in dem Fall mit den Zugangsdaten des Benutzers *uid=admin,ou=system* mit dem Passwort *secret* statt.

Listing 4.9 Zugangs- und Verbindungsparameter zum LDAP-Server

```
1 <bean id="ldapConnectionParams"
2 <property name="ldapServer" value="localhost" />
3 <property name="ldapPort" value="10389" />
4 <property name="ldapUser"
5 value="uid=admin,ou=system" />
6 <property name="ldapPassword" value="secret" />
7 </bean>
```

Die automatische Erstellung von Demo-Benutzern in der eigenen Datenbank, welche beim Neustart von Activiti erstellt werden, muss durch die Entfernung von *„demoDataGenerator“*-bean in *„activiti-standalone-context.xml“* deaktiviert werden, da die Benutzer ab jetzt nur durch *ApacheDS* verwaltet werden.

Kapitel 5

Zusammenfassung und Ausblick

Im Rahmen dieser Diplomarbeit wurden die vorhandenen Benutzerverwaltungs- und Sicherheitskonzepte im Geschäftsprozessmanagement untersucht.

Hierbei wurde ausführlich auf solche Sicherheitsanforderungen aus dem Geschäftsprozessmanagement wie die Autorisierung, Nutzungskontrolle, Funktionstrennung *(Separation of Duty)* und Aufgabenbindung *(Binding of Duty)* eingegangen.

Eine besondere Rolle spielt dabei das rollenbasierte Berechtigungskonzept, welches eine effektive und effiziente Verwaltung von Berechtigungen im IT- und Geschäftsbereich der Unternehmen ermöglicht. Es wurden dazu die Modellspezifikationen von vier rollenbasierten Zugriffsmodellen aus dem RBAC-Standard vorgestellt.

Um die Ausdrucksstärke des RBAC-Standards zu erhöhen, wurden im Laufe der Zeit mehrere Konzepte zur kontextbezogenen Zugriffskontrolle von Rollen wie z. B. die attributbasierte Zugriffskontrolle (ABAC) und die temporale Ergänzung des rollenbasierten Zugriffsmodells (GTRBAC) in das RBAC-Standard integriert und im Rahmen der Arbeit erläutert.

Aufgrund der unterschiedlichen Sicherheitslösungen der einzelnen Applikationen und den damit verbundenen Aufwand bei der Administration von Benutzern und Systemen in Unternehmen, wurde eine globale Lösung zur unternehmensweiten Zugriffskontrolle beschrieben. Diese Lösung wird mit Hilfe von Geschäftsrollen *(Enterprise Roles)* im ERBAC-Modell bewerkstelligt. In der Praxis verwendet man dazu *Identity Management*-Werkzeuge, die zentral und übergreifend die Administration der Geschäftsrollen regeln.

Im Laufe der Arbeit, war es wichtig festzustellen, wie die Sicherheitsanforderungen technisch umgesetzt werden können. Hierfür wurde eine Literaturrecherche durchgeführt und mehrere Forschungsarbeiten vorgestellt, die sich mit der Modellierung und Umsetzung der Sicherheitsanforderungen im Geschäftsprozessmanagement beschäftigen. Infolge der Analyse wurde ersichtlich, dass die Autoren sich hauptsächlich auf die Modellierung der Sicherheitsanforderungen konzentrieren und für die Umsetzung auf den XACML-Standard verweisen.

Im letzten Teil der Arbeit wurde eine Workflow- und BPM-Plattform namens Activiti vorgestellt, die sich auf Modellierung und Ausführung von Geschäftsprozessen konzentriert. Dabei fand eine Untersuchung der Sicherheitsfunktionen der Plattform und die gleichzeitige Einordnung von diesen, in die von *Common Criteria* definierten Funktionsklassen statt. Im

Anschluss daran wurde das Sicherheitskonzept von Activiti mit der Umsetzung der Funktionstrennung, Aufgabenbindung und der Einbindung der Plattform an ein Verzeichnisdienst (LDAP) erweitert.

Ausblick

Im Laufe der Arbeit wurde unter anderem die technische Umsetzung der Sicherheitsanforderungen im Geschäftsprozessmanagement untersucht. In der wissenschaftlichen Literatur wird dabei hauptsächlich die Umsetzung mit den *Security Policies* aus dem XACML-Standard beschrieben. Die Einhaltung der *Security Policies* während der Prozesslaufzeit übernehmen dann die XACML-Komponenten.

Da die Umsetzung meistens recht allgemein beschrieben wird, ist es empfehlenswert den genauen Aufwand beim Aufbau der von XACML benötigten Komponenten unter bestimmten Voraussetzungen zu untersuchen. Zusätzlich wäre es sinnvoll solche Aspekte wie die Performance, Skalierbarkeit und die Stabilität dieser Komponente während eines produktiven Betriebs zu ermitteln.

Literatur

Accorsi, Rafael (2013). *Sicherheit im Prozessmanagement, 1–5* (siehe S. 13, 41, 42).

Allweyer, Thomas (2009). *BPMN 2.0 - Business Process Model and Notation: Einführung in den Standard für die Geschäftsprozessmodellierung*. Books on Demand; Auflage: 2. Auflage. ISBN: 978-3839121344 (siehe S. 39, 43).

Asprion, Petra Maria (2013). *Funktionstrennung in ERP-Systemen*. Springer Fachmedien Wiesbaden. ISBN: 978-3-658-00037-0 (siehe S. 35).

Bertino, E., E. Ferrari und V. Atluri (1999). „The specification and enforcement of authorization constraints in workflow management systems“. In: *ACM Trans. Inf. Syst.Secur., 2*, 65–104 (siehe S. 13).

Bertino, E. und R. Sandhu (2005). „Database Security—Concepts, Approaches, and Challenges“. In: *IEEE Transactions on Dependable and Secure Computing* 2, S. 2–19 (siehe S. 37).

Bhatti, Rafae A. (2003). „X-GTRBAC: An XML-Based Policy Specification Framework and Architecture for Enterprise-Wide Access Control“. Magisterarb. Purdue University (siehe S. 34).

Botha, R. und J. Eloff (2001). „Separation of duties for access control enforcement in workflow environments“. In: *IBM Systems Journal, 40(3)*, S. 666–682. ISSN: 0018-8670. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.83.6573&rep=rep1&type=pdf (siehe S. 13, 28).

Criteria, Common (2012). *Common Criteria for Information Technology Security Evaluation*. Englisch. abgerufen am 31.07.13. URL: http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R4.pdf (siehe S. 11, 58–60).

Eckert, Claudia (2009). *IT-Sicherheit, Konzepte-Verfahren-Protokolle*. Oldenbourg. ISBN: 978348658999-3 (siehe S. 11, 14–17, 22, 23).

Eiff, Wilfried von und Ralf Ziegenbein (2003). *Geschäftsprozeßmanagement : Methoden und Techniken für das Management von Leistungsprozessen im Krankenhaus / Hrsg. von Wilfried von Eiff, Ralf Ziegenbein*. Gütersloh : Verlag Bertelsmann Stiftung (siehe S. 38).

Ferraiolo, David F. (2001). „Proposed NIST standard for role-based access control“. In: *ACM Transactions on Information and System Security (TISSEC)* 4 Issue 3. URL: http://dl.acm.org/citation.cfm?doid=501978.501980 (siehe S. 9).

Ferraiolo, David F. und D. Richard Kuhn (1992). „Role-Based Access Controls“. In: *15th National Computer Security Conference*, S. 554 –563. URL: http://csrc.nist.gov/rbac/ferraiolo-kuhn-92.pdf (siehe S. 9, 17, 21).

Fischer-Hübner, Simone (2001). *IT-Security and Privacy*. Springer, Berlin, S. 358. ISBN: 978-3-540-42142-9 (siehe S. 16).

Gropp, Alexander (2007). *Rollenbasierte Zugriffskontrolle (RBAC)*. deutsch. abgerufen am 01.08.2013. Institut für Programmstrukturen und Datenorganisation (IPD). URL: http://dbis.ipd.uni-karlsruhe.de/download/SS07GroppRbac-Ausarbeitung.pdf (siehe S. 22).

Information Technology, American National Standard for (2004). „Role Based Access Control". In: *Information Technology Industry Council (ITI)*. URL: http://profsandhu.com/journals/tissec/ANSI+INCITS+359-2004.pdf (siehe S. 22, 24–26, 28).

Joshi, James B.D. und Elisa Bertino (2005). „A Generalized Temporal Role-Based Access Control Model". In: *Ieee Transactions on Knowledge and Data Engineering* 17.1, S. 23. URL: http://www.sis.pitt.edu/~jjoshi/GTRBAC_IEEETKDE.pdf (siehe S. 32).

Joshi, James B.D., Elisa Bertino und Basit Shafiq (Apr. 2003). „Dependencies and Separation of Duty Constraints in GTRBAC". In: *Center for Education and Research inInformation Assurance and Security,Purdue University, West Lafayette, IN 4790* (siehe S. 33, 34).

Kern, Axel (2002). „Advanced Features for Enterprise-Wide Role-Based Access Control". In: S. 10 (siehe S. 34–36).

Klarl, H. (2011). *Zugriffskontrolle in Geschäftsprozessen - Ein modellgetriebener Ansatz.* Vieweg+Teubner Verlag Springer Fachmedien Wiesbaden GmbH. ISBN: 978-3-8348-1465-4 (siehe S. 9, 31, 36, 38, 41, 43, 63).

Koch, Susanne (2011). *Einführung in das Management von Geschäftsprozessen*. Springer. ISBN: 978-3-642-01121-4 (siehe S. 38).

Kocian, Claudia (2011). „Geschäftsprozessmodellierung mit BPMN 2.0, Business Process Model and Notation im Methodenvergleich". In: (Siehe S. 39).

Laudon, Kenneth (2010). *Wirtschaftsinformatik, Eine Einführung*. Pearson Studium. ISBN: 9783827373489 (siehe S. 38).

Lehnert, Volker, Katharina Stelzner und Peter John (2013). *SAP-Berechtigungswesen*. SAP PRESS, S. 773. ISBN: 978-3-8362-1825-2 (siehe S. 22).

Molitorisz, Korbinian (2008). „Rollenmodelle für die Zugriffskontrolle in Unternehmen". Magisterarb. URL: http://www.ipd.kit.edu/Tichy/uploads/publikationen/223/Molitorisz-RollenmodellefrdieZugriffskontrolleinUnternehmen.pdf (siehe S. 9, 24, 34).

Mülle, Jutta, Silvia von Stackelberg und Klemens Böhm (2011). „A Security Language for BPMN Process Models". In: *Karlsruhe Institute of Technology (KIT)*, S. 23. ISSN: 2190-4782. URL: http://digbib.ubka.uni-karlsruhe.de/volltexte/documents/1978092 (siehe S. 43, 45, 46, 62).

O'Connor, Alan C. und Ross J. Loomis (2010). *Economic Analysis of Role-Based Access Control*. Techn. Ber. RTI International. URL: http://csrc.nist.gov/groups/SNS/rbac/documents/20101219_RBAC2_Final_Report.pdf (siehe S. 19, 30, 37, 38).

Plötner, Johannes und Steffen Wendzel (2012). *Linux*. Galileo Computing. ISBN: 978-3-8362-1822-1 (siehe S. 67).

Priebe, Torsten und Wolfgang Dobmeier (2005). *ABAC – Ein Referenzmodell für attributbasierte Zugriffskontrolle*. deutsch. abgerufen am 01.08.2013. Lehrstuhl für Wirtschaftsinformatik I, Universität Regensburg, D-93040 Regensburg. URL: http://subs.emis.de/LNI/Proceedings/Proceedings62/GI-Proceedings.62-30.pdf (siehe S. 15, 16, 31, 32).

Rademakers, Tijs (2012). *Activiti in Action*. Manning Publications Co. ISBN: 9781617290121 (siehe S. 51, 72, 75).

Rodríguez, Alfonso, Eduardo Fernández-Medina und Mario Piattini (2007). „A BPMN Extension for the Modeling of Security Requirements in Business Processes". In: *IEICE TRANS. INF. & SYST.* E90–D, S. 745–752 (siehe S. 43, 47).

— (2010). „Semi-formal transformation of secure business processes into analysis class and use case models: An MDA approach". In: *Information and Software Technology archive Volume 52 Issue 9*, S. 945–971 (siehe S. 47).

Rupprecht, Josef und Felix Wortmann (2006). *Zugriffskontrolle in heterogenen Applikationslandschaften*. Springer, Berlin, S. 123–168. ISBN: 978-3-540-20506-7. URL: http://link.springer.com/chapter/10.1007%2F3-540-29480-5_5 (siehe S. 16).

Rücker, Bernd, Jakob Freund und Thomas Henninger (2010). *Praxishandbuch BPMN*. Carl Hanser Verlag GmbH & Co. KG. ISBN: 978-3446417687 (siehe S. 39).

Sandhu, Ravi (1996). „Roles Versus Groups". In: *George Mason University and SETA Corporation*. URL: http://www.profsandhu.com/workshop/role-group.pdf (siehe S. 20).

Schmaltz, Robert (2005). *IT-Unterstützung für das Wissensmanagement in Kooperationen*. Universitätsverlag Göttingen. ISBN: 3-938616-22-9 (siehe S. 16, 17).

Siegle, Klaus-Peter (1994). „Geschäftsprozesse und Kernkompetenzen". In: *Prozessmanagement: Konzepte, Umsetzungen und Erfahrungen des Reengineering*, S. 164–180 (siehe S. 38).

Tsolkas, A. und K. Schmidt (2010). *Rollen und Berechtigungskonzepte*. Vieweg Teubner Verlag Springer Fachmedien Wiesbaden GmbH. ISBN: 978-3-8348-9745-9 (siehe S. 22, 23, 25, 28).

Valek, Joel (2010). „Rollenbasierte Zugriffskontrollsysteme und deren praktische Umsetzbarkeit auf Betriebssystemebene unter Microsoft Windows 7 und Windows Server 2008". Magisterarb. Universität Linz, Technisch-Naturwissenschaftliche Fakultät (siehe S. 20, 21, 24).

Widmer, Markus (2008). „Role Based Access Control mit OpenLDAP". Magisterarb. Wilhelm-Schickard-Institut fur Informatik Universität Tübingen (siehe S. 67, 69).

Wiesner, Mike (2008). *Rollen sind keine Gruppen*. deutsch. abgerufen am 01.08.2013. URL: http://mwiesner.com/2008/03/28/rollen-sind-keine-gruppen/ (siehe S. 17, 20).

Winter, Robert und Elmar Sinz (2007). „Enterprise Architecture“.
In: *Information Systems and e-Business Management* Volume 5, S. 357–358 (siehe S. 41).
Wolter, Christian und Christoph Meinel (2010).
„An approach to capture authorisation requirements in business processes“.
In: *Requirements Engineering* 15, S. 359–373 (siehe S. 43–45).
Wolter, Christian, Andreas Schaad und Christoph Meinel (2007).
„Deriving XACML Policies from Business Process Models“.
In: *Web Information Systems Engineering – WISE 2007 Workshops*, S. 142–153 (siehe S. 45).
Wortmann, Felix und Robert Winter (2007). „Vorgehensmodelle fuer die rollenbasierte Autorisierung in heterogenen Systemlandschaften“.
In: *Wirtschaftsinformatik Band 49, Nummer 6 Seiten 439-447*. ISSN: 0937-6429.
URL: `http://link.springer.com/article/10.1007%2Fs11576-007-0096-4`
(siehe S. 9, 10, 17).
Zörner, Stefan (2009).
LDAP für Java Entwickler - Einstieg und Integration, 3. aktualisierte Auflage. entwickler.press.
ISBN: 978-3-86802-216-2 (siehe S. 72).

Akronyme und Abkürzungen

ABAC Attribute-Based Access Controll 4

ACL Access Control List 18

AD Microsoft Active Directory 67

BPEL Business Process Execution Language 39

BPM Business Process Management 3

BPMI Business Process Management Initiative 39

BPMN Business Process Model and Notation 4

BPMS Business Process Management System 7

CC Common Criteria for Information Technology Security Evaluation 5

CRAM-MD5 Challenge-Response Authentication Mechanism - Message Digest 5 71

DIT Directory Information Tree 5

DNS Domain Name System 67

DN Distinguished Name 68

DSOD Dynamic Separation of Duty 29

EPK Ereignisgesteuerte Prozesskette 39

ERBAC Enterprise Role-Based Access Control 4

ERP Enterprise Resource Planning 22

GTRBAC Generalized Temporal RBAC 4

IDE Integrated Development Environment 72

IdM Identity Management 36

LDIF LDAP Data Interchange Format 5

OMG Object Management Group 39

OTP One-Time-Password 71

PDP Policy Decision Point 45

PEP Policy Enforcement Point 45

PIP Policy Information Point 45

RACF Resource Access Control Facility 35

RBAC Role Based Access Control 3

RDN Relative Distinguished Name 68

RCP Rich Client Platform 72

SOA Service-Oriented Architecture 42

SSOD Static Separation of Duty 29

SASL Simple Authentication and Security Layer 71

TLS Transport Layer Security 71

UML Unified Modeling Language 39

XACML eXtensible Access Control Markup Language 31

Alle URLs wurden zuletzt am 2013-09-15 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift