UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Klassifikation und Analyse von IT-Sicherheitsmodellen

Masterarbeit

zur Erlangung des Grades eines Master of Science
im Studiengang Informatik

vorgelegt von

Vitali Keppel

Erstgutachter:	Prof. Dr. Rüdiger Grimm Institut für Wirtschafts- und Verwaltungsinformatik
Zweitgutachter:	Dr. Katharina Bräunlich Institut für Wirtschafts- und Verwaltungsinformatik

Koblenz, im Juni 2013

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

. . . . . .

(Ort, Datum) (Unterschrift)

Zusammenfassung

Im Rahmen dieser Masterarbeit wird ein umfassender Überblick über die Vielfalt der Sicherheitsmodelle gegeben, indem ausgewählte Sicherheitsmodelle beschrieben, klassifiziert und miteinander verglichen werden.

Sicherheitsmodelle beschreiben in einer abstrakten Weise die sicherheitsrelevanten Komponenten und Zusammenhänge eines Systems. Mit den Sicherheitsmodellen können komplexe Sachverhalte veranschaulicht und analysiert werden.

Da Sicherheitsmodelle unterschiedliche Sicherheitsaspekte behandeln, beschäftigt sich diese Arbeit mit der Ausarbeitung eines Klassifizierungsschemas, welches die strukturelle und konzeptuelle Besonderheiten der Modelle in Bezug auf die zugrundeliegenden Sicherheitsaspekte beschreibt. Im Rahmen des Klassifizierungsschemas werden die drei grundlegenden Modellklassen gebildet: Zugriffskontrollmodelle, Informationsflussmodelle und Transaktionsmodelle.

Sicherheitsmodelle werden in einem direkten und indirekten Vergleich gegenüber gestellt. Im letzten Fall werden sie einer oder mehrerer Modellklassen des Klassifizierungsschemas zugeordnet. Diese Klassifizierung erlaubt, Aussagen über die betrachteten Sicherheitsaspekte und die strukturellen bzw. konzeptuellen Besonderheiten eines Sicherheitsmodells in Bezug auf die anderen Sicherheitsmodelle zu machen.

Beim direkten Vergleich werden anhand der ausgewählten Kriterien die Eigenschaften und Aspekte der Sicherheitsmodelle orthogonal zu den Modellklassen betrachtet.

Abstract

With the scope of this master thesis, a comprehensive overview of the variety of security models is given by the description, classification and comparison of selected security models.

Security models describe the security-relevant components and relationships of a system in an abstract way. Thus complex security issues can be illustrated and analyzed.

Since security models approach different aspects of security, this master thesis deals with the development of a classification scheme, which describes the structural and conceptual features of the models with respect to the underlying security issues. As part of the classification scheme, the three basic model classes are defined: access control models, information flow models and transaction models.

Security models are compared in a direct and indirect way. In the latter case, they are assigned to one or more model classes of the classification scheme. This classification allows to make statements about the considered security aspects and about the structural or conceptual features of a security model concerning other security models.

In case of direct comparison, the features and aspects of security models are considered orthogonal to the model classes based on the selected criteria.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich während des Studiums und bei der Erstellung dieser Masterarbeit stets unterstützt und geholfen haben.

Als Erstes möchte ich meinen Betreuern Prof. Dr. Rüdiger Grimm, Dr. Katharina Bräunlich und Daniel Pähler danken. Sie ermöglichten mir, meine Masterarbeit auf dem umfangreichen und spannenden Gebiet der Sicherheitsmodelle zu schreiben und unterstützten mich dabei mit Rat und Tat.

Des Weiteren möchte ich allen meinen Verwandten und Freunden danken, die mich durch die Studienjahre begleitet und die Verwirklichung dieser Masterarbeit ermöglicht haben. Insbesondere möchte ich mich bei Daniel Janke für die zahlreichen Diskussionen auf dem Bereich der theoretischen Informatik und insbesondere der Prozessalgebren bedanken. Einen besonderen Dank möchte ich ihm für sein Korrekturlesen der Arbeit und seine Verbesserungsvorschläge aussprechen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Der Wunsch nach sicheren und korrekten Systemen existiert bereits, seitdem die ersten Computersystemen entwickelt wurden. Im Laufe der Jahre sind stets größere und funktionsfähigere Anwendungssysteme entstanden, die jedoch immer komplexer wurden. Mit der Komplexität stieg auch die Anzahl der potentiellen Fehlerquellen, sodass der Ruf nach Methoden und Konzepten zur systematischen Analyse und Entwicklung immer größer wurde. Infolgedessen wurden erste formale Beschreibungs- und Analysemethoden entwickelt, mit denen die Probleme der wachsenden Komplexität bewältigt werden sollten. Aufgrund der starren und aufwändigen Verwendbarkeit und mangels der entsprechenden Werkzeuge haben sich diese jedoch in der Praxis nicht durchgesetzt [CW96].

Die modernen IT-Systemen haben einen sehr hohen Komplexitätsgrad erreicht, was den Einsatz von standardisierten Spezifikations- und Analysemethoden im Rahmen der Hard- und Softwareentwicklung praktisch unabdingbar macht. Das hatte zur Folge, dass grundlegende Konzepte wie Modell-getriebene Systementwicklung, standardisierte Spezifikationssprachen wie zum Beispiel *Z* und internationale Standards wie zum Beispiel *Common Criteria* entstanden sind, mit deren Hilfe die Zuverlässigkeit, Vertrauenswürdigkeit, Korrektheit und Sicherheit der IT-Systeme verbessert bzw. durchgesetzt werden sollen.

Insbesondere die sicherheitskritischen Systeme verlangen nach geeigneten Analyse- und Evaluierungsmethoden, da im Falle einer Fehlfunktion oder eines unerwünschten Verhaltens hohe Personen- und Sachschaden zu erwarten sind. Um die Sicherheit solcher Systeme zu überprüfen, werden im Rahmen der Modell-getriebenen Systementwicklung die Sicherheitsmodelle eingesetzt.

Anhand eines Sicherheitsmodells sollen die sicherheitskritischen Aspekte eines Systems in einer abstrakten Weise veranschaulicht und untersucht werden. Dabei soll zum einen Klarheit über die gegebenen Sachverhalte gegeben werden. Zum anderen soll mit einem Sicherheitsmodell überprüft werden können, ob ein System die gewünschten sicherheitsspezifischen Eigenschaften besitzt. Falls ein Sicherheitsmodell ein System präzise genug abgebildet hat und die entsprechenden Eigenschaften nachgewiesen werden konnten, ist dieses System sicher.

Die Sicherheit ist dabei kein absoluter Begriff. Zum Beispiel sollen in einem Krankenhausszenario die vertraulichen Patientendaten geschützt werden, indem der Zugriff auf diese Daten für Besucher des Krankenhauses eingeschränkt wird. Damit kann aber nicht verhindert werden, dass beispielsweise die Krankenschwestern die Daten an die Besucher weiter reichen. Somit muss ein Sicherheitsmodell möglichst präzise definiert werden, sodass für die Beschreibung komplexer Systeme in der Regel formale oder semiformale Sicherheitsmodelle verwendet werden[1] [Sch99].

Die ersten formalen Sicherheitsmodelle wie zum Beispiel das Harrison-Ruzzo-Ullman-Modell (Abschnitt 4.1) wurden entwickelt, um die gemeinsame Nutzung der Systemressourcen zu regeln. Diese Sicherheitsmodelle sollten dabei auf der Betriebssystemebene ein Referenzmonitor-Konzept simulieren. Der Referenzmonitor ist eine Instanz, die Zugriffe auf die Systemressourcen mittels einer Wissensdatenbank regelt. Die Wissensdatenbank enthält dabei Informationen über die erlaubten und unerlaubten Zugriffe [HRU76]. Möchte ein Nutzer des Systems auf eine Ressource zugreifen, stellt er eine Anfrage an den Referenzmonitor, der diesen Zugriff erlaubt oder ablehnt. Der Referenzmonitor wird dabei mit einem Zustandsübergangssystem und einer Zugriffsmatrix simuliert, die die Wissensdatenbank darstellen soll.

Des Weiteren haben militärische Organisationen den Nutzen von Sicherheitsmodellen entdeckt. Das Ziel war dabei, die vertraulichen Informationen vor potentiellen Feinden oder Spionen zu schützen. Daraus entstand das Bell-LaPadula Modell (Abschnitt 4.1), welches sowohl die Betriebssystemebene als auch die Anwendungsebene abbildet. Auf der Anwendungsebene werden dabei die strikten militärischen Strukturen simuliert.

Im Laufe der Zeit wurden Sicherheitsmodelle entwickelt, mit denen auch moderne Systeme modelliert werden können. Mit solchen Modellen kann zum Bei-

[1] Dieser und weitere Aspekte der Sicherheitsmodelle werden im Kapitel 2 erläutert.

spiel die Sicherheit in offenen Systemen wie dem Internet untersucht werden.

Heutzutage existieren unterschiedlichste Sicherheitsmodelle, die verschiedenen Aspekte der Sicherheit betrachten und dabei spezifische Konzepte bzw. Strukturen verwenden. Aus dieser Tatsache resultieren die Ziele dieser Arbeit.

1.1 Ziele der Arbeit

Das Ziel dieser Arbeit ist, einen umfassenden Überblick über die Sicherheitsmodelle zu geben. Dabei sollen die ausgewählten Modelle analysiert und verglichen werden. Um einen besser Vergleich anzufertigen, soll ein Schema erstellt werden, mit dem unterschiedliche Sicherheitsmodelle klassifiziert werden können. Im Rahmen des Schemas sollen also Modellklassen gebildet werden, die die strukturellen und konzeptuellen Besonderheiten der Sicherheitsmodelle repräsentieren. Das Schema soll dabei abstrakt genug ausgelegt sein, um nicht nur die bereits existierenden sondern auch zukünftige Sicherheitsmodelle zu klassifizieren. Die wichtigsten Vertreter der einzelnen Modellklassen sollen dann beschrieben und verglichen werden.

1.2 Inhalte der Arbeit

Zunächst werden in dem Kapitel 2 die grundlegenden Aspekte betrachtet, welche in den weiterführenden Kapiteln Anwendung finden. Im Kapitel 3 wird die Strategie ausgearbeitet, mit der die unterschiedlichen Sicherheitsmodelle verglichen werden können. Dabei wird ein Klassifizierungsschema definiert, das die konzeptuellen und strukturellen Aspekte der Sicherheitsmodelle beschreibt. Im Kapitel 4 werden anhand der Beschreibungselemente des Klassifizierungsschemas die ausgewählten Sicherheitsmodelle beschrieben. Der direkte Vergleich der Modelle wird in dem Kapitel 5 behandelt. Des Weiteren werden in dem Kapitel 6 die HRU- und Gleichgewichtsmodelle exemplarisch mit der Prozessalgebra *CSP* beschrieben. Schließlich wird im Kapitel 7 eine kurze Zusammenfassung und der Ausblick auf weiterführende Arbeiten gegeben.

Kapitel 2

Grundlagen der Sicherheitsmodelle

In diesem Kapitel wird ein grober Überblick über die *Sicherheitsmodelle* gegeben, der zum einen eine allgemeine Vorstellung über *Sicherheitsmodelle* geben und zum anderen als Basis für die weiterführenden Kapiteln dieser Arbeit dienen soll. Dabei werden ausgewählte Eigenschaften und ein konzeptueller Entstehungsprozess der *Sicherheitsmodelle* beschrieben.

2.1 Allgemeine Definition

Ein Modell ist im Allgemeinen eine vereinfachte Darstellung eines komplexen Sachverhalts, welche durch Vernachlässigung der als unwesentlich erachteten Aspekte und Hervorhebung der wesentlichen Aspekte gebildet wird. Inwieweit der vorliegende Sachverhalt abstrahiert wird und welche Aspekte dabei als relevant angesehen werden, bestimmt in der Regel die jeweilige Problemstellung, welche die Modellentwicklung initiiert [Mod12, Eck09, Gri08, KM93].

Ein *Sicherheitsmodell* im Sinne eines Modells ist eine abstrakte Abbildung der sicherheitsrelevanten Teile von sicherheitskritischen Anwendungen oder Produkten. Für relevant werden dabei alle Aspekte gehalten, die die Sicherheit dieser Systeme sowohl direkt als auch indirekt beeinflussen können.

Mithilfe von *Sicherheitsmodellen* soll also ein Rahmen für die Analyse und Entwicklung von sicherheitskritischen Systemen geschaffen werden. Das Ziel ist da-

bei, durch die Vereinfachung und Konkretisierung sicherheitsrelevanter Systemteile aussagekräftige Erklärungen für die darin auftretender Abläufe, Phänomene und Eigenschaften zu geben, um mögliche Sicherheitslücken, Fehler, Inkonsistenzen oder unerwünschte Aspekte aufzudecken oder vorherzusagen [Gri08, Eck09, Sec12].

Im Allgemeinen soll mit dem Einsatz von *Sicherheitsmodellen* das Vertrauen sowohl der Betreiber als auch der Anwender in sicherheitskritische Systeme gestärkt werden [MSUV02].

2.2 Modelleigenschaften

Die *Sicherheitsmodelle* wie Modelle im Allgemeinen können spezifische Merkmale besitzen, die in der Regel bei der Erstellung eines solchen Modells angestrebt werden. In diesem Abschnitt werden einige davon definiert, welche für diese Arbeit relevant sind und beim Vergleich der *Sicherheitsmodelle* in Kapitel 5 als Vergleichskriterien eingesetzt werden.

2.2.1 Abstraktionsgrad

Mit dem Abstraktionsgrad wird hier die Sicht eines *Sicherheitsmodell* auf ein zu modellierendes System verstanden. Ein *Sicherheitsmodell* mit einem systemnahen Abstraktionsgrad stellt die Implementierung eines Systems dar. Die Modellierung wird dabei in der Regel stark durch konkrete Strukturen, Abläufe und Eigenschaften eines Systems geprägt.

Ist ein *Sicherheitsmodell* in der Lage, Wissen über das Anwendungsgebiet des Systems unabhängig von seiner Implementierung zu spezifizieren, so besitzt es einen anwendungsnahen Abstraktionsgrad. Wenn ein Sicherheitsmodell zum Beispiel den Ablauf des Erwerbs eines digitalen Guts beschreibt, dann ist diese Beschreibung unabhängig davon, mittels welcher Systeme dieser Vorgang abgewickelt wird und ist somit anwendungsnah.

2.2.2 Ausdruckskraft

Die Ausdruckskraft eines *Sicherheitsmodells* drückt aus, wie viele Anwendungsszenarien, Systeme bzw. andere *Sicherheitsmodelle* sich mit einem gegebenen Mo-

dell beschreiben bzw. simulieren lassen [San92, San88, Sec12]. Je mehr das sind, desto ausdrucksstärker ist dieses Modell. Diese Eigenschaft ist insbesondere bei den generischen *Sicherheitsmodellen* interessant, die im Unterabschnitt 2.8.1 beschrieben werden.

2.2.3 Analysierbarkeit

Eine zur Ausdruckskraft konkurrierende Eigenschaft ist die Analysierbarkeit. Sie sagt aus, ob die sicherheitsspezifischen Eigenschaften eines bestimmten Systems mit einem gegebenen Sicherheitsmodell überprüft werden können. Falls dies möglich ist, kann mit der Analysierbarkeit auch der dazu benötigte Aufwand angegeben werden [Eck09, Sec12, San88].

Die Angabe von der Analysierbarkeit ist insbesondere bei den sehr abstrakten und unendlichen *Sicherheitsmodellen* wichtig, da in diesem Fall die Analyse von den sicherheitsspezifischen Eigenschaften eines Systems nicht immer entscheidbar ist, was solche Modelle als Analysewerkzeug ungeeignet macht.

Im Allgemeinen schließen sich die Ausdruckskraft und Analysierbarkeit gegenseitig aus. Wenn ein *Sicherheitsmodell* eine sehr hohe Ausdruckskraft besitzt, sind seine Definitionen in der Regel für eine erfolgreiche Analyse zu abstrakt. Die Definitionen eines gut analysierbaren Modells sind dagegen zu strikt, um damit viele Anwendungsszenarien und Systeme beschreiben zu können.

2.2.4 Skalierbarkeit

Im Allgemeinen beschreibt die Skalierbarkeit, wie sich die Leistung eines Systems mit wachsenden Anforderungen wie zum Beispiel wachsender Größe bzw. Komplexität ändert [Hil90]. Bezogen auf die *Sicherheitsmodelle* wird mit Skalierbarkeit der Änderungs- bzw. Verwaltungsaufwand angegeben, wenn die Grundelemente bzw. Definitionen wie zum Beispiel die agierenden Subjekte eines Modells angepasst werden müssen.

2.2.5 Formalisierungsgrad

Der Formalisierungsgrad beschreibt, in welcher Notation die Definitionen und Strukturen eines *Sicherheitsmodells* erstellt worden sind. Im Allgemeinen wird zwi-

schen informalen, semiformalen und formalen Sicherheitsmodellen unterschieden [Gri08, Hil90, MSUV02, Sch07].

Die informalen *Sicherheitsmodelle* werden in der Regel mithilfe von den Beschreibungsmitteln notiert, die nicht eindeutig zu interpretieren sind. Solche Beschreibungsmittel sind zum Beispiel die natürliche Sprache oder diagrammatische Darstellungen. Ein informales *Sicherheitsmodell* ist daher in der Regel anschaulich aber für eine eindeutige Interpretation und systematische Analyse nicht präzise genug.

Falls eine Anforderung besteht, sicherheitsrelevante Sachverhalte präzise zu beschreiben, wird zu den semiformalen oder formalen *Sicherheitsmodellen* gegriffen. Wird ein exaktes *Sicherheitsmodell* mittels formaler Methoden notiert, so wird von einem formalen *Sicherheitsmodell* gesprochen. Während die semiformalen *Sicherheitsmodelle* frei aber möglichst präzise beschrieben werden, basieren formale *Sicherheitsmodelle* auf mathematischen Konzepten mit einer wohl-definierten Syntax und einer eindeutigen Semantik. Dies erlaubt eine präzise, aussagekräftige und rechnergestützte Analysierbarkeit von den zu modellierenden Systemen.

Die semiformalen und formalen *Sicherheitsmodelle* sind insbesondere bei der Modellierung von sicherheitskritischen Systemen von Bedeutung, da in diesem Bereich eine besonders hohes Maß an Präzision, Vertrauenswürdigkeit und Zuverlässigkeit verlangt wird. Daher liegt das Hauptaugenmerk dieser Masterarbeit auf der ausführlichen Betrachtung, Analyse und Auswertung solcher *Sicherheitsmodelle*.

2.3 Modellierungsmethoden

Modellierungsmethoden sind Beschreibungsmittel, die bei der Spezifikation von *Sicherheitsmodellen* verwendet werden. Die Anwendung einer bestimmten Methode beeinflusst einige Modelleigenschaften. Zum Beispiel hängt die bereits beschriebene Eigenschaft *Formalisierungsgrad* unmittelbar von der Wahl der Modellierungsmethoden ab. Des Weiteren können Beschreibungsmittel wie beispielsweise UML -Diagramme [uml07] die Anschaulichkeit von Modellen erhöhen.

Für die Beschreibung von dem Verhalten eines Systems existieren im Allgemeinen Modellierungsmethoden, die entweder auf zustandsorientierten oder ereignisorientierten Ansätzen beruhen [HF06].

Bei dem zustandsorientierten Ansatz stehen der Zustand eines Systems und

dessen Änderungen im Vordergrund. Eine auf diesem Konzept basierende Modellierungsmethode beschreibt mögliche Systemzustände und Aktionen, die diese Zustände verändern können. Beispiele für solche Methoden sind *endliche Automaten* und *Petri-Netze* [PW08].

Bei dem ereignisorientierten Ansatz wird von dem Zustandskonzept abstrahiert, wobei nun Grundelementen und die auf diesen Elementen basierenden Ereignisse im Vordergrund stehen. Eine ereignisorientierte Modellierungsmethode ist zum Beispiel die Prozessalgebra *CSP* [Hoa04], die Interaktionen von elementaren Prozessen beschreibt.

Beispiel 2.3.1:
In diesem Beispiel wird ein System aus ereignisorientierter Sicht mithilfe der *CSP*-Notation beschrieben (Angelehnt an [HF06, Hoa04]). In *CSP* wird zwischen primitiven Prozessen und komplexen Prozessen unterschieden. Letztere werden aus elementaren Prozessen zusammengesetzt. Mit einem Prozess wird ein bestimmtes Verhalten in einem System repräsentiert.

Ein Münzautomat soll mit *CSP* simuliert werden. Das System besteht aus den elementaren Prozessen $STOP$ und $ERROR$ und zwei komplexen Prozessen $VendingMashine$ und $SpendChoc$, welche miteinander mittels der Ereignisse $coin$, $noCoin$ und $choc$ interagieren:

VendingMashine $:=$
$$(coin \rightarrow SpendChoc) \mid (noCoin \rightarrow (STOP \parallel ERROR))$$
SpendChoc $:=$
$$(choc \rightarrow STOP)$$

Zunächst nimmt der Münzautomat $VendingMashine$ eine Zahlung für Schokolade entgegen. Dabei reagiert es auf das Ereignis $coin$ und geht zu der Warenausgabe $SpendChoc$ über. Falls eine erwartete Zahlung nicht stattgefunden hat, stoppt der Automat und reagiert parallel mit einer Fehlermeldung, die im Prozess $ERROR$ angezeigt wird. In dem Prozess $SpendChoc$ geschieht dann die Ausgabe der Schokolade - $choc$. Danach verhält sich der Automat wie der Prozess $STOP$. Im Prozess $STOP$ finden keine weiteren Interaktionen statt.

Um eine bessere Vergleichbarkeit zu erreichen, wird für die Spezifikation der

in dieser Masterarbeit beschriebenen formalen Sicherheitsmodelle der zustandsorientierte Ansatz modellübergreifend eingesetzt. Die dabei verwendete Modellierungsmethode wird in dem Kapitel 3 beschrieben.

2.4 Anwendungsbereiche

Die Sicherheitsmodelle können in unterschiedlichen Anwendungsbereichen eingesetzt werden. Die Entscheidung, in welchem Anwendungsbereich ein Modell verwendet werden soll, bestimmt oft über die Eigenschaften, Struktur bzw. Wahl der Modellierungsmethoden bei der Erstellung dieses Modells. In jedem spezifischen Bereich werden unterschiedliche Modelleigenschaften priorisiert.

Allgemein wird zwischen den akademischen, kommerziellen und militärischen Anwendungsbereichen unterschieden. Für den akademischen Bereich sind in der Regel abstrakte, unendliche und formale *Sicherheitsmodelle* typisch, die keinen konkreten Bezug zu einer Anwendungsdomäne besitzen. In diesem Fall rücken die theoretischen Aspekte der Modellierung wie zum Beispiel die Ausdruckskraft und Analysierbarkeit in den Vordergrund.

Im Gegensatz dazu sind die *Sicherheitsmodelle* aus den kommerziellen und militärischen Bereichen praxisorientiert. Ihre Struktur ist oft an eine konkrete Anwendung oder Anwendungsdomäne angelehnt. Priorisiert werden dabei Modelleigenschaften wie zum Beispiel die Skalierbarkeit.

2.5 Sicherheitsaspekte

Orthogonal zu den Anwendungsbereichen beschreiben die Sicherheitsaspekte den Modellierungsgegenstand eines Modells. Verschiedene *Sicherheitsmodelle* haben ihr Hauptaugenmerk auf der Beschreibung von den unterschiedlichen sicherheitsrelevanten Aspekten eines Systems. Dies kann zum Beispiel die Modellierung von Zugriffskontrolle, Informationsflüssen oder Transaktionen zwischen den kommunizierenden Partnern sein. Die Sicherheitsaspekte bestimmten in der Regel die Struktur und Komponenten eines *Sicherheitsmodells*.

2.6 Sicherheitsziele

Jedes *Sicherheitsmodell* besitzt ein übergeordnetes Sicherheitsziel, welches auf den *Modellierungsaspekten* beruht. Ein Sicherheitsziel beschreibt dabei, welche Wirkung mit einem gegebenen Sicherheitsmodell erzielt werden soll. Anhand des Sicherheitsziels kann entschieden werden, ob ein Modell für ein Anwendungsszenario bzw. ein zu modellierendes System geeignet ist [Gri08, GP]. Ein mögliches Sicherheitsziel wäre zum Beispiel, die sensiblen Daten eines IT-Systems vor unerlaubten Zugriffen zu schützen.

2.7 Entstehungsprozess

In diesem Abschnitt wird der allgemeine Entstehungsprozess und der daraus resultierenden Grundaufbau bzw. die Einsatzziele eines *Sicherheitsmodells* beschrieben [MSUV02].

In der Abbildung 2.1 ist ein Entstehungsprozess der semiformalen bzw. formalen *Sicherheitsmodelle* dargestellt. Die dabei angegebenen Entwicklungsschritte sind insbesondere für die praxisorientierten Anwendungsbereiche relevant. Einige dieser Schritte wie zum Beispiel die Erstellung einer Sicherheitspolitik oder einer Systemspezifikation können bei der Entwicklung akademischer *Sicherheitsmodelle* eventuell wegfallen.

In dem linken Teil der Abbildung 2.1 ist im Groben ein Systementwicklungsprozess beschrieben, welcher aus dem Anwendungsszenario, dem Systementwurf und der Implementierung besteht. Der Einsatz von den formalen bzw. semiformalen *Sicherheitsmodellen* kann in den unterschiedlichen Schritten der Systementwicklung geschehen [CW96, Gri08].

Zum einen können die *Sicherheitsmodelle* anhand eines Anwendungsszenarios oder während der Systementwicklung erstellt werden. In diesem Fall fungieren sie als eine Art Bauplan [Gri08] und legen eine Grundlage für die Implementierung von Mechanismen, die die Sicherheit der zukünftigen Systeme gewährleisten sollen. Zum anderen können sie zur Analyse eines bereits fertigen Systems eingesetzt werden.

In dem ersten Schritt wird ein sicherheitskritisches Szenario oder System auf Gefahren und Risiken hin analysiert, wobei entsprechend der *Sicherheitsziele* und zu dem *Anwendungsbereich* alle sicherheitsrelevanten Aspekte erfasst werden. Da-

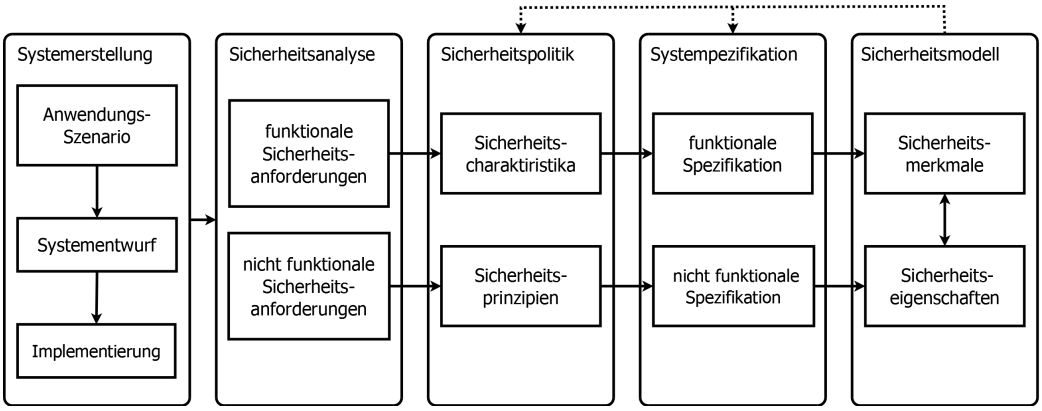


Abbildung 2.1: Entstehungsprozess eines Sicherheitsmodells

raus werden die funktionalen und nicht funktionalen Sicherheitsanforderungen abgeleitet. Mit ersteren werden die Anforderungen an die Funktionalität des sicherheitsrelevanten Teils eines Systems wie zum Beispiel das sicherheitkritische Systemverhalten definiert. Die nicht funktionalen Sicherheitsanforderungen beschreiben dabei, welche sicherheitsrelevanten Eigenschaften dieses System besitzen soll.

In dem nächsten Schritt wird anhand der Anforderungsanalyse eine Sicherheitspolitik definiert. Eine Sicherheitspolitik ist ein Regelwerk, mit dem festgelegt wird, wie die Sicherheitsanforderungen erfüllt werden müssen [Sec12]. Eine Sicherheitspolitik wird informal definiert und besteht aus Sicherheitscharakteristika und Sicherheitsprinzipien [MSUV02]. Die Sicherheitscharakteristika korrespondieren zu den funktionalen Sicherheitsanforderungen und beschreiben die Regeln, Begriffe, Aktionen und Strukturen, die die funktionalen Anforderungen erfüllen. Dementsprechend werden in den Sicherheitsprinzipien die Regeln festgelegt, die die in den nicht funktionalen Anforderungen geforderten Eigenschaften und Grundsätze erfüllen.

Des Weiteren werden anhand der Regeln der Sicherheitspolitik die sicherheitsrelevanten Teile eines Systems und ihre Eigenschaften formal beschrieben. Diese formale Systemspezifikation dient nun als Vorlage für die Erstellung eines formalen bzw. semiformalen *Sicherheitsmodells*.

Ein *Sicherheitsmodell* besteht dabei aus zwei grundlegenden Elementen: *Sicherheitsmerkmale* und *Sicherheitseigenschaften*. Die *Sicherheitsmerkmale* enthalten alle für die Gültigkeit der *Sicherheitseigenschaften* relevanten Aspekte und Sicherheits-

funktionen eines Systems. Dieser Bestandteil des Modells korrespondiert zu den Sicherheitscharakteristika der zugrundeliegenden Sicherheitspolitik. Die *Sicherheitseigenschaften* sind formale Beschreibungen der in den nicht funktionalen Sicherheitsanforderungen geforderten sicherheitsrelevanten Systemeigenschaften, welche auf den Regeln und Konzepten der Sicherheitsprinzipien der zugrundeliegenden Sicherheitspolitik beruhen [MSUV02].

Wenn ein *Sicherheitsmodell* erstellt wurde, werden damit mehrere Konsistenzbeweise geführt. Zum einen soll überprüft werden, ob die *Sicherheitsmerkmale* dieses Modells mit der funktionalen Systemspezifikation und mit den Sicherheitsprinzipien der zugrundeliegenden Sicherheitspolitik übereinstimmen. Falls dies nicht der Fall ist, weist die Sicherheitspolitik oder die formale Spezifikation Inkonsistenzen oder Fehler auf, die beseitigt werden müssen. Anderenfalls bildet das erstellte *Sicherheitsmodell* die geforderte Funktionalität ab.

Falls alle Inkonsistenzen beseitigt werden konnten, werden nun die *Sicherheitseigenschaften* des *Sicherheitsmodells* untersucht. Konnten die *Sicherheitseigenschaften* nachgewiesen werden, erfüllt die Sicherheitspolitik die gestellten Sicherheitsanforderungen. Im Falle eines existierenden Systems bedeutet dies, dass ein *Sicherheitsmodell* gefunden wurde, welches dieses System zutreffend beschreibt und dass dieses System sicher ist.

Wenn es sich um ein System handelt, in dem die sicherheitsrelevanten Aspekte noch nicht umgesetzt wurden, kann die sichere Sicherheitspolitik und formale Systemspezifikation zur Entwicklung von den Mechanismen verwendet werden, die diese Aspekte implementieren.

2.8 Modellbildung

Bei dem *Entstehungsprozess* existieren mehrere Ansätze, wie ein *Sicherheitsmodell* entwickelt werden kann.

Zum einen kann ein *Sicherheitsmodell* das Resultat einer kompletten Neuentwicklung sein. In diesem Fall wird es im Kontext eines konkreten Szenarios bzw. System gebildet. Die Anwendung dieser Methode verspricht ein Modell, welches exakt zu dem modellierten System passt und dabei die gewünschten Modelleigenschaften aufweist. Die Konsistenz (Widerspruchsfreiheit) und der Aufwand bei dem Nachweis der Sicherheitseigenschaften solcher Modelle sind dennoch schwer abzuschätzen.

Weitere Möglichkeiten der Modellbildung sind die Instantiierung von *generischen Sicherheitsmodellen* und das *Model-Engineering*, welche in nachfolgenden Unterabschnitten beschrieben werden.

2.8.1 Generische Sicherheitsmodelle

Ausgewählte Komponenten der generischen Sicherheitsmodelle sind variabel definiert, sodass der Einsatz solcher Modelle bei einer Reihe bestimmter Anwendungsszenarien möglich ist [MSUV02, GP]. Somit steht ein *generisches Sicherheitsmodell* für eine ganze Reihe von *Sicherheitsmodellen*.

Ein konkretes Sicherheitsmodell wird gebildet, indem die variablen Komponenten eines *generischen Modells* an ein spezielles Anwendungsszenario angepasst werden, wobei ihnen anwendungsspezifische Werte zugeordnet werden. Dieser Prozess wird *Instantiierung* genannt. In diesem Fall ist die Bildung eines konkreten Modells weniger aufwändig als bei einer Neuentwicklung, da dabei die allgemeine Grundstruktur bereits existiert.

Die Modell- und Sicherheitseigenschaften der *generischen Sicherheitsmodelle* sind in der Regel bekannt und werden von dem generischen Modell an die Instanzen vererbt. Diese Eigenschaften sind jedoch generisch und müssen deswegen bei den Instanzen zwar nicht mehr bewiesen aber überprüft werden.

Die Auswahl und Anpassung eines *generischen Sicherheitsmodells* an ein konkretes Anwendungsszenario können sich als schwierig erweisen, da dann nicht mehr sichergestellt ist, dass die dabei gebildete Instanz die Anforderungen an ein Modell bzw. Szenario exakt erfüllt.

2.8.2 Model-Engineering

Im Rahmen des Model-Engineerings werden neue Modelle gebildet, indem Grundelemente der bereits bekannten *Sicherheitsmodellen* wiederverwendet werden.

Im Mittelpunkt des *Model-Engineerings* steht ein abstraktes, universelles Kernmodell, welches das Fundament für eine Gruppe von Modellen aus einem spezifischen Anwendungsbereich bzw. einer Anwendungsdomäne legen [KP11, Sec12]. Entsprechend der jeweiligen Anforderungen werden aus einem Kernmodell durch Anwendung von Model-Engineering-Methoden spezifische Modelle abgeleitet. Aus den resultierenden *Sicherheitsmodellen* können wiederum neue Modelle oder

Hybridmodelle gebildet werden, die Eigenschaften der Elternmodelle vereinen. Die typischen Engineering-Methoden umfassen zum einen Bestimmung der Grundkomponenten und zum anderen Spezifizierung, Komposition, Kombination dieser Komponenten und Ergänzung neuer Elemente.

Das Ziel des *Model-Engineerings* ist die Bildung neuer Modelle mit den gewünschten Eigenschaften durch Wiederverwendung bekannter Modelle und Modellkomponenten.

Kapitel 3

Klassifizierung von Sicherheitsmodellen

Um einen umfassenden Überblick über die semiformalen bzw. formalen Sicherheitsmodelle zu geben, wird in diesem Kapitel mithilfe von Reverse-Engineering ein Klassifizierungsschema erstellt. Die hierbei gewonnenen Informationen werden in den Kapiteln 4 und 5 zu einer systematischen Beschreibung und Analyse von ausgewählten Sicherheitsmodellen verwendet. Die grundlegende Vorgehensweise ist dabei die konzeptuelle Seite der Sicherheitsmodelle im Allgemeinen mithilfe eines Klassifizierungsschema zu erfassen, um die Umsetzung dieser Konzepte bei der Beobachtung konkreter Modelle in Kapitel 4 zu veranschaulichen.

Zunächst werden die allgemeinen Aspekte und der Aufbau von dem Klassifizierungschema beschrieben. In den weiteren Abschnitten werden die Komponenten des Schemas im Einzelnen behandelt.

3.1 Klassifizierungsschema

Bei der Erstellung des Klassifizierungsschemas wurden zunächst im Rahmen des Revers-Engineering einige der bereits veröffentlichen Sicherheitsmodelle mit dem Ziel analysiert, die grundlegenden Strukturen und Beziehungen dieser Modelle zu ermitteln. Als Grundlage für die Modellanalyse wurde die Klassifizierung von Eckert in [Eck09] verwendet.

Im Rahmen des *Klassifizierungsschemas* werden sogenannte Modellklassen oder Modellgruppen gebildet. Diese Klassen stellen eine Aggregation aller Sicherheitsmodelle dar, die eine gemeinsame Struktur, Eigenschaft oder Beziehung miteinander verbinden. Als Grundlage für die Bildung dieser Klassen werden die *Anwendungsbereiche*, die dabei zu modellierende *Sicherheitsaspekte* und die daraus resultierenden Modellierungskonzepte verwendet. Die Modellierungskonzepte beschreiben hierbei, wie und mit welchen Mitteln die *Sicherheitsaspekte* modelliert werden sollen. Die Modellklassen sollen somit die Menge der Konzepte darstellen, die bestimmten *Sicherheitsaspekte* eines bestimmten Anwendungsbereichs erfüllen.

Eckert definiert in [Eck09] auf Basis von klassischen *Sicherheitsaspekten* zwei grundlegende Klassen von Sicherheitsmodellen: die Zugriffskontrollmodelle und die Informationsflussmodelle. Des Weiteren führt Grimm in [Gri08] eine weitere Klasse ein, nämlich die Transaktionsmodelle. Sie beruhen auf den *Sicherheitsaspekten* und den dazugehörigen Umsetzungskonzepten, die die sichere Kommunikation bzw. Kooperation in offenen Systemen regeln.

Durch die Anpassung und Erweiterung der erwähnten Klassen, werden nun weitere Modellklassen definiert. Die Gemeinsamkeit dieser Modellklassen bilden das Klassifizierungsschema, welches in der Abbildung 3.1 dargestellt ist. Das Schema wird dabei mithilfe eines UML-Klassendiagramms beschrieben.

Die abstrakte Oberklasse `CoreModel` beschreibt eine generelle Form von Sicherheitsmodellen. Sie umfasst die Grundelemente bzw. Kernkonzepte aller Sicherheitsmodelle (siehe Abschnitt 3.2).

In den abgeleiteten Klassen `AccessControlModel`, `InformationFlowModel` und `TransactionModel` werden die von der Oberklasse geerbten Grundelemente bzw. Kernkonzepte an die jeweiligen *Sicherheitsaspekte* und *Anwendungsbereiche* angepasst und um weitere sicherheitsspezifische Elemente und Konzepte erweitert (siehe Abschnitte 3.3, 3.4 und 3.5).

Nach dem Vererbungsprinzip werden schließlich die Klassen der Zugriffsmatrixmodelle (`AccessMatrixModel`), attributbasierten Modelle (`AttributeBasedModel`), rollenbasierten Modelle (`RoleBasedModel`) und Verband-basierten Modelle (`LatticeBasedModel`) gebildet, indem Elemente ihrer Oberklassen verfeinert und erweitert werden. Dabei wird jede dieser Klassen durch klassenspezifische Komponente und Konzepte spezifiziert, welche in den weiterführenden Abschnitten erläutert werden.

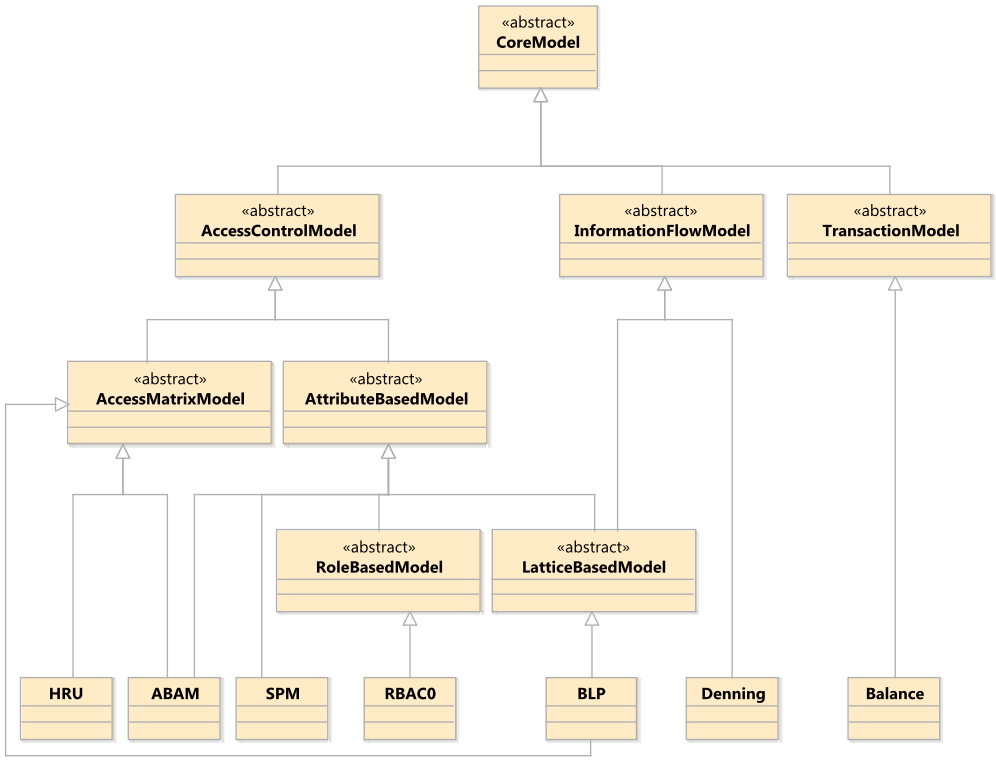


Abbildung 3.1: Modellklassifikation

Die beschriebenen abstrakten Modellklassen bilden eine Grundlage für ein Klassifizierungsschema der Sicherheitsmodelle. Ein Sicherheitsmodell wird mit diesem Schema klassifiziert, indem es in die Modellklasse bzw. Modellklassen eingeordnet wird, diese seine Eigenschaften beschreiben.

In der unteren Ebene des Diagramms aus der Abbildung 3.1 wurden ausgewählte Sicherheitsmodelle klassifiziert. Dabei handelt es sich um generische Sicherheitsmodelle, die die Modellklassen aller ihrer Instanzen bilden:

- Harrison-Ruzzo-Ullman-Modell (`HRU`)
- Attribute Based Access Matrix Model (`ABAM`)
- Schematic Protection Model (`SPM`)
- Bell-LaPadula Modell (`BLP`)
- Rollenbasiertes Zugriffskontrollmodell (`RBAC0`)

- Denning-Modell (`Denning`)
- Gleichgewichtsmodell (`Balance`)

Diese Sicherheitsmodelle werden in Kapitel 4 mithilfe der erstellten Klassifizierungsschemas beschrieben.

3.2 Kernmodell

Die Klasse *Kernmodell* beschreibt diejenige Grundelemente, die das Fundament jedes Sicherheitsmodells bilden.

Wie in dem Grundlagenkapitel bereits erwähnt wurde, besteht ein Sicherheitsmodell im Allgemeinen aus zwei Ebenen: den Sicherheitsmerkmalen und den Sicherheitseigenschaften. Die Sicherheitsmerkmale simulieren die Arbeitsweise eines Systems. Die Sicherheitseigenschaften beschreiben die sicherheitsrelevanten Eigenschaften, die von den Sicherheitsmerkmalen, also dem simulierten System, immer eingehalten werden müssen.

Um die Arbeitsweise des Systems zu modellieren, wird in dieser Arbeit ein Zustandsübergangssystem verwendet. Im Allgemeinen besteht ein Zustandsübergangssystem aus einer Zustandsmenge und einer Zustandsübergangsrelation. In der Systementwicklung ist ein Zustand ein Systemabbild, das alle variablen Systemgrößen zu einem diskreten Zeitpunkt umfasst. Bezogen auf Sicherheitsmodelle, umfasst ein Zustand alle sicherheitsrelevanten Größen bzw. Bestandteile eines Systems. Die Gesamtheit aller Zustände bildet einen Zustandsraum. Eine Zustandsübergangsfunktion definiert für einen Zustand die Menge seiner Nachfolgezustände. Ausgelöst werden Zustandsübergänge durch die im System stattfindende Aktionen, die zur Änderung eines Systemzustands führen.

In der Abbildung 3.2 sind die Bestandteile und Beziehungen eines Zustandsübergangssystems dargestellt, welche gleichzeitig die Grundkomponenten eines Sicherheitsmodells bilden. Das in dieser Arbeit verwendete Zustandsübergangssystem wird dabei als Tupel (Q, Σ, δ, q_0) notiert, wobei gilt:

- Q - Menge aller Systemzustände,
- Σ - Menge der möglichen Eingaben, die das System verarbeiten kann,
- $\delta : Q \times \Sigma \longrightarrow Q$ - Zustandsübergangsfunktion und

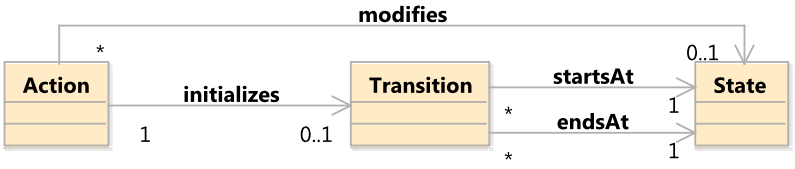


Abbildung 3.2: Zustandsübergangssystem

- $q_0 \in Q$ - der Initialzustand.

Die Sicherheitseigenschaften eines Sicherheitsmodells können nun mithilfe des Zustandsübergangssystems beschrieben und überprüft werden. Dazu werden sie auf die Spezifikation von sicheren Systemzustände und erlaubten Zustandsübergänge abgebildet. In Rahmen dieser Spezifikation werden die Regeln festgelegt, die durch geforderte Restriktionen, *Sicherheitsanforderungen* und *Sicherheitsaspekten* bestimmt werden. Ein Zustand bzw. ein Zustandsübergang ist sicher, wenn ein System in diesem Zustand bzw. bei diesem Zustandsübergang die definierten Regeln erfüllt.

Anhand der induktiven Eigenschaft eines Automaten kann überprüft werden, ob ein System die geforderten Sicherheitseigenschaften besitzt und damit die gestellten Sicherheitsanforderungen erfüllt. Wenn ein System sich in einem sicheren Zustand befindet, versetzt ein erlaubter Zustandsübergang dieses System wieder in einen sicheren Zustand. Falls das System sich stets in sicheren Zuständen befindet, besitzt es die geforderten Sicherheitseigenschaften und ist somit im Sinne des gestellten Sicherheitsziels sicher .

3.3 Zugriffskontrollmodelle

Ein typisches Sicherheitsproblem von IT-Systemen ist die Kontrolle der Zugriffe auf Systemressourcen. Die Sicherheitsmodelle, die diesen *Sicherheitsaspekt* behandelt, bilden die Klasse der *Zugriffskontrollmodelle*. Im Mittelpunkt solcher Modelle steht der Schutz von internen Systemressourcen bzw. Systemdaten. Ihr Aufbau und konzeptuelle Umsetzung ist dementsprechend für die Verwaltung der Zugriffsrechte und für die Überwachung der Datenzugriffe konzipiert [Eck09].

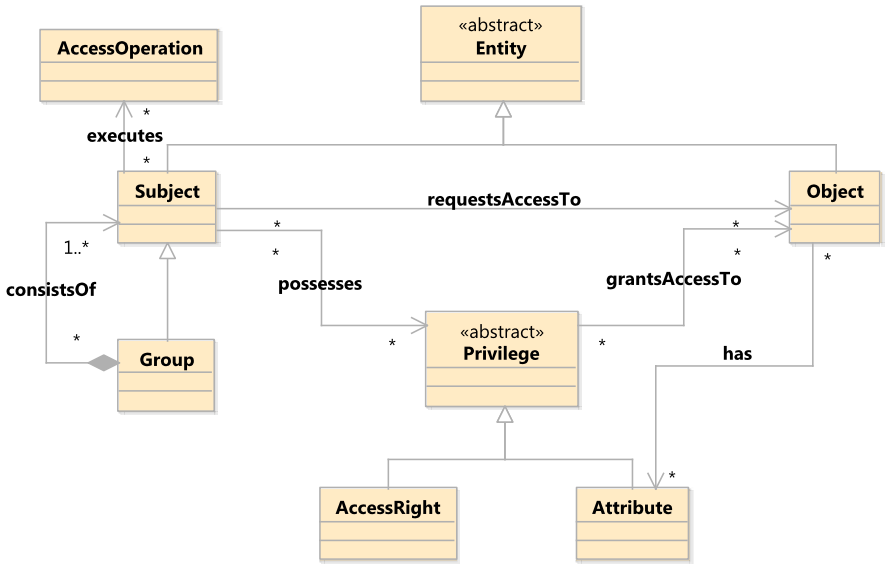


Abbildung 3.3: Klasse der Zugriffskontrollmodelle

In der Abbildung 3.3 sind abstrakte Merkmale und Beziehungen der Zugriffskontrollmodelle dargestellt. Die Grundelemente jedes Zugriffskontrollmodells bilden die Subjekte (`Subject`), Objekte (`Object`) und Privilegien (`Privilege`).

Subjekte sind die aktiven Einheiten eines IT-Systems oder Produkts. Ganz allgemein werden Benutzer, Programme oder Prozesse als Subjekte modelliert. Sie versuchen auf die Objekte zuzugreifen, indem sie bestimmte Zugriffsoperationen ausführen. Mehrere Subjekte können zu einer Gruppe (`Group`) zusammengefasst werden, die dann als eine Subjekteinheit fungiert.

Objekte sind die zu schützenden Einheiten eines IT-Systems oder Produkts wie beispielsweise Dateien, Datenbanktabellen oder andere Systemressourcen. Subjekte können aus Modellierungssicht ebenfalls zu Objekten werden, falls auf sie zugegriffen wird. So kann beispielsweise ein Prozess auf einen anderen Prozess zugreifen. Der zweite Prozess nimmt in diesem Fall die Rolle eines Objekts ein.

Um einen Zugriff auf Objekte zu erlangen, müssen Objekte bestimmte Privilegien besitzen. Der Begriff „Privilegien“ wird hier abstrakter aufgefasst. In der Regel werden unter Privilegien die Zugriffsrechte verstanden. Aber auch die Attribute der Subjekte wie Rollen, Typen, Sicherheitsklassen oder Alter und Wohnort eines Subjekts werden in dieser Masterarbeit als Privilegien verstanden. Mit

anderen Worten sind Privilegien die Gesamtheit aller Attribute und Zugriffsrechte eines Subjekts, die es besitzen muss, um auf das von ihm gewünschte Objekte zugreifen zu können [San92].

Wie Subjekte können auch Objekte Attribute haben. Sie können aber keine Zugriffsrechte besitzen. Sowohl Subjektattribute als auch Objektattribute werden hier als spezifische Merkmale verstanden, die die besonderen Eigenschaften der Entitäten ausdrücken oder deren Angehörigkeit zu einer bestimmten Kategorie festlegen.

Mit Zugriffsrechten wird festgelegt, welche Zugriffsoperationen ein Subjekt auf einem bestimmten Objekt ausführen darf. Eine Operation (`Operation`) definiert, wie auf die Objekte zugegriffen werden kann. Den Lese- bzw. Schreibrechten liegen zum Beispiel die Operationen `lesen` und entsprechend `schreiben` zugrunde. Die Zugriffsrechte können auf unterschiedliche Art und Weise modelliert werden. In dieser Arbeit wird eine explizite Modellierung von Zugriffsrechten bevorzugt. Das bedeutet, dass die Zugriffsrechte durch explizite Angaben der Zugriffsrechte wie beispielsweise *readRight* oder *writeRight* modelliert wird.

In der Theorie der Zugriffskontrollmodelle wird implizit oder explizit zwischen den neutralen Privilegien und den Kontrollprivilegien unterschieden. Im Gegensatz zu neutralen Privilegien kann der Besitz von Kontrollprivilegien Subjekte zu den Operationen autorisieren, die Änderungen im System verursachen können [San88].

Diese Unterscheidung ist auf verschiedene Konzepte der Zugriffskontrollmodellierung zurückzuführen. In einigen Modellen steht der Transport und die Verbreitung der Privilegien im Vordergrund. Dabei verursachen solche Operationen wie `lesen` oder `schreiben` keine Zustandsänderungen. Eine Übertragung von Privilegien zwischen Subjekten würde dagegen den Zustand eines Systems ändern[1].

In der Abbildung 3.3 wurden die für die Zugriffskontrolle grundlegenden Elemente und deren Beziehungen dargestellt. Nun soll mit der Abbildung 3.4 der Zusammenhang mit der abstrakten Struktur der Zugriffskontrollmodelle hergestellt werden. Im Mittelpunkt jedes Zugriffskontrollmodells steht die Beschreibung der Autorisierungsprozesse. In Verbindung damit können folgende klassenspezifische Beschreibungselemente definiert werden: Zugriffskontrollrelationen (`AccessControlRelation`), Sicherheitsschema (`SecurityScheme`) und

[1] Vgl. *Schematic Protection Model* in Abschnitt 4.3

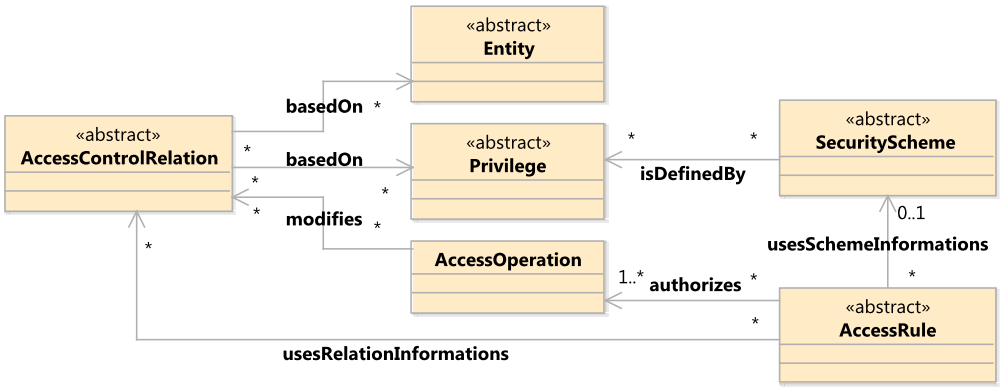


Abbildung 3.4: Merkmale von Zugriffskontrollmodellen

Zugriffsregeln (`AccessRule`).

Unter Zugriffskontrollrelationen werden hier alle Relationen verstanden, die die Verteilung von Privilegien in einem System beschreiben. Dazu gehören zum Beispiel die Zuordnungen der Attribute zu den Subjekten und Objekten. In diesem Fall beschreibt eine Zugriffskontrollrelation welche Entitäten welche Attribute besitzen. Eine der wichtigsten Zugriffskontrollrelationen ist die Relation, die Beziehungen zwischen Entitäten und Zugriffsrechten beschreibt. Mit anderen Worten wird damit die Verteilung von Zugriffsrechten in einem System festgelegt. Sei S die Menge der Subjekte, O die Menge der Objekte und R die Menge der Zugriffsrechte im System, dann gilt:

$$M : S \times O \rightarrow R.$$

Mit der Zugriffskontrollrelation M wird definiert, welche Subjekte welche Zugriffsrechte an den Elementen aus O besitzen.

Jedes Zugriffskontrollmodell besitzt in der Regel mindestens eine Zugriffskontrollrelation. Durch die Aktivitäten der Subjekte kann eine Zugriffskontrollrelation modifiziert werden. Zum Beispiel kann ein Subjekt ein Zugriffsrecht oder ein Attribut zugewiesen bekommen.

Ein Sicherheitsschema beschreibt die Beziehungen zwischen Privilegien. Das einfachste Beispiel für ein Sicherheitsschema ist ein implizites Schema, welches keine konkreten Beziehungen zwischen den Privilegien definiert. Angenommen, Subjekte in einem System besitzen bestimmte Attribute. In einem impliziten Schema stehen diese Attribute in keiner Beziehung. Im Gegensatz dazu existieren im

expliziten Schema semantische Beziehungen wie beispielsweise „hat höhere Geheimhaltungsstufe als“[2]. Ein explizites Sicherheitsschema auf Basis von Typen ist im Abschnitt 4.3 vertreten.

Ein Zugriffskontrollmodell besitzt immer ein Sicherheitsschema. Die expliziten Schemas werden hauptsächlich dann eingesetzt, wenn globale Zugriffsentscheidungen definiert werden müssen. Die Schemas werden in der Regel einmalig von einem Sicherheitsadministrator festgelegt und zur Laufzeit eines Systems nicht mehr geändert.

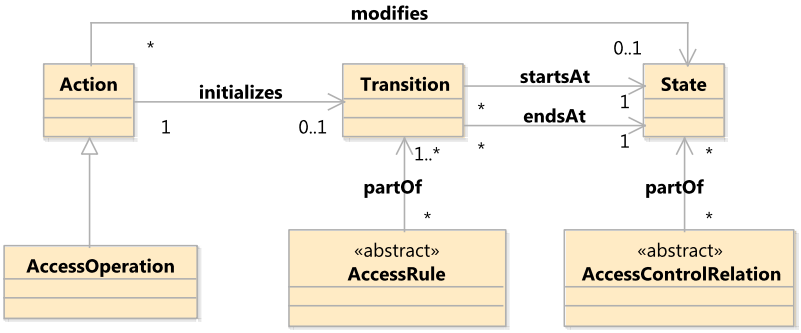


Abbildung 3.5: Beziehungen zu dem Kernmodell

Für die Durchsetzung des Zugriffsschutzes werden in einem Zugriffskontrollmodell Zugriffsregeln (`AccessRule`) modelliert. Mit den Zugriffsregeln wird bestimmt, ob die von den Subjekten ausgeführten Zugriffsoperationen zulässig sind. Dies geschieht, indem anhand einer aktuellen, in den Zugriffskontrollrelationen definierten Privilegienverteilung und der globalen Restriktionen aus einem Sicherheitsschema eine Entscheidung gefällt wird, ob eine bestimmte Zugriffsoperation ausgeführt werden darf.

Die Abbildung 3.5 stellt den Zusammenhang zwischen den Elementen der Zugriffskontrollmodelle und den Elementen des Kernmodells dar. Die Zugriffsoperationen sind Aktionen des Zustandsübergangssystems, die einen Zustandswechsel hervorrufen können. Eine Transitionen wird dabei mithilfe von Zugriffsregeln repräsentiert. Dabei können dieselben Zugriffsregeln mehrere Transitionen beschreiben. Die Zugriffskontrollrelationen, die von Subjekten geändert werden können, bilden die Zustände eines Systems.

[2]Die unterschiedlichen Schemas werden im Abschnitt 3.3.2 ausführlicher behandelt.

Beispiel 3.3.1:
Dieses Beispiel soll den Zusammenhang zwischen Zugriffsregeln und den Elementen eines Zustandsübergangssystems verdeutlichen.

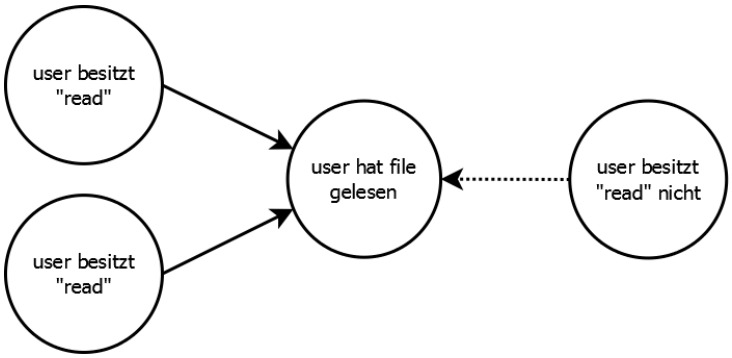


Abbildung 3.6: Transitionen im Beispielsystem

Sei ein Beispielsystem gegeben, in dem ein Subjet *user*, ein Objekt *file* und das Privileg *read* existieren. *user* möchte auf *file* lesend zugreifen. Durch eine Zugriffsregel der zugrundeliegenden Sicherheitspolitik kann festgelegt werden, dass *user* nur dann *file* lesen darf, wenn es ein Privileg $read \in P$ besitzt. Bezogen auf ein Zustandsübergangssystem beschreibt diese Regel eine Menge von Transitionen, die in Zuständen des Systems beginnen, in denen *user* das Privileg $read$ besitzt und in Zuständen enden, in denen $user\ file$ gelesen hat. Die Übergänge sind die durch die angegebenen Zugriffsregel definierten Transitionen (Abbildung 3.6). Die gestrichelte Transition verletzt die Zugriffsregeln und darf somit im System nicht vorkommen.

3.3.1 Zugriffsmatrix-Modelle

Eines der wichtigen Beschreibungselemente der Zugriffskontrolle sind die Zugriffskontrollrelationen, welche die Verteilung der Privilegien in einem System beschreiben. Es existiert eine ganze Reihe von Zugriffskontrollmodellen, in denen spezielle Konstrukte verwenden werden, um die Zugriffskontrollrelationen zu definieren. Dabei kann aus Modellierungssicht zwischen *low-level*- und *high-level*-Konstrukten unterschieden werden.

Ein *high-level*-Konstrukt ist eine abstrakte Relation, die einen bestimmten Sachverhalt ausdrückens soll. Im Gegensatz dazu wird ein low-level-Konstrukt gebil-

det, indem eine abstrakte Relation auf eine allgemein bekannte, leicht verständliche und leicht implementierbare Struktur abgebildet wird. Typische Beispiele für solche Strukturen sind Matrizen und Graphen. Eine Zugriffsmatrix ist demnach ein *low-level*-Konstrukt.

In der Klasse der Zugriffsmatrix-Modelle werden Matrizen in erster Linie für die Beschreibung der Zusammenhänge zwischen Entitäten und Zugriffsrechten eingesetzt. Aber auch die Beschreibung anderer Beziehungen wie zum Beispiel Subjekt-Rolle-Zuordnung aus 3.3.2.2 ist mit den Matrizen möglich.

	$Prozess1$	$Datei$
$User1$	$true$	$false$
$Prozess1$	$false$	$true$
$Pozess2$	$false$	$false$

(a) Bool'sche Matrix

	$Prozess1$	$Datei$
$User1$	$\{execute\}$	$\{read\}$
$Prozess1$	$\emptyset$	$\{write\}$
$Pozess2$	$\emptyset$	$\{read, write\}$

(b) Matrix mit Zugriffsrechten

Abbildung 3.7: Matrixmodellierung

Eine Matrix kann auf unterschiedliche Weise modelliert werden. In der Abbildung 3.7 sind zwei exemplarische Modellierungsmöglichkeiten abgebildet. In der Regeln bilden Subjekte die Zeilen und Objekte die Spalten einer Matrix. Sollten Subjekte auch zu schützende Objekte sein, so müssen sie ebenfalls in den Spalten der Matrix aufgeführt werden.

Im Fall *(a)* wird eine bool'sche Matrix modelliert. Dabei werden Zugriffsrechte mithilfe boolescher Werte definiert. Besitzt ein Subjekt ein Zugriffsrecht auf ein bestimmtes Objekt, so wird in die entsprechende Zelle $true$ und anderenfalls $false$ eingetragen. Mit so einer Matrix kann zwar festgelegt werden, ob ein Subjekt auf ein Objekt zugreifen darf, aber nicht welche Zugriffsoperationen dabei ausgeführt werden dürfen. Daher stellt die Matrix aus Abbildung 3.7 *(b)* die übliche Form einer Matrix dar, die in der Regel in Sicherheitsmodelle eingesetzt wird. Hier wird in den Zellen der Matrix die Menge der Zugriffsrechte eingetragen. Zum Beispiel besitzt *User1* das Recht den Prozess *Prozess1* auszuführen.

Eine Zugriffsmatrix ist eine der einfachsten Wege, eine Rechteverteilung in einem System zu beschreiben. Hierzu können standardisierte Zugriffsoperationen festgelegt werden, die für Zugriffsmatrix-Modelle eine einheitliche formale Basis bieten. Solche Operationen sind auf die Änderungen der Zugriffsmatrix bezogen.

Außerdem ist eine Matrix aus technischer Sicht in einem Computersystemen ohne großen Aufwand realisierbar. Daher weisen die Zugriffsmatrix-Modelle in

der Regel eine gute Implementierbarkeit auf.

Der Einsatz einer Zugriffsmatrix hat aber auch seine Grenze. So werden die Zugriffsrechte pro Subjekt und Objekt vergeben. Dadurch können zwar diskrete Rechteverteilungen beschrieben werden. Systemweite Zugriffsentscheidungen können damit aber nur sehr schwer oder gar nicht umgesetzt werden. Beispielsweise kann für ein Subjekt ein genereller Zugriffsverbot auf eine bestimmte Gruppe von Objekten mit einer Matrix schwer realisierbar werden. Sobald ein Subjekt die entsprechenden Zugriffsrechte erlangt hat, kann er uneingeschränkt auf die Objekte zugreifen. Häufig wird deswegen die Definition von zusätzlichen Regeln benötigt. Die Klasse der Sicherheitsmodelle, die solche Regeln auf Basis von Attributen umsetzten, wird in dem nächsten Abschnitt beschrieben.

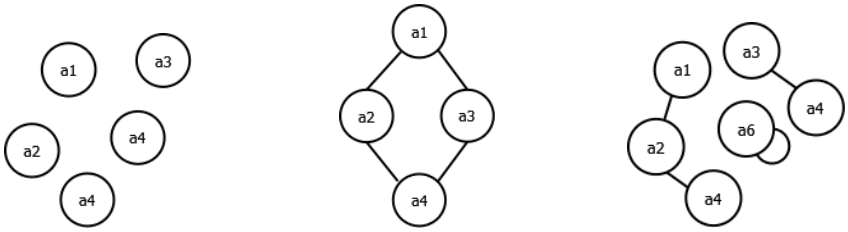
3.3.2 Attributbasierte Zugriffskontrollmodelle

In den attributbasierten Zugriffskontrollmodellen werden die Zugriffsentscheidungen anhand von Attributen getroffen. Wie im Abschnitt 3.3 erwähnt wurde, werden in dieser Arbeit unter anderem Sicherheitsklassen, Entitätentypen oder Rollen als Attribute verstanden. Ein Attribut soll dabei spezifische Eigenschaften bzw. Merkmale eines Subjekts oder Objekts ausdrücken, welche in einem anwendungsspezifischen Kontext relevant sind [ZLN05, PDMP]. Ein Beispiel dafür ist ein Betriebssystem, wo zwei Attribute *Administrator* und *User* definiert sind. So kann ein Subjekt mit dem Attribut *Administrator* das gesamte System administrieren, wobei ein Subjekt mit dem Attribut *User* nur beschränkte Rechte besitzt. Dabei können *Administrator* und *User* als abstrakte Attribute, Rollen oder Sicherheitsklassen modelliert werden.

Im Allgemeinen existieren zwei grundlegende Strategien, wie Attribute in einem Autorisierungsprozess eingesetzt werden. Zum einen können Zugriffsregeln nur anhand von Attributen definiert werden. Dieser Ansatz erlaubt eine flexible Verwaltung einer potentiell unendlichen Menge von Subjekten und Objekten. Ein typisches Beispiel dafür stellen die *rollenbasierten Zugriffskontrollmodelle* 3.3.2.2 dar. Dabei erlangen Subjekte erst dann die Zugriffsrechte, wenn sie von einem Administrator einer bestimmten Rolle zugewiesen werden.

Zum anderen können die Zugriffsrechte der Subjekte durch die Regeln an Attributen eingeschränkt werden. Typisches Beispiel dafür ist das *Bell-LaPadula-Modell* 4.4, wo die Zugriffsregeln auf Basis von Sicherheitsklassen die Zugriffs-

rechte der Subjekte dominieren.



(a) Implizites Schema (b) Geordnetes Schema (c) Komplexes Schema

Abbildung 3.8: Beispiele für attributbehaftete Sicherheitsschemas

In der Abbildung 3.8 sind mögliche Sicherheitsschemas der attributbasierten Modelle abgebildet. Die Sicherheitsschemas beschrieben dabei, in welcher Relation die einzelnen Attribute zu einander stehen. Das Sicherheitsschema aus dem Beispiel *(a)* ist implizit oder einfach. Implizit werden in dieser Arbeit diejenigen Schemas bezeichnet, in denen keine Attributbeziehungen festgelegt sind. Die Nichtexistenz solcher Beziehungen bedeutet, dass Attribute sich weder gegenseitig beeinflussen noch ausschließen.

Das geordnete Schema aus dem Beispiel *(b)* beschreibt hierarchische Beziehungen von Attributen. Ein geordnetes Sicherheitsschema kann zum Beispiel eingesetzt werden, um die Zugriffshierarchien innerhalb von militärischen oder kommerziellen Organisationen zu modellieren.

Das komplexe Sicherheitsschema *(c)* ist frei konfigurierbar. Zum Beispiel können einzelne Beziehungen zwischen Attributen anwendungsspezifisch festgelegt werden. In dem *Schematic Protection Model* 4.3 und dem *Attribute Based Access Matrix Model* 3.3.2 wird dieser Ansatz verwendet.

Die Sicherheitsschemas können auf unterschiedliche Weise modelliert werden. Dazu werden in der Regel Prädikate (Abschnitte 4.2 und 4.3), abstrakte Relationen (Abschnitt 4.3), algebraische Strukturen (zum Beispiel Verbände in 3.3.2.1) oder Graphen eingesetzt.

3.3.2.1 Verband-basierte Zugriffskontrollmodelle

Die verband-basierten Modelle sind eine Gruppe von Zugriffskontrollmodellen, die mit dem Ziel entwickelt wurden, Informationsflüsse im Rahmen der Zugriffskontrolle zu kontrollieren. Hierzu werden die Konzepte verwendet, die auf den theoretischen Grundlagen der *Informationsflussmodelle* 3.4 beruhen.

Die Modellklasse charakterisiert ein stark ausgeprägtes Sicherheitsschema, welches auf den Attributen in Form von Sicherheitsklassen bzw. Kategorien basiert[3]. Das Sicherheitsschema der verband-basierten Modelle beschreibt eine geordnete Menge von Sicherheitsklassen.

Jedem Subjekt und jedem Objekt werden Sicherheitsklassen zugeordnet. Die Sicherheitsklassen können ja nach Anwendung unterschiedlich definiert sein. Beispielsweise werden die Sicherheitsstufen $unklassifiziert$, $vertraulich$ und $geheim$ durch Sicherheitsklassen repräsentiert. Sie stehen in einer fest definierten Ordnungsrelation[4] zueinander. Dabei bedeutet eine gemäß der Ordnungsrelation größere Sicherheitsklasse ein höheres Informationsgehalt oder eine höhere Informationssensibilität. Die Ordnungsrelation wird in dieser Modellklasse durch einen Verband beschrieben.

Ein Verband ist eine partielle Ordnungsrelation $(SC, \leq)$ mit SC als Menge der Sicherheitsklassen für die gilt, dass für zwei beliebige Elemente immer eine eindeutige Oberschranke (Supremum) und eine eindeutige Unterschranke (Infimum) existieren muss. Sei $(SC, \leq)$ ein Verband, dann gilt für alle $x, y \in SC$:

$$x \leq supremum(x, y) \wedge y \leq supremum(x, y) \text{ und}$$
$$infimum(x, y) \leq x \wedge infimum(x, y) \leq y.$$

Diese Eigenschaften eines Verbands sind insofern wichtig, da in einem System oft Subjekte mit der höchsten Sicherheitsklasse wie zum Beispiel Administratoren und Objekte mit der niedrigsten Sicherheitsklasse wie zum Beispiel öffentliche Dateien existieren.

Anhand eines Verbands $(SC, \leq)$ werden also die Zusammenhänge zwischen Sicherheitsklassen festgelegt, um die Zulässigkeit von ausgeführten Aktionen und damit zulässigen Informationsflüssen zu überprüfen. Die genau Interpretation bzw. Anwendung der durch ein Verband festgelegten Beziehungen ist modellspezifisch.

Beispiel 3.3.2:
Gegeben sei Menge der Sicherheitsklassen $S = \{000, 001, \ldots, 111\}$, wo die Klas-

[3] Vgl. [San93]. Sandhu betrachtet in [San93] komplette Modelle. Hier ist das geordnete Sicherheitsschema das charakterisierende Merkmal der Verbandsmodelle.

[4] Eine Ordnungsrelation ist eine Relation, die reflexiv, antisymmetrisch und transitiv ist.

sen Bit-Vektoren darstellen. Nun sollen auf diesen Klassen mit einem Verband die möglichen Informationsflüsse modelliert werden.

Die Informationsflüsse sollen von einer niedrigeren Sicherheitsklasse zu einer höheren Sicherheitsklasse fließen, aber nicht andersherum:

$$supremum(A, B) = B \Rightarrow A \rightarrow B,$$

wobei A und B Sicherheitsklassen sind. $A \rightarrow B$ bedeutet, dass die Information von A nach B fließen darf.

In diesem Fall entspricht das Infimum dem bitweisen Operator „oder“ und das Supremum dem Operator „und“:

$$supremum(A, B) = or(A, B), \text{ z.B.: } 100 \ \ or \ \ 011 = 111;$$
$$infimum(A, B) = and(A, B), \text{ z.B.: } 100 \ \ and \ \ 011 = 000,$$

In der Abbildung 3.9 sind alle möglichen Flüssen innerhalb des Verbands modelliert.

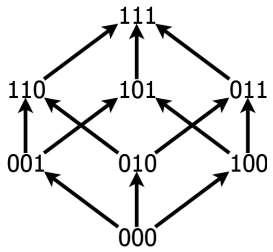


Abbildung 3.9: Beispiel für ein Verband. (Angelehnt an [DD77])

3.3.2.2 Rollenbasierte Zugriffskontrollmodelle

Die Familie der *rollenbasierten Zugriffskontrollmodelle* ist ein Spezialfall der *attributbasierten Zugriffskontrollmodelle*, bei dem ein Rollenkonzept eingeführt wird. Dabei besitzen Subjekte die Zugriffsrechte auf Objekten nicht direkt, sondern erst dann, wenn sie eine bestimmte Rolle einnehmen. Die Rollen sind Sammlungen von Zugriffsrechte, die aufgabenorientiert definiert werden. Das bedeutet, dass eine Rolle in der Regel für eine spezifische Aufgabe zugeschnitten ist und nur die Zugriffsrechte zugeteilt bekommt, die nur für diese Aufgabe notwendig sind.

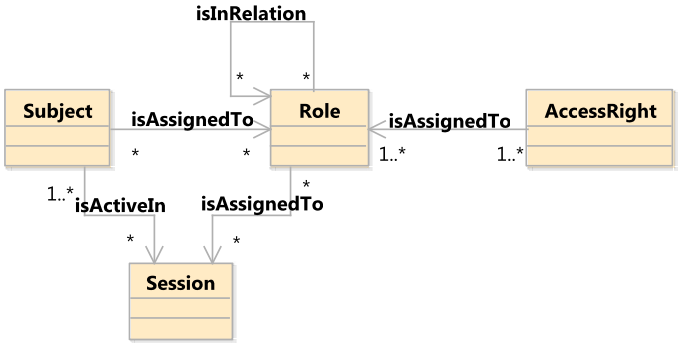


Abbildung 3.10: Grundstruktur von rollenbasierten Modellen (Angelehnt an [FK07])

Aus konzeptioneller Sicht kann die Struktur der *rollenbasierten Modelle* mit dem Diagramm 3.10 beschrieben werden.

Für die Ausführung von bestimmten Aufgaben werden Subjekte den Rollen zugeordnet. Den Rollen werden Zugriffsrechte zugeteilt, die für die Fertigstellung einer Aufgabe notwendig sind. Des Weiteren wird in den meisten *rollenbasierten Zugriffskontrollmodellen* ein Sitzungskonzept für eine dynamische Rollenaktivierung verwendet. Um eine bestimmte Aufgabe zu erledigen, müssen Subjekte eine Sitzung starten und diejenigen Rollen aktivieren, die für Umsetzung dieser Aufgabe notwendig sind. Dabei darf ein Subjekt nur die Rollen aktivieren, die für es bereits beispielsweise von einem Systemadministrator statisch autorisiert wurden. Somit können in der Regel drei wichtige Zugriffskontrollrelationen für ein *attributbasiertes Zugriffskontrollmodell* definiert werden:

- *Subjekt-Rollen-Relation*:
  beschreibt welche Rollen welchen Subjekten statisch zugeordnet sind;
- *Rolle-Zugriffsrechte*:
  beschreibt welche Zugriffsrechte zu einer Rolle gehören;
- *Sitzungsrelation*:
  beschreibt welche Subjekte welche Rollen in einer Sitzung aktiviert haben.

Das Sicherheitsschema der Modellklasse basiert auf Rollen. Das einfachste Schema gibt dabei keine Rollenbeziehungen vor. Das bedeutet, dass keine explizite Relationen zwischen Rollen existieren. Die komplexeren Schemas beschäfti-

gen sich mit der gemeinsamen Nutzung der Zugriffsrechte oder der Vermeidung von Interessenkonflikten [FK07]. Dabei werden so genannte Rollenhierarchien, Rollengraphen oder Konfliktrelationen definiert, anhand derer die globalen Zugriffsentscheidungen festgelegt werden können.

Bei einer hierarchischen Rollenstruktur erben Rollen die Zugriffsrechte der höher eingeordneten Rollen, was den Rollenverwaltungsaufwand erleichtert. Diese Struktur kann zum Beispiel wie im Falle von *verband-basierten Zugriffskontrollmodellen* mit einem Verband modelliert werden.

Mit einer Konfliktrelation wird festgelegt, dass kein Subjekt bei der Ausführung einer bestimmten Aufgabe sich ausschließende Rollen aktiviert haben darf. Zum Beispiel ist die gleichzeitige Mitgliedschaft in den Rollen „Prüfender“ und „Prüfling“ ausgeschlossen, da ein Subjekt für die Aufgabe „Prüfung“ die Zugriffsrechte beider Rollen nicht besitzen darf.

Die rollenbasierten Zugriffskontrollmodelle wurden für den Zugriffsschutz innerhalb von Unternehmen und Organisationen entwickelt. Ihre Struktur orientiert sich an der organisationsspezifische Einsatzumgebung und ist deshalb stark anwendungsbezogen ausgelegt.

3.4 Informationsflussmodelle

Im Gegensatz zu *Zugriffskontrollmodellen* steht bei *Informationsflussmodellen* die Modellierung von Informationsflüssen im Vordergrund. Die Struktur solcher Modelle ist deshalb auf die Beschreibung und Überwachung von möglicher Veränderung, Verbreitung und Flusskontrolle von Informationen in einem IT-System ausgelegt [Eck09, McL90].

Nichtsdestotrotz ist die Modellierung von Informationsflüssen mit der Zugriffskontrolle stark verbunden, da durch Subjekte ausgeführte Zugriffe ebenfalls Informationsflüsse verursachen. Der Zusammenhang zwischen Zugriffen auf Ressourcen und Informationsflüssen wurde in der Klasse der *verband-basierten Zugriffskontrollmodellen* in Abschnitt 3.3.2.1 betrachtet. In dieser Arbeit werden *Informationsflussmodelle* als Klasse der Modelle definiert, die auf der reinen Beschreibung von Informationsflüssen spezialisiert sind, weshalb die Zugriffe von Subjekten nur als eine mögliche Ursache von Informationsmanipulation gesehen werden. Solche Modelle abstrahieren in der Regel die Quellen möglicher Flüsse.

In der Abbildung 3.11 sind abstrakte Elemente und Beziehungen der *Informa-*

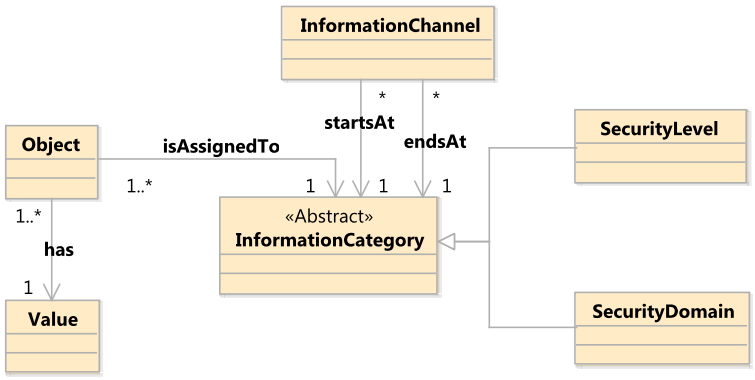


Abbildung 3.11: Klassenmerkmale der Informationsflussmodelle

tionsflussmodelle abgebildet. Die Grundelemente sind Objekte, Objektwerte, Informationskategorien und Informationskanäle

Objekte (`Object`) sind schon aus den Zugriffskontrollmodellen bekannt. Im Gegensatz zu Zugriffskontrolle ändert sich jedoch die konzeptuelle Sichtweise auf Objekte. Informationstechnisch stellen Objekte ein abstraktes Konzept dar, wie bestimmte Informationen repräsentiert werden. In diesem Fall sind Objekte Informationsbehälter, deren Inhalt sich ändern kann. Dies können zum Beispiel Ressourcen eines Systems wie Dateien, Verzeichnisse und Datenbanktabellen, aber auch Prozesse, Programme und deren Variablen und Methoden sein.

Objekte können Werte (`Value`) besitzen. Mit einem Wert wird der Inhalt eines Objekts modelliert. Dabei darf ein Objekt in der Regel nur einen Wert haben. Beispiele für Werte sind Variablenwerte oder auch Bezeichner, die den Informationsgehalt eines Objektes repräsentieren. Fließen Informationen von einem Objekt zu einem anderen, können sich ihre Werte ändern.

Mit den Informationskategorien (`InformationCategory`) wird die Information in einem System entsprechend einer anwendungsspezifischen Relevanz bzw. Wichtigkeit klassifiziert. Dementsprechend werden Objekten eines Systems Sicherheitskategorien zugeordnet. Die Sicherheitskategorien können zum Beispiel wie im Falle von *Bell-LaPadula* in Abschnitt 4.4 Sicherheitsklassen oder Sicherheitsbereiche sein, wobei sie dieselbe semantische Bedeutung der Informationskategorisierung haben.

Die Informationen zwischen Objekten eines Systems werden durch die Informationskanäle (`InformationChannel`) übertragen. Als Informationskanäle

werden mögliche Informationsflüsse bezeichnet. Durch die Abstraktion der Informationsflussmodelle werden die Informationskanäle in der Regel nicht zwischen einzelnen Objekten modelliert, sondern zwischen den Sicherheitskategorien. Durch diese Informationskanäle wird definiert, dass Informationsflüsse von allen Objekten der Start-Sicherheitsklasse zu allen Objekten der Ziel-Sicherheitsklasse gestattet sind.

Im Allgemeinen existieren offene und verdeckte Kanäle [Den82]. Offene Kanäle sind Informationskanäle, durch welche Informationsflüsse vorgesehen sind. Wenn Information durch einen Kanal fließen kann, der dafür nicht vorgesehen war, wird dieser Kanal verdeckt genannt. Unterschiedliche Informationskanal- und Informationsflussarten werden in dem Beispiel 3.4.1 betrachtet.

```
1  Integer n;                          L
2  Integer sum;                        H
3  Boolean flag;                       L
4  File f;                             L
5
6  n   = 0;                            0 → n (L → L)
7  sum = 0;                            0 → sum (L → H)
8  flag = getFlag(f);                  f → flag (L → L)
9
10 if (flag) {
11   n = n + 1;                        n ⊕ 1 → n (L → L)
12   sum = sum + n;                    sum ⊕ n → sum (H → H)
13 } else {                            flag → n ⊗ sum (L → L)
14   throw new
15   Exception("invalid value");
16 }
```

(a) Programm (b) Flussauswertung

Abbildung 3.12: Informationsflüsse in einem Programm (Angelehnt an [DD77])

Beispiel 3.4.1:
Das Beispiel soll die Definition und Auswertung von Informationsflüssen in einem System zeigen. In der Abbildung 3.12 *(a)* ist ein Beispielsystem als ein Java-Programm dargestellt. Die Abbildung 3.12 *(b)* zeigt die festgelegten Sicherheitsklassen der Variablen und Informationsflüsse in den einzelnen Anweisungen des Programms.

In den Zeilen 1 bis 4 werden zunächst den Objekten des System `n`, `sum`, `flag` und `f` eine der Sicherheitsklasse `L` oder `H` zugeordnet. Außerdem sind Konstanten ebenfalls Objekte, die stets der Sicherheitsklasse `L` zugeordnet sind. Informationen können von der Klasse `L` zu der höheren Klasse `H` und innerhalb dieser Klassen fließen. Mit anderen Worten werden hier drei gerichtete Informationskanäle definiert: $\texttt{L} \rightarrow \texttt{L}$, $\texttt{L} \rightarrow \texttt{H}$ und $\texttt{H} \rightarrow \texttt{H}$. Die Klassen und deren Beziehungen werden hier durch ein Verband modelliert, dessen Eigenschaften bereits in Abschnitt 3.3.2.1 erläutert wurden.

Bei den initialen Wertzuweisungen in den Zeilen 6 bis 8 fließen die Informationen von der rechten Seite einer Anweisung zu der linken. Dabei fließt der Wert einer Konstanten `0` in die Variablen `n` und `sum`. Da die Konstante und die Variable `n` die Sicherheitsklasse `L` besitzen, sind diese Informationsflüsse erlaubt. Des Weiteren wird ein bool'scher Wert aus der Datei `f` ausgelesen und der Variable `flag` zugewiesen (Zeile 8). Der Fluss von `L` nach `L` ist ebenfalls erlaubt.

In den Zeilen 10-13 werden die Informationsflüsse innerhalb einer `if`-Anweisung betrachtet. Zunächst fließen die Werte der Variablen `n` und der Konstanten `1` wieder in `n` (Zeile 10). Dabei wird mit dem aus dem Beispiel 3.3.2 bekannten Supremum-Operator $\oplus$ die kleinste oberste Schranke von `n` und `1` bestimmt. Die maximale Sicherheitsklasse, von der die Informationen fließen können, ist dabei die Klasse `L`. Der durch diese Anweisung verursachte Fluss ist daher legitim. Auf die gleiche Weise wird der Informationsfluss in der Zeile 12 überprüft.

Der Informationsfluss in der Zeile 13 wird indirekter bzw. impliziter Fluss genannt. Dieser wird nicht direkt durch eine Operation sondern indirekt durch die Variable `flag` im Kopf der `if`-Anweisung verursacht. Je nachdem welchen Wert die Variable `flag` besitzt, ändern sich die Werte von `n` und `sum` oder nicht. Dadurch fließt die Information indirekt in die Variablen `n` und `sum` ein. Dabei wird mit dem Infimum-Operator $\otimes$ die kleinste Sicherheitsklasse der beiden Variablen `n` und `sum` bestimmt. Diese darf nicht kleiner als die Sicherheitsklasse von `flag` sein. In diesem Fall ist der Fluss erlaubt, da $\texttt{L} \rightarrow \texttt{L}$ gilt.

Die bisher betrachteten Flüsse haben in den offenen Informationskanälen stattgefunden. Diese wurden im System erkannt und modelliert. Tatsächlich existieren in dem gegebenen System noch weitere Informationsflüsse, die nicht vorgesehen sind. In dem `else`-Teil der `if`-Anweisung wird eine Fehlermeldung geworfen. Diese Fehlermeldung kann beispielsweise von anderen Programmen ausgewertet werden, was zu nicht berücksichtigten Informationsflüssen führt. Diese

Art von Kanälen werden verdeckte Informationskanäle genannt.

Analog zur Zugriffskontrolle werden die für die Informationsflussmodelle typischen Beschreibungselemente definiert, welche in der Abbildung 3.13 dargestellt sind.

Das erste Beschreibungselement ist die Flusskontrollrelationen (`FlowControlRelation`). Eine Flusskontrollrelation beschreibt im Allgemeinen die Beziehung der Objekte zu den Informationen in einem System. Zum einen soll damit beschrieben werden, welche Objekte welchen Informationskategorien zugeordnet sind. Zum anderen ist die Zuordnung von Objekten zu ihren Werten ebenfalls eine Flusskontrollrelation.

In welcher Relation Informationskategorien zueinander stehen, wird mit einem Sicherheitsschema (`SecurityScheme`) festgelegt. Im Gegensatz zu den sehr abstrakten Sicherheitsschemata der Zugriffskontrollmodelle kann einem Schema dieser Modellklasse eine konkrete semantische Bedeutung zugeordnet werden. Ein Schema der Informationsflussmodelle soll mögliche Flüsse in einem System abbilden. Dabei wird bestimmt, durch welche Informationskanäle Informationen von einer Sicherheitsklasse zu einer anderen fließen dürfen. Ein Sicherheitsschema kann durch eine abstrakte Relation oder durch eine mathematische Struktur wie einem Verband modelliert werden. Die Modellierung von Flüssen mit einem Verband wurde bereits in dem Abschnitt 3.3.2.1 und in den Beispielen 3.3.2 und 3.4.1 behandelt. Ein weiteres Beispiel der Modellierung von Flüssen ist die strikte Beschreibung von Informationsbereichen, in die die Information nie einfließen darf. Mit anderen Worten werden dabei Informationsbereiche definiert, die sich gegenseitig nicht beeinflussen dürfen. Dieses Konzept der sogenannten Interference-Relationen wird in [GM82] behandelt. Schließlich können Informationsflüsse zwischen Sicherheitskategorien mithilfe eines Graphen beschrieben werden. Die Knoten und Kanten stellen dabei die Sicherheitskategorien und entsprechend die Informationskanäle dar.

Ein weiteres Beschreibungselement sind die Flussoperationen (`FlowOperation`). Jede Operation bzw. Anweisung, die sowohl implizite auch explizite Flüsse verursacht, wird hier Flussoperation genannt. Die Informationsflüsse finden statt, wenn die Objektwerte zum Beispiel durch eine Kopieren-Operation geändert werden. Außerdem kann eine Flussoperation die Sicherheitsklassen von Objekten ändern. Dadurch modifizieren Flussoperationen die Flusskontrollrela-

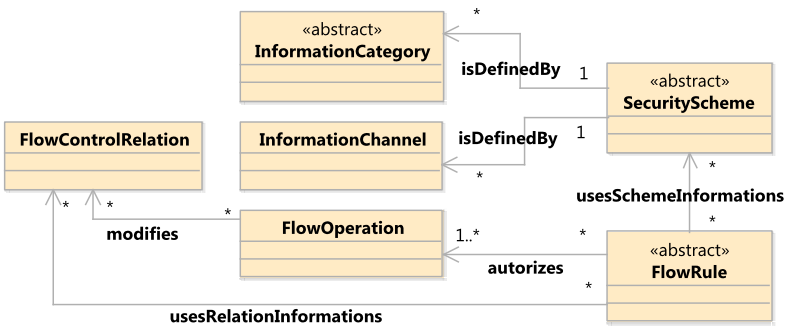


Abbildung 3.13: Struktur der Informationsflussmodelle

tionen.

Mit den Flussregeln (`FlowRule`) wird bestimmt, ob eine Flussoperation ausgeführt werden darf. Flussregeln gehören zu den Merkmalen eines Sicherheitsmodells und bilden damit die funktionalen Sicherheitsanforderungen einer Sicherheitspolitik ab. Wie im Falle von Zugriffsregeln der *Zugriffskontrollmodelle* aus Abschnitt 3.3 repräsentieren Flussregeln die in einer zugrundeliegenden Politik festgelegten Regeln des erlaubten Informationsflusses. Eine Flussregel autorisiert eine Flussoperation, indem anhand der Flusskontrollrelationen und eines Sicherheitsschemas eine Entscheidung gefällt wird, ob diejenige Operation zulässig ist.

Der Zusammenhang zwischen den Merkmalen der Informationsflussmodelle und des Kernmodells wird analog zu den Zugriffskontrollmodellen hergestellt. Die Flussregeln beschreiben die Zustandsübergänge eines Systems. Die dynamischen Flusskontrollrelationen bilden dabei seinen Zustand. Die Flussoperationen verursachen einen Zustandswechsel, indem sie Informationsflüsse initiieren.

3.5 Transaktionsmodelle

Das Anwendungsgebiet der bisher betrachteten Modellklassen beschränkt sich in der Regel auf geschlossene Systeme mit einer zentralen Kontrollinstanz. In dieser Hinsicht beschreiben die *Transaktionsmodelle* die Zusammenhänge und Abläufe außerhalb solcher Systeme. Der Hauptschwerpunkt der *Transaktionsmodelle* liegt daher in der Modellierung von offenen, verteilten und nicht vertrauenswürdigen

Umgebungen wie zum Beispiel dem Internet [GP, Gri08, Fox98].

Wie der Klassenname bereits verrät, steht die Sicherheit von Transaktionen im Mittelpunkt der *Transaktionsmodelle*. Die Grundelemente sind dabei die abstrakten Konzepte der Parteien und Güter, welche in der Abbildung 3.14 dargestellt sind.

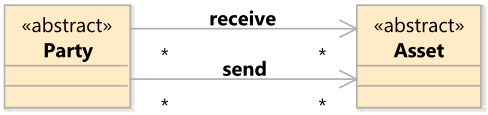


Abbildung 3.14: Klassenmerkmale der Transaktionsmodelle

Parteien (`Party`) sind Kommunikationspartner, die individuelle Interessen haben. Dies können beispielsweise Programme wie Client und Server oder Personen wie Käufer und Verkäufer sein. Parteien kommunizieren, indem sie sogenannte Güter senden und empfangen.

Güter (`Asset`) sind im Allgemeinen die Gegenstände der Kommunikation, die für Parteien einen bestimmten Wert darstellen. Typisches Beisiel für Güter sind Nachrichten. Die Nachrichten enthalten evtl. sensible Informationen, die als Gut angesehen werden können. Weitere Beispiele für Güter sind digitale Rechte, die eine Partei bedingt oder beschränkt an andere Parteien übertragen möchte.

Das Ziel der *Transaktionsmodelle* ist, die Kommunikation zwischen den Parteien so zu regeln, dass im Falle eines unerwünschten oder falschen Verhaltens seitens einer oder mehrerer Kommunikationspartnern der dadurch entstandene Schaden minimal oder begrenzt ist. Dadurch sollen die beteiligten Parteien und deren Güter geschützt werden. Die gemeinsame abstrakte Struktur der *Transaktionsmodelle* ist in der Abbildung 3.15 abgebildet.

Jede Partei besitzt eine Menge von Gütern (`AssetSet`), die sie an andere Kommunikationsteilnehmer übertragen möchte. Damit eine Partei ihre Güter transferieren kann, muss sie zunächst eine Übertragung (`Transaction`) initiieren, indem sie eine Teilmenge ihrer Güter an ihre Kommunikationspartner sendet. Die Kommunikationspartner können ihrerseits diese Güter empfangen. Ob die Übertragung stattfindet bzw. wie diese stattfinden soll, wird mit den Transaktionsregeln (`TransactionRule`) festgelegt. Mit anderen Worten beschreiben Transak-

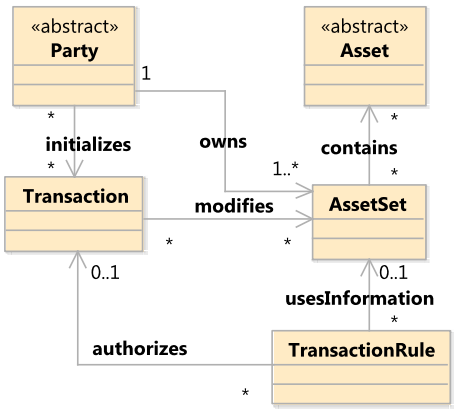


Abbildung 3.15: Struktur der Transaktionsmodelle

tionsregeln, wie Güter zwischen Kommunikationsteilnehmer getauscht werden können.

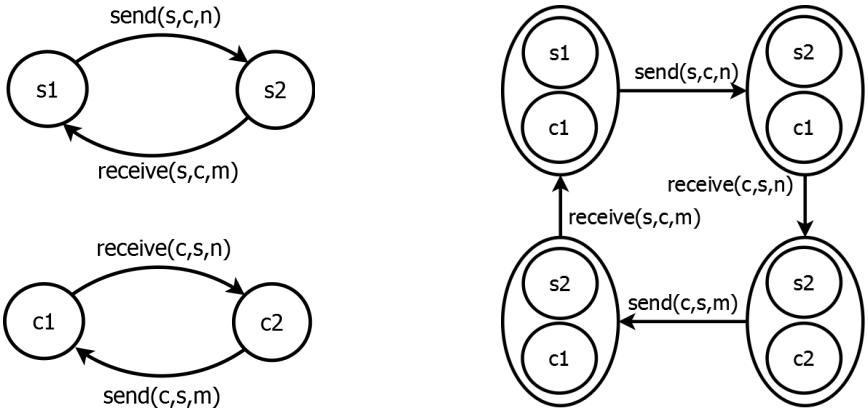
Da mit den *Transaktionsmodellen* die Dynamik der offenen und dezentralen Systeme beschrieben werden soll, rückt nun die Modellierung vom Verhalten der einzelnen Kommunikationsteilnehmer in den Vordergrund, die im Allgemeinen unabhängig voneinander und unbeschränkt miteinander interagieren können. Die Zustandsübergangssysteme der bisher beschriebenen Modelle sind für die Simulation eines zentralen System gedacht und deshalb für offene Systeme wenig geeignet. Infolgedessen wird bei den *Transaktionsmodellen* ein Konzept eingesetzt, bei dem die Dynamik jedes einzelnen Kommunikationsteilnehmer mit einem eigenen Zustandsübergangssystem modelliert wird. Kommunizieren die Parteien miteinander, dann interagieren ihre Automaten.

Die einzelnen Automaten aller Parteien können mit einem einzigen Automaten simuliert werden, indem jeder seiner Zustände die jeweiligen Sub-Zuständen aller Partei-spezifischen Automaten enthält. Dieser Sachverhalt wird in dem Beispiel 3.5.1 erläutert. Basierend auf den Informationen über die gesendeten und erhaltenen Güter der in dem Kommunikationsprozess beteiligten Parteien (Kommunikationsteilnehmer bzw. Kommunikationspartner) beschreiben die Transaktionsregel die Transitionen bzw. die Zustandsübergänge, die in einem offenen System stattfinden, indem sie die Transaktionen der Parteien autorisieren. Die Transaktionen ihrerseits verändern die Menge der aktuellen Güter und somit einen Zustand einer Partei (Abbildung 3.15).

Beispiel 3.5.1:
Sei ein dezentrales System mit zwei Parteien s und c gegeben, die gegenseitig Güter in Form von Nachrichten vom Typ n und m schicken und empfangen können. Für jede der Partei wird ihr Verhalten mit einem eigenen Automaten modelliert, welche in der Abbildung 3.16 *(a)* dargestellt sind. Die Zustände der Parteien sind entsprechend $s1$, $s2$ und $c1$, $c2$. Die Nachrichten werden mit den Transaktionen $send$ und $receive$ übertragen. Mit $send(s, c, n)$ wird eine Nachricht von Typ n von s zu c transportiert. Analog kann s von c mit $receive(s, c, m)$ eine Nachricht von Typ m empfangen.

Die Transaktionsregeln legen fest, dass die Partei s beim Senden einer Nachricht vom Typ n von der Partei c in den Zustand $s2$ wechselt. Dabei empfängt c diese Nachricht und wechselt ebenfalls ihren Zustand zu $c2$. Wenn c eine Nachricht vom Typ m an s sendet, findet ein analoger Ablauf statt.

Nun soll dieses System mit einem einzigen Automaten simuliert werden. Hierfür existieren mehrere Möglichkeiten. Eine davon ist in der Abbildung 3.16 *(b)* abgebildet.



(a) Interagierende Automaten (b) Simulation

Abbildung 3.16: Simulation zweier parallel interagierender Automaten

Dabei werden die Systemzustände durch die Sub-Zustände der einzelnen Parteien modelliert. Wechselt eine der Parteien ihren Zustand, so ändert sich der Zustand des gesamten Systems.

Kapitel 4

Formale und semiformale Sicherheitsmodelle

In diesem Kapitel werden die bereits veröffentlichen Sicherheitsmodelle beschrieben. Für die Beschreibung dieser Modelle wird das Spezifizierungsschema von Grimm in [Gri08] und die in dieser Arbeit ausgearbeiteten Beschreibungselemente des Klassifizierungsschemas verwendet.

Da die grundlegenden Konzepte der Sicherheitsmodelle bereits in den vorangegangenen Kapiteln beschrieben wurden, erlauben diese Erkenntnisse, bei der Modellbeschreibung den Schwerpunkt auf die Umsetzung der zugrundeliegenden Konzepte zu legen.

Im Mittelpunkt der Modellbeschreibung steht das Zustandsübergangssystem (Q, Σ, δ, q_0), dessen Zustände und Zustandsübergänge als grundlegende Beschreibungselemente verwendet werden. Folglich werden die *Sicherheitsmerkmale*[1] und *Sicherheitseigenschaften* der Sicherheitsmodelle auf Basis des Automaten beschrieben.

Des Weiteren wird das Beschreibungselement „*Grundmengen*“ definiert, mit dem die Elementarmengen der Sicherheitsmodelle wie zum Beispiel die Mengen der Subjekte und Objekte repräsentiert werden.

Bei den Zugriffskontrollmodellen soll unter anderem durch *Zugriffskontrollrelationen* die Verteilung der Privilegien in einem System beschrieben werden. In welcher Beziehung die Attribute eines Systems zueinander stehen, wird dabei

[1] Im Folgenden werden *Sicherheitsmerkmale* Modellmerkmale genannt.

mit einem *Sicherheitsschema* beschrieben.

In den Informationsflussmodellen beschreiben die *Informationsflussrelationen*, wie die Informationen in einem System verteilt werden können. Dazu zählen die Objekt − Sicherheitskategorie- und Objekt − Objektwert- Zuordnungen. In welcher Beziehung die Sicherheitskategorien zueinander stehen und welche Flüsse dazwischen fließen, wird mit den *geordneten Sicherheitsschemas* beschrieben.

Typisch für die Transaktionsmodelle ist die Beschreibung der Mengen der kommunizierenden Parteien und der Güter die zwischen den Parteien übertragen wurden.

Da es sich bei den ausgewählten Sicherheitsmodellen um *generische Sicherheitsmodelle* handelt, wird mithilfe eines Beschreibungspunkts „*Instantiierung*“, die variablen Komponente der Modelle angegeben und ihre Instantiierung beschrieben. Die Instantiierung der besonders komplexen Modelle, wird mit Beispielen untermauert.

4.1 Harrison-Ruzzo-Ullman-Modell

Das *Harrison-Ruzzo-Ullman-Modell* (HRU-Modell) aus dem Jahr 1976 ist eines der ältesten, vollständig formalisierten Zugriffskontrollmodelle [HRU76]. Das Modell stellt eine sehr abstrakte Beschreibung eines Computersystems dar und bildet eine theoretische Basis für eine ganze Reihe von Sicherheitsmodellen.

4.1.1 Sicherheitsziel

Das Sicherheitsziel des *HRU-Modells* ist die Wahrung der Integrität von Zugriffsrechten. Dabei soll verhindert werden, dass unautorisierte Subjekte Zugriffsrechte erhalten.

4.1.2 Modellmerkmale

Das *HRU-Modell* kann bezüglich seiner Merkmale als ein minimalistisches Sicherheitsmodell aus der Klasse der Zugriffskontrollmodelle bezeichnet werden. Es besitzt nur eine Zugriffskontrollrelation in Form einer Zugriffsmatrix und ein implizites Sicherheitsschema. Seine Notation stimmt mit der Notation des zugrundeliegenden Zustandsübergangssystems überein:

$$(Q, \Sigma, \delta, q_0)$$

Grundmengen Die Mengen der Subjekte und Objekte werden durch die potentiell unendlichen Mengen S und O modelliert, wobei gilt:

$$S \subseteq O,$$

da Subjekte auf andere Subjekte zugreifen können.

Die endliche Menge R stellt die Zugriffsrechte dar, wobei keine explizite Unterscheidung zwischen neutralen Zugriffsrechten und Kontrollrechten gemacht wird.

Zugriffskontrollrelationen Wie bereits erwähnt, besitzt das *HRU-Modell* eine Zugriffskontrollrelation, welche durch eine Matrix modelliert wird. Die Matrix $M : S \times O \rightarrow 2^R$ beschriebt die Rechteverteilung im System zu einem bestimmten Zeitpunkt.

Die Zeilen der Matrix bilden die Subjekte und die Spalten bilden die Objekte. Eine Einfache Zugriffsmatrix ist in der Tabelle 4.1 dargestellt.

	o_1	o_2
s_1	$\{read, write\}$	
s_2		$\{write\}$

Tabelle 4.1: Eine einfache Zugriffsmatrix

Zustände Ein Zustand wird durch die Zugriffsmatrix und die Menge der aktuellen Subjekte und Objekte modelliert. Sei q ein beliebiger Zustand, Q der Zustandsraum und M die Zugriffsmatrix des Modelles, dann gilt:

$$q = (P_o, M) \text{ und } Q = 2^O \times (S \times O \rightarrow 2^R)\ ,$$

wo $P_o \in 2^O$ ist.

Sicherheitsschema Das Modell besitzt ein implizites Sicherheitsschema, welches aus der Identitätsprüfung zwischen den Subjekten und Objekten, die eine Zugriffsaktion ausführen wollen, sowie den entsprechenden Entitäten der Matrix besteht. Sollte die Matrix einem Subjekt den Zugriff auf andere Identitäten

gestatten, so gibt es keine Möglichkeit diese Zugriff durch weitere Restriktionen zu beschränken. Somit könnten Subjekte in die Lage versetzt werden, anderen Subjekten Rechte zuzuweisen. Ein solcher Fall wird in der Fachliteratur eine diskrete Zugriffskontrolle genannt.

Zustandsübergänge Die Zustandsübergänge des Modells werden durch Ausführung der sogenannten Kommandos modelliert. Kommandos sind Makrooperationen, deren Ausführung die Zustandswechsel initiiert. Die Definition eines Kommandos besteht aus Zugriffsregeln, die die Ausführung dieses Kommandos autorisieren, und einer Menge von Elementaroperationen, die einen Zustandswechsel verursachen. Bei der Ausführung eines Kommandos werden eine oder mehrere Elementaroperationen sequentiell ausgeführt, was ein System in Zwischenzustände versetzt. Konnten alle Elementaroperationen eines Kommandos erfolgreich autorisiert werden, wechselt ein System sein Zustand. Dieser Zustand beschreibt ein System, nachdem alle Elementaroperationen des Kommandos abgearbeitet wurden. Falls die Ausführung mindestens einer Elementaroperationen fehlgeschlagen ist, bleibt das System in dem Zustand vor der Ausführung der ersten Elementaroperation.

Beispiel 4.1.1:
Die Abbildung 4.1 stellt die Ausführung eines Kommandos *createObjekt* exemplarisch dar, welches aus den Elementaroperation *op1*, *op2*, *op2* und *op4* besteht. Das System ändert seinen Zustand von *s1* zu *s2* nur dann, wenn alle Elementaroperationen erfolgreich ausgeführt werden konnten.

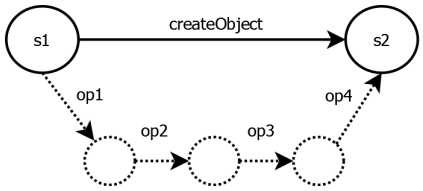


Abbildung 4.1: Zusammenhang zwischen Kommandos und Elementaroperationen

Die Menge der Kommandos eines Systems wird mit der Menge C modelliert. Mit dem Autorisierungsschema wird für jedes Kommando $\alpha \in C$ eine gemeinsa-

me Form der Autorisierungsregeln angegeben:

command $\alpha(x_1, x_2, \ldots, x_k)$
if $r_1 \in [x_{s1}, x_{o1}] \wedge r_2 \in [x_{s2}, x_{o2}] \wedge \cdots \wedge r_m \in [x_{sm}, x_{om}]$
then $op_1; op_2; \ldots; op_n$

Dabei sind $x_1, x_2, \ldots, x_k$ formale Parameter mit $x_i \in S \cup O$, $r_1, r_2, \ldots, r_m$ Zugriffsrechte mit $r \in R$, $[x_{s1}, x_{o1}], [x_{s2}, x_{o2}], \ldots, [x_{sm}, x_{om}]$ Zellen der Matrix M und $op_1, op_2, \ldots, op_n$ Elementaroperationen. Die Zahlen s_i und o_i sind aus dem Intervall $[1, k]$.

Möchte ein Subjekt ein bestimmtes Kommando α ausführen, so müssen in den Matrixzellen der beteiligten Entitäten die dazu benötigten Zugriffsrechte vorhanden sein.

Die Autoren des Modells definieren sechs grundlegende Elementaroperationen, die hier informal beschrieben werden[2]. Sei M eine Zugriffsmatrix und s und o zwei beliebige Objekte aus der Matrix, dann gilt:

- **enter** r **into** $M(s, o)$**:**
  fügt ein Zugriffsrecht r in eine bestimmte Matrixzelle $M(s, o)$ ein, falls es zuvor noch nicht vorhanden war;

- **delete** r **from** $M(s, o)$ **:**
  löscht ein Zugriffsrecht r aus einer bestimmten Matrixzelle $M(s, o)$, falls dieses Recht dort zuvor existiert hat;

- **create subject** s **:**
  erzeugt ein neues Subjekt s und eine neue leere Matrixzeile und eine neue leere Matrixspalte für dieses Subjekt;

- **create object** o **:**
  erzeugt ein neues Objekt o und eine neue leere Matrixspalte für dieses Objekt;

- **destroy subject** s **:**
  löscht ein Subjekt s und eine korrespondierende Matrixzeile;

- **destroy object** o **:**
  löscht ein Objekt o und eine korrespondierende Matrixspalte.

[2]Die formale Definition kann aus [HRU76] entnommen werden.

Beispiel 4.1.2:
Mit einem einem Kommando *deleteObject* versucht ein Subjekt *subject* eine Datei *file* zu löschen:

command $deleteObject(subject, file)$
if $deleteRight \in [subject, file]$
then $destroy\ file$

Falls *subject* für die Löschung von *file* ein entsprechendes Zugriffsrecht *deleteRight* besitzt, wird *file* aus dem System gelöscht und die Zugriffsmatrix angepasst.

Bezogen auf das zugrundeliegende Zustandsübergangssystem stellt die Menge der Kommandos C die Aktionsmenge des Automaten dar:

$$C \subseteq A$$

Die Menge der möglichen Eingaben Σ umfasst die Kommandobezeichner und aktuelle Parameter, die aus Subjekten und Objekten bestehen:

$$\Sigma = C \times 2^O$$

4.1.3 Sicherheitseigenschaften

Mit dem HRU-Modell ist es im Allgemeinen nicht möglich, die Rechteausbreitung zu kontrollieren, da die Subjekte nach eigenem Ermessen ihre Rechte weitergeben könnten. Deshalb haben die Autoren des Modells die *Safety*-Eigenschaft eingeführt [HRU76, Eck09]. Kann ein Subjekt zu keinem Zeitpunkt ein bestimmtes Recht erlangen, welches es nicht bereits besessen hat, dann ist die *Safety*-Eigenschaft erfüllt. Eine Missachtung dieser Einschränkung führt nicht automatisch zu einem unsicheren System. Allerdings kann ohne diese Einschränkung eine Sicherheitsanalyse unentscheidbar werden.

Bezogen auf das Zustandsübergangssystem wird die *Safety*-Eigenschaft wie folgt formuliert: für einen sicheren Zustand q gilt, dass wenn ein Subjekt s ein Recht r besitzt, dann darf es von q ausgehend keine Folge von Aktionen geben,

Grundmengen	
Menge der Objekte	O
Menge der Subjekte	$S \subseteq O$
Menge der Rechte	R
Zugriffskontrollrelationen	
Zugriffsmatrix	$M : S \times O \rightarrow 2^R$
Zustände	
Zustandsraum	$Q = 2^O \times (S \times O) \rightarrow 2^R$
Initialzustand	$q_0 \in Q$
Sicherheitsschema	
implizit	
Zustandsübergänge	
Kommandos	C
Elementaroperationen	$\{enter, create, delete, destroy\}$
Eingabealphabet	$\Sigma = C \times 2^O$
Zugriffsregeln	

Tabelle 4.2: Merkmale von *HRU-Modell*

bei denen ein von s verschiedenes Subjekt das Recht r erwirbt, sofern es r nicht bereits in q besessen hat.

Bei einer erlaubten Transition wird eine Aktion ausgeführt, bei der entweder kein Recht übertragen wird oder der Rechteempfänger dieses Recht bereit zuvor besessen hat.

Die Autoren des HRU-Modell haben bewiesen, dass die *Safty*-Eigenschaft im Allgemeinen zwar unentscheidbar ist, aber durch bestimmte Einschränkungen an dem Modell trotzdem mit einem Algorithmus bewiesen werden kann.

4.1.4 Modellinstantiierung

Das *HRU-Modell* wird instantiiert, indem ein Initialzustand und eine Menge von Kommandos definiert werden, die den Initialzustand verändern. Ein Initialzustand enthält die beim Start des Systems existenten Subjekte und Objekte sowie ihre jeweiligen Zugriffsrechte.

4.2 Attribute Based Access Matrix Model

Das *attributbasierte Zugriffsmatrix-Modell* (ABAM-Modell) ist ein theoretisches Modell aus dem akademischen Bereich. Aus Sicht des *Modell-Engineerings* werden Elemente und Konzepte der attributbasierten Modelle 3.3.2 und Zugriffsmatrixmodelle 3.3.1 verschmolzen. Die Autoren des Modells [ZLN05] versuchen ein flexibles Modell zu entwickeln, welches im Vergleich zu dem HRU-Modell 4.1 eine höhere Ausdruckskraft besitzt, indem sie den Zugriffsschutz mithilfe von Restriktionen auf der Zugriffsmatrix und auf den Attributen realisieren.

4.2.1 Sicherheitsziel

Das Sicherheitsziel des *attribut-basierten Zugriffsmatrix-Modells* stimmt mit den Zielen der meisten Modelle überein, die an HRU-Modell angelehnt sind. Konkret beschäftigt sich das Modell mit der Überwachung der Ausbreitung von Zugriffsrechten im System. Das Sicherheitsziel ist die Verhinderung einer unkontrollierten Rechteausbreitung.

4.2.2 Modellmerkmale

Die Attribute und die damit verbundenen Funktionen und Schemas zeichnen das *attribut-basierten Zugriffsmatrix-Modell* aus, sodass dessen Notation folgendermaßen aussieht:

$$(ATT, R, Q, \Sigma, \delta, q_0).$$

Die Zusammenfassung der Modellmerkmale ist in der Tabelle 4.3 abgebildet.

Grundmengen Subjekte und Objekte werden in dem Modell durch die potentiell unendlichen Mengen S bzw. O modelliert, wobei gilt:

$$S \subseteq O.$$

Die Objekte, die keine Subjekte sind, werden auch reine Objekte genannt. Eine endliche Menge R stellt die Menge der Zugriffsrechte dar.

Subjekte und Objekte besitzen Attribute. Die Menge der Attribute, die eine Entität besitzt, wird durch ein Tupel beschrieben. Jedes Attribut hat einen Wert,

aus einem bestimmten Wertebereich. Dies kann eine natürliche Zahl oder ein komplexer Datentyp wie eine Sicherheitsklasse sein.

Im *attribut-basierten Zugriffsmatrix-Modell* wird festgelegt, dass alle Entitäten die gleiche Anzahl an Attributen haben. Dabei hat jede Entität die für das System maximal mögliche Anzahl an Attributen. Besitzt eine Entität weniger Attribute als andere Entitäten, so haben die nicht benötigten Attribute den Wert $null$.

Beispiel 4.2.1:
Angenommen in einem System existieren ein Subjekt $s \in S$ und ein Objekt $o \in O$. Ihre Attribute sind wie folgt angegeben:

$(a_1 = 5, a_2 = vertraulich)$ und entsprechend $(a_1 = null, a_2 = geheim)$.

Aus dem Beispiel ist zu erkennen, dass in dem gegebenen System einer Entität maximal zwei Attribute zugeordnet werden. Dabei hat o nur ein Attribut a_2. Die Wertebereiche V_1 und V_2 der Attribute a_1 und a_2 bilden natürliche Zahlen $V_1 \subseteq \mathbb{N}$ und Menge der Sicherheitsklassen $V_2 = \{vertraulich, geheim, unklassifiziert\}$.

Die Gesamtheit aller Attribute eines Systems wird durch die Menge ATT modelliert, die eine Menge aller möglichen Attributetupeln darstellt:

$$ATT = ATT_1 \times \ldots \times ATT_n,$$

wo ATT_1 die Menge aller Attributwerte, die an der ersten Stelle in den Attributetupeln vorkommen. ATT_n enthält entsprechend alle Attributwerte, die sich an der letzten Stelle der Attributetupeln befinden.

Somit würde die Menge aller Attributausprägungen ATT aus dem Beispiel 4.2.1 folgendermaßen definiert:

$$ATT \subseteq \{(5, vertraulich), (null, geheim)\}.$$

Zugriffskontrollrelationen Zu den Zugriffskontrollrelationen des Modells gehören die Attributzuordnungsfunktion att und die Zugriffskontrollmatrix M. Die Funktion $att : O \rightarrow A$ bildet jede Entität aus O auf ein Attributetupel aus ATT ab. Mit anderen Worten beschreibt die Funktion att die Menge der Attributwerte, die eine Entität besitzt.

Wie es für Zugriffsmatrix-Modelle typisch ist, wird die Verteilung von Zugriffsrechten in einem System mithilfe einer Zugriffsmatrix $M : S \times O \rightarrow 2^R$

beschrieben. Die Matrix des Modells wird aus dem HRU-Modell übernommen. Die Zeilen der Matrix bilden die Subjekte und die Spalten bilden die Objekte.

Zustände In einem *attributbasierten Zugriffsmatrix-Modell* können zur Laufzeit neue Subjekte und Objekte erzeugt oder existierende gelöscht werden. Dadurch ändern sich die Zeilen und Spalten der zugrundeliegenden Zugriffsmatrix. Außerdem können die Werte von Attributen des Systems geändert werden. Schließlich dürfen Subjekte die Zugriffsrechte an andere Subjekte abgeben oder löschen, sodass die Systemzustände wie folgt modelliert werden können. Sei q ein beliebiger Zustand des Modelles und Q dessen Zustandsraum. Dann ist M eine Zugriffsmatrix, die die aktuelle Rechteverteilung beschreibt. So gilt:

$$q = (P_O, att, M) \text{ und } Q = 2^O \times (O \to A) \times (S \times O \to 2^R) \text{ ,}$$

wo $P_O \in 2^O$ ist.

Sicherheitsschema Das Sicherheitsschema des Modells bilden die einstelligen oder zweistelligen Prädikate. Prädikate sind Regeln über die Attributwerte der Entitäten. Besitzt ein Subjekt ein bestimmtes Attribut, so können mit einem Prädikat ein von der zugrundeliegenden Sicherheitspolitik erlaubter Wert oder Werterahmen für dieses Attribut angegeben werden.

Die einstelligen Prädikate machen Aussagen über ein einzelnes Attribut. Sei age ein Attribut aus der Attributmenge ATT_i mit dem Wertebereich der natürlichen Zahlen. So kann ein Prädikat $isAdult$ wie folgt definiert werden:

$$isAdult : age \geq 18.$$

Die zweistelligen Prädikate verbinden zwei Attribute. Zum Beispiel gehört das Attribut age zu den Subjekten $s1$ und $s2$. Mit dem Prädikat $s1.age \geq s2.age$ wird festgelegt, dass beide Attribute in einer $\geq$-Relation stehen, wenn das Prädikat zu $true$ ausgewertet werden kann.

Die Menge aller definierten Prädikate bildet das Sicherheitsschema des Modells.

Zustandsübergänge Wie im Falle von HRU-Modell werden bei *ABAM-Modell* die Zustandsübergänge durch Kommandos modelliert:

command $\alpha(x_1, x_2, \ldots, x_k)$
$r_1 \in [x_{s1}, x_{o1}] \wedge r_2 \in [x_{s2}, x_{o2}] \wedge \cdots \wedge r_m \in [x_{sm}, x_{om}]$
$p_1 \wedge p_2 \wedge \cdots \wedge p_n$
then $op_1; op_2; \ldots; op_n$

Im Vergleich mit dem *HRU-Modell* wird die Definition von Kommandos um Prädikate $p_1, p_2, \ldots, p_n$ aus dem Sicherheitsschema erweitert. Die Regeln und Beziehungen, die durch Prädikate festgelegt wurden, schränken die Ausführung der Kommandos ein. Damit ein bestimmtes Kommando α ausgeführt werden kann, sollen die Subjekte entsprechende Zugriffsrechte besitzen. Außerdem soll jedes der für dieses Kommando festgelegten Prädikate $true$ liefern.

Die Elementaroperationen *enter* und *delete* können aus dem *HRU-Modell* unverändert übernommen werden. Allerdings müssen die anderen Operationen angepasst werden, da die Subjekte in diesem Modell Attribute besitzen können. Des Weiteren kommt eine neue Operation *update* hinzu:

- **create subject** s **:**
  erzeugt ein neues Subjekt s und seine Attribute in Form eines Wertetupels. Das Wertetupel wird diesem Subjekt durch eine Anpassung der Funktion $att()$ zugewiesen. Schließlich wird eine leere Zeile und eine leere Spalte in der Matrix erzeugt;

- **create object** o **:**
  erzeugt ein neues Objekt o und seine Attribute. Das Wertetupel wird dem Objekt durch eine Anpassung der Funktion $att()$ zugewiesen. Eine leere Matrixspalte wird für dieses Objekt erzeugt;

- **destroy subject** s **:**
  löscht ein Subjekt s, die korrespondierenden Matrixzeile und Matrixspalte und sein Attributetupel;

- **destroy object** o **:**
  löscht ein Objekt o, eine korrespondierende Matrixspalte und sein Attributetupel.

Die Zugriffsoperation $update$ verändert einen Attributwert einer bestimmten Entität. Sei o eine Entität, $att(o) = (v_1 \ldots v_n)$ das dazugehöroge Attributetupel, $o.a_i$ ihr Attribut im Attributetupel an der Position i und v_i der Attributwert, der

geändert werden soll $o.a_i = v_i$, dann wird durch das Update Funktion $att(o)'$ wie folgt aktualisiert:

$$att(o) \rightarrow att(o)',$$
$$att(o)' = (v_1, \ldots, v_i', \ldots, v_n).$$

Der alte Attributwert v_i wird mit dem neuen Wert v_i' ersetzt, wobei die beiden Werte verschieden $v_i \neq v_i'$ und aus dem selben Wertebereich $v_i, v_i' \in V_{a_i}$ sein müssen.

Die Definition der Aktionsmenge A und der Menge der erlaubten Eingaben Σ stimmt mit der Definition aus dem *HRU-Modell* überein:

$$C \subseteq A, \Sigma = C \times 2^O$$

Dabei ist C Menge der Kommandos.

Grundmengen	
Menge der Objekte	O
Menge der Subjekte	$S \subseteq O$
Menge der Attributwerte	$ATT = ATT_1 \times \ldots \times ATT_n$
Menge der Rechte	R
Zugriffskontrollrelationen	
Attribut-Zuordnung	$att : O \rightarrow A$
Zugriffsmatrix	$M : S \times O \rightarrow 2^R$
Zustände	
Zustandsraum	$Q = 2^O \times (O \rightarrow A) \times (S \times O \rightarrow 2^R)$
Initialzustand	$q_0 \in Q$
Sicherheitsschema	
Prädikate	$p_1 \ldots p$
Zustandsübergänge	
Elementaroperationen	$A \subseteq \{enter, delete, create, destroy, update\}$
Menge von Kommandos	C
Eingabealphabet	$\Sigma = C \times 2^O$
Zugriffsregeln	

Tabelle 4.3: Merkmale des *attributbasierten Zugriffsmatrix-Modells*

4.2.3 Sicherheitseigenschaften

Wie beim HRU-Modell betrachten die Autoren des ABAM-Modells die Analysierbarkeit der *Safty*-Eigenschaft. Sie konnten dabei feststellen, dass die Analyse der *Safty*-Eigenschaft entscheidbar ist, wenn die Attributmenge endlich ist und die Beziehungen zwischen Attributen keine Zyklen enthalten. Ein Zyklus ist zum Beispiel gegeben, wenn zwei Attribute *Admin* und *User* mit dem Prädikaten $canGiveRight$ auf folgende Weise verbunden sind:

$$canGiveRight(Admin, User) \text{ und } canGiveRight(User, Admin).$$

In diesem Fall können Subjekte mit diesen Attributen sich gegenseitig Rechte vergeben, wodurch sie immer mächtiger werden können.

4.2.4 Modellinstantiierung

Ähnlich wie beim *HRU-Modell* wird das *ABAM-Modell* instantiiert, indem ein Initialzustand, die Menge der Kommandos und zusätzlich die Menge der Attributwerte festgelegt werden.

4.3 Schematic Protection Model

Schematic Protection Model (SPM-Modell) ist ein theoretisches Modell mit einem hohen Abstraktionsniveau aus der Klasse der *Zugriffskontrollmodelle*, welches im akademischen Bereich eingesetzt wird [San88]. Der Hohe Abstraktionsgrad erlaubt eine flexible Spezifikation der Modellkomponenten und deren Beziehungen, sodass sich mit dem *Schematic Protection Model* eine Vielzahl von Sicherheitspolitiken ausdrücken lässt. Eine Beispielinstanz des Modells wird in 4.3.4 betrachtet.

Mit dem *SPM-Modell* wurden die Konzepte des Send-Receive-Transport Modells von Minsky [Min84] generalisiert. Der Autor des *SPM-Modells* Sandhu versuchte dabei ein Modell zu entwickeln, welches eine hohe Ausdruckskraft bei gleichzeitig akzeptabler Analysierbarkeit besitzt. Die Grundidee der Modellentwicklung ist der Einsatz von einer statischen Typisierung für die Realisierung der globalen Zugriffsrestriktionen.

4.3.1 Sicherheitsziel

Die Definition des *Sicherheitsziels* des *Schematic Protection Models* basiert auf den Zielen der zugrundeliegenden Sicherheitspolitik. Da das Modell für unterschiedliche Szenarien konfiguriert werden kann, gibt Sandhu hiermit kein spezifisches *Sicherheitsziel* an. Wie für jedes Sicherheitsmodell aus der Klasse der *Zugriffskontrollmodelle* ist das oberste Sicherheitsziel des *Schematic Protection Modells* der Schutz von Systemressourcen vor unerlaubten Zugriffen.

4.3.2 Modellmerkmale

Die formale Definition des *Schematic Protection Models* kann in angepasster Form anhand der Notation von Kühnhauser in [Sec12] folgendermaßen notiert werden:

$$(T, R, Q, \Sigma, \delta, q_0),$$

wo neben den bekannten Kernkomponenten des Zustandsübergangssystems

$$(Q, \Sigma, \delta, q_0)$$

eine endliche Menge der Sicherheitstypen T und eine endliche Menge der Zugriffsrechten R definiert werden. Die Sicherheitstypen und Zugriffsrechte werden hier als Privilegien verstanden.

Die Merkmale des *Schematic Protection Models* sind in der Tabelle 4.4 abgebildet.

Grundmengen Die Grundmengen des Modells bilden die nicht unbedingt endliche Menge von Entitäten E, die endlichen Mengen von Zugriffsrechten R und Sicherheitstypen T.

Die Menge der Entitäten E besteht aus der Menge der Subjekte S und der Menge der Objekte O, welche disjunkt sind. Ebenso besteht die Menge T aus den disjunkten Mengen der Subjekt-Typen TS und Objekt-Typen TO:

$$T = TS \cup TO, \text{ wobei } TS \cap TO = \emptyset.$$

Die Zugriffsrechte R werden in *Schematic Protection Modell* explizit in neutrale Zugriffsrechte RI und Kontrollzugriffsrechte RC unterteilt, wobei gilt:

$$R = RI \cup RC \text{ und } RI \cap RC = \emptyset.$$

Zugriffskontrollrelationen Die Verteilung der Privilegien im System wird durch zwei Zugriffskontrollrelationen festgelegt. Zum einen wird durch die Funktion $\tau : E \to T$ die Zuordnung der Entitäten zu den Sicherheitstypen beschrieben. Jeder Entität $X \in E$ wird ein Typ aus $t \in T$ mit der Typ-Funktion τ zugeordnet, sodass gilt:

$$\tau(X) = t.$$

Nach einer einmaligen Konfiguration wird die Zuordnung τ durch die Abläufe innerhalb des Modells nicht mehr geändert. Wie bei den attributbasierten Modellen üblich werden die Entitäten zu einem bestimmten Typ zugeordnet, die ähnliche Aufgaben oder Eigenschaften besitzen. Zum Beispiel würden alle Nutzer eines Systems den Typ *user* und alle Administratoren den Typ *admin* zugeordnet bekommen.

Zum anderen wird eine detaillierte Rechtesituation in dem Modell durch die Funktion dom bestimmt:

$$dom : S \to 2^{E \times R}.$$

Die Funktion dom beschreibt dabei so genannte Subjekt-Domänen[3]. Eine Subjekt-Domäne besteht aus *Tickets*.

Ein *Ticket* X/r beschreibt die Rechte eines Subjekts an einem bestimmten Objekt. Es besteht aus einer Entität $X \in E$ und einem Zugriffsrecht $r \in R$, welches bestimmte Operationen auf X erlaubt. Ein Ticket X/z mit $z \in RC$ wird Kontrollticket genannt. Mehrere *Tickets* der Form X/w, X/v und X/y können zu einem *Ticket* X/wvy zusammen gefasst werden. Falls ein *Ticket* von einem Subjekt zu anderen Subjekten kopiert werden darf, wird es mit einer Markierung c gekennzeichnet: $X/r : c$ oder X/rc. Der Typ von einem *Ticket* wird wie bei den Modelleinheiten mit der Typ-Funktion τ bestimmt und ist ein Element der Menge $T \times R$: $\tau(X/r) = \tau(X)/r$. Dabei ist zu beachten, dass die *Tickets* X/r und X/rc unterschiedlichen Typen zugeordnet werden.

Zustände Zu den dynamischen Komponenten des Modells zählen die Subjekt-Domänen und die Menge der Modelleinheiten. Durch die Aktivitäten der Subjekte können diese geändert werden, sodass ein Zustand q und der Zustandsraum Q von dem *Schematic Protection Modell* wie folgt definiert werden[4]:

[3]Die Subjekt-Domänen sind mit den Zeilen einer Zugriffskontrollmatrix vergleichbar.
[4]Vgl. [Sec12]

$$q = (P, dom) \text{ und } Q = 2^E \times (S \to 2^{E \times R}),$$

wobei $q \in Q$, $P \in 2^E$.

Der Initialzustand q_0 ist ein Element aus dem Zustandsraum Q.

Sicherheitsschema Das *Schematic Protection Modell* besitzt ein komplexes Sicherheitsschema, welches auf den statischen Typ-Zuordnungen beruht. Die Schemainformationen werden in den *Zugriffsregeln* verwendet. Das Sicherheitsschema besteht aus:

- einer endliche Menge von $link$-Prädikaten $\{link_i | i = 1 \cdots N\}$,
- einer Menge von Filter-Funktionen $f_i : TS \times TS \to 2^{T \times R}$, wobei zu jedem $link$-Prädikat eine Filter-Funktion gehört,
- einer Demand-Funktion $d : TS \to 2^{T \times R}$,
- einer Menge von Tupeln, die durch die *can-create*-Relation $cc \subseteq TS \times T$ gebildet wird
- und einer *create*-Regel cr für jedes Tupel aus cc.

Mit einem *link*-Prädikat werden Subjekte bestimmt, zwischen denen die Tickets übertragen werden können. Falls das Prädikat $link(X, Y) = true$ liefert, können Tickets generell aus der Domäne von X in die Domäne von Y kopiert werden. Auf diese Weise werden Subjekte miteinander verbunden. Die Semantik des *link*-Prädikats besteht darin, die Domänen der zu verbindenden Subjekte daraufhin zu untersuchen, ob diese entsprechende Kontrollrechte für die Ticketübertragung besitzen. Somit besteht ein *link*-Prädikat aus der Konjunktion oder Disjunktion folgender Ausdrücke:

- $X/z \in dom(X)$, $X/z \in dom(Y)$,
- $Y/z \in dom(X)$, $Y/z \in dom(Y)$ oder
- $true$.

Dabei sind $X \in S$ und $Y \in S$ formale Parameter eines $link_i(X, Y)$-Prädikats und $dom(X)$, $dom(Y)$ entsprechend ihre Domänen. z ist ein beliebiges Kontrollrecht aus RC.

Beispiel 4.3.1:
Für ein besseres Verständnis werden an dieser Stelle mögliche *link*-Prädikate modelliert [San88]:

- $link(X, Y) \equiv Y/send \in dom(X) \wedge X/receive \in dom(Y)$:
  Damit das $link$-Prädikat $true$ wird, müssen zwei Bedingungen erfüllt werden. Zum einen muss X das Kontrollrecht $send$ besitzen, welches ihm erlaubt, Tickets an Y zu verschicken. Zum anderen soll Y das Recht $receive$ haben, um die Tickets von X zu erhalten.

- $link(X, Y) \equiv X/broadcast \in dom(X) \wedge Y/pickup \in dom(Y)$:
  In diesem Fall ist die Ticketübertragung nur von der eigenen Domäne eines Subjekts abhängig. Hat X das Recht Tickets an andere Subjekte zu versenden und hat Y das Recht Tickets von anderen Subjekten zu erhalten, so wird das Prädikat zu $true$ ausgewertet.

- $link(X, Y) \equiv true$:
  Unabhängig davon welche Kontrollrechte X und Y besitzen, werden sie durch das $link$-Prädikat verbunden.

Die $link$-Prädikaten reichen für das Kopieren von Tickets alleine nicht aus. Für jedes $link$-Prädikat soll eine Filter-Funktion $f_i : TS \times TS \rightarrow 2^{T \times R}$ angegeben werden. Mit einer Filter-Funktion wird festgelegt, welche Tickets kopiert werden dürfen. Die Entscheidung wird dabei anhand von Typen der beteiligten Subjekte und Tickets gefallen. Ein Beispiel zu einer Filter-Funktion wird in 4.3.2 erläutert.

Beispiel 4.3.2:
Angenommen zwei Subjekte X und Y sind mit einem $link$-Prädikat verbunden. Nun wird mit der Filter-Funktion f festgelegt, welche Ticket-Typen kopiert werden können [San88]:

- $f(\tau(X), \tau(Y)) = TO \times RI$:
  Nur die Tickets dürfen kopiert werden, die aus einem Objekt und einem neutralen Recht bestehen.

- $f(\tau(X), \tau(Y)) = \emptyset$:
  Keine Tickets dürfen kopiert werden.

Zustandsübergänge In dem *Schematic Protection Modell* werden drei *Elementaroperationen* modelliert, die einen Zustandswechsel initiieren können:

$$\{copy, demand, create\} \subseteq A,$$

wobei A die Aktionsmenge des Zustandsübergangssystems ist. Infolgedessen kann die Menge der erlaubten Eingaben Σ wie folgt definiert werden kann:

$$\Sigma = A \times 2^E \times 2^{(E \times R)},$$
$$\Sigma = \{copy, demand, create\} \times 2^E \times 2^{(E \times R)},$$

wo $2^E \times 2^{(E \times R)}$ Parameter für die Operationen aus A darstellen. 2^E steht dabei für die an der Operation beteiligten Subjekte und Objekte. $2^{E \times R}$ sind die für die Operation benötigten Tickets.

Mit der *copy*-Operation können Tickets von einem Subjekt zu einem anderen kopiert werden. Dabei wird ein zu kopierendes Ticket aus einer Subjekt-Domäne in die Zieldomäne übertragen, wobei das Original-Ticket unberührt bleibt. Außerdem können Subjekte versuchen, die Tickets zu erwerben, indem sie diese mit der *demand*-Operation bei anderen Subjekten anfordern. Mit der *create*-Operation können neue Modelleinheiten in das Modell eingefügt werden.

Für jede elementare Operation werden unterschiedliche *Zugriffsregeln* definiert. Die *Zugriffsregeln* basieren auf dem festgelegten Sicherheitsschema und auf der Überprüfung von der aktuellen Ticketverteilung.

Die durch *copy*-Operation verursachten Zustandsübergänge werden durch drei Regeln autorisiert. Sei Y ein Subjekt, in dessen Domäne ein Ticket $A/x : c$[5] aus der Domäne von X kopiert werden soll. So wird die *copy*-Operation nur dann ausgeführt, wenn alle folgenden Bedingungen erfüllt sind:

1. das zu kopierende Ticket befindet sich in der Domäne von X:
   $(A/x : c) \in dom(X)$,

[5] Nur mit c markierte Tickets können kopiert werden.

2. die Subjekte X und Y sind durch ein *link*-Prädikat verbunden:
   $link_i(X, Y) = true$,

3. der Typ von $A/x : c$ wurde mit der Filter-Funktion zum Kopieren freigegeben:
   $(\tau(A)/x : c) \in f_i(a, b)$.

Die beiden ersten Bedingungen beruhen auf den Tickets und sind damit von der Rechteverteilung in einem gegebenen Zustand abhängig. Die dritte Bedingung basiert auf dem Sicherheitsschema.

Die Demand-Funktion d autorisiert die *demand*-Operation. Mit der Demand-Funktion wird anhand von Sicherheitstypen festgelegt, welche Subjekte unmittelbar auf bestimmte Entitäten zugreifen können. Zum Beispiel kann ein Subjekt X Tickets vom Typ $\tau(A)/x : c$ erwerben, falls X diese anfordert und dabei $(\tau(A)/x : c) \in d(X)$ gilt. Als Ergebnis der *demand*-Operation gehört das angeforderte Ticket im Nachfolgezustand zur Domäne von X.

Die Regeln für die Ausführung von *create*-Operationen werden anhand der *can-create*-Relation und den *create*-Regeln festgelegt. Die *can-create*-Relation $cc \subseteq TS \times T$ beschreibt, welche Typen von Modelleinheiten von welchen Subjekt-Typen erzeugt werden können. Ein Subjekt X darf nur dann eine Modelleinheit Y erzeugen, falls gilt:

$$(\tau(X), \tau(Y)) \in cc\,.$$

Mithilfe der $create$-Regeln wird außerdem festgelegt, wie sich die Zugriffsrechte ändern, wenn neue Entitäten erzeugt werden. Dabei werden zwei Situationen unterschieden: ein Subjekt erzeugt ein neues Objekt und ein Subjekt erzeugt ein neues Subjekt. Im ersten Fall muss mit der *create*-Regel bestimmt werden, welche Rechte der Erzeuger an dem neuen Objekt bekommt. Zum Beispiel bekommt ein Subjekt X an einem von ihm erzeugten Objekt A nur dann das Recht $r \in R$, falls $(\tau(A)/r) \in cr(\tau(X), \tau(A))$ gilt.

Falls ein Subjekt einen neues Subjekt erzeugt, definiert eine *create*-Regel, welche Rechte das neu erzeugte Subjekt an seinem Erzeuger besitzen darf und umgekehrt. Folglich besteht eine *create*-Regel aus zwei Teilen. Angenommen ein Subjekt X vom Typ $x = \tau(X)$ erzeugt ein Subjekt Y vom Typ $y = \tau(Y)$. Die *create*-Regel wird wie folgt definiert:

$$cr(x, y) = LEFT \mid RIGHT,$$

wobei $LEFT \subseteq \{y/r \mid r \in R\}$ und $RIGHT \subseteq \{x/r \mid r \in R\}$. Das Subjekt X darf nur dann ein Recht r an Y bekommen, wenn $(y/r) \in LEFT$. Entsprechend bekommt Y nur die Tickets an X, deren Typ ein Element von $RIGHT$ ist.

Zusammenfassend werden die durch die *create*-Operation verursachten Zustandsübergänge erlaubt, wenn diese durch die *can-create*-Relation autorisiert wurden. Im Nachfolgerzustand werden dabei die Mengen von Subjekten oder Objekten um neue Elemente ergänzt und die Subjekt-Domänen anhand von *create*-Regeln angepasst.

Grundmengen	
Menge der Entitäten	$E = S \cup O$
Menge der Typen	$T = TS \cup TO$
Menge der Rechte	$R = RI \cup RC$
Zugriffskontrollrelationen	
Typ-Zuordnung	$\tau : E \rightarrow T$
Domänen-Funktion	$dom : S \rightarrow 2^{E \times R}$
Zustände	
Zustandsraum	$Q = 2^E \times (S \rightarrow 2^{E \times R})$
Initialzustand	$q_0 \in Q$
Sicherheitsschema	
$link$-Prädikat	$link : S \times S \rightarrow \{true, false\}$
Filter-Funktion	$f_i : TS \times TS \rightarrow 2^{T \times R}$
Demand-Funktion	$d : TS \rightarrow 2^{T \times R}$
can-create-Relation	$cc \subseteq TS \times T$
create-Regel	$cr \subseteq T \times R$
Zustandsübergänge	
Operationen	$A \subseteq \{copy, demand, create\}$
Eingabealphabet	$\Sigma = A \times 2^E \times 2^{(E \times R)}$
Zugriffsregeln	

Tabelle 4.4: Merkmale von *Schematic Protection Model*

4.3.3 Sicherheitseigenschaften

Wie bereits die HRU- und ABAM-Modelle beschränkt sich auch das SPM-Modell auf die Analyse von der *Safety*-Eigenschaft. Ähnlich wie beim ABAM-Modell versucht Sandhu die Beweisbarkeit der *Safety*-Eigenschaft mithilfe der Restriktionen

auf den Typen bzw. auf dem Sicherheitsschema nachzuweisen. Er definiert dabei, was zyklenfreie Sicherheitsschemas sind. Die zyklenfreien Sicherheitsschemas sind die Schemas, bei denen der Nachweis der *Safety*-Eigenschaft entscheidbar ist. Im Falle eines azyklischen Schemas existieren keine zyklischen Beziehungen zwischen Typen. Das Vorhandensein solcher zyklischen Beziehungen macht die Verbreitung der Rechte nicht abschätzbar, was zu der Unentscheidbarkeit führt. Die Zyklenfreiheit des Sicherheitsschemas bezieht sich dabei auf die *can-create*-Regel. Mit anderen Worten dürfen Subjekte keine Subjekte erzeugen, die mächtiger als sie selbst sind.

4.3.4 Modellinstantiierung

Bei der Instantiierung des Modells müssen die Mengen der Subjekt- und Objekt-Typen, Sicherheitsschema und ein initialer Zustand festgelegt werden.

In dem nachfolgenden Beispiel wird ein Szenario modelliert, in dem Subjekte selbst entscheiden können, ob sie ihre Rechte an andere Subjekte weitergeben[6]. Die Modellmerkmale sehen dann wie folgt aus:

- $S = \{s1, s2\}$, $O = \{o1, o2\}$, $RI = \{read : c, write : c\}$, $RC = \emptyset$,
- $dom(s1) = \{o1/read : c\}$, $dom(s2) = \{o2/write : c\}$,
- $TS = \{user\}$, $S = \{file\}$,
- $\tau(s1) = user, \tau(s2) = user, \tau(o1) = file, \tau(o2) = file$,
- $link(s1, s2) \equiv true$, $link(s2, s1) \equiv true$,
- $f(user, user) = \{file/read : c, file/write : c\}$,
- $d(user) = \emptyset$,
- $cc(user) = \{file\}$
- $cr(user, file) = \{file/read : c\}$

Das modellierte System besitzt jeweils zwei Subjekte und Objekte. Es existieren keine Kontrollrechte. In dem Anfangszustand kann $s1$ das Objekt $o2$ lesen und $s2$

[6] Angelehnt an [San88]

in das Objekt $o2$ schreiben. Beide Subjekte sind von dem Typ *user* und alle Objekte sind von dem Typ *file*. Unabhängig davon, ob und welche Kontrolltickets die Subjekte besitzen, können sie beliebige Dateien kopieren. Allerdings dürfen sie keine Tickets anfordern. Jedes Subjekt des Systems darf neue Dateien erzeugen. Dabei erlangt es einen lesenden Zugriff auf die erzeugte Datei. Allgemein ist diese Sicherheitseigenschaft aber nicht bewiesen.

4.4 Bell-LaPadula Modell

Das *Bell-LaPadula Modell* (BLP-Modell) wird hauptsächlich im militärischen oder kommerziellen Bereichen eingesetzt, wo eine strikte Trennung von Zuständigkeitsbereichen bzw. Informationen notwendig ist. In dem Modell wird eine Matrix zur Verwaltung von diskreten Zugriffsentscheidungen und ein Verband zur Festlegung von systembestimmten Regeln modelliert.

Es existieren verschiedene Varianten von BLP-Modellen. In diesem Abschnitt wird eines der einfachsten *BLP-Modelle* [Sec12] vorgestellt, in dem nur die Zugriffsoperationen *lesen* und *schreiben* modelliert werden.

4.4.1 Sicherheitsziel

Das Sicherheitsziel des *BLP-Modells* ist der Schutz von vertraulichen Informationen in einem System. Dabei sollen die gewollten Informationsflüsse überwacht und die Informationsflüsse zu den nicht autorisierten Subjekten verhindert werden.

4.4.2 Modellmerkmale

Das *BLP-Modell* kann wie folgt notiert werden:

$$(S, O, L, Q, \Sigma, \delta, q_0).$$

Die Übersicht über die grundlegenden Modellmerkmale des Modells ist in der Tabelle 4.5 dargestellt.

Grundmengen Subjekte und Objekte werden in dem Modell durch die endlichen Mengen S bzw. O modelliert, welche disjunkt sind.

Des Weiteren besitzen Subjekte und Objekte Sicherheitsklassen. Die endliche Menge aller Sicherheitsklassen wird mit L bezeichnet. Die Menge der Zugriffsrechte ist eine endliche Menge $R \subseteq \{read, write, control\}$. Das Zugriffsrecht $control$ berechtigt Subjekte zur Manipulation der Zugriffsrelationen.

Zugriffskontrollrelationen Die Zugriffsrelationen des Modells sind eine Klassifizierungsfunktion *cl* und eine Zugriffskontrollmatrix. Jedem Subjekt und Objekt eines Systems wird mit einer Funktion *cl* eine Sicherheitsklasse aus der Menge L zugeordnet.

$$cl : S \cup O \rightarrow L.$$

Die Verteilung von Zugriffsrechten wird mit einer Zugriffsmatrix festgehalten:

$$M : S \times O \rightarrow 2^R.$$

Zustände In der hier beschriebenen Variante des Modells können zum einen die Sicherheitsklassen der Subjekte und Objekte geändert werden. Zum anderen ist die Verteilung der Rechte in der Zugriffsmatrix ebenfalls dynamisch. Sei q ein beliebiger Zustand des Modelles, Q dessen Zustandsraum und M eine Zugriffsmatrix, die die aktuelle Rechteverteilung beschreibt. Dann gilt:

$$q = (cl, M) \text{ und } Q = (S \cup O \rightarrow L) \times (S \times O \rightarrow 2^R) \, .$$

Sicherheitsschema Das Sicherheitsschema des Modells ist ein Verband $(L, \leq)$ auf den Sicherheitsklassen aus L definiert.

Zustandsübergänge Die Definition von erlaubten Zustandsübergängen ist ein Teil des Modellinstantiierungsprozesses. Das bedeutet, dass die konkrete Ausprägung der Zustandsübergangsfunktion in dem Modell offen gelassen wird und erst bei einer Modellinstanz festgelegt wird.

$$\delta : Q \times \Sigma \rightarrow Q \, ,$$

wo Σ die Menge der akzeptierten Eingaben des Zustandsübergangssystems ist.

Grundmengen	
Menge der Subjekte	S
Menge der Objekte	O
Menge der Sicherheitsklassen	L
Menge der Rechte	$R = \{read, write, control\}$
Zugriffskontrollrelationen	
Klassifizierung	$cl : S \cup O \to L$
Zugriffsmatrix	$M : S \times O \to 2^R$
Zustände	
Zustandsraum	$Q = (S \cup O \to L) \times (S \times O \to 2^R)$
Initialzustand	$q_0 \in Q$
Sicherheitsschema	
Verband	$\geq$
Zustandsübergänge	
Zustandsübergangsrelation	δ

Tabelle 4.5: *Merkmale von BLP-Modell*

4.4.3 Sicherheitseigenschaften

Ein Zustands des *BLP-Modells* ist dann sicher, wenn die Informationen von Entitäten aus einer höherer Sicherheitsklasse zu den Entitäten einer niedrigeren Sicherheitsklasse fließen. Wenn ein Subjekt zum Beispiel eine Datei ausließt, erlagt es bestimmte Informationen. Wenn ein Subjekt in eine Datei schreibt, gibt er Informationen weiter. Im *BLP-Modell* wird somit die Beschreibung von zulässigen Informationsflüsse durch zwei Regeln ausgedrückt:

Die *no-read-up* Regel besagt, dass Subjekte nur die Objekte lesen dürfen, die die gleiche oder eine niedrigere Sicherheitsklasse besitzen, als sie selbst:

$$\forall s \in S, o \in O : read \in M(s, o) \longrightarrow cl(o) \leq cl(s) \; .$$

Die *no-write-down* Regel besagt, dass Subjekte nur in die Objekte mit einer gleichen oder höheren Sicherheitsklasse schreiben dürfen:

$$\forall s \in S, o \in O : write \in M(s, o) \longrightarrow cl(o) \geq cl(s) \; .$$

Ein Zustand (cl, M) ist im Sinne von *BLP-Modell* sicher, wenn der sowohl die *no-read-up* Regel als auch *no-write-down* Regel erfüllt.

4.4.4 Modellinstantiierung

Bei der Instantiierung des *BLP-Modells* werden die Mengen der Subjekte, Objekte, Sicherheitsklassen und Zugriffsrechte festgelegt. Des Weiteren wird ein Initialzustand definiert. Dabei wird eine initiale Zuordnung der Entitäten zu den Sicherheitsklassen und eine initiale Rechteverteilung in Form einer Zugriffsmatrix beschrieben. Außerdem müssen die Zustandsübergänge instantiiert werden

4.5 Rollenbasiertes Zugriffskontrollmodell

Das in diesem Abschnitt beschriebene *rollenbasierte Modell* (RBAC0-Modell) ist ein minimales Modell aus der Klasse der *rollenbasierten Zugriffskontrollmodelle* [FK07, Eck09]. Das *RBAC0-Modell* formalisiert die grundlegenden Rollen- und Sitzungskonzepte aus Abschnitt 3.3.2.2.

4.5.1 Sicherheitsziel

Das Sicherheitsziel des Modells ist die Durchsetzung der Zugriffskontrolle anhand der Rollen, welche Subjekten statisch zugeordnet werden. Ein Subjekt soll zur Ausführung einer Aktion nur die Rechte aktiviert haben, die es zur Ausführung dieser Aktion benötigt.

4.5.2 Modellmerkmale

Ein *RBAC0-Modell* ist ein Tupel [Eck09]:

$$(S, O, RL, R, sr, pr, session),$$

wobei S die Menge der Objekte, O die Menge der Subjekte, RL die Menge der Rollen, R die Menge der Zugriffsrechte und sr, pr, $session$ die drei in 3.3.2.2 beschrieben *Subjekt-Rollen-*, *Rollen-Zugriffsrechte-* und *Sitzungsrelationen* sind.

Um diese Definition auf das in dieser Arbeit zugrundeliegende Zustandsübergangssystem anzupassen, wird das RBAC0-Modell folgendermaßen definiert:

$$(S, O, RL, R, SE, Q, \Sigma, \delta, q_0),$$

wobei die Sitzungen hier als eine Menge SE modelliert wird. Die Modellmerkmale sind in der Tabelle 4.6 aufgeführt.

Grundmengen Die Menge der Subjekte S und die Menge der Objekte O sind disjunkt. Die Rollen werden durch die Menge RL modelliert. Weitere Grundmengen sind die Menge der Zugriffsrechte R und die Menge der Sitzungen SE. Im Gegensatz zu der Definition von Eckert [Eck09] werden Sitzungen nicht als Relation modelliert, sondern als Menge von Elementarelementen.

Zugriffskontrollrelationen Mit der Zugriffskontrollrelation $sr : S \rightarrow 2^{RL}$ wird statisch definiert, welche Subjekte welchen Rollen zugeteilt sind. Des Weiteren wird mit $pr : RL \rightarrow 2^{R \times O}$ festgelegt, welche Rolle welche Rechte an bestimmten Objekten besitzt. pr ist ebenfalls statisch und kann daher nur durch einen Systemadministrator geändert werden.

Dynamisch sind zwei weitere Zugriffskontrollrelationen $ss : SE \rightarrow 2^{S}$ und $sr : SE \rightarrow 2^{RL}$. Die Relation ss beschreibt, welche Subjekte sich in den aktuellen Sitzungen befinden. Welche Rollen Subjekte dabei aktiviert haben, wird mit der Relation sr ausgedrückt.

Zustände Die Menge der aktuellen Sitzungen kann sich ändern, indem neue Sitzungen erstellt oder gelöscht werden. Folglich verändern sich auch die Relationen ss und sr. Für jede aktive Rolle in einer Sitzung muss es ein aktives Subjekt in derselben Sitzung geben, sodass dieses Subjekt und diese Rolle in der *Subjekt-zu-Rollen*-Relation sr steht.

Sei q ein beliebiger Zustand, Q der Zustandsraum, dann gilt:

$$q = (P_{se}, ss, sr) \text{ und } Q = 2^{SE} \times (SE \rightarrow 2^{S}) \times (SE \rightarrow 2^{RL}) \,,$$

wo $P_{se} \in 2^{SE}$ ist.

Sicherheitsschema Das Sicherheitsschema dieser Variante des Modells legt keine Beziehungen zwischen Rollen fest und ist somit implizit.

Zustandsübergänge Die Menge der akzeptierten Eingaben des Modells wird folgendermaßen definiert.

$$\Sigma = SE \times A \times (S \cup O)^n,$$

wo A die Menge der Zugriffsoperationen und $(S \cup O)^n$ ein Vektor sind, die die formalen Parameter der jeweiligen Operation beschreiben. Außerdem wird festgelegt, in welcher Sitzung die Operationen ausgeführt werden dürfen.

$\delta((P_{se}, ss, sr), (se, act, x)) =$

if $\exists x_{s_i} \in S, x_{o_i} \in O, right \in R :$
$se \in P_{se} \wedge x_{s_i} \in ss(se) \wedge \exists r \in sr(se) :$
$(r, (right, x_{o_i}))$

then $act(x)$,

wo $right$ das für die Ausführung von act benötigte Zugriffsrecht ist.

Wenn in einem Zustand eine Sitzung existiert, in der ein Subjekt ein bestimmtes Objekt zugreifen möchte, dann muss für den Objektzugriff benötigte Rolle in derselben Sitzung aktiviert worden sein.

Grundmengen	
Menge der Subjekte	$S \subseteq O$
Menge der Objekte	O
Menge der Rechte	R
Menge der Rollen	RL
Menge der Sitzungen	SE
Zugriffskontrollrelationen	
Subjekt-Rollen-Relation	$sr : S \rightarrow 2^{RL}$
Rolle-Zugriffsrechte-Relation	$pr : RL \rightarrow 2^{R \times O}$
Sitzung-Subjekte-Relation	$ss : SE \rightarrow 2^{S}$
Sitzung-Rollen-Relation	$sr : SE \rightarrow 2^{RL}$
Zustände	
Zustandsraum	$Q = 2^{SE} \times (SE \rightarrow 2^{S}) \times (SE \rightarrow 2^{RL})$
Initialzustand	$q_0 \in Q$
Sicherheitsschema	
implizit	
Zustandsübergänge	
Zugriffsoperationen	A
Eingabealphabet	$\Sigma = SE \times A \times (S \cup O)^n$
Zugriffsregeln	δ

Tabelle 4.6: Merkmale von *RBAC0-Modell*

4.5.3 Sicherheitseigenschaften

Ein sicherer Zustand im *RBAC0-Modell* ist ein Zustand, in dem in jeder Sitzung nur solche Rollen aktiviert wurden, für die es in derselben Sitzung auch ein Sub-

jekt gibt, dem diese Rolle (statisch) zugeordnet ist. Ein Zustandsübergang ist erlaubt, wenn ein Subjekt nur die Aktionen ausführt, für die es die benötigten Rollen aktiviert hat.

4.5.4 Modellinstantiierung

Das *RBAC0-Modell* wird instantiiert, indem die Mengen der Subjekte, Objekte, Zugriffsrechte, Sitzungen und Rollen definiert werden. Des Weiteren soll bestimmt werden, welche Subjekte welche Rollen bekommen und welche Rollen welche Zugriffsrechte an Objekten bekommen. Mit einem Initialzustand werden die initiale Sitzungen festgelegt. Außerdem sollen die möglichen Zustandsübergänge definiert werden.

4.6 Denning-Modell

Ein Modell zur Beschreibung von Informationsflüssen wurde von Denning in [Den76] vorgeschlagen. Dabei stellte das Modell eine Erweiterung des *Bell-LaPadula-Modells* aus Abschnitt 4.4 dar. Es hatte das Ziel die statischen Informationsflüsse jenseits von Zugriffskontrolle zu beschreiben. In [Den82] erweitert Denning das Modell um dynamische Komponenten wie Zustände und Zustandsübergänge. Auf dieser Grundlage wird hier nun das *Denning-Modell* als ein typischer Vertreter der Klasse *Informationsflussmodelle* formal beschrieben.

4.6.1 Sicherheitsziel

Das Sicherheitsziel des *Denning-Modells* ist die Verhinderung von unautorisierten Informationsflüssen.

4.6.2 Modellmerkmale

Die Merkmale des Modells sind in der Tabelle 4.7 zusammengefasst. Das Modell kann wie folgt notiert werden:

$$(O, SC, W, Q, \Sigma, \delta, q_0)$$

Grundmengen Die Objekte werden durch die Menge O modelliert. Die möglichen Sicherheitsklassen eines System ist die Menge SC. Außerdem enthält die Menge W alle möglichen Werte, die Objekte einnehmen können.

Flusskontrollrelationen Das Modell besitzt zwei Flusskontrollrelationen:

- $oc : O \to SC$, die den Objekten die Sicherheitsklassen zuordnet und
- $ow : O \to W$, die den Objekten einen Wert zuordnet.

Zustände Ein Zustand eines Systems wird durch die aktuellen Sicherheitsklassen und Werten der Objekte definiert. Dabei kann sich die Sicherheitsklasse eines Objektes nur dann ändern, wenn sein Wert sich ändert. Nach Denning [Den82] können neue Objekte erzeugt oder vorhandene Objekte gelöscht werden. Ein Zustand q und der Zustandsraum Q des Modells werden somit folgendermaßen modelliert:

$q = (P_O, oc, ow)$ und $Q = 2^O \times (O \to SC) \times (O \to W)$, wobei $P_O \in 2^O$ ist.

Sicherheitsschema Das Sicherheitsschema ist wie im Falle des *Bell-LaPadula-Modells* ein Verband $(SC, \leq)$ auf den Sicherheitsklassen.

Zustandsübergänge Die akzeptierten Eingaben des Zustandsübergangssystems werden mit Σ modelliert:

$$\Sigma = (A \times (O \cup W \cup SC) \times O) \cup (A \times O).$$

Mit den Aktionen aus A können Objekte erzeugt oder gelöscht werden. Außerdem können die Werte der Objekte oder deren Sicherheitsklassen geändert werden. Hier werden die Zustandsübergänge modelliert, die durch die Veränderung des Werts von einem Objekt durch Kopieren verursacht wurden. Sei *copy* eine Operation zum Kopieren der Objektwerte, dann gilt:

$\delta((P_O, sc, ow), (copy, o_f, o_t)) =$
if $sc(o_f) \leq sc(o_t)$
then $copy(o_f, o_t)$,

wo o_f ein Objekt ist, von dem der Wert kopiert wird, und o_t ein Objekt ist, dem der Wert von o_f zugewiesen wird.

Wenn also die Sicherheitsklasse von einem Objekt o_f kleiner als die von einem anderen Objekt o_t ist, zu dem der Wert von o_f kopiert werden soll, dann darf die Kopier-Funktion act ausgeführt werden. In dem Nachfolgerzustand bleibt der Wert von o_f unverändert und o_t bekommt einen neuen Wert: den von o_f.

Grundmengen	
Menge der Objekte	O
Menge der Sicherheitsklassen	SC
Menge der Objektwerte	W
Flusskontrollrelationen	
Klassifizierung	$O \rightarrow SC$
Wertzuordnung	$O \rightarrow W$
Zustände	
Zustandsraum	$Q = 2^O \times (O \rightarrow SC) \times (O \rightarrow W)$
Initialzustand	$q_0 \in Q$
Sicherheitsschema	
Verband	$(SC, \leq)$
Zustandsübergänge	
Aktionen	$A \subseteq \{create, delete, copy, reclassify\}$
Eingabealphabet	$\Sigma = (A \times (O \cup W \cup SC) \times O) \cup (A \times O)$

Tabelle 4.7: Merkmale von *Denning-Modell*

4.6.3 Sicherheitseigenschaften

So wie beim *Bell-LaPadula-Modell* besagt die Sicherheitseigenschaft des *Denning-Modells*, dass keine Information höherer Sicherheitsklasse zu einer niedrigerer Sicherheitsklasse fließen darf. Diese Eigenschaft wird erreicht, indem ein System nur durch erlaubte Zustandsübergänge seinen Zustand ändert. Ein Zustandsübergang ist erlaubt, wenn dabei gemäß der Sicherheitseigenschaft keine unautorisierte Information fließen kann. Mit anderen Worten ist das *Denning-Modell* sicher, wenn keine Sequenz von Flussoperationen existiert, die unerlaubte Flüsse bezüglich der auf dem Verband festgelegten Regeln verursachen [Den76].

4.6.4 Modellinstantiierung

Bei der Instantiierung des Modells werden die Grundmengen und der Initialzustand definiert.

4.7 Gleichgewichtsmodell

Das *Gleichgewichtsmodell* ist ein Vertreter der Klasse *Transaktionsmodelle*, welches von Grimm in [Gri93, Gri08, Gri09] beschrieben wurde. Die Grundidee hinter dem Modell ist der Schutz der miteinander in einer offenen und dezentralen Umgebung kommunizierenden Parteien. Da in diesem Fall eine zentrale Regulierungsinstanz fehlt, können sich die Kommunikationsteilnehmer (Parteien) unangemessen in Hinsicht auf andere Kommunikationsteilnehmer oder zum eigenen Vorteil verhalten. Infolgedessen definiert Grimm das Konzept des Gleichgewichts.

Äußern die Kommunikationsteilnehmer einen Wunsch mit anderen Teilnehmer zu kommunizieren, der auch positiv entgegengenommen wird, werden sie zu Partnern einer verbindlichen Kooperation. Von da an besitzen die Kooperationspartner ein gemeinsames Kooperationsziel, nämlich die Erfüllung von Verpflichtungen, die sie gegenüber anderen Kooperationsteilnehmer erbringen müssen. Außerdem müssen sie nachweisen können, dass diese Verpflichtungen von ihnen tatsächlich erfüllt worden sind. Damit keine der beteiligten Parteien zu Schaden kommt und ein gemeinsames Kooperationsziel erreicht wird, müssen die Verpflichtungen und deren Nachweise im Gleichgewicht gehalten werden. Eine Kooperation ist im Gleichgewicht, wenn jede Partei ihren Verpflichtungen nachkommt und entsprechende Nachweise liefern kann. Bricht ein Kooperationspartner die Kommunikation ab oder bleiben seine Verpflichtungen bzw. Nachweise aus, ist ein Kooperationsziel nicht mehr erreichbar. Das *Gleichgewichtsmodell* hat die Aufgabe Regeln zu definieren, die anhand von Verpflichtungen und Nachweisen zu einer erfolgreichen Kooperation führen [Gri08].

4.7.1 Sicherheitsziel

Das Sicherheitsziel des *Gleichgewichtsmodells* bezieht sich auf zwei kooperierende Kommunikationsteilnehmer. Für beide Parteien gilt ein gemeinsames Kooperati-

onsziel, das sie zusammen erreichen wollen. Das Ziel des Modell ist die kooperierenden Parteien zu schützen, indem sie erfolgreich gekoppelt werden. Eine erfolgreiche Kopplung liegt vor, wenn entweder beide Parteien ein gemeinsames Kooperationsziel erreichen oder keiner. Dadurch wird eine Kooperation durchgesetzt, bei der keiner der Kooperationspartner benachteiligt wird.

4.7.2 Modellmerkmale

Das Gleichgewichtsmodell wird folgendermaßen formal notiert:

$$(S, N, \Sigma, Q, q_0).$$

Ein Überblick über alle Komponente des Modells ist in der Tabelle 4.8 gegeben. Die Elemente der Instanziierung und ein Beispiel zum Modell werden in dem Abschnitt 4.7.4 betrachtet.

Grundmengen Die Grundengen des Modell bilden eine potenziell unendliche Menge der kommunizierenden Parteien P und eine Menge von Nachrichten N, welche zwischen den Parteien getauscht werden. Des Weiteren wird eine weitere Menge der Transaktionen T definiert:

$$T = P \times \{+, -\} \times N \times P\,.$$

Die Transaktionen beschreiben, wer welche Nachrichten empfangen oder gesendet hat. Sei beispielsweise $p1, p2 \in P$ und $n \in N$. Dann bedeutet eine Transaktion $p1(+n : p2)$, dass $p1$ von $p2$ eine Nachricht n erhält. Im weiteren Verlauf der Arbeit wird $+n$ mit n abgekürzt. Damit wird die genannte Transaktion mit $p1(n : p2)$ angegeben. Wenn $p1$ an $p2$ eine Nachricht sendet, wird dies mit der Transaktion $p1(-n : p2)$ ausgedrückt.

Zustände Jede Partei besitzt einen eigenen Automaten. Die Dynamik dieser Partei-spezifischen Automaten wird durch einen einzigen Automaten simuliert, der das gesamte Kooperationssystem repräsentiert.

Ein Zustand q_p einer Partei besteht aus der Menge der Transaktionen, die er bereits getätigt hat. Somit ist ein Parteizustand eine Art Historie aller zu einem bestimmten Zeitpunkt von dieser Partei gemachten Transaktionen:

$$q_p \in P_T,$$

mit $P_T \in 2^T$.

Ein Zustand des Modells ist die Menge der Zustände aller Parteien zu einem bestimmten Zeitpunkt. Sei q_m ein Zustand und Q_m der Zustandsraum des Modells, dann gilt:

$$Q_m = \{q_m \mid q_m : P \to 2^T\} \text{ und } q_m \in Q_m \text{ und}$$

Ein Zustand des gesamten Systems wird also durch eine Funktion beschrieben, die jeder Partei die Menge ihrer aktuellen Transaktionen zuordnet. Der Zustandsraum ist dann die Menge all dieser Funktionen.

Zustandsübergänge Die Zustandsübergänge werden initiiert, wenn Parteien Transaktionen senden oder empfangen. Da die Semantik der Transaktionen diese Aktionen bereits impliziert, kann das Eingabealphabet des Modells wie folgt notiert werden:

$$\Sigma \subseteq T.$$

Beschrieben werden die Übergänge durch zwei Arten von Transaktionsregeln. Die erste Regelart sind explizite Regeln, die festlegen, wann eine Partei verpflichtet ist, bestimmte Transaktionen auszuführen.

Seien $\sigma_0 \ldots \sigma_i \in T$ die Transaktionen einer Partei p in einem Zustand q_p, die sie bisher getätigt oder empfangen hat. Das Vorhandensein dieser Transaktionen in dem Zustand q_p verpflichtet sie zur Vollbringung weitere Transaktionen $\tau_0 \ldots \tau_i \in T$ gegenüber der kooperierenden Partei bzw. Parteien:

if $\sigma_0 \in q_p(p) \wedge \ldots \wedge \sigma_i \in q_p(p)$
then $\tau_0; \ldots \tau_j;$

mit $i, j \in \mathbb{N}$.

Ein typisches Beispiel für diese Regelart ist die Bestimmung der Pflichtlage beim Verkauf von Gütern [Gri08]. Falls ein Käufer die Ware eines Verkäufers bezahlt hat, ist der Verkäufer verpflichtet, diese Ware auch auszuliefern.

Für jede dieser explizit zu definierenden Regeln existieren zwei implizite Regelarten, die die Beweislage beschreiben. Da die kooperierenden Parteien bei der Ausführung bestimmter Transaktionen Verpflichtungen eingehen, müssen diese Transaktionen auch nachgewiesen bzw. quittiert werden können.

Die erste implizite Regelart besagt, dass wenn eine Partei Transaktionen empfängt, die sie in eine Pflichtlage versetzen könnte, dann muss der Empfang dieser Transaktion gegenüber der kooperierenden Partei bzw. Parteien bestätigt werden. Diese Regel betrifft also die Vorbedingung der expliziten Transaktionsregel.

Sei $\sigma_k \in T$ eine empfangene Transaktion einer Partei p, und $\upsilon_k \in T$ eine Bestätigung dieser Transaktion, wobei $0 \leq k \leq i$ gilt. Regeln dieser Art lassen sich formal wie folgt definieren:

if $\sigma_k \in q_p(p)$
then υ_k.

Die Bestätigung υ_k wird also nach dem Empfang von σ_k an die entsprechenden Kooperationspartner gesendet. Zum Beispiel muss ein Verkäufer den Erhalt der Zahlung eines Käufers bestätigen, indem er eine Quittung dem Käufer ausstellt. Dabei ist anzumerken, dass in dem hier aufgeführten Beispiel ein Verkäufer erst nach der Eingang einer Zahlung die Ware an die Käufer liefern muss.

Die zweite implizite Regelart besagt, dass wenn eine Partei die Verpflichtungen gegenüber ihrem Kooperationspartner bzw. Kooperationspartnern erfüllt, dann müssen diese die Vollbringung der Verpflichtungen bestätigen.

Sei $\tau_l \in T$ eine Verpflichtung der Partei $p \in P$, die sie gegenüber dem Empfänger $p_e \in P$ vollbracht hat, und $\upsilon_l \in T$ die Bestätigung dieser Verpflichtung dann gilt:

if $\tau_l \in q_p(p_e)$
then υ_l.

Wenn also eine Partei p einer Verpflichtung τ_l gegenüber Partei p_e nachgekommen ist, dann muss die durch eine Transaktion υ_l von p_e an p bestätigt werden. Beispielsweise muss ein Käufer, der von einem Verkäufer die Ware erhalten hat, dies gegenüber dem Verkäufer auch bestätigen.

Analog zu den Makrooperationen vom *HRU-Modell* kann gleichzeitig eine Folge von Transaktionen ausgeführt werden, die seinen Zustand durch implizite Zwischenzustände in den Folgezustand versetzt. In diesem Fall ist eine Makrooperation nur so vorhanden, dass eine einzige Partei mehrere Transaktionen gleichzeitig tätigen kann. Zum Beispiel kann eine Transaktion zusammen mit Quittierungen geschickt werden.

Grundmengen	
Menge der Parteien	P
Menge der Nachrichten	N
Menge der Transaktionen	T
Zustände	
Zustandsraum	$Q_m = \{q_m \mid q_m : P \rightarrow 2^T\}$
Initialzustand	$q_{m_0} \in Q$
Zustandsübergänge	
Eingabealphabet	$\Sigma \subseteq T$
Operationen	$\{send, receive\}$
Transaktionsregeln	

Tabelle 4.8: Merkmale von *Gleichgewichtsmodell*

4.7.3 Sicherheitseigenschaften

Für die Sicherheitseigenschaften des Modells werden Regeln für die sicheren Zustände und erlaubte Zustandsübergänge definiert, die Verpflichtungen und ihre Beweise im Gleichgewicht halten sollen. Dabei soll verhindert werden, dass die Parteien mehrere aufeinander folgende Verpflichtungen ohne entsprechende Beweise haben.

Der Startzustand des Modell ist sicher. Ein sonstiger Zustand ist sicher, wenn in diesem Zustand jede Partei beweisen kann, dass die jeweiligen Kooperationspartner alle Transaktionen empfangen haben, die diese in eine Situation bringen könnte, in der sie einer Verpflichtung nachkommen müssen. Diese Bedingung muss nicht für Transaktionen erfüllt sein, die im Vorzustand gemacht worden sind. Mit anderen Worten muss der Empfänger einer Leistung beweisen können, dass der Kooperationspartner diese zu erbringen hat.

Ein Zustandsübergang t ist erlaubt, wenn das Modell sich in einem Zustand befindet, indem alle notwendigen Bestätigungen bzw. Quittungen erbracht worden sind oder durch t erbracht werden.

Wenn der aktuelle Zustand sicher ist und eine erlaubte Transition ausgeführt wird, dann ist der Folgezustand auch sicher [Gri09].

4.7.4 Modellinstantiierung

Bei der Instantiierung des Modells sollen der Startzustand, in dem jede Partei eine leere Transaktionshistorie hat, und die Grundmengen wie die beteiligten Parteien, Nachrichten und Transaktionen festgelegt werden. Des Weiteren sollen Transaktionsregeln festgelegt werden. Das bedeutet, wann eine Partei verpflichtet ist, bestimmte Transitionen auszuführen. Eine Beispielinstanz wird nun in dem Beispiel 4.7.1 betrachtet.

Beispiel 4.7.1:
In diesem Beispiel wird eine angepasste Version des HTTP-Protokolls mit dem *Gleichgewichtsmodell* modelliert.

In dem Beispielsystem existieren ein Server S und ein Client C, die einander Anfragen req, Antworten res und deren Bestätigungen $q(req)$ und $q(res)$ zuschicken können. Dabei kommunizieren sie mittels folgender Transaktionen:

$$T_S = \{S(req : C), S(-res : C), S(-q(req) : C), S(q(res) : C)\};$$
$$T_C = \{C(-req : S), C(res : S), C(q(req) : S), C(-q(res) : S)\},$$

wobei T_S die Transaktionen des Servers S und T_C die Transaktionen des Clients C sind. Dabei gilt:

$$T = T_S \cup T_C.$$

Die Transaktionsregel lautet, dass wenn der Server eine Anfrage req empfängt, muss er dem Client eine entsprechende Antwort res senden:

$$\textbf{if } S(req : C) \textbf{ then } S(-res : C).$$

Nach den Beweisregel muss der Server eine Bestätigung des Empfangs einer Anfrage an den Client senden. Außerdem muss der Client eine Bestätigung des Empfangs der Serverantwort an den Server schicken.

Der Ablauf der Modells ist in der Abbildung 4.2 dargestellt. In dem Anfangszustand haben Server und Client noch keine Transaktionen ausgeführt. In dem ersten Schritt *(1)* sendet Client eine Anfrage an den Server, der seinerseits diese empfängt. Das System ändert seinen Zustand, indem beide Transaktionen stattgefunden haben. In dem Folgezustand besitzen die Kooperationspartner die bisher getätigten Transitionen.

In dem zweiten Schritt *(2)* sendet der Server die Bestätigung des Anfrageempfangs und die entsprechende Antwort an den Client zurück, welche dieser auch empfängt.

Im dritten Schritt *(3)* versendet der Client eine Bestätigung, dass er die Antwort vom Server erhalten und zur Kenntnis genommen hat. Diese Bestätigung nimmt der Server entgegen.

Nach dem Ablauf der Kooperation besitzen die kooperierenden Parteien alle Transaktionen, die sie ausgeführt haben, und entsprechende Nachweise zu diesen Transaktionen, die sie in einem Streitfall bzw. im Falle einer Fehlfunktion des Protokolls vorlegen können.

Das angegebene Modell ist sicher bzw. besitzt die Gleichgewichtseigenschaft, da zum einen der Server vor dem Versenden einer Antwort, eine Bestätigung der Anfrage des Clients bestätigt hat. Zum anderen hat der Client eine Bestätigung der Antwort von dem Server gesendet, bevor von dem weitere Transaktionen ausgeführt wurden. Dabei ist es nicht von Belangen, ob zuerst die Antwort oder die Bestätigung des Servers vom Client empfangen wird.

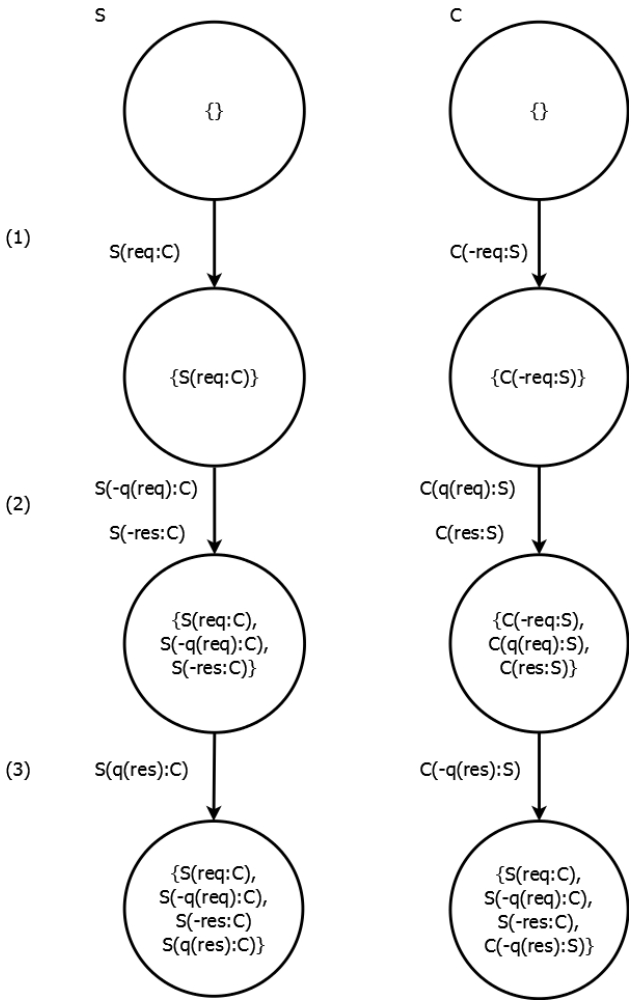


Abbildung 4.2: Server-Client Gleichgewicht

Kapitel 5

Vergleich der Sicherheitsmodelle

In diesem Kapitel werden die in dem vorangegangenen Kapitel beschriebenen Sicherheitsmodelle miteinander verglichen. Als Kriterien werden dabei die Eigenschaften und Aspekte der Sicherheitsmodelle aus dem Kapitel 2 verwendet. Ein zusammenfassender Überblick des Vergleichs ist in der Tabelle 5.1 gegeben.

5.1 Anwendungsbereiche

In dem Grundlagenkapitel (Kapitel 2) wurden akademischer, kommerzieller und militärischer Anwendungsbereiche eingeführt, in denen unterschiedliche Aspekte der Sicherheitsmodelle in den Vordergrund gestellt werden.

Mit den akademischen HRU-, ABAM- und SPM-Modellen werden die wissenschaftliche Seite der Zugriffskontrolle betrachtet. In diesem Fall wird untersucht, welche Konzepte sich mit welchen Mitteln im Allgemeinen umsetzten lassen, welche Probleme dabei entstehen und wie diese zu lösen sind. Wie in dem Grundlagenkapitel bereits erwähnt wurde, sind die typischen wissenschaftlichen Probleme zum Beispiel die Bestimmung der Komplexität der Modelle oder die allgemeine Verwendbarkeit solcher Modelle in bestimmten Anwendungsszenarien.

Die BLP-, RBAC- und Gleichgewichtsmodelle sind praxisorientiert ausgelegt. Mit solchen Sicherheitsmodellen soll im Bezug auf reale Systeme beispielsweise ein konkretes Anwendungswissen abgebildet werden können. Dabei muss festgestellt werden, welche Stakeholder in einem zugrundeliegenden Szenario bzw. System existieren und wie diese zu repräsentieren sind. In einem Krankenhaus-

szenario fungieren zum Beispiel Oberärzte, Ärzte und Krankenschwester als Stakeholder und haben dabei unterschiedliche Aufgabenbereiche. Mit einem RBAC-Modell soll dieses Anwendungswissen so repräsentiert werden, dass die zugrundeliegenden Sicherheitsziele erfüllt sind. Einer der Aspekt wäre in diesem Fall, die Übereinstimmung des Szenarios mit dem gebildeten Sicherheitsmodell zu untersuchen.

5.2 Formalisierungsgrad

Die meisten der in der Arbeit betrachteten Sicherheitsmodelle sind formal. Die einzige Ausnahme stellt das Gleichgewichtsmodell dar. Es impliziert das Vorhandensein einer unabhängigen Vertrauensinstanz. Im Falle eines Streits oder einer abgebrochenen Kooperation, können dieser Instanz die Quittungen bzw. Nachweise der erbrachten Verpflichtungen vorgelegt werden. Dadurch kann sie entscheiden, welche Partei ihre Verpflichtung erfüllt hat und welche ihrer ausstehenden Leistungen noch zu erbringen sind. Beispiele für solche Vertrauensinstanzen können ein Gericht oder eine virtuelle Anlaufstelle sein, denen alle Kommunikationsteilnehmer vertrauen.

5.3 Abstraktionsgrad

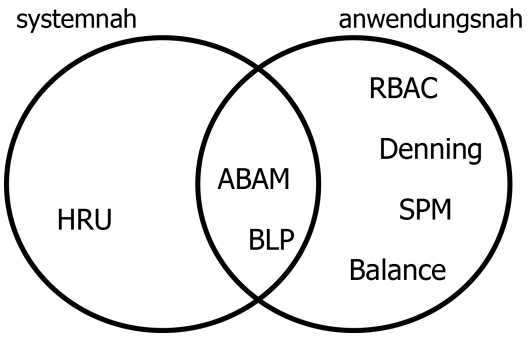


Abbildung 5.1: Abstraktionsgrad

Das HRU-Modell kann Systeme beschreiben, die eine Komponente für den Zugriffsschutz explizit implementiert haben. Es simuliert eine solche Komponen-

te mit einer Zugriffsmatrix und den entsprechenden Manipulationsregeln, weshalb eine Systemnähe gegeben ist.

Ähnlich wie bei dem HRU-Modell simuliert das BLP-Modell die Zugriffsschutzkomponente eines Systems, wodurch es systemnah ist. Allerdings besitzt BLP die Möglichkeit, Sicherheitsklassen und deren Beziehungen, wie sie in einer Anwendung vorliegen, zu definieren. Deshalb ist dieses Sicherheitsmodell anwendungsnäher als das HRU-Modell. Das ABAM-Modell simuliert auch eine Zugriffsschutzkomponente und ist dabei ebenfalls wie das BLP-Modell anwendungsnah.

Die RBAC-, Denning-, SPM- und Gleichgewichtsmodelle basieren auf dem Anwendungswissen und sind dabei unabhängig von einer möglichen Implementation.

5.4 Sicherheitsziel

In der Abbildung 5.2[1] wird der Zusammenhang zwischen den generell existierenden Sicherheitszielen und der Einsatzumgebung hergestellt, wo diese Ziele erreicht werden sollen.

Im Mittelpunkt der klassischen Zugriffs- und Flusskontrollen stehen geschlossene Systeme und der Schutz seiner systeminternen, sensiblen Informationen bzw. Ressourcen, wobei der Schutz der Interessen von den agierenden Subjekten eine untergeordnete Rolle spielt. (Es sei denn sie sind selbst zu schützende Objekte.) Das typische Sicherheitsziel ist dabei zum Beispiel die Rechteintegrität bei den HRU-,ABAM- und SPM-Modellen. Die Rechteintegrität ist gegeben, wenn Subjekte keine Zugriffsrechte erlangen können, die ihnen nicht zustehen. Weitere Sicherheitsziele aus diesem Bereich sind im Allgemeinen die Verhinderung von unerlaubten Informationsflüssen (Denning-Modell) und im Speziellen die Verhinderung von Flüssen von vertraulichen Informationen zu den unautorisierten Subjekten (BLP-Modell).

Im Gegensatz zu den Zugriffs- und Flusskontrollen steht bei der Transaktionskontrolle die Sicherheit in offenen Systemen wie das Internet im Vordergrund, wobei sich hieraus andere Sicherheitsziele ergeben. Die Sicherheitsziele der Transaktionskontrolle beziehen sich nun auf den Schutz der kommunizie-

[1] Inspiriert durch [SP04]

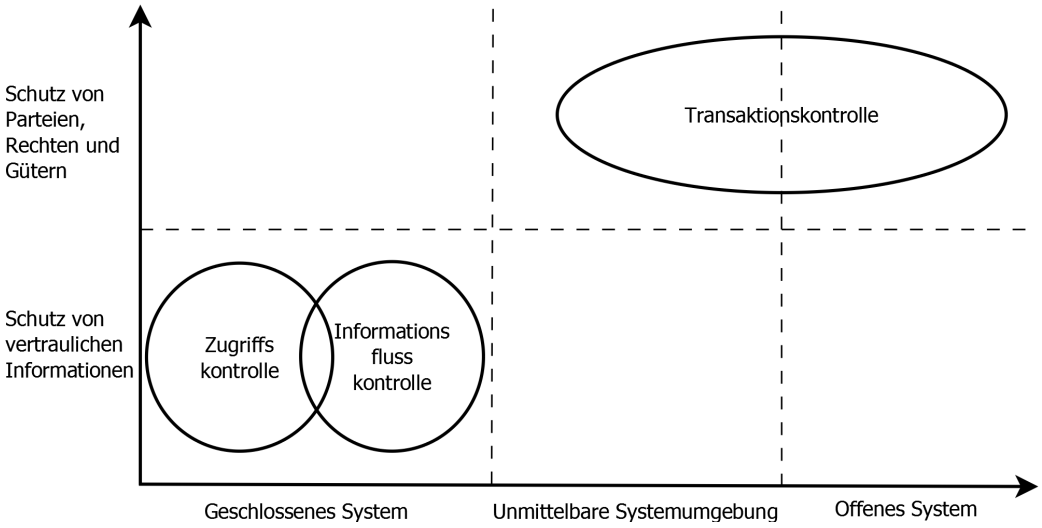


Abbildung 5.2: Sicherheitsziele

renden Parteien und ihren digitalen Rechten oder Gütern wie zum Beispiel Lizenzen oder Musikstücken. Die Kommunikation findet dabei nicht nur zwischen natürlichen Personen sondern auch zwischen Personen und maschinellen Systemen statt. Durch letzteres umfasst die Transaktionskontrolle ebenfalls die unmittelbare Systemumgebung. So ist beispielsweise das Sicherheitsziel von dem Gleichgewichtsmodell die kooperierenden Parteien in einem offenen System zu schützen, indem entweder alle oder keiner der Parteien die Kooperation erfolgreich abschließen können. Die Idee ist dabei, dass die kooperierenden Parteien eine beweispflichtige Bindung eingehen, die in einem offenen System sonst nicht gegeben ist.

5.5 Modellmerkmale

In den Kapiteln 3 und 4 wurden die grundlegenden Modellmerkmale bereit erläutert. Das spezifische Merkmal aller in dieser Arbeit betrachteten Sicherheitsmodelle stellt dabei ein Zustandsübergangssystem dar, woraus sich die gemeinsamen Modellkomponenten wie beispielsweise Zustände und Zustandsübergänge ergeben. Die Ausnahme stellt hier das Gleichgewichtsmodell dar, bei dem pro Partei ein eigener Automat definiert wird. Alle Automaten arbeiten dabei parallel.

Weitere spezifischen Modellmerkmale sind Attribute. Attribute sind im Grunde universell, weil sich damit Sicherheitstypen und Sicherheitskategorien ausdrücken lassen. Daher hängt die Wahl zwischen Attributen, Typen oder Sicherheitskategorien von der intuitiven Semantik ab, die einen gegebenen Zusammenhang am besten beschreibt. Mit den Attributen können sowohl diskrete als auch systembestimmte Regeln beschrieben werden. Im ersten Fall kann zum Beispiel pro Subjekt ein eigenes Attribut festgelegt werden. Der Vorteil beim Einsatz von Attributen liegt aber bei der Beschreibung von semantischen Gruppen, welche die gleiche Aufgaben bzw. Eigenschaften besitzen. Dadurch können zum Beispiel systembestimmte Zugriffsregeln flexibler und präziser konfiguriert werden. Die Verwendung von Attributen wurde in dieser Arbeit im Rahmen der Zugriffs- und Flusskontrolle gezeigt. Aber auch bei den Transaktionsmodellen können in der Regel Attribute verwendet werden, um beispielsweise bestimmte Kommunikationsmuster wie „Server“ oder „Client“ zu beschreiben.

5.6 Skalierbarkeit

Die Skalierbarkeit beschreibt den Verwaltungsaufwand, wenn neue Subjekte oder Objekte in ein Sicherheitsmodell eingefügt werden. Diesbezüglich skaliert das HRU-Modell schlecht, da beim Einfügen oder Löschen neuer Subjekte oder Objekte die Zugriffsregeln für jede Einheit angepasst werden müssen.

Bei den ABAM-, BLP und SPM-Modellen hängt die Skalierbarkeit von der Ebene, auf der die Anpassung von den Zugriffsregeln durchgeführt werden muss. Müssen die globalen Zugriffsentscheidungen auf den Attributen, Sicherheitsklassen oder Typen angepasst werden, dann ist der Aufwand eher gering, da es in diesem Fall genügt die neuen Elemente den bereits existierenden Attributen zuzuordnen. Wenn die Verteilung der Zugriffsrechte bzw. Tickets auf Ebene der Subjekte und Objekte verändert werden muss, dann ist der Aufwand wie bei dem HRU-Modell.

Die RBAC- und Denning-Modelle skalieren sehr gut, da die Regeln auf Rollen bzw. Klassen definiert werden und deshalb nicht weiter angepasst werden müssen, falls weiter Subjekte oder Objekte hinzukommen.

Falls neue Kommunikationspartner bei dem Gleichgewichtsmodell eingefügt werden, müssen die Transaktionsregeln für jede der Parteien festgelegt werden. Daher skaliert das Modell diesbezüglich schlecht.

5.7 Ausdruckskraft vs. Analysierbarkeit

Die Ausdruckskraft und die Analysierbarkeit sind zwei sich ausschließende Modelleigenschaften, die erst durch eine ausführliche Analyse und Beweisführung nachgewiesen werden müssen.

Die HRU-, ABAM- und SPM-Modelle besitzen eine sehr starke Ausdruckskraft, da ihre Definitionen für die unterschiedlichsten Szenarien flexibel konfiguriert werden können. Dadurch können mit diesen Sicherheitsmodellen eine breite Gruppe von Anwendungsszenarien und Systeme modelliert werden[San88, ZLN05, HRU76, Sec12].

Bei der Bestimmung der Analysierbarkeit steht ein Analyseziel im Mittelpunkt. Bei den HRU-, ABAM- und SPM-Modellen ist das Analyseziel die HRU-Sicherheit [Sec12]. Ein Modell gilt als HRU-sicher, wenn die HRU-Eigenschaft stets erfüllt ist. Die Analysierbarkeit besagt nun, wie aufwändig der Beweis dieses Analyseziels im Rahmen eines Sicherheitsmodells ist. Bei den Sicherheitsmodellen mit einem unendlichen Zustandsraum stellt sich dabei die Frage, ob ein Algorithmus existiert und terminiert, der diesen Beweis berechnen könnte. Wenn so ein Algorithmus existiert, ist die Analysierbarkeit entscheidbar, anderenfalls unentscheidbar.

Wenn die Analysierbarkeit eines bestimmtes Analyseziels entscheidbar ist, soll dann eine kleinste Oberschranke des dabei verrichteten Aufwands angegeben werden. Die Analysierbarkeit der Sicherheitsmodelle mit einem endlichen Zustandsraum ist entscheidbar.

Das in dieser Arbeit beschriebene HRU-Modell ist bezüglich der HRU-Sicherheit nicht entscheidbar [HRU76, Eck09]. Die SPM- und ABAM-Modelle sind dabei bezüglich der HRU-Sicherheit mit Annahme der Zyklenfreiheit des Sicherheitschemas entscheidbar. Wenn also in diesen Sicherheitsmodellen keine Subjekte erzeugt werden, die mächtiger als die erzeugenden Subjekte[2] sind, dann kann generell bestimmt werden, dass die HRU-Sicherheit mit computergestützten Mitteln nachgewiesen werden kann. Wie in [San88, ZLN05] beschrieben, reduziert die Zyklenfreiheit die praktische Anwendbarkeit der Modelle nicht.

[2]Solche Subjekte besitzen mehr Zugriffsrechte.

	HRU	ABAM	SPM	RBAC	BLP	Denning	Balance
Sicherheitsziel	Rechte-integrität	Rechte-integrität	Zugriffs-schutz	Zu-griffs-schutz	Schutz von vertraulichen lichen Infor-mationen	Verhin-derung von unautorisier-ten Informa-tionsflüssen	Gleich-gewicht
Anwendungsbereich	a	a	a	k	m	a	k
Abstraktionsgrad	system	system anw	anw	anw	system anw	anw	anw
Formalisierungsgrad	formal	formal	formal	formal	formal	formal	semiformal
Modellmerkmale	Matrix	Attribute, Matrix	Sicherheits-typen	Rollen	Sicherheits-klassen, Matrix	Sicherheits-klassen	parallele Automaten
Skalierbarkeit	-	+/-	+/-	+	+/-	+	-

[a] akademisch; [m] militärisch; [k] kommerziell;

[system] systemnah; [anw] anwendungsnah;

[-] schlechte Skalierbarkeit; [+/-] variable Skalierbarkeit; [+] gute Skalierbarkeit;

Tabelle 5.1: Modellvergleich

Kapitel 6

Alternative Modellierungsmethode

In den vorangegangenen Kapitel wurde die Dynamik der Sicherheitsmodelle mit einem Zustandsübergangssystem beschrieben. In diesem Kapitel sollen Eindrücke vermittelt werden, wie die Sicherheitsmodelle mit der *Communicatig Sequential Processes* (CSP) dargestellt werden können. CSP ist eine von Hoare entwickelte Prozessalgebra, die für die Beschreibung von interagierenden Prozessen benutzt werden kann.

6.1 Gleichgewichtsmodell mit CSP

In diesem Abschnitt wird exemplarisch gezeigt, wie das Gleichgewichtsmodell mittels CSP simuliert werden kann. Als Beispiel wird dabei eine Verkaufsaktion genommen. Die bei dem Gleichgewichtsmodell benötigten Quittungen und Belege werden zur Vereinfachung des Beispiels nicht dargestellt.

RUN := VENDOR || CUSTOMER
VENDOR := $ch!offer \rightarrow ch?order \rightarrow ch!delivery \rightarrow ch?payment \rightarrow$ VENDOR
CUSTOMER := $ch?offer \rightarrow ch!order \rightarrow ch?delivery \rightarrow ch!payment \rightarrow$ STOP

In dem Beispiel werden drei Prozesse RUN, VENDOR und CUSTOMER definiert. Der Prozess RUN sorgt dafür, dass die beiden anderen Prozesse parallel (||) ablaufen. Der Prozess VENDOR simuliert das Verhalten des Verkäufers und der Prozess CUSTOMER das Verhalten des Kunden.

Zunächst schickt (!) der Verkäufer ein Angebot ($offer$) über den Kommunikationskanal ch. Der Kunde empfängt (?) dieses Angebot über den selben Kommunikationskanal ch und schickt ein Kaufgebot ($order$) zurück. Darauf versendet der Verkäufer die Ware ($delivery$). Nach dessen Erhalt schickt der Kunde das Geld ($payment$) zurück und der CUSTOMER-Prozess wird durch STOP beendet. Nachdem der Verkäufer die Bezahlung erhalten hat, beginnt der VENDOR-Prozess von vorne. Der Verkäufer macht also auf dem Kommunikationskanal ein erneuertes Angebot.

6.2 HRU-Modell mit CSP

Da die vollständige Simulation des HRU-Modells mit CSP zu komplex wäre, wird im Rahmen des nachfolgenden Beispiels gezeigt, wie eine Zugriffsmatrix simuliert werden könnte.

$MATRIX = MATRIX(s, o, r)$
$MATRIX(s, o, r) =$
$(\mu X \bullet in?command \rightarrow in?s_i \rightarrow in?o_i \rightarrow in?r_i \rightarrow$
if $command = CONTAINS \vee command = DELETE$ **then**
if $s = s_i \wedge o = o_i \wedge r = r_i$ **then**
$out!FALSE \rightarrow X$
else
$in!command \rightarrow in!s_i \rightarrow in!o_i \rightarrow in!r_i \rightarrow$
$out?result \rightarrow out!result \rightarrow X$
else
if $s = s_i \wedge o = o_i \wedge r = r_i$ **then**
$in!command \rightarrow in!s_i \rightarrow in!o_i \rightarrow in!r_i \rightarrow Y$
else
$in!command \rightarrow in!s_i \rightarrow in!o_i \rightarrow in!r_i \rightarrow$
$out?result \rightarrow out!result \rightarrow X)$
$\rightarrow$

$(\mu Y \bullet in?command \rightarrow in?s_i \rightarrow in?o_i \rightarrow in?r_i \rightarrow$
if $command = CONTAINS \vee command = INSERT$ **then**
if $s = s_i \wedge o = o_i \wedge r = r_i$ **then**
$out!TRUE \rightarrow Y$
else
$in!command \rightarrow in!s_i \rightarrow in!o_i \rightarrow in!r_i \rightarrow$
$out?result \rightarrow out!result \rightarrow Y$
else
if $s = s_i \wedge o = o_i \wedge r = r_i$ **then**
$in!command \rightarrow in!s_i \rightarrow in!o_i \rightarrow in!r_i \rightarrow X$
else
$in!command \rightarrow in!s_i \rightarrow in!o_i \rightarrow in!r_i \rightarrow$
$out?result \rightarrow out!result \rightarrow Y)$

Für die Zugriffsmatrix $S \times O \times R$ existiert für jedes mögliche Tripel ein Prozess $MATRIX(s, o, r)$. Alle diese Prozesse werden durch den Prozess $MATRIX$ parallel ausgeführt. Jeder dieser Prozesse benötigt die Eingabeparameter $command$, s_i, o_i und r_i. $command$ gibt an, welche Aktion ausgeführt werden soll. Da einige Aktionen einen Rückgabewert benötigen, aber alle durch denselben Prozess simuliert werden, muss immer ein Rückgabewert zurückgegeben werden. Mögliche Aktionen sind:

INSERT : fügt (s_i, o_i, r_i) in die Matrix ein. Der Rückgabewert dieser Aktion kann ignoriert werden.

CONTAINS : liefert $TRUE$ zurück, falls (s_i, o_i, r_i) in der Matrix enthalten ist, ansonsten $FALSE$.

DELETE : löscht (s_i, o_i, r_i) aus der Matrix. Der Rückgabewert wird hier ebenfalls ignoriert.

$MATRIX(s, o, r)$ definiert für alle Tripel der Menge $S \times O \times R$ einen eigenen Prozess, bei dem s, o und r durch die jeweiligen Werte ersetzt werden. Diese Prozesse bestehen aus zwei μ-Konstrukten, die mit $\rightarrow$ verbunden sind. μX steht dabei für den Zustand, dass ein bestimmtes Tripel in der Matrix nicht existiert, also das r nicht in der Matrixzelle $m[s, o]$ enthalten ist. μY steht dabei für einen existierenden Eintrag in der entsprechenden Matrixzelle.

Als erstes wird der Fall μX betrachtet, indem ein Recht r nicht in $m[s, o]$ enthalten ist. Dabei werden zunächst alle vier Eingabeparameter eingelesen. So bedeutet beispielsweise $in?command$, dass von einem Eingabekanal in ein Wert eingelesen wird und in der Variable $command$ gespeichert wird.

Falls der aktuelle Befehl $DELETE$ oder $CONTAINS$ ist, wird überprüft, ob die Eingabeparameter dem aktuellen Prozess entsprechen. In diesem Fall wird in Zeile 6 $FALSE$ zurückgegeben. Dabei bedeutet $out!FALSE$, dass $FALSE$ auf dem Ausgabekanal out herausgeschrieben wird. Da es sich um den richtigen Prozess handelt, müssen die Eingabeparameter nicht erneuert auf den Eingabekanal in ausgeschrieben werden. Dadurch wird verhindert, dass falsche Prozesse diese Parameter erneuert einlesen können, was zu einer Endlosschleife führen würde.

Sollte dieser Prozess nicht der gewünschte sein, werden die Eingabeparameter in Zeile 8 zunächst wieder auf den Eingabekanal in herausgeschrieben, damit ein anderer $MATRIX(s, o, r)$-Prozess sie verarbeiten kann. Anschließend wartet der Prozess bis der richtige Matrix-Prozess die Ausgabe erzeugt hat. Sollte die Ausgabe vorliegen, kann sie in der Zeile 9 gelesen und wieder herausgeschrieben werden. Dadurch wird sichergestellt, dass der richtige Prozess auch irgendwann die Eingabeparameter einlesen kann und sie nicht nur von falschen Prozessen verarbeitet werden. Zum Abschluss springt der Prozess durch X wieder an den Anfang von μX.

Sollte ein $INSERT$-Befehl gekommen sein, werden die Eingabeparameter wieder auf den Eingabekanal in herausgeschrieben und der Prozess wechselt danach in den Zustand μY. Die Arbeitsweise in diesem Zustand läuft analog ab.

6.3 Sicherheitseigenschaften

CSP bietet Konstrukte, durch die die Menge aller möglichen *Traces* beschrieben werden. Ein *Trace* stellt dabei eine Folge von allen möglichen Ereignissen wie zum Beispiel das Lesen oder das Schreiben auf einem Kommunikationskanal dar. Die Sicherheitseigenschaften eines Sicherheitsmodells können hiermit als Aussagen über die *Traces* formuliert werden.

Kapitel 7

Fazit und Ausblick

In dem abschließenden Kapitel dieser Arbeit wird zunächst eine kurze Zusammenfassung der behandelten Themen und deren Umsetzung gegeben. Des Weiteren wird in dem Abschnitt 7.2 beschrieben, wie die behandelten Aspekte der Arbeit weitergeführt und erweitert werden können. Schließlich wird dieses Kapitel mit einem kurzen Fazit abgeschlossen.

7.1 Zusammenfassung

In dieser Arbeit wurden die verschiedenen Aspekte der Sicherheitsmodelle betrachtet und anhand einer systematischen Analyse untersucht. Dabei standen die konzeptuellen und strukturellen Besonderheiten der Modellierung im Vordergrund.

Sicherheitsmodelle sind abstrakte Abbildungen von realen Sachverhalten und Zusammenhänge. Mit den Sicherheitsmodellen sollen die komplexen, sicherheitsrelevanten Gegebenheiten veranschaulicht werden. Besondere Gruppen der Sicherheitsmodelle stellen dabei die formalen und semiformalen Sicherheitsmodelle dar, mit denen die Eigenschaften der zu modellierenden Szenarien und Systeme analysiert werden können. Wegen der Präzision und der computergestützten Interpretierbarkeit sind solche Modelle besonders für die Entwicklung von konsistenten und sicheren Systemen geeignet.

Die formalen und semiformalen Sicherheitsmodelle entstehen im Rahmen der ausführlichen Risiko- und Gefahrenanalysen eines Anwendungsszenarios oder Systems, woraus sich die gewünschten Anforderungen an die sicherheitsrelevan-

ten Funktionalität und Eigenschaften ergeben. Im Rahmen eines Spezifizierungsprozesses werden Sicherheitsmodelle erstellt, welche die zwei grundlegende Modellierungsebenen Sicherheitsmerkmale und Sicherheitseigenschaften besitzen. Dabei können Sicherheitsmodelle das Ergebnis einer Neuentwicklung, einer Instantiierung auf Basis eines generischen Sicherheitsmodells oder einer Komposition im Rahmen des Model-Engineerings sein. Die generischen Modelle sind dabei eine besondere Gruppe von Sicherheitsmodellen. Manche Definitionen dieser Modelle sind so variabel definiert, dass sie erst im Rahmen einer Instantiierung zu einem konkreten Modell werden. Dies erlaubt eine systematische Analyse solcher Sicherheitsmodelle, ohne dabei einen Bezug auf konkrete Szenarien bzw. Systeme zu nehmen.

Um die Sicherheitsmodelle besser analysieren zu können, wurde ein Klassifizierungsschema erstellt. Dabei wurden Modellklassen gebildet, deren Zusammensetzung durch die Sicherheitsaspekte und Modellierungskonzepte geprägt ist. Für eine bessere Vergleichbarkeit wurde ein Zustandsübergangssystem als Grundlage für die Beschreibung der Sicherheitsmodellen gewählt. Des Weiteren konnten sie im Groben in drei Modellklassen unterteilt werden: Zugriffskontrollmodelle, Informationsflussmodelle und Transaktionsmodelle.

Im Mittelpunkt der Zugriffskontrollmodelle steht der Schutz von Ressourcen eines Systems, welcher mittels der Überwachung aller Zugriffe von Subjekten geregelt wird. Ein typisches Beispiel der Zugriffskontrollmodelle ist das Harrison-Ruzzo-Ullman-Modell (Abschnitt 4.1), welches Zugriffsschutz mittels diskreter Zugriffsregeln umsetzt. Dies bedeutet, dass die Regeln pro Subjekt festgelegt werden.

Die Kernaspekte der Flusskontrollmodelle sind die Überwachung und der Transport von Information innerhalb eines geschlossenen Systems. Dabei soll mit den Informationsflussregeln verhindert werden, dass Informationen zu den unautorisierten Subjekten gelangen. Eines der Sicherheitsmodelle aus dieser Modellklasse ist das Denning-Modell (Abschnitt 4.6), welches anhand der global ausgelegten Flussentscheidungen die Informationsflüsse regelt.

Die Transaktionsmodelle beschäftigten sich mit der Sicherheit innerhalb offener Kommunikationsnetze. Dabei steht der Schutz von kommunizierenden Partner und Transfergütern im Vordergrund. Transaktionsmodellen verwenden daher das Konzept der parallel arbeitenden Automaten. Ein Vertreter der Klasse ist das Gleichgewichtsmodell (Abschnitt 4.7), welches die Verpflichtungen und

Nachweise zwischen kooperierenden Parteien im Gleichgewicht halten soll.

Da das Klassifizierungsschema lediglich einen indirekten Vergleich der Sicherheitsmodelle im Bezug auf deren Struktur und Konzepte bieten, wurden die ausgewählten Sicherheitsmodell in dem Kapitel 5 anhand unterschiedlicher Vergleichskriterien direkt miteinander verglichen. Die dabei verwendeten Kriterien sind zum Beispiel Formaliserungsgrad, Abstraktionsgrad, Sicherheitsziel und Skalierbarkeit.

In dem Kapitel 6 wurde alternativ zu der in dieser Arbeit eingesetzten Zustandsübergangssysteme die Prozessalgebra *CSP* verwendet, um exemplarisch einige Elemente des Harrison-Ruzzo-Ullman- und des Gleichgewichtsmodells zu beschreiben.

7.2 Ausblick

Das ausgearbeitete Klassifizierungschema bietet eine gute Grundlage, um die in dieser Arbeit betrachteten Sicherheitsmodelle klassifizieren zu können. Es kann aber in den weiterführenden Arbeiten verfeinert und erweitert werden, sodass auch weitere Sicherheitsmodelle klassifiziert werden können. Insbesondere die Informationsfluss- und Transaktionsmodelle bieten einen Raum für eine feingranularere Klassifizierung.

Des Weiteren bildet diese Arbeit eine Grundlage für die Betrachtung alternativer Darstellungsformen. Bei der Modellierung heutiger Systeme müssen viel mehr Aspekte betrachtet werden, als bei den klassischen Modellen zur Zugriffskontrolle. Ein solcher Aspekt ist beispielsweise die Parallelität und die damit zusammenhängende Nebenläufigkeit. Die Zustandsübergangssysteme sind zwar für Menschen intuitiv und besitzen eine breite Unterstützung durch computergestützten Tools. Dennoch stoßen sie bei der Modellierung von sehr komplexen bzw. parallelen Systemen an ihre Grenzen. Die Modellierung solcher Systemen führt in der Regel zu einer Zustandsexplosion, wobei die Modelle unübersichtlich und schwer analysierbar werden.

In dem Kapitel 6 wurde die Prozessalgebra *CSP* betrachten, die auf die Modellierung von Parallelität ausgerichtet ist. Aus den dabei modellierten Beispielen ist zu erkennen, dass sich bestimmte Szenarien wie die Verkaufsaktion damit gut modellieren lassen. Im Gegensatz dazu ist die Modellierung von einer Zugriffsmatrix sehr aufwendig. In den weiterführenden Arbeiten kann anhand dieser

Masterarbeit untersucht werden, inwieweit neben Zustandsübergangssystemen weitere Modellierungsmethoden für die Modellierung von Sicherheitsaspekten geeignet sind.

7.3 Fazit

Durch die systematische Analyse der veröffentlichen Sicherheitsmodelle konnte eine Klassifizierung gefunden werden, die strukturelle und konzeptuelle Besonderheiten der Sicherheitsmodelle zum Ausdruck bringt. Diese Klassifizierung dient als Grundlage für die Beschreibung und den Vergleich der einzelnen Sicherheitsmodelle. Der Vergleich hat ergeben, dass frühere Sicherheitsmodelle wie beispielsweise Harrison-Ruzzo-Ullman-Modell sich nur mit der elementaren Modellierung von Systemen beschäftigten, wobei die theoretischen Aspekte im Vordergrund stehen. Mit den neuen akademischen Sicherheitsmodellen wie beispielsweise Schematic Protection Model wird versucht, eine bessere Lösung für die gleichen Probleme wie bei dem HRU-Modell zu finden. Allerdings werden bei diesen neueren Modellen weitere Konzepte wie Rollen, Attribute und komplexe Sicherheitsschemas verwendet, um die Modellierung von komplexen Systemen in der praxisorientierter Umgebung zu vereinfachen.

Literaturverzeichnis

[CW96] CLARKE, EDMUND M. ; WING, JEANNETTE M.: Formal Methods: State of the Art and Future Directions. In: *ACM Computing Surveys* 28 (1996)

[DD77] DENNING, Dorothy E. ; DENNING, Peter J.: Certification of Programs for Secure Information Flow. In: *Communications of the ACM* 20 (1977)

[Den76] DENNING, Dorothy E.: A Lattice Model of Secure Information Flow. In: *Communications of the ACM* 19 (1976)

[Den82] DENNING, Dorothy E.: *Cryprography and Data Security*. Addison Wesley Publishing Company, 1982

[Eck09] ECKERT, Claudia: *IT-Sicherheit Konzepte-Verfahren-Protokolle*. 6. Oldenburg Verlag München, 2009

[FK07] FERRAIOLO, David F. ; KUHN, D. R. ; EDITION, Second (Hrsg.): *Role-Based Access Control*. ARTECH HOUSE, INC., 2007

[Fox98] FOX, Stefanie: *Spezifikation und Verifikation eines Separation of Duty-Szenarios als eine Verbindliche Telekooperation im Sinne des Gleichgewichtsmodells*, Johann Wolfgang Goethe-Universität, Diplomarbeit, 1998

[GM82] GOGUEN, J. A. ; MESEGUER, J.: Security Policies and Security Models. In: *IEEE Symposium on Security and Privacy'82* (1982), S. 10–20

[GP] GRIMM, Rüdiger ; PÄHLER, Daniel: A formal Digital Rights Model without Enforcement / Institute for IS Research, University of Koblenz-Landau. – Forschungsbericht

[Gri93] GRIMM, Rüdiger: Non-Repudiation In Open Telecooperation. In: *16th National Computer Security Conference*, 1993

[Gri08] GRIMM, Rüdiger: *IT-Sicherheitsmodelle*. Arbeitsberichte aus dem Fachbereich Informatik, 2008

[Gri09] GRIMM, Rüdiger: A Formal IT-Security Model for a Weak Fair-Exchange Cooperation with Non-Repudiation Proofs. In: *SECUWARE 2009, IARA, IEEE CS Press* (2009)

[HF06] HABRIAS, Henri ; FRAPPIER, Marc: *Software Specification Methods*. ISTE LtD, 2006

[Hil90] HILL, Mark D.: What is scalability? In: *ACM SIGARCH Computer Architecture News* 18 (1990), S. 18–21

[Hoa04] HOARE, C. A. R.: Communicating Sequential Processes. (2004), S. 260

[HRU76] HARRISON, Michael A. ; RUZZO, Walter L. ; ULLMAN, Jeffrey D.: Protection in Operating Systems. In: *Communications of the ACM* 19 (1976), S. 8

[KM93] KESSLER, V. ; MUND, S.: Sicherheitsmodelle / Siemens. 1993. – Forschungsbericht

[KP11] KÜHNHAUSER, Winfried E. ; PÖLCK, Anja: Towards Access Control Model Engineering. In: *Information Systems Security, LNCS 7093* (2011)

[McL90] MCLEAN, John: Security Models and Information Flow / Center for High Assurance Computer Systems Naval Research Laboratory. 1990. – Forschungsbericht

[Min84] MINSKY, Naftaly H.: Selective and Locally Controlled Transport of Privileges. In: *ACM Transactions on Programming Languages and Systems* 6 (1984), S. 573–602

[Mod12] Version: 2012. http://wirtschaftslexikon.gabler.de/Definition/modell.html, Abruf: 18.01.2013

[MSUV02] MANTEL, Heiko ; STEPHAN, Werner ; ULLMANN, Markus ; VOGT, Roland: Leitfaden für die Erstellung und Prüfung formaler Sicherheitsmodelle im Rahmen von ITSEC und Common Criteria. In: *Bundesamt für Sicherheit in der Informationstechnik* (2002), S. 105

[PDMP] PRIEBE, Torsten ; DOBMEIER, Wolfgang ; MUSCHALL, Björn ; PERNUL, Günther: ABAC - Ein Referenzmodell für attributbasierte Zugriffskontrolle.

[PW08] PRIESE, Lutz ; WIMMEL, Harro: *Petri-Netze*. Springer-Verlag Berlin Heidelberg, 2008

[San88] SANDHU, Ravinderpal S.: The Schematic Protection Model: Its Definition and Analysis for Acyclic Attenuating Schemes. In: *Journal of the Association for Computing Machinery* (1988), S. 404–432

[San92] SANDHU, Ravinderpal S.: The Typed Access Matrix Model. In: *Proceedings of IEEE Symposium on Security and Privacy, Oakland, California* (1992), S. 122–136

[San93] SANDHU, Ravinderpal S.: Lattice-Based Access Control Models. In: *IEEE Computer* 26 (1993), S. 9–19

[Sch99] SCHIER, Kathrin: *Vertrauenswürdige Kommunikation im elektronischen Zahlungsverkehr. Ein formales Rollen- und Aufgabenbasiertes Sicherheitsmodell für Anwendungen mit multifunktionalen Chipkarten.*, Universität Hamburg, Doktorarbeit, 1999

[Sch07] SCHWARZ, Hannes: Formalisierungsgrad. (2007)

[Sec12] Version: 2012. https://www.tu-ilmenau.de/vsbs/lehre/ss-2012/security-engineering/, Abruf: 18.01.2013

[SP04] SANDHU, RAVI ; PARK, JAEHONG: The UCON(ABC) Usage Control Model. In: *ACM Transactions on Information and System Security* (2004)

[uml07] Version: 2007. http://www.omg.org/spec/UML/2.1.2/Superstructure/, Abruf: 20.05.2013

[ZLN05] ZHANG, Xinwen ; LI, Yingjiu ; NALLA, Divya: An Attribute-Based Access Matrix Model. In: *Proceeding SAC '05 Proceedings of the 2005 ACM symposium on Applied computing* (2005), S. 359–363