[RKSt02] *Christoph Rensing, Martin Karsten, Ralf Steinmetz;* **Darstellung der Sicherheitsattribute von Kommunikationsbeziehungen mittels UML;** Enterprise Security, Klagenfurt, März 2002, S. 132-143.

Darstellung der Sicherheitsattribute von Kommunikationsbeziehungen mittels UML

Christoph Rensing · Martin Karsten · Ralf Steinmetz

Technische Universität Darmstadt
{christoph.rensing | martin.karsten | ralf.steinmetz} @kom.tu-darmstadt.de

Zusammenfassung

Die Sicherheit von verteilten Systemen besteht aus der Systemsicherheit der einzelnen Komponenten des verteilten Systems und der Kommunikationssicherheit zwischen den Komponenten. Mittels eines Security-Engineering soll der Grad der Sicherheit erhöht werden. Dabei umfasst Security-Engineering den gesamten Lebenszyklus von Systemen einschließlich des Entwurfes. Der Systementwurf wird verbreitet durch geeignete Hilfsmittel unterstützt. Weitverbreitet ist die Modellierung mit Hilfe von UML. Dieser Beitrag beschreibt ein neues Verfahren zur Darstellung von Sicherheitsattributen für Kommunikationsbeziehungen in verteilten Systemen in UML Diagrammen. Dazu werden*Association Classes* verwendet, welche die notwendigen Sicherheitsdienste spezifizieren, und weiterhin OCL konforme Anschriften, mit welcher die Methoden zur Realisierung der Sicherheitsdienste beschrieben werden können. Die Darstellung des Verfahrens erfolgt am Beispiel des RADIUS Protokolls.

1 Einführung

1.1 Motivation und Zielsetzung

Verteilte Systeme sind aufgrund ihrer Offenheit, Heterogenität und Komplexität besonderen Angriffsgefahren ausgesetzt. Neben der Gewährleistung der Funktionalität in verteilten Systemen, besitzt daher die Sicherheit der Systeme besondere Bedeutung für deren Nutzen. Die Sicherheit verteilter Systeme kombiniert sich aus der Sicherheit der einzelnen rechnerbasierten Komponenten des verteilten Systems (Systemsicherheit) und der Kommunikationskanäle zwischen den Komponenten (Kommunikationssicherheit). Um System- und Kommunikationssicherheit zu gewährleisten ist der gesamte Lebenszyklus des verteilten Systems vom Entwurf über die Entwicklung bis zum Betrieb zu betrachten. Ein solch umfassender Ansatz wird häufig als Security-Engineering bezeichnet.

Derzeit wird ein Security-Engineering nur selten konsequent verfolgt. Verteilte Systeme werden zumeist von Software Entwicklern ohne spezifische Sicherheitskenntnisse und ohne Berücksichtigung der Sicherheitsanforderungen entworfen und implementiert. Vielmehr werden meist erst zum Ende des Softwareentwicklungsprozesses, wenn überhaupt, entsprechende Sicherheitsspezialisten hinzugezogen und die Systeme um Sicherheitsmechanismen ergänzt [SR01]. Für ein Security-Engineering ist es aber notwendig, bereits im Entwurf die Sicherheitsanforderungen verteilter Systeme zu betrachten. Dazu müssen Modellierungsmethoden, die im Entwurf Verwendung finden, die Darstellung von Sicherheitsattributen erlauben. Solche Darstellungsmethoden fehlen zumeist. Ziel der Arbeit ist es daher, eine Technik zur Dar-

stellung von Attributen der Kommunikationssicherheit in einer weitverbreiteten Modellierungsmethode zu definieren.

1.2 Aufbau des Beitrags

In Kapitel 2 werden Anforderung an die im Entwurf zu verwendende Modellierungsmethode unter dem Gesichtspunkt der Unterstützung eines Security-Engineering definiert, existierende Methoden klassifiziert und an Hand der Anforderungen beurteilt. Kapitel 3 beschreibt am Beispiel des RADIUS Protokolls die Modellierung von verteilten Systeme in UML und die Methode mittels derer die UML - Darstellungsformen für die Modellierung von Attributen der Kommunikationssicherheit eingesetzt werden können. In Kapitel 4 definieren wir auf Basis unserer Erfahrungen mit der beschriebenen Technik ein Vorgehensmodell für den Entwurf sicherer verteilter Systeme mittels UML. Der Beitrag endet mit einer Zusammenfassung und einem Ausblick.

2 Modellierungsmethoden für verteilte Systeme

2.1 Anforderungen

Systementwurf wird definiert als die Erstellung einer Systemarchitektur und die Spezifikation von Systemkomponenten unter Einsatz geeigneter Methoden, Konzepte und Werkzeuge. Zur Beschreibung von Systemen werden in der Regel Modelle genutzt [Bal00]. Für die Modellierung von verteilten Systemen existieren verschiedene Methoden, die im folgenden Abschnitt näher betrachtet werden. Zunächst sollen die Anforderungen definiert werden, die an die Modellierungsmethode unter dem Gesichtspunkt eines Security-Engineering zu stellen sind.

Ein verteiltes System besteht aus Diensterbringern und Dienstnutzern. Diese lassen sich abstrakt durch Rollen beschreiben. Dabei repräsentiert eine Rolle eine Gruppe von Funktionen, die es dem Rollenträger ermöglicht, bestimmte Dienste für seine Umwelt bereitzustellen. Eine Rolle im Modell ist losgelöst von real existierenden Akteuren. Einerseits kann ein Akteur mehrere verschiedene Rollen vertreten, z.B. Dienstnutzer und Dienstanbieter zugleich. Andererseits können mehrere Akteure die gleiche Rolle annehmen. Z.B. existiert die Rolle *Nutzer* nur einmal, es können jedoch beliebig viele reale Personen als *Nutzer* auftreten.

Eine Modellierungsmethode für verteilte Systeme muss daher die Darstellung folgender verschiedener Komponenten unterstützen:

1. **Rollenträger:**
   Rollenträger, die sich über ihre Eigenschaften und ihr Verhalten charakterisieren lassen, sind die Elemente eines verteilten Systems.
2. **Kommunikationsbeziehungen:**
   Die Elemente des verteilten Systems, die Rollenträger, stehen über einen Datenaustausch zueinander in Beziehung und erbringen somit die Gesamtfunktionalität des verteilten Systems.

Für ein Security-Engineering sind Kommunikations- und Systemsicherheit zu betrachten und somit müssen die Sicherheitsattribute der Kommunikationsbeziehungen und der Rollenträger beschrieben werden können:

3. **Sicherheitsdienste der Kommunikationsbeziehungen:**
   Sicherheitsdienste repräsentieren die Sicherheitsanforderungen die mittels geeigneter Mechanismen zu gewährleisten sind. Für Kommunikationssicherheit sind dies Authentifizierung der Kommunikationspartner bzw. einer Datenquelle, Zugriffskontrolle, Vertraulichkeit von Daten, Integrität von Daten und Dienstannahme (Non-Repudiation).
4. **Sicherheitsmechanismen und Protokolle der Kommunikationsbeziehungen:**
   Die Sicherheitsdienste der Kommunikationsbeziehungen werden mittels verschiedener Mechanismen, wie beispielsweise Verschlüsselung, digitaler Signaturen oder anderer kryptographischer Mechanismen realisiert. Für eine sichere Kommunikation werden dann entsprechende Protokolle verwendet.
5. **Sicherheitsdienste der Rollenträger / Systeme:**
   Die einzelnen Elemente innerhalb eines einzelnen Systems, die in der Regel rechnerbasiert sind, verfügen ebenfalls über Sicherheitsanforderungen. Diese Sicherheitsdienste müssen ebenfalls modelliert werden.
6. **Sicherheitsmechanismen der Rollenträger / Systeme:**
   Gleiches gilt für die Mechanismen mit denen die Sicherheitsdienste realisiert werden.

Zusätzlich zu diesen Anforderungen sollte die Modellierungsmethode eine graphische Beschreibung unterstützen, da eine solche ein höheres Maß an Modellverständnis auch für Nicht-Spezialisten bietet.

2.2 Klassifikation und Beurteilung

Für die Modellierung von Systemen und insbesondere verteilten Systemen sind in den vergangenen Jahren eine Vielzahl verschiedener Methoden entwickelt worden, vgl. Abbildung 1. Sie lassen sich einteilen in formale und semiformale Methoden. Formale Methoden erlauben eine algorithmische oder mathematische Bearbeitung der Modelle und somit auch eine formal beweisbare Sicherheitsprüfung. Dazu muss das Modell korrekt das reale System abbilden. Eine solche hundertprozentig korrekte Abbildung ist bei komplexen Systemen sehr schwer durchzuführen. Die formalen Methoden werden wiederum unterteilt in logik- und zustandsbasierte Methoden. Formale logikbasierte Modellierungsmethoden, wie beispielsweise die Sprache Z [Toy00], sind grundsätzlich dadurch gekennzeichnet, dass sie Systeme nicht anhand ihrer Struktur beschreiben, sondern durch Definition ihrer Eigenschaften. Daher sind sie für den Entwurf kaum nutzbar.

Im Unterschied zu den logikbasierten Methoden beschreiben die zustands- und prozessorientierten formalen Methoden die Systemstruktur und stellen Prozessabläufe dar. Ein gemeinsames Merkmal ist die Nebenläufigkeit, d.h. es können mehrere Prozesse unabhängig voneinander parallel ablaufen, die miteinander kommunizieren. Synchrone Konzepte, wie Lotos [ISO8807] oder CSP [Hoa78], erfordern die Vereinbarung fester Zeitpunkte, zu denen Daten ausgetauscht werden können. Asynchrone Konzepte, beispielsweise SDL [Kai97] oder Petri- und Produktnetze [OP95], erlauben die Prozesskommunikation über das Senden und Empfangen von Ereignissen. Sie eignen sich daher besonders für die Modellierung von verteilten Systemen im Rahmen des Entwurfs.

Zu den semiformalen Methoden zählen sämtliche Methoden der Daten-, Funktions- und Objektmodellierung, wie etwa Entity Relationship Modelle (ERM) oder die Unified Modeling

Language (UML). Sie sind gekennzeichnet durch definierte graphische Modellelemente, besitzen zusätzlich normalsprachliche Komponenten. Daher sind eine automatisierte Analyse, Konsistenztests oder beweiskräftige Sicherheitsanalysen mit ihnen nicht möglich. Es handelt sich dabei um Methoden, die speziell für den Entwurf entwickelt wurden.

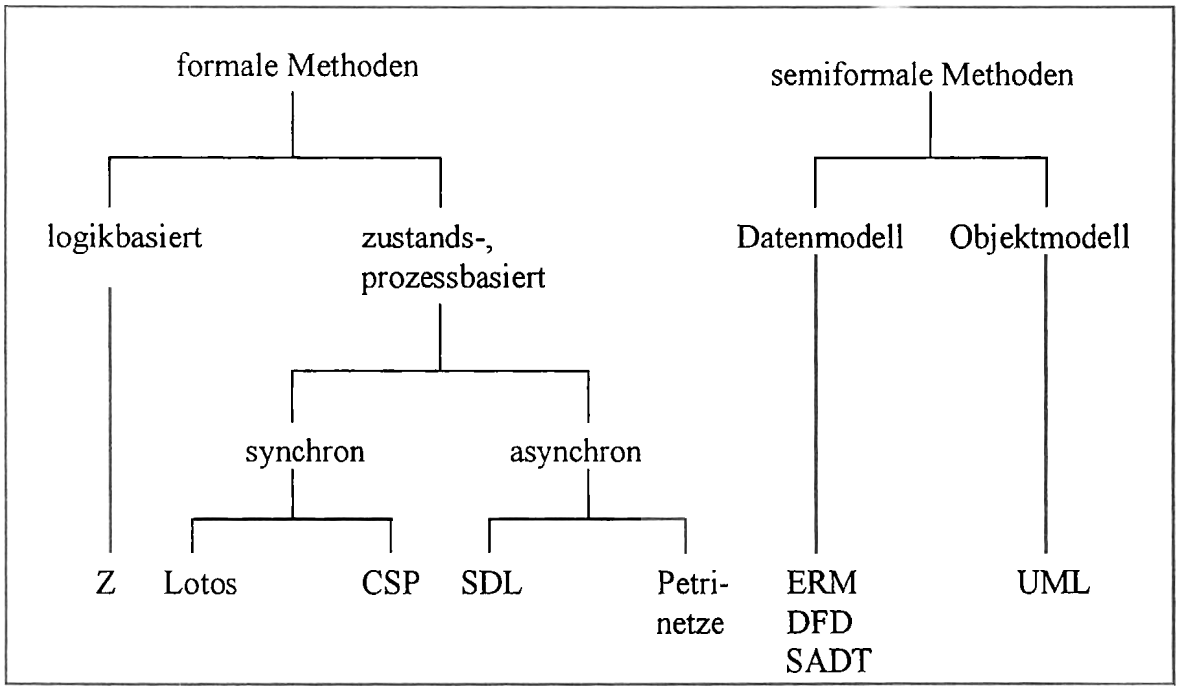

Abb. 1: Klassifikation von Modellierungsmethoden

Beurteilt man die verschiedenen Alternativen, so gilt für alle, dass der Modellierungsumfang die Darstellung der Rollenträger und Kommunikationsbeziehungen grundsätzlich erlaubt. Allerdings gilt für die formalen Methoden übereinstimmend, dass sie sich aufgrund ihrer Komplexität nur für begrenzte Systeme einsetzen lassen und sehr guter Methodenkenntnisse bedürfen, um korrekte Modelle zu bilden. Die semiformalen Methoden sind aufgrund ihrer relativen Einfachheit und Anschaulichkeit weitverbreitet. ERM werden z.B. für den Entwurf von Datenbanksystemen und UML allgemein in der Softwareentwicklung vielfach verwendet. Es bietet sich daher an, solche semiformalen Methoden weiter zu betrachten. Dazu ist es aber notwendig, dass die Sicherheitsattribute der Kommunikationsbeziehungen und Systeme (Anforderungen 3. bis 6.) geeignet darstellbar sind. Dazu existieren bisher allerdings nur sehr wenige Arbeiten.

3 Modellierung von Sicherheitsattributen

3.1 UML am Beispiel Unified Modelling Language

Die objektorientierte Modellierung hat seit Anfang der 90er Jahre zunehmend an Bedeutung gewonnen. Eine Sonderstellung in der objektorientierten Entwicklung besitzt die Unified Modeling Language (UML). Sie fasst die Vorzüge verschiedener objektorientierten Methoden zusammen. Die UML wurde 1997 erstmals vorgestellt, ist seit 1999 in ihrer Version 1.3 von

der OMG (Object Management Group) standardisiert [OMG99] und wird heute als Standard breit verwendet.

UML definiert eine Vielzahl verschiedener Darstellungsformen, die semantisch genau beschrieben sind und unterschiedliche Sichten (Darstellungsarten) auf dasselbe System erlauben. Zur Modellierung eines verteilten Systems lassen sich insbesondere folgende Diagrammtypen gut verwenden, wobei an dieser Stelle nicht detailliert auf die Modellierungstechniken von UML eingegangen werden kann:

- Mittels Klassendiagrammen können Rollenträger und Kommunikationsbeziehungen mit ihren Eigenschaften modelliert werden.
- Zustandsdiagramme sind geeignet, das Verhalten von Rollenträgern zu spezifizieren.
- Aktivitätsdiagramme oder Interaktionsdiagramme können den Kommunikationsablauf des gesamten verteilten Systems im Überblick darstellen.

UML kann aufgrund der angebotenen Darstellungsvielfalt sehr gut auch zur Spezifikation und zum Design verteilter Systeme genutzt werden, was wir an einem Beispiel im folgenden Abschnitt zeigen werden. Dabei beschränken wir uns auf die Darstellung in Form von Klassendiagrammen, weil wir diese für die Modellierung der Sicherheitsattribute der Kommunikationsbeziehungen verwenden.

3.2 UML-Modellierung am Beispiel RADIUS

Als Beispiel für die Modellierung haben wir das RADIUS Protokoll [RWR96] und die zugehörige Architektur gewählt, da RADIUS weithin bekannt und nicht zu umfangreich ist, um im Rahmen dieses Beitrages dargestellt zu werden. Nachfolgend beschrieben wir die Komponenten der RADIUS-Architektur und das RADIUS-Protokoll selbst, soweit diese für das Verständnis der Modelle notwendig sind.

- Der RADIUS Client stellt die Schnittstelle zum User (Dial-In User) dar, der sich authentifizieren muss, bevor er einen Netzwerkdienst nutzen kann. Er wird deshalb als *Network Access Service* (NAS) bezeichnet. Seine Aufgabe ist die Weiterreichung der vom User gesendeten Daten zum Login an den Server und die Konfiguration der angebotenen Netzwerkdienste nach erfolgreicher Authentifizierung.

  Der RADIUS Server führt die Authentifizierung von Usern durch und sendet bei Erfolg Daten zur Konfiguration des gewünschten Diensts zurück an den Client. Dazu muss er die zur Überprüfung der User-Authentifizierung notwendigen Daten speichern oder an anderer Stelle auf sie zugreifen können.

Client und Server kommunizieren über das RADIUS-Protokoll, welches folgende Sicherheitsmethoden definiert:

- Alle zwischen Client und Server ausgetauschten Daten werden authentifiziert. Client und Server besitzen ein gemeinsames geheimes Passwort. Die gesendeten Pakete werden um ein Message Authentication Code (MAC) der mittels dieses Passworts verschlüsselt wird ergänzt. Der MAC kann vom Kommunikationspartner unter Nutzung des gleichen Passworts verifiziert werden.

Nutzerpasswörter werden verschlüsselt übertragen.

Die Basisklassen des RADIUS Systems sind in Abbildung 2 dargestellt.

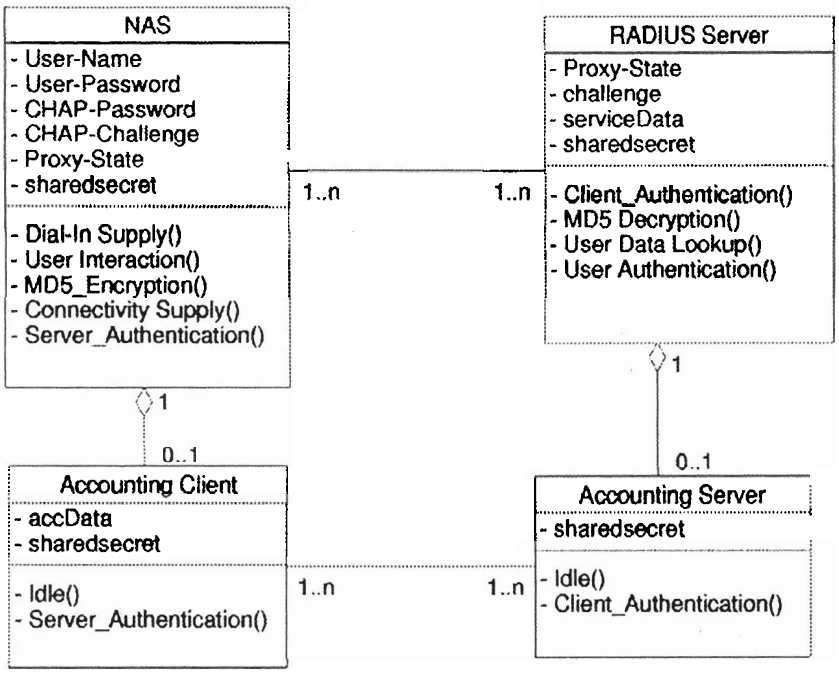

Abb. 2: RADIUS Basisklassenmodell

Zusätzlich zu diesen Basisklassen existiert eine Klasse USER, welche den Dial-In User repräsentiert. Da RADIUS eine Authentifizierung mittels PAP und CHAP und das RADIUS Challenge-Response Verfahren unterstützt, besitzt die Superklasse User drei Subklassen, wie in Abbildung 3 gezeigt.

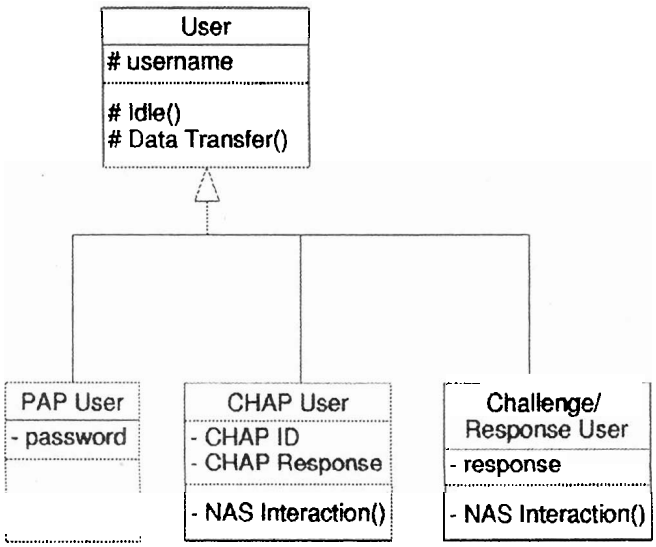

Abb. 3: RADIUS User Klassenmodell

In den obigen Klassenmodellen werden die Sicherheitsattribute der Kommunikationsbeziehungen bisher nicht dargestellt. Das ist aber für ein Security-Engineering wesentlich und wird daher im folgenden Abschnitt betrachtet.

3.3 Sicherheitsattribute der Kommunikationsbeziehungen

Die Sicherheitsattribute der Kommunikationsbeziehungen, vgl. Anforderung 3 und 4 in Abschnitt 2.1, sind Bestandteile der Beziehung zwischen den Rollenträgern. In UML Klassenmodellen werden diese Beziehungen als Assoziationen zwischen Klassen bzw. Instanzen der Klassen modelliert. Diese Assoziationen müssen mit ihren Attributen genauer charakterisiert werden können, d.h. die Assoziationen im UML-Klassendiagramm, die Kommunikationsbeziehungen repräsentieren, sind um zusätzliche Modellelemente zu erweitern.

In der UML gibt es mehrere Erweiterungsmöglichkeiten zur Verfeinerung der grafischen Darstellung:

- Normalsprachliche Anmerkungen in Form von Notizen, so genannte Notizelemente, können beliebig in UML-Diagrammen eingesetzt werden, um Zusatzinformationen zu ergänzen, die nicht mit Hilfe der UML auszudrücken sind.
- Die UML-Spezifikation sieht eine Möglichkeit vor, durch sogenannte *Constraints* Wertebereiche von Klassenattributen zu definieren. Zu diesem Zweck wurde eine formale, die UML erweiternde Sprache entwickelt, die sogenannte *Object Constraint Language* (OCL) [IBM00]. Constraints müssen jedoch nicht in OCL definiert sein, jede beliebige Sprache ist möglich. Sie werden dargestellt als Anschriften in geschweiften Klammern an Objekten oder Relationen. Zur Darstellung längerer Constraints dürfen Notizelemente verwendet werden [WAH98][SS99].
- Sogenannte *Association Classes* können Eigenschaften von Relationen beschreiben. In [OMG99] wird dieses Modellelement folgendermaßen definiert: "An association class is an association that is also a class. It not only connects a set of classifiers but also defines a set of features that belong to the relationship itself and not any of the classifiers". Eine Association Class wird als Klasse dargestellt, die über eine gestrichelte Linie mit der zugehörigen Assoziation verbunden ist [HOL99].

Jede der genannten Erweiterungsmöglichkeiten befasst sich mit einem anderen Darstellungsaspekt:

- Notizelemente lassen die größten Freiheiten zu, erlauben jedoch keine formale oder semiformale Modellierung. Durch sie wird eine ergänzende normalsprachliche Beschreibung des Modells in das Klassendiagramm aufgenommen.
- Die Ausdrucksmöglichkeiten der Object Constraint Language beziehen sich auf schon vorhandene Attribute und Operationen von Klassen oder auf Relationen. Voraussetzung für ihren Einsatz ist also die Existenz von Attributen, die näher beschrieben werden können.
- Association Classes können Attribute einer Assoziation beschreiben, jedoch gemäß UML-Spezifikation nicht deren Werte bzw. Ausprägungen.

Für die Darstellung von Sicherheitsattributen schlagen wir daher eine Kombination dieser Spracherweiterungen vor. Relationen, denen Kommunikationsbeziehungen zugrunde liegen, werden jeweils um eine Association Class mit dem Bezeichner *Trust Relationship* ergänzt. Die Association Class bezeichnet die notwendigen Sicherheitsdienste in Form der Attribute *integrity*, *confidentiality*, *authenticity*, *non-repudiation*. Es werden nur diejenigen Eigenschaften in die Klasse aufgenommen, die von der jeweiligen der Assoziation zugrunde liegenden Kommunikationsprotokoll und entsprechenden Sicherheitsmechanismen gewährleistet werden. Im RADIUS-Beispiel wird beispielsweise die Eigenschaft Vertraulichkeit zwischen NAS und RADIUS Server durch das Verschlüsseln der Passwörter realisiert, vgl. *Trust Relationship 1* in Abbildung 4. Die Authentizität wird durch die gegenseitige Authentifizierung von RADIUS Client und RADIUS Server sichergestellt. Mit Hilfe dieser Association Class kann während der Spezifikation einfach beschrieben werden, welche Sicherheitsdienste für die jeweilige Assoziation realisiert werden müssen.

In einem zweiten Schritt ist es wünschenswert eine präzisere formale Spezifikation der Attribute zu modellieren, indem dargestellt wird, welche Sicherheitsmethoden zur Gewährleistung der Sicherheitsdienste genutzt werden. Diese Aufgabe sollte im Entwurf ein Sicherheitsexperte übernehmen, der ebenso die Vollständigkeit und Korrektheit der modellierten Sicherheitsdienste überprüfen sollte, vgl. dazu auch Kapitel 4. Für eine solche genauere Spezifikation eignen sich Elemente der OCL. Die Sicherheitsmethoden werden durch *Constraint-Anschriften* für die Association Classes modelliert. Dafür gelten folgende Modellierungsvorschriften:

- Jede Eigenschaft der Association Class *Trust Relationship* kann entweder erfüllt oder nicht erfüllt sein, das zugehörige Attribut ist eine boolesche Variable. In der graphischen Darstellung werden nur die Attribute mit dem Wert *true* angeschrieben.
- Die Operationen der jeweiligen Klasse, die für die Erfüllung oder Nichterfüllung einer Eigenschaft verantwortlich sind, liefern einen booleschen Rückgabewert (Beispiel: Bei erfolgreicher Authentifizierung eines Client-Pakets durch den RADIUS Server gibt die Operation *Client_Authentication* den Wert *true* zurück).
- Der Wert jedes Eigenschafts-Attributs bestimmt sich aus einem logischen Ausdruck, welcher die für diese Eigenschaft ursächlichen Operationen beinhaltet (Beispiel: Authentizität ist ein Resultat erfolgreicher Authentifizierung).

Um die in der 3. Vorschrift geforderte Zuordnung ausdrücken zu können, ist es erforderlich, einem Attribut einer Association Class eine oder mehrere Operationen zuzuweisen, die sich in den an der Assoziation angrenzenden Klassen befinden. Dies ist möglich mit einem in der UML-Spezifikation beschriebenen Sprachelement der OCL und wird in [OMG99] Kapitel 7.5 als *Navigation from Association Classes* bezeichnet. Der Zugriff auf eine Operation erfolgt bei der dort definierten Syntax in der Form

Rolle.Operation,

wobei *Rolle* für die Rolle steht, die eine Klasse in einer Beziehung zu einer anderen Klasse innehat. Die Rolle *self* bezieht sich in diesem Fall auf die Association Class selbst. Rollenbezeichner werden im UML-Klassendiagramm als Anschriebe an den Assoziationsenden dargestellt (vgl. Rollen *client* und *server* in Abbildung 4). Mögliche Varianten einer Zuordnung von Eigenschaft und Ursache lassen sich damit in folgender Syntax darstellen:

self.Attribut = Rolle.Operation,
self.Attribut = (Rolle1.Operation_a) and (Rolle2.Operation_b).

Die Zuordnung lässt sich beliebig komplex gestalten, indem die rechte Seite der Gleichung um logische Verknüpfungsoperatoren ergänzt wird. Bei längeren Ausdrücken empfiehlt es sich, die Spezifikationen in Notizelemente zu fassen und an die Association Classes anzuhängen anstatt Anschriften in geschweiften Klammern zu verwenden.

Diese Darstellungsvariante wurde für die folgende Abbildung gewählt, welche die Sicherheitsdienste und -mechanismen der RADIUS Kommunikationsbeziehungen integriert in die zuvor modellierten Klassen darstellt.

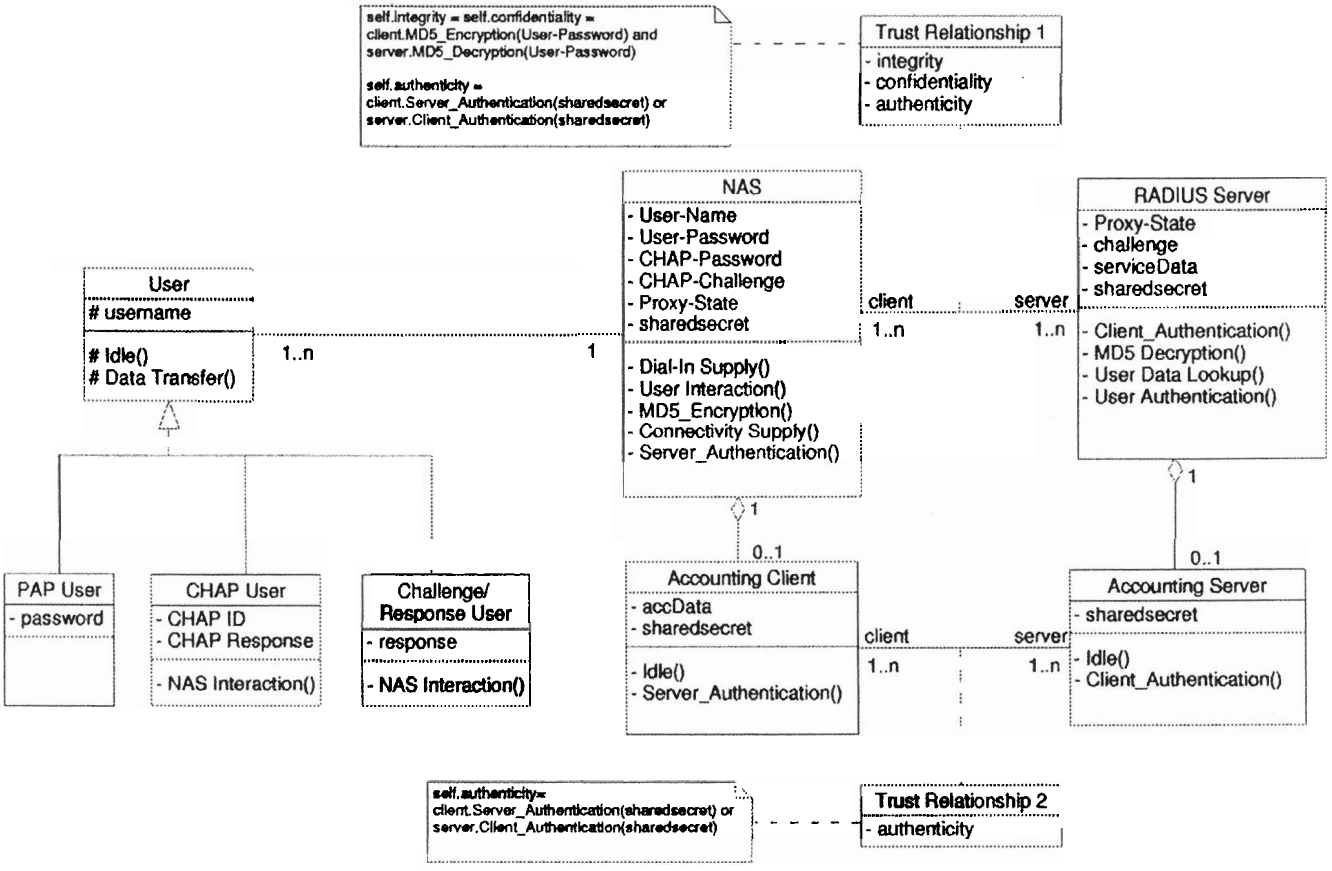

Abb. 4: RADIUS Gesamtmodell

Den Attributen der Association Classes und den zugehörigen formalen OCL-Anschriften in Abbildung 4 entnimmt man, dass das Vertrauensverhältnis zwischen NAS und RADIUS Server die Sicherheitsdienste Integrität und Vertraulichkeit bezüglich der Übermittlung des Nutzer-Passworts aufweist, da dieses verschlüsselt übertragen wird. Zusätzlich wird Authentizität der übertragenen Pakete durch die Operationen *Client_Authentication* und *Server_Authentication* garantiert, welche das gemeinsame Passwort *sharedsecret* nutzen. Das Vertrauensverhältnis zwischen Accounting-Client und Accounting-Server dagegen weist ausschließlich das Attribut *authenticity* auf, da zur Erfüllung der übrigen Attribute in [RIG00] keine weiteren Operationen spezifiziert werden.

4 Vorgehensmodell – Security-Engineering mit UML

Für ein Security-Engineering verteilter Systeme mittels UML hat sich in unseren weiteren Arbeiten, die wir an dieser Stelle nicht detailliert beschreiben können, ein dreistufiges Vorgehen bewährt:

Modellierung der Kommunikation zwischen Rollenträgern mit Hilfe von Aktivitätsdiagrammen und Definition der dazu notwendige Operationen.

Modellierung der Rollenträger in einem Klassendiagramm und Zuweisung der sie beschreibenden Attribute und Operationen.

- Identifikation von Vertrauensverhältnissen und Sicherheitseigenschaften der Kommunikation und deren Modellierung im Klassendiagramm.

In den ersten beiden Schritten wird die Funktionalität des Systems dargestellt. Sie können herkömmlich durch einen Softwaredesigner als Funktionsexperte ausgeführt werden. Im dritten Schritt muss dann ein Sicherheitsexperte eingebunden werden. Dieser kann auf einem bestehenden und, aufgrund der graphischen Darstellung, leichter verständlichen Modell aufsetzen und muss nicht selbst die gesamte Funktionalität modellieren. Das bestehende Modell ergänzt er gemeinsam mit dem Softwaredesigner in der in Kapitel 3 beschriebenen Art um die Sicherheitsattribute der Kommunikationsbeziehungen. Dieses Vorgehen bietet den Vorteil, dass für die Modellierung kein Methodenwechsel notwendig ist und somit keine Informationen verloren gehen und Funktions- und Sicherheitsexperte auf einer gemeinsamen Basis diskutieren können.

Auch wenn eine formale Uberprüfung der Sicherheitsdienste auf Vollständigkeit und Korrektheit mittels UML nicht erfolgen kann, so bietet diese Methode doch weitere Vorteile. Die Übersichtlichkeit der UML-Diagramme unterstützt die Überprüfung der Sicherheitsdienste auf Vollständigkeit. Im beschriebenen Beispiel erkennt man in Abbildung 4 leicht, dass die Accounting-Daten zwischen Client und Server unverschlüsselt übertragen werden, da für die Übertragung der Daten zwischen User und NAS in der RADIUS Spezifikation keinerlei Sicherheitsmechanismen vorgesehen sind. Dort besteht also eine Sicherheitslücke.

Das im dritten Schritt entstandene erweiterte Klassenmodell gibt dem Entwickler zusätzlich konkrete Angaben, mit welchen Sicherheitsmechanismen die Kommunikationsverbindungen ausgestattet werden müssen und somit Implementierungshinweise. Damit ist es möglich, bereits im Entwurfsprozess ein Sicherheitsengineering zu beginnen und zudem die Grundlagen für ein weiteres Sicherheitsengineering in der Implementierungs- und Nutzungsphase zu schaffen.

5 Zusammenfassung und Ausblick

Für ein Security-Engineering ist die Darstellung von Sicherheitsattributen in Entwurfsmethoden notwendig. Bisher fehlte in UML eine Technik zur Darstellung von Sicherheitsattributen der Kommunikationsbeziehungen in verteilten Systemen. Einzig der Ansatz von Jürjens [Jür01] verfolgt dieses Ziel. Wir bieten mit diesem Beitrag eine alternative Technik an, die diese Lücke schließt, indem er eine neue, zum UML-Standard konforme Technik zur Darstellung von Attributen der Kommunikationssicherheit in UML Klassendiagrammen be-

schreibt. Mit Hilfe der Assoziation Classes und den formalen OCL Anschriften, sowie der von uns beschriebenen Vorschriften zu ihrer Verwendung, lassen sich sowohl die Sicherheitsdienste der Beziehungen beschreiben als auch die zu implementierenden Sicherheitsmechanismen spezifizieren. Für kleine und mittlere Systeme erscheint diese Methode geeignet. Ihr Nutzen in komplexen Systemen ist von uns noch nicht überprüft worden.

Die Sicherheitsattribute der Rollenträger bzw. der einzelnen Elemente des verteilten Systems, d.h. die Aspekte der Systemsicherheit wurden nicht näher betrachtet. Zu deren Darstellung können sich aber zunächst grundsätzlich die Methoden- und Attributsbeschreibungen der UML Klassen eignen, die diese Elemente repräsentieren. Das Verhalten der einzelnen Komponenten eines verteilten Systems wird in UML zudem oft mit Hilfe von Zustandsdiagrammen beschrieben, vgl. Kapitel 3.1. Wir untersuchen daher derzeit die Übertragbarkeit von UML Zustandsdiagrammen in Produktnetze, einer Erweiterung von Petrinetzen, an Hand derer formale Beweise zum Nachweis der Sicherheitseigenschaften geführt werden können [OP95][GMD99]. Ähnliche Ansätze bestehen in den Arbeiten von Trcek, Eckert und Wirtz und Giese.

Trcek [TRC00] modelliert Sicherheits-Policies in verteilten Systemen mit Hilfe eins graphisches Modell basierend auf Datenflussdiagrammen, welches unter Verwendung der Sprache Z formalisiert werden kann. Eckert [ECK98] beschreibt einen umfassenden Top-Down-Ansatz zur Konzeption sicherer, verteilter Systeme, der die auf verschiedenen Abstraktionsebenen verfügbaren Methoden zusammenfasst. Sie verwendet ein rollenbasiertes Modellierungskonzept vor, in dem Systeme durch strukturierte Mengen von sogen. Komplexen beschrieben werden. Wirtz und Giese [WG00] stellen eine Methodik zur Spezifikation verteilter Softwaresysteme vor, die die Defizite der UML hinsichtlich Präzision und Möglichkeiten der Verhaltensbeschreibung von Systemen ausgleicht, indem die UML-Methoden um eine auf Petri-Netzen basierende Beschreibungsmethodik ergänzt.

Weiterhin überprüfen wir derzeit, inwieweit andere bekannte Methoden des Security-Engineering, wie Security Policies, Tree Representations, Evaluation Criteria oder Security Pattern [SR01] mit der Modellierung in UML verknüpft werden können.

Literatur

[Bal00] H. Balzert: Lehrbuch der Software-Technik, Band 1, Spektrum Akademischer Verlag, 2. Auflage, 2000.

[Bur97] R. Burkhardt: UML - Unified Modelling Language : Objektorientierte Modellierung für die Praxis, Addison-Wesley, 1997.

[Eck98] C. Eckert: Sichere, Verteilte Systeme - Konzepte, Modelle und Systemarchitekturen, Habilitationsschrift, Fakultät für Informatik, TU München, 1998.

[GMD99] GMD – Forschungszentrum Informationstechnik: SHVT Manual, Institut für Sichere Telekooperation, GMD Darmstadt, 1999.

[Hoa78] C.A.R. Hoare: Communicating Sequential Processes; in Communications of the ACM, Vol.21, No.8, 1978.

[Hol99] A.I. Holub: UML Reference Card, <URL:http://www.holub.com/>, 1999.

[IBM00] IBM: The Object Constraint Language, http://www-4.ibm.com/software/ad/standards/ocl.html, 2000.

[Jür01] J. Jürjens: Towards Secure Systems Development with UMLsec, Fundamental Approaches to Software Engineering (FASE/ETAPS) 2001, International Conference, Genoa April 2001.

[Kai97] M. Kaiser: Visualisierung von Protokollen am Beispiel des (simplifizierten) Kerberos Authentifikationsprotokolls, Studienarbeit, Fakultät für Informatik, Universität Stuttgart, 1997.

[OMG99] Object Management Group (OMG): Unified Modeling Language Specification, Version 1.3, http://www.omg.org/, 1999.

[OP95] P. Ochsenschläger, R. Prinoth: Modellierung verteilter Systeme: Konzeption, Formale Spezifikation und Verifikation mit Produktnetzen, Vieweg, 1995.

[Rig00] C. Rigney: RADIUS Accounting, Request For Comments:2866, Network Working Group, 2000.

[RWR96] C. Rigney, S. Willens, A. Rubens, W. Simpson: Remote Authentication Dial In User Service (RADIUS); Request For Comments: 2865, Network Working Group, 1996.

[SR01] M. Schumacher, U. Roedig: Security Engineering with Patterns; Pattern Languages of Programs (PLoP), 2001.

[SS99] J. Stocker, C. Schäffer:UML - Eine Einführung; Technologie-Zentrum Informatik, Universität Bremen, http://www.tzi.de/grp/flowtec/papers/doc/uml/html/uml.html, 1999.

[Toy00] I. Toy: Formal Specification - Z Notation - Syntax, Type and Semantics The Z Standards Panel, http://www.cs.york.ac.uk/~ian/zstan/, 2000.

[Trc00] D. Trcek: Security policy conceptual modeling and formalization for networked information systems, Computer Communications 23 (2000) 1716-1723, Elsevier Science B.V., 2000.

[Wah98] G. Wahl: UML Kompakt, OBJEKT Spektrum 2/1998, http://www.sigs.de/publications/docs/obsp/umlkompt/umlkompt.htm, 1998.

[WG00] G. Wirtz, H. Giese: Skalierbare Beschreibung des Verhaltens verteilter Softwaresysteme, AG Verteilte Systeme, Institut für Informatik, Universität Münster, http://wwwmath.uni-muenster.de/cs/u/versys/index.html, 2000.