Aus dem Institut für Medizinische Biometrie, Epidemiologie und
Informatik der Johannes Gutenberg-Universität Mainz

Konzepte und Komponenten für die Zugriffskontrolle in verteilten, heterogenen Krankenhaus-Informationssystemen am Beispiel des Mainzer Universitätsklinikums

Dissertation zur Erlangung des Doktorgrades
der physiologischen Wissenschaften
der Johannes Gutenberg-Universität Mainz

dem Fachbereich Medizin vorgelegt
von Marita Gabriele Sergl
aus Erlangen

Mainz, 2001

Tag der Promotion: 20.02.2002

Inhaltsverzeichnis

Kapitel 1

Einleitung

In den letzten zehn Jahren hat der Umfang der elektronischen Informationsverarbeitung im Mainzer Universitätsklinikum ebenso wie in anderen Krankenhäusern kontinuierlich zugenommen. Zu anfänglich vereinzelten Spezialsystemen beispielsweise in der Verwaltung, im Zentrallabor, und der Radiologie, sind inzwischen eine Reihe von teils fachspezifischen, teils allgemeinen Abteilungs-Informations- und Dokumentationssystemen hinzugekommen. Gleichzeitig wurde das Klinikum bis heute fast vollständig vernetzt. Die umfangreichen Möglichkeiten der elektronischen Informationsverarbeitung und -kommunikation haben neue Gefahren für die Vertraulichkeit und Integrität der Patientendaten sowie für die Nachvollziehbarkeit und Unabstreitbarkeit von Datenmanipulationen mit sich gebracht. Im Gegensatz zur Verwendung herkömmlicher papier-basierter Krankenakten ist der Zugriff auf elektronische Patientendaten oder ihre Weitergabe kein offensichtlicher Akt. Ohne besondere Vorkehrungen geschieht er unbemerkt und unkontrolliert. Elektronische Daten können beliebig und mit geringem Aufwand vervielfältigt werden. Mit ebenso geringem Aufwand sind Informationen aus großen Datenmengen ermittelbar. Informationen, die in unterschiedlichen Zusammenhängen über eine Person erfasst wurden, können zu einem umfassenden Personenprofil zusammengeführt werden.

Im Gegensatz zu den Gefahren stehen die Vorzüge der elektronischen Datenverarbeitung, z. B. die Arbeitserleichterung und Rationalisierung, die gleichzeitige Verfügbarkeit der Daten an verschiedenen Stellen, die Möglichkeit umfassender wissenschaftlicher Auswertungen oder die platzsparende Archivierung.

Das deutsche und das europäische Datenschutzgesetz sowie die Landeskrankenhausgesetze regeln in den deutschen Bundesländern, welche Zugriffe auf Patientendaten zulässig sind und in welchem Umfang der Schutz der Daten sichergestellt werden muss(→ 1.1). Klinika und Krankenhäuser sind verpflichtet alle möglichen Maßnahmen zur Einhaltung der Datenschutzbestimmungen zu treffen. Die Gestalt der konkreten Maßnahmen hängt von deren politischen, organisatorischen und strukturellen Rahmenbedingungen, den Bedürfnissen in Bezug auf Abläufe und Informationsflüsse sowie vorhandener und künftiger Infrastruktur und Ressourcen ab.

1.1 Das Rheinland-Pfälzische Landeskrankenhausgesetz

Zur Illustration der gesetzlichen Bestimmungen folgen die wesentlichen Aussagen des Rheinland-Pfälzischen Landeskrankenhausgesetzes [2], die auch für das Mainzer Klinikum gelten.

1.1.1 Patientendaten

Patientendaten sind im Sinne des Gesetzes alle Daten über einen Patienten, die im Rahmen der Behandlung im Krankenhaus erhoben werden sowie die Personendaten von Angehörigen, Bezugspersonen und sonstigen Dritten, die im Zusammenhang mit der Behandlung bekannt werden (Ehepartner, Arbeitgeber etc.).

1.1.2 Zweckbindung

§36, (2) des Landeskrankenhausgesetzes besagt im wesentlichen, dass Patientendaten nur im Rahmen des Behandlungsvertrages verarbeitet werden dürfen. Ausnahmen sind Forschungs- und Ausbildungszwecke, die mit anonymisierten Daten nicht erreicht werden können, sowie die Datenverarbeitung nach expliziter schriftlicher Einwilligung durch den Patienten. Die explizite Einwilligung des Patienten ist allerdings problematisch, da nicht angenommen werden kann, dass ein Mensch, der sich mit einer Erkrankung in die Obhut eines Arztes oder einer medizinischen Einrichtung begibt, (auch nach Aufklärung über seine Rechte) freien Willens einer Datenverarbeitung widerspricht. Diese Ausnahme muss daher grundsätzlich abgelehnt werden. Bei Bestehen eines Behandlungsvertrags wird das Einverständnis des Patienten zur Erfassung und Verarbeitung der notwendigen Daten auch ohne explizite schriftliche Einverständniserklärung konkludent angenommen. Die Übermittlung an Stellen außerhalb der Fachklinik ist nur zur Durchführung des Behandlungsvertrages, zur Abwehr von Gefahren für Leben, Gesundheit und Freiheit des Patienten oder Dritter (Abwägung der Rechtsgüter), zur Abrechnung, im Fall eines Gerichtsverfahren und unter bestimmten Umständen zur Qualitätssicherung im Krankenhaus und an Angehörige des Patienten gestattet (§36, (3)).

1.1.3 "Need-To-Know" – Prinzip der minimalen Rechte

Der Umfang der zu verarbeitenden oder übermittelnden Daten richtet sich nach der Erforderlichkeit im Behandlungszusammenhang. Das bedeutet, dass jede Person nur die Daten einsehen, erfassen oder weiterverarbeiten darf, die sie benötigt, um die optimale Behandlung und Versorgung des Patienten im Krankenhaus sicherzustellen. Entscheidend für die Bewertung der Erforderlichkeit ist das ärztliche Fachpersonal, insbesondere das der Einrichtung, die die Daten erhoben hat.

1.1.4 Maßnahmen

Nach §36, (8) hat das Krankenhaus alle technischen und organisatorischen Maßnahmen zu treffen, die zur Beachtung der Bestimmungen erforderlich und angemessen sind.

1.2 Ziele der Informationsverarbeitung

Den Anforderungen des Datenschutzes stehen die Ziele der Informationsverarbeitung gegenüber [13]. Elektronische Informationsverarbeitung im Klinikum hat vor allem das Ziel der Rationalisierung und Arbeitserleichterung. Im Gegensatz zur Informationsverarbeitung auf der Basis von Papierakten und -dokumenten, sind elektronische Informationen leicht und schnell zu transportieren und an verschiedenen Stellen wiederzuverwenden. Aus der Sicht der Benutzer elektronischer Informationssysteme gibt es folgende Ziele und Bedingungen:

1.2.1 Verfügbarkeit

Jeder an der Behandlung eines Patienten Beteiligte soll die Informationen erhalten und erfassen oder manipulieren können, die zur Ausübung seiner Tätigkeit im Sinne einer optimalen Patientenversorgung notwendig und sinnvoll sind. Das setzt vor allem voraus, dass die Informationen und Funktionen zeitnah zur Verfügung stehen. Die folgende Liste enthält Beispiele von Informationen und Vorgängen, die durch die elektronische Informationsverarbeitung unterstützt werden sollen:

- Das ärztliche Personal benötigt Informationen, die zur Diagnose und Therapie beitragen können oder die administrativen Abläufe unterstützen, z. B.
  - Kenntnis über Befunde, Diagnosen und bereits durchgeführte Therapiemaßnahmen zur aktuellen Erkrankung,
  - Kenntnis über frühere Erkrankungen und spezifische Erkrankungen von Familienmitgliedern
  - Kenntnis über Dauermedikationen, Unverträglichkeiten, Allergien, Risikofaktoren,
  - Kenntnis über die Existenz von Aufnahmen mit bildgebenden Verfahren (Röntgen, CT, MRT) auch aus anderen Erkrankungskontexten und Ansicht der relevanten Aufnahmen,
  - Kenntnis der Identifikationsmerkmale (Patienten-, Fall-IDs, Name, Vorname, Geburtstdatum etc.), der Daten von Angehörigen und der Versicherungsart (gesetzlich, privat),
  - Möglichkeit zur Dokumentation der Diagnostik, der Behandlungsziele, -maßnahmen und -ergebnisse,
  - Meldung der abrechenbaren Leistung an die Verwaltung,

  - Erstellung und gegebenenfalls Weiterleitung spezieller Dokumente (z. B. Arztbriefe).

- Die Verwaltung benötigt zu jedem Patienten Informationen über erbrachte Diagnose- und Behandlungsleistungen, sowie Namen, Adresse, Versicherungsdaten und Bankverbindung. Der gesamte Vorgang der Abrechnung wird mit Hilfe eines computergestützten Systems durchgeführt.

- Die Zentralküche benötigt Informationen über den Aufenthaltsort stationärer Patienten und über deren Ernährungsbedürfnisse.

1.2.2 Arbeitsunterstützung

Die Zeit, die Ärzte und Pflegekräfte für die Informationsgewinnung und Dokumentation aufwenden müssen soll reduziert und die Qualität der Dokumentation verbessert werden. Folgende Aspekte unterstützen die Arbeit des Klinikpersonals in diesem Sinne:

- Die Nutzung vorhandener Daten unabhängig davon, wo und in welchem Zusammenhang sie erfasst wurden, vermeidet arbeitsaufwendige Mehrfacheingaben und spart im Gegensatz zur herkömmlichen Kommunikation Transportwege, Telefonate etc. Dazu ist eine Kommunikationsinfrastruktur zwischen verschiedenen klinischen Systemen notwendig.

- Die automatische Präsenation relevanter Informationen oder notwendiger Bearbeitungsmöglichkeiten ermöglicht dem System-Benutzer eine schnelle Orientierung über vorhandene Informationen und erspart ihm, diese erraten und mühsam zusammensuchen zu müssen. Explizite Datenanforderung und Datenfreigabe sollten deshalb im Normalfall nicht notwendig sein.

- Die Bedienung der Informationssysteme soll unkompliziert sein, das beinhaltet auch, dass aufwendige Authentisierungsverfaren vermieden werden. Am besten wird dieses Ziel durch ein Single-Sign-On-Verfahren erreicht, das den Benutzern sowohl vielfache Anmeldungen als auch das Erinnern von vielen verschiedenen Authentisierungsmerkmalen (Benutzername, Paßwort, evtl. PIN) erspart.

1.2.3 Zuverlässigkeit

Jeder Benutzer eines informationsverarbeitenden Systems soll sich darauf verlassen können, dass

- keine Daten oder Aktionen angeboten werden, die er/sie nicht sehen oder ausführen darf, und dass er /sie in Ausnahmesituationen umfassend über alle Konsequenzen informiert wird,

- die Vertraulichkeit der Patientendaten auf Systemebene gewährleistet ist,

- die Integrität der Daten sichergestellt ist, d. h., dass die präsentierten Daten den erfassten Daten entsprechen,
- der Benutzer genau für die von ihm erfassten oder manipulierten Daten verantwortlich gemacht werden kann.

1.3 Sicherheitsprinzipien

Informationssysteme, die auf der einen Seite die Arbeit der Menschen im Klinikum erleichern sollen und andererseits die Einhaltung der gesetzlichen Bestimmungen gewährleisten sollen, unterliegen hohen Anforderungen an ihre Sicherheit und Zuverlässigkeit. Mit den Begriffen „Vertraulichkeit“, „Integrität“, „Nachvollziehbarkeit“ und „Verfügbarkeit“ sind die Eigenschaften eines verlässlichen Systems definiert [35].

1.3.1 Vertraulichkeit

Patientendaten müssen vor unbefugtem Lesen und Vervielfältigen sowohl am Ort ihrer Speicherung (Datenbanken, Filesysteme, Datenträger etc.) sowie auf dem Weg der Übertragung zwischen verschiedenen Systemen geschützt werden. Die Berechtigung für den Zugriff muss auf der Grundlage der Datenschutzbestimmungen, der ärztlichen Schweigepflicht und einer verbindlichen Zugriffspolitik des Klinikums gegeben (Autorisierung) und bei jedem Zugriff überprüft werden (Zugriffskontrolle). Dafür ist es notwendig, dass sich alle Zugreifenden (Personen oder Systeme) zweifelsfrei authentisieren können.

1.3.2 Integrität

Bei der Verarbeitung und Übertragung von Daten müssen ungewünschte Veränderungen sowohl durch Datendefekte als auch durch mutwillige Verfälschung eindeutig erkennbar sein.

1.3.3 Nachvollziehbarkeit, Unabstreitbarkeit

Daten und Datenmanipulationen müssen der verantwortlichen Person eindeutig und sicher zugeordnet werden können. Nur so ist Nachvollziehbarkeit und Rechtsverbindlichkeit von Handlungen gewährleistet.

1.3.4 Verfügbarkeit

Die Verfügbarkeit von Daten ist die Grundlage einer adäquaten und optimalen medizinischen Versorgung, auf die der Patient Anspruch hat. Sie spielt auch für den Schutz von Daten eine Rolle, da Authentisierung und Zugriffskontrolle nur auf der Basis verfügbarer Informationen über Benutzer und Rechte stattfinden kann.

1.4 Rahmenbedingungen

Die Organisationsstruktur, die vorhandene System- und Infrastruktur, sowie weitere Gegebenheiten eines Krankenhauses sind maßgeblich für die Art, wie Informationssysteme datenschutzkonform und nutzbringend eingesetzt werden können. Zur Veranschaulichung werden im folgenden die spezifischen Rahmenbedingungen des Mainzer Universitätsklinikums beschrieben.

1.4.1 Organisationsstruktur

Das Klinikum besteht aus voneinander unabhängigen Fachkliniken und Instituten, deren Leitungen im weitesten eigenverantwortlich nur durch allgemeine Vorgaben des Klinikvorstands beeinflusst über die Einführung von EDV-Systemen entscheiden. Viele der Einrichtungen besitzen eigene EDV-Mitarbeiter zur Einführung und Betreuung von Abteilungssystemen. In der Vergangenheit wurde ein Teil der Systeme in den Kliniken durch das dortige Fachpersonal entwickelt. Trotzdem gibt es auch Fachkliniken, die keine ausreichende Betreuung der EDV-Aufgaben haben.

Die Mainzer Uniklinik besitzt kein eigenes Rechenzentrum. Verschiedene zentrale Dienstleistungen werden von der EDV-Abteilung der Verwaltung und dem Institut für Medizinische Statistik und Dokumentation (IMSD) erbracht. Die EDV-Abteilung betreibt das Patientenverwaltungs-System, verschiedene zentrale Server (Informations-, Mail-, Fax- und Modemserver) und ist für den gesamten Netzausbau und die Netzverwaltung zuständig. Das IMSD betreibt das Firewall-System, das das Intranet des Klinikums schützt, den Kommunikationsserver, zentrale Datenbanken und ist derzeit mit der Einführung eines klinikweiten Archivsystems betraut. Beide Institutionen unterstützen die Kliniken durch Beratung und Hilfestellung bei Systembeschaffungen und -einführungen sowie bei Problemen inhaltlicher und technischer Art. Aufgrund der geringen Personalresourcen kann trotzdem keine flächendeckende Versorgung mit EDV-Leistungen gewährleistet werden.

1.4.2 Infrastruktur

Abgesehen von einem fast vollständig ausgebauten Intranet auf hohem technischen Niveau besitzt das Mainzer Uniklinikum inzwischen eine große Anzahl von verschiedenen Systemen, die Patientendaten verarbeiten:

- Seit Juli 1999 ist ein neues Patientenverwaltungs-System (SAP/IS-H) im Einsatz. Die Funktion des Patientenverwaltungs-Systems ist die Dokumentation der patientenbezogenen administrativen Vorgänge sowie der Leistungsdaten für die Abrechnung. SAP/IS-H besitzt eine Kommunikationsschnittstelle, durch die andere Systeme mit Informationen über administrative Ereignisse mit den zugehörigen Daten (inklusive abrechnungsrelevanter Diagnosen) versorgt und Informationen über erbrachte Leistungen zurückgemeldet werden können.

- In den Kliniken und Abteilungen gibt es für den lokalen Bedarf etliche Abteilungs-Informationssysteme, Systeme für Spezialdokumentation, Intensivüberwachung und bildverarbeitende Systeme. Die Vielfalt der Systeme hat auch eine große Vielfalt an Hard- und Software-Plattformen zur Folge.

- Die Radiologische Klinik und das Zentrallabor betreiben (unterschiedliche) Systeme, die kontrolliert klinikweiten Zugriff auf Befunddaten geben.

- Eine zentrale Patientendatenbank (Referenzdatenbank), die über die Schnittstelle des Verwaltungssystems kontinuierlich aktualisiert wird, dient zu Recherchen und zur Versorgung von Systemen, die keine ereignis- sondern eine zustands-gesteuerte Übernahme von Patientendaten unterstützen.

- Der Kommunikationsserver steuert den Datenaustausch zwischen verschiedenen Systemen. Er empfängt und identifiziert Nachrichten, die von Systemen im Klinikum verschickt werden, übersetzt die Nachricht gegebenenfalls in verschiedene Formate und Inhalte und verschickt sie an die zugehörigen Empfängersysteme. Aufgrund seiner Funktion als „Datendrehscheibe“ hat er bereits Anteil an der Zugriffskontrolle.

- Für den Betrieb der Mail-, Fax-, und Modemserver wurde ein zentrales Benutzerverzeichnis aufgebaut. Es umfasst bereits einen großen Teil des Klinikpersonals, nämlich die Personen, die für die Nutzung dieser Dienste registriert sind, und ist für die klinikweite Nutzung konzipiert.

1.4.3 „Widrigkeiten“

Bisher können weder die Sicherheitsziele noch die Ziele der Informationsverarbeitung im vollen Umfang erreicht werden. Das hängt zum Teil mit fehlender Infrastruktur, zum Teil mit den Erschwernissen durch eine stark verteilte heterogene Informationssystem-Landschaft sowie mit mangelnden Personalressourcen zusammen. Die folgende Auflistung enthält Widrigkeiten, die Informationslücken im Sinne des Informationsbedarfs der Nutzer und im Sinne der Zugriffskontrolle zur Folge haben:

- Bislang existiert keine flächendeckende Versorgung mit Aufnahmestationen des Verwaltungsystems, so dass noch immer ein Teil der Aufnahme- und Verlegungsmeldungen auf Papier erfasst und mit bis zu drei Tagen Verzögerung im Verwaltungssystem erscheinen. Die normale zeitliche Verzögerung zwischen Eingabe und Reaktion der Schnittstelle beträgt etwa sechs Minuten.

- Die Kommunikationsschnittstelle des Patientenverwaltungs-Systems gibt die erfasste Information nur in geringem Umfang an andere Systeme weiter. Sie übermittelt z. B. keine Informationen über dokumentierte Therapien.

- Es gibt bislang keine elektronische Leistungsanforderung und nur eine geringfügige elektronische Befundübermittlung.
- Auch Informationen über Mitbehandlung oder Konsile sind nicht in elektronischer Form verfügbar.
- Das existierende Benutzerverzeichnis ist noch unvollständig. Insbesondere fehlt die Möglichkeit zur Erfassung von Informationen zu Zugriffsrechten der Nutzer.

1.5 Ziel der Arbeit

Ziel der vorliegenden Arbeit ist die Erstellung eines Konzepts für die datenschutzkonforme Nutzung informationsverarbeitender Systeme in Klinika und Krankenhäusern im allgemeinen und konkret im Mainzer Universitätsklinikum, wobei der Schwerpunkt der Arbeit auf dem Aspekt der Zugriffskontrolle liegt. Das beinhaltet einerseits die Frage nach der Zulässigkeit von Zugriffen unter Berücksichtigung der gesetzlichen Bestimmungen und andererseits die Frage nach Mechanismen und Komponenten, mit deren Hilfe unzulässige Zugriffe a priori verhindert oder a posteriori aufgedeckt werden können, wobei der Informationsbedarf der Systemnutzer und die oben aufgeführten spezifischen Rahmenbedingungen zu berücksichtigen sind. Ein wichtiger Aspekt hierbei ist die Integrierbarkeit der Komponenten in die verteilte, heterogene klinische Systemlandschaft und in die klinischen Abläufe.

1.6 Allgemeine Bemerkungen zur Arbeit

Zeitlicher Rahmen Die vorliegende Arbeit entstand im Zeitraum zwischen Herbst 1999 und Sommer 2001. Alle Angaben z. B. zur aktuellen Systemlandschaft beziehen sich auf diese Zeit. Die Angaben zum Stand der Realisierung beziehen sich auf das Frühjahr 2001.

Beispiele Die Beispiele, die in der Arbeit zur Illustration des Modells oder zur Realisierung angeführt werden, sind i. a. konstruierte Szenarien auf der Basis realer Strukturen des Mainzer Klinikums. Beispiele für die Einrichtungshierarchie und Personalstruktur des Klinikums zeigen charakteristische Strukturen an Hand von Ausschnitten, erheben aber keinen Anspruch auf inhaltliche Vollständigkeit.

Notation Graphische Darstellungen zur Beschreibung des Modells sind in UML-Syntax notiert; die verwendeten Symbole werden deshalb nicht näher erläutert. Spezifikationen sind in IDL-Syntax notiert und durch eine Schriftart mit konstanter Zeichenbreite hervorgehoben.

Kapitel 2

Anforderungsanalyse

Die Zugriffskontrolle hat ein wesentliches Ziel: für jeden Zugriff auf Daten muss zu jedem Zeitpunkt und an jedem Ort über dessen Zulässigkeit gemäß der festgelegten Zugriffspolitik entschieden werden können.

Dieses grundsätzliche Ziel, übertragen auf die Informationsverarbeitung in Krankenhäusern, zieht eine Reihe von Anforderungen und Fragestellungen inhaltlicher und technischer Art nach sich, die im folgenden aufgeführt sind:

1. Für das jeweilige Klinikum oder Krankenhaus muss eine verbindliche Zugriffspolitik existieren, die aus den Datenschutzbestimmungen abgeleitet ist und alle internen Zugriffs-Vereinbarungen enthält.

2. Das Zugriffskonzept benötigt eine geeignete Ontologie, d. h. ein Modell aller Inhalte, die Zugriffe und Zulässigkeit von Zugriffen beschreiben, und ihrer Zusammenhänge. Dazu gehören die Fragen:

   - Was genau ist ein Zugriff? Was sind zu schützende Objekte? Wer oder was sind Subjekte?
   - Wie spiegeln sich umgebungsspezifische Strukturen, wie z. B. Fachkliniken und Abteilungen eines Klinikums in der Zugriffspolitik wieder, d. h., welche Strukturen muss ein Modell zur Beschreibung der Zugriffsregeln berücksichtigen?
   - Welche dynamischen Einflüsse, z. B. spezifischen Abläufe des Klinikums referenziert die Zugriffspolitik.

3. Das Zugriffskontrollsystem muss im klinischen Alltag handhabbar sein. Dazu gehört die Minimierung manueller Erfassung von Zugriffsinformation.

   - Zugriffsrechte sollen zu einem Zeitpunkt definiert werden können, zu dem Objekte und Subjekte noch nicht bekannt sind. Im klinischen Alltag ist es weder ökonomisch noch handhabbar, Zugriffsrechte für jeden Patientendatensatz zum Zeitpunkt der Erzeugung manuell zu

vergeben. Deshalb wird eine Sprache benötigt, die es ermöglicht, Zugriffe auf einer abstrakten Ebene a priori zu beschreiben.

- Informationen über dynamische Abhängigkeiten von Zugriffsrechten sollen automatisch zur Verfügung gestellt werden. Sie sollen sozusagen als „Nebenprodukt“ der Abläufe im Klinischen Informationssystem enstehen.

4. Das Wissen über die Zulässigkeit von Zugriffen muss allen Anbietern von Zugriffen auf Patientendaten in einer Form zur Verfügung gestellt werden können, die sie verarbeiten können. Das beinhaltet einerseits die Frage nach Semantik und Syntax von Zugriffsinformation und andererseits Architektur und Schnittstellen, mit deren Hilfe Zugriffsinformation verfügbar gemacht wird.

Die folgenden Abschnitten enthalten eine Untersuchung der gestellten Fragen und Anforderungen, wobei exemplarisch die Strukturen des Mainzer Universitätsklinikums zugrundegelegt werden.

2.1 Zugriffspolitik (Wer darf was, wann?)

Die Zugriffspolitik regelt zunächst zwei grundsätzliche Fragen, die sich bei der Formulierung von Zugriffrechten stellen. Welche der im Klinikum beschäftigten Personen sind im weiteren Sinne an der Behandlung welcher Patienten beteiligt, und auf welche Daten eines Patienten müssen sie aufgrund ihrer Funktion innerhalb der Behandlung zugreifen? Sie regelt darüberhinaus, nach welchen Kriterien diese Zuständigkeit zum jeweiligen Zeitpunkt erkannt wird. Das beinhaltet auch, wie mit fehlenden oder ungenauen Informationen über die Zuständigkeit vorübergehend umzugehen ist. Die folgenden Ausführungen basieren auf Ergebnissen, die in Kooperation mit dem Datenschutzbeauftragten des Mainzer Klinikums, den Landesdatenschutzbeauftragten und im Rahmen der GMDS-Arbeitsgruppe „Datenschutz und Datensicherheit in Gesundheitsinformationssystemen“ erarbeitet wurden.

2.1.1 Zuständigkeit für Patienten und Behandlungen

Behandlungsauftrag: Wichtigstes Kriterium für den Zugriff auf die Daten eines Patienten ist das Bestehen eines Behandlungsvertrags zwischen dem Patienten und einem Arzt oder einer klinischen Einrichtung stellvertretend für die jeweils diensthabenden Ärzte und deren Helfer. Insbesondere das Pflegepersonal der Stationen ist in den Auftrag eingeschlossen. Der Behandlungsvertrag beginnt, wenn der Patient die klinische Einrichtung zum Zweck der Behandlung oder Beratung aufsucht. Das Bestehen eines Behandlungsvertrags wird also für jeden registrierten Aufenthalt eines Patienten angenommen.

Mitwirkung: Auch das an der Behandlung mitwirkende Personal anderer klinischer Einrichtungen benötigt Zugriff auf die Daten eines Patienten, wie z. B. bei konsiliarischer Mitwirkung, bei Erbringung diagnostischer oder therapeutischer Leistungen, Medikamentierung durch die Zentralapotheke oder der Versorgung stationärer Patienten durch die Zentralküche. Kriterium für die Mitwirkung ist üblicherweise das Bestehen eines Auftrags an die entsprechende Person oder Einrichtung.

Verwaltung: Mitarbeiter der Patientenverwaltung haben zum Zweck der Abrechnung Zugriff auf Daten aller Patienten.

2.1.2 Datensicht und Funktionsumfang

Der Umfang des Zugriffs, d. h. auf welche Daten und mit welcher Art von Zugriffsoperation, richtet sich nach dem „Need-To-Know-Prinzip“ oder hier genauer „Need-To-Access-Prinzip“. Gemeint sind im Wesentlichen die Kenntnis der zur optimalen Behandlung notwendigen Informationen, die gesetzlich vorgeschriebene Dokumentation sowie jegliche Funktionalität, die der Unterstützung ärtzlicher oder pflegerischer Arbeitsabläufe dient. Die folgenden Regelungen sind grobe Maßgaben; sie müssen für jede Abteilung nach den Erfordernissen in der Praxis ausgestaltet werden.

- Daten, die Patienten identifizieren, Personendaten wie Adresse und Beruf, Krankenversicherungsdaten sowie die Besuchs-/Verlegungshistorie des aktuellen Behandlungsfalls dürfen von allen an der Behandlung beiteiligten Personen zur Kenntniss genommen werden.

- Daten, die der Auffindung und Anforderung von Informationen über relevante, frühere Erkrankungen dienen, müssen für die behandelnden Ärzte verfügbar sein.

- Zugriffsrechte von Mitarbeitern einer Fachklinik bezüglich der dort erhobenen Patientendaten liegen in der Verantwortung des Klinikleiters.

- Zur Behandlung eines Patienten gehören alle Daten, die im Zusammenhang mit der Behandlung erfasst werden. Das sind unter anderem Anamnesedaten, Diagnosen und Befunde aller an der Behandlung beteiligten Einrichtungen, alle Dokumente, die im Rahmen der Behandlung erstellt werden, Bilddaten und alle Aufzeichnungen über Behandlungsverlauf und Pflege. Grundsätzlich besitzen die Ärzte die fachliche Kompetenz festzulegen, welche Zugriffe auf diese Patientendaten üblicherweise im Rahmen von Behandlungen oder im Einzelfall notwendig sind. Die Notwendigkeit muss gegebenenfalls gegenüber dem Datenschutzbeauftragten der Klinik begründet werden.

- Ein behandelnder Arzt entscheidet zum Zeitpunkt der Behandlung, welche Daten für einen mitwirkenden Fachkollegen einer anderen Einrichtung zugreifbar sein müssen, und muß sie in diesem Fall aktiv freigeben. Für

häufige Kooperation zwischen bestimmten Fachabteilungen kann der Zugriffsumfang auch a priori definiert und automatisch angewendet werden.

2.1.3 Notfallregeln

Im Notfall wird der Zugriff auf Daten aller Patienten uneingeschränkt gewährt. Jeder Zugriff wird mit allen Details protokolliert und muss nachträglich überprüft und gerechtfertigt werden.

2.1.4 Abhängigkeit der Zugriffsrechte vom Inhalt

Daten müssen in Abhängigkeit ihres Inhalts für den Zugriff durch Personen anderer Einrichtungen grundsätzlich ausgeschlossen werden können. Dazu können z. B. psychiatrische und psychosomatische Diagnosen gehören, aber auch noch nicht freigegebene Befunde müssen - selbst im Notfall - vom Zugriff ausgeschlossen werden können. Die Entscheidung darüber wird von den für die Erhebung der Daten verantwortlichen Personen getroffen.

2.1.5 Zugriff auf Patientendaten zum Zweck von Forschung oder Lehre

Zugriff auf Patientendaten für die Ausbildung von Studenten oder Personal darf nur ohne Personenbezug stattfinden. Die Daten dürfen nichts beinhalten, was auf die Identität des Patienten schließen lässt. Zu Forschungszwecken sollte der Zugriff, wenn möglich, ebenfalls ohne Personenbezug stattfinden (Rheinland-Pfälzisches Landeskrankenhausgesetz §37, (4)). Grundsätzlich dürfen in Verantwortung des Leiters einer Fachklinik Patientendaten der eigenen Patienten zu Forschungszwecken verwendet werden.

2.1.6 Ausnahmeregelungen

Für Situationen bei der Patientenversorgung (nicht Forschung oder Lehre), in denen der Nachweis des Behandlungszusammenhangs zum Zugriffszeitpunkt nicht möglich ist, müssen Ausnahmeregelungen existieren. Die bislang unzureichende Infrastruktur des Mainzer Klinikums, beispielsweise das Fehlen zeitnaher Registrierung von Aufnahmen und Verlegungen sowie das völlige Fehlen einer elektronischen Auftrags-Registrierung ist der Grund dafür, dass hier derartige Situationen noch längere Zeit systematisch auftreten werden. Ein weiterer unvorhersehbarer Grund kann der Ausfall von Systemen oder Netzen sein. Mit folgenden Regeln wird daher der Rahmen für einen „eigenverantwortlichen Zugriff“ in Ausnahmesituationen vorgeschlagen:

- Einzelne Datensätze können gezielt abgerufen werden. Sie dürfen generell administrative Patientendaten und Information über den Aufenthaltsort des Patienten enthalten, nach Absprache auch weitere Patientendaten. Der Abruf erfolgt über die eindeutige Patienten- oder Aufnahmenummer

des Klinikums oder über ausreichend spezifizierte Identifikationskriterien, wie Familienname, Vornamen, Geburtsdatum, evtl. Geburtsname und Wohnort. Die Systematik darf keinen Abruf von Datensatz-Mengen unterstützen. Die Auswahllisten dürfen nur die notwendigen Identifikationskriterien anzeigen, die Menge der Identifikationssätze muss durch ausreichende Spezifikation der Suchkriterien begrenzt sein, um Massenabfragen von Daten zu unterbinden.

- Der Zugriff, auch jeder lesende, muss in vollem Umfang, d. h. unter Angabe von zugreifendem Subjekt, Uhrzeit, Art des Zugriffs und der betroffenen Daten, protokolliert werden.
- Zugriff und Autorisierung müssen nachträglich überprüft werden.
- Diagnosen aus Psychiatrie oder Psychosomatik sowie andere besonders sensible Patientendaten sollten von dieser Regelung ausgenommen sein. Bei begründeter Notwendigkeit können Sonderregelungen mit dem Datenschutzbeauftragten des Klinikums explizit vereinbart werden.

2.1.7 Regeln für Behandlungszusammenhänge

Die oben aufgeführten Kriterien für einen Behandlungsauftrag erlauben die Formulierung von Regeln für abteilungs- bzw. stationsspezifische Zugriffe. Der Umfang der Zugriffe bezüglich verschiedener Arten von Daten und Operationen muss für die einzelnen Organisationseinheiten nach deren Funktionen festgelegt werden. In den folgenden Regeln ist dieser Sachverhalt in der Formulierung *„in funktionsgemäßem Umfang“* enthalten. Die Regeln enthalten nur einen Teil der Zugriffspolitik, geben aber eine grundsätzliche Regelstruktur wieder.

Regel 1 Das medizinische Personal einer klinischen Organisationseinheit darf auf die vor Ort oder anderswo im Zusammenhang mit dem Aufenthalt erzeugten Daten in funktionsgemäßem Umfang zugreifen.

Regel 2 Wurde der Patient bereits in einer oder mehreren Organisationseinheiten des Klinikums bezüglich einer Erkrankung behandelt, so darf das medizinische Personal der Organisationseinheit, in der der Patient aktuell behandelt wird, auf die Daten der gesamten bisherigen Behandlung in funktionsgemäßem Umfang zugreifen, solange sich der Patientent zur Behandlung in der Einheit befindet und bis die gesetzlich vorgeschriebene Dokumentation abgeschlossen ist.

Regel 3 Besteht ein Auftrag an eine Person oder eine klinische Organisationseinheit zur Erbringung einer Leistung (diagnostische, therapeutische Verfahren, Konsil etc.), so darf die Person bzw. die Mitarbeiter der Organisationseinheit in funktionsgemäßem Umfang auf die Daten der gesamten bisherigen Behandlung zugreifen.

Regel 4 Mitarbeiter der Verwaltung dürfen auf Daten aller Behandlungen in funktionsgemäßem Umfang zugreifen.

Für die Aktualisierung redundanter Datenhaltung in verschiedenen Organisationseinheiten der Klinik gilt außerdem:

Regel 5 Organisationseinheiten, die in eigenen Systemen Daten anderer Organisationseinheiten oder Institute redundant speichern, erhalten Zugriff auf aktualisierte Daten (z. B. die geänderte Patientenadresse, Aufenthaltsort des Patienten, Entlassungsstatus) in dem Umfang, in dem sie während der Behandlung Daten erhalten hatten.

Die aufgeführten Zugriffsregeln referenzieren – neben dem Begriff des Zugriffs – Einrichtungen und Personal des Klinikums, Funktionen des Personals sowie Behandlungsverläufe, deren Strukturen Einfluss auf die Interpretation der Regeln haben. Diese Strukturen werden deshalb im folgenden am Beispiel des Mainzer Universitätsklinikums betrachtet.

2.2 Statische Einflüsse – Klinikstrukturen

Die Struktur der Einrichtungen und des Personals ändern sich i. A. nur im Rahmen einer Umstrukturierung, nicht im alltäglichen Betrieb, weshalb sie hier als statische Einflüsse bezeichnet werden.

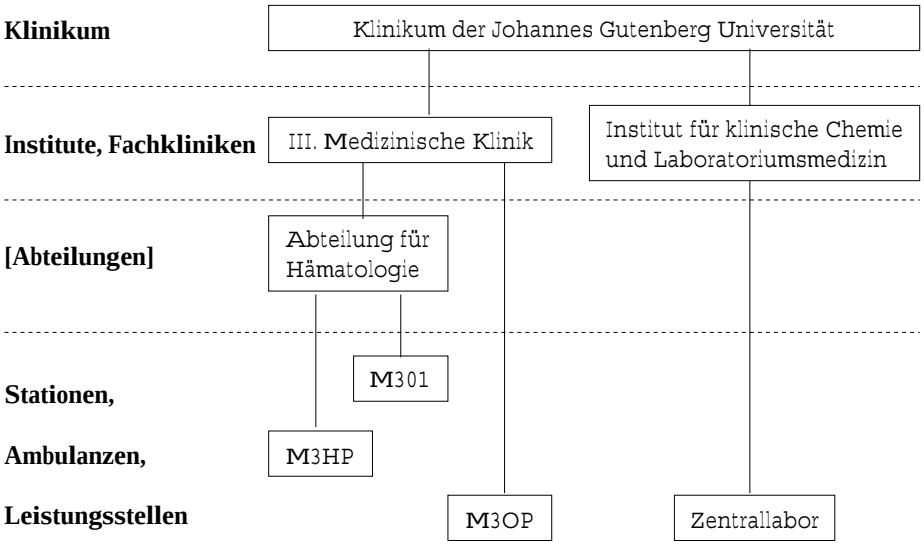

Abbildung 2.1: Ausschnitt aus der Einrichtungsstruktur des Mainzer Uniklinikums

2.2.1 Die Einrichtungsstruktur

Das Klinikum ist hierarchisch in Einrichtungen, sogenannte „Organisationseinheiten“, gegliedert. Die oberste Hierarchieebene umfasst das gesamte Klinikum. Darunter gibt es Fachkliniken und Institute, die wiederum in Abteilungen untergliedert sein können. Fachkliniken und Abteilungen enthalten als Untereinheiten Stationen, Ambulanzen und Leistungsstellen. Aus der Zugehörigkeit zu einer Organisationseiheit kann man immer die Zugehörigkeit zu allen übergeordneten Organsisationseinheiten ableiten. Abbildung 2.1 zeigt einen Ausschnitt aus der Einrichtungsstruktur des Mainzer Uniklinikums.

2.2.2 Das Klinikpersonal

Im Uniklinikum sind Personen vieler unterschiedlicher Berufsgruppen und Qualifikationen in verschiedenen Funktionen tätig. Die Aufgaben und Verantwortlichkeiten, die eine Person hat, sowie die vorausgesetzte berufliche Qualifikation sind durch ihre Stellenbeschreibung festgelegt. Der Teil des Klinikpersonals, der

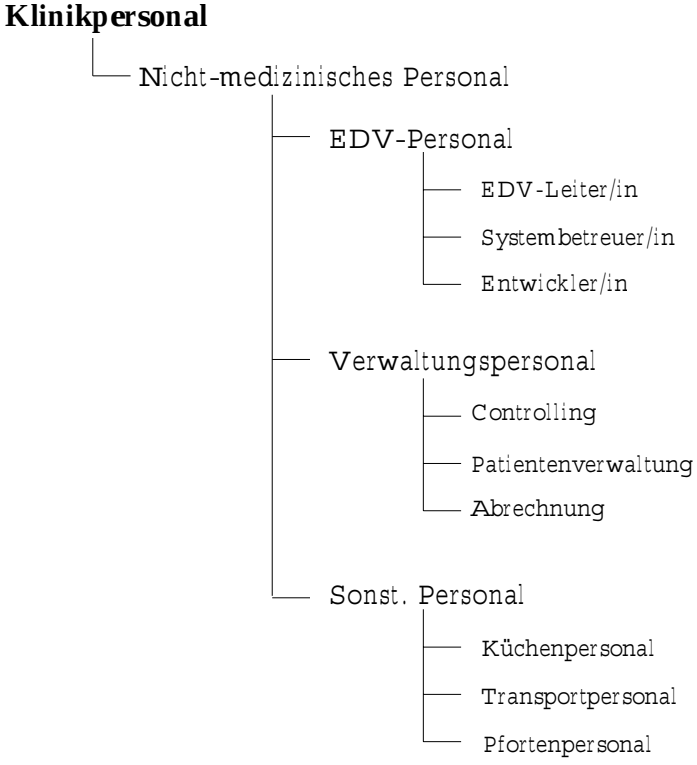

Abbildung 2.2: Das nicht-medizinische Personal des Klinikums (Ausschnitt)

bei seiner Arbeit mit Patientendaten zu tun hat, lässt sich nach verschiedenen Kriterien in Gruppen unterteilen. Eine beispielhafte Gliederung in Berufsgruppen und unterschiedliche Funktionen innerhalb der Berufsgruppen ist in den Abbildungen 2.2 und 2.3 dargestellt. Man sieht teilweise an den Funktionsbezeichnungen, dass Funktion und Qualifikation eng miteinander verknüpft sind. Aufgaben und Qualifikation gleich lautender Funktionsbezeichnungen (z. B. Oberarzt) unterscheiden sich aufgrund unterschiedlicher Spezialisierungen und Abläufe in verschiedenen Einrichtungen des Klinikums. Einige der Personengruppen stehen in einer hierarchischen Relation zueinander. So ist z. B. innerhalb einer Fachkli-

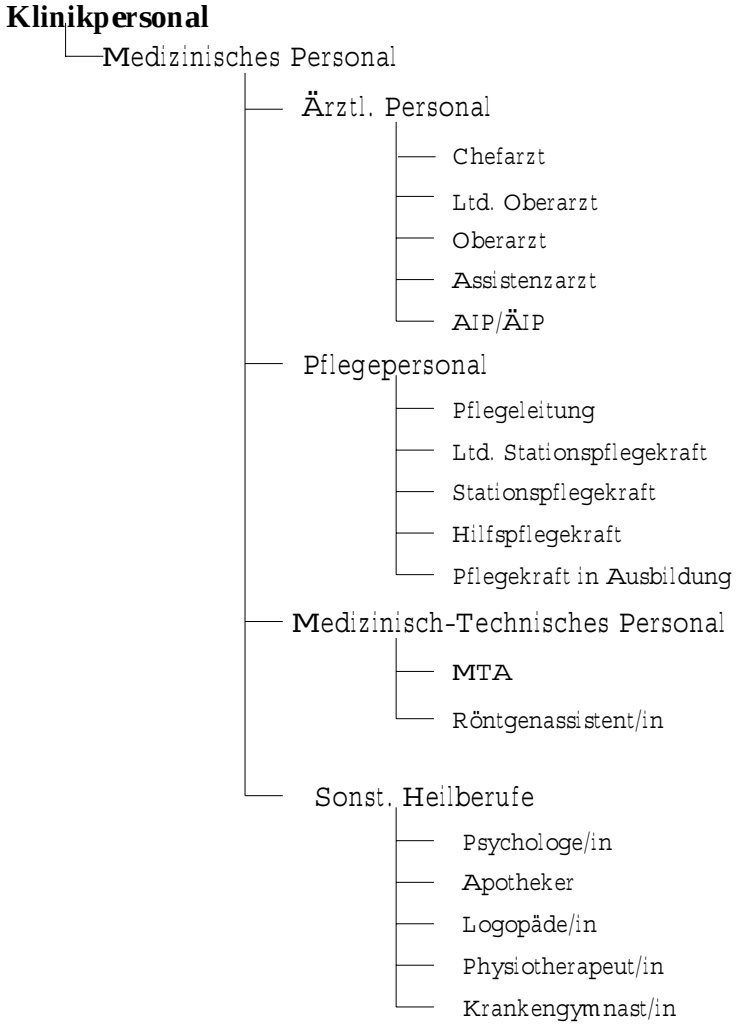

Abbildung 2.3: Das medizinische Personal des Klinikums (Ausschnitt)

nik der Chefarzt allen Ärzten übergeordnet, der leitende Oberarzt allen anderen Ärzten mit Ausnahme des Chefarztes, usw. Oft – wenn auch nicht immer exakt – entspricht die Hierarchie den Verantwortlichkeiten und damit den benötigten Rechten zum Zugriff auf Patientendaten.

2.3 Dynamische Einflüsse – Prozesse

Im Gegensatz zu Einrichtungs- und Personalstrukturen unterliegen die Patientenbehandlungen sowie die Personaleinteilung im Krankenhausbetrieb einem kontinuierlichen Prozess, der die Interpretation der Zugriffsregeln beeinflusst.

2.3.1 Dienstpläne

Gemäß der Notwendigkeit eines Vierundzwanzig-Stunden-Betriebs im Krankenhaus kann jede notwendige Funktion von verschiedenen Angehörigen des Personals wahrgenommen werden, z. B. pflegerische, ärztliche oder technische Funktionen. Personen werden außerdem in verschiedenen Funktionen eingesetzt, z. B. als Vertretung in einer anderen Abteilung. Ärzte in der Facharztausbildung arbeiten einem Rotationsverfahrens entsprechend in verschiedenen Abteilungen oder je nach Fach verschiedenen Fachkliniken. Pflegekräfte und Ärzte werden zumindest vertretungsweise in verschiedenen Stationen eingesetzt. Welche Person zu einem Zeitpunkt in welcher Funktion tätig ist, ist Inhalt der Dienstpläne, die für das ärztliche Personal von einem verantwortlichen Oberarzt und für das Pflegepersonal von der Pflegedienstleitung aufgestellt werden.

2.3.2 Der Behandlungsprozess

Die Abläufe bei der Behandlung von Patienten im Klinikum können zunächst aus zwei grundlegend verschiedenen Sichten gesehen werden, aus der Sicht der Behandler und aus der Sicht der Verwaltung. Beide Sichten spiegeln sich auch in der Dokumentation der Abläufe, d. h., in den Formularen, Dokumenten und Notizen sowie in den Informationsflüssen wieder.

Die medizinische Sicht ist problem- und ereignisorientiert. Die medizinische Behandlung beginnt mit dem ersten Kontakt zwischen Patient und Arzt bezüglich eines Problems und endet mit der Heilung, dem Abbruch der Behandlung oder dem Tod des Patienten, bei chronischen Erkrankungen kann sie beispielsweise über lange Zeit ohne Abschluss bleiben. Sie enthält in diesem Zeitraum eine Menge teilweise kausal verknüpfter Ereignisse und Maßnahmen (diagnostische und therapeutische), die an verschiedenen Orten stattfinden, eventuell in verschiedenen Krankenhäusern und Praxen niedergelassener Ärzte. Maßnahmen werden i. A. durch Verordnungen, Anforderungen oder Überweisungen des jeweils behandlenden Arztes initiiert. Ereignisse sind alle Zustandsänderungen, z. B. eine Veränderung des Gesundheitszustandes beim Patienten.

Die administrative Sicht ist von organisatorischen und abrechnungsspezifischen Erfordernissen geprägt. Die Verwaltung unterscheidet ambulante, teilsta-

tionäre und stationäre Fälle. Jeder Fall kann in erster Linie als Einheit bezüglich der Abrechnung angesehen werden. Der stationäre Fall beginnt mit der stationären Aufnahme oder einem von maximal drei vorstationären Besuche des Patienten in der Klinik innerhalb von fünf Tagen vor der stationären Aufnahme. Er endet mit der Entlassung oder mit dem letzten von maximal sieben nachstationären Besuchen, die innerhalb von vierzehn Tagen erfolgen müssen (§115 a, SGB V [1]). Ein stationärer Fall beinhaltet mindestens einen stationären Aufenthalt in einer Station des Klinikums, er kann aus einer Folge von Aufenthalten in verschiedenen Stationen verschiedener Fachkliniken oder Abteilungen sowie kurzen Aufenthalten zu Hause bestehen. Teilstationäre Fälle sind Phasen einer maximalen Dauer von einem Jahr, in denen der Patient wiederholt zur Behandlung auf eine bestimmte Station kommt. Ein typisches Beispiel ist die Dialysebehandlung. Der ambulante Fall beginnt mit der ersten Konsultation eines Arztes in einer ambulanten Einrichtung, enthält alle weiteren Besuche in dieser oder anderen Ambulanzen des Klinikums, zu denen der Patient üblicherweise einbestellt bzw. aufgefordert wird, und endet in der Regel mit Abschluss des Quartals. Ein ambulanter Besuch erstreckt sich über maximal einen Tag. Die administrative Sicht beinhaltet also Phasen von Patientenaufenthalten an einem Ort. Desweiteren berücksichtigt die Verwaltung alle abrechnungsrelevanten Leistungen eines Falls, auch solche ohne direkten Patientenkontakt wie z. B. Laborleistungen, die auf Anforderung durch die behandelnde Organisationseinheit erbracht werden. Jeder Aufenthalt oder Besuch wird durch einen administrativen Vorgang, wie Aufnahme, Verlegung oder Einbestellung eingeleitet und endet entweder nach einer vorgegebenen Zeit oder in Verbindung mit einem weiteren Vorgang wie Verlegung, Beurlaubung nach Hause oder Entlassung. Zu jedem Vorgang gibt es entsprechende Formulare. Jede erbrachte, abrechnungsrelevante Leistung muss in einer speziellen Form (gemäß den gesetzlichen Bestimmungen) dokumentiert werden. Jede Einrichtung, die Leistungen für andere erbringt, hat spezifische administrative Vorgänge und zugehörige Formulare für die Leistungsanforderung.

Beiden Sichten gemeinsam sind Phasen, die mit Organisationseinheiten oder Personen assoziiert sind. Der administrative Ablauf bildet den Rahmen für die medizinischen Abläufe. Er beinhaltet vor allem Phasen, in denen sich ein Patient in einer Organisationseinheit aufhält. Die medizinischen Abläufe umfassen Phasen, in denen einzelne Personen oder Personengruppen etwas für oder mit dem Patienten tun. Für die Zugriffsproblematik sind die Ereignisse von Bedeutung, die einen Wechsel der beteiligten Personen und Organisationseinheiten hervorrufen bzw. Phasen unveränderter Beteiligung, derart wie sie z. B. in den angegebenen Zugriffsregeln referenziert wurden. Betrachtet man den Behandlungsprozess auf der Basis dieser Beteiligungsphasen, so können weitere Eigenschaften festgehalten werden:

- Beteiligungsphasen sind durch Ereignisse[1] begrenzt.
- Beteiligungsphasen bilden Klassen bezüglich eines Patienten, eines Be-

[1] Administrative Vorgänge sind in diesem Sinne Ereignisse.

handlungsfalls und eines administrativen Falls. Sie besitzen außerdem eine zeitliche Ordnung.

- Die Beteiligungsphase einer Organisationseinheit kann aus einer Folge von Beteiligungsphasen untergeordneter Einheiten zusammengesetzt sein.
- Insbesondere impliziert jede Beteiligungsphase einer Organisationseinheit auch entsprechende Phasen aller übergeordneten Einheiten.
- Innerhalb der Beteiligungsphase einer Einheit können Wechsel in der Verantwortlichkeit von Personen für Teilphasen stattfinden. Diese Wechsel sind im Normalfall durch die Dienstpläne festgelegt. Der Personaleinsatz bei Mitbehandlung einer Fachabteilung ist i. a. ebenfalls durch den Dienstplan geregelt.

Tabelle 2.1 enthält die wichtigsten Beteiligungsphasen des Prozessmodells. Aus dem Informationsfluss im Klinikum und der Dokumentation der medizinischen und administrativen Abläufe können diese Phasen ermittelt werden.

Phase	**Beteiligung**	**Kriterien** (Beginn – Ende)
Verwaltungsfall	Patientenverwaltung	Aufnahme, Einbestellung – Entlassung, Quartalsende
Fachabteilungsphase	Fachabteilung	Aufnahme, Verlegung – Verlegung, Entlassung
Stationsphase	Station	Aufnahme, Verlegung – Verlegung, Entlassung
Ambulante Phase	Ambulanz	Aufnahme, Einbestellung – Abschluss oder Quartalsende
Fremdleistung	Leistungsstelle	Auftrag – Leistung
Mitbehandlung	Fachabteilung, angeforderte Personen	Anforderung – Abschluss der Behandlung
Konsil	angeforderter Kollege	Anforderung – Behandlungsabschluss

Tabelle 2.1: Phasen konstanter Behandlungsbeteiligung von Organisationseinheiten oder Personen

2.4 Die Zugriffs-Ontologie

Neben den Kliniksstrukturen referenziert die Zugriffspolitik Umfang und Art der Zugriffe. In diesem Abschnitt werden deshalb „Zugriffe“ genauer analysiert und in einen Gesamtzusammenhang mit den Strukturen der klinischen Umgebung gestellt. Das Ergebnis ist die „Zugriffs-Ontologie“. Sie umfasst alle Entitäten, die in einem Zugriffszenario direkt oder indirekt mitwirken und ihre Beziehungen zueinander. Sie greift dabei auf relevante Aspekte der „klinischen Prozesse“ als Grundlage der Zugriffspolitik und auf ein Modell der „klinischen Informationsverarbeitung“ als Grundlage der Zugriffsdefinitionen zurück. Sie integriert die statischen Strukturen des Klinikums.

2.4.1 Entitäten

Definitionen: Was ist ein Zugriff?

> Ein **Zugriff** ist die Ausführung einer *wohldefinierten Operation auf Patientendaten* durch ein zugreifendes *Subjekt*.

> Als **Zugriffsobjekte** sollen in diesem Zusammenhang die Einheiten von Patientendaten mit den darauf definierten Operationen betrachtet werden.

Die „Wohldefiniertheit“ der (abstrakten) Zugriffsobjekte, muss in einem *Informationsverarbeitungs-Modell des Klinikums* festgelegt sein. Insbesondere sind die real existierenden Zugriffsobjekte wohldefiniert.

> **Subjekte** sind alle *Mitarbeiter* und *Systeme* des Klinikums, die in der Patientenversorgung mitwirken, und im Rahmen ihrer *Funktionen* Zugang oder Schnittstellen zu (Informations-)Systemen benötigen.

Die Zugriffsumgebung

Jeder Zugriff findet in einer definierten Umgebung statt. Die Umgebung beinhaltet zum einen die räumliche oder organisatorische Einordnung, d.h. die **Organsisationseinheiten**, und zum anderen die vorhandenen **Informationssysteme**.

Der Behandlungsprozess

Jeder Zugriff steht in einem kausalen Zusammenhang zum Behandlungsgeschehen. Er findet i. A. während einer **Behandlungphase** statt (zeitliche Kausalität) und verarbeitet Informationen, die eine Behandlungsphase betreffen (semantische Kausalität).

Exkurs: Strukturen der klinischen Informationsverarbeitung

Zum besseren Verständnis der Zugriffe und Zugriffsobjekte benötigt man ein Modell, das beschreibt, wie Patientendaten innerhalb der klinischen Prozesse verarbeitet werden. Auf einer abstrakten Ebene, die keine Annahme über die Systemumgebung macht, beinhaltet dieses Modell zwei grundlegende Strukturen, die *elektronische Krankenakte* und die darauf stattfindenen *informationsverarbeitenden Funktionen*, wie z. B. die Gewinnung von Informationen über den Patienten oder die Dokumentation bestimmter Aspekte einer Behandlung.

- **Die elektronische Krankenakte** umfasst alle Daten (auch Bilddaten) eines Patienten, die jemals bei Behandlungen im Klinikum – unabhängig vom Ort und Zeitpunkt der Verarbeitung – erhoben werden.

- **Informationsverarbeitende Funktionen** setzen sich aus Operationen auf Ausschnitten einzelner elektronischer Krankenakten zusammen. Die Funktionen ergeben sich i. A. aus Anforderungen der gesetzlichen Dokumentationspflicht, der Entscheidungsunterstützung, der Organisationsunterstützung und der Kommunikation. Insbesondere kann eine Funktion auf verschiedene Krankenakten wirken und beliebig komplex zusammengesetzt sein. Welche Funktionen im einzelnen bei der Informationsverarbeitung auftreten, resultiert aus den *existierenden klinischen Informations-Systemen.*

2.4.2 Beispiele

Zugriffsobjekte

*Das **Abrufen eines Laborbefundes** beinhaltet das Lesen von Ergebnissen labortechnischer Untersuchungen zusammen mit dem Untersuchungsdatum, dem Untersuchungsverfahren, der Einheit, der Einordnung bezüglich der Normalwerte, den Patientenidentifikationsdaten, der Zuordnung zum zugehörigen Verwaltungsfall sowie Identifikation, Datum und Art der entnommenen Probe.*

*Bei der **Patientenaufnahme** werden zunächst alle elektronischen Krankenakten anhand von definierten Identifikationsdaten auf die Zugehörigkeit zum Patienten überprüft. Existiert eine Krankenakte für den Patienten, so werden Daten zu einer neuen Behandlung oder Behandlungsphase, wie zum Beispiel das Aufnahmedatum, die Station und die Aufnahmediagnose erfasst. Existiert noch keine Krankenakte das Patienten, so werden zusätzlich alle notwendigen Personendaten, Namen, Adresse, Krankenversicherungsdaten usw. erzeugt.*

Subjekte

*Subjekte im Sinne der Definition sind z. B.: alle Personen, die zum **medizinischen Personal** des Klinikums gehören, Personen, die zum **Personal der***

Patientenverwaltung *gehören,* ***EDV-Mitarbeiter*** *der Kliniken und der zentralen Institutionen, die klinische Systeme betreuen, sowie alle* ***Systeme****, die in den Organisationseinheiten des Klinikums betrieben werden und die patientenbezogene Daten austauschen oder auf Datenbestände anderer Organisationseinheiten zugreifen können. In Zukunft werden, z. B. im Zusammenhang mit klinikübergreifenden Kooperationen sowie multizentrischen Studien, auch externe Personen oder Systeme eine Rollen spielen.*

2.4.3 Relationen

Die oben aufgeführten Zusammenhänge zwischen Entitäten ergeben die konkreten Relationen der Zugriffsontologie. Abb. 2.4 zeigt die Relationen zwischen den Entitäten sowie deren Kardinalitäten.

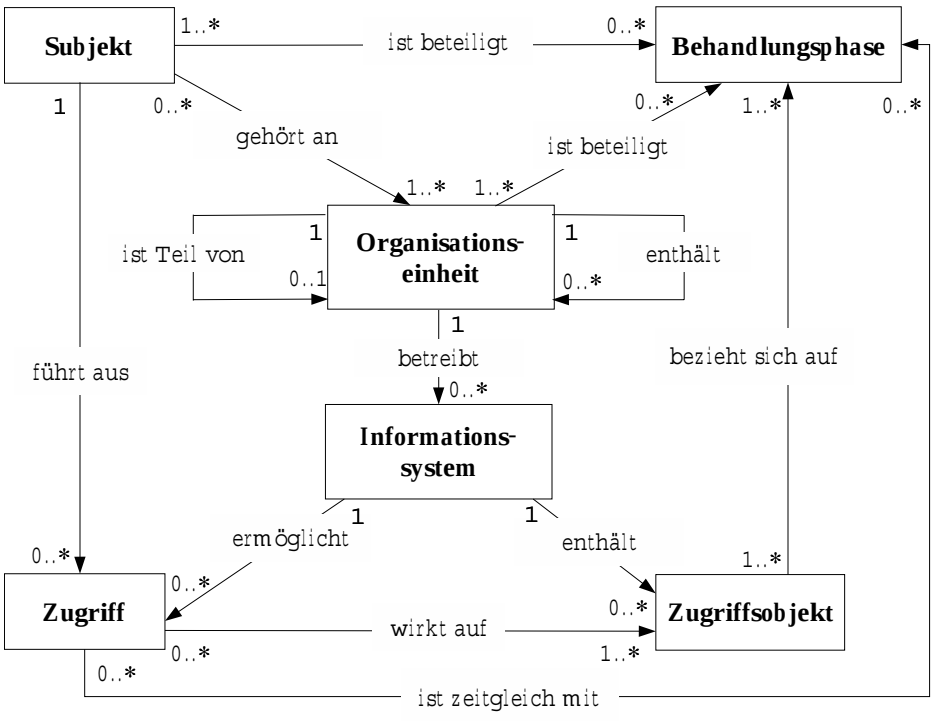

Abbildung 2.4: Entity-Relationship-Diagramm der Zugriffs-Ontologie

Zugriffsobjekt – Informationssystem – Zugriff

- Ein Zugriffsobjekt ist Ziel von Zugriffen; ein Zugriff kann auf verschiedene Zugriffsobjekte wirken (m:n).

- Ein Informationssystem enthält Zugriffsobjekte; jedes Zugriffsobjekt ist aber in genau einem Informationssystem enthalten (1:n).

- Ein Informationssystem ermöglicht verschiedene Zugriffe; jeder Zugriff wird innerhalb eines Informationssystems ausgeführt (1:n).

Zugriffsobjekt – Behandlungsphase – Zugriff

- Zugriffsobjekte enthalten Informationen zu mindestens einer Behandlungsphase; eine Behandlungsphase kann in mehreren Zugriffsobjekten referenziert werden (n:m).

- Ein Zugriff kann zeitgleich mit aktuellen Behandlungsphasen stattfinden. (1c:n)

Subjekt – Organisationseinheit – Behandlungsprozess

- Organisationseinheiten können Teil einer übergeordneten Einheit sein (n:1c) und mehrere untergordnete Organisationseinheiten enthalten (1:nc).

- Subjekte gehören zu wenigstens einer Organisationseinheit des Klinikums; zu einer Organisationseinheit gehören in der Regel mehrere Subjekte (n:m).

- An jeder Phase des Behandlungsprozess ist mindestens eine Organisationseinheit und ein Subjekt beteiligt; in der Regel sind Organisationseinheiten und Subjekte an mehreren Phase (insbesondere verschiedener Patienten) beteiligt (n:m).

Implizite Relationen

- Aufgrund der hierarchischen Struktur der Organisationseinheiten werden die Relationen von Organisationseinheiten zu Subjekten und Informationssystemen auch an alle übergeordneten Organisationseinheiten vererbt.

- Die Beteiligung an einer Behandlungsphase vererbt sich in ebenfalls an die übergeordneten Einrichtungen.

- Wieweit aus der Zugehörigkeit eines Subjekts zu einer Organisationseinheit die wechselweise Implikation der Beteiligung des Subjekts und der Beteiligung der Organsationseinheit an der Behandlungsphase folgt, ist umgebungsabhängig (z. B. von der Gestaltung der Verantwortlichkeiten in einer Klinik).

2.5 Zugriffsrechte

Die bisherige Analyse umfasst Zugriffe und ihre Beziehung zu den Strukturen des Klinikums, berücksichtigt aber nicht die Frage nach der Berechtigung eines Zugriffs. Die Zugriffspolitik ist die Menge aller Aussagen zur Zulässigkeit von Zugriffen. Zugriffsrechte dienen dazu, die Zugriffspolitik in formal überprüfbare Aussagen zu übersetzen. Sie referenzieren dabei die Menge der möglichen Zugriffe, d. h. implizit die Menge der Subjekte und der Zugriffsobjekte. Die Menge der möglichen Zugriffsobjekte hängt vom Informationsverarbeitungsprozess im Klinikum ab, d. h. von Funktionen und Informationen, die in den Systemen heute zur Verfügung stehen und in Zukunft zur Verfügung stehen werden. Im folgenden Abschnitt werden die Anforderungen an das Modell der Zugriffsrechte formuliert und die daraus resultierenden Strukturen beschrieben. Das schließt auch Anforderungen an die Verwaltung von Zugriffsrechten ein.

> **Ein Zugriffsrecht** enthält die Erlaubnis (oder das explizite Verbot) einen Zugriff, eine Menge oder eine Abfolge von Zugriffen durchzuführen.

Betrachtet man die Dynamik des Informationsverarbeitungs-Prozesses, so ist offensichtlich, dass die zu referenzierenden Zugriffsobjekte sehr kurzfristig entstehen und sich verändern. In kurzen Zeitintervallen werden elektronische Krankenakten erzeugt, verändert und fortgeschrieben. Neue Informationssysteme beinhalten weitere Patientendaten und Funktionen. Jeder Schichtwechsel des Personals und jeder Funktionswechsel einer Person verändert die Menge der Behandlungsbeziehungen und damit der aktuell gültigen Zugriffsrechte. Wollte man in den Zugriffsrechten explizit die zum Zeitpunkt vorhandenen Zugriffsobjekte und die gerade aktiven Subjekte referenzieren, so müsste man bei jeder Aktion des Informationsverarbeitungsprozesses und des Behandlungsprozesses sowie bei jedem Personalwechsel die Zugriffsrechte anpassen. Es ist leicht einzusehen, dass dieses Vorgehen im Klinikumsbetrieb bei der Menge der anfallenden Daten und der beteiligten Personen und Systeme zeitlich und organisatorisch nicht handhabbar ist. Daraus ergibt sich u. a. folgende Anforderung an das Berechtigungssystem: Zugriffsrechte müssen a priori, also unabhängig von der Existenz der Daten und den zugreifenden Personen abstrakt definiert werden können. Dazu müssen geeignete Klassifikationskriterien gefunden werden, durch die Zugriffe bzw. Mengen oder Abfolgen von Zugriffen abstrakt definiert werden können. Die Klassifizierung der Zugriffe ist implizit eine Klassifizierung der Subjekte und der Zugriffsobjekte, deren Struktur und Beziehungen zu anderen Entitäten im folgenden mit dem Ziel abstrakter Definitionen analysiert werden. Abb. 2.5 zeigt die Ontolgie der Zugriffsrechte, die sich aus dieser Systematik ergibt.

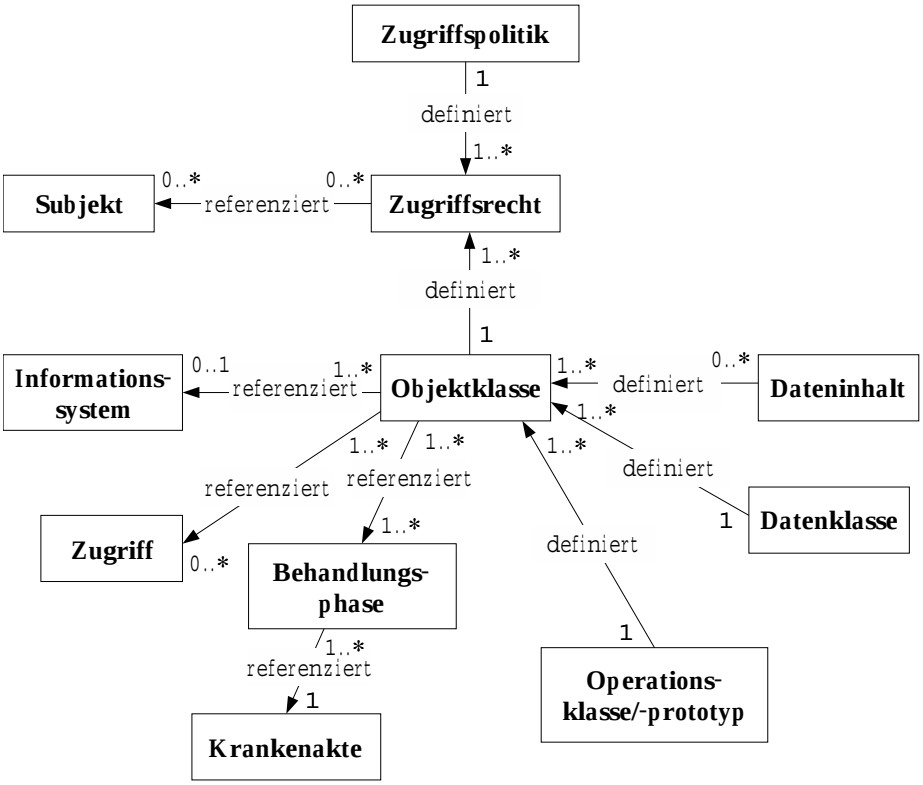

Abbildung 2.5: Die Ontologie der Zugriffsrechte

2.5.1 Klassifikation der Zugriffsobjekte

Für jedes Objekt lassen sich die Klassifikationskriterien in vier grundlegende Bereiche einordnen, nämlich Kriterien bezüglich der Zugriffsoperation, des zeitlichen Ausschnitts der elektronischen Krankenakte, der betroffenen Informationsarten und der Relationen zu anderen Entitäten.

Operationsprototypen

Die Menge der real ausführbaren Operationen auf den elektronischen Krankenakten ist durch die Funktionalität der vorhandenen Systeme gegeben. Es existieren viele semantische Äquivalente von Funktionen gleichermaßen in verschiedenen Informationsystemen, z. B. das *Anlegen einer neuen Krankenakte*, die *Dokumentation eines Ereignisses, einer Maßnahme oder eines Ziels*, die *Einsicht in die Krankenakte* eines Patienten, die *Abfrage von Listen* wie der Sta-

tionsbelegung oder *Kommunikationsfunktionen*. Diese Funktionen stehen in jedem System für die gleichen Informationsverarbeitungsprozesse und gehören zu den gleichen Ereignissen der realen Welt. Systemspezifisch sind i. a. die zugrundeliegenden Datenmodelle, die internen Kontrollstrukturen und beispielsweise die Benutzersicht und -steuerung. Für die semantische Klassifizierung der Datenoperationen[2] braucht man ein allgemeines Informationsverarbeitungsmodell. Jede Klasse von Operationen gleicher Bedeutung wird darin durch einen Operationsprototyp vertreten. Operationsprototypen können auf verschiedenen Abstraktionsebenen definiert sein und zueinander in Beziehung stehen. Sie können aus einfacheren Operationsprototypen zusammengesetzt sein.

Beispiele:

- *„Dokumentation eines Ereignisses“ ist eine abstrakte Funktion. Sie enthält z. B. keine Annahmen über die betroffenen Datenstrukturen oder -sichten. „Dokumentation der Aufnahmediagnose“ ist eine konkretere Funktion, die von der abstrakteren Funktion abgeleitet sein kann.*

- *„Anlegen einer Krankenakte“ ist eine komplexe Funktion. Sie besteht aus verschiedenen Funktionen, die teilweise in definierter Reihenfolge ausgeführt werden müssen. „Anlegen eines neuen Patienten“ ist eine einfachere Funktion, die Teilfunktion der komplexen Funktion sein kann.*

Die Operationsprototypen des Informationsverarbeitungsmodells müssen hinsichtlich ihrer Wirkung auf die elektronische Krankenakte und der erforderlichen Rahmenbedingungen systemunabhängig, aber unmissverständlich beschrieben werden, damit eine Zuordnung der Systemfunktionen möglich ist. Die Zuordnung muss explizit – sozusagen „manuell“ – erfolgen, da sie die genaue Kenntnis der Systeme voraussetzt. Die Klassifizierung der Zugriffsobjekte nach Operationsprototypen ist dann folgendermaßen definiert:

> Für einen Operationsprototypen P_I des Informationsverarbeitungsmodells I gilt: Ein Objekt ω ist Element der Objektklasse Ω_{P_I}, wenn die Zugriffsoperation von ω semantisch durch P_I repräsentiert werden kann.

Die Wahl des Abstraktionsgrads der Operationsprototypen für die Definition der Objektklassen ist eine Abwägung zwischen Handhabbarkeit und Mächtigkeit der Ausdrücke. Prototypen spezifischer Operationen definieren ebenso spezifische Objektklassen. Um die Menge aller möglichen Klassen zu beschreiben, müssen viele spezifische Prototypen definiert werden. Ihre Beschreibung ist detailliert, sie sind wenig robust gegen Änderungen von Funktionen und aufwendig zu verwalten. Abstrakte Operationsprototypen definieren unspezifische Objektklassen. Sie sind für die Umsetzung der Zugriffspolitik u. U. nicht aussagekräftig

[2] Gemeint sind alle möglichen oder a priori denkbaren Datenoperationen.

genug. Deshalb sollten Operationsprototypen in eine Abstraktionshierarchie gebracht werden, so dass der Aufwand für die Definition der Objektklassen durch geignete Wahl der Abstraktionsebene bei hinreichender Aussagekraft minimiert ist.

Datenklassen

Durch den Operationsprototypen sind oft die betroffenen Daten der Krankenakte implizit festgelegt (z. B. *„Laborwerte abrufen“*). Darüber hinaus kann die Objektklasse durch die Angabe spezifischer Teilmengen der Daten genauer festgelegt werden. Dazu müssen auch die Daten der Krankenakte klassifiziert werden.

Die Krankenakte enthält Daten über Patienten und Behandlungen, die je nach System unterschiedlich benannt und strukturiert sind. Bezeichnungen mit derselben Bedeutung wie z. B. „*Familienname*“ und „*Nachname*“ bilden eine „Synonymklasse“. Daten, die inhaltlich zusammengehören, bilden Kategorien, wie z. B. „*Patientenadresse*“. Kategorien können selbst Kategorien enthalten, beispielsweise die Kategorie „*Patientenstammdaten*“ enthält die Kategorie „*Patientenadresse*“. Kategorien mit Schnittmengen sind ebenfalls möglich, z. B. eine Kategorie „*Patientenidentifikationsdaten*“ und eine Kategorie „*Patientenstammdaten*“, die beide den „*Familiennamen des Patienten*“ enthalten. Synonymklassen und Datenkategorien werden im weiteren allgemein als Datenklassen bezeichnet. Wie bei den Operationsprototypen muss für jede Datenklasse eine unmissverständliche Beschreibung ihrer Bedeutung exstieren, d. h., die Datenklassen benötigen ein geeignetes Informationsmodell.

Die Klassifikation der Zugriffsobjekte nach Datenklassen hat folgende Definition:

> Für eine Menge von Datenklassen Δ gilt: Ein Objekt ω ist Element der Objektklasse Ω_Δ, wenn **alle** Attribute von ω Elemente aus Δ sind.

Bei der Definition der Objektklassen ist der Ausschluss von Objekten, bei denen auch noch weitere als die angegebenen Datenklassen vom Zugriff betroffen sind, eine wesentliche Voraussetzung für die Modellierung exakter Zugriffsrechte.

Beispiele:

- *Die Kategorie „Identifikationsdaten“ soll den Namen des Patienten, seinen Geburtsnamen, sein Geburtsdatum, seinen Geburtsort, seinen aktuellen Wohnort und seine Versicherungskarten-Nummer enthalten.*
- *Der Patientenname setzt sich aus dem Familiennamen, allen Vornamen, dem Titel sowie möglichen vor- oder nachgestellten Namenszusätzen zusammen.*

- *Objekte, die zusätzlich die volle Adresse des Patienten enthalten, können in diesem Fall nicht zur oben genannten Klasse der „Identifikationsdatenobjekte“ gehören.*

Daten-Ausprägungen

Die Klassifikation nach Ausprägungen setzt voraus, dass die Objekte Attribute der entsprechenden Datenklasse enthalten.

> Für eine Datenklasse δ und einen Wert ψ oder eine Wertemenge Ψ gilt: Ein Objekt ω ist Element der Objektklasse $\Omega_{\delta=\psi}$ bzw. $\Omega_{\delta\in\Psi}$, wenn es ein Attribut aus δ besitzt, dessen Ausprägung oder Wert gleich ψ ist bzw. in Ψ liegt.

Beispiele:

- *Objekte, die Diagnosen enthalten, können nach bestimmten Diagnosencodes, beispielsweise für psychische Erkrankungen, Geschlechtskrankheiten oder AIDS, klassifiziert werden, wenn die Zugriffspolitik einen speziellen Umgang mit diesen Informationen fordert.*
- *Kriterium für die Klasse der „Prominentenobjekte“, kann die Existenz eines VIP-Kennzeichens sein.*

Relationen zu anderen Entitäten

Durch die Ontologie sind die Relationen zwischen der Entität Objekt und anderen Entitäten beschrieben. Die Objektklassen sind durch Ausprägungen der jeweiligen Relationen definiert.

> Für eine Instanz λ oder eine Menge von Instanzen Λ einer Entität gilt: Ein Objekt ω ist Element der Objektklasse Ω_λ bzw. Ω_Λ, wenn es die Instanz λ oder ein Element aus Λ referenziert.

Zugriffsobjekte besitzen Relationen zu Behandlungsphasen, zu Informationssystemen und zu Zugriffen, die auf das Objekt bereits stattgefunden haben. Infolgedessen gibt es Klassen von Zugriffsobjekten zu Behandlungsphasen von Patienten, Klassen von Objekten eines Informationssystems und Klassen von Objekten im gleichen Zustand des Objekt-Lebenszyklus (*Hat ein bestimmter Zugriff auf das Objekt durch ein bestimmtes Subjekt stattgefunden?*). Ausprägungen von Relation zu Behandlungsphasen sind i. a. keine konkreten Behandlungsphasen, sondern selbst wieder – a priori – abstrakt definierte Mengen von Behandlungsphasen. Das wird insbesondere aus den Zugriffsregeln in Abschnitt 2.1.7 deutlich.

Beispiele:

- *Die elektronische Krankenakte des Patienten „Hans Muster“ ist die Klasse der Objekte, die sich auf den Patienten „Hans Muster“ beziehen. Objekte, die sich auf Behandlungsphasen in der Kinderklinik beziehen, bilden ebenfalls eine Klasse.*
- *Die Klasse der Objekte, die im Verwaltungssytem gespeichert sind.*
- *Die Klasse der Objekte, die von „Dr. X. Nahtlos“ nach dem 1.1.2000 erzeugt wurden.*

2.5.2 Verwaltung der Zugriffsrechte

Die Definition und Verwaltung der Zugriffsrechte fällt in unterschiedliche Zuständigkeitsbereiche. Die Zugriffspolitik stellt die Vergabe der internen Zugriffsrechte in die Verantwortung der Fachkliniksleiter. Außerdem finden administrative Vorgänge an verschiedenen Stellen des Klinikums statt: Dienstpläne werden in den Fachkliniken und Abteilungen erstellt, die Einstellung von Personal wird von Mitarbeitern der Personalabteilung bearbeitet. In Verbindung mit diesen Vorgängen werden Informationen zu Personen (Subjekten) und ihren Zuständigkeiten (Rechte) erfasst. Das heißt, Subjekte und Zugriffsrechte sollen durch verschiedene Personen an verschiedenen Stellen verwaltet werden können. Diese administrativen Zugriffe auf Subjekt- und Berechtigungsinformation müssen – ebenso wie die Zugriffe auf Patientendaten – kontrolliert werden. Zur Unterscheidung werden im folgenden die Subjekte der Administration **„Administratoren“** und die Zugriffsobjekte **„Administrationsobjekte“** genannt. Die Struktur der **„Administrationsrechte“** ist analog zur Definition der Zugriffsrechte in 2.5. Administrative Zugriffe beziehen sich auf Administrationsobjekte, welche spezifische Operationen, Daten und Relationen enthalten.

Beschreibung der Administrationsobjekte

Administrationsobjekte sind die Subjekte und Zugriffsrechte des Zugriffskontrollsystems zusammen mit den darauf definierten Operationen. Ihre Struktur und ihre Inhalte ergeben sich aus der Problemstellung, d. h. aus der Administration der Zugriffsrechte für Zugriffe auf Patientendaten.

Subjekte und Subjektklassen: Die Subjektdaten enthalten eindeutige Identifikationsmerkmale, die Qualifikation des Subjekts, die Zugehörigkeit zu Organisationseinheiten des Klinikums und die jeweilige Funktion in diesen. Darüberhinaus können weitere Informationen wie z. B. Telefonnummern, Lokalisation, und E-Mail-Adressen erfasst werden. Subjekte können nach diesen Eigenschaften klassifiziert werden, z. B. nach der Zugehörigkeit zu Organisationseinheiten, nach Funktionen oder nach Qualifikation. Die Subjekt-Verwaltung besteht im Wesentlichen aus dem Erfassen, Verändern und Löschen von Subjektdaten sowie dem Zuweisen und Entziehen von Rechten.

Zugriffsrechte: Zugriffsrechte enthalten Referenzen auf Zugriffsobjektklassen. Sie können danach klassifiziert werden, wozu sie berechtigen, z. B. zu Zugriffen auf Objekte bezüglich bestimmter Behandlungsphasen oder zum Zugriff auf bestimmte Systeme. Bereits angelegte Rechte können auch nach ihrem Autor klassifiziert werden. Die Operationen der Rechte-Verwaltung sind Anlegen und Löschen von Rechten sowie die Zuordnung der Objektklassen.

Anforderungen an ein Rechtemanagementsystem

Aus der Struktur der Zugriffsrechte (für Zugriffe auf Patientendaten) sowie aus den Rahmenbedingungen des klinischen Alltags, unter denen sie definiert und vergeben werden, resultiert nicht nur die Struktur der Administrationsrechte, sondern auch Anforderungen an die Funktionsweise eines Rechtemanagement-Systems. Diese und einige grundlegende Anforderungen an die Rechteadministration sind im folgenden aufgeführt:

Verteilte Administration Die Zugriffspolitik definiert einerseits Regelungen für abteilungsübergreifende Zugriffe, andererseits enthält sie auch Kriterien für abteilungsinterne Zugriffsregelungen, deren Umsetzung in der Verantwortung der Fachkliniks- bzw. Abteilungsleiter liegt. Abteilungsübergreifende Rechte sollen deshalb von einem zentralen Administrator definiert und vergeben werden, abteilungsspezifische Zugriffsrechte werden innerhalb der Fachabteilungen durch lokale Administratoren definiert und vergeben. Ein Rechtemanagementsystem muss also Strukturen für die Definition global wie lokal gültiger Zugriffsrechte sowie Werkzeuge für die Administration und Zuweisung der Rechte durch verschiedene Administratoren an unterschiedlichen Orten enthalten. Ein Szenario verteilter Administration der Zugriffsrechte ist in Abbildung 2.6 dargestellt.

Nutzung vorhandener Resourcen Für die Definition der Rechte werden Informationen über Personal und Systeme benötigt, die u. U. bereits in anderen Zusammenhängen verwendet werden. Im Mainzer Klinikum liegen solche Informationen – wenn auch unvollständig – im zentralen Benutzerverzeichnis der EDV-Abteilung vor. Dieses Benutzerverzeichnis sollte mit dem Ziel der Aufwandsminimierung und der Konsistenzerhaltung in die Konzeption einbezogen werden.

Redundanz- und Aufwandsminimierung Die Mengen der Zugriffsrechte verschiedener Subjekte oder bezüglich verschiedener Funktionen besitzen häufig Schnittmengen. Zur Minimierung des Definitionsaufwands und zur Verhinderung von Konsistenzverletzungen bei der Umsetzung der Zugriffspolitik sollten Rechte unabhängig von Subjekten definiert, untrennbare Rechtemengen gruppiert und diese Rechtegruppen zu funktionsspezifischen Rechtemengen zusammengestellt werden können. Subjekte sollten dann Inhaber dieser Rechtemengen werden, solange sie die entsprechende Funktion ausüben.

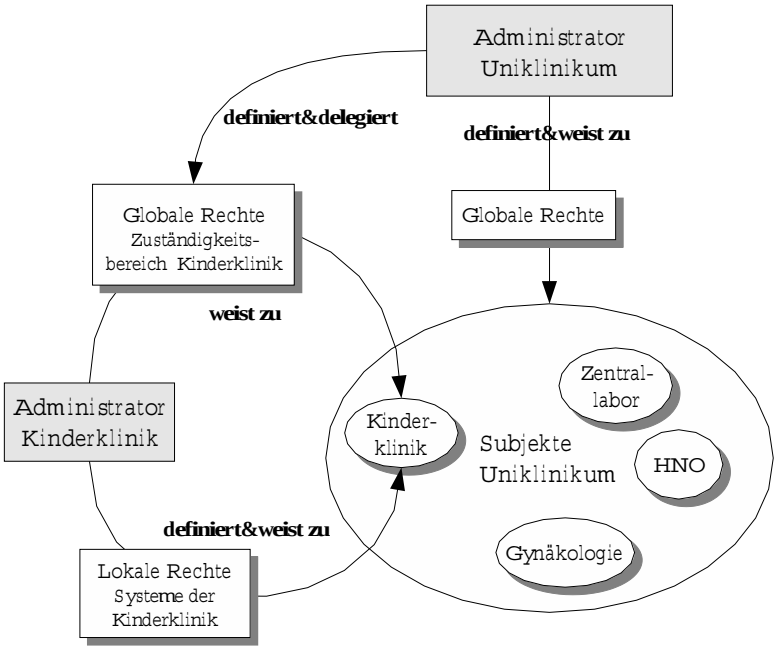

Abbildung 2.6: Administrations-Szenario

Wiederspruchsfreiheit Bei der Vergabe der Rechte müssen diese auf Widersprüche zu existierenden Rechten überprüft werden können.

2.6 Zugriffsentscheidungen

Zugriffsentscheidungen werden dort umgesetzt, wo Zugriffe ermöglicht werden, in den Informationssystemen. Die meisten der klinischen Informationssysteme haben eigene Zugriffskontrollmechanismen und unterscheiden verschiedene Benutzer und Benutzerprofile. Ein Konzept, das die Zugriffskontrolle ausschließlich den Informationsystemen überlässt, hat folgende Nachteile und Unzulänglichkeiten:

Redundantes und aufwendiges Benutzermanagement: Jedes System registriert alle Benutzer, die potentiell Zugriff auf die Objekte bekommen sollen. Dabei werden viele Benutzer in mehreren Systemen, i. a. mit verschiedenen Benutzerkennungen erfasst. Benutzerverzeichnisse verursachen Pflegeaufwand; vor allem müssen Benutzer, die die Klinik verlassen, zuverlässig entfernt werden. Bei Informationsservern, auf die Benutzer aus dem gesamten Klinkum zugreifen sollen, wird das Benutzermangement aufgrund des Umfangs sehr aufwendig.

Multiple Authentisierungsmerkmale: Benutzerverzeichnisse können in der Regel nicht miteinenader synchronisiert werden, d. h., dass die Benutzer einen Reihe von Systemzugängen mit verschiedenen Authentisierungsmerkmalen besitzen; derzeit findet die Authentisierung in den meisten Fällen mit Benutzerkennung und Passwort statt. Eine größere Menge dieser Authentisierungsmerkmale ist für den einzelnen Benutzer schwer (im Kopf!) zu verwalten.

Ungenügende Entscheidungskriterien: Wie man aus der Zugriffspolitik sehen kann, benötigen Zugriffsentscheidungen für einrichtungsübergreifende Zugriffe Informationen über den Behandlungsprozess. Die meisten Systeme verfügen nur über Informationen zu Teilen des Behandlungsprozesses, in der Regel eine Einrichtung bzw. Organsiationseinheit betreffend. Unter diesen Umständen können lokal keine korrekten Zugriffsentscheidungen getroffen werden.

Aufwendige Qualitätskontrolle: Zugriffsentscheidungen, die lokal in den Systemen meistens auf der Basis fest programmierter Algorithmen getroffen werden, können nur schwer kontrolliert werden. Man kann in der Regel nicht verhindern, das in verschiedenen Systemen, verschiedene u. U. widersprüchliche oder unzureichende Zugriffsentscheidungen getroffen werden.

Die genannten Nachteile können vermieden werden, wenn man die Zugriffspolitik in einem zentralen System umzusetzt, das Zugriffsinformation bzw. Zugriffsentscheidungen in nutzbarer Form an alle Systeme zur Verfügung stellt. Vor allem das Benutzermanagement sollte einheitlich und redundanzfrei organisiert sein.

Für dieses Vorhaben müssen zunächst die Möglichkeiten der vorhandenen und künftigen Systeme in Bezug auf die Verarbeitung externer Zugriffsentscheidungen und den Zugriff auf externe Verzeichnisse betrachtet werden.

2.6.1 Rahmenbedingungen der Zugriffskontrolle

Die verschiedenen Systeme haben unterschiedliche Bedürfnisse an verfügbare Zugriffsinformation und -entscheidungen, die in den folgenden Beispielszenarien erläutert werden:

Beispiel „Spezialsystem“: Systeme mit speziellen Aufgaben wie beispielsweise der Steuerung eines diagnostischen Geräts, die nur einem eingeschränkten Kreis von Benutzern Zugriff geben und deren Benutzer die gleichen Funktionen durchführen können, benötigen nur wenig Informationen für Zugriffsentscheidungen. Sie brauchen ausschließlich Kenntnis über die grundsätzliche Zugehörigkeit zum autorisierten Benutzerkreis.

Beispiel „Zentral genutzter Befundserver“: Die potentiellen Benutzer eines zentralen Befundservers sind z. B. alle Ärzte des Klinikums. Auf welche Befunde ein Arzt zu einem Zeitpunkt zugreifen darf, hängt vom Behandlungsprozess ab. Üblicherweise verfügt der Befundserver nicht über

die vollständige Information zu den Behandlungsprozessen. Er benötigt deshalb bei jedem Zugriff Information über den aktuellen Zustand des Behandlungsprozesses bzw. über die daraus resultierenden Rechte des Benutzers in der Funktion, in der er zugreift. Wenn der Befundserver nur eine Art von Zugriff (z. B. das Lesen der Befunde) anbietet, genügt für die Zugriffsentscheidung die Kenntnis der Fälle, an denen der Arzt beteiligt ist.

Beispiel „Abteilungssystem“: Abteilungssysteme enthalten Daten zu allen Patientenkontakten mit der Abteilung. Sind alle Benutzer gleichermaßen in alle Behandlungen in der Abteilung involviert, so werden keine externen Informationen zum Behandlungsprozess benötigt. Abteilungssysteme können aber unterschiedliche Funktionen und Datensichten für verschiedene Benutzerkreise anbieten. Hier benötigt das System Informationen über den autorisierten Benutzerkreis, und welcher Benutzer welche Funktion ausüben darf.

Neben der Frage, welche Informationen ein System für seine Zugriffsentscheidungen benötigt, stellt sich auch eine zweite, auf welchem Weg nämlich das System diese Informationen bekommen und wie es sie verarbeiten kann. Folgende Fälle hinsichtlich der Möglichkeit eines Informationsystems, Zugriffsentscheidungen zu treffen und Zugriffsinformationen zu verarbeiten, konnten identifiziert werden:

1. Zugriff auf ein externes Benutzerverzeichnisse (**ja**/**nein**)
2. Unterscheidung funktionaler Benutzergruppen (**ja**/**nein**)
3. Differenzierte Zugriffsrechte bezüglich
   (a) Funktionen (**interne**/**externe**/**keine** Definitionsmöglichkeit)
   (b) Datensichten (**interne**/**externe**/**keine** Definitionsmöglichkeit)
4. Berücksichtigung von Zuständigkeiten der einzelnen Benutzer für bestimmte Krankenakten

   (**interne**/**externe**/**keine** Verarbeitung von Informationen zum Behandlungsprozess oder aktueller Zuständigkeiten)

In den Fällen, in denen ein System weder externe Informationen berücksichtigen kann noch interne Informationen und Strukturen besitzt, die die Umsetzung der Zugriffspolitik für die spezifischen Systemfunktion gewährleistet, kann keine befriedigende Zugriffskontrolle stattfinden. Für alle anderen Fälle muss das Zugriffskonzept die Lösung vorsehen, bei der die Zugriffspolitik mit angemessenem Aufwand umgesetzt werden kann. Die Rahmenbedingungen der verschiedenen Systeme müssen in den Anforderungen an das Zugriffskontrollsystem berücksichtigt werden.

2.6.2 Anforderungen an die Repräsentation der Zugriffsrechte

Die Form, in der Informationssystemen Berechtigungsinformation vermittelt wird, entscheidet darüber, ob diese korrekt und mit vertretbarem Aufwand für die internen Zugriffsentscheidungen genutzt werden kann. Wie oben beschrieben, gibt es verschiedene Informationsverarbeitungsprozesse, die Berechtigungsinformation in unterschiedlicher Konkretheit und unterschiedlicher Komplexität benötigen. Es erscheint nicht sinnvoll, Berechtigungsinformation ausschließlich in der maximal möglichen Detailliertheit anzubieten. Ein System, das eine bezüglich der Berechtigungen homogene Funktionalität beinhaltet, benötigt nur die Information, ob ein Benutzer grundsätzlich zum Zugriff auf das System autorisiert ist. Ein System, das verschiedene Funktionalitäten unterscheidet, die

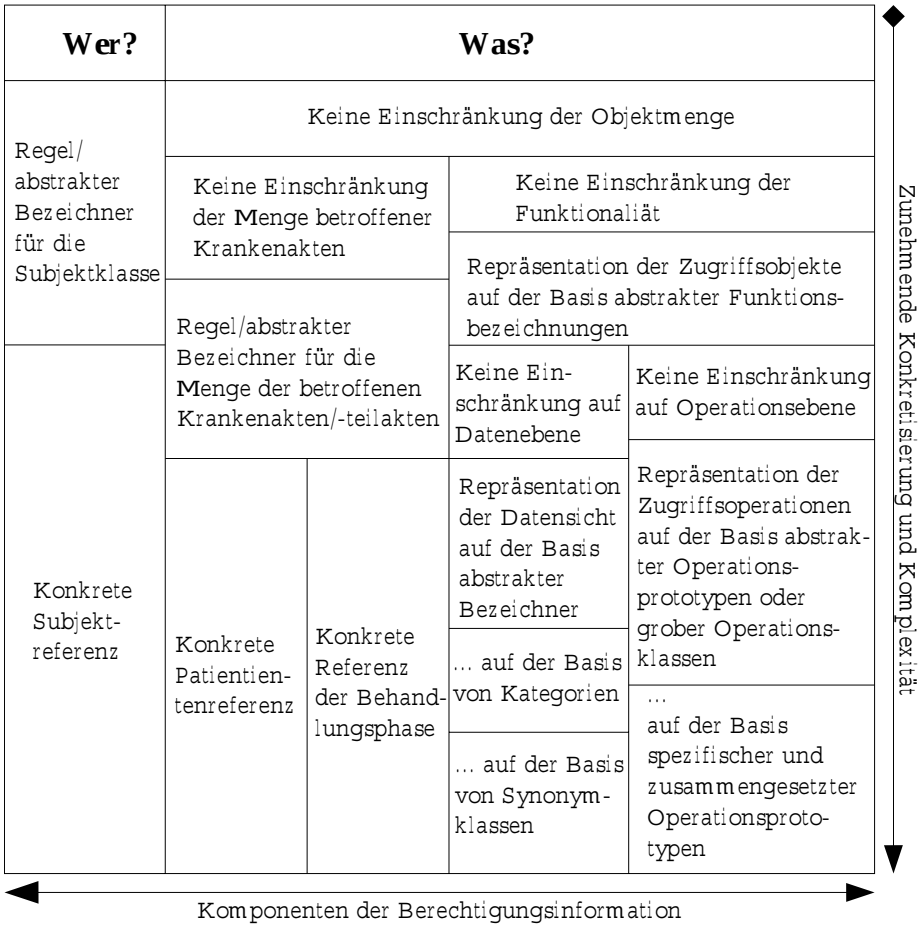

Abbildung 2.7: Abstraktionsebenen der Berechtigungsinformation

bestimmten Personenkreisen zugeordnet sein sollen, kann allein mit der Information, welche Funktion ein angemeldeter Benutzer hat, die richtige Zugriffsentscheidung treffen. Die möglicherweise ungenügende Übereinstimmung mit der Zugriffspolitik der Einrichtung bei einem kommerziellen System ist ein Problem, das auf dieser Ebene nicht gelöst werden kann.

Die Berechtigungsinformation soll auf jeder Abstraktionebene und in jeder Zusammensetzung angeboten werden, die ein System für seine Zugriffsentscheidungen benötigt. Abbildung 2.7 zeigt die verschiedenen Ebenen, in denen Berechtigungsinformationen ausgedrückt werden können, und die verschiedenen Komponenten, die in beliebiger Kombination der Ebenen zusammengesetzt sein können.

Die Forderung nach Berechtigungsinformation in unterschiedlicher Granularität und Zusammensetzung wirkt sich auch auf die Definition der Zugriffsrechte aus. Wenn nicht letztendlich für jedes einzelne System spezifische Rechte in der benötigten Feinheit und Zusammensetzung definiert werden sollen, was dem Gedanken einer einheitlich definierten Zugriffspolitik widersprechen und außerdem beachtlichen Aufwand verursachen würde, dann muss das Informationsmodell der Zugriffsrechte zusammen mit den darauf operierenden Retrievalfunktionen so konzipiert sein, dass es Berechtigungsinformation in jeder benötigten Feinheit und Zusammensetzung aus einer einheitlichen Definition effizient ermitteln und präsentieren kann. Eine wesentliche Voraussetzung hierfür ist z. B. die Abbildung der hierarchischen Beziehungen zwischen den Bestandteilen der Rechte sowie die Verwendung einheitlicher Begriffe. Andernfalls ist es schwierig, in den Zugriffsrechten Widersprüche, Unvollständigkeiten oder Verstöße gegen die Zugriffspolitik zu erkennen. Das heißt nicht, dass nicht in Einzelfällen auch spezifische Rechte für einzelne „besonders schwierige“ Systeme zugelassen werden sollten.

2.7 Anforderungen an ein zentrales Zugriffskontrollsystem

Die Aufgabe eines Zugriffskontrollsystems ist das Ermitteln, Sammeln und Kombinieren von Wissen, das von den Informationsystemen für Zugriffsentscheidungen benötigt wird, um es diesen in geeigneter Form verfügbar zu machen. Dazu sind verschiedene Komponenten und Schnittstellen erforderlich.

2.7.1 Komponenten und Schnittstellen für die Wissenakquisition

Langsam veränderliches Wissen wie die Einrichtungstruktur oder die Fakten und Regeln der Zugriffspolitik müssen über geeignete Benutzerschnittstellen erfasst werden können und im System verfügbar sein. Wissen, das häufigen Veränderungen unterliegt, z. B. die konkreten Behandlungsprozess-Instanzen und Funktionszuweisungen an Personen, ensteht im Arbeitsablauf, teilweise in anderen

Informationssystemen. Für dieses Wissen sind verschiedene automatische Methoden der Akquisition möglich:

Passive Akquisition Das System enthält eigene Repositorien, die ereignisgesteuert das relevante aktuelle Wissen aus den Prozessen erhalten, und Wissen für die Ermittlung der dynamischen Berechtigungsinformation zur Verfügung stellen.

Aktive Akquisition Das Zugriffskontrollsystem besitzt Schnittstellen zu Systemen, in denen Prozesswissen verfügar ist, und fragt das aktuelle Wissen zum Zeitpunkt der Zugriffsentscheidung ab.

Beispiele:

- *Im Patientendatenverwaltung-System SAP/IS-H werden alle Patientenaufenthalte erfasst. Es enthält die vollständige Aufenthaltshistorie aller Patienten des Klinikums seit Juli 1999. Ein Datenbestand aller Fälle seit Januar 1995 existiert in der Referenzdatenbank. Diese erhält außerdem kontinuierliche Aktualisierungen über die Kommunikationsschnittstelle von IS-H. Ein Zugriffskontrollsystem kann das benötigte Wissen aus den Nachrichten der Kommunikationschnittstelle erfassen und speichern oder den bereits vorhandenen Datenbestand der Referenzdatenbank abfragen.*

- *Das Patientenverwaltungs-System enthält auch Listen der Bezeichner für Stationen, Ambulanzen und Leistungsstellen und ihre Zuordnung zu Fachkliniken. Diese Bezeichner werden auch in den Patientendatensätzen referenziert. Mit Hilfe eines automatisierten Verfahrens müssen alle Änderungen der Bezeichner und Zuordnungen zu einer Aktualisierung der erfassten Einrichtungsstruktur führen.*

2.7.2 Retrievalmechanismen

Wann ein Informationssystem während des Wissensverarbeitungsprozesses Zugriffsentscheidungen trifft und welche Information es dazu benötigt, kann sehr unterschiedlich sein. Grundsätzlich gibt es aktive und passive Retrievalmechanismen, die je nach Bedarf und auch in Kombination eingesetzt werden können:

Passiver Mechanismus Das zugreifende Subjekt bringt bei der Anmeldung bereits eine Zusammenstellung von Subjekt-Informationen, insbesondere eine Liste seiner Zugriffsrechte mit, welche im Laufe der Sitzung für die Zugriffsentscheidungen ausgewertet wird. Für dieses Verfahren muss die Integrität der Subjekt-Information sichergestellt sein. Besteht die Berechtigungsinformation des Subjekts aus einer großen Menge komplexer Zugriffsrechte und oder besitzen die Rechte nur eine kurze Gültigkeitsdauer, so ist dieses Verfahren nicht geeignet.

Aktiver Mechanismus Das zugreifende Subjekt authentisiert sich nur mit seiner Kennung beim Informationsystem. Mit Kenntnis der Subjektidentität kann das Informationssystem für jede Zugriffsentscheidung aktuelle Berechtigungen des Subjekts abfragen. Inhalt der Abfrage kann eine Zugriffsentscheidung sein, d. h. ein expliziter Zugriff wird auf seine Zulässigkeit überprüft, worauf das Zugriffskontrollsystem eine positive oder negative Antwort gibt. Es kann aber auch sinnvoll sein, Abfragen von Mengen zulässiger Zugriffe zu ermöglichen, um so beispielsweise die effiziente Erstellung von Auswahllisten zu unterstützen. Bei einem aktiven Abfragemechanismus ist die Vertrauenswürdigkeit des Zugriffskontrollsystems bzw. seiner Schnittstellen wichtig.

Hybrider Mechanismus Das zugreifenden Subjekt bringt einen Teil der spezifischen Information bei der Anmeldung bereits mit, wie beispielsweise seine Funktion und Einrichtungszugehörigkeit. Weitere notwendige Berechtigungsinformation, z. B. dynamische Informationen, werden zur Zugriffszeit vom Informationssystem erfragt.

2.7.3 Interoperabilität

Die Schnittstellen zu verschiedenen anderen Systemen erfordern ein hohes Maß an Interoperabilität und Kommunikationsmöglichkeiten. Interoperabilität ist am besten erreichbar durch den Einsatz von standardisierten Technologien und Protokollen. Heutige klinische Informationssysteme bieten i. a. keine expliziten Mechanismen zur Integration externer Zugriffsinformation an. Bestenfalls wird die Nutzung eines externen Benutzerverzeichnisses auf der Basis des Lightweight Directory Protocols (LDAP) unterstützt. Deshalb werden Verfahren benötigt, die die Integration der Zugriffsinformation einfach und mit verhältnismäßig geringem (auch finanziellem) Aufwand ermöglichen. Informationssysteme, die externe Authentisierungs- und Zugriffskontrollverfahren nutzen können, unterstützen damit auch Single-Sign-On-Verfahren, so dass ein Benutzer sich nur einmal im verteilten Kliniksinformationssystem anmelden muss und dann während einer Sitzung gemäß seinen persönlichen Rechten auf verschiedene Subsysteme zugreifen kann.

2.7.4 Sicherheit

Auch für den Betrieb eines Zugriffskontrollsystems sind Sicherheitsmaßnahmen erforderlich. Am wichtigsten ist die Integrität der Zugriffsentscheidungen. Diese dürfen weder hinsichtlich der zugrundeliegenden Daten, der Regeln oder Algorithmen noch bei der Übermittlung manipulierbar sein und müssen inklusive der Entscheidungsgrundlagen nachvollzogen werden können. Nächstwichtiger Aspekt ist die Verfügbarkeit. Ohne Berechtigungsinformation können keine korrekten Zugriffsentscheidungen getroffen werden und somit keine kontrollierten Zugriffe stattfinden. Die Verfügbarkeit des Zugriffskontrollsystems ist ein empfindlicher Punkt im gesamten Kliniksinformationssystem. Vertraulichkeit ist

dort wichtig, wo direkt oder indirekt Personendaten verarbeitet werden. D. h. Personaldaten und Daten über Behandlungsprozesse, bei denen ein Patientenbezug hergestellt werden kann, müssen vor unbefugter Einsichtnahme geschützt werden. Hierbei gelten die gleichen Anforderungen an System- und Kommunikationssicherheit wie für klinische Informationssysteme im allgemeinen.

2.7.5 Wartbarkeit

Die Wartbarkeit eines Systems hängt u. a. davon ab, mit welchem Aufwand es an Veränderungen angepasst werden kann. Das betrifft vor allem die Komponenten, die Wissen über Umgebungsspezifika enthalten, wie z. B. die Einrichtungsstruktur des Klinikums und die Struktur des Behandlungsprozesses. Wird Umgebungswissen auf allgemeine generische Strukturen zurückgeführt, die mit Hilfe geeigneter Erfassungswerkzeuge spezifiziert werden können, so sind Veränderungen der Umgebung einfacher mitzuvollziehen. Der Aufwand für die Entwicklung und Anfangskonfiguration eines solchen Systems ist entsprechend größer. Die Frage, welcher Aufwand sinnvoll und gerechtfertigt ist, muss mit Hilfe einer Abschätzung der zu erwartenden Umgebungsveränderungen beantwortet werden. Im Fall der Einrichtungsstruktur des Klinikums ist eine einfache Erweiterung und Aktualisierung sinnvoll; so wurden beispielsweise im Mainzer Klinikum innerhalb des Jahres 2000 mehrere neue Stationen und Ambulanzen ergänzt. Die Struktur des Behandlungsprozesses wird i. a. nicht oder unwesentlich verändert. Hier muss sorgfältig abgewogen werden, ob z. B. mit Hilfe eines Workflow-Management-Systems die Struktur des Behandlungsprozesses manipuliert werden können soll. Die Regeln der Zugriffspolitik sind i. a. keinen schnellen Veränderungen unterworfen. Sie ändern sich mit den der Zugriffspolitik zugrundliegenden Gesetzen oder deren Interpretation. Übergangsregeln allerdings, die Teil einer „abgeschwächten Zugriffspolitik“ sind, haben keine lange Gültigkeitsdauer. Die Architektur des Zugriffskontrollsystems soll grundsätzlich ermöglichen, mit möglichst geringem Aufwand einzelne Teilebereiche zu ändern oder zu erweitern, oder zusätzliche Funktionalität zu ergänzen. Deshalb sollten Teilbereiche des Zugriffskontrollsystems, die unabhängig von anderen Bereichen Bedeutung und Funktionalität besitzen, als eigenständige Komponenten realisiert werden. Die Kopplung der Komponenten erfolgt dann über definierte Schnittstellen.

2.8 Anforderungen an die Protokollierung

Die Protokollierung von Zugriffen muss in den Systemen stattfinden, auf die Zugriffe ausgeführt werden. Vorgaben zur Protokollierungshäufigkeit und zum Umfang der Protokollierung verschiedener Zugriffsarten werden aber in der Zugriffspolitik allgemein verbindlich festgelegt. So wird für kontrollierte Lesezugriffe höchstens eine stichprobenartige Protokollierung in Verbindung mit Mechanismen zur Erkennung von auffälligen Abfragemustern (z. B. systematische Massenabfragen) gefordert. Bei Schreibzugriffen muss hingegen immer und mit

allen notwendigen Details protokolliert werden, so dass jede Veränderung von Informationen mit Zeitpunkt und Autor der Änderung nachvollzogen werden kann. Unkontrollierte Lesezugriffe im Notfallmodus müssen ebenfalls vollständig protokolliert werden; wahrscheinlich genügt es aber betroffene Patienten(teil)akten und grobe Datenkategorien festzuhalten, um so im Nachhinein die Zulässigkeit überprüfen und gegebenenfalls dem Patienten Auskunft über den Zugriff geben zu können.

Zu jedem Zugriffsrecht gehört also auch die Information über Häufigkeit und Umfang der Zugriffs-Protokollierung. Diese Information muss beim Entwurf der Informationsstrukturen sowie bei den Retrievalfunktionen berücksichtigt werden.

Kapitel 3

Zugriffskontrollmodelle

Seit den Anfängen von Computersystemen mit gemeinsamer Ressourcen-Nutzung durch verschiedenen Personen und Prozesse ist das Thema „Automatische Zugriffskontrolle“ Gegenstand der informatischen Forschung. Vor allem aus dem Bereich des Militärs und der Sicherheitsdienste stammen umfassende Zugriffskontrollmodelle und -lösungen[34, 41]. Inzwischen gibt es eine Reihe von Ansätzen, die für verschiedene Bedürfnisse wie z. B. die Kontrolle von Informationsflüssen, von Datenspeichern oder anderen Ressourcen entwickelt wurden.

Die folgenden Abschnitten geben einen Überblick über grundlegende Zugriffskontrollmechanismen und Zugriffspolitiken. Rollenbasierte Zugriffskontrollmodelle werden – als anerkannter Ansatz für die Zugriffskontrolle im Gesundheitswesen – besonders ausführlich beschrieben und diskutiert. Im Anschluss folgenden einige Beispiele publizierter Zugriffskontroll-Lösungen in Systemen des Gesundheitsbereichs.

3.1 Zugriffskontrollmechanismen

Das Grundkonzept der Zugriffskontrolle ist die Zugriffskontrollmatrix. In ihr sind alle Zugriffsberechtigungen abgelegt. Für jedes Subjekt existiert eine Zeile und für jedes Objekt eine Spalte. Ein Matrixeintrag definiert das Zugriffsrecht eines Subjekts bezüglich eines Objekts. Auf der Basis dieser Matrix kann jeder angeforderte Zugriff durch ein entsprechendes Zugriffssteuerungsmodul überprüft und erlaubt oder verweigert werden. Für die Umsetzung einer Zugriffskontrollmatrix gibt es verschiedene Möglichkeiten. Da die Matrix i. a. sehr umfangreich, aber nicht vollständig ausgefüllt ist, werden oft nur die relevanten Zeilen oder nur die Spalten der Matrix implementiert. Die Spalten beinhalten alle erlaubten Zugriffe von Subjekten auf ein bestimmtes Objekt und werden als „Access Control Lists (ACL)“ bezeichnet. Der dazu orthogonale Ansatz sind die sogenannten „Capabilities“, die Zeilen der Matrix, die zu einem Subjekt alle erlaubten Zugriffe auf Objekte festlegen. Beide Ansätze erlauben die redundanzfreie Speicherung der Zugriffsrechte und werden als Teil der Subjekt- bzw. der Objektinformation

gehalten. Der Preis für diese Modellierung der Informationen ist, dass bei bestimmten Fragestellungen entweder alle Objekte oder alle Subjekte überprüft werden müssen. ACLs können vereinfacht werden, indem man statt einzelner Subjekte Subjektgruppen berechtigt. Diese können auch durch u. U. komplexe Regeln erweitert werden, die genau festlegen, wann und wie welche Klassen von Benutzern zugreifen dürfen. Die Zugriffskontrollmechanismen in den heute gängigen Betriebssystemen basieren auf Access Control Lists. Die Kombination beider Ansätze eignet sich besonders für die Zugriffskontrolle in verteilten Systemen. Benutzer müssen sich nur einmal authentisieren[1] und erhalten dabei ihre „Capability List“, die sie zum Zugriff auf bestimmte Dienste des verteilten Systems berechtigt. Eine feinere Steuerung der Zugriffe kann jeder einzelne Dienst mittels ACLs regeln. Ein weiterer Ansatz basiert auf der Repräsentation der Zugriffskontrollmatrix als Berechtigungsrelationen oder -tabellen. Jeder nicht-leere Eintrag der Matrix ergibt ein Tripel, aus Subjekt, Zugriffsmodus und Objekt. Berechtigungsrelationen können je nach Bedarf nach Subjekten oder Objekten sortiert durchsucht werden. Sie benötigen im Gegensatz zu ACLs und Capabilities eine explizite Verwaltung – z. B. beim Löschen von Subjekten oder Objekten, da sie nicht direkt an Subjekte bzw. Objekte gekoppelt sind.

3.2 Zugriffspolitiken

Zugriffspolitiken definieren die Regeln und Kriterien, nach denen Zugriffsrechte vergeben und Zugriffsentscheidungen getroffen werden. In der Literatur sind vor allem drei Arten von Zugriffspolitiken beschrieben, „Discretionary Policies“, „Mandatory Policies“und „Role-Based Policies“ [34, 41]. Diese Ansätze haben sich in der Vergangenheit aus den Anforderungen von militärischen und anderen Multi-User-Informationssystemen entwickelt.

„Discretionary Policies“ Discretionary heißt „beliebig“ und bedeutet in diesem Zusammenhang, dass für jedes Subjekt bzw. jede Gruppe von Subjekten und jedes Objekt im System die zulässigen Zugriffsarten vom Eigentümer des Objekts nach eigenem Belieben vergeben werden. Bei der Erzeugung eines Objekts ist der Erzeuger immer zunächst auch der Eigentümer, solange bis er die Eigentümerschaft an ein anderes Subjekt überträgt. Bei jedem Zugriff werden Subjekt oder Subjektgruppe und Zugriffsart mit den gesetzten Rechte verglichen. Man unterscheidet offene und geschlossene Policies, je nachdem ob jede nicht explizit erfasste Kombination von Subjekt, Objekt und Zugriffsart als Berechtigung oder als Verbot interpretiert wird. Die Definition von sowohl positiven als auch negativen Berechtigungen ist schwer zu handhaben und widerspruchsfrei zu halten.

Der Ansatz stammt aus der Zugriffskontrolle in Dateisystemen. Zugriffsarten sind „Lesen“, „Schreiben“ und – für ausführbare Dateien –

[1] „Single-Sign-On“

„Ausführen“. Hinter den Zugriffsarten stecken die konkreten Betriebssystemfunktionen read(), write() und execute(). Dieser Ansatz schützt konkrete Objekte, die Kontrolle von Informationsflüssen ist mit damit aber nicht möglich, da die Informationen der Willkür des neuen Besitzers unterliegen, sobald sie z. B. durch einen Kopiervorgang weitergegeben wurden.

„Mandatory Policies“ Das Wesen der Mandatory Policies ist, Zugriffsrechte in Form von Regeln sozusagen „von oben“ verbindlich festzulegen. Der Ansatz eignet sich insbesondere zur Kontrolle von Informationsflüssen und stammt aus dem militärischen Bereich. Subjekte und Objekte werden dabei klassifiziert und mit einer Sicherheitsstufe assoziiert, die den Grad der Vertrauenswürdigkeit bzw. der Sensibilität festlegt. Die Urheberschaft der Informationen hat in der Regel keine Bedeutung für die Berechtigungen. Zugriffsrechte werden für jede Zugriffsart über das Verhältnis der Sicherheitsstufen von Subjekt und Objekt definiert. Darüberhinaus können Subjekte und Objekt in Kategorien eingeordnet werden, die Zugriffe auf einen Bereich einschränken – eine Annäherung an das Need-To-Know-Prinzip.

„Role-Based Policies“ Rollenbasierte Zugriffspolitiken haben sich aus Anwendungen entwickelt, wo Discretionary Policies zu wenig restriktiv sind und wo komplexere Aktivitäten von Subjekten im Vordergrund stehen, die sich nicht einfach in ein Gitter von Sicherheitsstufen projizieren lassen. Role-Based Policies haben sich aus dem „Mandatory“-Ansatz entwickelt. Rollen beinhalten die Berechtigung zu Aktionen, die zu einer Aufgabe gehören. Dabei werden zunächst keine Annahmen über die Spezifikation der Aktionen gemacht. Subjekte sind Inhaber einer oder mehrerer Rollen und können sich in der Rolle anmelden, die ihrer aktuellen Aufgabe entspricht. Durch geeignete Modellierung von Rollen kann das Prinzip der minimalen Rechte umgesetzt werden. Der rollenbasierte Ansatz ermöglicht verschiedene Administrationsmodelle - von der zentralen Administration der Rollen und Rechte bis hin zu einer delegierten, hierarchischen Administration.

Für die Anforderungen an die Zugriffskontrolle in Kliniken sind sowohl der „Discretionary“-Ansatz als auch der gitterbasierte „Mandatory“-Ansatz wenig geeignet. Der „Discretionary“-Ansatz verursacht hohen Verwaltungsaufwand für die gezielte Freigabe von Zugriffen und legt die gesamte Verantwortung in die Hände des jeweiligen Informationsautors, i. a. des behandelnden Arztes. Sicherheitsgitter reichen hingegen nicht aus, um die Dynamik der Aktionen eines Subjekts innerhalb der Behandlungsprozesse abzubilden. Von den aufgeführten Ansätzen ermöglicht der rollenbasierte Ansatz am besten die Abbildung der verschiedenen – mitunter wechselnden – Funktionen von Subjekten im Klinikum einschließlich der Dynamik der Behandlungsprozesse bei geeigneter Klassifizierung der Aktionen. Eine ausführliche Untersuchung des rollenbasierten Ansatzes durch das National Institute of Standards and Technology [36] hat dessen Eignung für die Zugriffskontrolle im Gesundheitsbereich bestätigt. Da der rollenbasierte Ansatz auch in dieser Arbeit präferiert wird, folgt in Abschnitt 3.3 ein Überblick über

die rollenbasierten Modelle nach Ravi Sandhu [39, 41, 40], der diesen Ansatz in einer Reihe von Veröffentlichungen umfassend behandelt hat.

3.3 Rollenbasierte Zugriffsmodelle

3.3.1 Was ist eine Rolle?

Der Rollenbegriff findet sich in der Literatur mit Variationen in folgender Form:

> Eine **Rolle** ist „die bezeichnete Menge von [Benutzern und] Rechten und möglicherweise anderen Rollen".

Im Mittelpunkt der Betrachtung stehen die Rechte, die zur Ausübung einer Funktion, Aufgabe oder Verantwortlichkeit benötigt werden. Deshalb findet man Defintionen des Rollenbegriffs, die den Benutzer zunächst ausklammern. Eine Benutzergruppe ist im Gegensatz dazu in erster Linie *„eine bezeichnete Menge von Benutzern und möglicherweise weiteren Benutzergruppen"*, die sinnvollerweise gemeinsame Rechte besitzen. Rollen- und Gruppenbegriff schließen sich gegenseitig nicht aus; sie beschreiben vor allem unterschiedliche Sichten auf die Rechte-Benutzer-Relation.

Folgende Eigenschaften des Rollenbegriffs sollen Grundlage für das hier entwickelte Zugriffsmodell sein:

- Eine Rolle steht für eine Funktion innerhalb einer Institution und beinhaltet die dazu benötigten Rechte.
- Eine Rolle existiert zunächst als abstraktes Konstrukt ohne die Notwendigkeit einer Zuweisung realer Personen oder konkreter Rechte.
- Rollen vereinfachen die Rechteverwaltung durch die Entkopplung und Strukturierung der Berechtigungs-Relationen.

3.3.2 Das RBAC96 Modell von R. Sandhu

Ravi Sandhu et al entwickelten in den 90er Jahren ein mehrstufiges rollenbasiertes Zugriffskontrollmodell, das unter dem Titel RBAC96 [40] veröffentlicht wurde. Es bietet eine systematische anwendungsunabhängige Grundlage für den Aufbau anwendungsspezifischer rollenbasierter Zugriffskontrollsysteme.

Das Kernmodell

Das RBAC-Kernmodell ($RBAC_0$) beinhaltet die Grundstrukturen Rolle (role), Subjekt (user) und Recht (permission) sowie die grundlegenden Relationen zwischen den Strukturen. Die Art, wie Rechte definiert werden, ist nicht näher spezifiziert, da sie stark anwendungsabhängig ist.

Folgende Zuordnungen und Funktionen beschreiben die Relationen zwischen Rollen und Subjekten, bzw. Rollen und Rechten:

„user role assignment“ (URA) Jedes Subjekt kann mehrere Rollen annehmen und jede Rolle kann mehreren Subjekten zugeordnet sein (m:n-Relation).

„permission role assignment“ (PRA) Jede Rolle kann mehrere Rechte beinhalten und jedes Recht kann in verschiedenen Rollen enthalten sein (m:n-Relation).

„session“ Der Begriff „session“ oder „Sitzung“ definiert eine Funktion, die ein Subjekt für einen Zeitraum an eine oder mehrere Rollen bindet, in denen es agiert und für die es autorisiert ist.

Die Strukturen des $RBAC_0$-Modells sind in allen weiteren RBAC-Modellen enthalten. Das Kernmodell beinhaltet keine Zuordnung von Rollen zueinander.

Hierarchien

Das $RBAC_1$-Modell ist eine Erweiterung des RBAC-Kernmodells durch Hinzunahme von Rollenhierarchien. Die Rollenhierarchie ist eine spezielle Zuordnung von Rollen zueinander, bei der Rechte vererbt werden. Die mächtigere „senior role“ erbt von einer oder mehreren[2] weniger mächtigen „junior roles“. Mit diesem Modell lassen sich vor allem hierarchische Kompetenzstrukturen abbilden. Ein wichtiges strukturelles Konzept in der Rollenhierarchie sind „private roles“. Von einer abstrakten Rolle (z. B. „*Arzt*“) wird eine Rolle für die nächsthöhere Kompetenzstufe (z. B. „*Oberarzt*“) und eine konkrete Rolle der gleichen Kompetenzstufe (z. B. „*Privatperson-Arzt*“) mit zusätzlichen Rechten, die der Vorgesetzte nicht teilt (z. B. „*Einblick in persönliche Dokumente*“), abgeleitet. Diese Konstruktion erlaubt die Erweiterung der klassischen hierarchischen Kompetenzstruktur.

Constraints

$RBAC_2$ differenziert das $RBAC_0$-Modell mit Hilfe von definierten Einschränkungen, sogenannten „Constraints“, die auf jeden Aspekt des allgemeinen Rollenmodells angewendet werden können. Das beinhaltet beispielsweise den generellen Ausschluss bestimmter Zuordnungen (URA, PRA), den wechselseitigen Ausschluss von Zuordnungen sowie Mengenbeschränkungen von Zuordnungen (Kardinalitätsconstraints). Constraints ermöglichen die Formulierung von Vorbedingungen für Zuordnungen, wie z. B. die Zuordnung einer Rolle zu einem Akteur unter der Bedingung, dass ihm bereits eine bestimmte andere Rolle zugeordnet wurde. Dynamische Constraints kontrollieren die Menge der gleichzeitig in einer oder in verschiedenen Sitzungen durch einen oder verschiedene Akteure aktivierten Rollen. Damit wird festgelegt, welche Rollen generell nicht gleichzeitig oder nicht gleichzeitig von ein und derselben Person aktiviert werden dürfen und welche Rollen nur in einer maximalen Anzahl von Sitzungen gleichzeitig aktiv sein sollen. Man verhindert beispielsweise, dass verschiedene

[2] Das Modell sieht multiple Vererbung vor.

Personen bestimmte Rechte gleichzeitig besitzen oder ausüben (z. B. die Änderung von Rechten während aktiver Sitzungen).

Consolidated Model

Das $RBAC_3$-Modell integriert Rollenhierarchien und Rollenconstraints unter Berücksichtigung von Wechselwirkungen zwischen beiden Paradigmen, die zu Widersprüchen führen können. Es erlaubt die Formulierung von Constraints bezüglich der Rollenhierarchie wie z. B. die Beschränkung der Anzahl von unter- und übergeordneten Rollen, oder den Ausschluss gemeinsamer unter-/übergeordneter Rollen. Folgende Fragestellungen veranschaulichen die Wirkung von Constraints auf die Rollenhierarchie:

- Ist ein wechselseitiger Ausschluß von Rollen mit gemeinsamer „senior role“ möglich?
- Wie zählen vererbte Rollen bei der Umsetzung von Kardinalitätsconstraints bezüglich der Zuordnung von Rollen zu Akteuren (URA)?

Das Modell schreibt keine Lösung vor, sondern zeigt nur Wechselwirkungen und potentielle Widersprüche auf. Es bleibt eine Frage der Zugriffspolitik, welche der möglichen Varianten umgesetzt wird.

Allgemeine Anmerkungen zur Anwendung

Bis auf wenige explizite Vorgaben bleiben die RBAC-Modelle offen für die spezifische Ausgestaltung bei der Umsetzung einer Zugriffspolitik. Eine Vorgabe ist, dass das Subjekt innerhalb einer Sitzung nicht geändert werden kann. Damit ist sichergestellt, dass alle Aktionen eindeutig personenbezogen nachvollziehbar bleiben. Inwieweit das Subjekt in einer Sitzung seine Rollen wechseln darf, kann mit Hilfe geeigneter Constraints festgelegt werden. Negative Berechtigungen sind explizit nicht vorgesehen. Sandhu et al weisen darauf hin, dass man stattdessen geeignete Constraints definieren kann. Darüberhinaus fehlen in diesem Ansatz bislang sequenzielle bzw. temporale Abhängigkeiten zwischen Rechten.

3.3.3 ARBAC97 Administrative Model

Das „Admininstrative RBAC Model“ von Sandhu et al. (ARBAC97) ermöglicht eine komfortable und übersichtliche Rollen- und Rechteverwaltung, auch wenn die Rollenhierarchie so groß und komplex ist, dass sie von einem Sicherheitsadministrator nicht mehr gut verwaltet werden kann. ARBAC97 ist eine Anwendung des rollenbasierten Zugriffsmodells auf die Administration eines rollenbasierten Zugriffskontrollsystems. Die Verteilung von Administrationsaufgaben auf Administrator-Rollen ermöglicht eine dezentrale Rollen- und Rechteverwaltung ohne die zentrale Kontrolle über die grundlegende Zugriffspolitik aufzugeben.

Die Administrator-Rollen selbst werden von einem hauptverantwortlichen Sicherheitsadministrator ("chief security officer") verwaltet. Eine wesentliche Forderung ist die Trennung von Administrator-Rollen und Benutzer-Rollen, eine konsequente Umsetzung des „Separation of Duty"-Prinzips. Die Administrationsaufgaben untergliedern sich in die drei Bereiche „Zuweisung von Rechten zu Rollen" (PRA), „Zuweisung von Subjekten zu Rollen" (URA) und „Zuweisung von Rollen zueinander" (RRA). Für jeden der drei Bereiche sind die Administrator-Zugriffe eindeutig festgelegt:

URA: Subjekt-Rollen-Zuweisungen und deren Rücknahme ("can-assign"- und „can-revoke"-Recht) sind hier die spezifischen Zugriffe. Durch Vorbedingungen ("prerequisite conditions"), die sich auf weitere Zuweisungen (URAs, PRAs und RRAs) beziehen, wird definiert, für welche Rollen und Subjekte diese Zugriffe durchgeführt werden dürfen. Bei der Rücknahme von Zuweisungen müssen Konsequenzen auf die Rollenhierarchie berücksichtigt werden. Sogenannter starker Rückzug entzieht dem Subjekt auch alle abgeleiteten Rollen ("senior roles") während schwacher Rückzug nur die explizite Rolle entzieht.

PRA: PRA enthält Funktionen analog zu URA, wobei durch Vorbedingungen bezüglich der Rollenhierarchie eingeschränkt ist, welchen Rollen Rechte zugeordnet oder genommen werden dürfen. Starker Rückzug eines Rechts bedeutet hierbei, dass das Recht auch allen „junior roles" genommen wird.

RRA: Für die Administration der Rollehierarchie werden zunächst drei Arten von Rollen unterschieden: „Abilities", für die es nur Rechte-Zuweisungen, aber keine Subjektzuweisungen gibt, „Groups", für die es keine Rechte-Zuweisungen, aber Subjektzuweisungen gibt, und „UP"-Roles"[3], für die es beide Arten von Zuweisungen gibt. Abilities beschreiben untrennbare Funktionen, die als Einheit zugewiesen werden sollen. Unter Groups sind homogene Subjektgruppen zu verstehen, die durch entsprechende Vererbung die gleichen Rechte erhalten. UP-Roles sind die abgeleiteten Rollen, die für konkrete Rechte konkreter Subjekte stehen. Sinn der Aufteilung ist eine bessere Vorstrukturierung und Verteilung der Administrationsaufgaben; jede Abteilung kann dadurch ihre eigene Rollenstruktur aufbauen. Die hierzu notwendigen Administrationzugriffe werden durch das „can-modify"-Recht erlaubt. Dieses Recht schließt die Operationen „role creation", „role deletion", „role-role edge insertion" und „role-role edge deletion" ein. Geeignete Constraints müssen die Konsistenzerhaltung der Rollenhierarchie und der Zuweisungen – z. B. beim Löschen von referenzierten Rollen – sicherstellen, und die Wechselwirkungen verschiedener Modifikationen der Rollenhierarchie sind zu berücksichtigen.

[3] User-Permission-Roles

3.3.4 Eignung des Ansatzes

Wie oben erwähnt, gibt es bereits eine umfassende Untersuchung des rollenbasierten Ansatzes auf seine Eignung für die Zugriffskontrolle im Gesundheitswesen. Dieser Abschnitt führt die wesentlichen Stärken aber auch die Grenzen aus der Sicht der Autorin auf.

Stärken

- Mit dem beschriebenen Ansatz werden verschiedene Sicherheitsprinzipien unterstützt, z. B. das Prinzip der minimalen Rechte und die Trennung verschiedener Aufgaben. Jeder Rolle sind genau die Rechte zugeordnet, die für die jeweilige Funktion notwendig sind. Eine andere Rechtezusammensetzung kann, falls notwendig, Bestandteil einer anderen Rolle sein. Konflikte, die durch gleichzeitige Ausführung oder Besitz von verschiedenen Rollen durch ein Subjekt entstehen, können vermieden werden.

- Die Modellierung von Funktionen als Rollen ermöglicht eine kontextorientierte Zuweisung von Rechten (ein explizites Recht kann in einem Zusammenhang gegeben sein, in einem anderen nicht).

- Das Sitzungskonzept ermöglicht die Wahl der kleinsten notwendigen Rechtemenge für einen Zeitraum. Benutzer sind Rolleninhaber (statische Zuweisung), und sie befinden sich in einer aktuellen Sitzung bezüglich einer Rolle (dynamische Zuweisung). Damit lassen sich Funktionswechsel und zeitlich verteilte Funktionen, beispielsweise „*der diensthabende Stationsarzt*“, abbilden. Falls notwendig kann durch ein Kardinalitätsconstraint sichergestellt werden, dass bespielsweise nur eine Person als verantwortlicher diensthabender Stationsarzt angemeldet sein kann, um so Konflikte durch konkurrierende Zugriffe zu vermeiden.

- Die Modellierung einer Rollenhierarchie vereinfacht die Rollenverwaltung und bildet bestehende Hierarchien ab (insbesondere die Kompetenzbereiche der einzelnen Kliniken).

- Durch „private roles“ kann die hierarchische Struktur erweitert werden. Beispiel: „*Der Chefarzt darf zwar im Prinzip alles, was ein Stationsarzt darf, aber wahrscheinlich soll er nicht jederzeit dessen private Arbeiten lesen dürfen. Die private Rolle* $Stationsarzt_{Privat}$ *wird ebenso wie die Chefarztrolle von der allgemeinen Rolle Stationsarzt abgeleitet und um die entsprechenden Rechte erweitert.*“

- Voraussetzungen für die Rollenzuweisungen können mit Hilfe von Constraints und abstrakten Rollendefinitionen modelliert werden. Beispiel: „*Jeder approbierte Arzt wird automatisch einer abstrakten Rolle „Arzt“ zugeordnet. Für alle Rollen, die zu speziellen ärztlichen Funktionen berechtigen, kann diese Zuweisung als Vorbedingung gesetzt werden.*“

- Mit Hilfe des Administrativen Rollenmodells kann eine kontrollierte dezentrale Rollenverwaltung umgesetzt werden. Die Administration der Zugriffsberechtigungen sind von den Funktionen der klinischen Informationsverarbeitung getrennt.

- Die relative Unabhängigkeit der drei Zuweisungsschichten liefert eine große Flexibilität für Änderungen und Weiterentwicklungen. Geänderte Regelungen können beispielsweise durch geeignete Definition von Rechten und deren Zuweisung an bestehende Rollen umgesetzt werden, ohne die Änderung der Subjekt-Rollen-Zuweisungen notwendig zu machen.

Grenzen

Das rollenbasierte Zugriffsmodell macht keine Annahme über Struktur und Eigenschaften von Subjekten und Rechten. Dementsprechend lässt es einige Probleme der Zugriffskontrolle offen:

- Die Frage nach der Repräsentation von Zugriffsrechten und den dahinterstehenden Zugriffsobjekten muss unabhängig vom Rollenmodell gelöst werden.

- Das Modell enthält kein explizites Modell der Zeit, so dass beispielsweise die Gültigkeitszeit von Zuordnungen außerhalb des Rollenmodells geregelt werden muss. Erweitert man das Rollenmodell um zeitliche Aspekte, so kann man Rollen zu definierten Zeiten für bestimmte Subjekte „aktivieren“.

- Das Administrationsmodell schließt die Definition und Verwaltung von Subjekten und Rechten nicht ein; diese Aufgabe muss zusätzlich geregelt werden.

- Der Aufwand für die Modellierung komplexer Funktionen darf nicht unterschätzt werden; er sollte der Komplexität des Problems angemessen sein. Die Frage, wie Rollendefinition und -administration unterstützt werden können, um diese Aufgaben zu erleichtern und den Aufwand zu minimieren, muss im Zusammenhang mit der Modellierung der Informationsstrukturen und dem Entwurf der Erfassungswerkzeuge beantwortet werden.

3.4 Zugriffskontrolle im Gesundheitswesen – Diskussion verschiedener Ansätze

In der Literatur gibt es eine Reihe von Lösungsansätzen und Beispielen für Zugriffskontrolle im Gesundheitswesen. Die wenigsten der Ansätze beschreiben allerdings Zugriffskontroll-Verfahren, die unabhängig sind von der Integration in ein konkretes Zugriffsverfahren oder System. Die meisten Ansätze verwenden

rollenbasierte Zugriffspolitiken mit Ausnahme der Sicherheitspolitik der British Medical Association (BMA) für den National Health Service (NHS) [4]. Die empfohlene Zugriffspolitik dort ist eine eingeschränkte Discretionary Policy, für die die Veränderung von Rechten stark reglementiert ist; siehe auch Abschnitt 3.4.1. Einige, aber nicht alle Beispiele berücksichtigen die Dynamik des Behandlungsprozesses, beschreiben also eine „kontextabhängige“ Zugriffskontrolle. Die Integration der Zugriffskontrollmechanismen variiert stark. In den folgenden Abschnitten werden einige Ansätze beschrieben und hinsichtlich der gewählten Zugriffspolitik, der Repräsentation der Zugriffsrechte – insbesondere der kontextabhängigen – sowie der Implementation der Zugriffskontrollmechanismen diskutiert.

3.4.1 Die NHS-Zugriffspolitik

Das Grundkonzept der NHS-Zugriffspolitik [4] ist die Patientenakte. Jede Patientenakte enthält eine Sicht auf die Patientendaten, die einer bestimmten Gruppe von Personen uneingeschränkt zugänglich sein soll. Der Patient selbst und der Arzt, der die Akte anlegt, sind grundsätzlich zugriffsberechtigt. Der Arzt ist hauptverantwortlich und darf, nach festen Regeln, den Zugriff auf die Akte an Kollegen weitergeben. Jede Akte besitzt somit eine Zugriffskontrolliste, die die zugriffsberechtigten Personen enthält. Der Informationsfluss zwischen zwei Akten ist explizit geregelt. Alle jemals erfassten Informationen müssen erhalten bleiben und jeder Zugriff wird protokolliert. Die Implementation dieser Zugriffspolitik wird als Aufgabe der Systemhersteller gesehen, muss aber in einer Form erfolgen, die für eine unabhängige Kontrollstelle überprüfbar ist.

Diskussion

Die Zugriffspolitik der NHS enthält bereits Vorgaben für das Informationsmodell und die Verwaltung von Zugriffsrechten, nämlich die Aufteilung der Patientendaten in verschiedene Sichten, die Existenz von expliziten Zugrifskontrolllisten und die Verwaltung der Rechte durch das ärztliche Personal. Sie enhält darüberhinaus konkrete Forderungen an die Hersteller von Informationssystemen im Gesundheitswesen. Die Umsetzung eines derartigen Konzepts ist in einer Einrichtung mit einer heterogenen Menge existierender Systeme nicht durchsetzbar. Die Verwaltung der Zugriffsrechte durch das ärztliche Personal ist ein Ansatz, der aufgrund des damit verbundenen Aufwandes für die Zugriffkontrolle in einem großen Krankenhaus nicht geeignet ist; er ist auf die Zugriffskontrolle in einem Netz von ärztlichen Praxen zugeschnitten, in denen die Fluktuation von Personal und Patienten wesentlich geringer ist, und in denen die Abläufe und Abhängigkeiten weniger komplex sind.

3.4.2 RBAC Demo-Projekt des NIST

Das National Institute of Standards and Technology (NIST) untersuchte in einem Demonstrationsprojekt [36] die Umsetzung rollenbasierter Zugriffspolitiken

für den Gesundheitsbereich. Inhalt der Untersuchungen waren einerseits die Frage nach einsetzbaren Technologien, d. h. Transportmechanismen, Interoperabilitätsstandards, sowie nach der Modellierung und Architektur des Systems, und andererseits die Frage nach der Erfüllbarkeit der Anforderungen im Gesundheitswesen. Das Demonstrations-System beinhaltet den Einsatz von CORBA, OLE/COM, RPC, Web-Techniken und Remote Database Access (RDA). Die Architektur des Systems basiert auf einem Drei-Schichten-Objektmodell, das die Informationsquellen von den Anwendungen trennt. Die Basiszugriffschicht enthält fest definierte Objektmethoden zum Zugriff auf die Informationsquelle, die Applikations-Interface-Schicht enthält die Methoden, die von einer Anwendung aus angesprochen werden können. In der mittleren Schicht sind die Zugriffsregeln mit Hilfe von korrespondierenden Methoden der Rollenobjekte umgesetzt. Hier sind auch die notwendigen Filtermethoden zur Generierung von Datenausschnitten realisiert. Mit dieser Methode bleiben Anwendungen und Informationsquellen weitgehend unabhängig gegenüber Änderungen der Zugriffspolitik.

Der Ansatz setzt eine Client-Server-Architektur voraus. Für jede Kombination von Client (Anwendung) und Server (Informationsquelle) müssen entsprechende Rollenobjekte mit den korrespondierenden Methoden implementiert werden. Die Zugriffskontrollpolitik ist in den Rollenobjekten festgelegt. Bei Aufruf einer Methode überprüft das Rollenobjekt, ob der Rolleninhaber grundsätzlich zugreifen darf und führt die notwendigen Filtermethoden aus. Behandlungskontexte werden nicht explizit berücksichtigt; die Demonstrations-Anwendung lässt implizit alle Rollen des Personals auf alle Akten und Patienten nur auf die eigene Akte zugreifen. Das Beispiel berücksicht nur ein flaches Rollenmodell, keine Hierarchien oder Constraints.

Diskussion

Für die Implementation eines Zugriffskontrollsystems in der vorgestellten Art, müssen für jede Anwendung und ihre Informationsquelle entsprechende Rollenobjekte implementiert werden. In einer Umgebung mit einer Vielzahl vorhandener Systeme, die in ein klinikweites Zugriffskontrollsystem integriert werden müssen, ist dieser Aufwand deutlich zu groß. Eine Umsetzung der rollenbasierten Zugriffspolitik in der Art der Demonstrations-Applikation ist deshalb nicht geeignet.

3.4.3 Beispiele dynamischer Zugriffskontrolle

Dynamische „Working Lists“ – Ansatz von Staccini et al

Staccini et al [43] beschreiben einen Ansatz zur Umsetzung einer Zugriffspolitik, die – vergleichbar der Zugriffspolitik des Mainzer Uniklinikums – für jeden Zugriff den Nachweis eines Behandlungszusammenhangs fordert. Dazu werden ereignisgesteuert sogenannte „Arbeitslisten“ erzeugt, die Beziehungen zwischen Personen bzw. Personengruppen und dem Behandlungs- und Pflegeablauf eines

Patienten enthalten. Die inhaltliche Differenzierung der Zugriffe erfolgt mit Hilfe von drei Benutzerprofilen ("Pflegekraft der Einheit", „Arzt der Einheit", „externer Arzt"), die die erlaubten Zugriffe bezüglich der Behandlungs-Informationen und -Aktionen durch fünf „Need-To"-Relationen ("Need-To-Know", „Need-To-Order", „Need-To-Act", „Need-To-Report" und „Need-To-Communicate") festlegen. Die Ereignisse kommen aus den Arbeitsabläufen der Klinik, also aus Patientenbewegungen, Anforderungen, etc. und müssen in den zugehörigen Systemen (Patientenverwaltung, Order-Entry-Systeme) generiert werden. Information über Benutzer, funktionale Gruppen, Organisationseinheiten und Workstations der medizinischen Einrichtung ist in zentralen Repositorien verfügbar. Nach der Authentisierung über eine lokale Workstation erhält der Benutzer spezifischen Zugriff auf Patientendaten abhängig von den aktuellen Einträgen in den korrespondierenden Arbeitslisten zu seiner Person und der Organisationseinheit, in der sich die Workstation befindet. Über die Verbindung mit anderen Workstations können weitere Fälle abgerufen werden.

Workflow-basierte Autorisierung – Das MobiMed-Projekt

Ultes-Nitsche und Teufel beschreiben die Umsetzung eines kontext-abhängigen Zugriffskontrollsystems für Zugriffe auf Patienteninformation in einer Datenbank unter Verwendung eines Workflow-Management-System [44]. Jeder Behandlungsprozess-Zustand wird mit den beteiligten Benutzergruppen, Daten und Zugriffsarten assoziiert. Bei einer Zugriffsanforderung werden Informationen über den aktuellen Workflow des Patienten ermittelt und zusammen mit der Benutzerinformation und dem Berechtigungsschema ausgewertet. Der berechtigte Benutzer wird einer Datenbank-Benutzergruppe mit uneingeschränktem Zugriff zugewiesen, so dass der angeforderte Zugriff durchgeführt werden kann.

Diskussion

Beide Ansätze beziehen sich auf eine konkrete System- oder Hardware-Architektur. Sie bieten also keine umfassende Lösung für das Integrationsproblem. Bei Staccini et al. sind Benutzerprofile und Zugriffklassen statisch modelliert und erlauben keine unmittelbare Erweiterung. Die inhaltliche Differenzierung der Zugriffe im MobiMed-System erfolgt auf der Basis der Datenbanktabellen, -felder und -funktionen. Beide Ansätze können nicht auf eine heterogene Systemlamdschaft übertragen werden.

Für den Aspekt der dynamischen Autorisierung liefern sowohl die „Arbeitslisten" als auch die workflow-basierten Berechtigungen eine ausreichende Lösung. Die geforderte Umsetzung des „erweiterten Need-To-Know-Prinzips" ist mit diesen Ansätzen möglich. Die automatische Generierung von Arbeitslisten impliziert, dass die Regeln der Zugriffspolitik in den Akquisitionsmechanismen enthalten sind („*Wann wird welche Zuordnung in die Liste eigetragen?*"). Der Wokflow-Ansatz steckt einen Teil der Regeln in die Modellierung des Worklflows („*Nur was als Zustand definiert ist, steht für die Autorisierung*

zur Verfügung!“). Das bedeutet, dass u. U. auch kleinere Änderungen der Zugriffsregelungen Erweiterungen des Modells und der Software erfordern.

3.4.4 Integration über zentrale Register

Das InterCare-Projekt

Potamias et al präsentieren einen Ansatz für die Zugriffskontrolle in einer hochgradig heterogenen Umgebung, nämlich dem Gesundheitsnetz der Region „Kreta“ [37]. Grundidee ist der Aufbau eines zentralen „Patient Clinical Data Dictionary“ (PCDD), der einheitlich strukturierte Informationen zu allen Patientenkontakten in verschiedenen Einrichtungen enthält. Jedes integrierte Informationssystem ist über ein Gateway mit dem PCDD verbunden und liefert Daten, die in das zentrale Datenschema übersetzt werden. Der PCDD ermöglicht, die verschiedenenen Teile der verteilten Patientenakte zu lokalisieren sowie kontrolliert auf die enthaltenen Daten zuzugreifen. Der Ansatz beinhaltet eine einfache rollenbasierte Zugriffspolitik. Mit Hilfe eines einfachen Rechte-Editors kann einer Rolle das Zugriffsrecht für Teile einer Patientenakte bezüglich eines registrierten Patientenkontaktes zugewiesen werden. Der Umfang des Zugriffs wird über Benutzerprofile gesteuert. Die für die regional verteilte elektronische Krankenakte definierte Zugriffspolitik gibt dem behandelnden Arzt Zugriff auf die von ihm erzeugten Daten und auf alle Daten zu überwiesenen Patienten. Patienten haben Zugriff auf ihre persönlichen Daten und können Zugriffe gewähren oder einschränken.

Diskussion

Der InterCare-Ansatz bietet eine Möglichkeit der zentral organisierten Zugriffskontrolle. Der einrichtungs- und system-übergreifende Zugriff wird über die Funktionen und das Datenschema eines zentralen Systems gesteuert, das den Anforderungen entsprechend modelliert wurde. Autorisiert werden also nur Zugriffe auf das zentrale Register oder, wenn man das Konzept entsprechend weiterdenkt, Zugriffe, die über das zentrale Register an die jeweiligen Systeme weitergeleitet werden, deren Umfang dann aber nicht mehr vorgegeben ist. Zugriffsmöglichkeiten der Systeme, die über die vorgesehene Funktionalität hinausgehen können von der Zugriffskontrolle nicht erfasst werden. Für die Anforderungen der Zugriffskontrolle im Klinikum bedeutet dieser Ansatz grundsätzlich einen hohen Integrationsaufwand oder stark eingschränkte Zugriffsmöglichkeiten.

3.4.5 CORBAMed – Resource Access Decision Service

CORBAMed ist eine vertikale Facility innerhalb des CORBA Standards [21] und spezifiziert Services, die speziell im Gesundheitswesen benötigt werden, wie z. B. den „Person Identification Service (PIDS)“ und den „Clinical Observation Access Service (COAS)“. Für die Zugriffskontrollproblematik in verteilten

Systemen des Gesundheitswesen beinhaltet CORBAMed einen Service zur Unterstützung von Zuriffsentscheidungen, den „Resource Access Decision Service (RADS)“ [24]. Das Grundkonzept geht vom Zugriff auf ein Zielobjekt durch einen geeigneten Klienten innerhalb einer CORBA-basierten Gesamtarchitektur aus. Das Zielobjekt, dessen Funktionalität nur entsprechend der Zugriffspolitik verfügbar sein soll, ruft die „$access\_allowed()$“-Methode des Access Decision Objects (ADO) auf und erhält einen boolschen Wert als Ergebnis. Dabei werden die Sicherheitsattribute des zugreifenden Subjekts, sowie der Name der angeforderten „Secured Resource“ und der Operation übergeben. Das „AccessDecision“ Objekt kommuniziert mit einem „Dynamic Attribute Service“, der weitere zeitabhängige Sicherheitssttribute ermitteln kann, einem Objekt, das die relevanten Zugriffspolitiken ermittelt (“PolicyEvaluatorLocator“), einem Objekt, das die zugehörige Zugriffspolitik interpretiert und eine lokale Zugriffsentscheidung trifft (“PolicyEvaluator“) und einem Objekt, das die verschiedenen Ergebnisse zu einer Zugriffsentscheidung kombiniert (“DecisionCombinator“). Die RADS Spezifikation enthält auch administrative Objekte, mit denen festgelegt wird, welche Policy, welche “SecuredResources“ behandelt.

Diskussion

Der CORBAMed RADS ist eine Lösung für die Integration von zentral verwalteten Zugriffspolitiken. Er spezifiziert eine konkrete Architektur, Objekte und Schnittstelle zur Abfrage von Zugriffsentscheiden, wobei folgende Aspekte unterstützt werden:

- Zu einem zugreifenden Subjekt (“Principal“) kann eine Liste von Sicherheitsattributen übergeben werden (z. B. auch der Ort, von wo der Zugriff stattfindet), die bei der Zugriffsentscheidung ausgewertet werden.
- Zusätzlich können dynamische Attribute ermittelt werden, die die Berücksichtigung der Beziehung zwischen Patienten und medizinischem Personal zu einem Zeitpunkt in den Zugriffsregeln ermöglichen.
- Verschiedene Zugriffspolitiken können definiert und gezielt eingesetzt werden.
- Für die Referenzierung der „Secured Resources“ ist ein hierarchischer Namensraum vorgesehen, der die Formulierung der Zugriffsregeln für Kategorien von Ressourcen erlaubt. Alle Attribute, die zur Beschreibung einer „Secured Recource“ benötigt werden, können in einer dynamischen Key-Value-Liste (“ResourceNameComponentList“) abgelegt werden.

Die inhaltlichen Aspekte der Zugriffskontrolle bleiben dabei offen, wie z. B. die Struktur der Zugriffspolitik und die Administration der darin enthaltenen Regeln, die Referenzierung bzw. Namensgebung der „Secured Resources“ und der Operationen sowie die Funktionsweise des „Dynamic Attribute Service“. Die Spezifikation beschränkt die Clientfunktionalität außerdem auf die Abfrage einer konkreten Zugriffsentscheidung; die gezielte Ermittlung von Rechte-Mengen (z. B. Fall-Listen) wird durch die Schnittstelle nicht unterstützt.

Ein CORBAMed RAD Service kann auch in eine „Nicht-CORBA-basierte“ Umgebung integriert werden, indem jedes beteiligte Informationsystem ein ADO-Client-Interface einbindet.

3.4.6 Zusammenfassung

Die vorangegegangen Abschnitte stellten verschiedene Modelle von Zugriffspolitiken, verschiedene Zugriffskontrollmechanismen und deren Integration in Gesundheitsinformationssysteme vor. Keiner der Ansätze behandelt oder erfüllt genau die in Kapitel 2 formulierten Anforderungen, aber einige der Ansätze erfüllen einzelne Aspekte der Anforderungen. Die Ansätze zur dynamischen Zugriffskontrolle [43, 44] sind gute Beispiele für die Umsetzung des „Need-To-Know“-Prinzips. Rollenbasierte Zugriffspolitiken unterstützen vor allem das „Prinzip der minimalen Rechte“, die Trennung von verschiedenen Aufgaben und die zentrale und dezentrale Administration der Zugriffspolitik [39, 41, 40, 36, 20]. Der RAD Service ist ein geeignetes Modell für die Integration eines zentral organisierten Zugriffskontroll-Mechanismus in heterogene Umgebungen [24, 7]. Alle Anforderungen können nur durch die Kombination verschiedener Lösungsansätze und durch Entwickung neuer Lösungen erfüllt werden. Insgesamt sieht man, dass die Anforderungen eines großen Klinikums mit stark heterogener IT-Landschaft noch nicht ausreichend umgesetzt sind.

Kapitel 4

Das Zugriffskontroll-System

Aus den Anforderungen in Kapitel 2 wird im folgenden ein Zugriffskontroll-System entwickelt, das die spezifischen Bedürfnisse des Mainzer Klinikums berücksichtigt, dessen Strukturen aber so flexibel sind, dass es auch auf andere Umgebungen übertragen werden kann. Dabei stehen folgende Ziele im Vordergrund:

- **Klinikweite Verfügbarkeit von Zugriffsinformation** basierend auf der aktuellen Zugriffspolitik des Klinikums und der einzelnen Fachkliniken sowie auf den jeweils aktuellen Behandlungszusammenhängen.

- **Einfache und dezentrale Autorisierung:** Für Funktionen, die in den meisten Kliniken vorkommen sollen zentrale funktionspezifische Rollen definiert werden, von denen einrichtungsspezifische Rollen abgeleitet werden. Die Zuweisung der Rollen an Personen oder Systeme ist Aufgabe lokaler Administratoren. Die Definition der Zugriffsrechte sollte nur die minimal notwendige Komplexität beinhalten (Nicht alles was machbar ist, sollte auch getan werden!).

- **Integrierbarkeit der Zugriffsentscheidungen** in bestehende Systeme.

- **Vertretbarer Implementationsaufwand:** Da die Brisanz der Zugriffskontrollproblematik mit wachsender Anzahl von Systemen und Zugriffsmöglichkeiten zunimmt, ist die baldige Verfügbarkeit einer (Teil-)Lösung ein wichtiges Ziel.

- **Integration vorhandener Infrastruktur:** Das Benutzerverzeichnis der EDV-Abteilung sollte für eine Benutzerverwaltung im gesamten Klinikum ausgebaut werden. Der Kommunikationsserver als zentrale Komponente für den Datenaustausch zwischen Systemen sollte als Lieferant von Kontextwissen z. B. der administrativen Ereignisse oder der leistungs-basierten Ereignisse genutzt werden.

4.1 Grundprinzip und Design

Das Grundprinzip des Systems ist das Verfügbarmachen von Berechtigungsinformation als Grundlage konkreter Zugriffsentscheidungen für die Systeme im Klinikum und das dazu notwendige Sammeln und Erfassen von Wissen über Zugriffsrechte und -regelungen. Deshalb setzt sich das System aus je einer Zugriffsschicht für die Wissenserfassung und für die Abfrage von Zugriffsinformation und konkreten Zugriffsentscheidungen zusammen, die auf drei Wissensservern operieren. Das zugrundeliegende Informationsmodell ist rollenbasiert, d. h. Rechte werden abstrakt erfasst und in Rollen strukturiert, die den Subjekten zugewiesen werden können. Statische Anteile der Rechte sind durch ein verbindliches kontrolliertes Vokabular strukturiert und festgelegt. Die dynamischen Anteile werden durch Kontextregeln definiert, die zu jedem Zeitpunkt eine spezifische Menge von Behandlungsphasen und damit von korrespondieren Patiententeilakten festlegen. Bei der Abfrage von Zugriffsentscheidungen oder Rechteinformationen, werden diese Kontextregeln ausgewertet und als Referenzen auf Patientenakten zurückgeliefert.

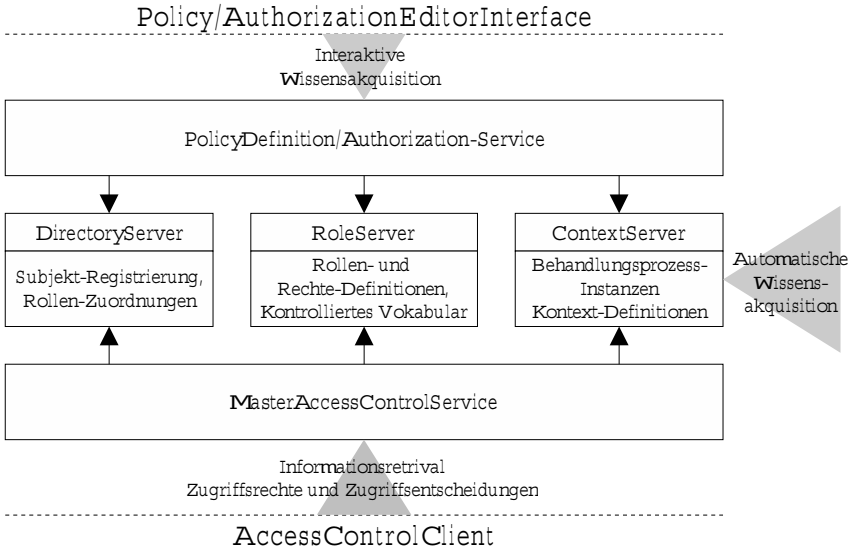

Abbildung 4.1: Die Architektur des Zugriffskontrollsystems

Die drei Wissenserver sind im einzelnen ein Berechtigungs-Server (“Role Server“), ein Kontext-Server (“Context Server“) und der Benutzer-Server der EDV-Abteilung (“Directory Server“). Diese Unterteilung resultiert einerseits aus der Trennung nach Funktionalität und andererseits – wie im Fall des Benutzerverzeichnisses – aus existierender Infrastruktur. Abb. 4.1 zeigt die grobe Architektur des Gesamtsystems einschließlich der Schnittstellen zwischen verschiedenen Komponenten.

4.1.1 Das Benutzerverzeichnis

Ursprüngliche Struktur

Die EDV-Abteilung des Klinikums betreibt bereits längere Zeit ein X.500-Verzeichnis, das Information über E-Mail- und Fax-Server-Nutzer enthält. Zugriffe auf das Verzeichnis erfolgen über das Lightweigth Directory Access Protocol (LDAP). Die Nutzerdaten sind in einer Klasse „UKMZPerson“ abgebildet, die eine Erweiterung der LDAP-Klasse „Person“ darstellt. Die X.500-Hierarchie „Land“ (c), „Organisation“ (o), „Organisationseinheit“ (ou) lautet: c=de, o=Klinikum der Johannes Gutenberg-Universität, ou=Personen. Unter diesem Knoten sind alle Nutzer mit Name, Vorname, Fachklinik, Telefonnummer, Faxnummer und E-Mail-Adresse registriert. Aus Gründen des Pflegeaufwands wurde die interne Einrichtungsstruktur bislang nicht abgebildet.

Strukturelle Erweiterungen

Für die Anforderungen der Autorisierung muss das Benutzerverzeichnis folgendermaßen strukturell erweitert werden:

- Die Klasse „UKMZPerson“ wird durch eine Klasse „UKMZSubject“ ersetzt. Diese erhält zusätzliche Attribute „subjecttype“ – zur Unterscheidung von Personen und Systemen – und „qualification“ – zur Differenzierung der Personalstruktur bzw. verschiedener Systemtypen.

- Die Zugehörigkeit zu verschiedenen Organisationseinheiten wird abgebildet. Das ist einerseits durch eine feinere Aufgliederung der Verzeichnisstruktur in Organisationseinheiten des Klinikums möglich, unter deren Knoten dann die Subjekte erfasst werden. Andererseits kann die Zugehörigkeit durch ein „unit“-Attribut der „UKMZSubjekt“-Klasse abgebildet werden, das die entsprechenden Organisationsbezeichner enthält.

- Die Zuordnung der Subjektrollen erfolgt alternativ durch ein „role“-Attribut der „UKMZSubject“-Klasse oder durch die Klasse „OrganizationalRole“, wobei das „role“-Attribut die eindeutigen IDs der zugeordneten Rollen aus dem Rollen-Server enthalten würde oder das „roleOccupant“-Attribut den „distiguished name“ der verzeichneten Subjekte. Zu jeder Rollenzuordnung muss außerdem ein Gültigkeitszeitraum abgelegt werden können. Bei der Autorisierung müssen die Komponenten der Zugriffsschicht sowohl auf den Berechtigungs-Server als auch auf das Benutzerverzeichnis zugreifen, um die referentielle Integrität der Zuordnungen zu gewährleisten.

4.1.2 Der Berechtigungs-Server

Kern des Berechtigungs-Servers ist eine relationale Datenbank, die Rollen- und Rechtedefinitionen enthält. Sie enthält keine Subjekt-Rollen-Beziehungen (→

Benutzer-Verzeichnis), aber in der derzeit geplanten Architektur wird sie zusätzlich die temporären Subjekt-Rechte-Beziehungen enthalten, die aus der expliziten Freigabe von Zugriffen an Personen resultieren. Der Berechtigungs-Server integriert außerdem das kontrollierte Vokabular, das für die Rechtedefinition benötigt wird.

4.1.3 Der Kontext-Server

Der Kontext-Server besteht ebenfalls aus einer relationalen Datenbank, in der Instanzen aller Phasen und Aktivitäten aus den Behandlungsprozessen gespeichert werden. Das sind einerseits Patientenaufenthalte und -besuche in verschiedenen Organisationseinheiten und andererseits Anforderungen und Leistungen von Organisationseinheiten sowie die explizite Behandlungsbeteiligung von Personen. Die Wissenserfassung erfolgt automatisch über ereignisgesteuerte Schnittstellen; sie kann durch ein Benutzerinterface ergänzt werden. Der Kontextserver enthält außerdem die Definitionen der Kontextregeln, die in den Zugriffsrechten referenziert und zum Zugriffszeitpunkt ausgewertet werden.

4.1.4 Die Zugriffsschichten

Die Zugriffsschicht für die Wissensakquisition („PolicyAdmin-Service“) beinhaltet sowohl das Benutzer-(Subjekt-)Management, die Definition der Zugriffsrechte, ihre Strukturierung in Rollen, und die konkrete Rollen- und Rechte-Zuweisung an Subjekte als auch die Erfassung der regelbasierten Teile der Zugriffspolitik, nämlich der Regeln zur Ableitung von Zugriffsberechtigungen aus Behandlungszusammenhängen. Zusätzlich besitzt das System Schnittstellen für die automatische Akquirierung von Kontextwissen und strukturellen Informationen aus anderen Systemen des Klinikums. Die Zugriffsschicht für das Informationsretrieval („MasterAccessControl-Service“) enthält Funktionen, die eine gezielte Abfrage von Zugriffsentscheidungen oder Rechtemengen ermöglichen.

Für die Modellierung der Zugriffsschichten wurde ein objektorientiertes Design gewählt. Jedes Objekt verwaltet einen spezifischen Aspekt der Information und bietet in der Regel Funktionen für beide Zugriffsschichten, d. h. für die Wissensakquisition und das Wissensretrieval. Einige der Objekte werden nur innerhalb der Zugriffsschichten angesprochen, andere bieten Schnittstellen zur Integration in Systeme oder für grafischen Benutzerwerkzeuge. Die Struktur der Objekte und Daten wird im Informationsmodell (4.2) beschrieben, die Funktionalität der Objekte und ihr Zusammenwirken ist Inhalt des Applikationsmodells (4.3).

4.2 Das Informationsmodell

Im Informationsmodell sind die Datenstrukturen und Objekte definiert, die in den Zugriffsschichten für die Aufgaben der Zugriffskontrolle verwendet werden. Es legt nicht die Art der Datenspeicher oder die Form der gespeicherten Daten (z. B. Datenbanktabellen) fest. Das Informationsmodell ist entsprechend den Zugriffsschichten in zwei Bereiche untergliedert. Das Wissensmodell beinhaltet die Repräsentation der Rollen und Rollenhierachien, der Subjekte und Rechte, der Organisationsstrukur sowie des Behandungsprozesses. Das Retrievalmodell enthält geeignete Anfrage- und Ergebnis-Strukturen für Zugriffsentscheidungen.

Die spezifischen Designentscheidungen für die Realisierung werden in den folgenden Abschnitten mit Hilfe von Objektdiagrammen und Typdefinitionen erläutert.

4.2.1 Strukturen für die Wissensakquisition

Abb. 4.2 gibt die wesentlichen Entitäten für die Wissenserfassung und ihre Relationen wieder. Aus den Entitäten werden im weiteren die Datenstrukturen

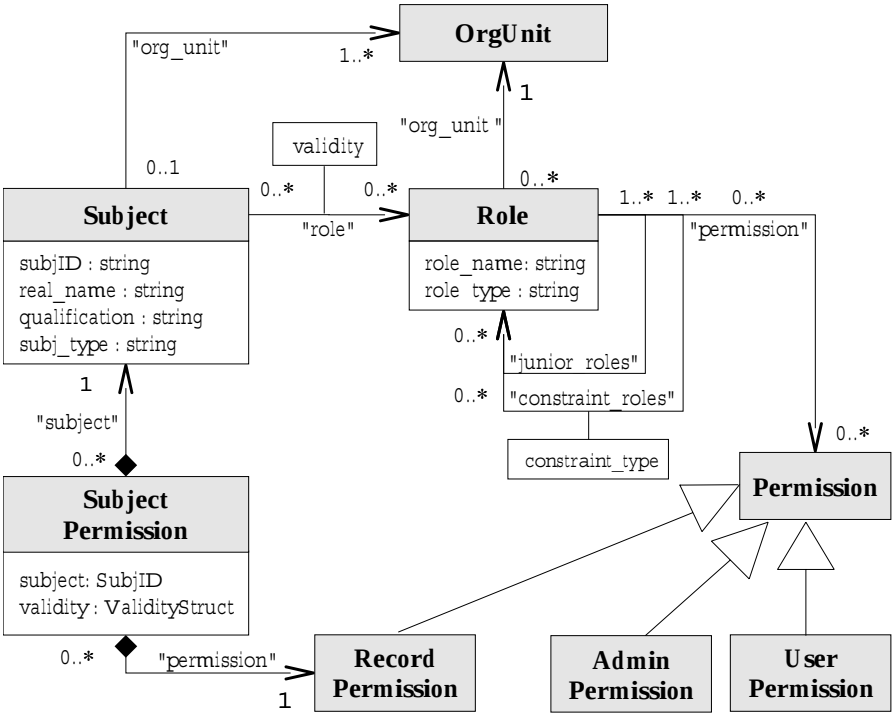

Abbildung 4.2: Strukturen für die Wissensakquisition

abgeleitet. Die Umsetzung in Objekte oder Strukturen sind durch den Ver-

arbeitungszusammenhang bedingt (→ Abschnitt 4.4); im allgemeinen werden Entitäten mit eigener Funktionalität oder abgeleiteten Entitäten als Klassen realisiert. Für das Informationsmodell wird zunächst nicht zwischen Strukturen und Objekten unterschieden.

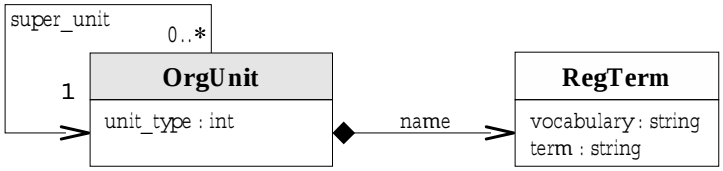

Abbildung 4.3: Die „OrgUnit“-Struktur

Organisationseinheiten

Die OrgUnit-Entität repräsentiert eine Organisationseinheit des Klinikums. Die abgeleitete OrgUnit-Struktur enthält einen eindeutigen Namen, den Typ der Organisationseinheit und eine Referenz auf die übergeordnete Organisationseinheit. Durch die Referenzierung der Instanzen untereinander wird die Einrichtungshierarchie des Klinikums abgebildet. Die Namen der Organisationseinheiten werden mit einem hierarchischen Bezeichner im kontrollierten Vokabular registriert.

Subjekte

Die „Subject“-Struktur enthält die Informationen zu einer Person oder einem System, die im Zugriffskontroll-System benötigt werden. Sie enthält nicht alle Attribute der UKMZSubject-Klasse des zentralen Benutzerverzeichnis, sondern nur den Ausschnitt, der bei der Autorisierung referenziert wird. Systeme, die Benutzerdaten zu anderen Zwecken abfragen oder manipulieren, verwenden andere Ausschnitte der Informationen. Bei der Autorisierung von Subjekten muss ein Administrator aufgrund der Identität der Qualifikation und der Einrichtungszugehörigkeit des Subjekts über Rollenzuordnungen entscheiden. Deshalb besitzt die Subject-Struktur eine eindeutige ID, einen aussagekräftigen Namen, den Subjekt-Typ (Person oder System) einen beruflichen oder funktionalen Qualifikationsbezeichner sowie Referenzen auf alle Organisationseinheiten, denen das Subjekt angehört. Die Subject-Entität referenziert außerdem eine Menge von Rollen, wobei die Referenz durch einen Gültigkeitszeitraum erweitert ist.

Rollen

Role-Strukturen bestehen aus dem eindeutigen Rollen-Namen, dem Rollen-Typ (Benutzer oder Adminstrator), einer Referenz auf die Organisationseinheit, die den Wirkungsbereich der Rolle festlegt, einer Liste von Referenzen auf die

Rollen-Rechte, einer Liste von Referenzen auf übergeordnete Rollen, von denen die Rolle abgeleitet wird, und einer Liste von Rollen, zu denen eine Constraint-Relation besteht. Die Constraint-Relation ist durch den Constraint-Typ (starker und schwacher wechselseitiger Ausschluss, vorausgesetzte Rolle → 3.3) erweitert. Die Subjekt-Rollen-Zuordnung ist als Referenz der Rolle in der Subject-Struktur realisiert und wird im Subjektverzeichnis gespeichert.

Rollenrechte

Rechte zum Zugriff auf Patientendaten (Benutzerrechte) und Administrationsrechte beziehen sich auf unterschiedliche Zugriffsobjekte und unterscheiden sich deshalb in ihrer Struktur. Benutzerrechte adressieren die Operationen auf Patientendaten, Administratorrechte die Operationen auf Subjekte, Rollen und Rechte des Zugriffskontroll-Systems. Deshalb werden aus einer abstrakten Permission-Entität eine „UserPermission"- und eine „AdminPermission"-Struktur abgeleitet. UserPermission-Objekte beinhalten die Berechtigung zu

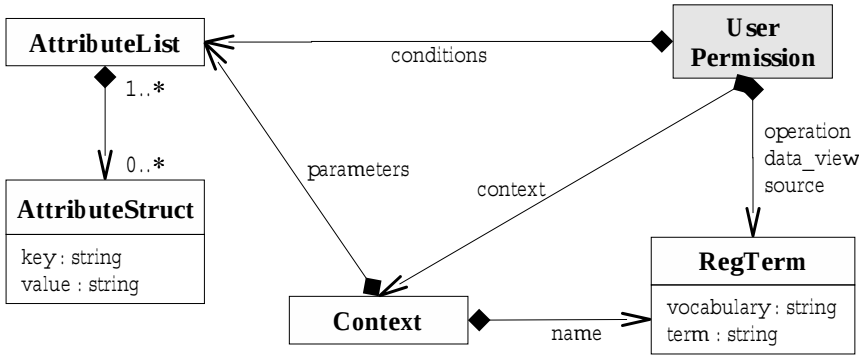

Abbildung 4.4: Struktur der Benutzerrechte

Klassen von Zugriffen auf Patientendatenobjekte. Die Zugriffsklasse wird definiert durch den Operationsprototyp (operation), den Bezug zur Patientenakte (trContext), den Datenausschnitt der Akte (dataView), einem Ausdruck für das Ursprungssystem (source), das die Zugriffe anbietet, sowie eine Liste von möglichen Bedingungen (conditions), die an das Recht gekoppelt sind (→ Abb. 4.4). Die UserPermission-Klasse referenziert nur die Namen der Operationen, Datensichten und Ursprungssysteme, die als hierarchische Bezeichner im kontrollierten Vokabular registriert sind. Die Ausdrücke (RegTerm) bestehen jeweils aus dem Namen des Vokabulars und aus einem eindeutigen Namen, der den vollständigen Pfad des Begriffs-Knotens im Vokabular enthält. Kontexte werden mit ihrem Namen und einer Parameterliste referenziert. Sie sind als Prototypen im Vokabular registriert. Das bedeutet, dass außer dem hierarchischen Namen auch eine Liste von Parametertypen abgelegt ist. Bei der Rechtedefinition müssen

für diese Parameter Werte vom zugehörigen Typ spezifiziert werden. Zu jedem Datensicht-Bezeichner und jedem Kontextprototypen im Vokabular existiert unter gleichem Namen eine DataView-Definitionsstruktur im Rollen-Server bzw. eine Kontextregel, deren Definition im Kontextserver abgelegt ist. Die Repräsentation von Datensichten und Datenklassen ist in Abschnitt 4.2.4, die Struktur der Kontextregeln in den Abschnitten 4.2.3 und 4.4.4 detailliert beschrieben. Das Vokabular wird in den Abschnitten 4.2.6 und 4.4.5 behandelt. Die Bedingungsliste enthält einfache Attributstrukturen, die aus einem Attributbezeichner (key) und einem Attributwert (value) bestehen, z. B. für die Protokollierung eines Zugriffs „*key=LOG_LEVEL, value=ALL*“.

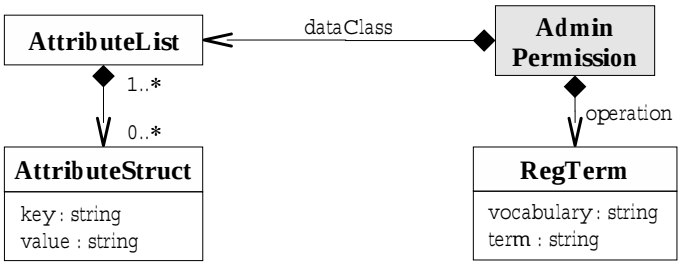

Abbildung 4.5: Struktur der Administrationsrechte

Administrationsrechte legen den Kompetenzbereich eines Administrators fest. Die Attribute der AdminPermission-Objekte sind der Bezeichner einer Operation, die der Administrator ausführen darf, und eine Liste von Attribut/ Wert-Paaren, die die Menge der betroffenen Strukturen einschränken (Abb. 4.5). Die Parameter zur Klassifikation von Rollen und Subjekten sind in Tabelle 4.1 aufgeführt. Mit der Klassifikation der Rollen nach der zugehörigen Organisationseinheit wird die dezentrale Administration in den verschiedenen Einrichtungen ermöglicht. Die Klassifikation entsprechend der Rollenhierarchie, die alle Abkömmlinge einer Rolle zusammenfasst, erlaubt die Trennung der Rollen verschiedener Berufsgruppen. Analog können auch die Subjekte nach ihrer Organisationszugehörigkeit und ihrer beruflichen bzw. funktionalen Einordnung klassifiziert werden, außerdem nach dem Subjekttyp. Die Klassifizierung der Rechte für eine dezentrale Administration ist ungleich komplizierter, da die betroffenen Patientenakten über Regeln festgelegt werden. Deshalb wird die dezentrale Rechtedefinition nur für lokale Informationssysteme und für die Definition von Administratorrechten im Zuständigkeitsbereich einer Organisationseinheit (`SYSTEM_ID, ADMIN_UNIT`) unterstützt. Die Delegation von Rechten zur Zuweisung an Rollen wird durch geignete Rollenkonstrukte unterstützt (→ Abschnitt 5.2.2).

Die Operationsbezeichner sind durch die Funktionalität der Rollendefinitions- und Autorisierungs-Schnittstellen bestimmt (→ Abschnitt 4.4).

Attribut	Wert	Wertemenge
ROLE_ID	Rollenidentifikator	eindeutiger Identifikator
ROLE_UNIT	Gültigkeitsbereich der Rolle	registrierter Bezeichner
JUNIORROLE_ID	„Junior-Role“ der Teilhierarchie	eindeutiger Identifikator
SUBJECT_ID	Subjektidentifikator	eindeutiger Identifikator
SUBJECT_UNIT	Subjektzugehörigkeit	registrierter Bezeichner
SUBJECT_QUAL	Qualifikation des Subjekts	registrierter Bezeichner
SUBJECT_TYPE	Art des Subjekts	{PERSON, SYSTEM}
SYSTEM_ID	Name der Datenquelle	registrierter Bezeichner
ADMIN_UNIT	verantwortl. Organisationseinheit	registrierter Bezeichner

Tabelle 4.1: Parameter zur Klassifikation von Rollen, Rechten und Subjekten

Subjektrechte

Die **„SubjectPermission“**-Struktur repräsentiert eine direkte Subjekt-Recht-Beziehung. Sie referenziert deshalb ein Subjekt bezüglich seiner Subjekt-ID und ein statisches Recht (**„RecordPermission“**), d. h. ein Recht zum Zugriff auf eine explizite (Teil-)Krankenakte, die durch die **„PatientRecord“**-Struktur definiert ist. Diese ist aus der eindeutgen Patienten-ID und einer Attributliste (**„treatment_parameters“**) zusammengesetzt, die den Teil der Krankenakte bezüglich der Behandlungsphase festlegt. Tabelle 4.2 enthält die Attribut/Wert-

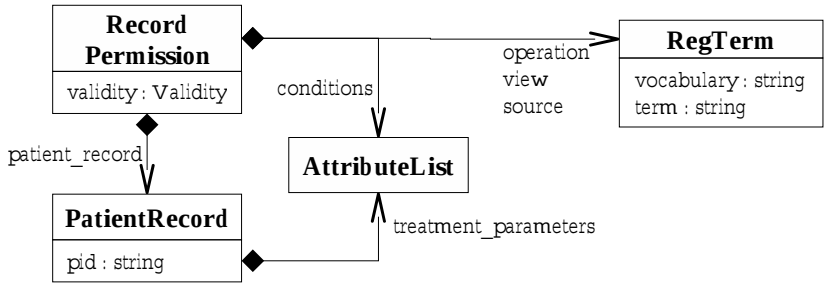

Abbildung 4.6: Struktur der Statischen Rechte

Paare dazu. Subjektrechte sind für zeitlich begrenzte Berechtigungen – z. B. bei konsiliarischer Beteiligung – konzipiert. Deshalb besitzen SubjectPermission-

Attribut	Wert	Wertemenge
ADMISSION_ID	ID des Verwaltungsfalls	eindeutiger Identifikator
START	Phasenbeginn	absoluter Zeitpunkt CCYYMMDDhhmmss
END	Phasenende (optional, nur abgeschlossene Phasen)	absoluter Zeitpunkt CCYYMMDDhhmmss
LOCATION	Patientenaufenthaltsort (optional, falls eindeutig)	registrierter Bezeichner (Organisationseinheit)
TREATMENT_ID	ID der Behandlungsphase (optional, falls vorhanden)	Identifikator, eindeutig innerhalb des Verwaltungsfalls

Tabelle 4.2: Parameter zur Behandlungsphase

Objekte eine zusätzliche Struktur (validity), die den Gültigkeitszeitraum des Rechts festlegt.

4.2.2 Strukturen für das Informationsretrieval

Die Schnittstelle des MasterAccessControl-Services benötigt Strukturen, die einerseits den Zugriff oder die Klasse von Zugriffen definieren, deren Berechtigung das Informationssystem bezüglich eines Subjekts überprüfen möchte, und andererseits Strukturen, die – bei unvollständig spezifizierten Anfragen – die Ergebnismenge in geeigneter Form zurückliefern (siehe Abb. 4.7).

Abfragestruktur

Die Abfragestruktur (**"Request"**) ist ähnlich aufgebaut wie die RecordPermission-Struktur aus Abschnitt 4.2.1. Statt der Referenz auf eine konkrete Patiententeilakte enthält sie eine Liste von Parametern (**search_ parameters**), die die Rechtemenge bezüglich der Patiententeilakten einschränken. Die Menge der Parameterbezeichner kann jederzeit erweitert und an neue Anforderungen angepaßt werden. In Tabelle 4.3 sind die Parameter erläutert, die aus der momentanen Sicht für eine Spezifikation des Suchraums sinnvoll erscheinen. Sie bestimmen eine Menge von Patiententeilakten durch Charakterisierung der Behandlungsphasen, die in den jeweiligen Teilakten dokumentiert sind. Dabei beziehen sich alle Parameter außer CURRENT und SCOPE_UNIT auf die Behandlungsphasen; CURRENT bezieht sich auf den gesamten Behandlungsfall und SCOPE_UNIT soll nach spezifischen UNIT -Parametern in den Kontextattributen der Rechte filtern, sofern solche existieren. Die Einschränkung von Abfragen auf Rechte mit spezifischen Kontexteigenschaften, verhindert die unnötige Auswertung von Kontextregeln und verbessert somit die Performanzeigenschaften des

Attribut	Wert	Wertemenge
UNIT	hauptverantwortliche Organisationseinheit der Behandlungsphase	registrierter Bezeichner
START	Beginn des Anfragezeitraums	absolutes Datum CCYYMMDD
END	Ende des Anfragezeitraums	absolutes Datum CCYYMMDD
CURRENT	Behandlungsphasen aktueller oder bereits abgeschlossener Fälle	{0,1}
ADMISSION_ID	ID des Verwaltungsfalls	eindeutiger Identifikator (klinikumsweit)
PATIENT_ID	Verwaltungs-Nummer des Patienten	eindeutiger Identifikator (klinikumsweit)
TREATMENT _TYPE	Typ der Behandlungsphase	{S, A, VST, NST, ...}
SCOPE_UNIT	Kontextparameter: „Organisationseiheit“	registrierter Bezeichner (Organisationseinheit)

Tabelle 4.3: Parameter zur Einschränkung des Anfrage-Ergebnisses bezüglich der Behandlungsphasen

MasterAccessControl-Service.

Zusätzlich kann die Menge der Patientenakten durch folgende Patienten-Identifikationskriterien eingschränkt werden: Den Familiennamen (`NAME`), den bzw. die Vornamen (`GIVENNAME`), den Geburtsnamen (`BIRTHNAME`), das Geburtsdatum (`BIRTHDAY`) und den aktuellen Wohnsitz des Patienten (`CITY, ZIP, COUNTRY`). Ein Interpretationskennzeichen `SEARCH_POLICY` $\in$ {`'EXACT', 'INITIALS', 'PHONETIC', 'BIRTHNAME'`} gibt an, ob die angegebenen Zeichfolgen als vollständige Namen oder Namensanfänge interpretiert werden sollen, ob ein phonetischer Algorithmus auf die Namen angewendet werden soll, und ob auch der Geburtsname gegen den Namen abgeglichen werden soll.

Die Request-Struktur wirkt wie ein positiver Filter; alle Rechte, die die angegebenen Attributewerte besitzen oder sich zum Auswertungszeitpunkt auf dieselben Behandlungen beziehen, werden zurückgeliefert. Sie liefert die Möglichkeit, Berechtigungsmengen auf die Struktur und den Inhalt eines spezifischen Informationssystems zuzuschneiden, und so die Anzahl der Einzelabfragen zu reduzieren und die Größe der zurückgelieferten Rechtemenge zu minimieren.

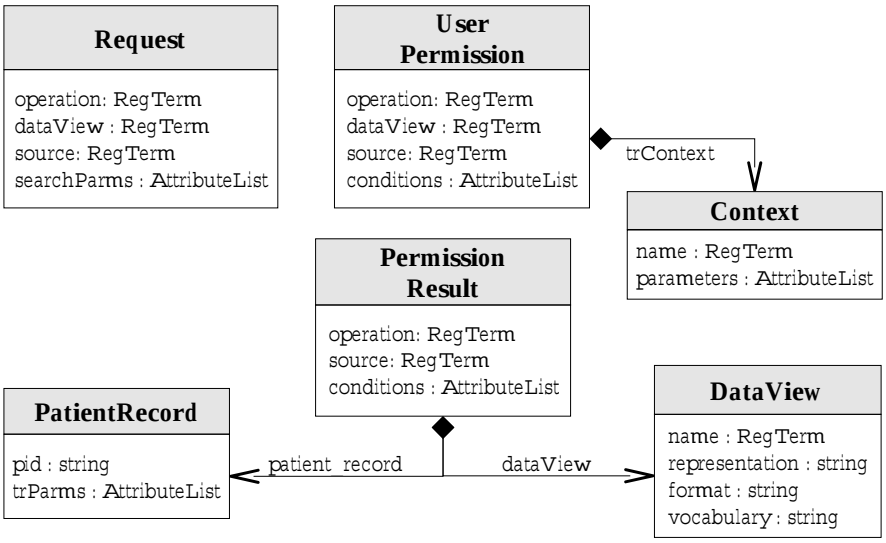

Abbildung 4.7: Strukturen für das Informationsretrieval

Ergebnisstruktur

Ergebnisse der Abfrage sind Mengen von Zugriffsrechten bezogen auf existierende Patientenakten. Die **„PermissionResult“**-Struktur entspricht der Record-Permission-Struktur mit einer geringfügigen Abweichung. Statt des Bezeichners für die Datensicht enthält die PermissionResult-Struktur eine vollständige **„DataView“**-Struktur bestehend aus dem registrierten Namen und der Definition der Datensicht, die wiederum aus einer text-basierten Repräsentation, dem Namen des Vokabulars, aus dem die Datenklassen-Bezeichner stammen, und einem Formatbezeichner zur Repräsentation zusammengesetzt ist. Alle weiteren Attribute der PermissionResult-Struktur sind auch inhaltlich analog zu den bereits erläuterten Attributen der RecordPermission-Struktur.

4.2.3 Behandlungsprozesse und -kontexte

Grundlage für die Kontextdefinitionen ist ein Informationsmodell, das die relevanten Strukturen des Behandlungsprozesses abbildet. Relevant für die Zugriffskontrolle sind zeitliche Phasen, die eine Organisationseinheit oder ein Subjekt in Beziehung zur Behandlung eines Patienten setzen, und so die Rückschlüsse auf die Existenz eines Behandlungsauftrags zwischen Subjekt bzw. Organisationseinheit und Patient zulassen. Jede derartige Phase wird im Behandlungsprozessmodell durch ein Behandlungselement (**“TreatmentElement“**) repräsentiert (Abb. 4.8). Die Ausprägung eines Behandlungselements ist durch den Elementtyp (z. B. „Aufenthaltphase“, „Leistungsanforderung“, „externe Diagnose-/Behandlungsleistung“) bestimmt. Das Attribut „**treatment_type**“

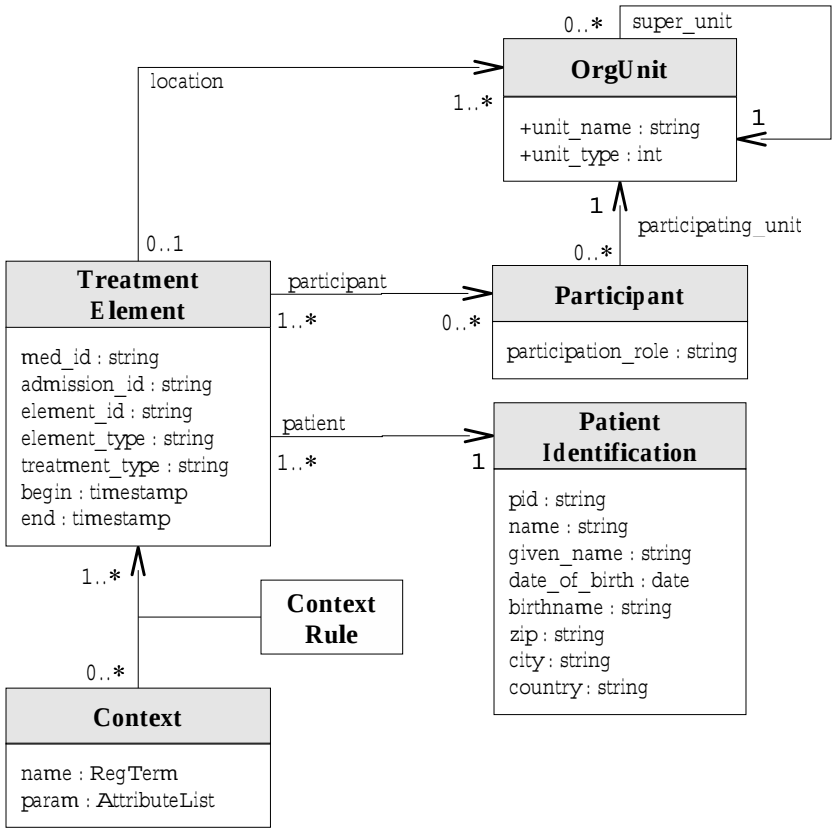

Abbildung 4.8: Strukturen für Kontextdefinition und -retrieval

ermöglicht eine weitere spezifische Verfeinerung der Elemente (z. B. „stationäre Aufenthaltsphase“, „Diagnostik“, „Folge therapeutischer Einzelleistungen“). Das Modell berücksichtigt Identifikations-Attribute (**„PatientIdentification“**-Objekt), um die Auffindung von Behandlungsinformationen unabhängig von der klinikweiten Patientennummer zu unterstützen. Zu jedem Identifikationsobjekt kann es beliebig viele Behandlungselemente geben. Behandlungselemente gehören zu einem medizinischen und einem Verwaltungsfall (**med_id** und **admission_id**). Je nach Elementtyp kann dem Element eine Organisationseinheit als Patientenaufenthaltsort (**location**) und/oder mehrere Organisationseinheiten im Sinne der Behandlungsbeteiligung (**participant**) zugeordnet sein. Die Art der Behandlungsbeteiligung ist durch die „**participation_role**“ spezifiziert.

Die Modellierung der Behandlungselemente mit Hilfe von Typen anstelle einer entsprechenden Klassenhierarchie von Behandlungselementen, hat den Zweck, spätere Erweiterungen des Behandlungsprozessmodells und der Kontextdefinitionen zu erleichtern, und insbesondere die Notwendigkeit von Programm-

erweiterungen zu minimieren.

Behandlungskontexte repräsentieren Mengen von Element-Instanzen, z. B. „*Alle Patientenaufenthalte, die die Station A1 der HNO als Aufenthaltsort referenzieren*“. Durch die Kontextregel und einen definierten Satz von Parameterkonstanten (**parameters**) ist zu jedem Zeitpunkt festgelegt, welche Element-Instanz zur jeweiligen Menge gehört. Die Parameter beziehen sich auf Attribute und Referenzen der TreatmentElement-Objekte.

Struktur der Kontextregeln

Kontextregeln sind ihrem Wesen nach spezifische Mengendefinitionen. Insbesondere hat jede Kontextregel die Form: „Behandlungsphase B gehört zur Kontextmenge $K_{b_1 \circ .. \circ b_n}$, wenn B die Bedingungen $b_1 \circ .. \circ b_n$ erfüllt“, wobei die Bedingungen logisch verknüpft sind („$\circ$“). Analog läßt sich auch die Kontextmenge definieren. Eine Möglichkeit Mengen und Bedingungen zu definieren, ist die Beschreibung der Element-Eigenschaften zusammen mit der Verwendung von Quantoren, sowie die Verknüpfung verschiedener Mengen bzw. Bedingungen durch Mengen- bzw. logische Operatoren. Die Elemente der Kontextmengen sind Behandlungselemente; ihre Eigenschaften (E_i) sind durch die Attributwerte bestimmt und können in folgender Form beschrieben werden:

$$E_i(B, p_1, .., p_n) = B.attribut_i \begin{cases} = \\ \neq \\ < \\ \leq \\ > \\ \geq \\ \in \\ \notin \end{cases} F(p_1, .., p_n) \in W_i,$$

wobei W_i die Wertemenge von B.$attribut_i$ ist und $p_1, .., p_n$ Parameter.

$F(p_1, .., p_n)$ ist im einfachsten Fall eine Konstante oder eine Menge von Konstanten, z. B. Bezeichner für Organisationseinheiten oder Elementtypen. Enthält das Attribut eine Zeitangabe oder einen Zahlenwert, so kann $F(p_1, .., p_n)$ ein mathematischer Ausdruck sein.

Verschiedene Eigenschaften eines Behandlungselements können mit „**und**“ bzw. „**oder**“ verknüpft werden. Die Wirkung der Eigenschafts-Definition auf die Mengendefinition wird durch Quantoren ($\forall$-Operator, $\exists$-Operator und alle anderen Arten von Existenzoperatoren) bestimmt. Quantoren können außerdem verschachtelt auftreten. Aus den so gebildeten Mengen entstehen durch die Bildung von Schnittmengen (für die Regel-Bedingungen: logisches „UND“), Vereinigungsmengen (logisches „ODER“) und Komplementmengen (logische Negation) weitere Mengen.

Die Quantoren und Operatoren der Mengen- bzw. Regeldefinitionen werden in konkreten Kontext-Retrieval-Funktionen abgebildet. Da die bisher formulierten Kontextregeln Regelmäßigkeiten erkennen lassen, kann man die Vielfalt der möglichen Regelausprägungen auf zwei Typen reduzieren. Die Bezeichnung der

Regeltypen wurde in Anlehnung an die implizit verwendeten Quantoren gewählt. Eine Regel vom Typ „`FORALL`“ definiert eine Menge von Behandlungselementen, die spezifische Bedingungen bezüglich ihrer Attributeigenschaften erfüllen. Regeln vom Typ „`FORALL_EXIST`“ definieren je eine Menge von Behandlungselementen bedingt durch die Existenz spezifischer Behandlungselemente (Indikatorelemente), die spezifische Eigenschaften erfüllen, aber nicht zwingend in der definierten Menge enthalten sein müssen. Die Regel-Definition besteht dann aus einer Liste von Inputparameter und je nach Regeltyp aus maximal zwei Listen von Bedingungen, nämlich Bedingungen, die die Indikatorelemente festlegen, und Bedingungen, die die Attributeigenschaften der Mengenelemente sowie gegebenenfalls die Abhängigkeiten zwischen den Mengenelementen und den Indikatorelementen definieren. Bedingungen werden durch die Condition-Struktur

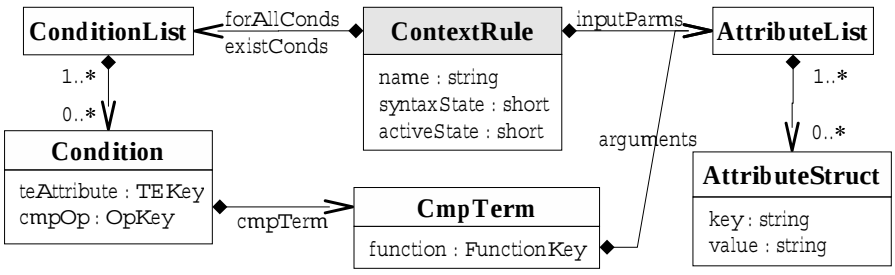

Abbildung 4.9: Struktur der Kontextregeln

ausgedrückt (→ Abb. 4.9). Die Condition-Struktur enthält einen logischen Ausdruck bestehend aus einem Attribut-Bezeichner (`teAttribute`), der ein Attribut der TreatmentElement-Klasse referenziert, einem Vergleichsoperator (`cmpOp`) und einem Vergleichsterm (`cmpTerm`). Durch Aufzählungstypen `TEKey` und `OpKey` und sind Attribut-Bezeichner und Vergleichsoperatoren festgelegt. Der Vergleichsterm besteht aus einer Funktion, die durch den Aufzählungstyp `FunctionKey` festgelegt ist, und einer Liste der zugörigen Funktionsargumente. Funktionsargumente sind Tupel bestehend aus einem Schlüssel, der den Argumenttyp angibt, und dem zugehörigen Wert. Sie können Input-Parameter, Konstanten, Schlüsselwörter oder Attribut-Bezeichner enthalten. Mit der CmpTerm-Struktur wird die Menge der möglichen Vergleichsterme auf einen kleinen Ausschnitt reduziert, mit dem einfache Werte, Wertemengen oder einfache vordefinierte mathematische Funktionen von Werten erfasst werden können.

Für jeden Inputparameter, der in einer Bedingungsdefinition verwendet wird, erhält man einen zugehörigen Parameter des Kontextprototypen, der bei der Rechtedefinition als Kontextparameter angegeben werden muss.

Beispiel: *„Fallkontext“*

Als Fallkontext soll die Menge aller Behandlungselemente B definiert sein, für die es im gleichen medizinischen Fall ein Behandlungselement B' gibt, das einen Aufenthalt des Patienten in der angegebenen Organisationseinheit (OE) impliziert und das nicht vor dem Beginn des jeweiligen Behandlungselements B geendet hat.

B $\in$ **Fallkontext**(OE), wenn gilt:

$\exists$ B' mit B'.*location* = OE und B.*med_id* = B'.*med_id* und B.*begin* $\leq$ B'.*end*

bzw.

Fallkontext(OE) = { $B_i \in B \mid \exists$ B' mit B'.*location* = OE und B_i.*med_id* = B'.*med_id* und B_i.*begin* $\leq$ B'.*end*}

4.2.4 Datensichten und Datenklassen

Datensichten haben die Funktion möglichst genau festzulegen, welche Daten von einem Zugriff betroffen sein dürfen und welche nicht. Sie sollen gewissermaßen wie ein Filter zwischen das anfragende Subjekt und die allgemeinst mögliche Datenmenge einer Zugriffsoperation gesetzt werden. Die Granularität, in der die Datensicht definiert sein kann, reicht von allgemeinen Datenkategorien bis hin zu einzelnen Daten-Items. Um derartige Definitionen für eine große Allgemeinheit von Informationssystemen verwertbar zu machen, benötigt man eine flexible und – wenn möglich – standardisierte Repräsentationsform.

Definition

Im folgenden werden zwei Ansätze zur Repräsentation von Datensichten vorgeschlagen, die sich beide auf ein kontrolliertes und strukturiertes Vokabular von Datenbezeichnern stützen. Sie unterscheiden sich durch die Beschreibungsprache die im einen Fall auf der Basis von XML-Elementen [10] definiert ist und im anderen Fall auf der Basis regulärer Ausdrücke [26]. Das Vokabular legt die hierarchische Strukturierung der Bezeichner in Daten-Kategorien bzw. -Klassen fest. Grundsätzlich muss zwischen positiven und negativen Datensichten unterschieden werden. Die positive Sicht enthält alle Items, die vom Zugriff betroffen sein dürfen, die negative ist im eigentlichen Sinn ein Filter und enthält alle Items, die vom Zugriff ausgeschlossen sind. Durch Kombination von positiven und negativen Teilsichten können Definitionen vereinfacht werden, z. B. durch Ausschluss eines einzelnen Items aus einer umfangreichen Datenkategorie. Bei der Berücksichtigung negativer und positiver Sichten werden Semantikprüfungung

und Auswertung einer Datensicht komplizierter. Deshalb müssen Definitions- und Auswertungsaufwand gegeneinander abgewogen werden. In manchen Fällen kann die Verwendung negativer Definitionen durch geeignete Anordnung des Datenvokabulars vermieden werden.

XML-Repräsentation Die XML-Repräsentation enthält für jede Hierachieebene ein korrespondierndes XML-Tag. Durch Attribute in den Tags wird das „Vorzeichen“ des Datensicht-Abschnitts bestimmt. Die XML-Repräsentation kann durch entsprechende Sprachkonstrukte auch Einschränkungen bezüglich der Inhalte, des Autors, des Modifikationszeitpunkts und weiterer Eigenschaften der Daten unterstützen. Im anschließenden Beispiel wird die XML-Beschreibungssprache näher erläutert.

Reguläre Ausdrücke Die Formulierung der Datensichten in Anlehnung an reguläre Ausdrucke führt zu wesentlich einfacheren Definitionen, die auch einfacher auszuwerten sind. Sie ist aber auch weniger mächtig als die XML-basierte Form. Die Beschreibungssprache bedient sich der Schreibweise hierarchischer Deskriptoren, wobei Hierarchieebenen durch Punkte voneinander getrennt werden. Ein senkrechter Strich im Ausdruck hat die Funktion des „logischen Oder“. Durch geeignete Klammerung können beliebige Teilbäume des Datenvokabulars referenziert werden. Ausschlüssen von Teilbäume wird ein Minuszeichen vorangestellt. Dieses Format erlaubt keine Klassifizierung nach Inhalten oder Eigenschaften von Datenobjekten. Beispiele für die Formulierung von Datensichten in dieser Art sind in 5.2.1 angegeben.

Beispiel: XML-Repräsentation

Das folgende Beispiel führt die Definition einer Datensicht auf der Basis eines HL7[1]-basierten Vokabulars und des daraus abgeleiteten XML-Schemas vor. Das Vokabular enthält je eine Hierarchiebene für HL7-Segmente, HL7-Felder und HL7-Komponenten. HL7-Segmente werden im Fall von Uneindeutigkeiten durch die Angabe des Kapitels eindeutig bestimmt (siehe Abb. 4.10). Jeder so gebildete Daten-Pfad definiert eine Klasse von Daten mit genau festgelegter Bedeutung.

HL7Vocabulary $\longrightarrow$ HL7Segment(Chapter) $\longrightarrow$ HL7Field $\longrightarrow$ HL7Component

Das abgeleitete XML-Schema enthält einen Elementtyp für jede Hierarchieebene (HL7View, HL7Segment, HL7Field, HL7Component), dessen Inhalt durch den Namen bzw. die Item-Nummer (bei HL7Field) oder die Position (bei HL7-Component) konkretisiert wird. Jede Instanz enthält beliebig viele Einträge der darunterliegenden Ebene. Enthält sie keinen Eintrag der darunterliegenden Ebene, so sind automatisch alle möglichen darunterliegenden Vokabular-Einträge implizit eingeschlossen. Aufgrund der potentiell großen Anzahl von HL7Field-Items unterhalb eines HL7Segments, erlaubt das XML-Schema den Ausschluss

[1] siehe [29], [30]

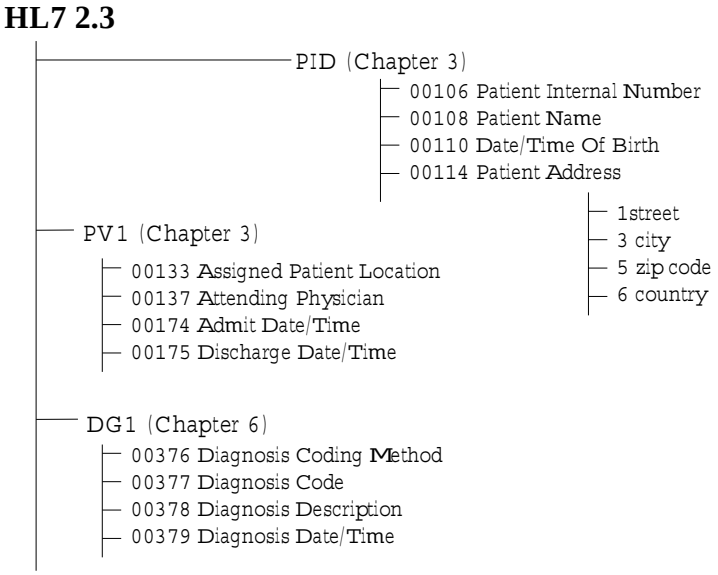

Abbildung 4.10: Beispiel-Vokabularstruktur auf der Basis von HL7-Bezeichnern

von HL7Field-Elementen. Das Attrubut „mode“ erhält dann den Wert „exclude“. Mit dem Elementtyp „restriction“ sind inhaltliche Einschränkungen möglich. Durch die Angabe der HL7-Item-Nummer (number) und – falls notwendig – der Komponenten-Position (position) wird das betroffene Daten-Item innerhalb des Segments bestimmt; „value“ enthält einen Ausdruck, der im Wert des Eintrags enthalten sein muss (mode=“include“) oder nicht enthalten sein darf (mode=“exclude“). Der „restriction“-Typ ist ein sehr grober, vereinfachter Lösungsansatz für die Definition inhaltlicher Ausschlusskriterien. Sie erfordert mehr als die Analyse einzelner Feldinhalte. Eine einfache Erweiterung wäre die Analyse von logisch verknüpften Feldkombinationen. Die vorliegende Arbeit enthält keine weiteren Konzepte zur Definition inhaltlicher Ausschlusskriterien, da diese die ohnehin hohe Komplexität der Datensichtdefinitionen noch erhöhen. Höhere Priorität haben Konzepte zur Vereinfachung und Handhabung der Rechtedefinition und -auswertung.

XML-Schema:

```
<?xml version="1.0" ?>
<schema xmlns="urn:W3C.org:xmlschema" xmlns:dt="urn:W3C.org:xmldatatypes">
<elementType id="hl7View">
<element type="#name" occurs="REQUIRED"/>
<element type="#hl7Segment" occurs="ZEROORMORE"/>
  <attribute name="chapter"/>
</elementType>

<elementType id="hl7Segment">
```

```
<element type="#name" occurs="REQUIRED"/>
<element type="#hl7Field" occurs="ZEROORMORE"/>
  <attribute name="mode" atttype="ENUMERATION"
                                values="include exclude" default="include"/>
<element type="#restriction" occurs="ZEROORMORE"/>
</elementType>

<elementType id="hl7Field">
<element type="#number" occurs="REQUIRED"/>
<element type="#name" occurs="OPTIONAL"/>
<element type="#hl7Component" occurs="ZEROORMORE"/>
</elementType>

<elementType id="hl7Component">
<element type="#position" occurs="REQUIRED"/>
<element type="#name" occurs="OPTIONAL"/>
</elementType>

<elementType id="restriction">
<element type="#number" occurs="REQUIRED"/>
<element type="#position" occurs="OPTIONAL"/>
<element type="#value" occurs="ONEORMORE"/>
    <attribute name="mode" atttype="ENUMERATION"
                           values="include exclude" default="include"/>
</elementType>

<elementType id="name">
<string/>
</elementType>

<elementType id="number">
<int/>
</elementType>
</schema>
```

Beispiel: „*Aufnahmedaten ohne HIV-Diagnosen*“

Die folgende Datensicht schließt Personendaten (PID) mit Ausnahme der Religionszugehörigkeit, administrative Daten der Patientenkontakte (PV1), Befunde (OBX), Anamnestische Daten (AL1) und Diagnosen (DG1) ein. Von den Diagnostischen Daten sind diejenigen ausgeschlossen, die in der Beschreibung die Zeichenkette „HIV“ enthalten.

```
<hl7View>
    <name> Arzt_ADT </name>
    <hl7Segment chapter=3>
        <name> PID </name>
        <hl7Field mode="exclude">
            <item> 00120 </item>
            <name> Religion <name>
        </hl7Field>
     </hl7Segment>
```

```
      <hl7Segment chapter=3>
         <name> PV1 </name>
      </hl7Segment>
     <hl7Segment chapter=7>
         <name> OBX </name>
      </hl7Segment>
      <hl7Segment chapter=3>
         <name> AL1 </name>
      </hl7Segment>
      <hl7Segment chapter=6>
         <name> DG1 </name>
         <restriction>
          <data>00378</data>
             <value mode="exclude"> "HIV" </value>
         </restriction>
      </hl7Segment>
   </hl7View>
```

Abfrage

Die Schnittstelle des Zugriffskontrollsystems sieht grundsätzlich die gezielte Abfrage von (u. U. vollständig spezifizierten) Rechten vor. Dabei enthalten die Abfragestrukturen Merkmale der gesuchten Rechte, z. B. die darin referenzierten und im Vokabular registrierten Bezeichner für Operationenen und Datensichten. Ordnet man die Datensichtbezeichner im Vokabular so an, dass Datensichten als Teilmengen anderer Datensichten erkannt werden, so kann die Suche über einen Datensichtbezeichner auch alle eingeschlossenen Datensichten liefern. Als wesentliche Funktion der Datensichten ist eher die Lieferung verfeinerter Information für die Anzeige oder Weitergabe von Daten zu sehen als die Lieferung einer Formulierungsgrundlage für gezielte Abfragen.

Auswertung

Die Auswertung der Datensichten findet in den Informationssystemen statt. Dazu müssen die vom Zugriff betroffenen Daten mit der Datensicht abgeglichen werden können. Grundsätzlich sind a priori verbindliche Vereinbarungen zwischen den Betreibern des Zugriffskontroll-Systems und den Herstellern eines Informationssystems bezüglich Struktur und Inhalt des Vokabulars notwendig. Der Aufwand der Abfrage und Auswertung der Rechtedefinitionen hängt nicht zuletzt von der Komplexität der Datensichtdefinitionen ab. Bei vollständig spezifizierten Abfragen entfällt die Auswertung von Datensichten im Informationssystem.

Informationssysteme, die ihre Daten bereits als XML-Dokumente verwalten, können die Datensicht-Auswertung in einen Mechanismus integrieren, der die Datensichten in spezifische Stylesheets übersetzt, die dann die Darstellung der Daten steuern.

4.2.5 Zugriffstickets

Zusätzlich zur Abfrage der aktuellen Berechtigungen eines Subjekts ist die Übermittlung von Berechtigungsinformationen in Form eines text-basierten Zugriffstickets vorgesehen. Dazu wird das Ticket einmalig zu Beginn einer Benutzer-Sitzung beim Zugriffskontroll-System angefordert – z. B. durch einen spezielles Programm, das das Ticket auf dem Arbeitsplatzrechner ablegt. Das Zugriffsticket enthält die wesentlichen Berechtigungs-Informationen, nämlich die Subjekt-Id, die Rolle, in der der Zugriff erfolgen soll, die zugehörige Organisationseinheit, den Gültigkeitszeitraum des Tickets und die Liste der Rechte. Es wird vom Zugriffskontroll-Server digital signiert, und ist dadurch in Verbindung mit dem Benutzerzertifikat fälschungssicher.

Systeme, in die keine Schnittstelle zum Zugriffskontroll-Server integriert werden soll oder kann, können das Zugriffsticket z. B. im Anmeldedialog entgegennehmen, und dadurch Informationen über die Rechte des angemeldeten Benutzers erhalten. Die Rechte im Zugriffsticket können keine konkreten Referenzen auf Patientenakten enthalten, wie die Strukturen in der Ergebnismenge einer Abfrage, da man erstens von einer wenigstens vierstelligen Zahl von betroffenen Fallreferenzen zu einer Rolle ausgehen muss, d. h. Tickets wären nicht mehr effizient zu übermitteln und auszuwerten, und zweitens die dynamischen Anteile der Rechte nur für kurze Zeit aktuell sind.

Die Rechte im Ticket enthalten deshalb nur die abstrakten Kontextdefinitionen, die vom Empfänger des Tickets interpretiert oder ignoriert werden müssen. Auch Datensicht-Definitionen dürfen nicht beliebig komplex sein, wenn eine effiziente Auswertung möglich sein soll. Infolgedessen kann das Zugriffsticket nur für spezielle Systeme und Zugriffskonstellationen sinnvoll eingesetzt werden, z. B. wenn alle Benutzer eines Systems grundsätzlich auf alle Patientenakten zugreifen sollen oder ein System nur Subjekt-ID und Rollenbezeichner des zugreifenden Benutzers auswertet. Die gezielte Definition speziell gekennzeichneter Rechte für das Zugriffsticket, ist dann eine mögliche Lösung der aufgeführten Probleme. Grundsätzlich kann das Zugriffsticket auch mit Anfragen an das Zugriffskontrollsystem kombiniert werden. Für die Definition der Zugriffstickets wurde eine XML-basierte Repräsentation gewählt, die folgendem Schema entspricht:

```
<?xml version="1.0" ?>
<schema xmlns="urn:W3C.org:xmlschema" xmlns:dt="urn:W3C.org:xmldatatypes">
<elementType id="ticket">
<element type="#user" occurs="REQUIRED" />
<element type="#role" occurs="REQUIRED" />
<element type="#unit" occurs="REQUIRED" />
<element type="#validity" occurs="REQUIRED" />
<element type="#permission" occurs="ZEROORMORE" />
<element type="#signature" occurs="REQUIRED" />
</elementType>

<elementType id="unit">
<element type="term" occurs="REQUIRED" />
<element type="vocabulary" occurs="OPTIONAL" />
</elementType>
```

```
<elementType id="validity">
<element type="notBefore" occurs="REQUIRED" />
<element type="notAfter" occurs="REQUIRED" />
</elementType>

<elementType id="permission">
<element type="#trContext" occurs="REQUIRED" />
<element type="#dataView" occurs="OPTIONAL" />
<element type="#action" occurs="ONEORMORE" />
<element type="#source" occurs="OPTIONAL" />
<element type="#condition" occurs="ZEROORMORE" />
</elementType>

<elementType id="trContext">
<element type="#name" occurs="REQUIRED" />
<element type="#vocabulary" occurs=" REQUIRED" />
<element type="#parameter" occurs="ZEROORMORE" />
</elementType>

<elementType id="parameter">
<element type="#key" occurs="REQUIRED" />
<element type="#value" occurs="REQUIRED" />
</elementType>

<elementType id="action">
<element type="#term" occurs="REQUIRED" />
<element type="#vocabulary" occurs=" REQUIRED" />
</elementType>

<elementType id="source">
<element type="#term" occurs="REQUIRED" />
<element type="#vocabulary" occurs=" REQUIRED" />
</elementType>

<elementType id="dataView">
<element type="#viewName" occurs=" REQUIRED" />
<element type="#viewDefinition" occurs="REQUIRED"/>
</elementType>

<elementType id="viewName">
<element type="#term" occurs=" REQUIRED" />
<element type="#vocabulary" occurs=" REQUIRED" />
</elementType>

<elementType id="viewDefinition">
<element type="#term" occurs=" REQUIRED" />
<element type="#format" occurs=" REQUIRED" />
<element type="#vocabulary" occurs=" REQUIRED" />
</elementType>

Alle nicht weiter spezifizierten Typen sind vom Typ :
<elementType id="*">
<string />
</elementType>
...
</schema>
```

4.2.6 Das kontrollierte Vokabular

Das kontrollierte Vokabular hat die Funktion, alle Begriffe, die bei der Definition, Abfrage und Auswertung von Berechtigungen notwendig sind, zusammen mit Informationen über ihre Bedeutung und mögliche Synonyme, strukturiert (i. A. in einer Hierarchie) zu registrieren und zur Verfügung zu stellen. Es ermöglicht damit die semantisch korrekte Kommunikation zwischen voneinander unabhängigen Teilen eines verteilten Systems. Das Vokabular, das als funktionale Einheit realisiert wird, ist in Bereiche für Operationsbezeichner, Datensichtbezeichner, und Kontextprototypen, Bezeichner für Organisationseinheiten und referenzierbare klinische Informations-Systeme sowie Datenklassen-Bezeichner für die Definition der Datensichten untergliedert. Jeder der Bereiche kann wiederum verschiedene voneinander unabhängige Vokabulare enthalten, die jeweils durch ihren Namen bestimmt sind. Für jeden Bereich gibt es ein Vokabular mit dem Namen „UKMZIntern“, das nicht-standardisierte klinikspezifische Bezeichner enthält.

Insbesondere für die Referenzierung der Datenklassen in den Datensichten sind verschiedene Vokabulare denkbar. Diese Vokabulare können auf der Basis von standardisierten Terminologien aufgebaut werden; sie können aber auch die spezifische Terminologie eines Informationssystems oder allgemeine Vorgaben des Klinikums enthalten.

Struktur

Die Struktur des Vokabulars soll die Zusammenhänge zwischen den einzelnen Begriffen wiederspiegeln. Deshalb sind die Begriffe in einer Baumstruktur hierarchisch angeordnet. Das heißt insbesondere, dass alle Begriffe, die im Teilbaum unterhalb eines Begriffsknoten stehen, in diesem logisch (im Sinne eines Teils oder einer Spezialisierung) enthalten sind. Für jede Knotenebene kann ein spezifischer Bezeichner erfasst werden, der bei den Datenklassen u. a. als XML-Tag für die Datensicht-Repräsentation verwendet wird. Jeder Begriff ist nur durch den vollständigen Pfad im Vokabular eindeutig bestimmt. Somit können Begriffe, die in unterschiedlichen Zusammenhängen verwendet werden (Homonyme), mit dem gleichen Bezeichner unterscheidbar registriert und verwendet werden.

Die hierarchische Anordnung erlaubt eine Vereinfachung der Definitionen, da immer der Begriff mit der kleinstnotwendigen Granularität ausgewählt werden kann. Die Struktur des Vokabulars hat außerdem Einfluss auf die Möglichkeiten der Abfrage von Rechten. Bei der Anfrage eines Informationssystems muss beispielsweise erkannt werden, ob die abgefragte Operation in der Klasse von Operationen enthalten ist, zu denen das Subjekt berechtigt wurde. Somit hängt der Umfang der Abfragemöglichkeiten davon ab, wie korrekt und vollständig semantische Zusammenhänge zwischen Begriffen abgebildet werden können. Beispiele für Vokabularstrukturen folgen in Abschnitt 5.2.1.

Das Vokabular für die Kontextprototypen hat eine flache Struktur. Einträge von Prototypen enthalten außer dem Namen der Kontextregel und dem Namen des Vokabulars die Liste der Argumente, die bei der Rechtedefinition angegeben

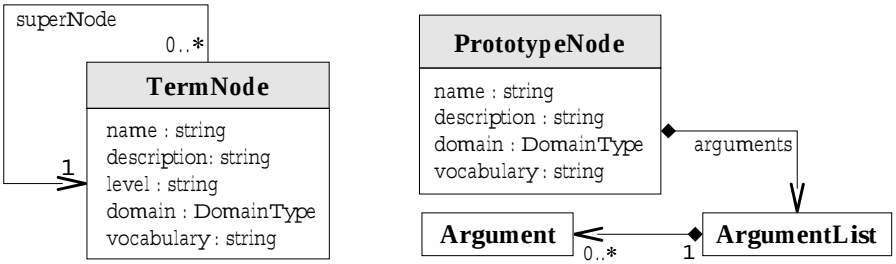

Abbildung 4.11: Informationsstrukturen des kontrollierten Vokabulars

werden müssen.

Äquivalenz von Vokabularen

Bei der gleichzeitigen Verwendung verschiedener voneinander unabhängiger Vokabulare für die Datensicht-Definitionen, stellt sich unweigerlich die Frage nach der Äquivalenz von Begriffen und Konstrukten. Dies führt weiter zur Frage nach der Vollständigkeit und der Granularität von Vokabularen. Es ist offensichtlich, dass die Übersetzung einer Datensicht in ein anderes Vokabular nicht grundsätzlich möglich ist, weil beispielsweise der Begriff des einen Vokabulars nicht oder mit einer zusätzlichen unerwünschten Bedeutung im anderen Vokabular existiert. Der Aufwand, alle Vokabulare auf ein Referenzvokabular abzubilden, um so die Übersetzbarkeit zu überprüfen und eventuell herzustellen, ist unangemessen groß im Verhältnis zum Nutzen, der Vermeidung von Mehrfachdefinitionen und potentiellen Fehlern. Bei den ersten Schritten im Aufbau eines komplexen Zugriffskontroll-Systems hat man einen größeren Nutzen von gut gestalteten Erfassungswerkzeugen, die Datensichten gut lesbar präsentieren und damit eine zuverlässige „manuelle“ Konsistenzkontrolle ermöglichen.

Beispiele standardisierter Vokabulare

HL7 Die Notwendigkeit von Interoperabilität verteilter Systeme im Gesundheitsbereich erfordert verbindliche Vokabulare in verschiedenen Zusammenhängen. Für die Kommunikation zwischen klinischen Systemen ist der Kommunikationsstandard HL7 [29, 30] weit verbreitet. Das zugrundeliegende Vokabular definiert Daten, die zwischen klinischen Systemen ausgetauscht werden können. Der HL7-Standard ist in verschiedene Kapitel untergliedert; jedes Kapitel enthält definierte Typen von Nachrichten, die bei bestimmten Ereignissen ausgetauscht werden. Jede Nachricht besteht aus einzelnen Segmenten, die Gruppen von logisch zusammengehörenden Datenfeldern enthalten. Für jedes Datenfeld ist ein Datentyp definiert, der wiederum in Komponenten und Subkomponenten unterteilt sein kann. Die Bedeutung jedes Daten-Items ist genau beschrieben. Seg-

mente, in denen Patientendaten zusammengefasst sind, beschreiben implizit einen Auschnitt der elektronischen Patientenakte. Sie sind deshalb auch als Grundlage für ein Vokabular geeignet, mit dem Datenklassen definiert werden können. Ein großer Vorteil des HL7-Vokabulars ist die Verbreitung des Standards u. a. im deutschen Gesundheitswesen. Die meisten Entwickler von klinischen Informationssystemen sind mit dem HL7-Standard (bis Version 2.3) wenigstens in den wesentlichen Kapiteln vertraut. Die meisten Systeme besitzen bereits HL7-Kommunikationsschnittstellen für administrative Vorgänge und Befundübermittlung. Für ein System, das eine HL7-Kommunikationsschnittstelle besitzt, existiert bereits eine teilweise Abbildung des internen Datenmodells auf die HL7-Datenstrukturen. Bei der Modellierung von HL7-basierten Datensichten kann auf dieses Wissen aufgebaut werden.

Ein Vokabular für klinische Dokumente Die „Document Ontology Task Force (DOTF)“ [14], eine Arbeitsgruppe der HL7-Organisation entwickelt eine Ontologie, die die Klassifikation von klinischen Dokumenten ermöglichen soll. Die Gruppe schlägt eine polyhierarchische Einordnung von Dokumentnamen mit Hilfe von vier terminologischen Achsen vor. Jedes Dokument wird bezüglich der durchgeführten Leistung (*service*), der Bedingungen (*conditions*), der klinischen Kategorie (*clinical category*) und des klinischen Umfelds (*practice setting*) eingeordnet. Hinter jeder der Achsen steht wiederum ein standardisiertes Vokabular. „*Services*“ werden durch die SNOMED RT Prozeduren beschrieben, „*Conditions*“ durch SNOMED RT findings, conclusions und/oder assessments, „*Clinical Categories*“ entstammen dem HL7 Reference Information Model (ab HL7 Version 3), ebenso die Bezeichner für das „*Practice Setting*“. Für Zugriffsberechtigungen, die klinische Dokumente betreffen, kann dieses Vokabular eine geeignete Grundlage darstellen.

Anmerkung

Standardisierte Vokabulare haben den Vorteil, dass sie von vorneherein für alle Nutzer verbindlich feststehen. Sie enthalten das Wissen und die Erfahrung zahlreicher Experten im jeweiligen Bereich und sind deshalb i. a. umfassend und gut strukturiert. Grundsätzlich bleibt aber immer die Frage, ob die existierenden standardisierten Vokabulare die Daten der Krankenakte vollständig, in ausreichender Granularität erfassen und geeignet strukturieren. Diese Frage wird in der vorliegenden Arbeit nicht weiter systematisch untersucht. Sie bleibt offen für weitere Arbeiten.

4.3 Das Applikationsmodell

In den folgenden Abschnitten wird die Gliederung der Zugriffsschichten in funktionale Module sowie die Funktionsweise und das Zusammenwirken der Objekte in den Modulen beschrieben.

4.3.1 Wissensakquisition

Die Zugriffsschicht für die Wissensakqusition bietet Schnittstellen für verschiedene unabhängige Aufgaben, die von verschiedenen Client-Programmen, hier den notwendigen grafischen Erfassungswerkzeugen, integriert werden können. Die einzelnen Aufgaben sind: die Definition der Rollenrechte, der Datensichten, der Kontextregeln, der Rollen und Rollenhierarchie sowie die Autorisierung, d. h. die Zuweisung der Rollen an Subjekte und die Definition von subjektspezifischen Rechten. Innerhalb jedes Moduls muss außerdem die Zulässigkeit der administrativen Zugriffe überprüft und alle Manipulationen protokolliert werden. Die Funktionalität wird dabei hauptsächlich vom **PolicyAdmin**-Objekt gesteuert. Es verwaltet Rechte und Datensichten und alle weiteren Objekte, die an der Wissensakqusition beteiligt sind.

Zugriffskontrolle

Die Zugriffskontrolle für die Verwaltung der Zugriffspolitik und der Autorisierungen wird durch **AdmAccessControl**-Objekte realisiert. Sie werden für jede Sitzung eines Administrators mit dessen Rechten erzeugt. Jedes Objekt, auf das das Client-Interface direkten Zugriff hat, erhält eine Referenz des zugehörigen AdmAccessControl-Objekts und kann so die Zulässigkeit eines Zugriffs überprüfen. Jedes der öffentlichen Interfaces implementiert außerdem eine Funktion, die die Namen aller geschützten Operationen zurückliefert. Diese Information wird automatisch im Vokabular erfasst und liefert die Operationsbegriffe zur Definition der Adminstratorrechte.

Definition der Rechte

Zugriffsrechte werden rollen-unabhängig definiert. Dabei wird zwischen den Rechten zum Zugriff auf Patientendaten und den Administratorrechten unterschieden; sie können insbesondere bei der Rollendefinition nicht kombiniert werden. Das PolicyAdmin-Objekt steuert die Erfassung und Verwaltung der Rechtedefinitionen. Es stellt dabei auch die Konsistenz der Rollendefinitionen beim Verändern und Löschen von Zugriffsrechten sicher und überprüft mit Hilfe des AdmAccessControl-Objects die Zulässigkeit der Operation. Inhaltliche Grundlage der Rechtedefinitionen sind die Bezeichner für Operationen, Datensichten und Kontextregeln, die im kontrollierten Vokabular registriert sind. Für die Bearbeitung der Rechte erhält der PermissionDefinition-Client Zugriff auf das **Vocabulary**-Objekt. Das Vocabulary-Objekt besitzt Methoden zur Erfassung und Manipulation der Begriffe und Begriffshierarchien aller Bereiche sowie der Kontextprototypen und zur gezielten Abfrage der Einträge. Die Definitionen der Datensichten und Kontextregeln werden unabhängig von der Rechtedefinition bearbeitet. Bei Löschen oder Manipulieren von Datensichten und Kontextregeln überwacht das PolicyAdmin-Objekt die Konsistenz der Rechte.

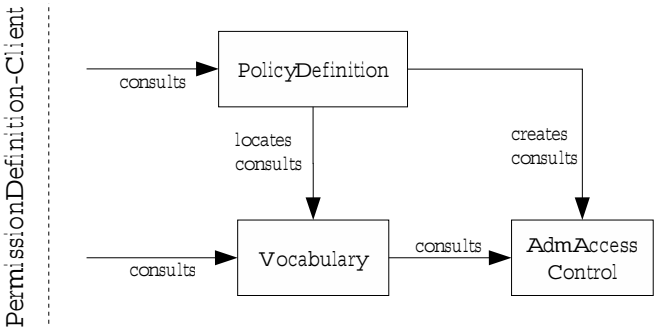

Abbildung 4.12: Definition von Zugriffsrechten

Definition von Datensichten

Das PolicyAdmin-Objekt liefert Funktionen zur Erfassung und Manipulation der DataView-Objekte. Es liefert insbesondere eine Referenz auf das Vocabulary-Objekt, das einerseits die Daten-Bezeichner zur Verfügung stellt, die in den DataView-Definitionen verwendet werden können, und in dem andererseits die Datensichten mit einem eindeutigen Bezeichner registriert werden. Das PolicyAdmin-Objekt bietet auch eine Schnittstelle, die es dem Vocabulary-Objekt bei der Manipulation von Einträgen jeglicher Art ermöglicht, zu überprüfen, ob der entsprechende Bezeichner bereits in einer Definition verwendet wird. Die Verwaltung der Datensichten wird durch die gleichen Objekte gesteuert wie die Verwaltung der Rechte; für die Integration in einen DataViewDefinition-Client und das Zusammenwirken der Objekte gilt analog die Darstellung in Abbildung 4.12.

Kontext-Definition

Das **ContextEvaluator**-Objekt steuert sowohl die Definition als auch die Auswertung der Kontextregeln. Es besitzt Funktionen zur Erfassung und Manipulation der spezifischen Regeltypen, die in Abschnitt 4.2.3 beschrieben wurden, und Funktionen, die zu einem Kontext, d. h. zu einer Kontextregel mit den spezifizierten Argumenten, und dem aktuellen Zeitpunkt die zugehörigen Behandlungsphasen ermittelt und zurückliefert. Bezeichner z. B. von Organisationseinheiten, die bei der Kontextdefinition verwendet werden, werden beim Vocabulary-Objekt abgefragt. Das ContextEvaluator-Objekt steuert außerdem die Registrierung der Kontextprototypen im Vokabular.

Rollen-Definition

Das **RoleHierarchy**-Objekt, verwaltet Rollen, Rollen-Rechte-Zuordnungen, Rollen-Rollen-Beziehungen und deren Konsistenz. Bei der Rollendefinition bzw. -manipulation, wird zunächst das RoleHierarchy-Objekt angesprochen, um eine

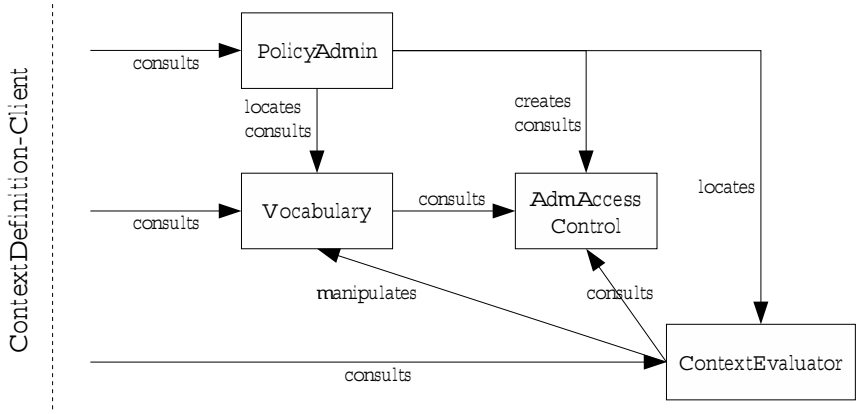

Abbildung 4.13: Definition von Kontexten und Datensichten

Rolle entweder auszuwählen oder anzulegen und in die Hierarchie einzuordnen. Dabei können Rollenconstraints, z. B. vorausgesetzte Rollen und wechselseitiger Ausschluss, definiert werden. Eine Rolle kann geändert und Rechte hinzugefügt oder entfernt werden. Die Rechte für diese Funktion liefert des PolicyAdmin-Objekt. Beim Löschen von Rollen stellt das RoleHierarchy-Objekt durch Konsultation des SubjectDirectory-Objekts sicher, dass die Rolle keinem Subjekt (mehr) zugewiesen ist.

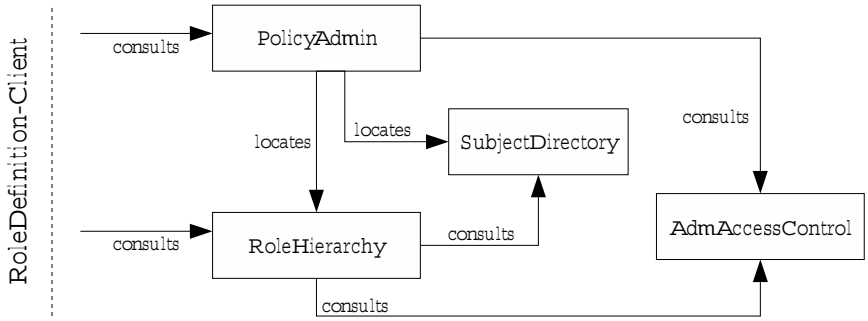

Abbildung 4.14: Rollen-Definition

Autorisierung

Die Autorisierung der Subjekte wird durch das **Authorization**-Objekt gesteuert. Es überprüft die Zulässigkeit des Zugriffs und konsultiert das RoleHierarchy- und das **SubjectDirectory**-Objekt. Das SubjectDirectory-Objekt verkapselt

die Anbindung des Benutzerverzeichnisses. Es liefert die Subjektinformation und steuert die Zuordnung der Rollen im Benutzerverzeichnis. Das RoleHierarchy-Objekt liefert die Rolleninformation.

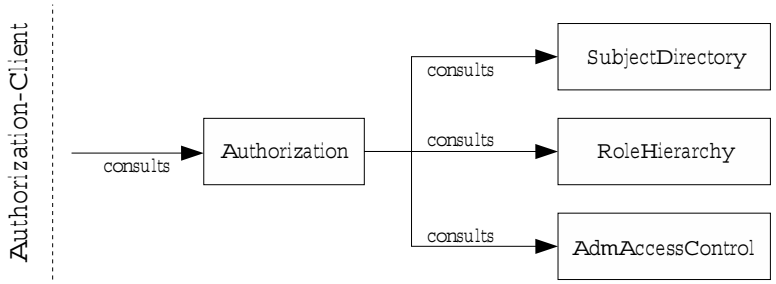

Abbildung 4.15: Autorisierung

Subjektrechte werden von normalen Benutzern oder Systemen (Nicht-Administratoren) gegeben, die das Recht besitzen, anderen Subjekten Zugriff auf spezifische Objekte zu gewähren. Die Erfassung dieser Rechte erfolgt automatisch als Folge einer Aktion (z. B. Anforderung eines Konsils) oder über ein geeignetes Freigabesystem, das wie ein Informationssystem behandelt wird, und demzufolge entsprechende Zugriffskontrollmechanismen integrieren muss. Die Freigabe von Daten muss dann als Operation registriert sein und in der Definition der Rechte referenziert werden können.

4.3.2 Zugriffskontrolle

Der AccessControl-Client, i. A. ein Informationssystem, das Benutzerrechte ermitteln möchte, integriert die Schnittstelle des **MasterAccessControl**-Objekts. Das MasterAccessControl-Objekt liefert Methoden, mit denen spezifische Rechtemengen oder Zugriffsentscheidungen bezüglich eines Subjekts und/oder einer Rolle und spezifischen Zugriffsmerkmalen (→ Abschnitt 4.2.2) ermittelt werden können. Zur Überprüfung der Subjektdaten[2] konsultiert das MasterAccessControl das SubjectDirectory-Objekt. Mit der Subjekt-ID und dem Rollenbezeichner erfragt es die Rollen- und Subjektrechte beim RoleHierarchy- bzw. **SubjectPermissions**-Objekt. Das PolicyAdmin-Objekt liefert dabei die Datensicht-Definitionen, die in den Rechten mit Namen referenziert sind. Für die Umsetzung der Kontexte aus den Rollenrechten wird das **ContextEvaluator**-Objekt angesprochen, das die zugehörige Kontextdefinition ermittelt, mit den spezifizierten Parametern auswertet und eine Liste von Attribut-Mengen zurückgibt, mit denen die (Teil-)Krankenakten zu den Behandlungsphasen bestimmt werden können. Von den ermittelten Berechtigungen

[2]Ist das Subjekt registriert? Besitzt es die angegebene Rolle?

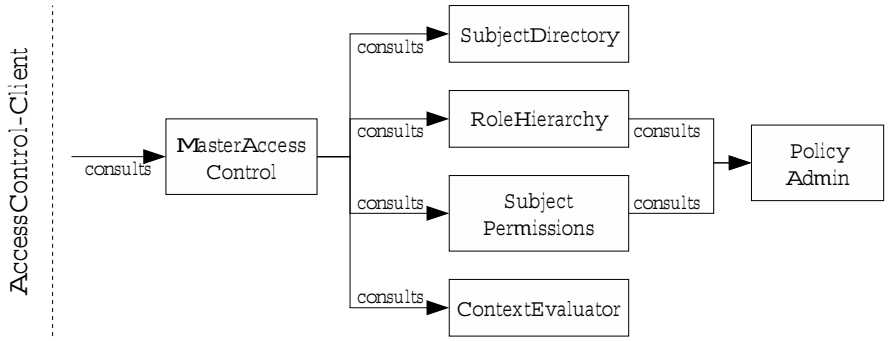

Abbildung 4.16: Abfrage der Zugriffsrechte

können verschiedene Ergebnismengen abgefragt werden: eine Liste der betroffenen Patienten, der betroffenen Verwaltungsfälle, der betroffenen Behandlungsphasen oder der vollständigen Berechtigungsinformationen. Eine weitere Funktion des MasterAccessControl-Objekts ist die Erzeugung der Zugriffstickets.

4.4 Schnittstellen

Die Funktionalität der verschiedenen Objekte des Zugriffskontroll-Systems wurde im Applikationsmodell grob beschrieben. Im folgenden sind die Schnittstellen der einzelnen Objekte angegeben. Da die Zugriffsschichten zum Teil schon jetzt und teilweise in Zukunft als CORBA-Dienste implementiert werden, sind die Strukturen und Schnittstellen in IDL-Syntax [21] formuliert. An Hand der Schnittstellen soll die Funktionalität detailliert dargelegt werden, wobei nicht der Anspruch einer exakten technischen Spezifikation gerfüllt wird. So wurde beispielsweise auf die Zusammenfassung zu Modulen verzichtet. Der erste Abschnitt liefert alle Strukturen für die Fehlerbehandlung in den beschriebenen Komponenten. Die Schnittstellen-Definitionen sind in den folgenden Abschnitten entsprechend ihrer überwiegenden Funktion nach Informationsserfassung und Informationsretrieval gruppiert. Einige der Schnittstellen beinhalten Funktionalität aus beiden Bereichen zu einer spezifischen Informationsart, weshalb die Trennung nicht streng durchgeführt werden kann. Die benötigten Datentypen sind immer dort definiert, wo sie zuerst referenziert werden. In den meisten Fällen sind die Datentypen nicht explizit erklärt, sondern erklären sich aus der Verwendung in den Funktionen und den Namen der Bezeichner.

4.4.1 Fehlerbehandlung

Die folgenden Definitionen beschreiben Typen und Strukturen für die Fehler- und Ausnahmenbehandlung geordnet nach Zusammenhängen, in denen sie auftreten, und werden im Anschluss erläutert.

Typen

```
enum ErrorScale { FATAL, NONFATAL };
enum ObjType { SUBJ_DEF, ROLE_DEF, PERM_DEF, VIEW_DEF };
enum RoleType { USER, ADMIN };
enum ConstraintType { WEAK_EXCLUSION, STRONG_EXCLUSION, PREREQUISITE_ROLE };
enum RelationType { INHERITANCE, CONSTRAINT };
enum VocDomain { D_OP, D_DATA, D_VIEW, D_CONTEXT, D_UNIT, D_SYSTEM };
```

Exceptions

```
// General technical exceptions

exception InternalError{
          ErrorScale escale;
};
exception InvalidParameter{
          string     pName;
          string     pValue;
};
exception DataBaseError{
          long       dbECode;
          string     dbEString;
          ErrorScale escale;
};
exception ConnectionError{
          long       sysECode;
          string     sysEString;
          ErrorScale escale;
};

exception LDAPError{
          long       ldapECode;ing
          string     ldapEMStr;
          ErrorScale escale;
};


// General semantic exceptions

exception UnknownID{
          ObjType   oType;
          string   id;
};
exception UnknownTerm{
          VocDomain domain;
          RegTerm   vocItem;
};
exception InvalidKey{
          string   key;
};
exception InvalidValue{
          string   key;
          string   value;
};

// Exceptions thrown by policy administration and authorization functions
```

```
exception IDInUse{
          ObjType    oType;
          string     id;
};
exception DuplicateID{
          ObjType    oType;
          string     id;
};
exception TermInUse{
          RegTerm    term;
};
exception DuplicateTerm{
          RegTerm    term;
};
exception RoleRoleConflict{
          RelationType relType;
          string     sRole;
          string     oRole;
};
exception RoleTypeConflict{
          string     roleID;
          RoleType   tNeeded;
};
exception ConstraintViolation{
          ConstrType cType;
          string     subjID;
          string     roleID;
};
exception PermissionDenied{
          string     action;
          ObjType    oType;
          string     id;
};
exception SyntaxError{
          string     errorReport;
};

// Exceptions thrown by information retrieval functions

exception TooManyRecords{
          long       threshold;
};
exception SubjectRoleConflict{
          string     subjID;
          string     roleID;
};

// Exceptions thrown by dynamic information acquisition functions

exception ObsoleteEvent{
         DateTime    currentTS
         DateTime    lastTS
};
```

Erläuterungen

ErrorScale beschreibt den Fehlergrad innerhalb einiger Exceptions. Variablen dieses Typs signalisieren, ob die Komponente aufgrund des Fehlers beendet werden sollte (FATAL) oder nicht (NONFATAL).

ObjType legt fest, welche Art von Objekt (Subjekt, Rolle, Recht, Kontextregel oder Datensicht) betroffen ist.

RoleType definiert den Rollentyp (Benutzer-Rolle oder Administrator-Rolle)

ConstraintType beschreibt, um welche Art von Rollenconstraint (Starker oder Schwacher Rollenausschluss, Vorausgesetzte Rolle) es sich handelt.

RelationType legt fest, um welche Rollen-Relation (Vererbungs-Relation oder Constraint-Relation) es sich handelt.

VocDomain definiert die gültigen Vokabular-Bereiche: Operationen, Daten, Datensichten, Kontextprototypen, Organisationseinheiten, Systeme.

InternalError Diese Exception signalisiert ein Problem, das durch fehlerhafte Programmlogik ausgelöst sein könnte.

InvalidParameter signalisiert einen allgemeinen Fehler in den übergebenen Parametern der Funktion, wie z. B. ungültige Formate für Zeitpunkte oder Begriffspfade, unzulässige Nullwerte etc. Die Exception liefert den Parameternamen und eine Beschreibung des unzulässigen Werts.

DataBaseError Alle Datenbankprobleme, die den Betrieb der Dienste beeinträchtigen wie z. B. Timeouts, Shutdown der Datenbank, Inkonsistenzen, werden mit dieser Exception angezeigt. Die Exception liefert den Datenbank-Fehlercode und die Fehlerbeschreibung.

ConnectionError Probleme, die in der Verbindung zwischen verschiedenen Komponenten auftreten, lösen die ConnectionError-Exception aus.

LDAPError Speziell für die Kopplung mit dem Benutzerverzeichnis liefert diese Exception spezifische Fehlermeldungen des LDAP-Protokolls.

UnknownID Jede Operation, die ein Objekt referenziert, löst diese Exception aus, wenn das Objekt nicht existiert. „`oType`“ gibt die Objekt- bzw. Datenart an (Rolle, Subjekt, ...)

UnknownTerm Diese Exception wird ausgelöst, wenn ein referenzierter Begriff im Vokabular nicht gefunden werden kann. „`tType`“ gibt den Vokabularbereich an, zu dem der Begriff gehört (Datensichten, Operationen, Kontexte, Datenbegriffe, ...). „`vocItem`“ enthalt den Begriff der nicht gefunden wurde; falls das spezifische Vokabular nicht existiert ist „`vocItem.term`“ leer.

InvalidKey Enthält eine Attributliste einen undefinierten Schlüsselbezeichner, so wird die InvalidKey-Exception ausgelöst. In „`key`“ wird der erste gefundene ungültige Schlüsselbezeichner zurückgeliefert.

InvalidValue Enthält eine Attributliste einen unzulässigen Wert, so wird die InvalidValue-Exception ausgelöst. Unzulässige Werte sind entweder nicht in der durch den Schlüssel festgelegten endlichen Wertemenge enthalten, oder sie besitzen ein falsches Format. In „`key`“ und „`value`“ wird das erste betroffene Attribut/Wert-Paar zurückgeliefert.

IDInUse Beim Löschen oder Verändern von Datenobjekten können Inkonsistenzen oder schwerwiegende Konsequenzen für alle Objekte auftreten, die das Datenobjekt referenzieren. Deshalb wird die IDInUse-Exception immer dann ausgelöst, wenn eine Operation aufgrund von bestehenden Referenzen nicht ausgeführt werden soll.

DuplicateID Jedes Datenobjekt wird unter einer eindeutigen ID abgelegt. bei dem Versuch ein neues Objekt mit der gleichen ID anzulegen wird die DuplicateID-Exception ausgelöst.

TermInUse Diese Exception entspricht der IDInUse-Exception in Bezug auf die Einträge des Vokabulars. Verwendete Begriffe dürfen weder gelöscht noch verschoben werden.

DuplicateTerm Existiert ein Begriff des Vokabulars bereits, so wird bei dem Versuch, ihn nochmal anzulegen, diese Exception ausgelöst.

RoleRoleConflict Bei allen Operationen, die Rollen miteinander in Beziehung setzen (Vererbungsrelationen, Constraints) können Konflikte auftreten. Beispielsweise dürfen Rollen für die ein „`STRONG_EXCLUSION`“-Constraint existiert, keine Vererbungsrelation zueinander besitzen. Würde durch eine Operation ein derartiger Konflikt auftreten, so wird diese Exception ausgelöst. „`RelationType`“ enthalt die Art der bereits bestehenden Relation.

RoleTypeConflict Diese Exception wird bei dem Versuch ausgelöst, einer Administrator-Rolle Benutzerrechte oder umgekehrt einer Benutzerrolle Administratorrechte zuzuordnen.

ConstraintViolation Eine Constraintverletzung bei der Subjekt-Rollen-Zuweisung löst diese Exception aus. „`ConstrType`“ enthält die Art des verletzten Constraints. Außerdem werden die Subjekt-ID und die Rollen-ID, die den Konflikt auslösen, zurückgeliefert.

PermissionDenied Diese Exception ist die Folge unzulässiger Administrator-Operationen. Sie zeigt an, wenn eine Operation (`action`) auf einem Objekt (`oType, id`) ausgeführt werden soll, das nicht im Zuständigkeitsbereich des Administrators liegt.

SyntaxError Datensichten und Kontextregeln enthalten Freitext-Bestandteile, die einer definierten Syntax entsprechen müssen. Beim Erfassen wird diese Syntax überprüft. Die Exception signalisiert einen Syntaxfehler. „`errorReport`“ liefert die Fehlerbeschreibung des internen Parsers.

TooManyRecords Bei allen Operationen des Informations-Retrievals, wo Rechtemengen geliefert werden, können aufgrund ungünstiger oder fehlender Suchraum-Einschränkungen sehr große Anzahlen von Datensätzen angefordert werden. Die Exception signalisiert, dass eine festgelegte Obergrenze für die Größe der Rechtemenge überschritten wurde. „`threshold`“ enthält diese Obergrenze.

SubjectRoleConflict Wird bei einer Anfrage an das MasterAccessControl-Objekt eine Kombination von Subjekt und Rolle angegeben, für die es im Benutzerverzeichnis keine Entsprechung gibt, so wird diese Exception ausgelöst. Sie liefert das unzulässige Subjekt/Rollen-Paar zurück.

ObsoleteEvent Bei der automatischen Erfassung der Behandlungsprozess-Instanzen wird diese Exception ausgelöscht, wenn Fehler durch chronologisch vertauschte Ereignismeldungen entstehen könnte.

4.4.2 Kontrolle der administrativen Zugriffe

AdmAccessControl-Interface

Das AdmAccessControl-Interface liefert die Zugriffskontroll-Funktionalität für die Administratorzugriffe. Für jede Sitzung wird ein AdmAccessControl-Objekt mit der Rechte-Liste des angemeldeten Administrators erzeugt. Jedes Objekt, das Funktionen für Administrations-Klienten anbietet, erfragt vor der Ausführung einer zugriffsbeschränkten Funktion die Zulässigkeit des Zugriffs.

```
typedef struct AdminInfo {
               string        subjId;
               string        realName;
               string        roleId;
};
typedef enum ScopeKey {ROLE_UNIT, ROLE_ID, JUNIORROLE_ID,
                       SUBJECT_ID, SUBJECT_UNIT, SUBJECT_QUAL, SUBJECT_TYPE,
                       SYSTEM_ID, ADMIN_UNIT};
typedef struct ScopeAttrib{
               ScopeKey       key;
               string         value;
};
typedef sequence<ScopeAttrib> ScopeSeq;
```

ScopeAttrib definiert ein spezielles Attribut/Wert-Paar für die Definition von Zuständigkeitsbereichen für Operationen auf den Daten der Zugriffspolitik. Die Einträge klassifizieren Rollen, Rechte und Subjekte. Zur Interpretation der Schlüsselbegriffe siehe Abschnitt 4.2.1, Tabelle

4.1. Dieser Datentyp wird auch im RoleHierarchy-, Authorization- und SubjectDirectory-Interface verwendet.

```
interface AdmAccessControl{
    readonly attribute AdminInfo      admin;
    readonly attribute AdmPermSeq     permissions;
             attribute RoleHierarchy  roleControl;

                       AdmAccessControl(in  AdminInfo adm)
    raises(InternalError, UnknownID, DataBaseError);

    boolean            checkAccess(in  RegTerm operation, in ScopeSeq scope)
    raises(InternalError, UnknownTerm, InvalidKey, InvalidValue);
};
```

readonly attribute AdminInfo enthält alle Angaben zum Administrator, die für die Ermittlung der Rechte und während einer Sitzung, z. B. für die Protokollierung der Administrator-Zugriffe, benötigt werden.

readonly attribute AdmPermSeq enthält die Rechte-Liste des aktuell angemeldeten Administrators.

attribute RoleHierachy enthält die Referenz des RoleHierachy-Objekts (der aktuellen Sitzung) zur Überprüfung von Rollenrelationen.

AdmAccessControl() Mit diesem Konstruktor wird das AdmAccessControl-Objekt für den jeweiligen Administrator initialisiert, wobei auch die zugehörigen Rechte ermittelt werden.

checkAccess() überprüft die Zulässigkeit der Operation „`operation`“ auf den Bereich, der durch „`scope`“ definiert ist. Der betroffene Bereich wird auf der Basis von Rollen-, Rechte- und Subjekt-Klassen beschrieben. Falls der Administrator berechtigt ist, die Operation auszuführen, d. h. falls ein Eintrag „`perms[i].operation == operation`“ existiert, wird der betroffene Bereich „`scope`“ mit dem berechtigten Bereich „`perms[i].scope`“ verglichen. Ist kein direkter Vergleich der Bereichsdefinitionen möglich, so ermittelt das AdmAccessControl-Objekt zusätzliche Informationen, z. B. ob eine Vererbungsrelation zwischen der betroffenen Rolle und einer Junior-Rolle des Berechtigungsbereichs existiert. Der Rückgabewert ist entweder „`true`“, wenn das Recht zum Zugriff besteht oder anderfalls „`false`“.

4.4.3 Rollendefinition und Autorisierung

Die Autorisierung, d. h. die Zuweisung der Rollen und Rechte an Subjekte sowie die Rollen- und Rechtedefinition, geschieht mit Hilfe geeigneter grafischer Benutzerwerkzeuge. Die Definitionswerkzeuge mit den zugehörigen Client-Objekten sowie die Datenbankstrukturen für die Speicherung der Informationen werden zur Zeit im Rahmen einer Diplomarbeit entwickelt. Deshalb folgt an dieser Stelle nur die Definition der Schnittstelle, die den Funktionsumfang der Services festlegt, sowie Anforderungen und Vorschläge für die Gestaltung der Benutzeroberflächen, jedoch keine Spezifikation der Erfassungswerkzeuge. Die Datenstruktur

der Objekte, deren Interface nicht explizit beschrieben ist, kann den grafischen Darstellungen des Informationsmodells entnommen werden.

PolicyAdmin-Interface

Das PolicyAdmin-Interface ist der Teil der Schnittstelle, der bei der Definition und Manipulation der Rollen und Rechte initial von einem Benutzerinterface-Client angesprochen wird. Außer den Funktionen zur Steuerung der Erfassung und Manipulation von Rechten und Datensichten liefert es auch die Objektreferenzen einiger anderer Objekte, die an der Definition der Zugriffspolitik mitwirken. Die Klassen, die Rechte und Datensichten repräsentieren, haben keine bzw. geringe Funktionalität und werden deshalb in diesem Abschnitt bei den Datenstrukturen aufgeführt.

```
// Basic Classes

interface Permission {
    attribute string       permId;
    attribute RegTerm      operation;
};
interface UserPermission : Permission {
    attribute RegTerm      viewName;
    attribute RegTerm      source;
    attribute AttributeSeq conditions;
};
interface AdminPermission : Permission {
    attribute ScopeSeq     admScope;
};
interface DataView {
             attribute RegTerm     name;
             attribute DVDef       definition;
    readonly attribute SyntaxState syntax;

    SyntaxState       syntaxCheck(out string errorReport);
};
```

interface Permission, UserPermission, AdminPermission beschreiben die Klassen, die die Struktur der Rechte entsprechend dem Informationsmodell abbilden. Die Permission-Klasse ist die abstrakte Oberklasse, von der weitere Klassen abgeleitet sind.

interface DataView kapselt die Definition einer Datensicht. Die Klasse beinhaltet außerdem für jedes erlaubte Format, die zugehörige Syntaxprüfroutine, die durch die Funktion **syntaxCheck()** aufgerufen wird. Der Rückgabewert liefert den Status der Prüfung (`NO_CHECK, CORRECT, INCORRECT`), `errorReport` enthält die Ausgabe des internen Parsers.

```
// Datatypes

typedef enum SyntaxState {NO_CHECK, CORRECT, INCORRECT};

typedef sequence<AdminPermission> AdminPermSeq;
typedef sequence<UserPermission> UserPermSeq;
```

```
struct RegTerm{
       string         vocabulary;
       string         term;
};
typedef sequence<RegTerm> TermSeq;

struct AttributeStruct{
       string         key;
       string         value;
};
typedef sequence<AttributeStruct> AttributeSeq;

typedef enum TEKey {PID, MED_ID, ADM_ID, LOCATION, ETYPE, TTYPE, PART_UNIT,
                    PART_ROLE, BEGIN, END};
struct TEAttribute{
       TEKey          key;
       string         value;
};

typedef sequence<TEAttribute> TEAttributeSeq;

struct ContextStruct{
       RegTerm        name;
       TEAttributeSeq parms;
};

struct DVDef{
       string         vocabulary;
       string         format;
       string         rep;
};
```

RegTerm referenziert einen Begriff, der im kontrollierten Vokabular registriert ist; dabei enthält „`vocabulary`“ den Namen des Vokabulars, und „`term`“ den hierarchischen Bezeichner des Begriffs.

AttributeStruct enthält ein unspezifisches Attribut-/Wert-Paar.

TEAttribute definiert ein spezifisches Attribut-/Wert-Paar, dessen Attribut (vom Typ `TEKey`) die Eigenschaft einer Behandlungsphase des Prozessmodells beschreibt.

ContextStruct enthält eine Kontext-Definition bestehend aus dem Namen der zugehörigen Kontextregel und einer Liste von `TEAttrib`-Strukturen.

DVDef enthält den Definitionsteil des DataView-Objekts.

```
interface PolicyAdmin {
readonly attribute AdmAccessControl admAC;
readonly attribute RoleHierarchy roleControl;
readonly attribute SubjectDirectory subjectControl;
readonly attribute Vocabulary vocabularyControl;
readonly attribute ContextEvaluator contextControl;
```

```
long              getOperations(out TermSeq operations)
raises(InternalError);

long              getUserPermissions(out UserPermSeq perms)
raises(InternalError, DataBaseError);
long              getAdminPermissions(out AdminPermSeq perms)
raises(InternalError, DataBaseError);

UserPermission    addUserPermission(
                  in  string         pname,
                  in  ContextStruct  trContext,
                  in  RegTerm        operation,
                  in  RegTerm        dview,
                  in  RegTerm        source,
                  in  AttributeSeq   conds
)
raises(InteralError, DataBaseError, DuplicateID, UnknownTerm,
       InvalidKey, InvalidValue);

UserPermission    updateUserPermission(
                  in  string         pname,
                  in  ContextStruct  trContext,
                  in  RegTerm        operation,
                  in  RegTerm        dview,
                  in  RegTerm        source,
                  in  AttributeSeq   conds
)
raises(InternalError, DataBaseError, UnknownID, UnknownTerm,
       InvalidKey, InvalidValue);

AdminPermission   addAdminPermission(
                  in  string         pname,
                  in  RegTerm        operation,
                  in  ScopeSeq       admScope
)
raises(InternalError, DataBaseError, DuplicateID, UnknownTerm,
       InvalidKey, InvalidValue);

AdminPermission   updateAdminPermission(
                  in  string         pname,
                  in  RegTerm        operation,
                  in  ScopeSeq       admScope
)
raises(InternalError, DataBaseError, UnknownID, UnknownTerm,
       InvalidKey, InvalidValue);

void              removePermission(in  string pname)
raises(InternalError, DataBaseError, UnknownID, TermInUse);

boolean           termIsUsed(in VocDomain vdomain, in RegTerm vocItem);
raises(InternalError, DataBaseError, UnknownTerm);

long              getDataViews(out DataViewSeq dviews)
raises(InternalError, DataBaseError);

DataView          addDataView(
```

```
                    in  RegTerm        vname,
                    in  DVDef          def
    )
    raises(InternalError, DataBaseError, DuplicateID);
    DataView          updateDataView(
                    in  DataView       view
    )
    raises(InternalError, DataBaseError, UnknownID);

    void              removeDataView(in  RegTerm vname)
    raises(InternalError, DataBaseError, UnknownID, TermInUse);

    SyntaxState       dvSyntaxCheck(in  RegTerm vname, out string errorReport)
    raises (InternalError, DataBaseError, UnknownTerm, SyntaxError);

    void              registerDataView(in  RegTerm        vname)
    raises (InternalError, DataBaseError, SyntaxError);

    void              unregisterDataView(in  RegTerm        vname,
    )(InternalError, DataBaseError, UnknownTerm);

    // information retrieval functions

    void              getDVDefinition(
                    in  RegTerm        vname,
                    out DVDef          vdefinition)
    raises(InternalError, DataBaseError, UnknownTerm);
};
```

readonly attribute AdmAccessControl() enthält das AdmAccessControl-Objekt der aktuellen Administratorsitzung mit den zugehörigen Administratorrechten.

readonly attribute RoleHierarchy() enthält die Referenz des RoleHierarchy-Objekts (der aktuellen Sitzung), das die Erfassung, Darstellung und Manipulation der Rollen und Rollenhierarchie steuert.

readonly attribute SubjectDirectory() enthält die Referenz des SubjectDirectory-Objekts (der aktuellen Sitzung), das alle notwendigen Zugriffe auf das Benutzerverzeichnis kapselt.

readonly attribute Vocabulary() enthält die Referenz auf das Vocabulary-Objekt, das alle Bereiche des kontrollierten Vokabulars verwaltet.

readonly attribute ContextEvaluator() enthält die Referenz auf das ContextEvaluator-Objekt, das die Erfassung und Auswertung der Kontextregeln steuert.

getOperations() liefert die Operationsbezeichner, die bei der Definition der Administratorrechte berücksichtigt werden müssen.

getUserPermissions(), getAdminPermissions() ermitteln alle bereits definierten Benutzerrollen- bzw. Administratorrollen-Rechte. Der Rückgabewert der Funktionen ist die Anzahl der gefundenen Rechte. Die Liste

„perms“ enthält die Rechte in Form von User- bzw. AdminPermission-Objekten. Durch die Erweiterung des PolicyAdmin-Interfaces um entsprechende parametrisierte Funktionen können auch spezifische Teilmengen der Rechte erfragt werden.

addUserPermission() definiert ein neues Benutzerrecht mit dem Namen „pname“ und der Definition bestehend aus dem Kontext „trContext“ dem registrierten Operations-Bezeichner „operation“, dem registrierten Datensicht-Bezeichner „dview“ dem registrierten System-Bezeichner[3] „source“, und einer Liste spezifischer Bedingungen „conds“. Alle Parameter werden auf die Existenz der registrierten Bezeichner und der zugehörigen Definitionen überprüft.

updateUserPermission() aktualisiert das Benutzerrecht „pname“ mit dem Kontext „trContext“, der Operation „operation“, dem Datensicht-Bezeichner „dview“ dem System-Bezeichner „source“ und der Bedingungs-Liste „conds“.

addAdminPermission() definiert ein neues Administratorrecht mit dem Namen „pname“ und der Definition bestehend aus dem registrierten Operationsbezeichner „operation“ und einer Liste von Attribut/Wert-Paaren, die den Zuständigkeitsbereich bezüglich der Operation auf Rollen, Rechte und Subjekte definieren („admScope“). Zur Menge der gültigen Schlüsselbezeichner und Wertemengen siehe Abschnitt 4.2.1 Tabelle 4.1.

updateAdminPermission() aktualisiert das Administratorrecht „pname“ mit dem Operationsbezeichner „operation“ und den Zuständigkeitsattributen „admScope“.

removePermission() löscht das Recht mit dem eindeutigen Namen „pname“, wobei „pname“ ein Benutzer- oder ein Administratorrecht referenzieren kann.

termIsUsed() durchsucht alle Permission-Definitionen nach dem Ausdruck „vocItem“. Dabei gibt „vdomain“ den Vokabular-Bereich an. Für „vdomain == D_UNIT“ sind die Rollen- und Subjekt-Definition zu überprüfen. Dazu werden die Funktionen „RoleHierarchy::termIsUsed()“ und „SubjectDirectory::termIsUsed()“ aufgerufen. Die Funktion liefert den Rückgabewert „true“, falls der Ausdruck gefunden wurde, andernfalls „false“.

getDataViews() ermittelt alle existierenden Datensichten. Rückgabewert der Funktion ist die Anzahl der gefundenen Datensichten. Die Liste „dviews“ enthält die DataView-Objekte, die mit Hilfe eines „Datensicht-Editors“ visualisiert und bearbeitet werden können.

[3] „source“ kann auch den Wert „GLOBAL“ haben. Das bedeutet, dass das Recht in allen Fällen gilt, in denen kein System explizit gefordert ist.

addDataView() erzeugt ein neues DataView-Objekt mit der Definitionsstruktur „`vdef`“, und registriert es mit dem Namen „`vname`“ im DataView-Bereich des Vokabulars. „`vdef`“ enthält außer der text-basierten Definition das verwendete Vokabular für die Datenklassen und einen Formatbezeichner. Der Syntaxstatus wird auf „`NO_CHECK`“ gesetzt.

updateDataView() aktualisiert die Datensicht „`view.name`“ mit der Definition „`view.definition`“. Der Syntaxstatus wird auf „`NO_CHECK`“ gesetzt. Bereits referenzierte Datensichten können nicht geändert werden.

removeDataView() löscht das DataView-Objekt das unter dem Namen „`vname`“ im DataView-Bereich des Vokabulars registriert ist. Dabei wird auch der Begriff aus dem Vokabular entfernt. Bereits referenzierte Datensichten können nicht entfernt werden.

dvSyntaxCheck() ruft die Prüfroutine des DataView-Objekts „`vname`“ auf. Der Rückgabewert liefert den Syntaxstatus, „`errorReport`“ liefert die Ausgabe des internen Syntaxparsers.

registerDataView() registriert die Datensicht mit dem Namen „`vname`“. Voraussetzung ist, dass der Syntaxstatus gegebenenfalls nach durchgeführter Syntaxprüfung den Wert „`CORRECT`“ hat.

unregisterDataView() entfernt die Registrierung des Datensicht-Namen im Vokabular, sofern er noch nicht referenziert wurde.

getDVDefinition() ermittelt die Datensicht-Definition zum registrierten Namen „`vname`“. Nach erfolgreicher Ausführung der Funktion ist die Definitionsstruktur in „`vdefinition`“ enthalten.

RoleHierarchy-Interface

Das RoleHierarchy-Interface liefert alle Funktionen, die zur Erfassung und Verwaltung der Rollendefinitionen und der Beziehungen von Rollen zueinander sowie dem Retrieval von Rolleninformation benötigt werden.

```
// Datatypes

typedef enum InheritType {NOTHING, JUNIOR, SENIOR};
typedef struct RoleInfo {
               string         name;
               RoleType       type;
               RegTerm        unit;
};
typedef sequence<RoleInfo> RoleInfoSeq;

typedef struct Constraint{
               ConstraintType type,
               string         srname;  // constraint subject name
               string         orname;  // constraint object name
};
typedef struct ConstraintInfo{
```

```
              ConstraintType type,
              RoleInfo       srole;  // constraint subject info
              RoleInfo       orole;  // constraint object info
};

typedef sequence<Constraint> ConstraintSeq;
typedef sequence<ConstraintInfo> ConstraintInfo;

struct UserPermInfo{
       ContextStruct          trContext;
       RegTerm                viewName;
       RegTerm                viewDef;
       RegTerm                operation;
       RegTerm                source;
       AttributeSeq           conditions;
}
typedef sequence<UserPermInfo> UserPermInfoSeq;

interface RoleHierarchy {
    attribute AdmAccessControl admAC;
    attribute PolicyAdmin      policyControl;
    attribute SubjectDirectory subjectControl;

    long             getOperations(out TermSeq operations)
    raises(InternalError);

    long             getRoles(in  ScopeSeq scope, out RoleInfoSeq roles)
    raises(InternalError, DataBaseError);

    long             getRootRoles(in  ScopeSeq scope, out RoleInfoSeq roles)
    raises(InternalError, DataBaseError);

    long             getJuniorRoles(
                     in  string          rname,
                     in  ScopeSeq        scope,
                     out RoleInfoSeq     roles
    )
    raises(InternalError, DataBaseError, UnknownID);

    long             getSeniorRoles(
                     in  string          rname,
                     in  ScopeSeq        scope
                     out RoleInfoSeq     roles
    )
    raises(InternalError, DataBaseError, UnknownID);

    RoleInfo         newRole(
                     in  string          rname,
                     in  RoleType        rtype,
                     in  string          unit
    )
    raises(InternalError, DataBaseError, DuplicateID, InvalidParameter,
           PermissionDenied);

    RoleInfo         updateRole(
                     in  string          rname,
                     in  RoleType        rtype,
                     in  string          unit
```

```
)
raises(InternalError, DataBaseError, UnknownID, UnknownTerm,
       PermissionDenied);

void              removeRole(in  string rname)
raises(InternalError, DataBaseError, UnknownID, IDInUse,
       PermissionDenied);

long              getRolePermissions(
                  in  string           rname,
                  out PermSeq          permissions)
raises(InternalError, DataBaseError, UnknownID);

void              addRolePermission(
                  in  string           rname,
                  in  Permission       permission)
raises(InternalError, DataBaseError, UnknownTerm, RoleTypeConflict,
       PermissionDenied);

void              removeRolePermission(
                  in  string           rname,
                  in  Permission       perm)
raises(InternalError, DataBaseError, UnknownID, PermissionDenied);

void              insertInheritance(
                  in  string           srname,
                  in  string           jrname
)
raises(InternalError, DataBaseError, UnknownID, RoleRoleConflict,
       ConstraintViolation, PermissionDenied);

void              removeInheritance(
                  in  string           srname,
                  in  string           jrname
)
raises(InternalError, DataBaseError, UnknownID, IDInUse,
       PermissionDenied);

InheritType       roleRelation{
                  in  string           relSName, // relation subject
                  in  string           relOName  // relation object
}
raises(InternalError, DataBaseError, UnknownID);

void              addConstraint(in  Constraint constraint)
raises(InternalError, DataBaseError, UnknownID, RoleRoleConflict,
       ConstraintViolation, PermissionDenied);

void              removeConstraint(in  Constraint constraint)
raises(InternalError, DataBaseError, UnknownID, PermissionDenied);

long              getConstraints(
                  in  string           srname,
                  in  string           orname,
                  out ConstraintSeq    constraints
)
raises(InternalError, DataBaseError, UnknownID);
```

```
    long              getConstraintInfos(
                      in  string           srname,
                      in  string           orname,
                      out ConstraintInfoSeq contraints
    )
    raises(InternalError, DataBaseError, UnknownID);

    boolean           termIsUsed(in string vocItem);
    raises(InternalError, DataBaseError, UnknownTerm);

// information retrieval functions

    void              getRoleUnit(in  string rname, out RegTerm unit)
    raises(InternalError, DataBaseError, UnknownID);

    long              getUserPermissions(
                      in  string rname,
                      out UserPermInfoSeq permissions)
    raises(InternalError, DataBaseError, UnknownID, RoleTypeConflict);

    long              getAdminPermissions(
                      in  string rname,
                      out AdminPermSeq permissions)
    raises(InternalError, DataBaseError, UnknownID, RoleTypeConflict);
};
```

attribute AdmAccessControl() enthält das AdmAccessControl-Objekt der aktuellen Administratorsitzung mit den zugehörigen Administratorrechten.

attribute PolicyAdmin enthält die Objekt-Referenz des PolicyAdmin-Objekts (der aktuellen Sitzung), das für die Abfrage von Rechten benötigt wird.

attribute SubjectDirectory() enthält die Referenz des SubjectDirectory-Objekts (der aktuellen Sitzung), das bei Konsistenzprüfungen konsultiert werden muss.

getOperations() liefert die Operationsbezeichner, die bei der Definition der Administratorrechte berücksichtigt werden müssen.

getRoles() ermittelt alle Rollen-Definitionen und liefert als Rückgabewert, die Anzahl der gefundenen Objekte. Durch Einträge der übergebenen Attributliste „`scope`“ kann der Suchbereich der Rollen eingeschränkt werden. Dabei werden die Attributschlüssel folgendermaßen interpretiert:

1. „`scope[i].key == 'ROLE_UNIT'`“
   Wenn „`scope[i].value`“ einen gültigen Bezeichner einer Organisationseinheit entält, dann gilt für die Ergebnismenge:
   `roles[i].unit == scope[i].value` oder „`roles[i].unit`“ referenziert eine Organisationseinheit, die der in „`scope[i].value`“ untergeordnet ist.

2. „`scope[i].key == 'ROLE_ID'`“
   Wenn „`scope[i].value`“ einen gültigen Rollen-Namen enthält, dann gilt: Die Ergebnismenge enthält genau ein Element „`roles[0]`“ und `roles[0].name == scope[i].value`.
3. „`scope[i].key == 'JUNIORROLE_ID'`“
   Wenn „`scope[i].value`“ einen gültigen Rolle-Namen enthält, dann gilt für die Ergebnismenge:
   `roles[i].name == scope[i].value` oder „`scope[i].value`“ referenziert eine Junior-Rolle von „`roles[i]`“.

getRootRoles() ermittelt die Rollen-Objekte der obersten Hierarchie-Ebene im angegebenen Bereich. Die Funktionsweise ist analog zu „`getRoles()`“.

getSeniorRoles() ermittelt die Rollen-Objekte, die unmittelbar von der Rolle „`rname`“ abgeleitet sind und im Bereich „`scope`“ liegen. Die Funktionsweise ist analog zu „`getRoles()`“.

getJuniorRoles() ermittelt die Rollen-Objekte, von denen die Rolle „`rname`“ unmittelbar abgeleitet ist und die im Bereich „`scope`“ liegen. Die Funktionsweise ist analog zu „`getRoles()`“.

newRole() legt eine neue Rolle mit dem eindeutigen Namen „`rname`“ und den Attributen „`rtype`“ und „`unit`“ an, wenn nicht bereits eine Rolle mit dem gleichen Namen existiert. „`unit`“ ist ein hierarchischer Bezeichner, der die Hierarchie der Organisationseinheiten wiederspiegelt.

updateRole() aktualisiert die Rolle „`rname`“ mit den Attributen „`rtype`“ und „`unit`“, falls „`rname`“ eine existierende Rolle referenziert.

removeRole() entfernt die Rolle mit dem Namen „`role`“, sofern sie keinem aktuellen Benutzer zugewiesen ist, keine Abhängigkeiten anderer Rollen bestehen und der angemeldete Administrator die Berechtigung besitzt.

getRolePermissions() ermittelt die unmittelbaren (nicht die geerbten) Rollenrechte der Rolle „`rname`“. Nach erfolgreicher Ausführung enthält permissions die Rollen-Rechte in Form von Permission-Objekten. Mit einem „dynamischen Cast“ können die dahinter verborgenen UserPermission- oder AdminPermission-Objekte angesprochen werden.

addRolePermission() weist der Rolle „`rname`“ das durch „`perm`“ repräsentierte Recht zu. Dabei wird die Konsistenz bezüglich Rollentyp „`type`“ und Permission-Objekt überprüft.

removeRolePermission() entfernt das durch „`perm`“ repräsentierte Recht aus der Menge der Rollenrechte der Rolle „`rname`“.

insertInheritance() definiert eine Vererbungsrelation zwischen der Junior-Rolle mit der ID „`jrname`“ und der Senior-Rolle mit der ID „`srname`“. Voraussetzung für die Funktionsausführung sind gültige Rollen-IDs; die

Organisationseinheiten der beiden Rollen müssen gleich sein, oder die der Senior-Rolle muss der der Junior-Rolle untergeordnet sein. Die Senior-Rolle erbt dabei logisch (nicht physisch) die Rechte der Junior-Rolle. Deshalb darf die Senior-Rolle noch keinem Subjekt zugewiesen sein.

removeInheritance() entfernt eine Vererbungsrelation zwischen der Junior-Rolle mit der ID „`jrname`“ und der Senior-Rolle mit der ID „`srname`“. Diese Funktion ist nur erfolgreich, wenn keine weiteren Rollen aus der Senior-Rolle abgeleitet sind, keine Subjekt-Rollen-Relation existiert, und die Senior-Rolle auch durch kein Rollen-Constraint referenziert wird.

roleRelation() liefert die Relation ($\in$ { `NOTHING, JUNIOR, SENIOR`) zu einem Rollenpaar. Das Ergebnis muss in der Form „`relSName` ist `RelationType` von `relOName`“ interpretiert werden.

addConstraint() definiert eine Constraint-Relation zwischen den Rollen „`constraint.srole`“ und „`constraint.orole`“. „`constraint.type`“ definiert die Art des Constraints. Vorbedingungen der Funktion sind, dass beide Rollen-Objekte existieren und dass durch bereits existierende Subjekt-Rollen-Relationen keine Constraint-Verletzung entsteht. Der Constraint-Typ „`STRONG_EXCLUSION`“ bedeutet, dass ein Subjekt nicht gleichzeitig Inhaber beider Rollen sein kann. „`WEAK_EXCLUSION`“ erlaubt die Zuweisung beider Rollen; sie dürfen aber nicht gleichzeitig von einem Subjekt aktiviert werden. „`PREREQUISITE_ROLE`“ verlangt, dass ein Subjekt als Voraussetzung für die Zuweisung der Rolle „`orole`“ bereits Inhaber der Rolle „`srole`“ ist.

removeConstraint() löscht die Constraint-Relation zwischen den Rollen „`constraint.srole`“ und „`constraint.orole`“.

getConstraints(), getConstraintInfos() liefern je eine Liste der Constraints, die die Eingabeparameter erfüllen. Die Liste enthält je nach Funktion nur den Rollen-Name oder zusätzliche Attribute der betroffenen Rollen. Als Wildcard-Symbol für den Rollennamen kann das „*“-Zeichen gesetzt werden.

termIsUsed() vergleicht das „`unit`“-Attribut aller Rollen-Definitionen mit „`vocItem`“. Die Funktion liefert den Rückgabewert „`true`“, falls eine Rolle „`r`“ mit „`r.unit == vocItem`“ gefunden wurde, andernfalls „`false`“.

getRoleUnit() ist eine Funktion, die für das Informations-Retrieval die Organisationseinheit der Rolle mit dem Rollen-Namen „`rname`“ ermittelt und zurückliefert. Vorbedingung ist, dass die Rolle mit „`rname`“ existiert. Nach erfolgreicher Ausführung der Funktion enthält „`unit`“ den Bezeichner der Organisationseinheit.

getUserPermissions(), getAdminPermissions() ermitteln alle logischen Rechte der Benutzer- bzw. Administrator-Rolle mit dem Namen „`rname`“, d. h. auch die aus der Rollen-Hierarchie geerbten Rechte. Vorbedingung

ist ein gültiger Rollenname und der passende Rollentyp. Nach erfolgreicher Operation enthält „`permissions`“ die Rechte. Die UserPermInfo-Struktur unterscheidet sich vom UserPermission-Objekt dadurch, dass sie statt des Datensicht-Namens das vollständige DataView-Objekt enthält. Zu diesem Zweck muss für jedes UserPermission-Objekt die DataView-Definitionsstruktur ermittelt werden (`PolicyAdmin::getDVDefinition()`). Der Rückgabewert der Funktion ist die Anzahl der gefundenen Rechte.

Authorization-Interface

Mit den Funktionen des Authorization-Interface wird die Erfassung und Manipulation der Rollen-Subjekt-Zuweisungen gesteuert.

```
// Datatypes

typedef string < 14 > DateTime;
struct Validity{
       DateTime        begin;
       DateTime        end;
};
struct RoleEntry{
       string          rname;
       Validity        valid;
};
typedef sequence<RoleEntry> RoleEntrySeq;
typedef enum SubjType {PERSON, SYSTEM};
struct Subject{
       string          subjID;
       SubjType        type;
       string          realName;
       RegTerm         unit;
       string          qual;
       RoleEntrySeq    roles;
};
typedef sequence<Subject> SubjectSeq;

interface Authorization(
    attribute AdmAccessControl admAC;
    attribute RoleHierarchy    roleControl;
    attribute SubjectDirectory subjectControl;

    long               getOperations(out TermSeq operations);

    long               getSubjects(
                       in  ScopeSec       scope,
                       out SubjectSeq     subjects
    )
    raises(InternalError, ConnectionError, LDAPError);

    long               getRoles(in  ScopeSec scope, out RoleInfoSeq roles)
    raises(InternalError, DataBaseError);

    void               assignRole(
                       in  Subject        subject,
                       in  RoleEntry      rEntry
```

```
    )
    raises (InternalError, ConnectionError, LDAPError, UnknownID,
            ConstraintViolation, PermissionDenied);

    void               removeAssignment(
                       in  Subject        subject,
                       in  RoleEntry      rEntry
    )
    raises (InternalError, ConnectionError, LDAPError,
            ConstraintViolation, PermissionDenied);
);
```

attribute AdmAccessControl() enthält das AdmAccessControl-Objekt der aktuellen Administratorsitzung mit den zugehörigen Administratorrechten.

attribute RoleHierarchy() enthält die Referenz des RoleHierarchy-Objekts (der aktuellen Sitzung), das die Rolleninformation liefert.

attribute SubjectDirectory() enthält die Referenz des SubjectDirectory-Objekts (der aktuellen Sitzung), das alle notwendigen Zugriffe auf das Benutzerverzeichnis kapselt.

getOperations() liefert die Operationsbezeichner, die bei der Definition der Administratorrechte berücksichtigt werden müssen.

getSubjects() ermittelt alle Benutzer, denen der aktuelle Administrator eine Rolle oder ein Subjektrecht zuweisen oder entziehen darf. Nach erfolgreicher Ausführung enthält `subjects` die ermittelten Subjekt-Objekte. Der Rückgabewert enthält die Anzahl der Einträge.

getRoles() ermittelt alle Rollen, die der aktuelle Administrator zuweisen darf. Nach erfolgreicher Ausführung enthält `roles` die Rollen als Strukturen vom Typ „`RoleInfo`“. Der Rückgabewert enthält die Anzahl der Einträge.

assignRole() weist dem Subjekt `subject` die Rolle `rEntry.name` zu. Mit Hilfe des RoleHierarchy-Objekts wird überprüft, ob `rEntry.name` ein gültiger Rollenbezeichner ist, und ob ein widersprüchliches Rollenconstraint (STRONG_EXCLUSION-Constraint) existiert. `rEntry.valid` definiert den Gültigkeitszeitraum der Zuweisung. Existiert bereits eine Zuweisung der selben Rolle, so wird der Gültigkeitszeitraum der bereits zugewiesenen Rolle bei Überschneidung der beiden Zeiträume aktualisiert.

removeAssignment() nimmt die Subjekt-Rollen-Zuweisung zwischen `subject` und `rEntry.name` für den Zeitraum `rEntry.valid` zurück. Auch hier wird zuvor die potentielle Verletzung eines Rollenconstraints (PREREQUISITE_ROLE) ausgeschlossen.

Bei allen Funktionen der Subjekt-Rollen-Zuweisung wird das SubjectDirectory-Objekt konsultiert, das die Kopplung mit dem Benutzerverzeichnis realisiert.

SubjectDirectory-Interface

Das SubjectDirectory-Interface bietet eine Schnittstelle zum klinikweiten Benutzerverzeichnis. Es beinhaltet Funktionen zur Ermittlung von Benutzerdaten und zur Manipulation der Rolleninformationen im Benutzerverzeichnis. Die grundsätzliche Erfassung und Manipulation der Benutzerdaten ist nicht Inhalt dieser Schnittstelle und dieser Arbeit. Das Benutzerverzeichnis besitzt bereits Schnittstellen für die Erfassung der Benutzer und die Verwaltung weiterer Benutzerdaten, z. B. der Telefon- und Faxnummern, sowie der Mailadressen. In den Funktionen des SubjectDirectory-Interfaces werden keine Konsistenzprüfungen in Bezug auf die Gültigkeit von Rollenbezeichnern und Rollenconstraints durchgeführt. Die Konsistenz muss durch das übergeordnete Objekt (Authorization-Objekt) sichergestellt werden.

```
interface SubjectDirectory(

    long               getSubjects(
                       in  SubjType       type,
                       in  RegTerm        unit,
                       in  string         qual,
                       in  string         roleName,
                       out SubjectSeq     subjects
    )
    raises(InternalError, ConnectionError, LDAPError, UnknownID);

    long               getAssignedRoles(
                       in  string subjID,
                       out RoleEntrySeq entries
    )
    raises(InternalError, ConnectionError, LDAPError, UnknownID);

    void               assignRole(
                       in  string         subjID,
                       in  RoleEntry      rEntry
    )
    raises(InternalError, ConnectionError, LDAPError,
           ConstraintViolation, UnknownID);
    void               removeRole(
                       in  string         subjID,
                       in  RoleEntry      rEntry
    )
    raises(InternalError, ConnectionError, LDAPError,
           ConstraintViolation, UnknownID);
);
```

getSubjects() ermittelt die Daten aller Subjekte aus dem Benutzer-Verzeichnis. Dabei kann die gewünschte Ergebnismenge durch die Angabe des Typs (`type`), der Organisationseinheit (`unit`), der Qualifikation (`qual`) und eines Rollenbezeichners (`roleName`) eingegrenzt werden. Nach erfolgreicher Ausführung enthält „`subjects`“ die Informationen zu den Subjekten; der Rückgabewert der Funktion liefert die Anzahl der Einträge.

getAssignedRoles() ermittelt alle zugewiesenen Rollen eines Subjekts oder aller Subjekte, falls `subjID='*'`. Nach erfolgreicher Ausführung enthält

„`entries`“ die Einträge aus Rollen-Name und Gültigkeitsintervall; der Rückgabewert der Funktion liefert die Anzahl der Einträge.

assignRole() trägt für das Subjekt „`subjID`“ die Rolle „`rEntry.name`“ mit dem Gültigkeitszeitraum „`rEntry.valid` “ im Benutzer-Verzeichnis ein.

removeRole() entfernt die Rollen-Zuweisungen „`rEntry`“ des Subjekts „`subjID`“ aus dem Benutzer-Verzeichnis.

Anmerkungen zur Gestaltung der Benutzeroberfläche

Für die Gestaltung der Benutzeroberflächen gelten die üblichen Anforderungen hinsichtlich der Ergonomie und Ästhetik. Die folgenden Anmerkungen sind Empfehlungen zu bereichsspezifischen Aspekten:

Zugriffssteuerung Die Berechtigungen eines Policy-Administrators sollten bereits in der Oberfläche berücksichtigt werden, so dass Menüs zur Funktionsauswahl nur die Funktionen anzeigen, die auch ausgeführt werden dürfen, oder zumindest alle anderen nur abgeschwächt anzeigen. Bei jeder Benutzeraktion (z. B. Auswahl einer Rolle oder Rollen-Teilhierarchie) sollten Menüs entsprechend angepaßt werden.

Auswahllisten Kontextprototypen, Operationen und Datensichten sowie Begriffe und Strukturen für die Datensichtdefinition sollen als Auswahllisten bzw. -bäume zur Verfügung gestellt werden. Die Wahl von Parameterschlüsseln (z. B. Kontextparameter) und Wertemengen (z. B. Organisationseinheiten) ist ebenfalls durch Auswahlhilfen zu unterstützen.

Anzeige von Entitäten Das Auffinden von Entitäten (z. B. Rechte, Datensichten, Subjekte und Rollen) in den Auswahllisten muss durch die übersichtliche Anzeige wesentlicher Merkmale unterstützt werden. Für die Rechte sind das z. B. der Name des Rechts, der Kontextname, der Operationsbezeichner und der Datensichtname; die Auswahlliste für Datensichten kann den Namen und eine Kurzbeschreibung enthalten. Für die jeweils ausgewählte Listenzeile kann die Datensicht-Definition in einem Vorschaufenster präsentiert werden. Vorteilhaft sind Filter- und Sortierkriterien beim Aufbau von Listen.

Syntaxunterstützung Vor allem bei der Definition der Datensichten sollte der Editor die Umsetzung der ausgewählten Vokabularstrukturen in das entsprechende Datensicht-Repräsentationsformat unterstützen.

Grafische Erfassung von Relationen Die Rollenhierarchie – eventuell auch die Constraint-Relationen – sollten grafisch dargestellt und manipuliert werden können (Drag&Drop).

Pflege der vorhandenen Rollen und Zuweisungen Für die Definition von aktuellen erweiterten Rollen sollen vorhandene Rollen kopiert und bearbeitet werden können. Der Autorisierungs-Editor sollte außerdem

den Austausch der zugewiesenen Rolle bei allen Subjekten durch die aktuelle Rollen-Version unterstützten.

4.4.4 Kontextregeln

Das KontextEvaluator-Interface liefert die Funktionen zur Definition und Manipulation der Kontextregeln, sowie zur Auswertung der Kontexte. Die Daten der Kontextregeln werden in der Kontext-Server-Datenbank gespeichert. Beim Systemstart (und jeder Aktualisierung) werden ContextRule-Objekte erzeugt und mit den Regeldaten initialisiert. In den ContextRule-Objekten liegen die Regeln dann als dynamisch erzeugte parametrisierte SQL-Statements vor, womit eine effiziente Auswertung der Kontexte erreicht werden kann. Die Erfassung und Manipulation von Regeln kann nie im Laufzeitbetrieb durchgeführt werden. Regeln müssen zunächst auf einem Testsystem definiert und überprüft werden. Während der Aktualisierung des Produktionssystems muss der Betrieb durch ein Ersatz-System aufrechterhalten werden (→ Abschnitt 5.1.5).

Bei der Auswertung werden aus den Behandlungselement-Instanzen die Bezugs-Phasen abgeleitet, die mit Teilen der Krankenakten assoziiert sind. Die Rückgabe-Strukturen enthalten die Patienten-ID, die Aufnahmenummer des Verwaltungsfalls, Beginn und Ende[4] der Phase, sowie weitere optionale Attribute.

ContextEvaluator-Interface

```
// Datatypes

typedef enum TEKey {PID, MED_ID, ADM_ID, LOCATION, ETYPE, TTYPE, PART_UNIT,
                    PART_ROLE, BEGIN, END};

typedef sequence<TEKey> TEKeySeq;

typedef struct ContextPrototype{
               string       name;
               TEKeySeq     parmtypes;
};
typedef sequence<ContextPrototype> ContProtSeq;
typedef enum ActiveState {ACTIVE, INACTIVE};

typedef enum OpKey {EQUAL, NOT_EQUAL, LESS, LESS_OR_EQUAL, GREATER,
                    GREATER_OR_EQUAL, IN, NOT_IN};
typedef enum FunctionKey {SINGLE_VALUE, VALUE_SET, ADD_DATE, SUBSTRACT_DATE};
typedef enum ArgumentKey {DATE, DAY, MONTH, YEAR, INT_CONST, STRING_CONST,
                          EXIST_ATTRIBUTE};

struct Argument {
               ArgumentKey  key;
               string       value;
};
typedef squence<Argument> ArgumentSeq;
```

[4]Das Ende wird nur für abgeschlossene Behandlungsphasen angegeben.

```
typedef struct CmpTerm {
               FunctionKey  function;
               ArgumentSeq  arguments;
};

typedef struct Condition{
               TEKey        tekey;
               OpKey        operator;
               CmpTerm      cTerm;
};

typedef sequence<Condition> ConditionsSeq;

typedef enum RuleType {FORALL, FORALL_EXIST};

struct PatRecord {
       string          patID;
       AttributeSeq    tparms;
};

typedef sequence<PatRecord>PatRecordSeq;


interface ContextEvaluator{
    attribute AdmAccessControl admAC;
    attribute Vocabulary       vocabularyControl;

    long               getOperations(out TermSeq operations);
    long               getContextPrototypes(out ContProtSeq prototypes);

    ContextRule        getRuleDefinition(in string rname)
    raises(InternalError, DataBaseError, UnknownID);

    ContextRule        setRuleDefinition(
                       in  string         rname,
                       in  RuleType       rtype,
                       in  TEKeySeq       iparms,
                       in  ConditionsSeq  existCond,
                       in  ConditionsSeq  forAllCond,
    )
    raises(InternalError, DataBaseError, InvalidParameter);

    ContextRule        updateRuleDefinition(
                       in  string         rname,
                       in  RuleType       rtype,
                       in  TEKeySeq       iparms,
                       in  ConditionsSeq  existCond,
                       in  ConditionsSeq  forAllCond
    )
    raises(InternalError, DataBaseError, UnknownID);

    ContextRule        activateRuleDefinition(in  string rname, out string errorReport)
    raises(InternalError, DataBaseError, UnknownID, SyntaxError);

    ContextRule        deactivateRuleDefinition(in  string rname)
    raises(InternalError, DataBaseError, UnknownID);
```

```
    void              removeRuleDefinition(in  string rname);
    raises(InternalError, DataBaseError, UnknownID);

    void              updateRuntimeObjects()
    raises(InternalError, DataBaseError, SyntaxError);

    long              getRecords(
                      in  ContextStruct  trContext,
                      in  AttributeSeq   sparms,
                      out PatRecordSeq   records
    )
    raises(InternalError, DataBaseError, UnknownID, InvalidKey, InvalidValue,
           TooManyRecords);
);
```

attribute AdmAccessControl() enthält das AdmAccessControl-Objekt der aktuellen Administratorsitzung mit den zugehörigen Administratorrechten.

attribute Vocabulary() enthält die Referenz auf das Vocabulary-Objekt zur Aktualisierung der Kontext-Prototyp-Einträge.

getOperations() liefert die Operationsbezeichner, die bei der Definition der Administratorrechte berücksichtigt werden müssen.

getContextPrototypes() liefert die Liste der Kontext-Prototypen zu allen existierenden Regeln. Diese Liste wird bei der Kontextdefinition benötigt um Regeln zur Bearbeitung auszuwählen.

getRuleDefinition() liefert das Regelobjekt zur bezeichneten Regel (`rname`).

setRuleDefinition() erzeugt und speichert eine Kontext-Regel mit dem Namen „`rname`“. „`rtype`“ ($\in$ {`FORALL, FORALL_ EXIST`}) definiert den Regel-Typ, „`iparms`“ die Regel-Parameter, „`existCond`“ die Bedingungen für das Indikatorelement, und „`forAllCond`“ die Bedingungen für die Behandlungselemente der Regelmenge. Bei Regeln des „FORALL“ - Typs ist die `existCond`-Liste leer. Die Parametererfassung muss durch ein geeignetes Benutzerinterface unterstützt werden, das Auswahllisten für Attribut-Bezeichner, Operatoren, Schlüsselworter und Funktionen zur Verfügung stellt. Der Aktivierungsstatus der Regel erhält automatisch den Wert „`INACTIVE`“, der Syntaxstatus den Wert „`NO_CHECK`“.

updateRuleDefinition() aktualisiert die Regel „`rname`“ mit den Parametern „`rtype` “, „`iparms`“, „`existCond`“ und „`forAllCond`“. Der Aktivierungsstatus der Regel wird automatisch auf „`INACTIVE`“ gesetzt, der Syntaxstatuts auf „`NO_CHECK`“.

activateRule() überprüft die Regelsyntax und setzt den Aktivierungsstatus der Regel auf „`ACTIVE`“, falls die Syntaxprüfung erfolgreich war, d. h. wenn der Syntaxstatus anschließend den Wert „`CORRECT`“ hat. Rückgabewert ist der aktuelle Syntaxstatus der Regel. „`errorReport`“ enthält die vollständige text-basierte Ausgabe der Syntaxprüfung.

deactivateRule() Setzt den Aktivierungsstatus der Regel auf „`INACTIVE`“.

removeRuleDefinition() löscht die bezeichnete Regel, wobei sichergestellt wird, dass keine aus der Regeldefinition abgeleiteten Kontexte in aktuellen Benutzerrechten referenziert sind.

updateRunTimeObjects() aktualisiert die ContextRule-Objekte mit den in der Datenbank gespeicherten aktiven, syntaktisch korrekten Regeln. Aktualisiert dabei auch die Einträge des Vokabulars. Danach können die neuen oder veränderten Regeln bei der Definition von Rechten verwendet werden.

getRecords() ist Teil der Informationsretrieval-Funktionalität. Die Funktion sucht das zugehörige ContextRule-Objekt und ruft dessen Retrieval-Funktion mit den Context-Parametern, den Suchparametern und der leeren Record-Liste auf (`ContextRule::getRecords(trContext.parms, sparms, records)`). Wurde die Funktion erfolgreich durchgeführt, so enthält `records` die ermittelten Behandlungsphasen. Die Funktion liefert die Anzahl der Listeneinträge zurück.

ContextRule-Interface

Das ContextRule-Objekt kapselt die Funktionalität zur Überprüfung und Auswertung einer spezifischen Kontextregel. Es wird mit dem eindeutigen Regel- bzw. Kontextnamen identifiziert.

```
interface ContextRule{

    readonly attribute string      name;
    readonly attribute SyntaxState syntax;
    readonly attribute ActiveState active;
    readonly attribute TEKeySeq    inputParms;
    readonly attribute CondSeq     existConds;
    readonly attribute CondSeq     forAllConds;

    void               initStatements(in string ruleName)
    raises(InternalError, DataBaseError, SyntaxError);

    SyntaxState        syntaxCheck(out string errorReport)
    raises(InternalError, DataBaseError);

    SyntaxState        getSyntaxState()
    raises(InternalError, DataBaseError);

    long               getRecords(
                       in  AttributeSeq   cparms,
                       in  AttributeSeq   sparms,
                       out PatRecordSeq   records
    )
    raises(InternalError, DataBaseError, InvalidKey, InvalidValue,
           TooManyRecords);
);
```

initStatements() erzeugt aus der Regeldefinition dynamische, parametrisierte SQL-Statements, die in den Attributen des ContextRule-Objekts abgelegt werden.

syntaxCheck() führt eine Überprüfung der Regelsyntax durch und liefert den aktuellen Syntaxstatus zurück. „`errorReport`“ enthält die vollständige text-basierte Ausgabe des Regelparsers.

getSyntaxState() liefert den aktuellen Syntaxstatus der Regel.

getRecords() Zu den Kontextparametern „`cparms` “ und den Suchparametern „`sparms`“ ermittelt das ContextRule-Objekt die Behandlungsphasen, die die Teile der Krankenakten für den Zugriff bestimmen. Voraussetzung der erfolgreichen Funktionsausführung sind gültige Kontext- und Suchraumparameter. Nach erfolgreicher Durchführung enthält `records` die Phasendefinitionen in Form von `PatRecord`-Strukturen, die folgendermaßen zu interpretieren sind: „`pID`“ ist die eindeutige und klinikumsweit gültige Patienten-ID. „`tparms`“ ist eine Liste von Attribut/Wert-Paaren. Sie besitzt immer einen Eintrag „`tparms[i].key == 'ADMISSION_ID'`“; „`tparms[i].value`“ enthält dann die eindeutige Aufnahmenummer des Verwaltungsfalls. `tparms` besitzt außerdem je einen Eintrag „`tparms[i].key == 'START'`“ und „`tparms[i].key == 'END'`“, falls die Phase bereits abgeschlossen ist. Die zugehörigen Werte definieren das Zeitintervall, auf das sich die Krankenakte bezieht. Es ist unabhängig vom Zugriffszeitpunkt. Ein weiteres optionales Attribut ist `'LOCATION'`, dessen Wert die Krankenakte auf den Teil einschränkt, für den die referenzierte Organisationseinheit verantwortlich ist. Sofern die Phase selbst mit einer eindeutigen, klinikweit verbindlichen ID assoziiert ist, wird diese mit dem Attributschlüssel `'TREATMENT_ID'` mitgeliefert. Der Rückgabewert der Funktion liefert die Anzahl der Einträge. Die Wirkung der bei der Anfrage übergebenen Suchraum-Parameter ist folgendermaßen definiert:

1. `sparms.key == 'PATIENT_ID'`
   Enthält `sparms.value` eine gültige Patientennummer, so gilt für alle `records[i]`: `records[i].pID == sparms.value`.
2. `sparms.key == 'ADMISSION_ID'`
   Enthält `sparms.value` eine gültige Aufnahmenummer, so gilt für alle `records[i]`:
   Es gibt einen Eintrag `records[i].tparms[j].key == 'ADMISSION_ID'` mit `records[i].tparms[j].value == sparms.value`.
3. `sparms.key == UNIT`
   Enthält `sparms.value` einen gültigen Organisationseinheits-Bezeichner, so gilt für alle `records[i]`:
   Es gibt einen Eintrag mit `records[i].tparms[j].key == 'UNIT'` und `records[i].tparms[j].value == sparms.value` oder `records[i].tparms[j].value` referenziert eine untergeordnete Einheit von `sparms.value`.

4. `sparms.key == 'START'`
   Enthält `sparms.value` ein gültiges Datum, so gilt für alle `records[i]`:
   Es gibt einen Eintrag mit `records[i].tparms[j].key == 'END'` und `records[i].tparms[j].value > sparms.value` oder es gibt garkeinen Eintrag `records[i].tparms[j].key == 'END'`.
5. `sparms.key == 'END'`
   Enthält `sparms.value` ein gültiges Datum, so gilt für alle `records[i]`:
   Es gibt einen einen Eintrag mit `records[i].tparms[j].key == 'START'` und `records[i].tparms[j].value < sparms.value`.
6. `sparms.key == 'TREATMENT_TYPE'`
   Enthält `sparms.value` einen gültigen Behandlungstyp, so gilt für alle `records[i]`:
   Es gibt einen Eintrag mit `records[i].tparms[j].key == 'TTYPE'` und `records[i].tparms[j].value == sparms.value`.
7. `sparms.key == 'CURRENT'`
   Ist `sparms.value == 1` , so enthält `records` nur Einträge, die aktuelle „nicht-abgeschlossene“ Behandlungsphasen referenzieren. Für `sparms.value == 0` werden nur abgeschlossene Phasen berücksichtigt.

Gestaltung des Regeleditors

Der Regeleditor soll durch geignete Bedienungselemente und entsprechende Benutzerführung die Generierung von beliebig vielen Bedingungen in der `forAllCond`-Liste und je nach Regeltyp in der `existCond`-Liste ermöglichen. Die Definition einer Bedingung ist durch entsprechende Auswahllisten von Attributschlüsseln (`TEKey`), Operatoren (`OpKey`), Funktionen und Schlüsselwörtern zu unterstützen. Parametertypen sollen ebenfalls auswählbar sein; Parameternamen werden dabei dynamisch erzeugt ($1, .. $n). Variablennamen und Schlüsselwörter sind vorgegeben (z. B. `E_ADM_ID` für die Verwaltungsnummer des bedingten Elements, `TODAY` für das aktuelle Tagesdatum und `ADD_DATE(date, days)` für die Addition von Tagen zu einem Datumswert).

4.4.5 Das Vokabular

Das Vocabulary-Interface verwaltet das kontrollierte Vokabular, das für die Definition der Rechte benötigt wird. Es besteht aus unabhängigen Bereichen für Operationsbezeichner, Datenklassen, Bezeichner für definierte Datensichten, eindeutige Systemnamen und Bezeichner für Organisationseinheiten sowie einen Bereich für die Kontextprototypen. Jeder Bereich kann außerdem verschiedene unabhängige Vokabulare enthalten. Gültige Bereichsnamen sind: OP, DATA, VIEW, CONTEXT, UNIT und SYSTEM. Für die Bereiche UNIT und SYSTEM ist zu einem Zeitpunkt immer nur jeweils ein Vokabular gültig und verbindlich, das dann auch ohne Angabe des Vokabular-Bezeichners referenziert werden kann. Die Vokabulare, die Begriffe, keine Prototypen, enthalten, sind hierarchisch strukturiert. Jeder Eintrag innerhalb des hierarchischen Vokabulars ist durch seinen Namen und den Pfad in der Hierarchie eindeutig bestimmt. Die Hierarchiebenen werden im Pfadnamen durch Punkte

getrennt (z. B. *('OP', 'UKMZIntern', 'Lesen.Befundabfrage')* oder *('DATA', 'HL7', 'PID.PatientAddress'*). Zu jedem Begriff wird außerdem ein beschreibender Text abgelegt. Für die XML-Repräsentation von Datensichten können Knotenbezeichner für jede Hierarchiebene des jeweiligen Vokabulars erfasst werden.

Die Vokabular-Einträge der Prototypen besitzen zusätzlich eine Liste der zugehörigen Parameter. Diese Vokabulare sind nicht hierarchisch strukturiert.

```
typedef enum DomainType {OP, DATA, VIEW, CONTEXT, UNIT, SYSTEM};
typedef sequence<string>  NameSeq;
typedef TermSeq OPSeq;

interface Vocabulary{
    attribute PolicyAdmin      policyAdm;
    attribute AdmAcessControl  admAC;

    long               getOperations(out TermSeq operations);

 // Registration of terms

    void               regTerm(
                       in  string         name,
                       in  string         path,
                       in  string         vocabulary,
                       in  DomainType     domain,
                       in  string         descript
    )
    raises(InternalError, DataBaseError, Invalid Parameter, DuplicateTerm);

    void               unregTerm(in  RegTerm term)
    raises(InternalError, DataBaseError, Invalid Parameter, TermInUse);

    void               moveTerm(
                       in  string         name,
                       in  string         opath,
                       in  string         npath,
                       in  string         vocabulary,
                       in  DomainType     domain
    )
    raises(InternalError, DataBaseError, InvalidParameter, DuplicateTerm,
           TermInUse);

 // Registration of vocabulary specific information

    void                addLevelName(
                        in  string         vocabulary,
                        in  DomainType     domain,
                        in  long           level,
                        in  string         name
    )
    raises(InternalError, DataBaseError, UnknownTerm);

    string              getLevelName(
                        in  string         vocabulary,
                        in  DomainType     domain,
```

```
                        in  long           level
    )
    raises(InternalError, DataBaseError, UnknownTerm);

 // Registration of context prototypes

    void               regContextProt(
                       in  RegTerm        name,
                       in  TEKeySeq       parms,
                       in  string         descript
    )
    raises(InternalError, DataBaseError, InvalidParameter, DuplicateTerm);

    void               unregContextProt(in RegTerm name)
    raises(InternalError, DataBaseError, InvalidParameter, UnknownTerm,
           ContextInUse);

 // Retrieval of structures

    long               getVocabularies(
                       in  DomainType     domain,
                       out NameSeq        vocabularies
    )
    raises(InternalError, DataBaseError, InvalidParameter);

    long               getTerms(
                       in  string         vocabulary,
                       in  DomainType     domain,
                       in  string         searchpath,
                       out NameSeq        terms
    )
    raises(InternalError, DataBaseError, InvalidParameter, UnknownTerm);

    long               getPrototypes(
                       in  string        vocabulary,
                       in  DomainType     domain,
                       out ContProtSeq   prototypes
    )
    raises(InternalError, DataBaseError);
};
```

attribute PolicyAdmin enthält die Objekt-Referenz des PolicyAdmin-Objekts.

attribute AdmAccessControl enthält die Objekt-Referenz des AdmAccessControl-Objekts.

getOperations() liefert die Operationsbezeichner, die bei der Definition der Administratorrechte berücksichtigt werden müssen.

regTerm() legt im Vokabular „`vocabulary`“ des Bereichs „`domain`“ einen neuen Begriffsknoten mit dem lokalen Bezeichner „`name`“ unterhalb des Knotens „`path`“ und mit der Beschreibung „`descript`“ an.

unregTerm() entfernt den Knoten „`path.name`“ aus dem Vokabular „`vocabulary`“ des Bereichs „`domain`“, sofern er nicht in einer Definition referen-

ziert wird.

moveTerm() verschiebt den Begriffsknoten „`name`“ im Vokabular „`vocabulary`“ des Bereichs „`domain`“ von „`opath`“ nach „`npath`“, vorausgesetzt der Begriff wurde bislang in keiner Definition verwendet.

addLevelName() Für jedes spezifische Vokabular können Hierarchie-Ebenen mit Namen versehen werden. Die Funktion setzt den Namen „`name`“ als Name der Hierarchieebene „`level`“ im Vokabular „`vocabulary`“ des Bereichs „`domain`“, wobei die Ebenen mit „1“ beginnend durchgezählt werden. Der Default-Name jeder Ebene ist „DataClass“. Die Ebenen-Namen können als tag-Namen in den XML-Repräsentationen verwendet werden.

getLevelName() liefert den Namen der Hierachie-Ebene „`level`“ des Vokabulars „`vocabulary`“ im Bereich „`domain`“.

regContextProt() registriert einen Kontext-Prototyp namens „`name`“, den Kontext-Parametern „`parms`“ und der Beschreibung „`descript`“ als Begriff des Vokabulars „`vocabulary`“ im Bereich CONTEXT.

unregContextProt() entfernt den Prototypen „`name`“ aus dem Vokabular „`vocabulary`“ des Bereichs CONTEXT, falls er nicht aktuell referenziert ist.

getVocabularies() liefert die Namen aller Vokabulare des Bereichs „`domain`“. „`vocabularies`“ entält die Liste der Namen. Der Rückgabewert liefert die Anzahl der Einträge.

getTerms() ermittelt alle Begriffsknoten einer Hierarchieebene unterhalb des Knotens „`search_path`“, vorausgesetzt „`search_path`“ referenziert einen existierenden Knoten des Vokabulars „`vocabulary`“ im Bereich „`domain`“. Nach erfolgreicher Ausführung enthält „`terms`“ die Knotennamen. Der Rückgabewert der Funktion liefert die Anzahl.

getPrototypes() ermittelt alle Prototypen des Vokabulars „`vocabulary`“ im Bereich „`domain`“. Nach erfolgreicher Ausführung enthält „`prototypes`“ die Kontext-Protoypen in Form von Context-Prototype-Strukturen (siehe Abschnitt 4.4.4), der Rückgabewert der Funktion liefert die Anzahl der ermittelten Einträge.

Die Erfassung, Verwaltung und Verwendung der Vokabulare soll durch eine Benutzeroberfläche wesentlich erleichtert werden, die die Hierarchie grafisch wiedergibt und Funktionen zum Markieren und Verschieben von Knoten und Teilhierarchien anbietet.

4.4.6 Subjektrechte

Subjektrechte werden über das SubjectPermissions-Interface zur Verfügung gestellt und verwaltet. Die Funktionen greifen dabei auf die Datenquelle zu, in der die Subjektrechte abgelegt sind.

Für die Erfassung der Subjektrechte wird ein Client-Interface benötigt, das Zugriff auf das Vokabular (Auswahl von Operationen und Datensichten) und auf den Kontext-Server (Suche und Auswahl vorhandener Patientenakten/Behandlungsphasen nach Patientenname oder klinik-internen Schlüsseln der Krankenakten) hat. Ein derartiges Interface einschließlich der notwendigen Zugriffskontroll-Funktionen wird im Rahmen dieser Arbeit nicht explizit beschrieben.

```
typedef struct RecordPermission{
               PatRecord      record;
               RegTerm        vname;
               RegTerm        operation;
               RegTerm        source;
               AttributeSeq   conditions;
};

typedef struct SubjPerm{
               SubjPermId        permID;
               Subject           subject,
               RecordPermission  perm,
               Validity          valid
};
typedef sequence<SubjPerm> SubjPermSeq;

typedef struct DataViewInfo {
               RegTerm     name;
               DVDef       definition;
};

typedef struct PermResult {
               PatRecord        record;
               DataViewInfo     view;
               RegTerm          operation;
               RegTerm          source;
               AttributeSeq     conditions;
};

typedef sequence<PermResult> PermResultSeq;

interface SubjectPermissions{

    attribute PolicyAdmin policyAdm;

    SubjPermSeq        getSubjectPermissions(
                       in  Subject          subject,
                       in  AttributeSeq     pattrib,
                       in  Validity         valid
    )
    raises(InternalError, DataBaseError, InvalidParameter,
           InvalidKey, InvalidValue);

    SubjPermId         assignPermission(
                       in  Subject          subject,
                       in  RecordPermission  perm,
                       in  Validity         valid
    )
```

```
    raises(InternalError, DataBaseError, InvalidParameter,
           InvalidKey, InvalidValue);

    void               removePermission(in SubjPermId permID)
    raises(InternalError, DataBaseError, InvalidID);

    // information retrieval functions

    long               getSubjectPermResults(
                       in  string           subjID,
                       in  AttributeSeq     sparms,
                       out PermResultSeq    results
    )
    raises(InternalError, DataBaseError, InvalidKey, InvalidValue,
           UnknownID, TooManyRecords);
};
```

attribute PolicyAdmin enthält die Objekt-Referenz des PolicyAdmin-Objekts.

getSubjectPermissions() ermittelt alle Subjektrechte, wobei der Suchraum durch die Angabe eines Subjekts (`subject`), eines Zeitraums (`valid`), innerhalb dessen das Recht gültig sein muss, und durch Attribute bezüglich der Patientenakte (`pattrib`) eingeschränkt werden kann. Das Ergebnis ist eine Liste von `SubjPerm`-Strukturen, die eine Verwaltungs-ID, das Subjekt, die Berechtigungsdefinition und den Gültigkeitszeitraum enthalten. Die Berechtigungsdefinition (`RecordPermission`) enthält die `PatRecord`-Struktur, die im Gegensatz zur Kontextdefinition anhand der Patientennummer und weiterer Attribute zur Behandlungsphase eine konkrete Kranken(teil)akte bestimmt. Sie referenziert eine Operation, eine Datensicht und ein Ursprungssystem, und enthält außerdem eine Liste von Bedingungen (Typ `AttributeSeq`), die an das Zugriffsrecht geknüpft sind.

assignPermission() weist das Subjektrecht „`perm`“ dem Subjekt „`subject`“ für den Gültigkeitszeitraum „`valid`“ zu. Der Rückgabewert ist eine interne Verwaltungs-ID, mit der das Recht referenziert werden kann.

removePermission() entfernt das Subjektrecht, das durch „`permID`“ referenziert wird.

getSubjectPermResults() liefert alle Subjektrechte eines Subjekts, deren Gültigeitsintervalle den Abfragezeitpunkt einschließen. Vorbedingungen sind ein gültiger Wert in „`subjID`“, und gültige Attribut/Werte-Paare in „`sparms`“ (siehe Abschnitt 4.2.2 insbesondere Tabelle 4.3). Für die Interpretation der Suchparameter gilt:

1. Generell: Nicht spezifizierte Attribute in der `sparms`-Liste bedeuten, dass Rechte mit allen Werten dieses Attributs zurückgeliefert werden.
2. Generell: Nicht spezifizierte Attribute in der `perm.record.tparms`-Liste des gefundenen Subjektrechts bedeuten, dass das Recht alle

spezifischeren Rechte einschließt. Unter folgenden Voraussetzungen wird deshalb ebenfalls ein „Treffer“ angenommen:

3. UNIT-Attribut: Beim Vergleich wird die Einrichtungshierarchie berücksichtigt. So werden beispielsweise für den Eintrag `(sparms[i].key='UNIT', sparms[i].value='HN')` auch alle Rechte, die UNIT-Attribute mit untergeordneten Einheiten von 'HN' besitzen, ermittelt.
4. START- und END-Attribute: Die Werte werden als Datum interpretiert und verglichen.
5. CURRENT-Attribut: Das Attribut wird nicht berücksichtigt, da keine dynamische Information zu den referenzierten Behandlungselementen miteinbezogen wird.

„`results`“ enthält alle gefundenen Einträge vom Typ „`PermResult`“. Der Funktionsaufruf „`PolicyAdmin::getDVDefinition(results[i].view.name, results[i].view.definition)`“ ergänzt jeden Eintrag results[i] um die DataView-Definition. Der Rückgabewert der Funktion enthält die Anzahl der gefundenen Einträge.

4.4.7 Informationsretrieval

Das MasterAccessControl-Interface bietet die wesentliche Funktionalität des Informationsretrievals. Es wird von allen Subsystem-Clients integriert. Für die Abfrage der Berechtigungen und die Rückgabe der Ergebnismenge sind geeignete Datentypen gemäß dem Informationsmodell (siehe auch Abb. 4.7) definiert.

```
// Datatypes

typedef struct AdminRecord {
       string          patID;
       string          admID;
};
typedef sequence<AdminRecord> AdminRecordSeq;

typedef struct PermRequest {
               AttributeSeq      sparms;
               RegTerm           view;
               RegTerm           operation;
               RegTerm           source;
};

typedef struct UserPermInfo {
               ContextStruct     trContext;
               DataViewInfo      view;
               RegTerm           operation;
               RegTerm           source;
               AttributeSeq      conditions;
};

typedef sequence<UserPermInfo> UserPermInfoSeq;
typedef sequence<octet> OctetSeq;
```

```
interface MasterAccessControl {

    long               requestPermissions(
                       in  string              subj,
                       in  string              role,
                       in  PermRequest         preq
    )
    raises(InternalError, UnknownID, SubjectRoleConflict,
           InvalidKey, InvalidValue, TooManyRecords);

    long               getPatients(out PatientIdSeq pIDSeq)
    raises(InternalError);

    long               getAdminRecords(out AdminRecordSeq arSeq)
    raises(InternalError);

    long               getTreatmentRecords(out PatRecordSeq prSeq)
    raises(InternalError);

    long               getPermissions(out PermResultSeq results)
    raises(InternalError);

    long               getTicket(
                       in  string              subj,
                       in  string              role,
                       out OctetSeq            ticket
    )
    raises(InternalError, UnknownID, SubjectRoleConflict, TooManyRecords);
};
```

requestPermissions() ermittelt die Rechtemenge eines Subjekts oder einer Rolle (oder eines Subjekts in einer Rolle) eingeschränkt durch die Abfragestruktur „`preq`“. Vorbedingung ist, dass „`subj`“ und/oder „`role`“ gültige Werte enthalten. Enthalten beide Variablen einen gültigen Wert, so muss das Subjekt „`subj`“ Inhaber der Rolle „`role`“ sein. Die Abfragestruktur „`preq`“ muss folgende Bedingungen erfüllen:

1. „`preq.sparms`“ enthält gültige Attributbezeichner und dazugehörige Werte. Gültige Attributbezeichner und Werte sind die Schlüsselwörter und Wertebereiche, die in Tabelle 4.3 und im weiteren Abschnitt 4.2.2 beschrieben sind.
2. „`preq.view`“ enthält einen im Vokabular registrierten DataView-Bezeichner, oder „`preq.view.term == '*'`“.
3. „`preq.operation`“ enthält einen im Vokabular registrierten Operationsbezeichner, oder „`preq.operation.term == '*'`“.
4. „`preq.source`“ enthält den registrierten Namen eines Ursprungssystems, oder „`preq.source.term == '*'`“

Mit der Funktion „`RoleHierarchy::getUserPermissions(role, p)`“ liefert das RoleHierarchy-Objekt die Rollenrechte zur Rolle „`role`“; durch den Aufruf „`SubjectPermissions::getSubjPResults(subj, preq.sparms,`

p)“ liefert das SubjectPermissions-Objekt die Subjektrechte zum Subjekt „subj“. Die Ergebnismengen werden durch die Suchkriterien reduziert. Dabei gilt für jedes Recht „p[i]“:

1. „p[i].vname.term“ und „preq.view.term“ stimmen bei identischem Vokabular in den ersten n Zeichen überein, wobei n die Länge von „preq.view.term“ ist.
2. „p[i].operation“ und „preq.operation“ stimmen bei identischem Vokabular in den ersten n Zeichen überein, wobei n die Länge von „preq.operation“ ist.
3. p[i].source.term == preq.source.term

Das ContextEvaluator-Objekt ermittelt die aktuell betroffenen Behandlungsphasen der Rollenrechte. Dazu wird für jedes Rollenrecht „p[i]“ die Funktion „ContextEvaluator::getRecords(p[i].trContext, preq.sparms)“ aufgerufen. Jede zurückgelieferte PatRecord-Struktur liefert zusammen mit den Werten p[i].view, p[i].operation, p[i].source und p[i]. conditions einen Eintrag der internen Ergebnisliste vom Typ PermResult. Weitere Einträge der Ergebnisliste sind – falls vorhanden – die ermittelten Subjektrechte. Rückgabewert der „requestPermissions()“-Funktion ist die Anzahl der Listeneinträge. Die interne Ergebnisliste bleibt bis zum Abbruch der Verbindung oder bis zum erneuten Aufruf der Funktion erhalten.

getPatients() liefert in „pIDSeq“ alle verschiedenen Patienten-IDs aus der aktuellen Ergebnisliste.

getAdminRecords() liefert in „arSeq“ alle verschiedenen Fälle in Form von Patienten-/Aufnahme-ID-Paaren aus der aktuellen Ergebnisliste.

getTreatmentRecords() liefert alle verschiedenen PatRecord-Strukturen der aktuellen Ergebnisliste, wobei die Verschiedenheit nur insofern sichergestellt ist, als sie mit Hilfe der Logik des Behandlungsprozess-Modells erkennbar ist.

getPermissions() liefert die aktuelle Ergebnisliste.

Durch diese Variation der Ergebnisse kann ein spezifischer Client die für ihn relevanten Aspekte in einer gut verwertbaren Form erhalten. Der Rückgabewert jeder der Funktionen ist die Anzahl der ermittelten Einträge.

getTicket liefert das textbasierte Ticket für das Subject „subj“ und zur Rolle „role“; zur Repräsentation der Tickets siehe 4.2.5.

4.4.8 Automatische Wissensakquisition

Automatische Erfassung des Wissens ist überall dort notwendig, wo Zusamenhänge und Abhängigkeiten sich schnell und häufig verändern; sie ist bei allen Vorgängen möglich, die bereits an anderer Stelle erfasst werden. Typisch

dafür sind die Behandlungsprozesse, die in Form von Patientenaufenthalten, Leistungsanforderungen und Leistungsrückmeldungen im Verwaltungssystem und in verschiedenen Spezialsystemen (z. B. im RIS und in Laborsystemen) erfasst werden, aber auch Dienstpläne, die in Personalmanagementsystemen einiger Abteilungen erstellt werden. Das Verwaltungssystem besitzt eine ereignisgesteuerte Kommunikations-Schnittstelle, die Informationen über Patientenaufenthalte und Prozeduren an Subsysteme übermittelt. Ein elektronisches Anforderungssystem für Aufträge an das Zentrallabor befindet sich langfristig in Planung.

Behandlungsprozesse - Das ProcessAcquisition-Interface

Das ProcessAcquisition-Interface gibt die Strukturen und Funktionen vor, mit denen Informationen zum Behandlungsgeschehen in das Prozessmodell umgesetzt werden können, um damit für die Auswertung der Kontexte in geeigneter Form zur Verfügung zu stehen.

```
// Datatypes

typedef struct PatIdent{
                string        pID;
                string        fname;
                string        gname;
                string        dayOfBirth;
                string        birthname;
                string        zip;
                string        city;
                string        countrycode;
                DateTime      eventTS;
};
typedef struct TrElement{
                string        pID,
                string        medID,
                string        admID,
                string        teID,
                string        etype,
                string        ttype,
                RegTerm       location,
                DateTime      begin,
                DateTime      end
                DateTime      eventTS;
};
typedef struct Particip{
                string        pID;
                string        admID;
                string        teID;
                EType         etype;
                RegTerm       unit;
                string        role;
                DateTime      eventTS;
};
```

Das Behandlungsprozess-Modell unterscheidet Behandlungsphasen eines Patienten in einer Organisationseinheit und Beteiligung einer Organisationseinheit an der Behandlungsphase. Die Behandlungssphase wird durch den Datentyp

„TrElement“ repräsentiert. Er enthält die klinikweit gültige Patienten-ID (pID), eine ID des medizinischen Falls (medID), die administrative Fallnummer (admID), eine Element-ID (teID), die die Bewegungsnummer der Verwaltung oder eine Auftragsnummer sein kann, den Elementtyp (etype), einen Behandlungs- bzw elementspezifischen Typ (ttype), die hauptverantwortliche Organisationseinheit der Phase (location) sowie Phasenbeginn und -ende (begin und end). Das Konzept „Medizinischer Fall“ wurde eingeführt, um Behandlungen in Zukunft über den Verwaltungsfall hinaus in einen Zusammenhang bringen zu können. Die notwendige Information steht bislang noch nicht zur Verfügung. Die Menge der Elementtypen (etype) und der elementspezifischen Typen (ttype) können jederzeit erweitert werden. Derzeit gilt: etype ∈ {PRESENCE, DIAGNOSTIC, PROCEDURE} und ttype ∈ {AMB, TST, HST, NST, VST}, falls etype == PRESENCE. Die Differenzierung der Kategorien „DIAGNOSTIC“ und „PROCEDURE“ wurde noch nicht festgelegt, da bislang die Möglichkeit zur automatischen Erfassung der zugehörigen Behandlungselemente fehlt. „Particip“ repräsentiert die Beteiligung an einer Behandlungsphase. Dabei wird die Behandlungsphase mit den Attributen „pID“, „admID“, „teID“ und „etype“ referenziert, „partUnit“ enthält die beteiligte Organisationseinheit und „partRole“ die Art der Beteiligung. Im Datentyp „PatIdent“ werden Identifikationsdaten des Patienten übergeben (Familienname, Vornamen, Geburtsdatum, Geburtsname und Angaben zum Wohnort). Sie werden erfasst, um Behandlungselemente bei der Abfrage der Rechte nach patientenspezifischen Kriterien suchen zu können, und um bei der Erfassung von Subjektrechten das Auffinden der „autorisierbaren“ Elemente (im Sinne der Teilakten) ebenfalls über patientenbezogene Kriterien zu ermöglichen. Jeder der Datentypen hat eine Zeitstempel-Variable (eventTS), die den Zeitpunkt des Ereignisses enthält. Damit kann die Chronologie der Ereignisse korrekt nachvollzogen werden.

```
interface ProcessAkquisition {

    void               addPatient(in  PatIdent idata)
    raises(InternalError, DataBaseError, InvalidParameter, DuplicateID);

    void               updatePatient(in  PatIdent idata)
    raises(InternalError, DataBaseError, InvalidParameter, UnknownID,
           ObsoleteEvent);

    void               removePatient(in  string pID, in  DateTime  eventTS)
    raises(InternalError, DataBaseError, UnknownID, ObsoleteEvent);

    void               mergePatient(
                       in PatIdent    idata
                       in string      oldPID
    )
    raises(InternalError, DataBaseError, UnknownID);

    void               addTreatmentElement(in TrElement trElement)
    raises(InternalError, DataBaseError, InvalidParameter, DuplicateID);

    void               updateTreatmentElement(in TrElement trElement)
    raises(InternalError, DataBaseError, InvalidParameter, UnknownID,
```

```
        ObsoleteEvent);

    void            removeTreatmentElement(
                    in  string     pID,
                    in  string     admID,
                    in  string     teID,
                    in  EType      etype,
                    in  DateTime   eventTS
    )
    raises(InternalError, DataBaseError, UnknownID, ObsoleteEvent);

    void            notifyDischarge(
                    in  string     pID,
                    in  string     admID,
                    in  string     date_time
                    in  DateTime   eventTS
    )
    raises(InternalError, DataBaseError, InvalidParameter, UnknownID,
        ObsoleteEvent);

    void            updateDischarge(
                    in  string     pID,
                    in  string     admID,
                    in  string     date_time
                    in  DateTime   eventTS
    )
    raises(InternalError, DataBaseError, InvalidParameter, UnknownID,
        ObsoleteEvent);

    void            removeDischarge(
                    in  string     pID,
                    in  string     admID,
                    in  DateTime   eventTS
    )
    raises(InternalError, DataBaseError, InvalidParameter, UnknownID,
        ObsoleteEvent);

    void            addParticipation(in Particip part)
    raises(InternalError, DataBaseError, InvalidParameter, UnknownID,
        ObsoleteEvent);

    void            removeParticipation(in Particip part)
    raises(InternalError, DataBaseError, InvalidParameter, UnknownID,
        ObsoleteEvent);
};
```

addPatient() legt einen neuen Datensatz mit Patienten-Identifikationsdaten „`idata.pID`“ an, vorausgesetzt es existiert noch kein Satz mit der Patienten-ID „`idata.pID`“.

updatePatient() aktualisiert den Identifikations-Datensatz zur Patienten-ID „`idata.pID`“ mit den Werten in „`idata`“.

removePatient() entfernt den Identifikations-Datensatz zur Patienten-ID „`pID`“, falls ein entsprechender Datensatz existiert.

mergePatient() führt zwei Datensätze, die zu einer Person gehören, zu einem Datensatz zusammen. „`idata`“ enthält die aktuell gültigen Identifikationsdaten. Dabei wird auch in allen „`TrElement`“- und „`Particip`“-Instanzen „`oldPID`“ durch „`idata.pID`“ ersetzt.

addTreatmentElement() erzeugt eine neue Behandlungselement-Instanz mit den Daten „`trElement`“, sofern noch keine Instanz mit den Attributen „`trElement.pID`“, „`trElement.admID`“, „`trElement.teID`“ und „`trElement.etype`“ existiert. Diese Attribute legen ein Behandlungselement eindeutig fest.

updateTreatmentElement() aktualisiert die Element-Instanz, die durch die Attribute „`trElement.pID`“, „`trElement.admID`“, „`trElement.teID`“ und „`trElement.etype`“ eindeutig definiert ist, falls diese existiert.

removeTreatmentElement() entfernt die Element-Instanz mit den Attributen „`trElement.pID`“, „`trElement.admID`“, „`trElement.teID`“ und „`trElement.etype`“ – falls vorhanden.

Manche der Daten – insbesondere das Phasenende – werden beim Erfassen der Behandlungselemente nicht übergeben. In diesem Fall kann das Phasenende teilweise intern nach festgelegten Regeln aus weiteren Ereignissen implizit ermittelt werden. Speziell bei Aufenthaltsphasen ist das Phasenende durch den Beginn der nächsten Phase des gleichen administrativen Falls oder durch die Entlassung definiert. Deshalb muss der Entlassungszeitpunkt zusätzlich erfasst und aktualisiert werden.

notifyDischarge() zeigt das Ende eines Verwaltungsfalls (Entlassung) an. Der Entlassungs-Zeitpunkt „`date_time`“ wird als Ende der Element-Instanz mit dem jüngsten Startzeitpunkt interpretiert, die Element-Instanz entsprechend aktualisiert.

updateDischarge() zeigt eine Änderung des Entlassungszeitpunkts an. Der Endzeitpunkt der letzten Element-Instanz gemäß der Chronologie der Anfangszeitpunkte wird durch „`date_time`“ ersetzt.

removeDischarge() zeigt an, dass das zuvor registrierte Fall-Ende ungültig ist. Die letzte Element-Instanz gemäß der Chronologie der Anfangszeitpunkte wird als unbeendet gekennzeichnet.

addParticipation() registriert die Beteiligung der Organisationseinheit „`part.unit`“ mit der (Beteiligungs-)Rolle „`part.role`“ an der Behandlungsphase, die durch „`part.pID`“, „`part.admID`“, „`part.teID`“ und „`part.etype`“ referenziert wird.

removeParticipation() löscht die referenzierte Beteiligung – falls vorhanden.

Für jede Funtionen, die vorhandene Instanzen verändert, mit Ausnahme von „`mergePatient()`“ gilt: Liegt der Ereigniszeitpunkt „`eventTS`“ der Funktionsparameter in der Chronologie vor dem zuletzt erfassten Ereignis, so wird die

Funktion nicht durchgeführt. Damit wird verhindert, dass Ereignisse die außerhalb der Chronologie registriert werden zu falschen Zuständen führen. Die abgewiesenen Aktualisierungen müssen gegebenfalls manuell korrigiert werden.

Dienstpläne

Dienstpläne enthalten Informationen über den Einsatz von Personen in bestimmten Funktionen und zu bestimmten Zeiten. Durch Modellierung geeigneter Rollen können Dienstpläne als Rollenzuweisungen im Subjektverzeichnis abgebildet werden. Für die automatische Wissenserfassung kann dann das Authorization-Interface folgendermaßen genutzt werden: Die Dienstplan-Anwendung erhält eine interne, im günstigsten Fall konfigurierbare Zuordnung zwischen Aufgaben und korrespondierenden Rollen. Jeder Eintrag aus Person, Aufgabenbereich und Zeitraum im Dienstplan wird durch einen „`assignRole()`“-Funktionsaufruf des Authorization-Objekts in eine Rollenzuweisung übersetzt. Durch die Definition von „vorausgesetzten Rollen“ (Prerequisite-Constraints), kann außerdem festgelegt werden, welchen Personen welche Rollen grundsätzlich zugewiesen werden dürfen. Diese Rolleninformation kann die Dienstplan-Anwendung wiederum zur Vorauswahl der Personen verwenden.

Kapitel 5

Realisierung

Zur Realisierung des Zugriffskontroll-Systems gehören die Implementation der Komponenten, die Integration der MasterAccessControl-Schnittstelle in vorhandene Informationssysteme und die Erfassung der Zugriffspolitik, also der notwendigen Vokabulare, Regeln, Rechte und Rollen. Abschnitt 5.1 enthält Angaben zum Stand der Implementation für das Mainzer Universitätsklinikum, und ein gestaffeltes Konzept zur weiteren Entwicklung der beschriebenen Schnittstellen und der benötigten Benutzerwerkzeuge inklusive der eingesetzten bzw. einzusetzenden Hard- und Software. Der Abschnitt enthält ferner Betrachtungen zum Aufwand, der Performanz und Verfügbarkeit des Systems sowie der Vertraulichkeit und Integrität der enthaltenen Daten. Abschnitt 5.2 behandelt die Umsetzung der Zugriffspolitik. An Hand von Beispielen wird die Definition der Rollen und Rechte, der Vokabulare, Kontextregeln und Datensichten sowie die korrespondierenden Abfragen demonstriert. Die Integration der Client-Funktionalität in klinische Informationssysteme ist Inhalt von Abschnitt 5.3. Neben allgemeinen technischen Anmerkungen zur Integration werden die Möglichkeiten der Integration an drei konkreten Systembeispielen aus dem Klinikum diskutiert.

5.1 Implementation

Das Zugriffskontrollsystem besteht aus mehreren, nur durch definierte Schnittstellen miteinander gekoppelten Komponenten: dem Kontext-Service, dem Rollen-Service, dem Autorisierungs-Service und dem Zugriffskontroll-Service. Die verschiedenen Komponenten werden unabhängig voneinander in mehreren Phasen realisiert. In der ersten Phase soll eine Kernfunktionalität erreicht werden, mit der das System in ausgewählte Klienten integriert und mit einer begrenzten Benutzerzahl erprobt werden kann. Das erfordert vor allem den vollständigen Funktionsumfang des Informationsretrievals. In der zweiten Phase muss die Funktionalität in einem Umfang ergänzt werden, der den Einsatz auch mit einer großen Benutzerzahl ermöglicht. Das erfordert vor allem die effiziente Nutzbar-

keit der Autorisierungswerkzeuge. In weiteren Phasen soll die volle Funktionalität erreicht werden, was vor allem eine Vervollständigung der Administrationswerkzeuge beinhaltet.

Der folgende Abschnitt enthält für jede Komponente eine kurze Beschreibung der Implementationsschritte, der verwendeten Werkzeuge und Techniken, den derzeitigen Realisierung-Stand sowie die geplante Verteilung auf verschiedene Rechner.

5.1.1 Der Rollen-Service

Die Umsetzung des Rollen-Service, d. h. der Rollen- und Rechtedefinition sowie des Rechteretrievals wurde als Thema einer Diplomarbeit vergeben, die zur Zeit am Institut für Medizinische Statistik und Dokumentation angefertigt wird. In diesem Rahmen entsteht ein Service mit web-basierten Klienten, der in der ersten Realisierungsphase die notwendige Kernfunktionalität zur Verfügung stellt. Die gesamte Realisierung kann in folgende Phasen – nach abnehmender Dringlichkeit – gegliedert werden:

1. **Phase: Kernfunktionalität** Die Kernfunktionalität umfasst die Definition von Rechten aus vorgegebenen Operationen, Datensichten und Kontexten, die Zusammenstellung von Rollen auf der Basis der definierten Rechte und ihre hierarchische Anordnung, sowie das Retrieval der Rollenrechte als Schnittstelle zum Zugriffskontroll-Service. Es ist sinnvoll, schon in dieser Phase Zugriffskontrollmechanismen für die dezentrale Rollendefinition zu implementieren. Ein einfach strukturiertes Vokabular kann zunächst direkt mit vorhandenen Datenbank-Importwerkzeugen erfasst werden, einfache Datensichten können ohne Eingabeunterstützung textbasiert erstellt und auf die gleiche Weise gespeichert werden. Eine Benutzeroberfläche zur Bearbeitung von Kontextregeln entfällt in der ersten Realisierungsphase (→ 5.1.2).

2. **Phase: Vokabular- und Datensicht-Editor** In dieser Phase sollen Werkzeuge entstehen, die die Definition der Rechte-Bausteine auf komfortable Art und Weise, auch durch Personen ohne spezifische Kenntnisse des Systems, erlauben. Das beinhaltet einen grafischen Editor zur Erfassung und Pflege des Vokabulars. Hier sollen auch Konsistenzprüfungen bezüglich bereits verwendeter Begriffe integriert werden. Die Funktion des Datensicht-Editors ist die unterstützte Erfassung von Datensichten in unterschiedlichen Formaten. Er soll die Zusammenstellung einer Datensicht mit Drag& Drop- oder Markierungs-Operationen auf einer grafischen Repräsentation des ausgewählten Datenklassen-Vokabulars ermöglichen. Bei einer vollständig automatisierten Generierung des Datensichtformats aus der grafischen Darstellung müssen keine Sytaxprüfungen implementiert werden. Empfehlungen zur Gestaltung der Oberflächen wurden bereits in Abschnitt 4.4.3 formuliert.

3. **Phase: Kontextregel-Editor** Die Funktion des Kontextregel-Editors ist

die unterstütze Erfassung der formulierten Regelarten, d. h. insbesondere der Bedingungs- und Parameterlisten aus den vorgebenen Typen und Wertemengen. Für die Bedingungsdefinitionen sind Syntaxprüfungen[1] zu integrieren. Empfehlungen zur Gestaltung des Kontextregel-Editors wurden auch schon in Abschnitt 4.4.4 gegeben.

Die Implementation erfolgt als CORBA-Service mit ORBacus-4.0.4 und Java-Klienten; die Daten werden in einer Postgres-Datenbank gespeichert. Die Datenbank und der CORBA-Service werden zunächst testweise auf einem LINUX-Server laufen. Eine Produktionsversion mit hinreichender Verfügbarkeit kann künftig auf dem hochverfügbaren HP-Cluster des IMSD betrieben werden.

5.1.2 Der Kontext-Service

Der Kontext-Service besteht aus zwei unabhängigen Implementationen, der automatischen Wissenserfassung und dem ContextEvaluator-Modul, die auf denselben relationalen Datensätzen operieren. Die Datenbanktabellen wurden im Hinblick auf eine optimale, performante Kontextregel-Auswertung entworfen.

Datenbankstrukturen

Die Kern-Tabelle „ORG_RELATION “ enthält sowohl die Information über Behandlungselemente als auch über Beteiligungen. Sie besitzt folgende Struktur:

PID	enthält die klinikweit eindeutige Patienten-ID
MED_ID	enthält eine ID, die an alle Behandlungselemente vergeben wird, die zum gleichen medizinischen Fall gehören. Zur Zeit fehlt diese Information. Deshalb wird diese Spalte bisher ersatzweise mit der administrativen Fall-ID gefüllt.
ADM_ID	enthält die administrative Fall-ID.
ELEMENT_ID	enthält eine eindeutige ID des Behandlungselements, z. B. die Bewegungsnummer des Verwaltungssystems.
SOURCE	enthält einen Bezeichner für die Herkunft der Daten. Diese Information wird für die internen Verarbeitungsvorgänge verwertet.
TYPE	enthält den Elementtyp.

[1] Innerhalb der Bedingungen müssen die Kombinationen von Operatoren und Operanden bzw. Funktionen und Argumenten sowie Schlüsselwörter, Werte und Wertebereiche überprüft werden.

SUBTYPE	enthält einen weiteren elementspezifischen Typ.
ORG_UNIT	enthält den Bezeichner der Organisationseinheit der Beziehung.
BEGIN	enthält den Anfangszeitpunkt der Behandlungsphase.
END	enthält den Endzeitpunkt der Behandlungsphase.
PRESENT	ist ein boolscher Wert, der angibt, ob der Patient sich bezüglich dieses Falls in einer Behandlungsphase befindet.
MED_BEGIN	enthält den Anfangszeitpunkt des gesamten Falls.
MED_END	enthält den Abschlusszeitpunkt des gesamten Falls.

Aus den Ereignissen der Verwaltung entstehen Einträge, deren „TYPE“-Spalte den Wert `'PRESENCE'` (für Patientenaufenthalte) enhält. In der Spalte „SUBTYPE“ wird die Behandlungsart (`'HST'` für hauptstationär, `'AMB'` für ambulant, etc.) gesetzt. In „ORG_UNIT“ steht der Aufenthaltsort des Patienten bzw. die Organisationseinheit, in der die Behandlung stattfindet. Bei der Erfassung von Beteiligungen werden die IDs und die zeitlichen Informationen der korrespondierenden Behandlungsphasen übernommen. Die Spalte „TYPE“ enthält dann den Wert `'PART'`, „SUBTYPE“ den Bezeichner der „Beteiligungs-Rolle“ und „ORG_UNIT“ den Bezeichner der beteiligten Organisationseinheit.

Zu jedem registrierten Patienten existiert außerdem ein Eintrag in der Tabelle „PID_SEARCH, der Identifikationsdaten (Name, Vornamen, Geburtsname, Geburtsdatum, Wohnort) und phonetische Formen des Patientennamen enthält.

Optimierungsaspekte

Die Datenrepräsentation wurde so gewählt, dass alle wesentlichen Abfragen bei der Kontextauswertung über genau eine Tabelle ausgeführt werden. Das wurde im einzelnen durch die Hinzunahme der Felder „PRESENT“, „MED_BEGIN“ und „MED_END“, die Informationen zum gesamten Fall enthalten, und durch die Abbildung der Beteiligungen als speziellen Typ von Behandlungselementen erreicht. Mit Hilfe geeigneter Tabellen-Indizes können die Kontextregeln mit der notwendigen Geschwindigkeit ausgewertet werden. Bei der Suche mit Patienten-Identifikationsdaten wird vorrangig über die Tabelle „PID_SEARCH“ abgefragt, die zugehörigen Fälle werden durch den Fremdschlüssel „PID“ effizient gefunden.

Aufgrund der hierarchischen Bezeichner für Organisationseinheiten, die in den Behandlungselementen referenziert werden, findet die Suche nach Phasen einer Organisationseinheit durch Verwendung regulärer Ausdrücke ohne Mehraufwand auch alle Phasen untergeordneter Einrichtungen, was ebenfalls zur Auswertungsgeschwindigkeit beiträgt.

Automatische Wissenerfassung

Die automatische Wissenserfassung wurde bisher in Teilen prototypisch implementiert und ist Teil der ersten Realisierungsphase. Sie basiert auf einer Kopplung an den Kommunikationsserver, der Nachrichten über klinische Ereignisse verteilt. Einzige Quelle der Information über Behandlungsprozesse ist zum momentanen Zeitpunkt das Patientenverwaltungssystem SAP/IS-H. Zu allen administrativen Ereignissen (Aufnahme, Verlegung, Entlassung, Aktualisierungen etc.) werden Nachrichten generiert, die (in Zeitperioden von sechs Minuten) über eine Subsystemschnittstelle zum Kommunikationsserver gelangen. Dort werden die behandlungsprozess-relevanten Informationen extrahiert und an den Kontextserver weitergeleitet. Ein permanenter Prozess des Kontextservers nimmt die Nachrichten vom Kommunikationsserver entgegen, interpretiert die Nachrichtenstruktur und ruft Funktionen zur Erfassung der Behandlungselemente auf. Die Funktionen sind Teil einer C-Bibliothek, die bereits fertiggetstellt wurde; der Prozess läuft in der beschriebenen Form seit April 2001. Behandlungsprozessdaten zu früheren Fällen wurden aus dem Datenbestand der Referenzdatenbank übernommen.

Informationen über die Beteiligung an einer Behandlung sollen in einer der späteren Phasen gewonnen werden, sobald eine Kopplung an Anforderungssysteme für Befund- und Behandlungsleistungen sowie Terminplanungssysteme möglich ist. Dazu muss die Bibliothek um Routinen erweitert werden, die Information über „Behandlungs-Beteiligungen“ in der oben beschriebenen Weise integrieren.

Bis zur Realisierung dieser Systeme müssen fehlende Informationen bei der Umsetzung der Zugriffspolitik berücksichtigt werden.

Das ContextEvaluator-Modul

Da die Implementation der allgemeinen Kontextregelerfassung und -auswertung relativ aufwendig ist, werden die wichtigsten Kontexte (siehe Abschnitt 5.2.1) für die Kernfunktionalität zur Zeit als statische Routinen implementiert. Sie erlauben die Verwendung der Retrieval-Funktionen so, wie in den Schnittstellen (Abschnitt 4.4.4) beschrieben, ersparen aber zunächst die aufwendigere Implementation und Regeldefinition. In der zweiten oder dritten Realisierungsphase sollte die dynamische Definition und Auswertung von Kontextregeln umgesetzt werden, da dies die Anpassung des Systems an verschiedene Umgebungen und Zugriffspolitiken ermöglicht und weitere Programmänderungen auf lange Sicht überflüssig macht. Die Regeldefinitionen werden dann in der Kontextdatenbank gespeichert. Zur Laufzeit (und zur Aktivierung neuer oder geänderter Regeln) werden sie vollständig in den Programmspeicher geladen und zu Gunsten der Abfrageperformanz bereits in dynamische SQL-Statements umgesetzt. Der ContextEvaluator wird als CORBA-Objekt mit ORBacus-4.0.4 in C++ implementiert und soll grundsätzlich auf dem Rechner laufen, auf dem sich die Kontextdatenbank befindet.

5.1.3 Der Autorisierungs-Service

Der rollen-basierte Autorisierungs-Service

In der ersten Realisierungs-Phase können die Rollenzuweisungen – für eine geringe Benutzerzahl – direkt in das Benutzerverzeichnis erfasst werden. Dazu genügen einfache Perl-Skripte, die die LDAP-Schnittstelle des Benutzerverzeichnisses ansprechen. Für eine größere Anzahl von Subjekten wird ein Autorisierungswerkzeug benötigt, dass auch von Personal ohne spezielle Systemkenntnisse zunächst an zentraler Stelle bedient werden kann. Der Funktionsumfang eines solchen Autorisierungs-Editors beinhaltet die Anzeige der vorhandenen Rollen, der Subjekte und der bereits existierenden Zuweisungen sowie leicht bedienbare Funktionen, um Rollen-Zuweisungen zu bearbeiten. Zu dieser Realisierungsphase gehört auch die Implementation der notwendigen Abhängigkeitsprüfungen (Rollen-Constraints). Mit wachsender Subjektzahl müssen die Werkzeuge für eine dezentrale Nutzung erweitert werden. Dafür werden vor allem Kontrollfunktionen für Rechte und Zuständigkeitsbereiche verschiedener Administratoren benötigt.

Der Autorisierungs-Server soll als CORBA-Service mit webbasierten Klienten implementiert werden. Autorisierungs-Service und Rollen-Service können in Form einer Komponente implementiert werden; andernfalls sollten sie zumindest auf dem gleichen Rechner laufen, da sie auf gemeinsame Datenquellen zugreifen. Die Kommunikation mit dem Benutzerverzeichnis findet über eine SSL-gesicherte Verbindung statt.

Ein eigenständiger Teil der Autorisierung, der von normalen Nutzern ohne spezielle Administratorrechte ausgeführt werden soll, ist die Definition der Subjektrechte. Dieser Teil sollte realisiert werden, sobald das Zugriffskontroll-System mit einer größeren Zahl von Personen produktiv eingesetzt wird, da die explizite Datenfreigabe gut geeignet ist, über Informationslücken des Zugriffskontrollsystems hinwegzuhelfen. Der Datenfreigabe-Service umfasst neben dem Retrieval der Subjektrechte folgende Funktionalität für die Definition und die Bearbeitung der expliziten Rechte: Anzeige und Auswahl von Subjekten, Anzeige und Auswahl von Patientenakten, Anzeige und Auswahl von Operationen und Datensichten und Erfassung des Gütigkeitszeitraums. Die Anzeige der Subjekte wird über eine gemeinsame Schnittstelle mit dem rollenbasierten Autorisierungs-Service zum Benutzerverzeichnis (→ SubjectDirectory-Interface) realisiert. Die Anzeige von Patienten(teil)akten erfolgt auf der Basis der Kontextinformationen; hier wird eine weitere Schnittstelle zum Kontextserver benötigt. Es sollte aber zusätzlich möglich sein, Patientenakten über Patienten-, Fall-IDs und Zeiträume frei zu referenzieren, um auch Lücken im Kontextwissen überbrücken zu können. Für die Referenzierung der Operationen und Datensichten in den Rechten, wird die Schnittstelle des Rollen-Servers zum Vokabular (→ Vocabulary-Interface) verwendet. Die Subjektrechte können gemeinsam mit den Rollenrechten gespeichert werden. Der Service sollte als eigene Anwendung (CORBA-Service mit Web-Clients) umgesetzt werden. Eine Integration in den rollen-basierten Autorisierungsservice ist nicht sinnvoll, da

dieser ein Administrations-Werkzeug darstellt, wohingegen der Datenfreigabe-Service ein Benutzerwerkzeug ist.

5.1.4 Der Zugriffskontroll-Service

Der Zugriffskontroll-Service wird zur Zeit als CORBA-Service mit ORBacus-4.0.4 in C++ realisiert. Er umfasst die volle Funktionalität, die in der Schnittstellen-Beschreibung (→ 4.4.7) angegeben wurde. Für jede Sitzung eines AccessControl-Client wird ein MasterAccessControl-Serverobjekt erzeugt, das die Sitzungsinformation verwaltet. Für die Verbindung zwischen Client und Server wird das „Secure Socket Layer Protocol“ (SSL) mit Client- und Server-Authentisierung verwendet. Das bedeutet, dass sich sowohl der Server als auch jeder Client mit einem gültigen X.509-Zertifikat authentisiert. Die Client-Funktionalität wird in Form einer Bibliothek zur Verfügung gestellt, die vom jeweiligen Informationssystem eingebunden werden muss. Die Funktionen sind so implementiert, dass ein lokales Objekt, das mit den notwendigen Verbindungs- und Zertifikat-Informationen initialisiert wird, die vollständige Verbindungssteuerung übernimmt, alle Funktionsaufrufe an den Service vermittelt und die Ergebnisse zurückliefert. Es verhält sich dabei wie ein lokaler MasterAccessControl-Server. Funktionen zur Aufbereitung der Ergebnismenge können dabei im client-seitigen Objekt ablaufen, ohne den Server zu belasten. Die Integration der Client-Funktionalität in vorhandene Systeme wird in Abschnitt 5.3 behandelt.

5.1.5 Sicherheitsbetrachtungen

Informationen über registrierte Benutzer und ihre Rechte, sowie die implizit im Zugriffskontroll-System enthaltenen Informationen über Behandlungen müssen mit der gleichen Sorgfalt behandelt werden wie Personendaten im allgemeinen. Alle notwendigen Maßnahmen dazu werden bei der Implementation berücksichtigt. Die Sicherheitsfrage wird hinsichtlich der folgenden drei Aspekte betrachtet, der Verfügbarkeit, der Integrität und der Vertraulichkeit der Informationen.

Verfügbarkeit

Hinsichtlich der Verfügbarkeit kann man die verschiedenen Funktionen des Systems drei Gruppen zuordnen: alle Datenbankserver und alle Funktionen des Informationsretrievals, die höchste Verfügbarkeit benötigen, die Funktionen der automatischen Wissenserfassung – insbesondere des Kontextwissens, die eine hohe Verfügbarkeit erfordern und alle Funktionen der manuellen Wissenserfassung, für die eine geringere Verfügbarkeit ausreicht. Die verschiedenen Verfügbarkeitsstufen haben folgende Bedeutungen und Konsequenzen:

Höchste Verfügbarkeit heißt in diesem Zusammenhang, dass das System innerhalb von maximal einigen Sekunden ein korrektes Anfrageergebnis zurückliefern muss. Eine derart hohe Verfügbarkeit kann nur durch ausreichende Redundanz, durch Stabilität und durch Performanz-Optimierung

erreicht werden. Voraussetzung für Stabilität sind sorgfältige Programmierung und zuverlässige Server. Die Performanz wird durch leistungfähige Rechner und Optimierung jeder einzelnen beteiligten Komponente erreicht. Ein Beispiel ist die oben erwähnte Optimierung des Kontextservers schon beim Datenbank-Design. Darüberhinaus kann das notwendige Maß an Verfügbarkeit nur durch Redundanz aller beteiligten Komponenten erreicht werden. Das heißt, dass der MasterAccessControl-Server parallel auf zwei Rechnern laufen muss, die alternativ angesprochen werden können. Ebenso sollten je zwei Rollen- und Kontext-Server parallel betrieben werden. Bei der Wissenserfassung müssen jeweils beide Komponenten versorgt werden. Problematisch bleibt hier auch, dass der Ausfall einzelner Netzwerk-Komponenten trotz aller Maßnahmen die Abfrage von Rechten verhindern kann.

Hohe Verfügbarkeit erfordert eine Reaktionszeit von wenigen Minuten[2] – in diesem Fall, bis eine Zustandsänderung im Kontextserver registriert ist. Als schlimmste Konsequenz werden unvollständige Abfrageergebnisse generiert, also implizit Zugriffe abgelehnt. Dieser Fall muss, wie auch alle anders verursachten Wissenslücken[3], als Ausnahme in den Informationssystemen behandelt werden. Da die Verfügbarkeit der Datenquelle aufgrund des oben beschriebenen Konzepts gewährleistet ist, genügt ein Mechanismus, der den Wissenserfassungs-Prozess überwacht, Fehlerbenachrichtigungen generiert, und den Prozess gegebenfalls neu startet.

Geringe Verfügbarkeit erlaubt notfalls auch längere Ausfälle, in denen keine neuen Rechte oder Rollen definiert und keine Benutzer autorisiert werden können. Für die Erfassungswerkzeuge müssen also nur „normale Vorkehrungen“ getroffen werden.

Abgesehen von der Duplizierung der produktiven Komponenten, ist ein vollständiges Testsystem notwendig, das bei Performanz- und Funktionstests nach Korrekturen oder Erweiterungen der Software sowie bei der Anbindung neuer Klienten zum Einsatz kommt.

Jedes System, das die Funktionalität des Zugriffskontroll-Systems integriert, muss trotz aller Maßnahmen zur Sicherstellung der Verfügbarkeit ein Ausfallkonzept implementieren. Längere Ausfälle dürfen in keinem Fall dazu führen, dass Zugriffe zu möglicherweise notwendigen Informationen verwehrt werden. Bei einem Ausfall sollte der Notfall-Zugriffsmodus aktiviert werden, der vollen Lesezugriff auf die Daten in Verbindung mit detaillierter Protokollierung der Zugriffe gewährt. Bei Verzögerungen oder kurzen Ausfällen sollte die Abfrage durch das System angemessen oft wiederholt werden.

[2] Optimalerweise sollte eine Minute nicht überschritten werden.

[3] Die derzeitige Kommunikationsschnittstelle des Verwaltungssystems hat beispielsweise eine Latenzzeit von sechs Minuten zwischen Ereignis und Nachrichtenversand.

Integrität und Vertraulichkeit

Um die Informationen vor Verfälschung und Kenntnisnahme durch Unbefugte zu schützen werden folgende Maßnahmen getroffen:

SSL-basierte Verbindungen Alle Verbindungen zwischen Klienten und Servern des Zugriffskontroll-Systems werden unter Verwendung des Secure Socket Layer Protokolls realisiert. Auch die nachrichten-basierten Schnittstellen der Wissenserfassung und die Verbindung zum LDAP-Benutzerverzeichnis verwenden SSL. Das bedeutet, dass einerseits alle Teilnehmer des Systems durch X.509-Zertifikate authentisiert werden, und andererseits, dass alle Daten grundsätzlich verschlüsselt übertragen werden. Dadurch sind die Daten bei der Übertragung vor Verfälschung und Einsicht durch Unbefugte geschützt.

Lokale Datenbankzugriffe Da die verfügbaren Datenbanken keinen geschützten Remote-Zugriff ermöglichen, müssen alle Komponenten, die auf Datenbanken zugreifen (Rollen-Server, Kontext-Server, Datenfreigabe-Service) entweder auf dem Rechner betrieben werden, auf dem die jeweilige Datenbank läuft, oder die Verbindungen müssen auf Rechner- und Netzwerkebene adäquat geschützt sein. In [16, 18] sind grundsätzliche Verfahren und Maßnamen zum Schutz von Systemen und Netzen beschrieben.

Rechnersicherheit Alle Rechner, auf denen Personendaten gespeichert werden, müssen mit der notwendigen Sorgfalt administriert werden [18]. Das heißt im einzelnen, dass keine zusätzlichen Anwendungen laufen, insbesondere keine, die Zugriff auf den Rechner erlauben, wie Webserver, Remote-Login, FTP, NFS, und dass keine unnötigen Benutzer-Konten angelegt werden. Die notwendigen Komponenten, z. B. CORBA-ORB, Webserver müssen mit minimalen Rechten konfiguriet werden.

Protokollierung der administrativen Zugriffe Alle Manipulationen von Rechten, Rollen und Autorisierungs-Relationen werden protokolliert.

5.2 Umsetzung der Zugriffspolitik

Der folgende Abschnitt enthält Beispiele und Vorschläge, wie Rechte, Kontextregeln, Operationen und Datensichten sowie Rollen und Rollenbeziehungen erfasst werden können, um die wesentlichen Bestimmungen der Zugriffspolitik mit angemessenem Aufwand und unter Berücksichtigung der technischen Möglichkeiten der beteiligten Informationssysteme zu erfüllen. Die Feinheit, in der die Berechtigungen vergeben werden müssen, wird grundsätzlich unter Mitwirkung des Datenschutzbeauftragten geregelt, und kontinuierlich an verbesserte technische Möglichkeiten angepasst, bis die aus der Zweckbindung der Patientendaten abgeleiteten Zugriffsregelungen vollständig erfüllt sind. Der folgende Abschnitt soll vor allem verdeutlichen, dass die gewählte Systematik neben komplexen

Rechten und Rechtebausteinen, für die teilweise Beispiele z. B. in 4.2.4 angegeben wurden, auch die Definition einfacher Rechte mit relativ geringem Aufwand in der Anfangsphase unterstützt.

Zum jetztigen Zeitpunkt[4] müssen vor allem die Systeme, die klinikweit Zugriff auf Patientendaten ermöglichen sollen, mit Berechtigungsinformation versorgt werden. Zu diesen Systemen gehören das Patientenverwaltungssystem, alle existierenden zentralen Befundserver, die Referenzdatenbank, ein System zur Meldung von Tumordiagnosen an das Krebsregister und der Kommunikationsserver. In einer weiteren Phase werden voraussichtlich ein System für die elektronische Leistungsanforderung, das System zur Freigabe von Patientendaten (z. B. für Konsile) und einige klinische Informations- und Dokumentationssysteme hinzukommen, die für eine Kooperation zwischen verschiedenen Fachkliniken eingesetzt werden sollen.

5.2.1 Rechtedefinition – Vokabulare

Operationen

Der erste Schritt der Definitionsphase ist die Analyse der adressierten Informationssysteme hinsichtlich ihrer Funktionalität. Dabei sollte zunächst eine Gliederung in grobe Funktionsbereiche vorgenommen werden. Im vorliegenden Fall könnte das eine Aufteilung in die Bereiche „Information“, „Dokumentation“, „Anforderung“ und „Datenfreigabe“ sein, die jederzeit erweiterbar ist. Der Be-

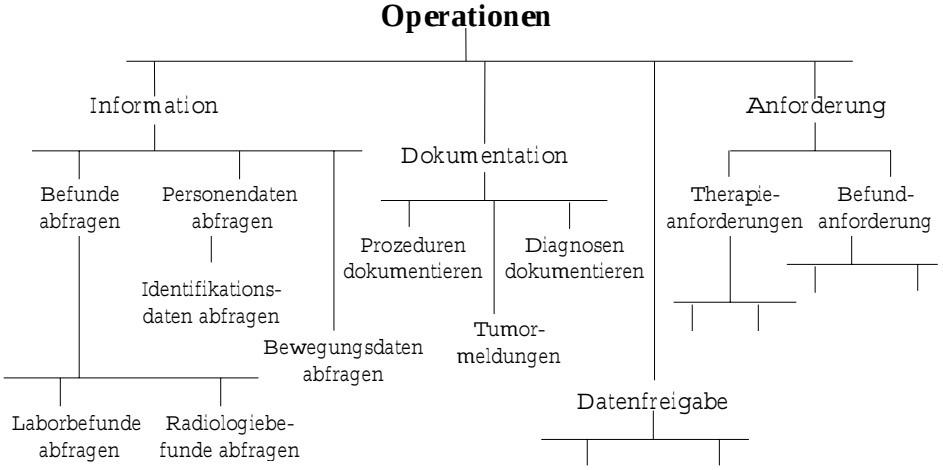

Abbildung 5.1: Beispiel einer Funktionsgliederung

reich der Patientenverwaltung und Abrechnung wird hierbei nicht berücksichtigt, da das Patientenverwaltungs-System eine sehr differenzierte und eng mit

[4] April 2001

dem System verflochtene Berechtigungssystematik besitzt, die nur mit unangemessen großem Aufwand externalisiert werden könnte.[5] Mit einem speziellen Schlüsselwort, „INTERN“, kann bei der Definition der Rechte signalisiert werden, dass die Differenzierung der Operations- und Datenklassen nach Benutzern vom System intern und eigenständig durchgeführt werden muss. Diese Variante kann nur für systemspezifische Rechte umgesetzt werden, die aber im Sinne einer einheitlichen Zugriffspolitik – wenn möglich – vermieden werden sollten.

Jeder einzelne Bereich wird soweit untergliedert, wie es die Funktionalität der adressierten Systeme bzw. die Differenzierung der Berechtigung erfordert. Die Gliederungsschritte sollten dabei so gewählt sein, dass je nach Bedarf mehr oder weniger differenzierte Rechte abgeleitet werden können. Abbildung 5.1 zeigt eine einfache (unvollständige) Funktionsgliederung. Für diese Funktionssystematik kann nun entweder ein Vokabular von Operationsbezeichnern selbst definiert, oder – soweit möglich – exstierende Vokabulare auf die Systematik abgebildet werden. Wichtigster Grundsatz bei der Modellierung ist, die Systematik so einfach wie möglich und so differenziert wie nötig zu gestalten.

Datenfilter- und klassen

Der vorangegangenen Abschnitt zeigt, dass die Datenklassen, die von einem Zugriff betroffen sind – oft schon durch die Funktion festgelegt – implizit im Operationsbezeichner enthalten sind. Differenzierte Datensichten sind aufwendig zu definieren und für die adressierten Systeme ebenso aufwendig auszuwerten. Sie sind vor allem dann sinnvoll, wenn verschiedene Personengruppen die gleiche Funktion eines Systems auf unterschiedlichen Datenausschnitten ausführen dürfen. In den ersten Realisierungsphasen muss abgewogen werden, ob und wo differenzierte Datensichten benötigt werden. Für das Abfragen von Befunden ist beispielsweise zunächst keine Unterscheidung zwischen verschiedenen Datenausschnitten für verschiedene Personengruppen zu erwarten. Die Unterscheidung zwischen dem Lesen der Befunddaten und dem Feststellen der Existenz eines Befundes (z. B. eines Röntgenbildes) wird durch entsprechende Funktionen, d. h. durch die Operationsbezeichner, modelliert.

Für die Fälle, in denen verschiedenen Datensichten von Funktionen benötigt werden, müssen die notwendigen Datenklassen definiert werden. Hier stellt sich wie auch bei den Operationsbezeichnern die Frage nach der Feinheit der Datenklassen und ob die Systematik eines bereits existierenden Vokabulars für Datenklassen-Bezeichner auf die benötigte Datenklassenstruktur abgebildet werden kann. Diese Frage kann nur für jedes einzelne Vokabular beantwortet werden, und wird im Rahmen dieser Arbeit nicht umfassend untersucht. Die Klassifikation der Daten sollte alle Abstraktionsebenen enthalten, die auch einfache Definitionen von Datensichten ermöglichen. Abbildung 5.2 zeigt eine beispielhafte Gliederung der Daten, die aus den Empfehlungen der GMDS-AG „Datenschutz und Datensicherheit in Gesundheits-Informationssystemen“ abgeleitet wurde. Die Datenstrukturen für Datensichten wurden so flexibel model-

[5] siehe auch Abschnitt 5.3.2

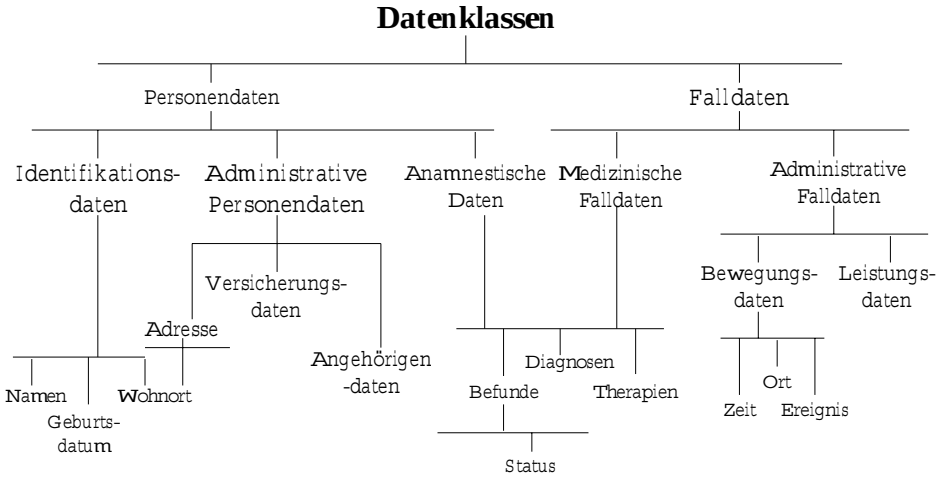

Abbildung 5.2: Datenklassen-Hierarchie

liert, dass nicht nur das Vokabular sondern auch das Format, in dem die Datensicht kommuniziert wird, hinsichtlich des Aufwands bei der Definition und der Auswertung optimiert werden kann. Das folgende Beispiel veranschaulicht die Schritte des Definitionsvorgangs:

Beispiel: *„Patienteninformationen aus der Referenzdatenbank“*

Die Referenzdatenank enthält Personen- und Falldaten aller Patienten aus den Jahren 1995 bis heute. Die Personendaten umfassen Identifikationsdaten, Patientenadresse und Versicherungsdaten. Zum Fall existieren Bewegungsdaten, verschiedene Leistungsdaten und abrechnungsrelevante Diagnosen. Lesezugriff auf diese Daten wird durch eine Applikation ermöglicht, die als Funktionen die Ansicht der Personendaten, der Falldaten zusammen mit fall-übergreifenden Diagnosen und der einzelnen Aufenthalte eines Falls zusammen mit den zugehörigen Diagnosen anbietet.

Die Berechtigung zu diesen Funktionen benötigt folgende Operationsbezeichner:
`OP.Information.Personendaten_abfragen`
`OP.Information.Bewegungsdaten_abfragen`

Für die weitere Differenzierung kann alternativ ein zusätzlicher Operationsbezeichner `OP.Information.Diagnosen_abfragen` oder entsprechend differenzierende Datensichten definiert werden.

Die Datensichten erfordern eine Klassifizierung der Daten in:

```
DATA.UKMZIntern.Falldaten.Bewegungen
DATA.UKMZIntern.Falldaten.MedDaten.Diagnosen
```

In der Schreibweise regulärer Ausdrücke (→ 4.2.4) auf der Basis des oben angegebenen Vokabulars , sehen die Definitions-Strings für die oben genannten Fälle (mit verkürzten Bezeichnern) folgendermaßen aus:

```
view.name.term='FallView01.Adm01'
view.definition.rep='Falldaten.Bewegungen'

view.name.term='FallView01'
view.definition.rep='Falldaten.(Bewegungen|MedDaten.Diagnosen)'
```

Datensichten stellen immer Teilhierarchien der Datenklassen-Hierarchie dar. Auch Datensichten untereinander können überlappen und vollständig ineinander eingeschlossen sein. Bei der Registrierung der Datensicht-Namen sollte darauf geachtetet werden, dass eine Datensicht, die vollständig in einer anderen eingeschlossen ist, auch in der Hierarchie unter der umschließenden Datensicht angelegt wird. Im vorangestellten Beispiel ist die Datensicht `'FallView01.Adm01'` in der Datensicht `'FallView01'` enthalten.

Kontextregeln

In Abschnitt 2.1.7 wurden Regeln für die dynamischen Zusammenhänge zwischen Behandlungen, Organisationseinheiten und daraus abgeleiteten Zugriffsberechtigungen formuliert. Im folgenden Abschnitt wird gezeigt, wie diese Regeln in Kontextregeln umgesetzt werden können, so dass sie als Kontextprototypen für die Definition der Rechte zur Verfügung stehen.

Regel 1: „Abteilungskontext“ Der Abteilungskontext wird durch eine „ForAll“-Regel mit einem Parameter ($1) vom Typ „UNIT“ ausgedrückt. Für alle Behandlungsphasen gilt folgende Bedingung:

1. `elements[i].location == $1`

Regel 2: „Fallkontext“ Der Fallkontext wird durch eine „ForAllExist“-Regel mit einem Parameter vom Typ „UNIT“ ($1) und einem zweiten ganzzahligen Parameter ($2), der eine Zeitspanne in Tagen festlegt, ausgedrückt. Der Zeitparameter gibt die Anzahl der Tage nach Beendigung der Phase an, in denen weiterhin (z. B. zur Arztbriefschreibung) Zugriff auf die Akte gewährt wird. Die Bedingungen für jedes Indikatorelement „`indElement`“ lauten:

1. `indElements[i].location == $1`
2. `indElements[i].end + $2` $\leq$ `NOW`

Die Beziehung zwischen den Indikatorelementen und den gesuchten Elementen ist durch folgende Bedingung festgelegt [6]:

1. `indElements[i].med_id == elements[j].med_id`
2. `indElements[i].begin` $\geq$ `elements[j].begin`

Für die gesuchten Behandlungselemente sind keine weiteren Bedingungen formuliert.

Regel 3: „Beteiligungskontext“ Der Beteiligungskontext wird ebenfalls durch eine „ForAllExist“-Regel mit zwei Parametern (Typen „UNIT“ und „INT“ wie in Regel 2) ausgedrückt. Ein Beteiligungskontext zur Formulierung spezifischer Beteiligungen (z. B. Erbringung von Laborleistungen) kann durch einen weiteren Parameter vom Type „PART_ROLE“ verfeinert werden. Im unspezifischen Fall gelten für jedes Indikatorelement „`indElements[i]`“ und eine Beteiligung„ `participants[j]`“ die Bedingungen 1. und 2. im spezifischen Fall gilt zusätzlich die Bedingung 3.:

1. `indElements[i].participants[j].unit == $1`
2. `indElements[i].end + $2` $\leq$ `NOW`
3. `indElements[i].participants[j].role == $2`

Die Beziehung zwischen Indikatorelementen und gesuchten Elementen ist analog zu Regel 2.

Regel 4: „ALL-Kontext“ Der ALL-Kontext enthält keine Bedingungen, er schließt alle Behandlungsphasen aller Patienten ein.

Regel 5: „Update-Kontext“ Der Update-Kontext wird durch eine „ForAllExist“-Regel mit einem Parameter vom Type „UNIT“ ausgedrückt. Hierbei sollte jedoch zwischen einem „FallUpdate“-Kontext und einem „PersonUpdate“-Kontext unterschieden werden.[7] Die Bedingungen für jedes Indikatorelement „`indElement`“ lauten in beiden Fällen folgendermaßen:

1. `indElements[i].location == $1`

Für die gesuchten Elemente des „FallUpdate“-Kontexts gilt:

1. `indElements[i].med_id=elements[j].med_id`

[6] Es genügt das Vorhandensein einer(!) solchen Beziehung für jedes gesuchte Element.

[7] Beim „passiven“ Zugriff können aufgrund der Ereignis-Systematik fallunabhängige Personendaten auch im Zusammenhang mit Fällen aktualisiert werden, auf die das Empfängersystem keinen Zugriff erhält. Die notwendige Beschränkung wird durch die Wahl der Aktualisierungsoperationen oder durch Datensichten sichergestellt.

2. `indElements[i].begin` $\leq$ `elements[j].begin`

Für die gesuchten Elemente des „PersonUpdate“-Kontexts gilt:

1. `indElements[i].patient.pid=elements[j].patient.pid`
2. `indElements[i].begin` $\geq$ `elements[j].begin`

Auf die Angabe einer beispielhaften Übersetzung der formalen Regeldefinitionen in die korrespondierenden Datenstrukturen wird an dieser Stelle verzichtet.

Zur Weiterführung des Beispiels „Patienteninformationen aus der Referenzdatenbank“ soll im folgenden „Regel 2“ durch „FallKontext(UNIT \$1, INT \$2)“ repräsentiert werden. Die Zugriffe werden im Beispiel ausschließlich durch die Operationsbezeichner differenziert.

Fortsetzung Beispiel: *„Patienteninformationen aus der Referenzdatenbank“*

Für den Zugriff auf die Daten der Referenzdatenbank soll folgendes Recht vergeben werden:

Lesezugriff auf Personen-, Bewegungs- und medizinische Daten der aktuellen Fälle aller Patienten, die sich in zum jeweiligen Zeitpunkt in Behandlung der Station „KI123“ befinden. Der Zugriff soll auch noch innerhalb von 7 Tagen nach Entlassung oder Verlegung des Patienten möglich sein.

Dafür müssen folgende drei „UserPermission“-Objekte up_1, .., up_3 erzeugt werden:

```
up1.name = 'Refdb0001'
up1.operation.term = 'Information.Personendaten_abfragen'
up1.operation.vocabulary = 'UKMZIntern'
up1.viewName.term = '*'
up1.viewName.vocabulary = '*'
up1.context.name.term = 'FallKontext'
up1.context.name.vocabulary = 'UKMZIntern'
up1.context.parms[0].key = 'UNIT'
up1.context.parms[0].value = 'KI123'
up1.context.parms[1].key = 'INT'
up1.context.parms[1].value = '7'
up1.source.term = 'GLOBAL'
up1.source.vocabulary = 'UKMZIntern'
up1.conditions[0].key = 'LOG_OBJECT'
up1.conditions[0].value = 'DATA'
```

```
up1.conditions[1].key = 'LOG_LEVEL'
up1.conditions[1].value = 'SAMPLE'

up2.name = 'Refdb0002'
up2.operation.term = 'Information.Bewegungsdaten_abfragen'
up2.operation.vocabulary = 'UKMZIntern'
...

up3.name = 'Refdb0003'
up3.operation.term = 'Information.Diagnosen_abfragen'
up3.operation.vocabulary = 'UKMZIntern'
...
```

Das Ursprungssystem

Mit der Angabe eines Systembezeichners („`source`“) werden die Rechte für Zugriffe auf ein spezifisches System definiert. Diese Einschränkung ermöglicht die Berücksichtigung von Systemen, deren Berechtigungsstruktur nicht in ein allgemeines Konzept integriert werden kann. Grundsätzlich sollten Rechte jedoch so allgemein formuliert werden, dass sie von allen Systemen gleichermaßen interpretiert werden können, da eine einheitliche Zugriffspolitik angestrebt wird. Die Einschränkungsmöglichkeit für spezifische Systeme sollte nur in Ausnahmefällen genutzt werden. Im allgemeinen Fall hat die `source`-Variable den Wert „`GLOBAL`“.

Zugriffsbedingungen

Die Zugriffsbedingungen („`conditions`“) ermöglichen die Kopplung von „Auflagen“ an ein Recht. Im Moment wird diese Möglichkeit genutzt, um die Art der erforderlichen Protokollierung festzulegen, die sich für die verschiedenen denkbaren Zugriffe unterscheidet. Dabei wird zwischen den zu protokollierenden Objekten („`LOG_OBJECT`“) und dem Umfang der Protokollierung („`LOG_LEVEL`“) unterschieden. Für Schreibzugriffe ist immer eine vollständige Protokollierung der Zustandsänderung erforderlich (`(LOG_OBJECT, TRANSITION)` und `(LOG_ LEVEL, ALL)`). Für autorisierte Lesezugriffe wird nur eine stichprobenhafte Protokollierung verlangt (`LOG_OBJECT, DATA`) und (`LOG_LEVEL, SAMPLE)`), bei Lesezugriffen im Notfallmodus (ohne Einschränkung) müssen hingegen die betroffenen Datenklassen im wesentlichen nachvollziehbar sein ((`LOG_OBJECT, DATA`) und (`LOG_LEVEL, ALL`)).

Administratorrechte

Die Definition von differenzierten Administratorrechten ermöglicht eine Delegation von Verantwortung bei der Definition der Zugriffspolitik an die eigenständigen Einrichtungen des Klinkums. Das momentane Konzept sieht eine weitgehend zentrale Definition der Rechte, Rechtebausteine und der Basisrollen vor. Die lokale Definition von Rechten ist nur explizit für Zugriffe auf lokale Systeme gedacht. Die Ableitung spezifischer Rollen und die Autorisierung der Benutzer soll hingegen dezentral durchgeführt werden.

Operationen, die in den Administratorrechten referenziert werden können, sind durch die Wissenserfassungsfunktionen des PolicyAdmin-, RoleHierarchy-, ContextEvaluator- und Authorization-Objekts festgelegt. Die Datenbereiche werden durch die Klassifikationsparameter definiert. Das folgende Beispiel veranschaulicht die Definition der Administratorrechte.

Beispiel: *„Administration der Zugriffspolitik einer Abteilung“*

Die Verwaltung der Zugriffspolitik für den Bereich der Med. Notaufnahme (MN) soll u. a. folgendes Recht beinhalten, wobei vorausgesetzt wird, dass eine abstrakte bereichspezifische Arzt-Rolle „arztMN001“ gibt, von der alle weiteren ärztlichen“ Rollen des Bereichs abgeleitet werden, und dass der Qualifikationsbezeichner „Arzt“ Subjekte als Angehörige des ärztlichen Personals kennzeichnet:
Dann werden für die Zuweisung und Entziehung der klinkspezifischen Ärzterollen folgende „AdminPermission“-Objekte ap_1 und ap_2 benötigt:

```
ap1.name = 'Adm0001'
ap1.operation.term = 'Aut.AssignRole'
ap1.operation.vocabulary = 'UKMZIntern'
ap1.dataClass[0].key = 'ROLE_UNIT'
ap1.dataClass[0].value = 'MN'
ap1.dataClass[1].key = 'JUNIORROLE_ID'
ap1.dataClass[1].value = 'arztMN001'
ap1.dataClass[2].key = 'SUBJECT_UNIT'
ap1.dataClass[2].value = 'MN'
ap1.dataClass[3].key = 'SUBJECT_TYPE'
ap1.dataClass[3].value = 'PERSON'
ap1.dataClass[4].key = 'SUBJECT_QUAL'
ap1.dataClass[4].value = 'Arzt'

ap1.name = 'Adm0002'
ap2.operation.term = 'Aut.RemoveAssignment'
ap2.operation.vocabulary = 'UKMZIntern'
...
```

5.2.2 Rollen und Rollenhierarchie

Für die Rollenhierarchie wird eine Gliederung entsprechend der hierarchischen Anordnung der Organisationseinheiten und der verschiedenen funktionalen Personal- und Systemgruppen empfohlen. Wenigstens für jede wesentliche „Subjektgruppe“ und für jede Fachklinik bzw. zentrale Einrichtung sollte eine „Basisrolle“ angelegt werden. Die Basisrollen können – müssen aber noch keine – Rechte besitzen. Auf jeder Hierarchieebene können untrennbare Rechtemengen zu „funktionalen Rollen“ zusammengefasst und als abstrakte Bausteine „konkreter Rollen“ verwendet werden. Jede konkrete Rolle erbt von mindestens einer Einrichtungs- und einer Funktions-Rolle. Abbildung 5.3 zeigt eine derartige Anordnung von Einrichtungs- und Funktionsrollen. Das Beispiel der Abb. 5.3

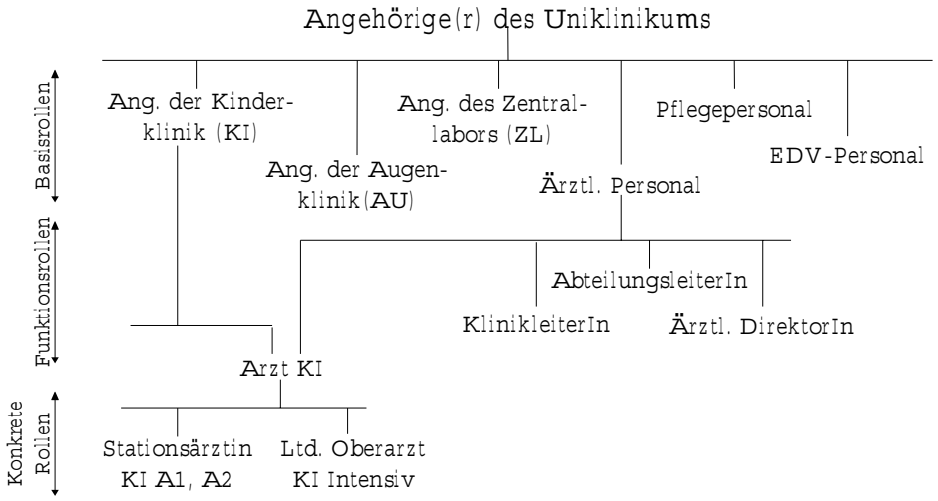

Abbildung 5.3: Beispiel-Rollenhierarchie

beinhaltet folgende Annahmen über die Berechtigungsbedürfnisse: Von der Basisrolle „Ärztliches Personal“ wurden funktionale Rollen „Ärztl. DirektorIn“, „KlinikleiterIn“ und „AbteilungsleiterIn“ abgeleitet, von denen angenommen wird, dass sie auch besondere Berechtigungen über den Bereich ihrer Fachklinik oder Abteilung hinaus besitzen. Von den Basisrollen „Angehöriger der Kinderklinik“ und „Ärztliches Personal“ wurde eine grundlegende Rolle „Arzt KI“ abgeleitet. Diese Rolle kann Rechte besitzen, die alle Ärzte der Kinderklinik gleichermaßen benötigen. Sie ist außerdem Baustein (Junior-Rolle) für alle weiteren ärztlichen Rollen der Kinderklinik. Durch die zentrale Definition solcher „Rollen-Bausteine“ und die Autorisierung lokaler Administratoren zur Ableitung weiterer spezifischerer Rollen kann die Delegation von Rechte-Zuweisungen realisiert werden.

Zusätzlich zu den Benutzerrollen benötigt jede Fachklinik eine Administrator-Rolle eventuell für jede eigenständige Abteilung. Diese beinhal-

tet insbesondere die Rechte zur Zuweisung der bereichsspezifischen Rollen an die Angehörigen der Fachklinik und die Ableitung spezifischer Rollen für den jeweiligen Bereich. Die Administrator-Rollen der Abteilungen können je nach Bedarf vom Administrator der jeweiligen Klinik erstellt werden.

Für die Modellierung spezieller Rollen können hier keine Empfehlungen gegeben werden. Die benötigten Zugriffsberechtigungen müssen für alle Bereiche und Systeme ermittelt und sinnvoll zusammengefasst werden. Systeme, die verschiedene Benutzerarten unterscheiden, aber keine extern definierten Rechte verarbeiten, können bei der Modellierung der Rollenhierarchie berücksichtigt werden. Die Benutzergruppen werden entweder als separate Rollen ohne explizite Rechte angelegt, oder bereits definierte Rollen werden den entsprechenden Benutzern zugewiesen, sofern sie den Anforderungen entsprechen. Für eine sehr grobe Unterscheidung der Benutzer verschiedener Kliniken könnte eine Rolle in der Art der „Arzt KI“-Rolle des obigen Beispiels verwendet werden. Für jede vollständige Funktion, in der ein Subjekt tätig ist, sollte – wenn möglich – nur eine Rolle definiert werden, so dass der Benutzer innerhalb einer Tätigkeit, z. B. der ärztlichen Tätigkeit auf einer Station, nicht zum Rollenwechsel gezwungen ist. Bei einem unvermeidbaren Rollenwechsel sollte keine erneute Authentisierung erzwungen werden (siehe auch Abschnitt 5.4 zum Single-Sign-On-Verfahren).

Rollenconstraints

Rollenconstraints sollen die Einhaltung spezifischer Regeln bei der Modellierung der Rollenhierarchie und der Zuweisung der Subjekte sicherstellen. Folgende Rollenconstraints werden empfohlen:

- Für jede Kombination von Administratorrolle und Benutzerrolle existiert ein schwacher Ausschluss, d. h. einerseits, dass kein Subjekt gleichzeitig eine Administratorrolle und eine Benutzerrolle aktivieren kann, und andererseits, dass Benutzerrollen und Administratorrollen nicht voneinder abgeleitet werden dürfen. Grund für diesen Ausschluss ist ein Sicherheitsgrundprinzip, das insbesondere die gezielte Manipulationen der eigenen Rollen durch eine Person erschweren soll.

- Mit Hilfe von Prerequsite-Constraints kann sichergestellt werden, dass Rollen nur bestimmten Gruppen von Subjekten zugewiesen werden. Insbesondere sollten die Basisrollen der Kliniken und Berufsgruppen als Voraussetzung für die Zuweisung jeder spezifischeren Rolle gesetzt werden. Diese vorausgesetzten Rollen werden selten und sehr allgemein für jedes Subjekt von zentraler Stelle (z. B. in der Personalverwaltung) vergeben. Die spezifischere Zuordnung findet dann je nach Bedarf in den einzelnen Kliniken und Instituten statt. Mit einer geeigneten Rollenstruktur kann auf diese Weise auch die Kopplung an eine automatische Dienstplanerstellung unterstützt werden. Jede Person würde dabei entsprechend ihrer Einsatzmöglichkeiten eine oder mehrere „rechtefreie Voraussetzungsrollen“ erhalten. Der Dienstplanersteller erhält so Information über die

Personalressourcen. Bei der Einteilung der Personen wird dann erst die „wirksame Rolle“ für den erforderlichen Zeitraum zugewiesen.

Starker wechselseitiger Ausschluss von zwei Rollen ist dort notwendig, wo unter allen Umständen vermieden werden soll, dass eine Person beide Rollen (auch nacheinander) aktivieren kann. Ob es Anwendung für solche Restriktionen gibt, muss im Einzelfall ermittelt werden, und wird hier nicht weiter vertieft.

5.2.3 Abfragen

Abgesehen von der Ticketerzeugung gibt es für die Abfrage der Berechtigungsinformation zwei grundsätzliche Strategien: der Zugriffskontroll-Klient erfragt eine Zugriffsentscheidung zu einem ausreichend spezifizierten Zugriff, oder er erfragt eine mehr oder weniger eingeschränkte Menge von Rechten. Die Abfrageparameter und ihre Interpretation wurden in den Abschnitten 4.2.2 und 4.4 bereits umfassend erläutert. Die Mengen-Strategie ist sinnvoll, wenn die ermittelte Information zur weiteren Benutzerführung verwendet werden soll, die Programmlogik des Klienten keine Abfrage zu einzelnen Zugriffen erlaubt, oder die Performanz des Klienten durch die Häufigkeit der Anfragen wesentlich beeinträchtigt würde. Es ist wichtig darauf zu achten, dass keine unnötigen oder unnötig aufwendigen Anfragen implementiert werden. Die folgenden Beispiele illustrieren das Zusammenwirken von Rechtedefinitionen und Abfragen.

Beispiel 1: *„Abfrage einer Rechtemenge für den Zugriff auf Befunddaten“*
Ein typischer Befundserver bietet genau eine Funktion: das Lesen der Befunddaten. Für jede Organisationseinheit „OE“, deren Angehörige Befunddaten abfragen dürfen, wird deshalb genau ein UserPermission-Objekt mit folgendem Inhalt definiert und den jeweiligen Rollen zugewiesen:

```
up.name = 'BefundeOE001'
up.operation.term = 'Information.Befunde_abfragen'
up.operation.vocabulary = 'UKMZIntern'
up.viewName.term = '*'
up.viewName.vocabulary = 'UKMZIntern'
up.context.name.term = 'FallKontext'
up.context.name.vocabulary = 'UKMZIntern'
up.context.parms[0].key = 'UNIT'
up.context.parms[0].value = 'OE'
up.context.parms[1].key = 'INT'
up.context.parms[1].value = '7'
up.source.term = 'GLOBAL'
up.source.vocabulary = 'UKMZIntern'
up.conditions[0].key = 'LOG_OBJECT'
up.conditions[0].value = 'DATA'
```

```
up.conditions[1].key = 'LOG_LEVEL'
up.conditions[1].value = 'SAMPLE'
```

Eine Abfrage (`MasterAccessControl::getPermissions()`) mit folgender „PermissionRequest"-Struktur „`preq`" liefert für alle Rollen mit dem Recht „BefundeOE001" die Menge der aktuellen Behandlungphasen:

```
preq.operation.term = 'Information.Befunde_abfragen'
preq.operation.vocabulary = 'UKMZIntern'
preq.viewName.term = '*'
preq.viewName.vocabulary = 'UKMZIntern'
preq.sparms[0].key = 'CURRENT'
preq.sparms[0].value = 'TRUE'
preq.source.term = 'GLOBAL'
preq.source.vocabulary = 'UKMZIntern'
```

Da Operation und Datenausschnitt für alle Zugriffe identisch ist, müssen nur die Fälle unterschieden werden, auf die der Zugriff gestattet wird. Sie können mit `MasterAccessControl::getAdminRecords()` abgerufen werden. Das Befund-Informationssystem kann mit Hilfe der Fall-Listen dem Benutzer gezielt die Fälle anbieten, auf die er zugreifen darf.

Nach Mengenabfragen wie in Beispiel 1 müssen zunächst nicht zwangsläufig weitere Abfragen gestellt werden. Alle benötigten Informationen können auf einmal ermittelt werden. Bei längeren Sitzungen ist es trotzdem notwendig, die Abfrage in regelmäßigen Abständen zu wiederholen, da das Kontextwissen einer kontinuierlichen, schnellen Veränderung unterliegt. Wichtig bei solchen Abfragen ist außerdem die Einschränkung der Menge durch einen Zeitraum oder – wie in Beispiel 1 – auf die aktuellen Fälle. Andernfalls liefert der Kontextserver Mengen mit vierstelligen Elementzahlen. Diese Gefahr besteht – wie das folgende Beipiel veranschaulicht – bei der Abfrage von konkreten Zugriffsentscheidungen nicht, welche aber dafür für jeden Zugriff einzeln abgerufen werden müssen.

Beispiel 2: *„Abfrage einer Zugriffsentscheidung"*

Mit dem Szenario des Befundservers aus Beispiel 1 würde eine Zugriffsentscheidung für den Zugriff auf die Befunde des Patienten mit der Patienten-ID '12345' bezüglich des Falls '00734578' auf folgende Weise ermittelt werden:

Die Abfrage (`MasterAccessControl::getPermissions()`) wird mit der „PermissionRequest“-Struktur „`preq`“, die die Werte für den Patienten und den Fall enthält, gestartet. Wird innerhalb des Falls zwischen Befunden aus verschiedenen Phasen unterschieden, so kann die Struktur mit den Such-Parametern „START“ und „END“ noch um ein eingrenzendes Zeitintervall erweitert werden.

```
preq.operation.term = 'Information.Befunde_abfragen'
preq.operation.vocabulary = 'UKMZIntern'
preq.viewName.term = '*'
preq.viewName.vocabulary = 'UKMZIntern'
preq.sparms[0].key = 'PID'
preq.sparms[0].value = '12345'
preq.sparms[1].key = 'ADM_ID'
preq.sparms[1].value = '0734578'
preq.source.term = 'GLOBAL'
preq.source.vocabulary = 'UKMZIntern'
```

Existiert die Berechtigung zum Zugriff auf den angegebenen Fall, so liefert die Funktion einen Rückgabewert größer null. Wenn das System keine zusätzliche Information z. B. über Protokollierungsverfahren benötigt, müssen die ermittelten Rechte nicht explizit abgefragt und ausgewertet werden. Als Zugriffsentscheidungskriterium genügt die Fallunterscheidung bezüglich der Anzahl gefundener Rechte (0, > 0).

5.3 Integration

Die Integration der Zugriffskontroll-Funktionalität in klinische Informationssysteme kann auf vielfältige Weise geschehen. Der Umfang und die Art der benötigten Berechtigungsinformation ist sehr verschieden. Abhängig davon müssen Abfragen während oder nach der Anmeldung und/oder innerhalb verschiedener Aktionen gestartet werden. Zwei grundlegende Verfahren der Integration sind zu unterscheiden:

1. Die AccessControl-Client-Funktionaltität wird in den Server des Informationssystems eingebunden. Der Informationssystem-Client erfasst alle notwendigen Benutzerdaten (Zertifikat, evtl. Rolle), und nutzt – wann immer notwendig – die AccessControl-Client-Funktionen zur Abfrage der aktuellen Rechte des angemeldeten Benutzers (Abb. 5.4). Die Parameter der Abfrage sind teils durch Spezifika des Systems festgelegt, teils durch den Funktionskontext, in den sie eingebettet sind, und teils durch benutzergesteuerte (Navigations-)Parameter. Mit diesem Verfahren können sehr differenzierte Zugriffsentscheidungen getroffen werden.

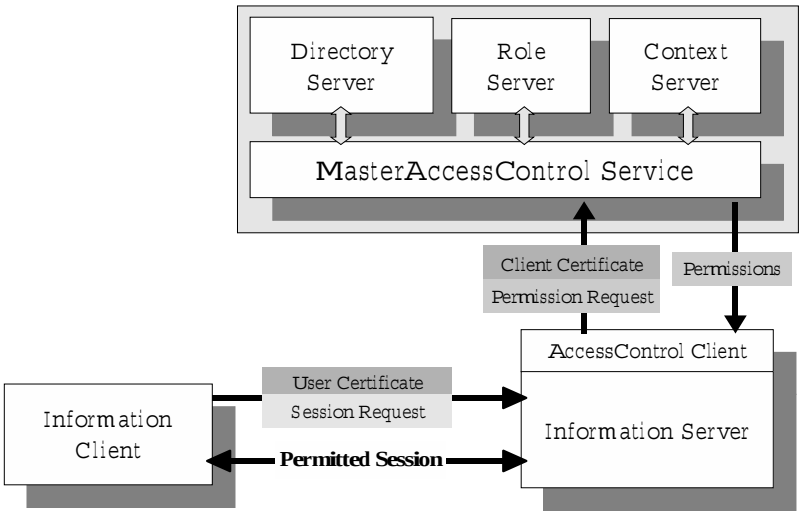

Abbildung 5.4: Integration des Zugriffskontroll-Service in das Informationssystem

2. Der Benutzer-Client des Informationssystems kann beim Anmeldedialog Berechtigungsinformation für den aktuellen Benutzer in Form des Benutzertickets entgegennehmen, und übermittelt dieses zur Auswertung an den Informations-Server (Abb. 5.5). Das Ticket kann dabei entweder vom Benutzer-Client direkt beim MasterAccessControl-Server abfragt werden. Dazu muss der Client die AccessControl-Client-Bibliothek integrieren. Oder das Ticket wird von einem unabhängigen vorgeschalteten Programm abgerufen, das die AccessControl-Client-Funktionalität beinhaltet, und im Dateisystem des Arbeitsplatzrechners deponiert, von wo es für verschiedene Benutzer-Clients zur Verfügung steht. Die Berechtigungsinformation des Tickets ist zugunsten einer Größenbegrenzung allgemein gehalten und deshalb nicht für jedes System geeignet (→ 4.2.5). Es eignet sich insbesondere für Systeme, deren Benutzer alle auf sämtliche darin verfügbaren Patientenakten – auch in unterschiedlichen Funktionen – zugreifen dürfen.

Im folgenden werden technische Aspekte der Integration, Systemvoraussetzungen und inhaltliche Aspekte teilweise an Hand von Beispielen beschrieben.

5.3.1 Technische Aspekte

Die Funktionalität des MasterAccesControl-Service wird über die AccessControl-Client-Bibliothek zur Verfügung gestellt. Die Bibliothek verfügt über eine C++- und C-Schnittstelle. Folgende Schritte müssen bei der Integration in eine Systemkomponente durchlaufen werden.

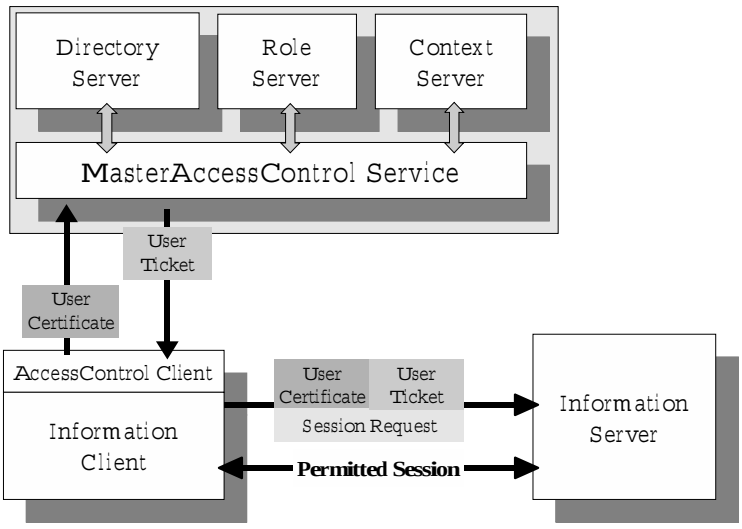

Abbildung 5.5: Integration der Zugriffinformation mittels Zugriffsticket

1. Auf dem (Entwicklungs-)Rechner müssen ORBacus-4.0.4, FSSL-2.0.1 und OpenSSL-0.9.6 (oder neuere Versionen) installiert werden, da die AccessControl-Client-Bibliothek auf Bibliotheken dieser Softwarepakete zugreift.

2. Verantwortliche des Klinkums müssen mit dem Systemhersteller alle Werte, die in der Berechtigungsabfrage verwendet werden sollen, verbindlich vereinbaren. Das betrifft insbesondere die Begriffe des Vokabulars sowie Formate und Suchparameter. Es empfielt sich, die Parameter als konfigurierbare Werte in die Informationssysteme zu integrieren, da damit leichtere Anpassungs- oder Erweiterungsmöglichkeiten bestehen.

3. Sofern das System keine „User-Exits“ vorsieht, die den Aufruf externer Funktionen ermöglichen, muss der Quellcode der Komponente um die konkreten Berechtigungsprüfungen erweitert und neu kompiliert werden. Die Komponente wird dann neu erstellt. Bei kommerziellen Systemen ist dieser Vorgang in der Regel nur durch den Hersteller möglich.

Für eine Einbindung der Funktionalität in Skripte – z. B. Perl, PHP – können vorgefertigte ausführbare Programme verwendet werden; die Parameterübergabe erfolgt dann über Standard-Input/Output. Die Erweiterung von Skripten ist i. a. mit geringem Aufwand verbunden.

5.3.2 Beispiele

An Hand der folgenden Beispiele verschiedener System-Szenarien des Mainzer Universitätsklinikums werden die inhaltlichen und systemspezifischen Aspekte der Integration beleuchtet, und dabei die Möglichkeiten und Grenzen des Konzepts demonstriert.

Der Kommunikationsserver

Der Kommunikationsserver vermittelt nachrichten-basiert Informationen über administrative Vorgänge, Diagnosen und Befunde an Subsysteme des verteilten Klinikinformationssystems. Dazu werden die Nachrichten, die der Kommunikationsserver von den Quell-Systemen empfängt, identifiziert, analysiert, in das Zielformat übersetzt und an die Zielsysteme weitergeleitet. Für die Frage, welche Daten an welche Systeme weitergeleitet werden dürfen, muss der Kommunikationsserver dynamische Berechtigungsinformation einbeziehen. Die Software des DataGate-Servers, der im Mainzer Uniklinikum betrieben wird, erlaubt innerhalb der Identifikations- und Übersetzungs-Routinen Aufrufe externer Funktionen, die in einen vorgegebenen RPC-Server integriert werden. Diese Möglichkeit wird für die Integration der Berechtigungsprüfungen genutzt. Dazu müssen folgende Arbeitsschritte durchgeführt werden:

1. Einmalige Installation der Bibliotheken auf dem HP-Server-Cluster.
2. Einmalige Implementierung der Berechtigungsabfrage in eine externe Prozedur[8]: Die „getPermissions()“-Funktion der AccessControl-Client-Bibliothek wird in der Prozedur aufgerufen. Parameter des Aufrufs sind die Subject-ID des Zielsystems, die entsprechende Rolle für das Zielsystem sowie die „PermissionRequest“-Struktur preq, für die folgendes gilt:
   - `preq.*.vocabulary = '*'`.
   - `preq.view.term = '*'`. Die Auswahl der Datenfelder wird in den Übersetzungstabellen des Kommunikations-Servers zusammen mit den Formatkonvertierungen definiert und muss hier nicht berücksichtigt werden. Prinzipiell können aber korrespondierende Datensichten definiert und in den Rechten referenziert werden. Diese Datensichten werden nicht ausgewertet und können deshalb leere Definitionswerte haben. Sie können eingesetzt werden um die Berechtigungen zum Empfang verschiedener Ausschnitte eines Nachrichtentyps zu differenzieren. `preq.view.term` enthält dann den korrespondierenden Datensicht-Bezeichner zur Übersetzungstabelle. Das gleiche Ergebnis erreicht man durch entsprechende Differenzierung der Operationsbezeichner.
   - In `preq.operation.term` wird der Operationsbezeichner übergeben, der die passende Operation zum Nachrichtentyp bzw. zur Art der

[8] Zur Implementation und Funktionsweise der externen Prozeduren siehe „Monk Programmer's Reference Guide“, DataGate 3.5 [11]

übertragenen Daten angibt, z. B. „Personendaten_empfangen“, „Falldaten_empfangen“ und „Laborbefunde_empfangen“.

- Je nach Nachrichtentyp enthalten die Suchparameter die Patienten- oder Fallnummer (`preq.sparms[i].key = 'PATIENT_ID' bzw. preq.sparms[j].key = 'ADMISSION_ID'`)
- In `preq.source.term` wird die Subjekt-ID des sendenden Systems oder der Wert „GLOBAL“ übergeben.

Für zulässige Datenübertragungen liefert die Funktion einen Rückgabewert größer Null. Ein nachrichtenspezifischer Wert (hier Patienten- oder Fallnummer) kann im „Input“-Parameter an die externe Prozedur übergeben werden. Alle anderen verbindungsspezifischen Parameter werden bei der Konfiguration der Nachrichtenidentifikation im „Flags“-Parameter der externen Prozedur festgelegt. Er hätte in diesem Fall beispielsweise die Form: '<*SubjectID*> | <*RoleID*> | <*View*> | <*Operation*> | <*Quellsystem*>'. Die externe Prozedur liefert den Rückgabewert „TRUE“ oder „FALSE“, der die Nachrichten-Identifikation erfolgreich sein oder scheitern lässt. Scheitert die Identifikation, so werden keine Daten übertragen.

3. Registrierung aller Zielsysteme als Subjekte und Definition der Rollen und Rechte für den Datenempfang.

Das Intensivpflege-System der Kinderklinik (fiktive Subjekt-ID „KI_CareVue“) sei Inhaber einer Rolle mit dem Recht „DGISH2KICV_001“, das mit folgenden Einträgen definiert ist, und dazu berechtigt, Falldaten vom Patientenverwaltungs-System zu empfangen:

```
up.name.term = 'DGISH2KICV_001'
up.viewName.term = '*'
up.viewName.vocabulary = '*'
up.operation.term = 'Information.Falldaten_empfangen'
up.operation.vocabulary = 'UKMZIntern'
up.context.name.term = 'FallUpdate'
up.context.name.vocabulary = 'UKMZIntern'
up.context.parms[0].key = 'UNIT'
up.context.parms[0].value = 'KLINIKUM.KI.KI11'
up.source.term = 'SAP_ISH'
up.source.vocabulary = 'UKMZIntern'
```

4. Konfiguration der Nachrichten-Identifikations-Tabellen des Kommunikationsservers mit den systemspezifischen Parametern für jede Kombination Nachrichtentyp/Zielsystem, für die eine Übermittlung stattfinden könnte.

Für die oben genannte Beipiel erhielte die externe Prozedur folgende Parameter:
Input: Das Feld der Nachricht, das die Fallnummer enthält.
Flags: `'KI_CareVue|*|*|Falldaten_empfangen|SAP_ISH'`

Die Möglichkeiten der Zugriffskontrolle im Kommunikationsserver sind durch die Tatsache eingschränkt, dass an externe Prozeduren nur ein dynamischer Paramter übergeben werden kann, der außerdem in einem Nachrichtenfeld der empfangenen Nachricht enthalten sein muss. Trotzdem genügt diese Möglichkeit, da die Anforderungen des Datenversands nur eine Parametrisierung nach Patienten- oder Fallnummer verlangen. Auf die beschriebene Weise kann deshalb der gesamte Datenversand im Kommunikationsserver entsprechend den Datenschutzregelungen kontrolliert werden.

SAP/IS-H

Das Patientenverwaltungs-System SAP/IS-H verwendet ein komplexes SAP-internes Berechtigungskonzept [3]. Bei der Entwicklung von Systemfunktionalität (auch durch SAP-fremde Entwickler) können Berechtigungsobjekte definiert und Berechtigungsprüfungen bezüglich dieser Berechtigungsobjekte implementiert werden. Berechtigungsobjekte erhalten einen Namen und bis zu zehn Parameter, sogenannte Berechtigungsfelder. Diese sind in einem Data-Dictionary registriert und mit Systemfeldern assoziiert. Berechtigungsfelder enthalten Werte- oder Fremdschlüsselmengen in Form regulärer Ausdrücke, die z. B. Dokumente, Systemkomponenten oder Aktionen referenzieren. Durch geeignete Namensgebung in den Systemfeldern kann die Berechtigungsdefinition wesentlich vereinfacht werden. Berechtigungsobjekte sind entsprechend den Systembereichen in Objektklassen organisiert. Eine Berechtigung ist die Menge der Kombinationen zulässiger Werte aller Felder eines Berechtigungsobjekts. Die gesamte Berechtigungsinformation wird in Datenbanktabellen des Systems abgelegt. Die Berechtigungsprüfung findet mit Hilfe des ABAP/4-Befehls „AUTHORITY-CHECK“ statt, wobei der Name des Berechtigungsobjekts und die zu überprüfenden Werte übergeben werden. Mit Hilfe von Profilen[9] werden einem Benutzer Berechtigungs-Mengen zugewiesen. Zu diesem Zweck gibt es ein Werkzeug, den Profilgenerator, der die Berechtigungsdefinition auf komfortable Weise unterstützt. Der Administrator wählt einen groben Bereich aus und definiert darunter Aktivitätsgruppen, die bestimmten Systemfunktionen entsprechen. Zu den Aktivitäten präsentiert der Profilgenerator alle Berechtigungsobjekte, für deren Berechtigungsfelder Werte bestimmt werden müssen. Nur aus der vollständig bestimmten Aktivitätsgruppe erzeugt der Profilgenerator das Benutzerprofil, unvollständige Berechtigungen werden visuell besonders hervorgehoben. Außer den Funktionen zur Profilerstellung bietet das System

[9] ab Release 4.6C heißen Profile „Rollen“

auch Möglichkeiten, die Wirksamkeit der definierten Berechtigungen zu überprüfen.

Für den medizinischen Bereich ermöglicht dieses Konzept die statische Klassifikation der Berechtigungen nach Aktionen, Datenarten, Organisationseinheiten und beispielsweise nach Patienten- oder Fallnummern – sobald diese bekannt sind. Die regel-basierte Klassifikation von Berechtigungen z. B. nach dem „Fallkontext“ einer Organisationseinheit oder anderen Beteiligungen ist nicht möglich, obwohl Information über alle Patientenbewegungen sowie Leistungen im System enthalten ist.

Für die Definition und Kontrolle der statischen Berechtigungsanteile (Datenarten, Aktionen) bietet das Berechtigungsverfahren einen angemessenen Funktionsumfang. Stattdessen extern definierte Berechtigungsdefinitionen zu importieren, erscheint nicht sinnvoll, weil

1. die systemspezifische Definitionsunterstützung wegfallen würde,
2. der Aufwand für die Kopplung der internen Systemfeldbezeichner mit den externen Vokabularen sehr groß wäre und u. U. bei jedem Releasewechsel erneut anfallen würde,
3. der Aufwand für Übersetzungs- und Importfunktionen sehr groß wäre und u. U. bei jedem Releasewechsel erneut anfallen würde,
4. interne Prüfungen weniger effizient durchgeführt werden könnten,
5. und der Vorteil einer klinikweit einheitlichen zentral organisierten Berechtigungsdefinition die genannten Nachteile nicht aufwiegt.

Stattdessen wäre es wünschenswert in Kooperation mit SAP, das Berechtigungskonzept derart zu erweitern, dass auch dynamische Berechtigungsprüfungen ermöglicht werden. Ein denkbarer Weg wäre die Modellierung dynamischer Berechtigungsobjekte, die neben statischen auch dynamische Berechtigungsfelder besitzen können. Ein dynamisches Berechtigungsfeld wäre dann mit einer externen Prozedur assoziiert – vergleichbar den externen Prozeduren des Kommunikationsservers – die durch die AUTHORITY-CHECK-Funktion mit definierten Parametern aufgerufen werden müsste. Das Zusammenwirken von Rechten im Zugriffskontroll-System und Parametern der externen Prozedur könnte folgendermaßen aussehen:

- Für jede Kombination von Kontext und Organisationseinheit wird ein entsprechendes Recht definiert, das im Operationsfeld einen Referenznamen erhält.
- Alle so definierten Rechte werden einer speziellen Rolle „SAP_ISH“ zugeordnet.
- Die Referenznamen müssen in IS-H registriert und bei der Berechtigung als Werte dynamischer Berechtigungsfelder gesetzt werden können.

- Bei der Berechtigungsprüfung wird dann das Berechtigungsobjekt mit dem zu überprüfenden Wert, z. B. der aktuellen Fallnummer, übergeben. Die Prüfroutine stößt die externe Prozedur an, die die Zugriffskontrollfunktion enthält.

Beispiel für ein ISH-spezifisches „Referenz-Kontext-Recht“:

```
...
p.operation.term = 'Fallkontext_KI'
p.context.name = 'Fallkontext'
p.context.parms[0].key = 'UNIT'
p.context.parms[0].value = 'KI'
p.context.parms[1].key = 'INT'
p.context.parms[1].value = '7'
p.context.source.term = 'SAP_ISH'
...
```

In IS-H müsste dann ein dynamisches Berechtigungsfeld „FALL“ existieren, das den Wert „Fallkontext_KI“ erhält.

Der Aufruf bei der Überprüfung würde folgendermaßen lauten:

„`getPermissions(NULL, 'SAP_ISH', preq)`“, mit

`preq.viewName.term='*'`,
`preq.operation.term='Fallkontext_KI'`,
`preq.source.term='SAP_ISH'`,
`preq.sparms[0].key='ADMISSION_ID'` und
`preq.sparms[0].value` =<*aktuelle Fallnummer*>.

Im Einzelnen kann die Durchführbarkeit dieses Vorgehens nur mit genauer Kenntnis der SAP-Funktionen und -Datenstrukturen untersucht werden. Eine konkrete Realisierung kann insbesondere nur durch SAP erfolgen.

Eine Alternative ist die explizite Berücksichtigung der Behandlungszusammenhänge, die aus dem internen Datenbestand des Systems ermittelt werden können. Eine derartige Lösung wurde durch einen Kooperationspartner von SAP für ein klinisches Arbeitsplatz-System realisiert, das als integriertes Modul vollen Zugriff auf den IS-H-Datenbestand besitzt. Für den Zugriff auf klinische Dokumente wurde eine zusätzlich Berechtigungsprüfung unter Berücksichtigung der Patientenaufenthalte und der Zuständigkeit von Personen für definierte Organisationseinheiten implemetiert. Die implementierte Regel entspricht im wesentlichen dem hier beschriebenen Fallkontext. Ein Nachteil dieser Lösung ist, dass jede benötigte Regel als Prüfroutine fest implementiert wird, und von den

Verantwortlichen im Klinikum nicht kontrolliert oder verändert werden kann. Diese Art von Lösung ist deshalb weniger zu empfehlen.

Für den Bereich der Benutzerverwaltung bietet SAP ab dem Release 4.6C eine Auslagerung der Benutzerdaten in ein zentrales Benutzerverzeichnis an. Der Zugriff auf einen LDAP-Verzeichnisdienst wird – laut Dokumentation – unterstützt.

Der Befundserver des Zentrallabors

In der Klinik für klinische Chemie und Laboratoriumsmedizin wurde ein webbasierter Befundserver implementiert, der die Laborbefunde in den Klinken über Standard-Webbrowser zur Verfügung stellt. Zusätzlich zu den Laborbefunden sollen auch die Befunde des Instituts für Mikrobiologie und eventuell anderer Labore der Klinik über diesen Server veröffentlicht werden. Die Befunddaten liegen in einer relationalen Datenbank, die zugreifende Anwendung wurde mit HTML-Seiten und PHP-Skripten realisiert. Für die momentane Situation, in der noch keine zentrale Berechtigungsinformation zur Verfügung steht, wurde ein eigenes Berechtigungskonzept integriert. Beim Zugriff wird der Rechner mittels seiner IP-Adresse identifiziert. Auf der Basis der Routingtabellen der zentralen Netzadministration kann auf eine Klinik oder Abteilung zurückgeschlossen werden. Für jeden so festgelegten Bereich gibt es eine Zuordnung von Kliniken, auf deren Patienten von dem entsprechenden Netz aus zugegriffen werden darf. Es findet derzeit also insbesondere keine individuelle Authentisierung statt.

Für die Navigation im System wählt der Benutzer die Klinik und eine darunterliegende Organisationseinheiten aus. Er hat die Wahl zwischen einer Befundsuche nach „Auftragsdatum“ oder nach „Patientenname“. Angezeigt werden nur Befunde zu Fällen, innerhalb deren der Patient in der ausgewählten Organisationseinheit behandelt wurde. Information darüber ermittelt der Befundserver aus der Verlegungshistorie des Patienten, die durch den Datenversand des Verwaltungssystems via Kommunikationsserver erfasst und aktualisiert wird.

Abgesehen von der fehlenden Benutzerauthentisierung erfüllt dieses System die Datenschutzanforderungen und die Anforderungen der Nutzer zu einem großen Teil. Dabei gibt es aber noch folgenden Schwachstellen:

1. Ein Arzt kann die Befunde seiner Patienten nicht ausnahmsweise von einem anderen Netzbereich aus abrufen.

2. Ein konsiliarisch hinzugezogener Kollege aus einem anderen Bereich, kann keinen individuellen Zugriff auf die Befunde erhalten. Generell können keine Zugriffe erfolgen, die aufgrund anderer Zusammenhänge als der Verlegungshistorie gerechtfertigt sind.

3. Aber jeder, der bis zum Stationsrechner gelangt (Besucher?, Reinigungspersonal?, ...) oder seinen Rechner mit einer freien IP-Nummer eines berechtigten Netzes konfiguriert, könnte auf Befunde von Patienten zugreifen.

4. Es gibt keine Möglichkeit zur individuellen Protokollierung von Zugriffen (Wer hat sich wann an- und abgemeldet?).

Durch die künftige Nutzung der zentralen Zugriffskontroll-Komponenten könnten einige der Lücken gefüllt werden:

Individuelle Authentisierung Der Befundserver sollte eine individuelle Anmeldung erzwingen. Dazu muss kein eigenes Benutzerverzeichnis gepflegt werden, sondern es sollte das zentrale Benutzerverzeichnis des Klinikums genutzt werden. Die Authentisierung durch Benutzerzertifikate sollte, wenn möglich, unterstützt werden. Ansonsten kann die Login/Passwort-Kombination beim Benutzerverzeichnis überprüft werden. Bei der schwachen Authentisierung sollten die Verbindung aber unbedingt verschlüsselt erfolgen. Dazu genügt ein SSL-fähiger Webserver. Informationen, die jetzt aus der Zuordnung von Kliniken zu Netzbereichen stammen, können dann für jeden Benutzer individuell in Form der Zugehörigkeit zu verschiedenen Organisationseinheiten und der zugeordneten Rollen beim Benutzerverzeichnis abgefragt werden. Damit fällt einerseits die Ortsabhängigkeit des Zugriffs weg, andererseits sind die Zugriffe personenbezogen nachvollziehbar.

Gezielte Berechtigungsabfragen Mit gezielten Berechtigungsabfragen kann die bisherige Navigation unterstützt werden. Die auswählbaren Kliniken und die möglichen Rollen werden direkt den Benutzerdaten entnommen und zur Auswahl angeboten. Für die Auswahl der Befunde wird eine Berechtigungsabfrage wie in folgendem Beispiel gestellt:

Beispiel: *Berechtigungsabfrage für den Zugriff auf die aktuellen Befunde des Zentrallabors durch den Benutzer '1234567' in der Rolle 'ArztKI0001'.*

Die Abfrage soll dabei auf den Zuständigkeitsbereich der Station A1 der Kinderklinik (= „KLINIKUM.KI.KI01“) eingeschränkt sein. Die Rolle 'ArztKI0001' muss ein Recht wie in Abschnitt 5.2.3, Beispiel 1, 'BefundeOE001' mit OE = 'KLINIKUM.KI.KI01' besitzen.

Der Aufruf `getPermission('1234567', 'ArztKI0001', preq)` ermittelt die berechtigten Patienten(teil)akten, wobei:

```
preq.operation.term = 'Information.Laborbefunde_abfragen'
preq.operation.vocabulary = 'UKMZIntern'
preq.viewName.term = '*'
preq.viewName.vocabulary = '*'
preq.sparms[0].key = 'CURRENT'
preq.sparms[0].value = '0'
```

```
preq.sparms[1].key = 'SCOPE_UNIT'
preq.sparms[1].value = 'KLINIKUM.KI.KI01'
preq.source.term = 'GLOBAL'
preq.source.vocabulary = 'UKMZIntern'
```

Mit dem anschließendem Aufruf `getAdminRecords()` erhält das System eine Liste der Patienten-/Fall-ID-Paare.

Die Menge der zurückgelieferten Fälle muss dann mit der Menge der vorhandenen Befunde abgeglichen werden und liefert so die Auswahlliste für den Benutzer. Erst wenn der Benutzer Befunde eines anderen Bereichs einsehen möchte, muss eine neue Abfrage gestartet werden. Für eine Auswahl der Befunde nach dem Patientennamen, so wie sie vom Befundserver angeboten wird, benötigt die Berechtigungsabfrage die Suchparamter (`'PATIENT_NAME'`, <*Namensanfang*>) und (`'SEARCH_POLICY'`, `'INITIALS'`).

Die Funktionalität kann mit Hilfe eines vorgefertigten Programms in die PHP-Skripte integriert werden, das mit definierten Parametern gestartet wird und die Rückgabewerte nach Standard-Output schreibt. Die Kommando-Parameter umfassen Benutzerinformationen (Subjekt-ID, Rolle, Benutzerzertifikat und privater Schlüssel), Suchparameter der spezifischen Abfrage (`START, END, CURRENT, SCOPE_UNIT`) sowie die Informationsquelle.

5.4 Exkurs: Single-Sign-On

Die Forderung nach einem „Single-Sign-On“-Verfahren ist ein sehr aktuelles Thema. Die wachsende Zahl der Systeme, mit denen jeder einzelne Benutzer konfrontiert wird, und die Anzahl der heute noch damit verbundenen Loginname/ Passwort-Kombinationen, die er sich zu merken hat, sind jetzt schon nicht mehr handhabbar. Für die Anforderungen an Vertraulichkeit und Nachvollziehbarkeit von Zugriffen ist eine rechner-basierte Authentisierung nicht ausreichend; sie schränkt außerdem die Ortsunabhängigkeit stark ein. Deshalb wächst der Druck nach einem arbeitsplatz-unabhängigen Single-Sign-On-Konzept für die Systeme des Uniklinikums. „Single-Sign-On“ hat einen inhaltlichen und einen mehr technischen Aspekt:

Inhaltliche Voraussetzungen Für ein arbeitsplatz-unabhängiges Single-Sign-On-Verfahren muss jeder Benutzer im Klinikum mit genau einer eindeutigen ID für alle Systeme des Klinikums registriert sein. Die Authentisierung an den einzelnen Systemen findet dann entweder auf der Basis von Zertifikaten oder, wenn das nicht möglich ist, mit Loginname und Passwort

statt, wobei Loginname/Passwort-Kombinationen beim zentralen Benutzerverzeichnis überprüft werden müssen. Auch alle zusätzlich benötigte Benutzerinformationen (z. B. Organisationszugehörigkeit, Qualifikation, Rollen) werden dem zentralen Benutzerverzeichnis entnommen. Systeme, die keine zentrale Benutzerverwaltung integrieren können, müssen eine automatische Synchronisierung der Benutzerinformationen mit dem zentralen Benutzerverzeichnis einrichten.

Technische Voraussetzungen Jeder Benutzer meldet sich einmal lokal an seinem (oder irgendeinem) Arbeitsplatz in der Klinik an, wobei eine Sicherheitsumgebung aktiviert wird, die die Credentials (Zertifikat und privater Schlüssel) lokalisiert bzw. Loginname und Passwort sicher entgegennimmt und alle weiteren Authentisierungs-Dialoge im Hintergrund durchführt. Bei parallelem Einsatz von starker und schwacher Authentisierung erfolgt die Anmeldung mit Loginname, Passwort und PIN. Diese Informationen werden bis zur Abmeldung in einem verschlüsselten Verzeichnis der Sicherheitsumgebung (oder auf einer Smartcard) gehalten. Abbildung 5.6 zeigt

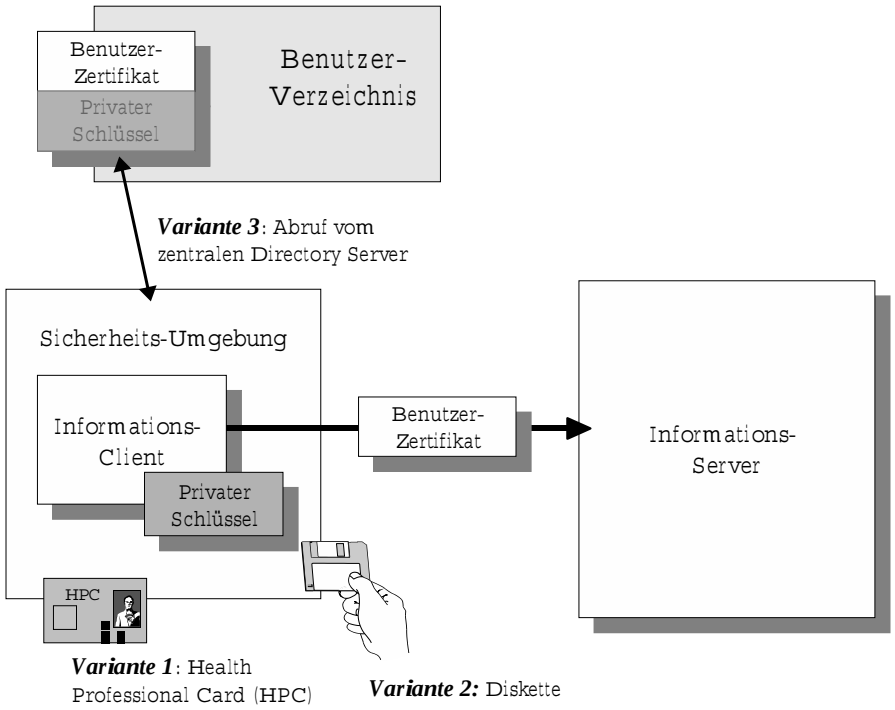

Abbildung 5.6: Quellen der Credentials

die möglichen Quellen der Credentials für arbeitsplatzunabhängige Zugriffe, wobei Variante 1, die karten-basierte Lösung, als sicherste Methode

langfristig angestrebt wird, während die Varianten 2 und 3 vorübergehend alternativ – je nach den Möglichkeiten der Umgebung[10] – eingesetzt werden sollen. Unabhängig vom Aufbewahrungsort muss der private Schlüssel immer kryptografisch geschützt sein. [15] beschreibt eine Umgebung, mit der Verfahren zur Authentisierung, Verschlüsselung und sicheren Aufbewahrung und Handhabung privater Schlüssel integriert werden können.

Single-Sign-On und das Zugriffskontroll-System

Das Konzept des Zugriffskontroll-System liefert einen Teil der inhaltlichen Voraussetzungen für die Einrichtung eines arbeitsplatz-unabhängigen Single-Sign-On-Verfahrens:

- Es liefert ein einheitliches Modell des Benutzers (Subjekts) und seiner Rechte, das jedes „beteiligte“ System verarbeiten kann.
- Das zentrale Benutzer-Verzeichnis liefert alle notwendigen Benutzerinformationen inklusive der Credentials für den ortsunabhängigen Zugriff. Es ist auf der Basis des standardisierten Lightweight Directory Access Protokolls (LDAP) erreichbar, für das inzwischen Bibliotheken und Module plattformunabhängig in vielen Programmiersprachen angeboten werden.
- Durch das Rollenkonzept ist die Autorisierung zu verschiedenen Funktionen und die Zugriffskontrolle unabhängig vom zugreifenden Subjekt. Das heißt, dass selbst bei Funktionswechseln keine erneute Authentisierung stattfinden muss. Stattdessen muss ein Rollenwechsel unterstützt werden, der entweder vom Informationssystem gesteuert wird, oder – bei der Verwendung von rollenspezifischen Tickets – durch den vorgeschalteten Zugriffskontroll-Client, wobei der Informationssystem-Client das aktualisierte Ticket entgegennehmen kann oder erneut gestartet wird. In beiden Fällen muss der Benutzer keine Anmeldeprozedur durchlaufen.

[10] Ein frei zugängliches Diskettenlaufwerk ist grundsätzlich eine Sicherheitsschwachstelle des Rechners.

Kapitel 6

Diskussion

Für die meisten Probleme gibt es mehr als eine Lösung, für Probleme wie das vorliegende gibt es aber selten eine hundertprozentig vollständige Lösung, und jeder Vorzug eines Lösungswegs hat gleichzeitig seinen Preis. Das vorgestellte Konzept und das daraus entwickelte System ist **ein** möglicher Weg, um die gestellten Anforderungen zu erfüllen. Es ist der Versuch die Anforderungen so umfassend wie – unter den gegebenen Bedingungen – möglich zu erfüllen. Trotzdem bleiben auch hier Lücken und offene Fragen. Dieses abschließende Kapitel fasst zusammen, welche Anforderungen erfüllt und welche Entscheidungen bei der Konzeption getroffen wurden, wie der Ansatz in die Arbeiten anderer Wissenschaftler-Gruppen einzuordnen ist und welche Fragen und Probleme offengeblieben sind.

6.1 Eigenschaften des Lösungsansatzes

6.1.1 Zugriffspolitik

Das Konzept stellt die wesentlichen Punkte einer Zugriffspolitik vor, wie sie auch für das Mainzer Universitätsklinikum verbindlich festgelegt werden muss. Sie stützt sich auf Vorarbeiten, die in Kooperation mit dem Datenschutzbeauftragten des Klinikums, den Landesdatenschutzbeauftragten und im Rahmen der GMDS-Arbeitsgruppe „Datenschutz und Datensicherheit in Gesundheitsinformationssystemen“ geleistet wurden. Detailliertere Informationen können in den Empfehlungen der Arbeitsgruppe [31] und in den Informationen zum Datenschutz der Landesdatenschutzbeauftragten von Rheinland-Pfalz [17] nachgelesen werden.

6.1.2 Eigenschaften des Modells

Zugriffskontroll-Modell

Das Modell umfasst alle Informationsstrukturen und Relationen, die zur Umsetzung der Zugriffspolitik in ein Zugriffskontroll-System benötigt werden, und erfüllt dabei folgende Eigenschaften:

- Es erlaubt die Abbildung abstrakter Zugriffe durch Klassifizierung der Zugriffsmerkmale. Damit können Rechte a priori definiert und vergeben werden.

- Die Definition umfasst vier Achsen, Operation, Datensicht, Systembezug und Behandlungsbezug, deren Inhalt weitgehend unabhängig voneinader gewählt werden kann. Damit können Rechte flexibel an die zu modellierenden Funktionen angepasst werden.

- Die Vokabulare bieten eine verbindliche, aber flexible Begriffsgrundlage; je nach Bedarf können beliebige Vokabulare hinzugenommen werden. Die interne Hierarchie der Vokabulare liefert bereits eine Vorstrukturierung der Rechte, die Definition und Auswertung vereinfacht. Sie ermöglicht außerdem die Belegung der Rechteachsen je nach Anforderung in unterschiedlichen Komplexitäts- und Abstraktionsstufen.

- Rechte modellieren einerseits Funktionen von Systemen andererseits den dynamischen regelbasierten Bezug zu Behandlungsphasen. Durch die Erfassbarkeit von Kontextregeln kann jederzeit ein weiterer dynamischer Regelbaustein definiert oder verfeinert und somit auf neue Anforderungungen reagiert werden.

- Komplexe Datensichten erlauben eine sehr feine Differenzieung der adressierten Daten nach Datenklassen, Inhalten und weiteren Aspekten.

- Das Konzept behandelt gleichermaßen Personen und Systeme als Subjekte von Zugriffen. Es beinhaltet sowohl die Autorisierung zu abstrakten funktionalen Zuständigkeitsbereichen in Form der Rollen als auch zu vorübergehenden statischen Rechten, die Zugriffe auf konkrete Patientenakten referenzieren.

- Zusätzlich liefern die Rechte Informationen über Zugriffsbedingungen, z. B. den Umfang der geforderten Protokollierung.

- Durch die Erfassung der Organisations-Hierachie, der Organisationszugehörigkeit von Personen, durch die Möglichkeit, Dienstpläne zu berücksichtigen und vor allem durch die Integration von Wissen über Behandlungsprozesse, werden die umgebungsspezifischen Einflüsse auf Berechtigungen abgebildet.

- Die Informationsstruktur der Abfrage-Ergebnisse ist unabhängig vom Behandlungsprozess-Modell. Sie ist deshalb invariant gegen Veränderungen des Modells und der Regeln und ermöglichen damit problemlose Anpassungen, z. B. Verfeinerungen der einrichtungsbezogenen Phasen zu personenbezogenen Phasen und der allgemeinen Beteiligungen zu speziellen Handlungen.

Das Zugriffsmodell, insbesondere die regelbasierten Aspekte und die Integration dynamischer Informationen, ermöglichen die Umsetzung der Zugriffspolitik und dabei vor allem eine **gute Annäherung an das „Need-To-Access"-Prinzip**.

Administrations-Modell

Bei der Modellierung der Informationsstrukturen wurde auch die Notwendigkeit einer handhabbaren Administration der Zugriffspolitik berücksichtigt.

- Die Rollendefinition und die Rollenhierarchie bilden Funktionen von Personen und Gruppen ab. Die Berechtigungen werden dadurch überschaubar und können leichter auch von Administratoren ohne spezifische Kenntnisse des Modells zugeordnet werden. Die Rollenhierarchie erleichtert die Definition komplexerer Berechtigungen und liefert Strukturen für die Kontrolle der administrativen Zugriffe, wie z. B. die Festlegung von Zuständigkeitsbereichen und die Delegation von Rechtezuweisungen.

- Die Struktur der Administratorrechte ermöglicht die teilweise Dezentralisierung der Definitions- und Autorisierungsaufgaben.

Die Strukturierung der Berechtigungsinformation und die Berücksichtigung von unterschiedlichen Zuständigkeitsbereichen unterstützen die **Handhabbarkeit der Administrationsaufgaben**.

6.1.3 Eigenschaften des System-Konzepts

Wissenserfassung

- Die Wahl webbasierter Technologie für die Realisierung der Erfassungswerkzeuge macht diese dezentral einsetzbar und fordert keinen zusätzlichen Wartungsaufwand an den lokalen Einsatzorten. Da sich die Erfassungswerkzeuge außerhalb dieser Arbeit in der Entstehungsphase befinden, wird an dieser Stelle nicht weiter auf die Vorteile der vorgeschlagenen Funktionsweise eingegangen.

- Die Verwendung vorhandener Wissensquellen, wie das Verwaltungssystem und künftig auch Anforderungssysteme, sowie die Nutzung vorhandener Infastruktur, wie das Benutzerverzeichnis und der Kommunikationsserver, verringert den Gesamtaufwand sowohl bei der Realisierung als auch beim Betrieb des Zugriffskontrollsystems.

- Durch die Automatisierung der Informationserfassung, insbesondere der Behandlungsprozess-Information, wird die zeitnahe Verfügbarkeit der Information verbessert und zusätzlicher Erfassungsaufwand überflüssig gemacht. Die Lösung enthält auch Konzepte für die künftige Integration weiterer Informationen wie z. B. Dienstpläne.

Verfügbarkeit

- Informationen über aktuelle Rechte von Personen oder Systemen sind in vielfältiger Form abrufbar. Durch die Suchparameter bei der Abfrage der Information, können Rechtemengen hinsichtlich spezifischer Funktionen, Systeme und Patientenakten sowie konkrete Zugriffsentscheidungen hinsichtlich eines ausreichend spezifizierten Zugriffs ermittelt werden. Dieses Verfahren bietet eine hohe Flexibilität für die Integration in die Funktionalität bestehender Systeme, wie in den Beispielen des Kapitels 5 demonstriert wurde.

- Durch spezielle Funktionen wird die ermittelte Berechtigungsinformation auf spezifische Aspekte der Patientenakte reduziert und ermöglicht so eine effizientere Auswertung.

- Die Unabhängigkeit des Modells von Annahmen über Funktionen und Datenmodelle der Informationssysteme und die Möglichkeit der Verwendung standardisierter Modelle bietet breite Integrationsmöglichkeiten.

- Die Verwendung standardisierter Technologie (CORBA) für den MasterAccessControl-Service, ermöglicht die Unterstützung der gängigsten Plattformen und Entwicklungsumgebungen. Die Generierung von Tickets ermöglicht in eingeschränktem Umfang die Einbindung von Systemen, in die der MasterAccessControl-Service nicht integriert werden kann oder soll.

- Die Systemunabhängigkeit der Lösung, die zentrale Position des Zugriffskontroll-Systems und die Auslagerung der Subjektinformation in ein zentrales Benutzerverzeichnis unterstützen die Verwendung eines Single-Sign-On-Mechanismus für die Informationssysteme des Klinikums.

- Das Konzept enthält verschiedene Überlegungen zur Optimierung und zur Frage der Performanz des Systems. Die Entkopplung von Teilaufgaben und deren Verteilung auf unabhängige Server, die jeweils auch redundant und parallel betrieben werden können, bieten Skalierungsmöglichkeiten für den Betrieb.

Sicherheit

- Die Authentisierung der AccessControl-Clients verhindert unkontrollierte Zugriffe und macht insbesondere Zugriffe mit destruktivem Charakter (Denial-Of-Service-Attacks) nachvollziehbar und unabstreitbar.

- Die Verwendung des SSL-Protokolls schützt die Daten bei der Übertragung.
- Für die Komponenten des Systems wurden außerdem allgemeine Sicherheitsanforderungen formuliert, die bei Implementation und Betrieb berücksichtigt werden müssen.

6.2 Design-Entscheidungen

Im Laufe der Modellierung wurde einige Entscheidungen getroffen, die nicht zwingend waren, aber den Lösungsweg charakterisieren. Im folgenden werden einige dieser Entscheidungen erläutert und begründet. Nicht alle Entscheidungen werden hier aufgeführt; viele folgten direkt aus der Analyse der Anforderungen, oder wurden, wie z. B. die Wahl des rollen-basierten Modells, bereits umfassend begründet.

6.2.1 Modellierung

Klassifikationskriterien

Die Klassifikation der Zugriffsobjekte berücksichtigt Operationsklassen, Datenklassen, Systeme und Klassen von Behandlungsphasen. Der Autor oder die Einrichtung der Datenerfassung werden nicht wie in den klassischen „Discretionary Policies“ explizit berücksichtigt. Der Grund für diese Entscheidung liegt im Wesen der Zugriffspolitik: Die Regeln referenzieren nicht den Autor der Daten, sondern die medizinischen Umstände unter denen Daten erfasst werden. Systeme, die den Autor als Klassifikationskriterium zur internen Verwaltung der Daten berücksichtigen, erhalten bei der Anmeldung des Benutzers ausreichende Informationen über Subjekt, Rolle und Organisationszugehörigkeit. In dieser Art kann auch die Kontrolle der administrativen Zugriffe verfeinert werden, falls es z. B. die Menge der Administratoren für einen Bereich erfordert.

Vokabulare

Grundgedanke des kontrollierten Vokabulars war es, die Sprache, in der der Informationsfluss zwischen Definition und Auswertung stattfindet, verbindlich und nachvollziehbar an zentraler Stelle abzulegen und somit auch eine sprachliche Konsistenzkontrolle der Rechte zu ermöglichen. Die Registrierung der definierten „Rechtebausteine“ wie Datensichten und Kontextprototypen geben diese implizit für die Verwendung frei, d. h., erst wenn der Begriff im Vokabular registriert ist, kann die Kontextregel oder Datensicht referenziert werden. Ferner entlastet die Registrierung der Kontextprototypen den Kontextserver von Zugriffen bei der Definition der Rechte. Die Möglichkeit, für jeden Bereich mehrere Vokabulare zu erfassen, wurde eingeräumt, um einerseits die maximal mögliche Anpassungsfähigkeit bei der Integration zu erreichen und andererseits im Zyklus der Weiterentwicklung verschiedene Versionen eines Vokabulars differenzieren

zu können. Die Erfahrung hat gezeigt, dass sich insbesondere die klinikweiten Bezeichner der Organisationseinheiten des öfteren ändern.

Attribute der Ergebnismengen

Die einzelnen Ergebnisse aus einer Berechtigungsabfrage referenzieren Krankenteilakten auf der Basis der eindeutigen IDs für Patienten und Verwaltungsfälle sowie der zeitlichen Bezugsphase. Weitere Attribute sind optional, wie beispielsweise die Organisationseinheit, die Phasen-ID und der Behandlungstyp. Für eine zusammenhängende Phase sind diese Attribute nicht immer eindeutig definiert. Die Frage war, welche Attribute zur Festlegung der erlaubten Zugriffe notwendig sind, und ob innerhalb einer zeitlichen Phase des gleichen Falls Daten entstehen können, die selbst durch die zusätzliche Angabe der Zugriffsoperation und der Datenklasse im Sinne der Zugriffspolitik nicht spezifisch genug referenziert werden können. Aus der Sicht der Autorin ist diese Differenzierung die Aufgabe einer sinnvollen Klassifikation der verschiedenen Datenarten, die im klinischen Ablauf entstehen. Die Referenzierung des Behandlungsbezugs in Form zusammenhängender Phasen ist sicherlich die Lösung, die sich am effizientesten auswerten lässt.

Festgelegte versus generische Strukturen

Bei der Modellierung des Modells stellte sich die Frage, welche Strukturen als gegeben angenommen werden können und somit direkt als Datenstrukturen oder Algorithmen in das Modell eingehen und welche Strukturen variabel sind und deshalb als generische Strukturen abgebildet werden müssen. Ohne alle Details künftiger Aspekte der Zugriffspolitik zu kennen, bieten generische Kontextregeln eine hohe Flexibilität bei der Umsetzung. Der Aufwand für die Implementation wurde durch die Festlegung spezifischer Regeltypen auf ein akzeptables Maß gebracht. Die sehr schlichten Strukturen des Behandlungsprozessmodells besitzen eine geringe Variabilität auf der Basis der Elementtypen und der Beteiligungsrollen. Da die Behandlungsprozess-Struktur Grundlage der Regeldefinitionen ist, hätte eine größere Variabilität die Komplexität der Kontextregel-Definitionen vervielfacht. Deshalb wurde von einem stärker modifizierbaren Prozessmodell, z. B. auf der Basis eines Workflow-Management-Systems wie dem in der Arbeit von Ultes-Nitsche und Teufel [44] beschriebenen Abstand genommen. Abgesehen davon genügen die modellierten Strukturen, um die Regeln der Zugriffpolitik umzusetzen, wie in Abschnitt 5.2.1 gezeigt wurde.

Anzahl der Rollen pro Sitzung

Das Modell begrenzt die Anzahl der Rollen, die bei einer Berechtigungsanfrage übergeben werden können, auf eine. Eine Rolle steht jeweils für eine vollständige Funktion, mit allen benötigten Rechten. Bei der Anmeldung legt sich der Zugreifende auf die jeweils passende Rolle fest. Eine Abfrage der Rechte aller Rollen einer Person würde die Nachvollziehbarkeit der Berechtigungsgrundlage

von Zugriffen erschweren oder verhindern und unnötig umfangreiche Verarbeitungsprozesse und Ergebnismengen verursachen.

6.2.2 Methoden und Technologie

Objektorientiertes Design, relationale Datenbanken und web-basierte Benutzeroberflächen entsprechen dem heutigen Stand der informatischen Technik für Design und Implementation komplexer Software-Systeme und werden deshalb auch hier eingesetzt.

CORBA

Die Common Object Request Broker Architecture (CORBA) ist ein Standard zur Interoperabilität von Systemen, für den es bereits einige auch frei erhältliche Implementationen gibt. Basis der Architktur sind die Object Request Broker (ORB), die das Zusammenwirken verschiedener Objekte und Dienste steuern und die plattform und sprachspezifische Übersetzungen vornehmen. Die Spezifikation enthält eine Vielzahl von Diensten für verschiedene Aufgaben und Domänen (z. B. auch den RAD-Service der CORBAMed-Spezifikation), die aber nur in geringem Umfang als Implementationen verfügbar sind. Die erhältlichen Implementationen enthalten i. a. grundlegende Funktionen zur Kommunikation, zur Benennung und Lokalisation von Objekten und teilweise auch Sicherheitsfunktionen, wie die Unterstützung SSL-basierter Kommunikation. Die gewählte ORB-Software, ORBacus (Version 4.0.4), bietet unter anderem diese Sicherheitsfunktionen. Sie unterstützt außerdem Objektimplementationen in C++ und Java und ist für nicht-kommerzielle Entwicklungen frei verfügbar. Durch die Wahl dieser Technologie für die Realisierung der Zugriffskontroll-Services wird die Integration in die heterogene verteilte Systemumgebung des Klinikums wesentlich erleichert.

Architektur

Die Architektur des Zugriffskontroll-Systems ist durch die Aufteilung der Funktionalität in unabhängige Services, die Sammlung der Informationen in zentrale Repositorien und die Verfügbarkeit der Informationen über eine spezifische Retrieval-Schnittstelle charakterisiert.

Die Aufteilung der Server und Services entkoppelt die verschiedenen Informationsarten und Aufgaben und ermöglicht eine bessere Verteilung der Systembelastung.

Die Entscheidung, alle Informationen über Behandlungsprozesse in einem zentralen Wissensserver zu sammeln und nicht dort abzufragen, wo sie entstehen, nämlich beispielsweise im Verwaltungssystem oder dem Laborsystem, wurde wegen der Art der vorhandenen Schnittstellen und der Infrastruktur getroffen. Die gesammelte Information stammt aus dem Informationsfluss, der vom jeweiligen System über den Kommunikationsserver weitergeleitet wird. Diese Schnittstellen exitieren bereits oder müssen für weitere Erfordernisse ohnehin

eingerichtet werden und sind mit einfachen Verfahren und unter Ausschöpfung der Funktionalität des Kommunikationsservers nutzbar. Der Zugriff auf die ursprünglichen Systeme würde pro System eine zusätzliche Dialog-Schnittstelle erfordern, auf deren Performanz kaum Einflussmöglichkeiten bestünden. Der hier modellierte Wissenserver und das zugrundeliegende Informationsmodell wurden mit Blick auf die Reaktionszeit konzipiert und können noch weiter optimiert werden. Nachteil dieses Wegs ist die Verzögerung, mit der die Kommunikationsschnittstellen das Wissen abgeben.

Der Ansatz von Staccini et al [43] propagiert die Verwendung von Directory-Servern, die die kontextabhängige Information über eine LDAP-Schnittstelle zur Verfügung stellen. Die Integration von Zugriffen über verfügbare LDAP-Module ist noch einfacher als die vorgeschlagene Vorgehensweise mit einer CORBA-basierten prozeduralen Abfrageschnittstelle. Das Konzept von Staccini et al erlaubt aber keine Verknüpfung von statischen und kontextabhängigen Berechtigungsinformationen, sondern nur allgemeine Rechte-Profile, die für alle Behandlungszusammenhänge eines Subjekts gleichermaßen gelten. Zugriffsarten zu verschiedenen Behandlungszusammenhängen können damit nicht differenziert werden. Das vorliegende Konzept erlaubt hingegen die Kombination von statischem und dynamischem Wissen zum Abfragezeitpunkt auf der Basis der Kontextreferenzen in den definierten Rechten.

6.3 Einordnung der Lösung

Manche der verwendeten Lösungsansätze stammen aus der Literatur, wie z. B. das Rollenmodell, das fast vollständig umgesetzt wurde. Für andere Fragen enstanden Lösungsansätze unabhängig von veröffentlichten Ansätzen, aber mit einem ähnlichen Resultat, wie im Fall des MasterAccessControll-Service, der in Struktur und Funktion dem RAD-Service [24] ähnelt. Die meisten Ansätze ließen sich jedoch nicht ausreichend auf die gestellten Anforderungen übertragen.

Hinsichtlich der Interoperabilität des Systems stellte sich die Frage, ob es sinnvoller wäre, den standardisierten RAD-Service anstelle des selbstkonzipierten MasterAccessControl-Services zu realisieren. Der Vorteil eines standardisierten Service ist, dass unabhängig von dessen Implementation bereits Klienten realisiert werden können, die überall einsatzbar sind, wo der standardisierte Service angeboten wird. Insbesondere in Umgebungen, in denen noch weitere CORBAMed-Services existieren, können so viele Absprachen und Vereinbarungen eingespart werden. Im vorliegenden Fall ist weder zu erwarten, dass es in absehbarer Zeit Systeme mit RAD-Client-Funktionalität gibt[1], noch würde der standardisierte Service die Absprache der Vokabulare und die Attribute zur Beschreibung der Zugriffsobjekte (secured resources) ersparen. Darüberhinaus bietet der MasterAccessControl-Service einen größeren Funktionsumfang zur gezielten Abfrage differenzierter Berechtigungsinformation, während der RAD-

[1] Da könnte man die Implementation des RAD-Service gegebenenfalls als Wegbereiter für den Standard rechtfertigen.

Service nur Zugriffsentscheidungen auf eindeutig spezifizierte Zugriffsobjekte unterstützt.

Neu an dem vorliegenden Lösungsansatz ist neben den differenzierten Retrievalfunktionen auch die Möglichkeit, komplexe Zugriffsregeln dynamisch zu erfassen und verschiedene strukturierte Vokabulare für die Definition der Rechte einzubinden. Das Modell der Zugriffsrechte ist insgesamt umfassender als die in der Literatur beschriebenen Ansätze, es beinhaltet z. B. Zugriffsbedingungen, berücksichtigt differenzierte Datensichten von Zugriffen und ist außerdem unabhängig von konkreten Systemen oder Datenmodellen.

6.4 Probleme und offene Fragen

Trotz der umfangreichen Lösung gibt es Anforderungen, die unvollständig oder nicht zufriedenstellend gelöst bleiben. Die Lösung selbst wirft spezifische weiterführende Fragen auf, und manche Probleme können nur durch Veränderung der Rahmenbedingungen behoben werden. Im folgenden werden die offenen Fragen diskutiert, Handlungsbedarf aufgezeigt und die Richtung künftiger Arbeiten vorgezeichnet.

6.4.1 Offene Fragen des Modells

Vokabulare

Für die geforderten Begriffssysteme wurden in der vorliegenden Arbeit generische Vokabular-Strukturen angegeben sowie Vorschläge und Beispiele, wie diese Strukturen ausgefüllt werden können. Die Menge der Beispiele umfasst sowohl standardisierte Begriffssysteme als auch selbstdefinierte Begriffsklassifikationen. Die letztendliche Wahl und Festlegung eines oder mehrerer Begriffssysteme bleibt in diesem Rahmen offen. Standardisierte Informationsmodelle, wie das HL7 Reference Information Model (RIM) und das Informations-Modell des CEN/TC 251[2] sind derzeit in einem Entwicklungsprozess, der insbesondere eine Annäherung der Modelle anstrebt. Die Wahl eines Vokabulars auf der Basis solcher standardisierter Referenzmodelle erscheint in jedem Fall sinnvoll, jedoch erst wenn deren Entwicklung weitgehend abgeschlossen ist.

Das Konzept enthält kein Modell, wie verschiedene Vokabulare im Sinne einer einheitlichen Zugriffspolitik auf ein Referenzvokabular abzubilden sind. In diesem Zusammenhang wäre ebenfalls die Berücksichtigung von Homonymen innerhalb eines Vokabulars zu behandeln. Zu dieser Fragestellung, nämlich der Einordung von Vokabularen in einen übergeordneten semantischen Zusammenhang, gibt es aus dem Gebiet der termininologischen Systeme bereits umfassende Arbeiten und Entwicklungen. Ein konkretes Modell ist das Unified Medical Language System (UMLS) [42], das eine große Zahl medizinischer Terminologien aus verschiedenen Bereichen umfasst und miteinander in Beziehung setzt.

[2]European Standardisation Committee/Technical Committee for Medical Informatics

Ein grundlegender Ansatz für die Abbildung von Terminologien sind konzeptuelle Graphen [38]. Sie ermöglichen die Repräsentation semantischer Konzepte und Relationen der Begriffe. Durch Terminologie-Server werden terminologische Systeme für Informationssysteme nutzbar gemacht; mit dem CORBAmed LexiconQuery-Service [22] wurde ein solcher Service für die Integration in eine verteilte Umgebung spezifiziert und als Standard verabschiedet. Die genannten Methoden und Werkzeuge sind sehr mächtig und komplex, und deshalb nur mit relativ hohem Aufwand einsetzbar. Der hier benötigte Ausschnitt der Terminologie, insbesondere die Datenkategorien, betreffen nur einen kleinen Ausschnitt der gesamten medizinischen Terminologie. Es ist also fraglich, ob die Verwendung der genannten Werkzeuge in diesem Fall so große Vorteile bringt, dass sie den Aufwand kompensieren.

Datensichten

Auch die Definition und Verarbeitung von Datensichten konnte in dieser Arbeit nur in Form von Lösungs*vorschlägen* auf der Basis eines flexiblen Modells behandelt werden. Ein Grund dafür ist, dass es keine Beispiele gibt, wie die Auswertung der Datensichten in den Informations-Systemen aussehen wird. Vor allem die Berücksichtigung von Dateninhalten als „Filterkriterien“ stellt eine hohe Anforderung an die Mächtigkeit der zugrundeliegenden Sprache.

Die vorgeschlagene XML-basierte Sprachsystematik besitzt zwar ausreichende Mächtigkeit, verursacht aber nicht zu vernachlässigenden Auswertungsaufwand. In diesem Zusammenhang muss die Möglichkeit, aus XML-basierten Datensichten automatisch Darstellungsinformtionen für XML-basierte Daten zu generieren, untersucht werden. Die alternative Repräsentation der Datensichten in der Art regulärer Ausdrücke ist einfacher zu verarbeiten, ist aber auch weniger mächtig und nur für einfache Datensichten konzipiert. Die Dateninhalte sind hierbei nicht berücksichtigt. Inhaltliche Kriterien für den Ausschluss ganzer fachlicher Bereiche (z. B. Informationen aus Psychiatrie und Psychosomatik) können alternativ bei der Definition der Kontextregeln berücksichtigt werden. Die Regel wird dann so formuliert, dass Behandlungsphasen in den entsprechenden Bereichen ausgeschlossen werden. Für keine der Methoden wurden bislang semantischen Prüfungen vorgesehen; im günstigsten Fall können sowohl syntaktische Fehler als auch Widersprüche durch eine geeignete Eingabeunterstützung vermieden werden.

Der gesamte Bereich der Daten-Klassifikation wirft ausreichend Fragen für zukünftige Arbeiten auf.

Parametrisierung von Rechten

In den bisherigen Überlegungen wurden Rechte als statische Strukturen unabhängig von der späteren Rollendefinition behandelt. Die Umsetzung der Zugriffspolitik mit bereichspezifischen Rollen zeigt aber, dass häufig gerade die Rechte eine Rolle charakterisieren, deren Kontext mit der Organisationseinheit der Rolle parametrisiert ist. Mit Blick auf eine mögliche Erleichterung der De-

finitionsaufgaben erscheint die Einführung eines Schlüsselworts (z. B. EXTERNAL_ROLEUNIT) sinnvoll, das als Platzhalter an die Stelle des entsprechenden UNIT-Parameters tritt und erst bei der Rollendefinition durch den endgültigen Parametereintrag ersetzt wird. Eine echte Erweiterung des Rechtemodells wird notwendig, wenn vererbte Rechte den Parameter der abgeleiteten Rolle erhalten sollen.

Die Parametrisierung von Rechten erweitert nicht die Mächtigkeit des Modells; sie dient ausschließlich der Vereinfachung der Administrationsaufgaben.

Bereichspezifische Kontexte

Am Beispiel der Parametrisierung von Rechten wird ein anderes Problem deutlich; tatsächlich sind fast alle Kontextregeln mit der Organisationseinheit parametrisiert, deren Zuständigkeit sie betreffen. Die UNIT-Parameter von Kontexten wären demnach ein sinnvolles Klassifikationskriterium für Kontexte und damit für Rechte. Bei den Parametern, die den Suchraum einer Berechtigungsabfrage beschreiben, wird dieses Filter-Kriterium – ausgedrückt durch den SCOPE_UNIT-Schlüssel – bereits genutzt. Die dadurch mögliche Vermeidung unnötiger Kontextauswertungen ist ein wichtiger Beitrag zur Performanz der Abfragen. Trotzdem ist die Verwendung dieses Parameters mit einem Fehlerrisiko verbunden. Es gibt keine Prüfung, die sicherstellen kann, dass alle UNIT-Parameter von Kontextregeln auch wirklich die Zuständigkeiten der Organisationseinheit meinen. Im offensichtlichsten Fall ist der UNIT-Parameter in der Regel negiert, erzeugt also eine komplementäre Menge. Diese Fehlerquelle muss bei künftigen Definitionen berücksichtigt und durch die Erweiterung des bestehenden Modells gegebenenfalls behoben werden.

6.4.2 Komplexität

Das beschriebene Modell besitzt eine große Ausdrucksstärke in Bezug auf Zugriffsrechte und Abfragen und infolgedessen eine hohe innere Komplexität. Das zeigt sich deutlich in den Wechselwirkungen zwischen Vokabulargestalt, Rollendefinitionen und Abfragen. Insbesondere die Gestalt der Vokabulare hat direkten Einfluss auf die Anzahl der Rechte, die für eine Kontrollanforderung benötigt werden. Die definierten Rechte sind wiederum die Voraussetzung für sinnvolle Abfrageergebnisse. Optimalerweise wären die Definition der Zugriffspolitik und die Überprüfung der Berechtigungen vollständig entkoppelt und wechselwirkungsfrei. Dies erforderte aber einen Konsens über alle möglichen Funktionen und eine eindeutige Sprache, mit der die zugehörigen Zugriffe referenziert würden. Insbesondere wäre dann auch die klinikumsweit einheitliche Umsetzung der Zugriffspolitik sichergestellt. Bezüglich des Behandlungsbezugs wurde diese Einheitlichkeit erreicht; Kontextregeln sind vollständig unbhängig von den Spezifika der Informations-Systeme. Für die Referenzierung der Zugriffsoperationen scheitert dieser Anspruch an der Heterogenität der Systemlandschaft. Die Möglichkeit zur Definition systemspezifischer Rechte mit dafür geeigneten Vokabularen muss eingeräumt werden. Der Preis dafür ist die hohe Komplexität,

die wiederum zur Folge hat, dass Vokabulare und Rechte weitestgehend zentral definiert werden sollten, um die Einheitlichkeit der Zugriffspolitik in groben Zügen zu gewährleisten. Die Implementation der Abfragen in den Informations-Systemen sollte ebenfalls von zentraler Stelle betreut und mit den Systemverantwortlichen zusammen umfassend getestet und optimiert werden, bevor eine Schnittstelle in Betrieb genommen wird. Vor allem Abfragen, die Rechtemengen liefern, müssen so konfiguriert werden, dass eine sinnvolle Größe der Ergebnismenge nicht überschritten wird und akzeptable Antwortzeiten sichergestellt sind.

6.4.3 Rahmenbedingungen

Grundlegende Infrastruktur

Für den flächendeckenden Betrieb des Zugriffskontrollsystems in der beschrieben Art fehlt dem Mainzer Klinikum noch ein Teil der vorausgesetzten Infrastruktur. Dazu gehört die Erweiterung des Benutzerverzeichnisses, die Sicherheitsinfrastruktur mit einer Certification Authority und außerdem ein verbindlicher Zeit-Service, da die Zuverlässigkeit der Zugriffskontrolle nicht zuletzt von vertrauenswürdigen und korrekten Zeitangaben abhängt. Die Bereitstellung dieser Strukturen muss in die Planung des Gesamtsystems mit einbezogen werden.

Probleme durch Informationslücken

Abgesehen vom Fehlen grundlegender Komponenten für den Betrieb des Zugriffskontroll-Systems gibt es Strukturschwächen des verteilten Informations-Systems, die sich vor allem in Informationslücken des Kontextwissens äußern, und voraussichtlich nicht in absehbarer Zeit behoben werden können. Die Informationslücken treten in Form verspäteter, unvollständiger und grundsätzlich nicht verfügbarer Infomationen auf. Bis zur Einführung einer umfassenden elektronischen Leistungsanforderung können beispielsweise keine Beteiligungen der jeweiligen Leistungsstellen berücksichtigt werden. Bislang fehlt außerdem jegliche Information, welche der Verwaltungsfälle medizinisch zusammengehören. Die Informationen über Zeitraume sind oft unvollständig. So werden ambulante, vor- und nachstationäre Besuche nur mit ihrem Beginn registriert. Für ambulante Fälle gibt es i. a. kein Entlassungsereignis; das Fallende ist willkürlich durch das Quartalsende gegeben. Verspätete Eingaben und lange Latenzen bei der Informationsweiterleitung verursachen temporäre Informationslücken.

Als Folge einer Informationslücke kann ein faktisch zulässiger Zugriff zu einem Zeitpunkt nicht gerechtfertigt werden, was im Konflikt zur Forderung nach Verfügbarkeit aller notwendigen Informationen steht. Für den Umgang mit Informationslücken und die Minimierung der Folgen werden deshalb Übergangslösungen benötigt. In Fällen, wo die Lücke mit geringen Verletzungen der Datenschutzbestimmungen durch großzügigere Rechte kompensiert werden kann, werden entsprechende Zugriffe gewährt, wie z. B. Zugriffe auf die Stammdaten aller Patienten durch Mitarbeiter des Zentrallabors. Hieran sollten grund-

sätzlich strengere Protokollierungsanforderungen gebunden werden. Für Konstellationen mit nur selten auftretenden Informationslücken, muss die manuelle Erfassung zusätzlicher Informationen in Erwägung gezogen werden, z. B. bei der medizinischen Einordnung eines Verwaltungsfalls. Die unvollständigen Zeitangaben können teilweise extrapoliert werden, ohne in einen wesentlichen Konflikt zur Zugriffspolitik zu geraten, wie im Fall der ambulanten, vor- und nachstationären Besuche, die aufgrund des fehlenden Intervallendes nur tagesgenau berücksichtigt werden. Schon wegen der temporären Informationslücken müssen Informationssysteme die Möglichkeit vorsehen, authentisierten Benutzern auf Wunsch bei entsprechend umfassender Protokollierung alle Lesezugriffe anzubieten.

Diese grundsätzlichen Schwächen des Informationsangebots sind langfristig zu lösen.

6.5 Zusammenfassung der Ergebnisse

Das vorgestellte Konzept bietet eine Lösung der beschriebenen Zugriffskontroll-Problematik, die alle wesentlichen gestellten Anforderungen erfüllt. Es vereinigt ein einheitliches Modell, mächtig genug, die formulierte Zugriffspolitik abzubilden, mit der Flexibilität, die notwendig ist, sich an die Informationssystem-Umgebung anzupassen. Das Konzept enthält neben dem Modell die Spezifikation der gesamten Funktionalität, konkrete Vorschläge für die Realisierung geeigneter Erfassungswerkzeuge sowie einen mehrstufigen Plan für die Implementation und Inbetriebnahme des Systems. Aufgrund der äußeren Bedingungen konnte das System im Rahmen dieser Arbeit nicht soweit realisiert werden, dass Evaluationsergebnisse oder praktische Erfahrungen hätten einfließen können. Die Tauglichkeit des Modells wurde deshalb an Beispielen konkreter existierender Systeme sowie an konstruierten Fallbeispielen demonstriert. Die technischen Aspekte, z. B. die Performanz des Systems, konnte nur theoretisch auf der Basis persönlicher Erfahrungen erörtert werden. Einige der Fragen können durch weiterführende Arbeiten vertieft und gegebenenfalls eleganter und umfassender gelöst werden. Eine vergleichbar umfassende Lösung für die Zugriffskontrolle in der heterogenen verteilten Informationssystem-Landschaft eines großen Klinikums konnte jedenfalls in der Literatur nicht gefunden werden.

Kapitel 7

Zusammenfassung

7.1 Problemstellung

Die wachsende Zahl der informationsverarbeitenden Systeme und die fortschreitende Vernetzung auch im Gesundheitswesen bringen eine Fülle neuer informationstechnischer Möglichkeiten mit sich, die vor allem Arbeitserleichterung und Qualitätssicherung ermöglichen und deren Nutzung unbedingt gefördert und unterstützt werden soll. Gleichzeitig sind durch die vielfältigen Möglichkeiten auch neue Gefahren für die Sicherheit von Patientendaten, insbesondere für deren Vertraulichkeit und Integrität entstanden. Der Umfang, in dem Patientendaten in einem Krankenhaus verarbeitet werden dürfen und in dem Schutzmaßnahmen getroffen werden müssen, ist gesetzlich geregelt. Die Gesetze besagen im Kern, dass jeder Vorgang der Informationsverarbeitung, wie z. B. Datenerfassung oder Dateneinsicht, genau in dem Umfang stattfinden darf, in dem er durch die Erfordernisse der Behandlung gerechtfertigt ist (→ *„Zweckbindung“, „Prinzip der minimalen Rechte“*). Das heißt, dass Informationssysteme einem Benutzer nur die Zugriffe gewähren dürfen (und möglichst auch nur die anbieten sollen), die in diesem Sinne gesetzeskonform sind. Die meisten Systeme besitzen nicht die vollständigen Rahmeninformationen, aufgrund deren Entscheidungen über die Zulässigkeit von Zugriffen getroffen werden können. Zusätzliche Kontrollstrukturen und Informationen müssen integriert werden. Die vorliegende Arbeit enthält ein Konzept und den Entwurf eines Zugriffskontroll-Systems, mit dem diese Anforderungen in hohem Maß erfüllt werden können.

7.2 Methodik

Im ersten Schritt wurde eine detaillierte Analyse der Anforderungen durchgeführt. Die Analyse umfasst allgemeine und spezifische Anforderungen an das Zugriffs-Modell, die aus der Struktur der Zugriffspolitik und aus den organisatorischen Strukturen und Abläufen des Klinikums resultieren, Anforderungen an die Zugriffskontrollstrukturen und -mechanismen, die sich aus

den Eigenschaften insbesondere der Heterogenität des verteilten Krankenhaus-Informationssystems ergeben, sowie allgemeine Anforderungen, deren Ziel die Minimierung des Aufwands bei der Nutzung der Zugriffskontrollstrukturen im gegebenen Umfeld ist. Im zweiten Schritt wurden klassische Zugriffskontroll-Verfahren und -Mechanismen sowie spezielle Beispiel-Lösungen für die Zugriffskontrolle in Systemen des Gesundheitswesens hinsichtlich der formulierten Anforderungen betrachtet und die Eignung der verschiedenen Ansätze diskutiert. Ein Teil der Problemlösung basiert auf geeigneten klassischen Verfahren, zum großen Teil wurden eigene Lösungsansätze entwickelt. Die Lösung enthält ein Modell für die Zugriffskontrolle und darauf aufbauend den Entwurf und die Spezifikation des Zugriffskontrollsystems. Für die Durchführung wurde ein Realisierungsplan in mehreren Stufen angegeben, außerdem Vorschläge und Empfehlungen für die Systemnutzung. Da das Zugriffskontrollsystem im Rahmen der Arbeit nur stückweise prototypisch implementiert werden konnte, wurde die Integration am Beispiel dreier konkreter Systeme des Mainzer Klinikums durchgespielt und dabei Möglichkeiten und Grenzen aufgezeigt.

7.3 Ergebnisse

Ergebnis der Analyse

Die Zugriffspolitik enthält Regeln, die abstrakte Zugriffsrechte von potentiellen Informationssystem-Benutzern beschreiben und die implizit Wissen über Behandlungsvorgänge und Funktionen von Personen im Klinikum referenzieren. Infolgedessen und weil die Umsetzung der Zugriffsregeln in statische Zugriffsrechte, die in Abhängigkeit der Vorgänge im Klinikum manuell angepasst werden müssen, zu großen Aufwand verursachen würde, benötigen die Kontrollstrukturen regelbasierte Anteile, die den Bezug von Personen zu Behandlungen herstellen. Zugriffsrechte spiegeln außerdem Zugriffsfunktionen der Systeme wieder. Daraus resultiert ein Modell der Zugriffsrechte, das dynamische, regelbasierte Strukturen (→ *Behandlungszusammenhänge*) und statische Strukturen (→ *Datenklassen, Datenoperationen, Systeme*) abbildet. Darüerhinaus ergeben sich folgende Notwendigkeiten für das Modell:

- Die Formulierung und Umsetzung der abstrakten Zugriffsrechte erfordert klinikweit gültige Vokabulare für Datenklassen und Datenoperationen.
- Statische Strukturen z. B. für Organisationseinheiten und Regelstrukturen für Behandlungszusammenhänge müssen modelliert werden.
- Es ist sinnvoll, bereits bei der Definition der Rechte den unterschiedlichen Protokollierungsbedarf von Zugriffen zu berücksichtigen.

Informationssysteme haben je nach Funktionalität unterschiedlichen Bedarf an Information über Zugriffsrechte. Insbesondere enthalten die meisten Systeme eigene Zugriffskontrollstrukturen, die in die Überlegungen mit einbezogen

werden müssen, die häufig aber nicht ausreichen, um eine klinikweite gesetzeskonforme Zugriffskontrolle sicherzustellen. Deshalb muss das fehlende Wissen durch ein zentrales Zugriffskontrollsystem so zur Verfügung gestellt werden, dass es von den Informationssystemen integriert werden kann.

Die Zugriffsrechte müssen so zusammengefasst und wiederverwendbar sein, dass der Administrationsaufwand des Zugriffskontrollsystems von vorneherein minimiert wird. Außerdem sollen die Administrationsvorgänge delegierbar sein, um sie an organisatorische Abläufe, wie z. B. die Personalverwaltung oder die Dienstplanerstellung koppeln zu können.

Stand der Literatur

Von den betrachteten klassischen Zugriffskontrollmodellen gilt der rollenbasierte Ansatz für die Zugriffskontrolle im Gesundheitswesen als am besten geeignet, was in dieser Arbeit ebenfalls bestätigt wurde.

Verschiedene aktuelle Arbeiten über Zugriffskontrolle im Gesundheitswesen wurden betrachtet. Die Betrachtung fokussiert auf unterschiedliche Aspekte in den Arbeiten: die zugrundeliegende Zugriffspolitik, die Repräsentation der Rechte, Beispiele für den Einsatz dynamischer, kontext-abhängiger Kontrollstrukturen und für die Integration der Zugriffskontrolle in heterogene, verteilte Systeme. Keiner der Ansätze erfüllt die Anforderungen vollständig und umfassend. Entweder die Zugriffspolitik oder die Repräsentation der Rechte entsprechen nicht den gestellten Anforderungen, oder der Ansatz enthält nur die Lösung von Teilaspekten. Einigen Ansätzen fehlt der dynamische, regelbasierte Aspekt, oder die Kopplung von Behandlungsbezug und Zugriffsfunktion. Die meisten der gefundenen Arbeiten enthalten keine Lösung für die Integration in heterogene, verteilte Systeme. Der CORBAMed Resource Access Decision Service enthält einen weitreichenden Lösungsvorschlag für die Integration, lässt die Umsetzung der Zugriffspolitik aber weitgehend offen.

Von den vorgestellten Lösungsansätzen wurde das rollen-basierte Modell übernommen; für die Integration ergab sich unabhängig von den beschriebenen Ansätzen eine Lösung, die ähnliche Techniken verwendet wie der CORBAMed-Ansatz.

Lösung

Als Lösung wurde ein Zugriffskontrollsystem entworfen und spezifiziert, das die Anforderungen in hohem Maß erfüllt und mit angemessenem Aufwand realisiert werden kann. Folgende Eigenschaften charakterisieren das System:

- Das Informationsmodell bildet die geforderte Struktur der Zugriffsrechte ab. Die definierten Rechte sind in einem hierarchischen Rollenmodell organisiert, somit gut strukturierbar und können in verschiedenen Zusammenhängen angewendet werden. Es entkoppelt die interne Repräsentation der Rechte von der Repräsentation der Berechtigungsinformationen, die den Systemen zur Verfügung gestellt werden.

- Alle Informationen über Rechte, Rollen, Vokabulare, Regeln und Behandlungsinformationen werden in zentrale Server erfasst und zur Verfügung gestellt. Dazu wurde je eine Zugriffsschicht für die Wissensakquisition und das Wissensretrieval spezifiziert. Die Schnittstelle für das Wissensretrieval ermöglicht gezielte Abfrage von Zugriffsrechte-Mengen und Zugriffsentscheidungen und ist weitgehend unabhängig von der internen Repräsentation der Zugriffsrechte.

- Ein bereits existierendes Benutzerverzeichnis wird in erweiterter Form über eine LDAP-Schnittstelle genutzt.

- Für die Erfassung des Wissens über dynamische Vorgänge im Klinikum, insbesondere Behandlungen und Patientenaufenthalte wurde eine automatische Schnittstelle entwickelt.

- Das System realisiert außerdem ein dezentrales, rollenbasiertes Administrationsmodell, das Administratoren verschiedener Zuständigkeitsbereiche berücksichtigt.

Realisierung

Die Arbeit enthält ein Realisierungskonzept, das in der ersten Stufe die Erreichung einer Kernfunktionalität vorsieht, die zum Teil weniger aufwendige Übergangslösungen enthält, und mit der ein Pilotbetrieb möglich sein wird. In der zweiten Stufe ist der Ausbau der Komponenten geplant, die einen klinikweiten Betrieb mit großen Benutzerzahlen ermöglichen. Die dritte und möglicherweise weitere Stufen umfassen die Arbeiten, die die Flexibilität und Wartbarkeit des Systems erhöhen; hier werden die Übergangslösungen ersetzt. Zum jetzigen Zeitpunkt ist etwa die Hälfte der geplanten Funktionalität der ersten Phase prototypisch realisiert. Eingesetzte Technologien und Werkzeuge sind: CORBA (ORBacus mit C++), relationale Datenbanken (Postgres), SSL (OpenSSL), und LDAP.

Für die Umsetzung und die Administration der Zugriffspolitik im Zugriffskontroll-System enthält die Arbeit konkrete Vorschläge und Empfehlungen, wie z. B. für die inhaltliche Strukturierung der verschiedenen Vokabulare, für Basisrollen und Rollenhierarchien, für die Umsetzung der Regeln sowie einige grundsätzliche Abfragemuster.

Die Integration in Informationssysteme wird durch zwei grundsätzliche Mechanismen unterstützt, einen Mechanismus auf der Basis von Zugriffstickets, der nur eingeschränkt nutzbar ist, und einen Client-Server-Mechanismus auf der Basis verteilter Objekte, der die volle differenzierte Funktionalität der Abfrageschnittstelle einschließt. Die exemplarisch durchgespielte Integration des Kommunikationsservers, des Patientenverwaltungssystems und des Labor-Befundservers zeigt, wie vielfältig und flexibel die Integrationsmöglichkeiten sind, aber dass manche noch bestehenden Hindernisse nur in Kooperation mit den Systemherstellern beseitigt werden können.

7.4 Fazit

Mit dem beschriebenen Zugriffskontroll-System wurde ein mächtiges Werkzeug konzipiert, das die gestellten Anforderungen in hohem Maß erfüllt. Trotzdem bleiben auch offene Fragen. So darf die Komplexität des Systems nicht unterschätzt werden. Sie ist bei der Nutzung, insbesondere bei der Definition der Vokabulare und bei der Implementation der Abfragen zu berücksichtigen. Die Integration in Informationssysteme kann nicht allein den Systemherstellern oder -betreibern überlassen werden. Welche Entscheidungen hinsichtlich der angeforderten Zugriffe getroffen und wie Zugriffe protokolliert werden, bedarf einer grundsätzlichen Überprüfung. Die Frage nach Auswahl und Gestaltung der Vokabulare kann in Richtung standardisierter Terminologien und allgemeiner terminologischer Systeme noch vertieft werden.

Anhang A

Glossar

Credentials sind vertrauenswürdige Informations-Items. Die Vertrauenswürdigkeit erlangt das Informations-Item entweder durch die Signatur einer vertrauenswürdigen Institution, wie z. B. bei der Zertifizierung öffentlicher Schlüssel durch eine „Certification Authority“ (CA), oder durch die Verschlüsselung mit einem gemeinsamen symetrischen Schlüssel, wie z. B. beim Kerberos-System. Beispiele für Credentials sind Zertifikate, signierte Tickets oder elektronisches Geld.

CORBA Die **C**ommon **O**bject **R**equest **B**roker **A**rchitecture ist ein Interoperabilitäts-Standard auf der Basis verteilter Objekte mit einem zentral organisierten Kommunikationsmechanismus in Form des Object Request Brokers (ORB). Der Standard enthält eine Menge von Grundfunktionen (Common Services), Spezifikationen für Komponenten verschiedener funktionale Bereiche (Common Facilities) und Spezifikationen für Komponenten verschiedener Domänen (Domain Facilities), u. a. auch für das Gesundheitswesen (CORBAmed). [46] liefert einen umfassenden Überblick über CORBA.

CORBAmed ist die „Healthcare Domain Facility“ von CORBA. Sie umfasst eine Reihe von Service-Spezifikationen, die in verteilten Systemen des Gesundheitswesens benötigt werden, wie z. B. den „Person Identification Service (PIDS)“ zur systemübergreifenden Identifikation von Personen, den „Clinical Observations Access Service (COAS)“, um Befundinformationen verfügbar zu machen, den „Lexicon Query Service“ für den Umgang mit verschiedenen klinischen Terminologien und den „Resource Access Decision Service (RADS) als Schnittstelle für die Zugriffskontrolle“.

HL7 Health Level Seven ist ein weit verbreiteter Kommunikationsstandard für medizinische Informationssysteme. Er enthält ein nachrichten- und ergeignis-basiertes Kommunikationsmodell, d. h. er gibt Nachrichten- und Ereignistypen vor, mit deren Hilfe ein gezielter Datenaustausch zwischen

verschiedenen Systemen eines Krankenhauses stattfinden kann. Das Modell umfasst verschiedene Bereiche der Kommunikation, wie z. B. Patientenadministration (Aufnahme, Verlegung, Entlassung), Leistungsanforderung, Befundrückmeldung, Abrechnung, Zeitplanung etc. Bis einschließlich Version 2.4 basiert HL7 auf fest vorgegebenen Nachrichtentypen, die aus spezifischen Segmenten und Feldern bestehen. Ab HL7 Version 3 besitzt HL7 ein objektorientiertes Referenzinformationsmodell als Grundlage der Nachrichtendefinition.

IDL Mit Hilfe der **I**nterface **D**efinition **L**anguage werden CORBA-Objekte und -Services plattform- und sprachunabhängig beschrieben.

Kommunikationsserver Der Kommunikationsserver ist ein zentrales System, das den Datenaustausch zwischen verschiedenen Informationssystemen steuert und vereinfacht. In Mainz wurde dieser Service mit einer kommerziellen Software (DataGate^{TM}) realisiert. Eine kurze Beschreibung der Funktionsweise kann in 1.4.2 und 5.3.2 nachgelesen werden.

LDAP Das **L**ightweight **D**irectory **A**ccess **P**rotocol liefert eine Sprache zum Zugriff auf Online-Verzeichnisdienste. Ein Verzeichnis besteht aus hierarchisch angeordneten Einträgen, die aus Attributen zusammengesetzt sind. Jedes Attribut hat einen Typ und kann einen oder mehrere Werte besitzen. Eine Teilmenge der Werte bildet den sogenannten „distinguished name“, der für jeden Eintrag eindeutig sein muss.

OpenSSL ist eine freie Software, die in Form einer Bibliothek Funktionen zur Integration SSL-basierter Kommunikation in beliebige Anwendungen zur Verfügung stellt. OpenSSL ist für eine Vielzahl von Plattformen erhältlich. In ORBacus können SSL-Funktionen in Form des FreeSSL-PlugIns (FSSL) für C++ integriert werden.

ORBacus ist die Implementation eines Object Request Brokers, die für die nicht-kommerzielle Nutzung frei verfügbar ist.

SSL Das **S**ecure **S**ocket **L**ayer Protocol definiert ein Verfahren, mit dem die Vertraulichkeit und Integrität der socket-basierten Kommunikation von Programmen mit Hilfe cryptografischer Algorithmen geschützt werden. Das Protokoll beruht auf dem Drei-Wege-Challange-Response-Verfahren mit X.509-Zertifikaten.

Ticket, Zugriffsticket Unter einem Ticket oder Zugriffsticket wird eine Informationseinheit verstanden, die im Rahmen einer Zugriffsanforderung oder Anmeldung an ein System übermittelt wird und sicherheitsrelevante Informationen zum zugreifenden Subjekt enthält, im beschriebenen Fall z. B. den Namen, die Organisationseinheit, die Qualifikation und eine Liste der Zugriffsrechte. Vertrauenswürdig wird das Ticket durch eine Signatur der zentralen (vertrauenswürdigen) Stelle, die es erzeugt.

XML Die E**x**tensible **M**arkup **L**anguage ist eine Sprache zur Beschreibung strukturierter Daten. Mit Hilfe sogenannter „tags“ (<*tag_name*> *Daten-Item* </*tag_name*>) werden spezifische Informationseinheiten markiert und angeordnet. Im Gegensatz zur **H**yper**t**ext **M**arkup **L**anguage (HTML), deren „tags“ fest vorgegeben sind, werden diese in XML je nach Bedarf mit XML-Schema-Beschreibungen definiert. XML umfasst darüberhinaus eine ganze Familie von Technologien, z. B. zur Beschreibung der Datenstrukturen, zur Definition von Hyperlinks oder zur Darstellung der Daten.

Zertifikat Mit Hilfe eines Zertifikats wird ein Name und eventuell weitere Attribute, die ein Subjekt charakterisieren, durch die digitale Signatur einer vertrauenswürdigen Stelle an den öffentlichen Schlüssel des Subjekts gebunden. Der Signaturschlüssel der Zertifizierungsstelle kann von einer weiteren Zertifizierungsstelle signiert sein. Zusammen mit seinem privaten Schlüssel, kann sich ein Subjekt bei allen Servern oder Anwendungen authentisieren, die eine gemeinsame übergeordnete Zertifizierungsstelle anerkennen. Der X.509-Standard definiert Struktur und Inhalte von Zertifikaten.

Zugriffspolitik Unter einer Zugriffspolitik versteht man die Menge aller Regeln, die festlegen, unter welchen Bedingungen der Zugriff eines Subjekts auf ein Objekt erlaubt ist. Verschiedene Ansätze sind in Abschnitt 3.2 beschrieben.

Literaturverzeichnis

[1] Sozialgesetzbuch (SGB), Fünftes Buch [online]. Erhältlich im WWW: <http:// www.sidiblume.de/info-rom/arb_re/sgb/sgb5.htm>, 1988 [zitiert: 17.6.2001].

[2] Landeskrankenhausgesetz Rheinland-Pfalz, 1995.

[3] SAP AG. BC – Benutzer und Berechtigungen, SAP R/3-System Release 4.0B, 1998.

[4] Ross Anderson. Security in clinical information systems [online]. Erhältlich im WWW: <http://www.cl.cam.ac.uk/users/rja14/policy11/policy11.html>, 1996 [zitiert: 17.6.2001].

[5] Konstantin Beznosov. Issues in the Architecture of the Computerized Patient Record Enterprise [online]. Erhältlich im WWW: <http://it.baptisthealth.net/Content/Publications/bhs-cpr-security/architecture-issues/plaintxt/CPR_Security_Architecture_Issues.html>, 1998 [zitiert: 17.6.2001].

[6] Konstantin Beznosov and Yi Deng. A Framework for Implementing Role-Based Access Control Using CORBA Security Service [online]. Erhältlich im WWW: <http://www.acm.org/pubs/citations/proceedings/commsec/319171/p19-beznosov>, 1999 [zitiert: 17.6.2001].

[7] Konstantin Beznosov, Yi Deng, Bob Blakly, Carol Burt, and John Barkley. A Resource Access Decision Service for CORBA-based Distributed Systems [online]. Erhältlich im WWW: <http://www.acsac.org/1999/abstracts/fri-b-0830-beznosov.html>, 1999 [zitiert: 17.6.2001].

[8] B. Blobel, K. Engel, P. Pharow, and V. Spiegel. Health Level Seven Security Services Framework, Part 1: Basiscs of HL7 Security [online]. Erhältlich im WWW: <http://www.hl7.org/library/comittees/Secure/HL7basics3rtf.RTF>, 1999 [zitiert: 17.6.2001].

[9] B. Blobel, K. Engel, P. Pharow, and V. Spiegel. Health Level Seven Security Services Framework, Part 2: Fundamentals of HL7 Security [online]. Erhältlich im WWW: <http://www.hl7.org/library/comittees/Secure/HL7fundam3rtf.RTF>, 1999 [zitiert: 17.6.2001].

[10] W3 Consortium. Extensible Markup Language (XML) [online]. Erhältlich im WWW: <http://www.W3C.org/XML>, 1997-2001 [zitiert: 17.6.2001].

[11] Software Technologies Corporation. Monk Programmer's Reference Guide, Datagate 3.5, 1997.

[12] Dorothy E. Denning. *Cryptography and Data Security.* Addison-Wesley Publishing Company, 1982.

[13] Richard S. Dick and Elaine B. (Editors) Steen. *The Computer-Based Patient Record.* National Academy Press, 1991.

[14] Document Ontology Task Force DOTF. Proposal for an Ontology for Exchange of Clinical Documents [online]. Erhältlich im WWW: <http://www.hl7.org/Special/dotf/docs/DocumentOntologyProposalJuly00.doc>, 2000 [zitiert: 17.6.2001].

[15] U. Faltin, P. Glöckner, U. Viebeg, A. Berger, H. Giehl, D. Hühnlein, S. Kolletzki, and T. Surkau. On the development of a security toolkit for open networks – New security features in SECUDE [online]. Erhältlich im WWW: <http://sit.gmd.de/SICA/Publications/SECUDE/vispap.PDF>, 1997 [zitiert: 17.6.2001].

[16] The Internet Engineering Task Force. IP Security Protocol (ipsec) [online]. Erhältlich im WWW: <http://www.ietf.org/html.charters/ipsec-charter.html>, 2001 [zitiert: 17.6.2001].

[17] Der Landesbeauftragte für den Datenschutz Rheinland-Pfalz. Informationen zum Datenschutz Heft4: Datenschutz im Krankenhaus [online]. Erhältlich im WWW: <http://www.datenschutz.rlp.de/download/lfdheft4.doc>, 1999 [zitiert: 17.6.2001].

[18] Bundesamt für Sicherheit in der Informationstechnik. IT-Grundschutzhandbuch [online]. Erhältlich im WWW: <http://www.bsi.de/gshb>, 2001 [zitiert: 17.6.2001].

[19] Simson Garfinkel and Gene Spafford. *Practical UNIX and Internet Security (Second Edition).* O'Reilly & Assoziates, 1996.

[20] Cheh Goh and Adrian Baldwin. Towards am more Complete Model of Role [online]. Erhältlich im WWW: <http://www.hpl.hp.com/techreports/98/HPL-98-92.pdf>, 1998 [zitiert: 17.6.2001].

[21] Object Mangement Group. CORBA [online]. Technical report, Object Management Group, 1998 [zitiert: 17.6.2001].

[22] Object Mangement Group. Lexicon Query Service [online]. Technical report, OMG and CORBAmed DTF, 1999 [zitiert: 17.6.2001].

[23] Object Mangement Group. Person Identification Service (PIDS) [online]. Technical report, OMG and CORBAmed DTF, 1999 [zitiert: 17.6.2001].

[24] Object Mangement Group. Resource Access Decision (RADS) [online]. Technical report, OMG and CORBAmed DTF, 1999 [zitiert: 17.6.2001].

[25] Object Mangement Group. Clinical Observations Access Service (COAS) [online]. Technical report, OMG and CORBAmed DTF, 2000 [zitiert: 17.6.2001].

[26] J. Gulbins and K. Obermayr. *UNIX*. Springer-Verlag, 1995.

[27] John Halamka, Peter Szolovits, David Rind, and Charles Safran. A WWW Implementation of National Recommendations for Protecting Electronic Health Information. *Yearbook of Medical Informatics*, pages 398–404, 1999.

[28] M. Hayden and J. Prichard. Role-Based Access Control Techniques for Open Medical Information Systems [online]. In *Proceedings of the Decision Sciences Institute Conference*. East Carolina University, 1998.

[29] Inc. Health Level Seven. HL7 Version 2.3 (ballot draft) [online]. Erhältlich im WWW: <http://www.mcis.duke.edu/standards/HL7/pubs/version2.3/html/>, 1996 [zitiert: 17.6.2001].

[30] Inc. Health Level Seven. HL7 Version 2.3.1 (ballot draft) [online]. Erhältlich im WWW: <http://www.hl7.org/library/General/v231.zip>, 1999 [zitiert: 17.6.2001].

[31] GMDS-Arbeitsgruppe „Datenschutz in Gesundheitsinformationssysteme“. Zugriff auf Patientendaten im Krankenhaus [online]. Erhältlich im WWW: <http://info.imsd.uni-mainz.de/AGDatenschutz/Empfehlungen/Zugriff.html>, 1999 [zitiert: 17.6.2001].

[32] College of American Pathologists. SNOMED International: The Systemized Nomenclature of Medicine [online]. Erhältlich im WWW: <http://www.snomed.org>, 2001 [zitiert: 17.6.2001].

[33] Commission of the European Communities DG XIII/F AIM (Editor). *Data Protection and Confidentiality in Health Informatics*. IOS Press, 1991.

[34] Klaus Pommerening. *Datenschutz und Datensicherheit*. BI-Wissenschaftsverlag, 1991.

[35] Klaus Pommerening. Datenschutz und Datensicherheit – Verläßliche IT-Systeme (Vorlesung) [online]. Erhältlich im WWW: <http://www.uni-mainz.de/ pommeren/DSVorlesung/>, 2000.

[36] Joseph Poole, John Barkeley, Kevin Brady, Anthony Cincotta, and Wayne Salamon. Distributed Communication Methods and Role-Based Access Control for Use in Health Care Applications [online]. Technical report, National Institute of Standards and Technonlogy, 1996 [zitiert: 17.6.2001].

[37] G. Potamias, M. Tsiknakis, D. Katehakis, E. Karabela, V. Moustakis, and S. Orphanoudakis. Role-Based Access To Patients Clinical Data: The InterCare Approach in the Region of Crete. In *Medical Infobahn for Europe, Proceedings of MIE2000 and GMDS2000*, pages 1074–1079. IOS Press, 2000.

[38] A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proceedings of AAAI'97*. AAAI Press, 1997.

[39] Ravi S. Sandhu. Role-Based Access Control Models. *IEEE Computer*, 29:38–47, 1996.

[40] Ravi S. Sandhu. Role-Based Access Control. *Advances in Computers*, 46, 1998.

[41] Ravi S. Sandhu and Pierangela Samarati. Authentication, Access Control, and Intrusion Detection. In *The Computer Science and Engineering Handbook*, pages 1929–1948. CRC Press, 1997.

[42] Catherine R. Selden and Betsy L. Humphreys. Unified Medical Language System (UMLS) [online]. Erhältlich im WWW: <http://www.nlm.nih.gov/pubs/cbm/umlscbm.html>, 1997 [zitiert: 17.6.2001].

[43] P. Staccini, M. Joubert, D. Fieschi, and M. Fieschi. Confidentiality Issues within a Clinical Information System: Moving from Data-driven to Event-driven Design. *Methods of Information in Medicine*, pages 298–302, 1999.

[44] Ullrich Ultes-Nitsche and Stefanie Teufel. Secure Access to Medical Data over the Internet. In *Trends in Information and Communication Systems for the 21st Century – Proceedings of the 8th European Conference on Information Systems*, volume 2, pages 1331–1337. Wirtschaftsuniversität Wien, 2000.

[45] Mark Wilcox. *Implementing LDAP*. Wrox Press Ltd., 1999.

[46] Zhonghua Yang and Keith Duddy. CORBA: A Platform for Distributed Object Computing [online]. Erhältlich im WWW: <http://www.infosys.tuwien.ac.at/Research/Corba/archive/intro/OSR.ps.gz>, [zitiert: 17.6.2001].