Formale Modellierung von Access-Control-Policies in Dynamischen Koalitionen

Sebastian Bab
sebastian.bab@tu-berlin.de

Nadim Sarrouh
n.sarrouh@tu-berlin.de

Abstract: Das Konzept der dynamischen Koalitionen beschreibt die temporäre Vernetzung autonomer Agenten, die zum Erreichen eines übergeordneten Ziels Ressourcen und Informationen miteinander teilen. Durch moderne Technologien, etwa durch Service-orientierte Konzepte, können diese Koalitionen über jegliche Systemgrenzen hinweg eingegangen werden. Insbesondere auch im Gebiet der Medizin ist die Betrachtung dynamischer Koalitionen von großem Interesse, da die Behandlung von Patienten heutzutage nicht starren Abläufen folgt, sondern aufgrund der Möglichkeiten, die die weltweite Vernetzung und die deutlich gestiegene Mobilität bieten, dynamischeren Prozessen unterworfen ist, die neue Formen der Organisation benötigen. Im Rahmen medizinischer Abläufe ist hierbei insbesondere die Frage der Sicherheit der in einer dynamischen Koalition geteilten Informationen zu stellen.

Wir stellen in dieser Arbeit ausgehend von bestehenden formalen Arbeiten ein Modell für dynamische Koalitionen vor, welches Anwendung im medizinischen Bereich finden kann. Hierzu erweitern wir das Modell um ein Access-Control-Konzept, welches die Verteilung von Informationen in einer dynamischen Koalition über entsprechende Policies regelt. Als Ausblick der Arbeit diskutieren wir anhand der Formalismen der *Algebraic High Level Nets* und der *Abstract State Machines* Möglichkeiten der formalen Beschreibung und Modellierung der Prozessabläufe und Gegebenheiten, denen dynamische Koalitionen unterworfen sein können.

1 Einleitung und Motivation

Moderne Technologien wie Service-Orientierte Architekturen, Soziale Netze im Internet, Web 2.0, etc. erlauben es Unternehmen und Organisationen, dynamische Koalitionen über die eigenen Systemgrenzen hinweg einzugehen. Eine dynamische Koalition ist zuallererst nicht mehr als eine über eine Netzwerktechnologie (temporär) verbundene Gruppe von autonomen Agenten, die ihre Ressourcen oder Informationen mit einem gemeinsamen Ziel teilen. Dieses gemeinsame Koalitionsziel ist dabei nicht zwingend das konkrete Ziel jedes einzelnen Agenten, sondern stellt den Beweggrund und die Motivation der Koalitionsgründung dar.

Obwohl jede Koalition ihre eigenen Charakteristika aufweist, haben sie viele Gemeinsamkeiten, wie dynamisch wechselnde Mitgliedschaften, Datentransfermechanismen und Authorisationsstrukturen. Beispiele für dynamische Koalitionen können Unternehmen sein, die sich aufgrund einer Marktchance zu einer Zusammenarbeit entschließen, oder auch Katastrophenszenarien, in denen Notfalldienste, Militär und Zivileinrichtungen kurzfristig zusammenarbeiten müssen, um eine drohende Gefahr abzuwenden. Das Wort „dynamisch“

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4
weitere Artikel online:
http://informatik2011.de/519.html

bezieht sich auf verschiedene Aspekte der Koalitionen. Einerseits wird hierbei ein zeitlicher Aspekt betrachtet: aufgrund von akuten Gegebenheiten müssen Koalitionen oft sehr schnell zustande kommen und lösen sich oft auch nach kurzer Zeit wieder auf. Strukturelle Dynamik ist ebenso ein wichtiges Charakteristikum: Mitglieder können evtl. bestehenden Koalitionen beitreten oder diese während der Laufzeit verlassen.

Auch in der Medizin spielen die dynamischen Faktoren einer solchen Koalition eine große Rolle. Die Möglichkeiten, die sich heute durch Vernetzung und Mobilität bieten, erfordern neue Organisationskonzepte, welche sich wegbewegen von einer statischen Betrachtung klinischer Pfade hin zu dynamischeren Verfahren. So könnte man die Patientenbehandlung selbst als eine dynamische Koalition betrachten:

> Während der Behandlung in einem oder mehreren Krankenhäusern durchläuft der Patient verschiedenste Stationen und organisatorische Einheiten. Diese Abteilungen sind nicht selten räumlich, technologisch und informationstechnisch weit voneinander entfernt und agieren als autonome Instanzen, die ihre eigenen Befunde und Diagnosen erstellen und diese an die entsprechenden weiteren Instanzen weiterleiten (heutzutage zumeist in Papierform).

Da in den meisten Szenarien verschiedenste Institutionen Informationen transferieren und teilen müssen, sind Fragen des Informationsflusses, der Sicherheit, des Datenschutzes und des Vertrauens von großer Bedeutung. Vor allem im Gesundheitswesen sind diese Frage von so kritischer Natur, dass sie den Ausgang einer ganzen Softwareentwicklung entscheiden können. Nichtsdestotrotz fehlt Architekturentwicklern heute die formale Basis, um schon während der Entwicklung die wahrscheinlichen Konsequenzen bezüglich ihrer Entwurfsentscheidungen und Policy-Definitionen zu evaluieren.

In [BFJ+06] schlagen Bryans et al. die Nutzung von formalen Modelltechnologien vor, um eine solche Basis bereitzustellen. Anhand der Beschreibungssprache VDM[1] erstellen sie ein formales Basismodell für dynamische Koalitionen, das aus mehreren Dimensionen besteht. Jede Modelldimension identifiziert eine bestimmte Perspektive, die für Softwarearchitekten von Interesse sein könnte, wie beispielsweise die Perspektiven Koalitionsmitgliedschaft, Informationsübertragung oder *Trust*. Angelehnt an das Paradigma der Objektorientierung in der Softwareentwicklung bestehen die einzelnen Modelle aus verschiedenen Datentypen sowie aus über diesen Datentypen definierten Operationen und Invarianten. In verschiedenen Fallstudien demonstrieren Bryans et al. die Praxistauglichkeit ihres Modells (siehe [BFJM06, BFG+08]).

In anderen, von dynamischen Koalitionen unabhängigen Arbeiten stellen sie Konzepte zur formalen Modellierung von Access-Control-Policies in VDM vor. Dabei übersetzen sie Kernkompenenten des OASIS[2]-Standards XACML (*eXtensible Access Control Markup Language*) in eine VDM-Notation. In einem Ausblick schlagen sie selbst die Integration dieses Ansatzes mit den Modellen der dynamischen Koalitionen vor [BF07].

[1] VDM steht für die Spezifizierungssprache Vienna Development Method: http://www.vdmportal.org/twiki/bin/view

[2] OASIS ist ein offenes Konsortium zur Definition von Standards für Datenaustausch im Internet, siehe: http://www.oasis-open.org/

ISBN 978-3-88579-286-4

Vor allem in dynamischen Koalitionen ist das Modellieren und Testen vorgeschlagener Software-Architekturen mit Hinblick auf die Zugriffsbefugnisse von großer Wichtigkeit: Die unterschiedlichen Access-Control-Policies der einzelnen Koalitionsteilnehmer können unter Umständen dazu führen, dass ein reibungsloser Prozessablauf gestört oder gar verhindert wird, wenn die einzelnen Policies nicht von Anfang an in der Entwicklung der Prozesse und der Infrastruktur berücksichtigt werden. Bryans et al. stellten daher ein Konzept der Evolution von Access-Control-Policies in dynamischen Koalitionen vor (siehe Abbildung 1). In diesem Konzept entsteht durch Koalitionskonsens ein vorgeschlagener Prozess, welcher bestimmte Zugriffsanforderungen vorgibt. Diese Anforderungen erreichen in formalisierter Weise jeden Teilnehmer des Koalitionsworkflows, welche jetzt anhand ihrer einheitlichen formalen Repräsentation der dynamischen Koalition Tests und Simulationen laufen lassen können, um Fehlern und Widersprüchen von vornherein gerecht zu werden.

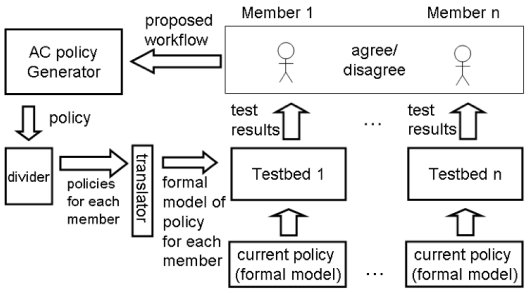

Abbildung 1: Evolution von Access-Control-Policies in dynamischen Koalitionen [BF07]

In dieser Arbeit präsentieren wir einen ersten Versuch, diese beiden Modellierungsansätze zu integrieren und damit eine weitere Dimension dynamischer Koalitionen zu betrachten: Die Access-Control-Dimension. Durch die Nutzung dieses OASIS-Standards bleibt die Implementierung trotz des formalen Charakters praxisnah. Vor allem in Hinblick auf die Anwendung für medizinische Entwicklungen möchten wir ein Modell vorstellen, das dynamische Koalitionen unter Berücksichtigung von Einstellungen für Privatsphäre und Datenschutz abbilden kann. Die verschiedenen Ansprüche könnten so im Zuge eines *Model Driven Development* von Anfang an in den Entwicklungsprozess einer Plattform für dynamische Koalitionen integriert werden. Bestimmte geforderte Datenschutz-Eigenschaften können auf Grund des formalen Charakters der benutzten Werkzeuge validiert und verifiziert werden. Aufgrund des starken Werkzeugsupports für die VDM-Spezifizierungssprache ist zudem eine Laufzeitüberprüfung der aktuellen Policies einzelner Koalitionspartner denkbar. Somit kann das Modell ebenso die eigentliche Koalitionsformierung, -gründung und -ausführung (in Form eines auf dem Modell basierenden Werkzeugs zur Gründung, Ausführung und Wartung Dynamischer Koalitionen) unterstützen.

Die Modelle von Bryans et al. betrachten lediglich die strukturellen Eigenschaften dynamischer Koalitionen. Über die rein strukturellen Eigenschaften hinausgehend ist jedoch insbesondere die Betrachtung der Prozesse von Bedeutung, denen die Koalitionen unterliegen. Wir diskutieren als Ausblick dieser Arbeit, wie formale Methoden wie *Algebraic High*

Level Petri Nets (AHL-Netze) oder *Abstract State Machines* (ASMs) dazu dienen könnten, diese Prozesse formal abzubilden. In beiden Fällen wird hierbei das VDM-Modell als eine Art *state* aufgefasst, der durch Transitionen oder Übergangsgleichungen in den verschiedenen Formalismen modifiziert werden kann, wodurch Prozessschritte und in der Gesamtheit vollständige Prozesse modelliert werden können. Die Vorteile der Integration in AHL-Netze oder ASMs ist hierbei vor allem in der bereits bestehenden Theorie hinter diesen formalen Methoden zu finden: So können durch bereits erforschte Methoden unter gewissen Voraussetzungen bestimmte Eigenschaften wie beispielsweise *Liveness*- oder *Safety*-Eigenschaften überprüft werden. In der Praxis könnte über diese Methoden somit beispielsweise untersucht werden, ob die verschiedenen Access-Control-Policies der einzelnen Agenten einer dynamischen Koalition einen reibungslosen Prozessablauf erlauben und – falls nicht – aufzeigen, an welcher Stelle im Modell das Problem auftritt.

2 Stand der Forschung

In den letzten Jahren wurden eine Reihe von Formalisierungen einzelner Aspekte dynamischer Koalitionen vorgestellt. Haidar et al. präsentieren in [HCA+09] ein formales Modell auf Basis eines Prozesskalküls und der formalen Sprache Z, welches PKI-basierte Authentifikation in dynamischen Koalitionen abbildet. Bocchi et al. stellen formale Beschreibungsansätze für Breeding Environments in virtuellen Organisationen vor, welche vor allem für das grid-computing relevant sein dürften. [BFRRM09]

Neben den Taxonomien von Bryans et al. (siehe [BFJM06, BF07, BFG+08, BFJ+06]) existieren nur wenige ganzheitliche Modellierungsansätze für dynamische Koalitionen. Der unserem Ansatz ähnlichste Ansatz findet sich in [MST09]. Hier beschreiben McGinnis et al. ein formales Rahmenwerk für Virtuelle Organisationen in Service-Grid-Umgebungen. Das Rahmenwerk erfasst die formale Beschreibung von Agenten, Services, Rollen und Workflows. Die benutzten Beschreibungssprachen basieren jedoch auf keinem uns bekannten Standard. Eine Werkzeug-unterstützte Entwicklung und Evaluierung von dynamischen Koalitionen, wie von uns in der vorliegenden Arbeit vorgeschlagen, ist daher mit diesem Rahmenwerk vorerst noch nicht möglich. Koshutanski et al. modellieren HDC (Highly Dymamic Coalitions), eine von ihnen definierte Unterklasse von dynamischen Koalitionen mit stark eingeschränkten Lebenszeiten. Dabei halten sie sich bezüglich der Modellierung an keinen uns bekannten Standard, was sich vor allem im medizinischen Sektor als großes Problem erweist. Desweiteren setzt ihr Modell eine zentrale Verwaltung in Form einer Koalitionsplattform voraus. Beide Einschränkungen sind für die von uns zu betrachtenden Anwendungsgebiete zu speziell bzw. restriktiv. Andere Ansätze sind weniger formal und stellen Modelle aus Sicht der Wirtschaftswissenschaften vor. [Klu98, Let01].

Unser Ansatz besteht in der Integration der Modellierung dynamischer Koalitionen mit dem Access-Control-Standard XACML. Als dritten Aspekt möchten wir in zukünftigen Arbeiten Prozessformalisierungen anhand gängiger Formalismen (siehe Abschnitt 4) einbauen. Sowohl zu Access-Control-Konzepten als auch zu Prozessformalisierungen über AHL-Netze oder ASMs existiert reichlich Literatur. Nach unserem derzeitigen Kenntnisstand ist unser integrierender Ansatz nicht direkt mit aktuellen Arbeiten vergleichbar.

ISBN 978-3-88579-286-4

3 Grundlagen

Im Folgenden beschreiben wir inhaltliche Voraussetzungen, die zum Verständnis des in diesem Beitrag vorgestellten Modells von Nöten sind. Als erstes geben wir einige einführende Anmerkungen bezüglich der benutzten Modellierungssprache VDM. Im zweiten Abschnitt stellen wir das in [BFJM06] beschriebene Modell für dynamische Koalitionen vor. Im dritten Teil geben wir eine kurze Einführung in den OASIS-Standard XACML, für den auch eine VDM-Spezifikation existiert.

3.1 VDM und VDM++

Das in diesem Beitrag vorgestellte Modell ist in der objektorientierten Erweiterung der Vienna Development Method (VDM) Spezifizierungssprache beschrieben. Diese Erweiterung wird sinngemäß auch VDM++[3] genannt. Diese Sprache spezifiziert Daten, Zustand und Funktionalität und bietet sich daher für die Modellierung der Struktur und der Funktion von dynamischen Koalitionen an. Eine Vielzahl detaillierter Lehrbücher und Unterrichtsmaterialen sind sowohl in gedruckter Form als auch online verfügbar[4] [FL09, FLM+05].

Ein VDM-Modell besteht aus Klassendefinitionen, in denen Instanzvariablen, Typen und Funktionalität spezifiziert werden. Instanzvariablen sind die lokalen Variablen eines Objekts. Typen können strukturlos sein (sogenannte *token*) oder komplex definiert werden. Die Typzugehörigkeit kann durch Invarianten eingeschränkt werden. Funktionalität wird hierbei ausgedrückt durch Operationen auf den Instanzvariablen oder durch Hilfsfunktionen, welche die lokalen Variablen unberührt lassen. Die Nutzung von Operationen und Funktionen kann durch logische Pre- und Post-Conditions eingeschränkt werden. Die Nähe der Notation zu gängigen objektorientierten Programmiersprachen wie etwa Java erlaubt trotz des formalen Charakters ein leichtes und eingängiges Verständnis, so dass sich Softwareentwickler schnell damit zurechtfinden.

VDM kann auf eine breite Nutzung in der Modellierung Computer-gestützter Systeme zurückblicken [LFB96, FL06]. Einer der großen Vorteile neben der weiten Verbreitung in der Industrie ist ein guter Toolsupport. Das für diesen Beitrag benutzte VDMTools[5] vereint nützliche Werkzeuge wie Syntax, Typen und Integritäts-Checker mit einem Interpreter, der die Ausführung der Modelle anhand von Testdatensätzen und Nutzung der modellierten Operationen erlaubt. Das Ausführen von Testskripten sowie eine Umwandlung in Programmcode werden ebenfalls unterstützt.

[3] Der notationellen Einfachheit halber, sprechen wir im folgendenen nichtsdestotrotz von einem VDM-Modell, da VDM++ auf der Basissprache VDM basiert.

[4] Für weitere Informationen siehe: http://www.vdmportal.org/

[5] Auch als *lite*-Version erhältlich unter: http://www.vdmtools.jp/en/

ISBN 978-3-88579-286-4

3.2 Dynamische Koalitionen in VDM

Das von Bryans et al. vorgeschlagene VDM-Modell dynamischer Koalitionen besteht in der Basis aus Agenten, die Gruppen von Agenten, sogenannten Koalitionen, beitreten oder diese verlassen können. Dabei tragen Agenten eine Menge von Informationen, die sie anderen Agenten oder Koalitionen mitteilen können. In [BFJ+06] werden verschiedene Dimensionen von dynamischen Koalitionen vorgestellt, die auf dieser Basis sieben weitere Aspekte modellieren, etwa den Informationsaustausch, Koalitionsmitgliedschaften oder Autorisationsstrukturen. Die praktische Anwendbarkeit dieser Modelle wurde in Fallstudien in Industrie [BFJM06] und Militär [BFG+08] nachgewiesen.

Das in dieser Arbeit vorgestellte Modell ist charakterisiert durch die Anreicherung des Basismodells dynamischer Koalitionen mit Access-Control-Mechanismen. Aufgrund dieser Integration eines neuen Aspekts kann dieses Modell als weitere Dimension der dynamischen Koalition verstanden werden: Dimension Access-Control. Das Modell von Bryans et al. wurde dabei an einigen Stellen verändert, erweitert oder verkürzt, um dem Scope der Betrachtung dieses Papiers gerecht zu werden.

3.3 XACML in VDM

XACML ist ein auf XML basierender OASIS-Standard für die Beschreibung und Definition von Access-Control-Policies in verteilten Systemen. Eine vereinfachte Darstellung der Funktionsweise von XACML-Policies findet sich in Abbildung 2. Nutzungsanfragen für eine bestimmte Ressource werden vom Policy Enforcement Point (PEP) abgefangen und in Form eines standardisierten XACML-Requests an den Policy Decision Point (PDP) weitergeleitet. Als Halter der Policies entscheidet der PDP anhand vom Nutzer vordefinierter Regeln, ob ein Zugriff erlaubt und abgelehnt wird, und teilt die Antwort dem PEP mit. Dieser ist dann für die Vermittlung oder Verhinderung des eigentlichen Zugriffs auf die Ressource zuständig.

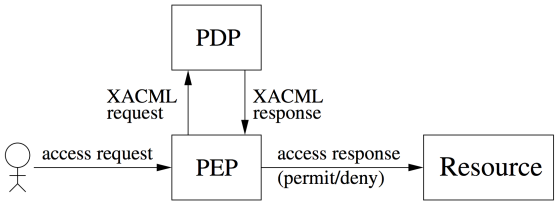

Abbildung 2: Vereinfachte Darstellung von Access Control in XACML. [BFP06]

In [BF07] wird ein VDM-Modell für XACML Access-Control-Policies nach OASIS Standard vorgestellt: Es betrachtet den einfachen Fall einer Access-Control-Komponente, die

weder Umgebungsvariablen (etwa Uhrzeit, Zugriffsort, etc.) noch Rollen-basierte Zugriffskontrolle betrachtet. [BFP06] Beide Aspekte sind vor allem in unserem medizinischen Anwendungsszenario von großer Wichtigkeit und werden in unseren zukünftigen Arbeiten mit berücksichtigt werden (mehr dazu in Abschnitt 5).

4 Dimension Access Control: Unser Modell

Im Folgenden beschreiben wir unser Modell für die Dimension Access Control in dynamischen Koalitionen. Das Modell basiert dabei lose auf den Vorarbeiten von Bryans et al. aus [BFP06], wurde allerdings an vielen Stellen erweitert und verändert. Als erstes stellen wir die Basistypen unseres Modells vor, welche sich unterteilen in Typen zur Beschreibung der dynamischen Koalitionen und Typen, die als Access-Control-Komponenten nach XACML Standard fungieren. Als letztes präsentieren wir beispielhaft einige Operationen, mit denen ein einfacher Informationstransfer innerhalb einer Koalition ermöglicht wird. [6]

4.1 Dynamic Coalition Types

Den Ansätzen von Bryans et al. entsprechend basiert unser Model auf *Agenten*, die Informationen tragen und sich zu *Koalitionen* zusammenschließen können, um in diesen Koalitionen Informationen zu teilen. Sowohl Agenten als auch Koalitionen sind durch eindeutige IDs verknüpft (Aid- und Cid-*token*). Vom Inhalt der Informationen abstrahieren wir zunächst und modellieren sie als unstrukturierte Daten (*token-Typ*). Zusätzlich zu dieser Basis fügen wir sowohl bei Agenten als auch bei Koalitionen einen PDP ein, der die jeweiligen Policies beherbergt, die für den späteren Zugriff ausgewertet werden müssen.

```
public Agent       ::  info   : set of Information
                       aac    : PDP;

public Coalition   ::  agents : set of Aid
                       info   : set of Information
                       cac    : PDP;

public Information ::  item   : token;

instance variables

coals : map Cid to Coalition := {|->};
agents: map Aid to Agent := {|->};
inv forall c in set dom coals &
        (coals(c).agents subset dom agents)
```

[6] Das vollständige Modell inklusive eines Testszenarios sind einzusehen unter: http://www.user.tu-berlin.de/nsarrouh/ACDimension.vpp

Die Invariante (*inv*) besagt, dass nur bekannte Agenten in einer Koalition sein dürfen. Der PDP einer Koalition und die PDPs der zugehörigen Agenten stehen in folgender Beziehung: Wird eine Menge von Informationen von einem Agenten in einer Koalition freigegeben, werden die zugehörigen Access-Control-Policies zusammen mit den Daten an die Koalition übergeben und in den PDP der Koalition eingefügt.[7]

4.2 Access Control Component Types

Wir integrieren den Ansatz aus [BFP06] in unser Modell dynamischer Koalitionen: Laut dem OASIS-Standard XACML bestehen Policies aus Regeln (*rules*), welche wiederum aus einem optionalem Ziel (*target*) und einem Effekt (*effect*) bestehen. Ziele bestehen aus Subjekten (*subjects*: in unserem Fall Agenten, die einen Zugriff ausüben möchten), Ressourcen (*ressources*: in unserem Fall Mengen von Informationen, auf die zugegriffen werden soll) und Aktionen (*actions*: die Art und Weise des Zugriffs, in unserem Fall beschränkt auf Lesen und Schreiben). Stimmt das Ziel einer Zugriffsanfrage (*request*) mit dem Ziel einer Regel überein, so wird der Effekt der Regel ausgegeben. Effekte können sein: Zugriff erlauben (*permit*), Zugriff verbieten (*deny*) oder nicht anwendbar (*not applicable*), falls das Ziel der Regel nicht mit dem Ziel der Anfrage übereinstimmt. Hat eine Regel kein target, so wird sie für jede Anfrage evaluiert.

```
public Rule    ::  target    : [Target]
                   effect    : Effect;

public Target  ::  subjects  : set of Aid
                   resources : set of Information
                   actions   : set of Action;

public Request ::  target    : Target;

public Action = (<WRITE> | <READ>);
public Effect = (<PERMIT> | <DENY> | <NOTAPPLICABLE>);
```

Policies enthalten eine Menge von Regeln sowie einen Algorithmus zum Kombinieren von verschiedenen Effekten, falls mehrere Regeln auf eine Anfrage zutreffen. Wir beschränken uns auf Basisalgorithmen aus dem XACML: Ablehnen überschreibt andere Effekte (*deny overrides*) und Zulassen überschreibt andere Effekte (*permit overrides*). Policies enthalten obligatorische Ziele, sodass sie nur für passende Anfragen ausgewertet werden. Der Effekt einer Policy wird an den *Policy Decision Point (PDP)* weitergegeben, der sie wiederum anhand eines Kombinierungsalgorithmus zu einem Effekt verbindet. Jede Anfrage auf eine Information wird von dem PDP evaluiert, wodurch die Konformität des Zugriffs zu den Policies gewährleistet wird.

[7] Um die Policies im Nachhinein zu ändern, müsste der Besitzer der policy als Attribut mitgeführt werden. Da wir Methoden zur Änderungen von Policies in diesem Papier nicht weiter betrachten, lassen wir auch diesen Spezialfall vorerst außer Acht.

ISBN 978-3-88579-286-4

```
public PDP    ::  policies      : set of Policy
                  policyCombAlg : CombAlg;

public Policy ::  target        : Target
                  rules         : set of Rule
                  ruleCombAlg   : CombAlg;

public CombAlg = (<DENYOVERRIDES> | <PERMITAOVERRIDES>);
```

4.3 Operationen

Die Modelloperationen *CreateEmptyCoalition*, *CreateNewAgent* und *Join* erzeugen leere Koalitionen, neue Agenten, die Informationen tragen, und fügen Agenten zu Koalitionen hinzu. Agenten in einer Koalition können Informationen in der Koalition freigeben (Methode *shareInfo*). Dabei werden die passenden Policies des PDP des Agenten in den PDP der Koalition abgelegt.

```
public shareInfo : Aid * Cid * set of Information ==> ()

ShareInfo(a,c,i_set)==
(
for all i in set i_set do
     coals:= coals ++ {c |-> mu(coals(c),
      info |-> coals(c).info union i_set,
          cac |-> mk_PDP(coals(c).cac.policies union
              GetMatchingPolicies(agents(a).aac, i),
                  <DENYOVERRIDES>))};
)
pre a in set dom agents and c in set dom coals and
     i_set subset agents(a).info
;
```

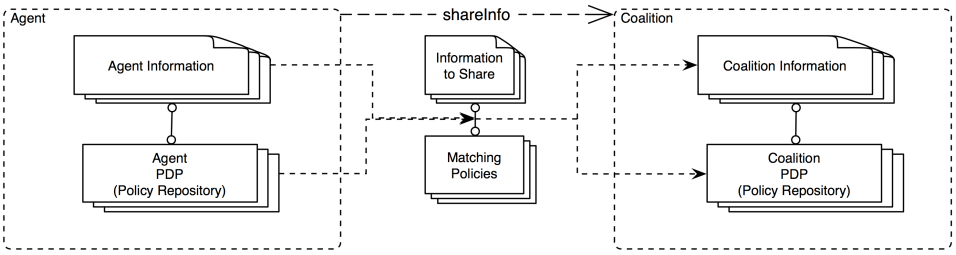

Abbildung 3: Visualisierung der Operation zum Freigeben von Informationen in Dynamischen Koalitionen

Möchte ein Agent eine Aktion auf Informationen der Koalition ausführen, so wird zuerst der PDP der Koalition mit der Anfrage evaluiert und der Zugriff dann erlaubt oder abgelehnt.

```
public RequestInfo : Aid * Cid *  Action * set of
     Information ==> Effect
RequestInfo(a,c,act,i_set)==
(
     evaluatePDP(mk_Request(mk_Target({a},i_set,{act})),
      coals(c).cac)
)
pre a in set dom agents and c in set dom coals and
     i_set subset coals(c).info
;
```

Die Funktion *evaluatePDP* prüft den Kombinierungsalgorithmus für die Policies und ruft die entsprechendende Funktion auf. Die einzelnen Policies und Regeln werden über *evaluatePolicy* und *evaluateRule* berechnet, welche wiederum je nach Kombinierungsalgorithmus die entsprechende Funktion zum Berechnen der Effekte benutzt. Aus Platzgründen verzichten wir hier auf eine ausführlichere Beschreibung dieser Funktionen und verweisen auf die vollständigen Modellbeschreibungen im Internet.

4.4 Ein erstes Beispiel

Das für unsere Arbeiten benutzte Toolkit *VDMTools* erlaubt es, Modelle mit Testdatensätzen zu füllen und die Funktionalitäten des Modells entweder über ein Interpreter-Fenster oder über extern gesteuerte Skripte zu testen. In diesem Rahmen haben wir ein kurzes, fikitives Beispielszenario implementiert[8], welches sich eng an das Beispiel aus [BF07] hält: Das Dokumentenverwaltungssystem einer Chemiefabrik verwaltet Produktionspläne und Gefahrenanalysen neuer Chemikalien. Aufgrund von Marktgelegenheiten lagert die Fabrik die Erstellung der Gefahrenanalysen an eine externe Firma aus und geht damit eine Art dynamischer Koalition ein. Weiterhin nehmen wir an, dass es die Gesetzeslage erfordert, dass der Autor einer Gefahrenanalyse sicherstellt, dass der Produktionsplan die durch die Analyse vorgegeben Sicherheitsempfehlungen implementiert. Die Chemiefabrik erlaubt allerdings aufgrund ihrer Geschäftspolitik nur den eigenen Mitarbeitern einen Zugriff auf die Produktionspläne. Beim Testen des vorgeschlagenen Workflows taucht diese Diskrepanz im Modell auf. Es liegt jetzt an der ersten Firma, den Workflow oder ihre Policies so anzupassen, dass ein reibungsloser Ablauf gewährleistet ist. Eine Prozess- und Workflow-Modellierung ist durch das bisherige Modell nicht explizit unterstützt. Tests müssen durch mühsame Erstellung von Testdatensätzen und externen Skripts durchgeführt werden. Diese Werkzeuge sind nur unzureichend formal. Aus diesem Grund stellen wir im folgenden Abschnitt einige Ideen zur Integration von bestehenden Formalismen vor.

[8] Siehe dazu http://www.user.tu-berlin.de/nsarrouh/Test.vpp.

5 Ideen zur Modellierung prozessualer Abläufe in dynamischen Koalitionen

Wie bereits in der Einleitung dieser Arbeit angesprochen, ist neben der strukturellen Modellierbarkeit dynamischer Koalitionen im medizinischen Sektor insbesondere die formale Modellierbarkeit und Handhabung der Prozessabläufe, denen die Koalitionen im Rahmen des Access Controls unterworfen sind, von Bedeutung. In diesem Abschnitt soll nun als Abschluß und Ausblick dieser Arbeit initial diskutiert werden, inwieweit mit Hilfe existierender formaler Methoden der Prozessbeschreibung in der Informatik dieses Ziel der Modellierung erreicht werden kann.

Eine formale Betrachtung von Prozessabläufen erlaubt neben der eigentlichen *Beschreibung* eines Prozesses insbesondere auch die *Analyse* des Prozesses beispielsweise in Hinblick auf prozess-immanente Invarianten oder spezielle relevante Eigenschaften (wie beispielsweise Eigenschaften zur Fehlerkontrolle und Fehlervermeidung). Unter diesen relevanten Eigenschaften sind insbesondere *Safety* und *Liveness* als besonders wichtige Eigenschaften hervorzuheben. *Safety* beschreibt hierbei die Eigenschaft, dass ein ungewünschter Systemzustand im Prozessablauf nie auftreten darf, das System somit also nicht in einen Fehlerzustand übergehen kann. *Liveness* beschreibt hingegen die Eigenschaft, dass ein günstiger Systemzustand von jedem erreichbaren Zustand eines Prozesses aus stets wieder erreicht werden kann und das System somit nicht in einen Deadlock übergehen kann. Eine Kombination beider Eigenschaften bildet die Grundlage einer jeden verlässlichen dynamischen Modellierung.

Die Wichtigkeit der Betrachtung der beiden Eigenschaften ist insbesondere auch im Kontext dieser Arbeit offensichtlich. So sollten die Prozessabläufe innerhalb einer dynamischen Koalition im medizinischen Bereich stets einen kontinuierlichen Fortschritt ermöglichen (liveness) und weiterhin ungünstige Abläufe und Fehler vermieden werden (saftey). In Hinblick auf Access-Control-Policies sind beide Eigenschaften ebenfalls von entscheidender Bedeutung, da fehlerhafte Policies innerhalb einer Koalition den prozessualen Ablauf verzögern oder ganz beenden oder Verletzungen der Access Control auftreten können. Es verbleibt die Untersuchung der Frage, wie diese Fragestellungen mit Hilfe formaler Methoden realisiert werden können.

Zur Modellierung von Prozessen in der Informatik sind verschiedene Methoden etabliert, wobei im Folgenden insbesondere zwei Techniken exemplarisch vorgestellt werden sollen, die in der Praxis vermehrt Anwendung finden. Die erste Technik ist durch die *Algebraic High Level Nets* (kurz *AHL-Netze*, siehe beispielsweise [Rei10]) gegeben, die eine Erweiterung des Konzepts der Petrinetze darstellen. Die zweite Technik sind *Abstract State Machines* (kurz *ASMs*, siehe beispielsweise [GKOT00, Bör03]), die eine vergleichsweise junge Disziplin darstellen und im Gegensatz zu den AHL-Netzen ein sehr breites Spektrum an Modellierungen erlauben. Beide Formalismen eignen sich prinzipiell zur Beschreibung der Prozesse, denen die in dieser Arbeit betrachteten dynamischen Koalitionen unterliegen. Dennoch besitzen beide Formalismen auch ihre Vor- und Nachteile, die im Rahmen eines Anwendungsfalls bedacht, studiert und gewichtet werden müssen.

AHL-Netze bilden eine Erweiterung des Konzepts der Petrinetze, in denen die Einschrän-

kung der zur Verfügung stehenden Tokens auf ununterscheidbare Dateneinheiten aufgehoben wurde. Petrinetze im Allgemeinen (und somit auch AHL-Netze als Erweiterung von Petrinetzen) sind spezielle visualisierbare Stellen- und Transitionssysteme, die über das Aktivieren und Schalten von Transitionen eine Prozessablauf-Modellierung erlauben. Sind in allgemeinen Petrinetzen Tokens hierbei stets als ununterscheidbare Dateneinheiten zu verstehen, so erlauben AHL-Netze die Verwendung komplexer Datentypen als Tokens über die Integration algebraischer Spezifikationen. AHL-Netze ermöglichen hierüber auch die Modellierung nebenläufiger Prozesse, wobei die Schaltbedingungen der Transitionen durch Gleichungen über der algebraischen Spezifikation realisiert werden.

Eine Verwendung von AHL-Netzen für die Modellierung der Abläufe in dynamischen Koalitionen ist aufgrund der breiten Forschungsarbeiten in diesem Gebiet eine naheliegende Möglichkeit. AHL-Netze können allgemein als Methode zur Modellierung sehr konkreter Probleme verstanden werden. Die Modellierung der Variabilität dynamischer Koalitionen bezüglich der Zu- und Abwanderung von Mitgliedern ließe sich über eine Variante der AHL-Netze modellieren, nämlich über die sogenannten *offenen AHL-Netze*. Offene AHL-Netze zeichnen sich durch die Existenz von Stellen aus, an denen Tokens (im Falle von AHL-Netzen also komplexe Datentypen) generiert oder eliminiert werden können. Diese offenen Stellen könnten hierbei zur Modellierung der Zu- und Abwanderung von Mitgliedern in der Koalition verwendet werden.

In einer Modellierung mit AHL-Netzen würde das VDM-Modell als *state* aufgefasst werden, der als komplexer Datentyp im Rahmen einer algebraischen Spezifikation modelliert wird und als Token im AHL-Netz zur Verfügung steht. Durch Transitionen oder Übergangsgleichungen kann dieser *state* modifiziert werden, wodurch die Abfolge der Zustandsübergange somit die Modellierung von Prozessschritten und in der Gesamtheit die Modellierung vollständiger Prozesse ermöglicht. Die Verwendung von AHL-Netzen erfordert jedoch in zwei Aspekten eine weitere Betrachtung, die eine direkte Verwendung und Analyse der Prozesse in dynamischen Koalitionen erschweren:

1. AHL-Netze erlauben über die Integration von algebraischen Spezifikationen die Verwendung komplexer Datentypen, jedoch keine direkte Verwendung von VDM als Beschreibungssprache. Hierzu muss zunächst eine Übersetzung des VDM-Modells in eine algebraische Spezifikation vorgenommen werden, was insbesondere Fragen bei der Abbildung von Bedingungen und Abhängigkeiten aufwirft, da diese im Rahmen der algebraischen Spezifikation nur über Gleichungen abgebildet werden können, die bei hinreichender Komplexität des VDM++-Modells eine Abbildung des Modells als algebraische Spezifikation verhindern können. Neben dieser Problematik stellt eine Übersetzung von VDM in eine algebraische Spezifikation auch stets eine zusätzliche Ebene dar, die bei einer formalen Handhabung hinsichtlich der Korrektheit berücksichtigt werden muss. So muss insbesondere formal sichergestellt und geprüft werden können, ob das VDM-Modell semantisch äquivalent zur algebraischen Spezifikation ist, um weiterhin die Möglichkeit zu haben, Aussagen über die dynamische Koalition aus Aussagen über das AHL-Netz zu deduzieren. Eine solche Betrachtung erfordert somit ein zusätzliches Maß an formalem Aufwand.

2. Die Frage der Analyse von AHL-Netzen und insbesondere offenen AHL-Netzen in Hinblick auf Eigenschaften wie *Liveness* oder *Safety* ist nicht absolut zu beantworten, da durch die freie Verwendung komplexer algebraischer Spezifikationen und durch die offenen Stellen schnell die Grenzen nicht nur der praktischen sondern auch der allgemeinen Berechenbarkeit erreicht werden. In der Literatur beschriebene Methoden sehen unter gewissen Voraussetzungen die Konvertierung eines AHL-Netzes in ein Standard Petrinetz vor, für das die Frage von *Liveness* oder *Safety* Eigenschaften im Falle von endlichen Petrinetzen algorithmisch erfassbar ist. Durch die Komplexität der AHL-Netze ist jedoch nicht sichergestellt, dass das entstehende Petrinetz ein endliches Netz ist. Aufgrund der Komplexität des AHL-Formalismus existiert keine allgemeine algorithmische Vorgehensweise für die Ermittlung von *Liveness*- und *Safety*-Eigenschaften. Es verbleibt somit nur die Möglichkeit zu versuchen, diese Fragen für jedes VDM-Modell eigenständig neu zu analysieren und zu beantworten.

Im Gegensatz zu den AHL-Netzen sind Abstract State Machines als Konzept weniger auf die Erfassung konkreter Probleme ausgelegt, sondern bieten stattdessen eine Methode, die ein breites Spektrum an Anwendungen erfassbar macht. Hierbei sind ASMs nicht allein auf die Modellierung von Rechner bezogenen Prozessen festgelegt, sondern sind darüber hinaus frei in der Modellierung zusätzlicher menschlicher Aspekte oder allgemeiner Probleme der realen Welt (vergleiche zum Beispiel [Bör03]). ASMs zeichnen sich hierbei durch eine sehr freie Vorgehensweise aus, die es dem Modellierer erlaubt, frei aus verschiedenen zur Verfügung stehenden Konzepten auszuwählen und diese Konzepte innerhalb der ASMs zu benutzen. In Hinblick auf die Fragestellung dieser Arbeit ist insbesondere die Verwendung von VDM als ein solches zu integrierendes Konzept vorstellbar. ASMs stellen ein mathematisches Framework zur Verfügung, welches eine formal verlässliche Modellierung von prozessualen Abläufen erlaubt. Sind die sogenannten *abstract states* einer ASM im Prinzip mit den Zuständen, in denen sich ein AHL-Netz befinden kann, gleichzusetzen, so sind die Schaltbedingungen in ASMs im Gegensatz zu den AHL-Netzen dezidierter beschreibbar und näher an übliche Konzepte der Programmierung angelehnt. Allgemein können ASMs somit als eine Art der Erweiterung von *Finite State Machines* um beliebige abstrakte Datentypen verstanden werden.

Im Gegensatz zu AHL-Netzen erlauben ASMs eine direktere Verwendung von VDM als Modellierungssprache. Durch die sehr breite Anwendbarkeit und Allgemeinheit von ASMs sind jedoch spezielle Eigenschaften wie *Liveness* oder *Safety* im allgemeinen Fall wiederum schwerer zu erfassen und bedürfen eines eigenständigen Studiums spezieller ASMs über VDM-Spezifikationen. Insbesondere hat sich in der Forschungsgemeinde zu ASMs eine Diskussion über die praktische Anwendbarkeit von ASMs eröffnet, die verschiedene Ideen zur Einschränkung des Formalismus auf berechenbare Algorithmen nach sich gezogen haben. Diese Einschränkungen jedoch nehmen dem Konzept auch die Offenheit und Allgemeinheit, auch algorithmische Aspekte der realen Welt, die nicht implementierbar sind, in die Modellierung mit einzubeziehen. Die Offenheit von ASMs zur natürlichen Modellierung von Aspekten außerhalb der reinen Systembetrachtung ist aufgrund des speziellen Szenarios in medizinischen Abläufen jedoch ein entscheidender Vorteil des ASM-Ansatzes. Abschließend lässt sich somit feststellen, dass sowohl AHL-Netze

als auch ASMs als formale Methode für die Modellierung der Ziele dieser Arbeit angesehen werden können, wobei ASMs insbesondere in Fragen der Modellierung von nicht rein systemisch bedingten Aspekten die deutlich natürlicheren Ansätze bieten. In Hinblick auf die Analyse wichtiger Eigenschaften wie *Liveness* und *Safety* sind hingegen für jedes konkretere Modell eigenständige Überlegungen zum Nachweis der Eigenschaften vorzunehmen.

6 Fazit und Ausblick

Wir haben in der vorliegenden Arbeit ein Modell in der formalen Spezifikationspsprache VDM vorgestellt, welches dynamische Koalitionen im medizinischen Bereich beschreiben kann und hierbei insbesondere Fragen des Access Controls durch Verwendung entsprechender Policies miteinbezieht. Unser Beitrag findet sich dabei vor allem in der Integration zweier bereits existierender VDM-Formalisierungen zu Dynamischen Koalitionen bzw. XACML Access-Control-Policies. Wir haben anhand eines kurzen, fiktiven Beispiels gezeigt, dass sich das Modell als Testumgebung für Access-Control-Policies und deren Auswirkungen in dynamischen Koalitionen eignet.

Als einen nächsten Schritt möchten wir das Modell erweitern, so dass kontext-sensitives XACML abgebildet werden kann. Dabei handelt es sich um die Miteinbeziehung von Umgebungsinformationen (z.B. Uhrzeit oder Zugriffsort) bei der Ausrechnung von Regeln und Requests. Des Weiteren möchten wir versuchen, das XACML-Profil für rollenbasierte Zugriffskontrolle (*role based access control/RBAC*) zu implementieren, welche für eine praktische Anwendung vor allem im medizinischen Bereich unverzichtbar ist.

Unsere bisherigen Arbeiten modellieren nur strukturelle Aspekte dynamischer Koalitionen. Testabläufe und Simulationen werden durch externe Skripte und Testklassen gesteuert. In unserem Ausblick schildern wir einige grundlegende Ideen, die Prozessabläufe innerhalb unseres Modells zu formalisieren. Dabei stellen wir mit den *Algebraic High Level Nets* und den *Abstract State Machines* zwei Formalisierungstechniken vor und diskutieren knapp Vor- und Nachteile ihrer Anwendung für unser Szenario.

Als nächster Schritt steht nun ausgehend von den in dieser Arbeit vorgestellten Ideen und Beispielen die Anwendung des Modells auf ein hinreichend großes und repräsentatives Testszenario an, das als Ausgangspunkt für eine formale Betrachtung der bei einem solchen Szenario anfallenden Prozessabläufe dienen kann. Zu untersuchen sind hierbei insbesondere Fragen des zu verwendenden Formalismus sowie Fragen nach der Möglichkeit einer algorithmischen Untersuchung der Prozessabläufe auf bestimmte Eigenschaften wie *Safety* oder *Liveness*.

Literatur

[BF07] Jeremy Bryans und John Fitzgerald. Formal Engineering of XACML Access Control Policies in VDM++. In Michael Butler, Michael Hinchey und María Larrondo-Petrie,

Hrsg., *Formal Methods and Software Engineering*, Jgg. 4789 of *Lecture Notes in Computer Science*, Seiten 37–56. Springer Berlin / Heidelberg, 2007.

[BFG+08] J. W. Bryans, J. S. Fitzgerald, D. Greathead, C. B. Jones und R. J. Payne. A Dynamic Coalitions Workbench: Final Report. Technical report, Newcastle University, 2008.

[BFJ+06] J. W. Bryans, J. S. Fitzgerald, C. B. Jones, I. Mozolevsky, Jeremy W. Bryans, John S. Fitzgerald, Cliff B. Jones und Igor Mozolevsky. Dimensions of dynamic coalitions. Bericht, 2006.

[BFJM06] J. W. Bryans, J. S. Fitzgerald, C. B. Jones und I. Mozolevsky. Formal modelling of dynamic coalitions, with an application in chemical engineering. Bericht, Newcastle University, 2006.

[BFP06] J. W. Bryans, J. S. Fitzgerald und P. Periorellis. Model Based Analysis and Validation of Access Control Polcies. Bericht, Newcastle University, School of Computing Science, 2006.

[BFRRM09] Laura Bocchi, José Luiz Fiadeiro, Noor Rajper und Stephan Reiff-Marganiec. Structure and Behaviour of Virtual Organisation Breeding Environments. In *FAVO*, Seiten 26–40, 2009.

[Bör03] Egon Börger. Abstract State Machines: A Method for High-Level System Design and Analysis, 2003.

[FL06] John S. Fitzgerald und Peter Gorm Larsen. Triumphs and Challenges for Model-Oriented Formal Methods: The VDM++ Experience (Abstract). In *ISoLA*, Seiten 1–4, 2006.

[FL09] John S. Fitzgerald und Peter Gorm Larsen. *Modelling Systems - Practical Tools and Techniques in Software Development (2. ed.)*. Cambridge University Press, 2009.

[FLM+05] John Fitzgerald, Peter Gorm Larsen, Paul Mukherjee, Nico Plat und Marcel Verhoef. *Validated Designs For Object-oriented Systems*. Springer-Verlag TELOS, Santa Clara, CA, USA, 2005.

[GKOT00] Yuri Gurevich, Philipp W. Kutter, Martin Odersky und Lothar Thiele, Hrsg. *Abstract State Machines, Theory and Applications, International Workshop, ASM 2000, Monte Verità, Switzerland, March 19-24, 2000, Proceedings*, Jgg. 1912 of *Lecture Notes in Computer Science*. Springer, 2000.

[HCA+09] Ali Nasrat Haidar, P. V. Coveney, Ali E. Abdallah, Peter Y. A. Ryan, B. Beckles, J. M. Brooke und M. A. S. Jones. Formal Modelling of a Usable Identity Management Solution for Virtual Organisations. In *FAVO*, Seiten 41–50, 2009.

[Klu98] R. Klueber. A framework for virtual Organizing. In *VoNet Workshop*, 1998.

[Let01] Nick Lethbridge. An I-Based Taxonomy of Virtual Organisations and the Implications for Effective Management. *Informing Science*, 4 No 1, 2001.

[LFB96] Peter Gorm Larsen, John Fitzgerald und Tom Brookes. Applying Formal Specification in Industry. *IEEE Softw.*, 13:48–56, May 1996.

[MST09] Jarred McGinnis, Kostas Stathis und Francesca Toni. A Formal Framework of Virtual Organisations as Agent Societies. In *FAVO*, Seiten 1–14, 2009.

[Rei10] Wolfgang Reisig. *Petrinetze: Modellierungstechnik, Analysemethoden, Fallstudien*. Leitfäden der Informatik. Vieweg+Teubner, 15 July 2010. 248 pages; ISBN 978-3-8348-1290-2.