ポリシに基づくWebサービス・コミュニティ連合のモデル

来間啓伸[†1,†2] 本位田真一[†3,†4]

Webサービスを仲介するシステムをクローズドな仲介システムの緩やかな連合によって実装する際には，次の3点が課題となる．(1) 連合によって個々の要素が受ける影響の局所化，(2) 連合関係の変化への柔軟な適応，(3) 連合によって不具合が起こらないことの検証．本稿では仲介システムを介して結ばれる要素の集まりをコミュニティと考え，要素間の静的なアクセス関係をポリシとして，コミュニティの連合を規定するためのポリシの枠組みを導入した．この枠組みに基づいて，連合によって組み合わされたポリシを実現するためのコミュニティ・モデルを示した．コミュニティ・モデル上では，コミュニティ間の協調のためのインタラクションとコミュニティ内のインタラクションを階層的に表現することで (1) と (2) が，ポリシとインタラクションが形式的に対応することで (3) が解決される．一方，コミュニティ・モデル記述にメタ階層に基づく言語を用いているため，実装との対応は明確ではない．ポリシとして表現する対象の拡大と，コミュニティ・モデルから実装への過程の明確化が，今後の課題である．

A Model for Federation of Service Communities on the Web

Hironobu Kuruma[†1,†2] and Shinichi Honiden[†3,†4]

The service mediation system on the Web could be constructed as a federation of service communities, in which each community provides and mediates limited number of services. In implementing the federation, (1) scalability of each community, (2) flexibility to the change of federation relation, and (3) verifiability of policy compliance should be considered. In this paper, we introduce a notion of policy of community based on access control among players and show a community model that is aimed at representing communications between players compliant with policy. The community model represents communications for the cooperation of communities separately from the communications for service request, mediation, and provision. As the result, it (1) represents communications between players in a modular way, (2) encapsulates the dependencies on partner communities, and (3) provides a basis for verification of policy compliance. The future work is to extend our notion of policy and to establish the implementation method based on the community model.

1. はじめに

インターネットの拡大，高性能な計算機の遍在化，プラットフォームや計算機言語に依存しないメッセージング技術の登場により，個々のシステムやソフトウェアの構成に依存しない「サービス」[4] をネットワークを通じて提供/利用する，Webサービスが実現しつつある．オープンなWeb環境では提供されるサービスがつねに変化しているため，利用者の要求に適合するサービスを探し出して利用者と提供者を動的に結合させるサービス仲介は，Webサービスの提供と利用のための中核的な役割を果たす．また，Web上に混在する粒度や信頼性の異なるサービスに対するアクセスをガイドする点でも，サービス仲介の役割は重要である．ここで，数が多く変化も激しいWebサービスを集中的な仲介システムですべてカバーすることは不可能であり，クローズドな仲介システムの緩やかな連合による実装が適切である．

連合したシステムでは，あるサービス仲介システムに属する利用者は，連合先のサービス仲介システムに属する提供者からのサービスを受けることができる．その際，個々のサービス仲介システムの制約は，連合したシステムでも基本的に保持されていることが求められる．たとえば，あるサービス仲介システムで利用者保護のために利用者と提供者の直接のアクセスを禁

†1 株式会社日立製作所システム開発研究所
Systems Development Laboratory, Hitachi, Ltd.
†2 総合研究大学院大学
The Graduate University for Advanced Studies
†3 国立情報学研究所
National Institute of Informatics
†4 東京大学
The University of Tokyo

じ，利用者へのサービス提供を仲介者が代行する形式をとっているとすると，他のサービス仲介システムの提供者とも間接的なアクセスが行われるようにシステムを構成する必要がある．しかし，このような制約をサービス仲介システム間で整合的に保つための調整は連合相手に依存し，連合関係の拡張や解消などの変化にも対応できるようシステムを構成することは難しい．

本稿では，Web サービスに関わる利用者，提供者，仲介者などの要素の間のアクセス関係の制約をポリシとし，ポリシに従ったサービス仲介システムの連合を構成するためのモデルを示す．モデル化することで，ポリシが組み合わさることによる複雑な構造のポリシ作成と，それを実現するサービス仲介システムの系統的な構築のための枠組みが提供できる．このため，まずコミュニティのポリシの概念を導入し，連合におけるアクセス関係を形式的に規定する．次に，要素間のインタラクションを階層的に表現するコミュニティ・モデルを導入し，サービス仲介システム間の協調をコミュニティ・モデル上に記述する．さらに，コミュニティ・モデルがコミュニティのポリシを実現していることを形式的に検証するための枠組みを示す．なお，本稿ではインタラクションがポリシに違反しないときに，モデルがポリシを実現するという．

以下，2 章では本稿のモデルを構成する基本要素について述べ，3 章で本稿で扱うポリシを定義しコミュニティの連合におけるポリシの保持について述べる．4 章では，コミュニティの連合を例に基づいて示す．5 章では連合の実現における課題について述べ，コミュニティ・モデル記述の枠組みを導入する．6 章で 4 章の例についてコミュニティ・モデルの例を示し，7 章で本研究のアプローチを評価する．8 章では関連研究について触れ，9 章でまとめる．

2. 基本構成要素

2.1 エージェント

本稿では，サービスの提供/利用/仲介に関わる実体をエージェントと呼ぶ．たとえば，サービスが計算機システムによって自動処理される場合にはその計算機システムを，人間が直接関与して処理される場合にはその人間をエージェントと見なす．以下では，すべてのエージェントの集合を A と表す．

2.2 ロール

エージェントがサービスの提供/利用/仲介に関して担う役割を，ロールと呼ぶ．あるサービスに関して，サービス提供者，利用者，仲介者は典型的なロールである．1 つのエージェントは複数のロールを担うことができる．以下ではすべてのロールの集合を R とし，エージェントとそれが担うロールの対応を関数 $g : A \to 2^R$ によって表す．

2.3 コミュニティ

コミュニティは，一定のロールを担うエージェントの集まりである．以下では，コミュニティを識別する名前の集合を C とし，コミュニティ $c \in C$ のロールの集合を R_c と表す．$R_c \subseteq R$ である．名前が $c \in C$ であるコミュニティ A_c は，次のように表される．

$$A_c = \{x | g(x) \cap R_c \neq \emptyset\}$$

1 つのコミュニティの中では，1 つのエージェントが複数のロールを同時に担うことはないとする．すなわち，任意の $a \in A$，$c \in C$ に対して，$X, Y \in R_c$ かつ $X, Y \in g(a)$ となる X，Y が存在するならば，$X = Y$．

3. ポリシ

3.1 エージェントのポリシ

任意のエージェント a，b について，a が b にメッセージを送り b がそれに基づいて処理を行うとき，本稿では a から b へのアクセス関係があるという．サービスの提供/利用/仲介ではアクセスの主体と対象がともに自律的であり，対象となるエージェントが主体に応じて処理内容を選択しうるが，本稿ではアクセス関係の単純な有無のみに基づいてポリシを考える．

エージェントのポリシを，エージェント間の 2 項関係 $p \subseteq A \times A$ と定義する．

p は個々のエージェントのアクセス関係を表し，$x, y \in A$ である x，y について $(x, y) \in p$ ならば，x から y へアクセス可能であるとする．エージェントのポリシは，たとえば契約関係の有無など，エージェント個別の制約を表す．

3.2 コミュニティのポリシ

任意のコミュニティ $c \in C$ に対して，コミュニティのポリシ p_c を R_c の間の 2 項関係 $p_c \subseteq R_c \times R_c$ と定義する．

p_c はロール間のアクセス関係を表し，任意の $X, Y \in R_c$ に対して $(X, Y) \in p_c$ ならば X から Y へアクセス可能であるとする．コミュニティのポリシは，エージェントがコミュニティに参加する際に結ぶ契約などに記載される，ロールを担うエージェントに対する包括的な制約を表す．コミュニティ内のエージェントのアクセス可能性は，次のように表せる．

エージェントのポリシが p，コミュニティ c のポリシが p_c のとき，任意の $x, y \in A$ について，$(x, y) \in p$ かつ $X \in g(x)$，$Y \in g(y)$，$(X, Y) \in p_c$ となる X，

Y が存在するとき，かつそのときに限って，コミュニティ c において x から y へアクセス可能である．

3.3 コミュニティの連合

複数のコミュニティを融合し，各コミュニティのロールを包含するコミュニティを構成することを，本稿ではコミュニティの連合と呼ぶ．連合によって構成されたコミュニティのポリシは，3.2 節のコミュニティのポリシで表現される．連合の際に各コミュニティが提示するアクセス関係の静的な委譲を，連合のポリシとする．連合のポリシは，連合したコミュニティのポリシを定める際に，ポリシの保持を判定するための基準となる．

3.3.1 連合のポリシ

コミュニティ m と n が連合する際の m から n へのアクセス関係の委譲を，ロール間の 2 項関係 $P_{mn} \subseteq R_m \times R_n$ によって定め，連合のポリシと定義する．

$(X, Y) \in P_{mn}$ は，コミュニティ m のロールに対して X が持つアクセス関係を，コミュニティ n のロール Y に委譲することを意味する．

3 つ以上のコミュニティが連合する場合には，アクセス関係の委譲は推移的であると定める．連合のポリシの推移閉包を $P^\dagger$ とする．$P^\dagger$ は，連合に含まれるすべてのコミュニティの組 m と n に対して以下を満たす最小の集合である．

- $P_{mn} \subseteq P^\dagger$ かつ $P_{nm} \subseteq P^\dagger$
- 任意のロール X，Y，Z に対して，$(X, Y) \in P^\dagger$ かつ $(Y, Z) \in P^\dagger$ ならば，$(X, Z) \in P^\dagger$

$(X, Y) \in P^\dagger$ は，X が各コミュニティのロールに対して持つアクセス関係を Y に委譲することを意味する．

3.3.2 コミュニティのポリシの保持

m 個のコミュニティ $c_1, ..., c_m$ が連合してコミュニティ c を構成するとする．$c_1, ..., c_m$ のポリシをそれぞれ $p_1, ..., p_m$，c のポリシを p_c とすると，p_c が以下を満たすとき c のポリシは連合のポリシ $P^\dagger$ のもとで $c_i (1 \leq i \leq m)$ のポリシを保持するという．

- $p_i \subseteq p_c$
- c_i のロール Y に対して $(X, Y) \in p_c$ ならば，$(X', X) \in P^\dagger$ かつ $(X', Y) \in p_i$ である X' が存在する．

4. コミュニティの連合の例

4.1 ポリシの異なるコミュニティの例

ある研究所で教授が Web を通じて研究情報を共有するコミュニティA を考える．A の利用者は仲介者に希望する情報の種類を伝え，情報提供を依頼する．仲介者は適切な提供者にアクセスして情報を受け取り，利用者に伝える．A では利用者と提供者の匿名性確保のため，これ以外のアクセスは許さないものとする．この様子を，**図 1** に示す．図中のメッセージ左側の数字は，メッセージの順序を表す．ここで，仲介者は適切な提供者を探す際に自分自身に陽にメッセージを送るものとした．利用者と提供者はコミュニティに加わる際に情報の授受に関する包括的な規約に合意するものとすると，この過程は個々のエージェントによらないロール間のアクセス関係ととらえることができる．利用者，仲介者，提供者のロールを各々 $A.R$，$A.M$，$A.P$ とすると，A のポリシは，次のものである．

$$p_A = \{(A.R, A.M), (A.M, A.P), (A.M, A.M)\}$$

ここで，$(A.R, A.M)$ はメッセージ `broker` とそれに対する返信 `result`，$(A.M, A.P)$ は `service_request` と `result`，$(A.M, A.M)$ は `recommend` と `answer` による．

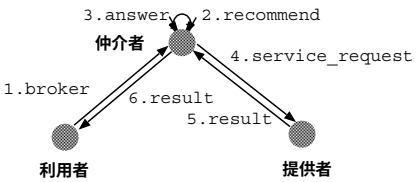

図 1 ブローカレッジ
Fig. 1 Brokerage.

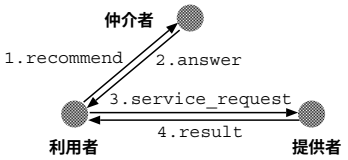

図 2 リコメンデーション
Fig. 2 Recommendation.

一方，大学院生の情報共有を行うコミュニティB では，**図 2** のように，利用者が仲介者に希望する情報の種類を示し，仲介者から指示された提供者にアクセスして情報を得るものとする．利用者，仲介者，提供者のロールを各々 $B.R$，$B.M$，$B.P$ とすると，B のポリシは，次のものである．

$$p_B = \{(B.R, B.M), (B.R, B.P)\}$$

4.2 連合のポリシ

コミュニティA と B の連合では，A の利用者が A と同様の匿名性の下に B の情報も利用できることが求められているとする．この場合，A の仲介者は B の利用者のアクセス関係を委譲される必要がある．したがって，B から A への連合のポリシを次のようにとる．

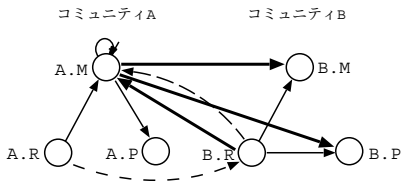

図 3 連合したコミュニティのポリシ
Fig. 3 Policy of federation.

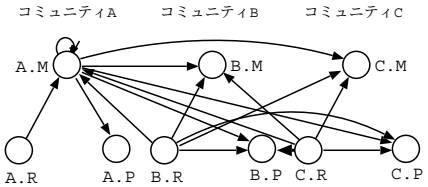

図 4 連合関係の連鎖
Fig. 4 Policy of chained federation.

$$P_{BA} = \{(B.R, A.M)\}$$

一方，B の利用者が A の情報も利用できることが求められているとすると，B の利用者は A の利用者のアクセス関係を委譲される必要がある．したがって，

$$P_{AB} = \{(A.R, B.R)\}$$

これらより，

$$P^{\dagger} = P_{AB} \cup P_{BA} \cup \{(A.R, A.M)\}$$

P_{AB} と P_{BA} は連合の枠組みを規定するため，他の値を設定すれば異なるアクセス関係が許容される．たとえば，

$$P'_{BA} = \{(B.R, A.R)\}$$

ととれば，コミュニティのポリシで A の利用者から B の仲介者，提供者へのアクセスを許すことが可能になる．

4.3 連合したコミュニティのポリシ

拡張したポリシに対して，A と B の連合によって構成されるコミュニティを D とし，そのポリシ p_D を以下のように設定する．

$$p_D = p_A \cup p_B \cup \{(A.M, B.M), (A.M, B.P), (B.R, A.M)\}$$

このポリシは，A と B のポリシを保持したポリシであるための 3.3.2 項の条件を満たす．すなわち，

- $p_A \in p_D$ かつ $p_B \in p_D$
- $(A.M, B.M) \in p_D$ に対して $B.R$ が存在し，$(B.R, A.M) \in P^{\dagger}$ かつ $(B.R, B.M) \in p_B$
- $(A.M, B.P) \in p_D$ に対して $B.R$ が存在し，$(B.R, A.M) \in P^{\dagger}$ かつ $(B.R, B.P) \in p_B$
- $(B.R, A.M) \in p_D$ に対して $A.R$ が存在し，$(A.R, B.R) \in P^{\dagger}$ かつ $(A.R, A.M) \in p_A$

D のポリシを**図 3** に示す．この図で，丸はロールを，破線の矢印は連合のポリシを表す．実線の矢印はコミュニティのポリシを示し，始点が X，終点が Y ならば，$(X, Y) \in p_D$ である．

簡単のため，上記の例では A，B ともに利用者/提供者のロールを 1 つずつとしたが，複数のロールを設けることでより詳細なポリシも設定できる．

4.4 連合関係の連鎖

連合により構成されたコミュニティがさらに別のコミュニティと連合することにより，連合関係の連鎖が起こる．たとえば，4.3 節のコミュニティ D が，学部生の間の情報共有を行うコミュニティ C と連合する場合を考える．C では B と同様の手順で情報へのアクセスが行われるものとする．利用者，仲介者，提供者のロールを各々 $C.R$，$C.M$，$C.P$ とすると，C のポリシは次のようになる．

$$p_C = \{(C.R, C.M), (C.R, C.P)\}$$

C と D の連合にあたって，A の仲介者と B の利用者には C の利用者のアクセス関係が委譲され，C の利用者には A と B の利用者のアクセス関係が委譲されるとすると，

$$P_{CD} = \{(C.R, B.R), (C.R, A.M)\}$$
$$P_{DC} = \{(A.R, C.R), (B.R, C.R)\}$$

C と D の連合によって構成されるコミュニティを E とすると，次の p_E は C と D のポリシを保持したポリシである．

$$p_E = p_D \cup p_C \cup \{(C.R, A.M), (C.R, B.M), (C.R, B.P), (A.M, C.M), (A.M, C.P), (B.R, C.M), (B.R, C.P)\}$$

E のコミュニティのポリシを**図 4** に示す．

5. コミュニティ・モデル

5.1 連合の構成における課題

コミュニティを実現する際には，ポリシに従ったインタラクションを行うエージェントが実現されていなければならない．しかし連合の場合には，連合先のエージェントにアクセスするための多様なインタラクションを個々のエージェントに実現することは困難である．連合したコミュニティを効率的に構成するためには，以下の課題を解決する必要がある．

(1) 連合関係がエージェントに及ぼす影響の局所化
4 章の例では，大学院生からは提供者が教授か大学院生か学部生かによってアクセス先が変わるが，それを大学院生に意識させないシステム

とすることが望ましい．

(2) 連合関係の変化に柔軟に対応するための枠組みの提供
学部生間の情報共有コミュニティが連合する際に，すでに存在する教授/大学院生間の情報共有コミュニティのシステムを大きく変えないですむことが望ましい．

(3) 各コミュニティのエージェントの振舞いがコミュニティのポリシを損なわないことの保証
連合によってアクセス関係が複雑化した場合にも，教授へのアクセスが間接的であることが確認できるようシステムが設計されている必要がある．

(1) はスケーラビリティへの対応の点からの課題である．エージェントは所属するコミュニティのポリシに従う必要があるため，コミュニティが連合する際には連合先のロールの組合せに対してアクセス関係を調整することが求められる．多くのエージェントがスケーラブルに協調するためには，このような調整を個々のエージェントとは可能な限り独立に行うための仕組みを用意し，連合関係がエージェントに与える影響を吸収することが必要となる．

(2) はオープン環境への適応の点からの課題である．Web サービス・システムでは提携関係の変化などによる連合関係の変更や拡張がしばしば起こりうるので，それに対応する際のシステムの変更が少ないことが求められる．このためには，連合関係の変更に対してシステムの特定箇所の変更で対応できるよう設定された枠組みが必要である．

(3) は検証の点からの課題であり，システムが連合したコミュニティのポリシを遵守していることの保証が求められる．そのためには，連合したシステムの仕様レベルで検証できる枠組みが必要である．

5.2 実現プロセス

本稿では 5.1 節の課題を解決するため，コミュニティ・モデルを導入する．**図 5** は，コミュニティ・モデルを介したコミュニティの連合の実現プロセスである．この実現プロセスでは，個々のサービス仲介システムをサービスの提供/利用/仲介に直接関わるインタラクションとコミュニティの協調に関わるインタラクションの点から抽象化してコミュニティ・モデルを構成した後，それを具象化することで連合したシステムを構成する．具象化プロセスでは，コミュニティ・モデルに表現されたコミュニティ間の協調のためのインタラクションを抽出し，システム間を結合するモジュールの機能として実装する．この意味で，コミュニティ・モデルは連合システムを構成するための機能仕様となる．

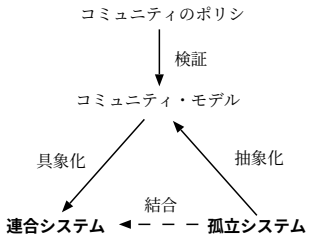

図 5 連合したシステムの実現プロセス
Fig. 5 Policy implementation process.

5.3 コミュニティ・モデルの記述言語

コミュニティ・モデルを記述するための枠組みとして，場とメタ階層を用いた記述言語[9] を用いる．構文の概要を付録に示す．

この言語では，システムをエージェントと場の集まりとして表現する．1 つのエージェント機能は，0 レベルから n レベルまでの $n+1$ 階層に分けて記述する．ここで，n は陽に記述されたメタ階層の上限であり，n の値は記述の必要に応じて増やすことができる．付録の構文では，<エージェント記述子>の前の m の数がメタ階層のレベルを表す．簡単のため，以下では 0 レベルをベースレベル，1 レベルをメタレベルと呼ぶ．各レベルは，以下の機能を持つ．

- そのレベルの内部状態の更新
- 同一レベルのメッセージ送/受信

i レベルのメッセージは，$i+1$ レベルからは<項> `dequeue` によってデータとして取り出し，操作することができる．i レベルのメッセージ送/受信は，このデータを $i+1$ レベルで受け渡し，受信側エージェントで `execute` を適用して実行することで行われる．すなわち，$i+1$ レベルのメッセージ送/受信を通じて i レベルのメッセージ・データを受け渡すことと，i レベルでメッセージ送/受信を行うことを同一視する．

場は，エージェントの空間的な影響範囲を限定する．エージェントは 1 つ以上の場に属し，n レベルではエージェントがメッセージ送/受信できるのは同一場内のエージェントに限られる．

言語の意味を，以下に簡単に述べる．ここでは，メタ階層のレベルを示す 0 以上の整数の集合を N，メッセージ・パターンの集合を M，場の集合を F とする．

5.3.1 メッセージの実行

$i \leq n$ レベルのメッセージ処理を示す関数

$$acceptable : M \times A \times N \rightarrow bool$$

を次のように定義する．

- エージェント a の i レベルに m を処理するメ

ソッドが記述されているとき
$acceptable(m, a, i) = true$
- それ以外のとき
$acceptable(m, a, i) = false$

i レベルのメッセージは $i+1$ レベルから操作可能であるため，メッセージ実行を示す関数
$executable : M \times A \times N \rightarrow bool$
を次のように定義する．
- $i = n$ のとき
$executable(m, a, i) = acceptable(m, a, i)$
- $i < n$ のとき
$i+1$ レベルで execute が陽に適用されるか，標準メソッドが用いられる場合，
$executable(m, a, i) = acceptable(m, a, i)$
$i+1$ レベルからの操作により i レベルで m が m' に書き換えられる場合，
$executable(m, a, i) = executable(m', a, i)$
それ以外の場合，
$executable(m, a, i) = false$

5.3.2 メッセージの伝達

場 f_1 にあるエージェント a_1 が，場 f_2 にあるエージェント a_2 に i レベルでメッセージ m を送信する場合を考える．このメッセージは，$i+1$ レベルからの操作によって a_2 以外のエージェントに渡されて実行されうる．場 f にあるエージェント a へのメッセージ伝達を示す関数
$transfer : M \times (A \times F) \times (A \times F) \times N \rightarrow bool$
は次のように定義することができる．

$$transfer(m, (a_1, f_1), (a, f), i) = forward(wrap(m), (a_1, f_1), (a, f), i+1) \wedge executable(m, a, i)$$

i レベルのメッセージ m_i に対して，$wrap(m_i) \in M$ は m_i に対応するデータを $i+1$ レベルで受け渡すメッセージを表す．$i+1$ レベルに記述がない場合には標準メソッドを仮定して，すべての $a \in A$ に対して $acceptable(wrap(m_i), a, i+1) = true$ とする．

$forward : M \times (A \times F) \times (A \times F) \times N \rightarrow bool$
は上位レベルでのメッセージ受け渡しを示す関数で，場 f_1 のエージェント a_1 から場 f_2 のエージェント a_2 への $i+1$ レベルのメッセージ m_{i+1} に対して，次のように定義できる．
- $i = n$ のとき
$f = f_1 = f_2$ かつ $a = a_2$ の場合，
$forward(m_{i+1}, (a_1, f_1), (a, f), i+1) = true$
それ以外の場合，
$forward(m_{i+1}, (a_1, f_1), (a, f), i+1) = false$
- $0 \leq i < n$ のとき
メッセージを転送するエージェント a_3 が存在する場合，

$$forward(m_{i+1}, (a_1, f_1), (a, f), i+1) = forward(m_{i+1}, (a_1, f_1), (a_3, f_3), i+1) \wedge transfer(m_{i+1}, (a_3, f_4), (a, f), i+1)$$

それ以外の場合，

$$forward(m_{i+1}, (a_1, f_1), (a, f), i+1) = transfer(m_{i+1}, (a_1, f_1), (a, f), i+1)$$

ここで f_3 と f_4 は a_3 が所属する場で，a_3 が複数の場に属するとき $f_3 \neq f_4$ であってもよい．

5.4 コミュニティのポリシとの対応

コミュニティ・モデル上でのロール間のアクセス関係 Acc を，次のように定義する．

$$Acc = \{(X, Y) | \exists a, b \in A(\exists u, v \in F(\exists m \in M (X \in g(a) \cap R_u \wedge Y \in g(b) \cap R_v \wedge transfer(m, (a, u), (b, v), 0))))\}$$

ここで，R_u と R_v は各々場 u と v に対応するコミュニティのロールの集合とする．

コミュニティのポリシ p_c に対して
$Acc \subseteq p_c$
ならば，コミュニティ・モデルは p_c の実現である．

6. コミュニティ・モデルの例

6.1 コミュニティ・モデル記述の過程

コミュニティ・モデルを記述するための抽象化プロセスは，基本的に以下の段階からなる．

(1) サービスの提供/利用/仲介のための基本的なインタラクションのモデル化
(2) コミュニティの協調に関わるインタラクションのモデル化
(3) 連合したシステムでのサービスの提供/利用/仲介のためのインタラクションのモデル化

4 章のコミュニティA，B，C にそれぞれ場 A，B，C を割り当て，各コミュニティの典型的なエージェントを記述する．エージェントのベースレベルの記述は，以下のようになる．
- 仲介者，利用者，提供者を表すエージェントは，A の場合には図 1，B，C では図 2 のメッセージ授受を行う．

次に，コミュニティ間の協調に関わる以下の機能を仲介者のメタレベルに記述する．
- 受信エージェントが場内にないベースレベル・メッセージを仲介者間で転送する．
- A の仲介者は，自身が受信者である場合を除いて場内の受信者に対する場外からのベースレベル・

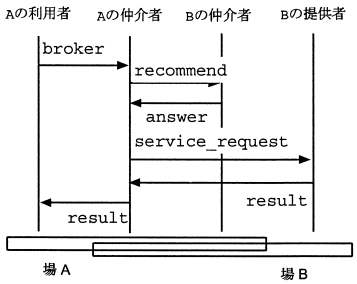

図 6 A から B へのメッセージ・シーケンス
Fig. 6 Message sequence (A to B).

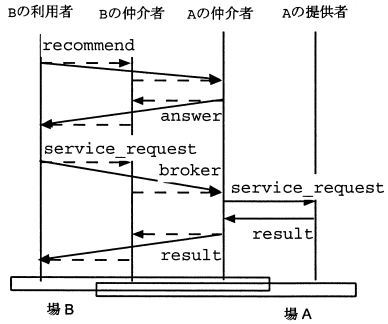

図 7 B から A へのメッセージ・シーケンス
Fig. 7 Message sequence (B to A).

メッセージは失敗させる．

- B と C の仲介者は，受信エージェントが場内にあるベースレベル・メッセージを受信エージェントに届ける．

これに，連合したシステムでのサービスの提供/利用/仲介のためのインタラクションを追加する．4 章のコミュニティA，B，C では仲介者間で仲介を委託するため，各仲介者のメタレベルに以下が追加される．

- ベースレベルの `recommend` メッセージを受け取った場合，それをベースレベルで実行して，要求されたサービスが場内にあるときはベースレベルの `answer` メッセージを返す．要求されたサービスが場内にないときには，`recommend` メッセージを転送して連合先の仲介者に推薦要求を委託する．

このようにエージェントを構成し，A の仲介者が同時に場 B に属するとともに B の仲介者が場 A に属することで，4.1 節の連合関係がコミュニティ・モデル上で表現される．また，A の仲介者がさらに場 C に属し，C の仲介者が場 A に属することにより，4.4 節の連合関係が表現される．

6.2 インタラクションの概要

図 6 は，4.1 節のコミュニティA の利用者がコミュニティB の提供者のサービスを利用する際のインタラクションの概略である．同様に，**図 7** は，4.1 節のコミュニティB の利用者がコミュニティA の提供者のサービスを利用する際のインタラクションの概略である．ここで，実線の矢印はベースレベルのメッセージ送/受信を，破線の矢印はメタレベルのメッセージ送/受信を表す．

6.3 課題との対応

5.1 節であげた課題 (1)，(2)，(3) は，コミュニティ・モデルによって各々以下のように解決される．

6.3.1 連合関係に対するエージェントの独立性

連合にともなうインタラクションの調整の大部分は複数の場に属するエージェントが行うため，モデル上では連合関係の有無によって個々のエージェントが受ける影響は少ない．たとえば，6 章の例では連合先の個々のコミュニティへの依存性は複数の場に属する仲介者に局所化されている．各エージェントは，連合元のコミュニティのポリシに従うことで連合先のコミュニティのポリシにも適合することになり，連合のために担う調整は限定的である．この結果，図 7 のように B の大学院生が仲介者に送った推奨依頼は A の仲介者に転送され，A の仲介者が見かけ上の提供者となることで情報が提供される．

コミュニティ・モデルでは，メタ階層を使って以下を表現できる．

- メッセージ送/受信先の変更
- 複数のメッセージを 1 つのメッセージに結合
- 1 つのメッセージを複数のメッセージに分割

これらの操作をさらに上位階層から操作することによって，より高次の操作も表現できる．

エージェントがポリシに従ったインタラクションを行うための調整は，たとえば各コミュニティにアクセス・コントローラを置いて，アクセス・コントローラ間で協調させることで具象化することができる．このとき，エージェントはアクセス・コントローラを介して他のエージェントとインタラクションすることになる．コミュニティ・モデルに表現された機能のうちコミュニティの協調に関わる部分は，アクセス・コントローラの機能を表現する．たとえば 6 章のモデルでは，複数の場に属する仲介者のメタレベルの機能がアクセス・コントローラの機能の中核部分を構成する．コミュニティB の具象化では，より単純に仲介者にアクセス・コントローラの機能を持たせることも考えられる．その場合，大学院生は B の仲介者の指示により，情報の提供を受けるために A の仲介者にアクセスするかもしれないが，これは B の仲介者によって制御されたアクセスである．

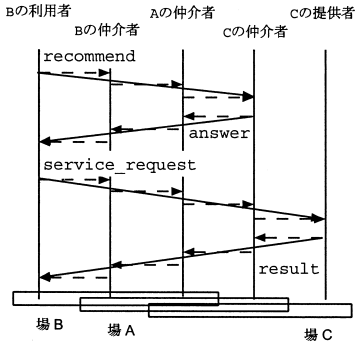

図 8 B から C へのメッセージ・シーケンス
Fig. 8 Message sequence (B to C).

6.3.2 連合関係の変化への対応

図 8 は，4.4 節の連合関係の連鎖においてコミュニティB の利用者がコミュニティC の提供者のサービスを利用する際の，インタラクションである．この図は，場 B の利用者によるベースレベル・メッセージ `service_request` が，各場の仲介者によって転送されてコミュニティC の提供者に送られることを表す．6.1 節のコミュニティ・モデルでは，このようなベースレベル・メッセージの転送をコミュニティ間に共通の方法でメタレベルで行う．この結果，4 章の教授/大学院生間の情報共有コミュニティに学部生間の情報共有コミュニティが連合する際にも，6.1 節に示したように少なくともモデル上は前者を大きく変更する必要はない．コミュニティ・モデルによって，以下が得られる．

- コミュニティ間の協調のための共通基盤
- 連合関係の変化による影響範囲の限定

一般に連合関係の連鎖が複雑になると，コミュニティ間のメッセージ転送経路が複数存在する場合が起こりうる．そのような場合でも，次に述べるように転送経路を解析することによって，ポリシ違反が起こらないかどうかを知ることができる．

6.3.3 コミュニティのポリシの保持

4.3 節に示したように，p_D はコミュニティA，B のポリシを保持したポリシである．したがって，コミュニティ・モデルに表現されたインタラクションが p_D を満たすことを示せばよい．

A の利用者，提供者，仲介者エージェントを各々 ar，ap，am，B のそれらを br，bp，bm と表すと，関数 $transfer$ の値が真になるのは以下の組合せのみである．

$transfer(broker, (ar, A), (am, A), 0)$
$transfer(recommend, (am, A), (am, A), 0)$
$transfer(recommend, (am, A), (bm, B), 0)$
$transfer(service\_request, (am, A), (ap, A), 0)$
$transfer(servive\_request, (am, A), (bp, B), 0)$
$transfer(recommend, (br, B), (bm, B), 0)$
$transfer(recommend, (br, B), (am, A), 0)$
$transfer(service\_request, (br.B), (bp, B), 0)$
$transfer(service\_request, (br, B), (am, A), 0)$

これらから 5.4 節に従って，

$$Acc = \{(A.R, A.M), (A.M, A.M), (A.M, B.M), (A.M, A.P), (A.M, B.P), (B.R, B.M), (B.R, B.P), (B.R, A.M)\}$$

これは，

$$Acc \subseteq p_D$$

であるため，D のポリシを満たす．したがって，このコミュニティ・モデルは連合 D のポリシの実現であることがいえる．すなわち，このようなインタラクションを行う B の大学院生は，教授に対して間接的なアクセスのみを行うことがいえる．

7. 議　　論

自律的なエージェントをスケーラブルに結合するためには，ポリシに基づく緩やかな協調のモデルは効果的であると考えられる．本稿ではコミュニティの連合を扱うため，個々のエージェントのポリシとは別にコミュニティのポリシを導入した．本稿のコミュニティ・モデルはコミュニティのポリシの実現を目的としており，各エージェントはロールの代表者として記述される．一方，同じ枠組みのもとで個々のエージェントをモデル化するアプローチもありうる．その際，ポリシ違反の動的な監視やエージェント間のネゴシエーションの記述において，より上位のメタ階層からの操作が有効であると予想される．

本稿で述べたコミュニティ・モデルは，コミュニティのポリシの実現において以下の効果を持つ．

- コミュニティ間の協調のための明示的な枠組みを，メタ階層によって与える．この結果，1 つのコミュニティが他のコミュニティに与える影響を明確にするとともに，連合関係が変化した際に変更すべき箇所が特定される．
- コミュニティ内外のインタラクションを，場によって分離する．この結果，連合関係への依存度が局所化され，コミュニティのサービス提供/利用/仲介の変更や拡張に柔軟に対応できる．

コミュニティ・モデル記述言語は強力な表現力を持つため，様々なシステムを多様な方法で表現することができる．たとえば 6 章のコミュニティA で利用者の場と提供者の場を分け，仲介者がそれらの場に同時

に属するようモデル化すれば，利用者と提供者の分離を明確にできる．その一方，メタに強く依存するモデルは具象化が困難になる．

8. 関連研究

ロール・ベース・アクセス制御（RBAC）の概念をインターネット上の計算環境に適用した例として，RBAC/Web[3] があげられる．RBAC/Web は，利用者による Web 上の情報へのアクセスを，利用者に割り当てられたロールに基づいて制御する．利用者の権限をロールに基づいて管理することで，機能の変更や拡張に柔軟なシステムを構成することができる．たとえば，機能拡張にともなって新たな権限を付与する場合には，ロールに対して権限を付与すれば十分であり，個々の利用者に個別に権限を付与する必要はない．その一方，RBAC/Web には本稿の連合のポリシの概念はなく，複数のコミュニティの連合を扱う枠組みは提供されない．このため，コミュニティ間の拡張に関する柔軟性は制約される．

UDDI（Universal Description, Discovery and Integration）[5] は，Web サービスの提供者が自らのサービスに関する情報を公表し，サービスの要求者が発見するための方法を与える．UDDI 3.0 では複数のレジストリ間の相互運用に関するインタフェースが加えられた一方，それらのインタラクションに関するポリシは仕様の対象外となっている[6]．複数の UDDI レジストリを含むサービス仲介システム間のインタラクションを表現するうえで，本研究のポリシの概念とコミュニティ・モデルは有効であろう．

1980 年代後半以降，メタレベル・アーキテクチャおよびリフレクションに関して多くの研究がなされている[1),8)]．また，エージェントの面からはリフレクションを使った自己再構成可能なエージェントに関する研究[2] などがあげられる．これらの研究の多くが実行時にメタレベルからの介入によって動作を変えるシステムの実現を目的とするのに対し，本研究ではメタ階層を使って記述したコミュニティ・モデルをもとに，人手であるいは半自動的に，実行可能なプログラムを作成するアプローチをとる．これは，本研究の目的がすでに実現されているシステムの結合にあり，各システムへの手直しを可能な限り行わないで，コミュニティ間の協調機能を結合モジュールないしはアクセス・コントローラとして実現することを意図していることによる．この結果，本研究のコミュニティ・モデルは基本的に人間が読み解析できるものであれば十分であり，実行可能であることが要求される場合に比べて記述の自由度を高く設定できる．その一方，コミュニティ・モデルからの具象化プロセスにはメタ記述からプログラムコードへの変換が含まれるので，実行可能なプログラムの作成は容易ではない．具象化プロセスおよび実装の正しさを検証する枠組みの明確化は，今後の課題である．将来リフレクション分野の研究が進めば，コミュニティ・モデルからラッパが生成され，自律的なエージェントによる直接実装が可能になることも考えられる．

9. おわりに

本稿では，サービス仲介に関わる要素間のアクセス関係の制約に基づいてポリシを定義し，ポリシを保持したコミュニティの連合を実現するための多階層のコミュニティ・モデルについて述べた．

一般に，オープンな環境にある要素を緩やかに結合する際に，ポリシに基づく制約は，好ましくない結合の排除において有効であることが期待される．しかし，要素数の増加やポリシの複雑化によって単一の管理システムで要素を管理することは困難になり，複数の管理システムの連合が求められる．本稿のアプローチは，このような連合の実現に関して以下の特徴を持つ．

- コミュニティを単位とするモジュラリティの向上
- コミュニティの協調構造の共通化による連合関係の変化への対応
- 連合したコミュニティのポリシに対する実現の正当性の検証

静的なアクセス関係からポリシの範囲を拡張することと，コミュニティ・モデルから多様な実装への具象化プロセスを明確にすることは，今後の課題である．

参考文献

1) Maes, P.: Issues in Computational Reflection, *Meta-Level Architectures and Reflection*, pp.21–35, Elsevier Science Publishers B.V., North-Holland (1988).
2) Charlton, P.: Self-Configurable Software Agents, *Advances in Object-Oriented Metalevel Architectures and Reflection*, pp.103–127, CRC Press (1996).
3) Kuhn, D., Barkley, J., Cincotta, V., Ferraiolo, D. and Gavriella, S.: Role Based Access Control for the World Wide Web, *Proc. 20th National Information Systems Security Conference*, pp.331–340 (1997).
4) Toyouchi, J., Funabashi, M. and Strick, L.: Service Integration Platform based on TINA 3-tier Model and Interfaces, *Proc. TINA 2000*

CONFERENCE, pp.21–26 (2000).
5) *UDDI Technical White Paper*, Universal Description, Discovery and Integration (2000). http://www.uddi.org
6) *The Evolution of UDDI*, UDDI.org White Paper (2002). http://www.uddi.org
7) Bishop, M.: *Computer Security*, Addison-Wesley (2003).
8) 渡部卓雄：リフレクション，コンピュータソフトウェア，Vol.11, No.3, pp.5–14 (1994).
9) 来間啓伸，大須賀昭彦，本位田真一：協調アーキテクチャに基づくソフトウェア・モジュールの仕様記述モデル，情報処理学会論文誌，Vol.37, No.6, pp.1171–1186 (1996).
10) 本位田真一，飯島　正，大須賀昭彦：エージェント技術，共立出版 (1999).

付　　録

コミュニティ・モデル記述言語の構文規則

```
<エージェント> ::=
  ‘‘{’’ エージェント記述子 ‘‘|’’
  [‘‘<attribute>’’ <属性>
    (‘‘,’’ <属性>)* ‘‘;;’’]
  [‘‘<method>’’ <メソッド>
    (‘‘,’’ <メソッド>)* ‘‘;;’’]
  ‘‘}’’
<エージェント記述子> ::=
  エージェント名
  | ‘‘m(’’ <エージェント記述子>‘‘)’’
<属性> ::=  変数名 ‘‘:’’ <項>
<メソッド> ::=
  <メッセージパターン>‘‘:’’ <手続き>
<メッセージパターン> ::=
  メッセージ識別子 <項>*
<手続き> ::= <文> (‘‘;’’ <文>)*
<文> ::= <条件文> | < case 文>
  | <代入文> | <式>
<条件文> ::= ‘‘if’’ <式>
  ‘‘then’’ ‘‘(’’ <手続き>‘‘)’’)+
  [‘‘else’’ ‘‘(‘‘<手続き>‘‘)’’]
< case 文> ::=
  ‘‘case’’ <式>‘‘of’’
  (<変数名パターン>‘‘:’’
    ‘‘(’’ <手続き>‘‘)’’)+
  [‘‘otherwise’’ ‘‘:’’
    ‘‘(’’ <手続き>‘‘)’’]
<代入文> ::= ‘‘let’’ <パターン式>
<式> ::=
  <メッセージ式> | <項> | <パターン式>
  | ‘‘for some’’ 変数名
    ‘‘with (’’ <式>‘‘)’’ <式>
  | ‘‘for all’’ 変数名
    ‘‘with (’’ <式>‘‘)’’ <式>
<メッセージ式> ::=
  <項>‘‘<-’’ <メッセージパターン>
<項> ::= ‘‘(’’ <式>‘‘)’’
  | <エージェント名パターン> | 変数名
  | ‘‘dequeue’’ | ‘‘execute(’’ 変数名‘‘)’’
<パターン式> ::=
  <パターン>‘‘=’’ <パターン>
<パターン> ::= <式> | 変数名パターン
<エージェント名パターン> ::=
  エージェント名 ‘‘.’’ 場の名前
```

(平成 15 年 10 月 14 日受付)
(平成 16 年 3 月 5 日採録)

来間　啓伸（正会員）

1958 年生．1981 年広島大学理学部物理学科卒業．1984 年同大学大学院理学研究科物理学専攻博士課程後期中退．同年（株）日立製作所入社．現在，同社システム開発研究所に所属．2003 年総合研究大学院大学入学，大学院博士後期課程に在学．主として，プログラミング言語，ソフトウェア工学，コンピュータセキュリティの研究に従事．日本ソフトウェア科学会，ACM，IEEE 各会員．

本位田真一（正会員）

1953 年生．1976 年早稲田大学理工学部電気工学科卒業．1978 年同大学大学院理工学研究科電気工学専攻修士課程修了.（株）東芝を経て 2000 年より文部科学省国立情報学研究所教授，現在に至る．2001 年より東京大学大学院情報理工学系研究科教授を併任，現在に至る．2002 年 5 月〜2003 年 1 月英国 UCL ならびに Imperial College 客員研究員（文部科学省在外研究員）．工学博士（早稲田大学）．1986 年度情報処理学会論文賞受賞．エージェント技術，オブジェクト指向技術，ソフトウェア工学の研究に従事．IEEE，ACM，日本ソフトウェア科学会等各会員．