パターンを利用したセキュアかつ効率的なモバイルエージェントアプリケーション開発

田原康之[†] 吉岡信和[††]
大須賀昭彦[†] 本位田真一[†††]

インターネットやイントラネットといった広域開放型ネットワークが拡大するにつれて，モバイルエージェント技術が注目を集めつつある．モバイルエージェントは，自律性，移動性，知性，協調性，あるいは即応性といった特徴により，開放型ネットワークにおける環境変化や多様な要求に対応することのできるソフトウェア単位である．しかし現在では，セキュリティと性能のトレードオフを解消可能で，モバイルエージェントの利用を考慮した分散アプリケーションの開発手法について十分検討されていないため，本技術はまだ広く普及するに至っていない．本論文では，セキュリティの問題と性能の問題のトレードオフを解消可能で，モバイルエージェントの利用も考慮した分散アプリケーション開発を支援するための，形式的枠組みを提案する．本枠組みでは，分散アプリケーションの設計において計算コストとセキュリティを考慮したモデルを構築し，パターンを組み合わせることによって，これらのモデルを満たすように分散アプリケーションを設計する．これにより，セキュリティと性能のトレードオフを考慮しながら，アプリケーション開発を行うことができる．また，モデルとパターンは形式的に定義されているので，生成された挙動がモデルを厳密に満足することを保証できる．さらに，パターンの組合せはあるアルゴリズムによって自動的に導出されるので，セキュリティ要求が変化した際にも容易にエージェントの挙動を修正することができる．

Secure and Efficient Mobile Agent Application Development Using Patterns

Yasuyuki Tahara,[†] Nobukazu Yoshioka,[††] Akihiko Ohsuga[†] and Shinichi Honiden[†††]

As wide-area open networks like the Internet and intranets grow larger, mobile agent technology is attracting more attention. Mobile agents are units of software that can deal with environmental changes and the various requirements of open networks through features such as autonomy, mobility, intelligence, cooperation, and reactivity. However, since the usual development methods of secure and efficient distributed applications considering mobile agents are not sufficiently investigated, the technology is not yet widespread. In this paper, we propose a formal framework that supports distributed application development considering mobile agents and resolving the trade-offs between the security issues and the performance issues. In our framework, we design the behavior of mobile agents by building models including the computational costs and the security policies, and combining patterns so that the combination satisfies the models. Therefore we can develop the application resolving the security and the performance trade-offs. Since the models and the patterns are presented according to a formal framework, we can make sure that the pattern combination satisfies the models rigorously. In addition, if the security policy changes, we can easily modify the behavior because the pattern combination can be figured out automatically by an algorithm.

1. はじめに

インターネットやイントラネットといった，広域開放型ネットワークが拡大するにつれて，急速な変化や多様な要求に対応できる分散システムへの需要が高まってきている．エージェント技術[3)]，特にモバイルエージェント技術は，そのようなシステムを開発するための基礎として期待されている．エージェントは，

† 株式会社東芝研究開発センター知識メディアラボラトリー
Knowledge Media Laboratory, Corporate Research and Development Center, Toshiba Corporation
†† 国立情報学研究所
National Institute of Informatics
††† 国立情報学研究所/東京大学
National Institute of Informatics/The University of Tokyo

自律性，移動性，知性，協調性，あるいは即応性といった特徴により，開放型ネットワークにおける環境変化や多様な要求に対応することのできるソフトウェア単位である．

近年はエージェントの実際的な適用が検討されつつあるため，エージェントシステム開発支援技術が現れてきている．特に，デザインパターン[8]をエージェントシステム開発に適用することが検討され始めている[1),12),15),16),22)]．デザインパターンは，ソフトウェア開発における過去の優れた経験を定式化し，明示的に表現することにより，それら優れたソフトウェア設計の再利用を容易にするものである．

一方，モバイルエージェント技術の実際的な適用においては，セキュリティ問題が最も重要であると考えられている[5)]．この問題への解として，多くの研究者が様々な手法を提案している[11),17),19),23)]．しかし，これらの技術がモバイルエージェントを利用した実際的なシステムの開発に適用できるかどうかは，まだ明らかでない．主な問題点として，計算コストとセキュリティの向上とはトレードオフの関係にあるため，セキュリティ機構の導入がシステム性能を大きく下げることがよく起こる，という点がある．したがって，このようなトレードオフを解消するような技術が必要である．1つの方法としては，適切な性能とセキュリティ要求との関係に対する上位レベルの要求を明らかにすることがあげられる．このようなモバイルエージェント技術の課題のため，実用的な広域開放型分散アプリケーションの開発には，従来型のクライアント・サーバ（C/S）手法が利用されているのが現状である．

そこで本論文では，与えられた要求から，計算コストとセキュリティを考慮したモデルを構築し，パターンを組み合わせることによって，これらのモデルを満たすように，モバイルエージェントを利用した，分散アプリケーションを設計する枠組みを提案する．このような枠組みにより，セキュリティと性能のトレードオフを考慮しながら，アプリケーション開発を行うことができる．また，モデルとパターンは形式的に定義されているので，パターンの組合せがモデルを厳密に満足することを保証できる．さらに，パターンの組合せはあるアルゴリズムによって自動的に導出されるので，セキュリティ要求が変化した際にも容易にエージェントの挙動を修正することができる．

なお，本論文の手法は，自動的にシステムの動作を設計することを目的としているので，扱うことのできるモデルは，単純なものに限られる．そこで，実用的なシステム設計への適用可能性を広げるため，現実的な状況のモデルを扱えるような拡張についても検討する．

また本論文の手法は，モバイルエージェントを利用したアプリケーション設計を行うものである．しかし，目的はあくまでも，ネットワーク上の資源の利用や，セキュリティに関する与えられた要求を満たし，かつ効率の良い分散アプリケーションを開発することである．したがって，本手法を適用した場合，結果的に，モバイルエージェントの移動機能を利用せず，C/S 手法のみを利用した設計となることはありうる．すなわち本手法は，モバイルエージェントの利用も考慮することにより，少なくとも C/S 手法のみ利用した場合よりは優れた設計が可能な点が特長となっている．

本論文の構成は次のとおりである．2 章では，提案する開発手法を説明する．3 章では，例を用いて我々の手法の評価を行う．4 章では，現実的な状況のモデルを扱えるような拡張について検討する．5 章では，関連研究との比較を行う．6 章では，結論と今後の課題を示す．

2. 開発手法

本章では，本論文で提案する開発手法について，例題を用いて説明する．本手法は，以下のプロセスから構成される．

(1) まず 4 種類のモデルを構築する．モデルの種類は，ネットワークモデル，計算・データモデル，セキュリティモデル，および計算コストモデルである．

(2) パターンの適用により，アプリケーションの挙動を設計する．ここでパターンとは，アプリケーションの抽象的な挙動の断片を表す．パターンを適用するというのは，エージェントと非エージェントアプリケーション間のコラボレーションに対してパターンを具体化し，それら具体化したパターンを組み合わせることを意味する．

本章で用いる例題では，ネットワーク上の各ノードに分散しているアプリケーションを連携させるシステムを取り上げる．連携の内容は，データベースアプリケーションからデータを取得し，他のアプリケーションで加工し，その後クライアントノードに表示したり，別のデータベースアプリケーションに格納したりする，といったものである．

以下，本手法の詳細について説明する．まず 2.1 節では，本手法が適用できるための前提条件を述べる．2.2 節では，本手法で最初に構築する 4 種類のモデルを説明する．2.3 節では，本手法で利用するパターン

について説明する．2.4 節では，パターンを適用して求めるアプリケーションの挙動を生成するためのアルゴリズムを示す．2.5 節では，本手法の枠組みが実際に形式的枠組みとして整合的であることを示す．具体的には，本アルゴリズムによって生成される挙動が，与えられたモデルと整合的であるなど，正しい挙動であることの証明を行う．

2.1 前　　提

本手法が適用できるための前提条件は以下のとおりである．

- 1 システムでは 1 つのモバイルエージェントのみを使用する．したがって，エージェントの複製や，エージェント間相互作用は考慮しない．また，モバイルエージェントを用いない，すなわち，クライアント・サーバアーキテクチャのみによる構成も，エージェントがまったく移動しないような設計により実現可能である．
- 計算コストはあらかじめ評価できるものとする．すなわち，各ノード内のアプリケーションとエージェントが，動作中にどのくらい時間を消費するか，およびある量のデータが各ネットワークリンクを流れるのにどのくらいの時間が必要か，といったことがあらかじめ評価できるものとする．
- エージェントの移動先は，あらかじめすべて既知である．したがって，移動先が実行時に動的に決定されるような，自律性を持つエージェントを取り扱うことはできない．
- 後述のモデルで規定される情報（ネットワーク構成やセキュリティ要件など）が，アプリケーション実行中は不変である．
- エージェントは，認証に必要な情報を，実行前にあらかじめすべて取得済みである．したがって，実行中の認証手続きは必ず成功する．
- 対象となるセキュリティ課題として，セキュアでないネットワーク上での盗聴防止と，信頼できないホストから来たエージェントやデータの認証に対応する．

なお 4 章で，これらの前提条件を緩めることができるような拡張についても検討する．

2.2 モ デ ル

次に，4 種類のモデル，すなわち，ネットワークモデル，計算・データモデル，セキュリティモデル，および計算コストモデルについて説明する．

2.2.1 ネットワークモデル

ネットワークモデルとは，アプリケーションを実装するネットワークの物理的構成を示すものである．具体的には，ネットワークを構成するノードと，ノード間のリンクトポロジを表現する．

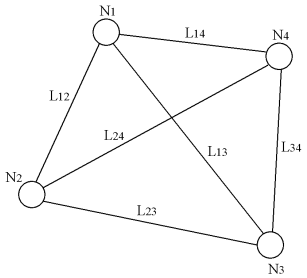

図 1 ネットワークモデル
Fig. 1 Network model.

図 1 はネットワークモデルの例である．この図において，ネットワークノードは $N_i\,(i = 1, 2, 3, \ldots)$ で表され，いくつかのノード対，たとえば N_i と $N_j\,(i \leq j)$，は L_{ij} で表されるネットワークリンクで接続されている．本例ではすべてのノードが相互に接続されているが，接続されていないノード対があっても良い．そのような一般的な場合では，接続されていないノード間の通信のためには，他のノードが仲介する必要がある．

2.2.2 計算・データモデル

計算・データモデルは，エージェントと他のプログラム間の相互作用のプロセスフロー，およびそこで流れるデータ量を表すものである．プロセスフローは，エージェント，プログラム，および相互作用を含むコラボレーション図で表す．形式的には，コラボレーション図は有向グラフである．さらに，本モデルでは，モバイルエージェントが最初に生成されるノードを指定する．通常は，このようなノードはエージェント利用者が管理するノードである．モバイルエージェント移動時には，エージェントのプログラムコードがネットワーク上を流れるので，本モデルで規定するデータ量はエージェントのプログラムコード量を含む．

図 2 は計算・データモデルの例である．顔マークがモバイルエージェントを表し，非エージェントアプリケーションと相互作用を行っている．この図では，エージェントは最初ノード N_2 で生成されることが規定されている．相互作用は，エージェントとノードの間，または 2 つのノード間を結ぶ矢印で表される．矢印は，データが始点から終点に渡されることを表す．本手法では，1 ノードでいくつのアプリケーションが動作しているかは考慮しないので，1 ノードのアプリケーションは，ノードに対応する円でまとめて表される．なお，ノード N_2 の近辺の相互作用矢印において，7 のみがノード N_2 を終点とし，1，2，3，4，6 はエー

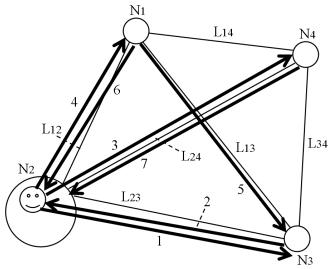

図 2 計算・データモデル
Fig. 2 Computation and data model.

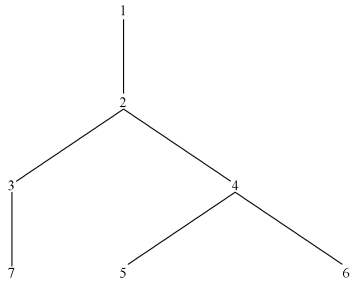

図 3 相互作用の順序制約
Fig. 3 Order constraint.

ジェントマークと結ばれている．

図 3 は，相互作用の順序制約を表す．詳しくは，相互作用は各線分において上にある端点から下にある端点に向かう順番でしか起こらないことを表す．したがってこの例では，3 つの相互作用の列 (1, 2, 3, 7)，(1, 2, 4, 5)，および (1, 2, 4, 6) は，それぞれその中の順番でしか起こらない（たとえば，7 の後に 3 が起こることはない）．

相互作用 i で流れるデータの量は ad_i で表され，モバイルエージェントのプログラムコード量は aa で表される．

本モデル例は，具体的には，例題における次のような連携動作を想定している．

(1) エージェントは，ノード N_3 のデータベースアプリケーションにデータ取得要求を出し（相互作用 1），結果を受け取る（相互作用 2）．
(2) エージェントは，(1) で受け取った結果について，ノード N_4，および N_1 にそれぞれ処理要求を出す（それぞれ相互作用 3，4）．
(3) ノード N_1 は，処理結果をノード N_3 に送って表示させ（相互作用 5），それにともないエージェントに処理結果を送る（相互作用 6）．
(4) ノード N_4 は，処理結果をノード N_2 に送ってデータベースアプリケーションに格納させる（相互作用 7）．

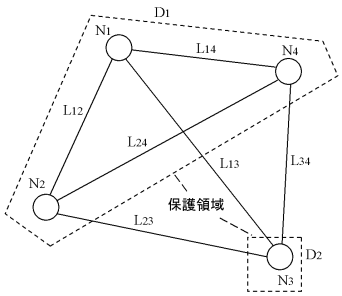

図 4 セキュリティモデル
Fig. 4 Security model.

2.2.3 セキュリティモデル

セキュリティモデルは，保護領域（protection domain）の仕様を記述するものである．なお本論文では，保護領域はノードの集合を指し，ネットワーク全体は保護領域によって完全に分割される．セキュリティモデルは，1 保護領域内の通信はセキュアだが，異なる保護領域間の通信はそうでなく，したがってデータやエージェントの認証と暗号化が必要であることを示す．

図 4 はセキュリティモデルの例である．この例では，ネットワークが 2 つの保護領域 D_1 と D_2 に分割されている．リンク L_{12}，L_{14}，および L_{24} を通じての通信はセキュアだが，L_{13}，L_{23}，および L_{34} を通じての通信はそうでない．

2.2.4 計算コストモデル

計算コストモデルでは，各ノード N_i に対し，以下のようなノード上の計算コストを規定する．

固有計算コスト 固有計算コストは，アプリケーションに固有の計算で消費される時間を表す．非エージェントアプリケーションに対するこの種のコストは，ノード N_i ごとに固定であり，pcc_i で表す．したがって，これらのコストの合計は一定値 $\Sigma_{i=1}^{n}\mathrm{pcc}_i$ となる．エージェントの固有計算コストは，各ノードの計算能力によって変化し，ノード N_i でのコストは，pcca_i で表す．

認証コスト 認証コストは，電子署名と認証の手続きで消費される時間である．署名手続きと認証手続きのコストはノード N_i ごとに固定で，それぞれ sc_i と ac_i で表す．

暗号コスト 暗号コストは，暗号化と復号化の手続きで消費される時間である．これらのコストは，暗号化されるデータの量に比例する．量が a のデータの，ノード N_i での暗号化と復号化のコストは，それぞれ $\mathrm{ec}_i a$ と $\mathrm{dc}_i a$ で表す．

通信コスト 通信コストは，ネットワークリンクをデータが流れるのに消費される時間である．こ

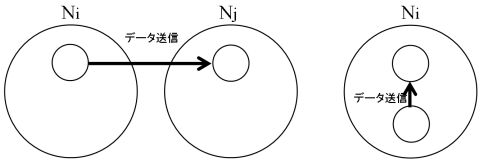

図 5 パターン P_1
Fig. 5 Pattern P_1.

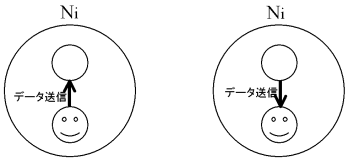

図 6 パターン P_2 および P_3
Fig. 6 Pattern P_2 and P_3.

のコストは流れるデータの量に比例する．量が a のデータがリンク L_{ij} を流れる際のコストは $coc_{ij}a$ で表す．

1 計算プロセスあたりの合計の計算コストは，上記の式で計算される，プロセスの各ステップでの消費時間の和で計算する．

2.3 パターン

本手法においてパターンは，モバイルエージェントアプリケーションの抽象的な挙動の断片を表す．具体的には，モバイルエージェントアプリケーションの，以下のような基本的な挙動から構成される．

データ送信 データ送信は，モバイルエージェントとアプリケーションの間，または 2 つの非エージェントアプリケーション間のデータ転送を表す．データ転送は，ネットワークリンクを通じて，あるいは 1 ノードの内部で起こりうる．

移動 移動は，モバイルエージェントが 2 ノード間を移動することを表す．

セキュリティ手続き セキュリティ手続きは，セキュリティ実現に必要な手続きを表す．本手続きは，暗号化，署名，認証，および復号化から構成される．

各パターンは，拡張されたコラボレーション図の断片で表す．パターンは，データ送信を表す通常の矢印のほかに，移動を表す太線で書かれた矢印をも含むことができる．さらに，パターンはセキュリティ手続きを含むこともできる．

図 5 ～ 図 10 は，セキュリティ手続きを含まないパターンのリストを示し，図 11 はパターンに含まれうるセキュリティ手続きを示す．パターンの詳細は以下のとおりである．

- パターン P_1（**図 5**）は，モバイルエージェントでない 2 つのアプリケーション間で，データが転送されることを意味する．
- パターン P_2 と P_3（**図 6**）は，ノード内部でモバイルエージェントとアプリケーション間でデータが転送されることを意味する．エージェントは P_2 ではデータを送信し，P_3 ではデータを受信する．
- パターン P_4（**図 7**）と P_5（**図 8**）は，データがネットワークリンクを通じて，異なるノード間を転送される点を除き，P_2 および P_3 と同じである．
- パターン P_6（**図 9**）と P_7（**図 10**）は，モバイルエージェントがまず 2 ノード間を移動し，その後移動後のノード内で動作しているアプリケーションとデータを交換することを意味する．エージェントは P_6 ではデータを送信し，P_7 ではデータを受信する．なおエージェントは，これらのパターンにおいてのみ移動する．したがって，これらのパターンを適用しないことにより，クライアント・サーバアーキテクチャによる構成も設計可

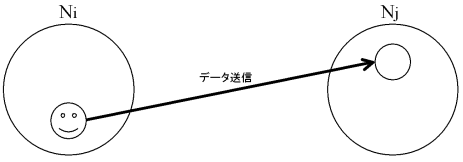

図 7 パターン P_4
Fig. 7 Pattern P_4.

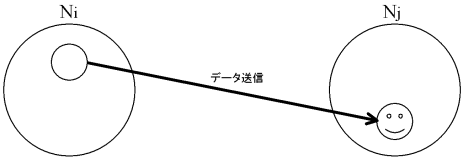

図 8 パターン P_5
Fig. 8 Pattern P_5.

図 9 パターン P_6
Fig. 9 Pattern P_6.

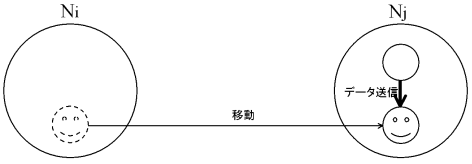

図 10 パターン P_7
Fig. 10 Pattern P_7.

図 11 セキュリティ手続き
Fig. 11 Security procedures.

表 1 パターンの計算コスト
Table 1 Computational costs of the patterns.

パターン	セキュリティ以外のコスト	セキュリティコスト
P_1, P_4 P_5	coc_{ij}·ad	sc_i+ac_j+(ec_i+dc_j)ad
P_2, P_3	0	0
P_6	coc_{ij}(aa+ad)	sc_i+ac_j+(ec_i+dc_j)(aa+ad)
P_7	coc_{ij}·aa	sc_i+ac_j+(ec_i+dc_j)aa

能である．

- パターン P_1，P_4，P_5，P_6，P_7 は，暗号化，署名，認証，および復号化をこの順番で実行するセキュリティ手続きを含むことができる（**図 11**）．セキュリティ手続きを含むこれらのパターンは，P_i^s $(i = 1, 4, 5, 6, 7)$ で表す．

表 1 に各パターンの計算コストを示す．パターンがセキュリティ手続きを含むときは,「セキュリティコスト」の列の値を「セキュリティ以外のコスト」に加える．本表では，エージェントコードのデータ量を aa，転送されるデータの量を ad とする．

パターンは，計算・データモデルに適用され，具体的なエージェント挙動の断片へと具体化される．具体化においては，N_i，N_j を相互作用が起こる実際のノード名に変える．ただしパターンの適用は，パターン適用対象の相互作用および適用の文脈が，パターンの条件に適合する場合のみ可能である．ここで文脈というのは，パターンが後述のパターン適用アルゴリズムの手続きにおいて順番に適用され，したがって各パターン適用はその手続きの中のある文脈で実行される，という意味である．たとえばパターン P_5 の適用により，相互作用が N_i（に割り当てられたノード）から開始され，N_j（に割り当てられたノード）に対して実行される，という挙動の一部が生成される．なお本パターンは，エージェントが最初 N_j で動作しているという文脈のもとでのみ適用可能である．

我々の手法の目的は，これらのパターンを適用することにより，4 つのモデルを満たしかつ全体の計算コストを最小にするシステムの挙動を生成することである．本論文の例では，次のパターン列で表される挙動がそのような解である．

$$P_4^s \rightarrow P_5^s \rightarrow P_4 \rightarrow P_4 \rightarrow P_1 \rightarrow P_5 \rightarrow P_1$$

すなわち，エージェントがずっと N_2 にとどまる，というものである．しかし，この挙動が解であるというのは自明でないので，それを見出す方法が必要となる．

2.4 パターン適用アルゴリズム

解となるアプリケーションの挙動を生成する方法は，次の 2 つのステップから構成される．

(1) 候補となる挙動を，非決定的なパターン適用アルゴリズムによって生成する．後で示すように，生成された挙動は与えられたモデルを満足することが証明できる．

(2) 候補の中で計算コストが最小なものを選ぶ．コストは，挙動におけるパターンのインスタンスのコストと，固有計算コストとを加えたものである．

パターン適用アルゴリズムは，以下の疑似コードで表される非決定的手続きを，計算・データモデルの各相互作用に対して，1 つずつ順に適用するものである．ただし，適用順序は，順序制約を満たすように決定する．初期手順として，モバイルエージェントの現在位置を，計算・データモデルで，エージェントが最初に生成されると規定されたノードに設定する．

```
if（相互作用にモバイルエージェントが
現れない）{
    P1 を適用
} else if（相互作用が 1 ノード 内で
起こる）{
    P2 または P3 を、相互作用の
    方向に応じて適用
} else if（エージェントがデータを送信）{
    次を非決定的に実行
    {
    P4 を適用
    } または
    {
    P6 を適用;
    モバイルエージェント の現在位置
    = データの送信先ノード
    }
} else {
    次を非決定的に実行
    {
    P5 を適用
    } または
    {
    P7 を適用;
    モバイルエージェント の現在位置
    = データの送信元ノード
    }
};
if（相互作用が異なる保護領域を
またがって起こる）{
```

セキュリティ手続きを追加
}

なお本アルゴリズムの計算量は次のように評価できる．まず，1つのフローにパターンを適用する手続きは，固定的な条件判断を行って，適用するパターンと必要なパラメータを選定するのみであり，その計算量は一定である．しかし，非決定的なパターンの選択が起こると，その後最大2つのスレッドに分岐しうる．したがって，全体の計算量は，フローの数を n とすると，最大 2^n のオーダとなる．

2.5 アルゴリズムの正しさの証明

次に，上記のアルゴリズムの正しさを厳密に証明することにより，本手法が実際に形式的枠組みに基づいていることを示す．詳しくは，下記のような定式化を行う．

定理1 前節のアルゴリズムによって正しいシステムの挙動を構築できる．ここでシステムの挙動が正しいというのは，以下の主張のことを指す．

- 相互作用の起こる順序が順序制約を満たす．
- エージェントの移動が連続的である．これは，エージェントの移動が，エージェントが移動前に動作していたノードから始まっている，という意味である．
- 挙動が計算・データモデルのプロセスフローを正しく実装している．特に，エージェントがネットワーク上を移動することによってデータを運ぶ場合，データの流れがプロセスフローと一致していることが必要である．
- 挙動がセキュリティモデルで表されているセキュリティ制約を満たしている．すなわち，エージェント移動を含めたデータの転送が，異なる保護領域間をまたがる場合，データが暗号化され，認証手続きが実行される．

証明：まず順序制約に関しては，アルゴリズムの1ステップの適用順序が順序制約を満たし，かつ適用順序がそのまま相互作用の起こる順序に反映されることにより示される．

次に，エージェント移動の連続性は次のように示される．証明のためには，アルゴリズムの各ステップにおいて，エージェントが動作を開始するノードが，その前に適用されたパターンでエージェントが動作を終えたノードか，またはアルゴリズムの最初のステップにおいてエージェントが最初に生成されたノードである，ということを示せばよい．このことは，アルゴリズムの各ステップにおいて，パターン適用の各場合を調べれば簡単に分かる．

次に，プロセスフローの実装の正しさについては，アルゴリズムの各パターン適用において，パターンのデータフローとモデルのそれとが一致することを直接確かめることによって示すことができる．

最後に，セキュリティ制約については，異なる保護領域間をまたがって実行される相互作用に対しては，必ずセキュリティ手続きが追加されるので，明らかに満足されている． □

3. 手法の評価

本章では，本論文の手法を例題を通じて評価する．詳しくは，以下の点について評価を行う．

- 計算コストとセキュリティのトレードオフをどの程度解消できるか？
- セキュリティ要求の変更に対し，どの程度柔軟に対応できるか？

評価においては，計算コストのパラメータを以下のように設定する．なお，評価においては，実際の数値は必要でなく，コストの比率と大小が計算できれば十分であるので，以下に示すものは相対的な数値である．

- $\mathrm{pcc}_i = \mathrm{pcca}_i = \mathrm{sc}_i = \mathrm{ac}_i = \mathrm{ec}_i = \mathrm{dc}_i = 1$ $(i = 1, \ldots, 7)$
  なお，この設定によって，どの部分での計算に含まれる固有計算コストも同じとなるので，以下の比較においては固有計算コストを省略できることに注意する．
- $\mathrm{coc}_{ij} = 1$ $(i, j = 1, \ldots, 7)$
- $\mathrm{aa} = \mathrm{ad}_1 = \mathrm{ad}_3 = \mathrm{ad}_4 = \mathrm{ad}_5 = \mathrm{ad}_7 = 2$
- $\mathrm{ad}_2 = 3$
- $\mathrm{ad}_6 = 1$

3.1 トレードオフの解消

まず，アルゴリズムの適用で生成されたパターン列の候補のいくつかを示す．なお，これらの候補は，いずれも相互作用の番号順にパターンを適用して得られたものである．

- 列1：$\mathrm{P}_6 \to \mathrm{P}_3 \to \mathrm{P}_4 \to \mathrm{P}_4 \to \mathrm{P}_1 \to \mathrm{P}_5 \to \mathrm{P}_1$
- 列2：$\mathrm{P}_4 \to \mathrm{P}_5 \to \mathrm{P}_4 \to \mathrm{P}_4 \to \mathrm{P}_1 \to \mathrm{P}_5 \to \mathrm{P}_1$
- 列3：$\mathrm{P}_6 \to \mathrm{P}_3 \to \mathrm{P}_6 \to \mathrm{P}_4 \to \mathrm{P}_1 \to \mathrm{P}_5 \to \mathrm{P}_1$

本論文の手法がどの程度トレードオフを解消できるかを見るために，以下の3ケースについてこれらの列の計算コストを比較する．

- ケース1では，セキュリティをまったく考慮しない．すなわち，あたかもネットワーク全体が1つの保護領域の中にあるため，セキュリティ手続きを追加する必要がないかのように考えてパターン

表 2 トレードオフ解消の評価
Table 2 Evaluation of the trade-off resolution.

	ケース 1	ケース 2	ケース 3
列 1	13	53	53
列 2	14	34	60
列 3	15	41	57

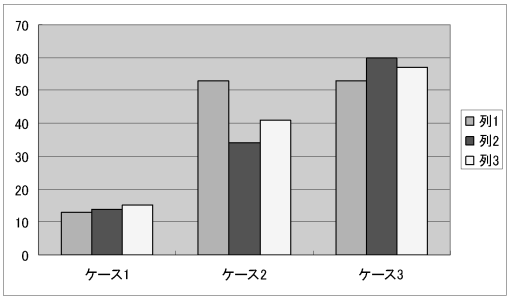

図 12 トレードオフ解消の評価のグラフ
Fig. 12 Graph for the evaluation of the trade-off resolution.

を適用する．

- ケース 2 では，通常どおりセキュリティモデルを考慮する．
- ケース 3 では，どのリンク上のデータ転送でもセキュリティ手続きを実施する．すなわち，あたかもノード 1 つ 1 つが保護領域を構成しているため，すべてのリンクが異なる保護領域間をまたがるかのように考えてパターンを適用する．

結果を**表 2** と**図 12** に示す．

この結果から，次の事実が分かる．

- 列 1 はケース 1 と 3 では低コストだが，ケース 2 では列 2 の方が良い．したがって，セキュリティの考慮によって選択すべき挙動は異なる．
- ケース 3 のように，セキュリティを過度に考慮すると，通常非常に高コストとなる．

以上により，セキュリティを考慮する度合いに応じて，それぞれ適切な挙動を選択できていることが分かる．したがって，本手法により，セキュリティと性能のトレードオフに対し，適切に対応できていると考えられる．

3.2 セキュリティ要求変更への対応

次に，本論文の手法の，セキュリティ要求変更に対する柔軟性を評価する．例として，セキュリティモデルが図 4 に示したもの（以下，セキュリティモデル 1 と呼ぶ）から，**図 13** に示すもの（以下，セキュリティモデル 2 と呼ぶ）に変更したとする．

これらのセキュリティモデルに対する計算コストの比較を，**表 3** と**図 14** に示す．

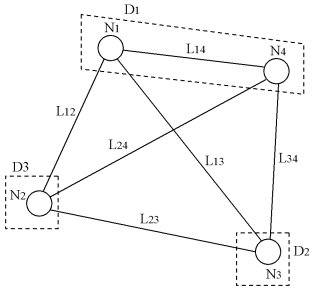

図 13 変更後のセキュリティモデル
Fig. 13 Security model after the change.

表 3 セキュリティ要求変更への柔軟性の評価
Table 3 Evaluation of the flexibility for security policy changes.

	セキュリティモデル 1	セキュリティモデル 2
列 1	53	53
列 2	34	60
列 3	41	41

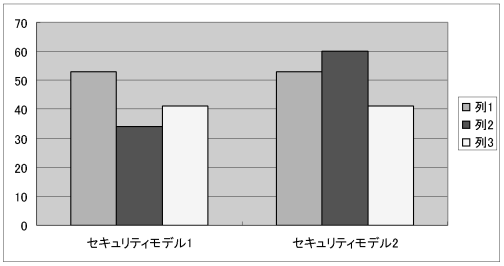

図 14 セキュリティ要求変更への柔軟性の評価の図
Fig. 14 Graph for the evaluation of the flexibility for security policy changes.

表 3 より，セキュリティモデル 2 に対しては列 3 を選択すべきことが分かる．セキュリティ要求変更時に，適切なシステム挙動を見いだすのは，アドホックな方法では困難だが，本論文の手法により，自動的かつ容易に，システムの挙動を適切に修正できることが分かる．ここで「適切に」というのは，変更後のセキュリティ要求を侵害することなく，計算コストを最小にとどめることができる，ということを意味する．

以上のことから，本手法によりセキュリティ要求の変化にも適切に対応できると考えられる．

4. 手法の拡張

本論文の手法は，システム動作を自動生成することを目的としているため，比較的単純なモデルを用いている．そのため，そのままでは現実の開発に適用するのは困難である．本章では，現実の状況に近づけるために必要と思われる，手法を拡張する方向性について

検討する．具体的には，以下のものを取り上げる．

- エージェントを 1 つだけでなく，2 つ以上利用したいという，いわゆるマルチエージェント化は自然な要求である．そのため，エージェントに複製・合流機能を持たせ，さらに複数エージェントの非同期な動作を可能にするような拡張を行う．
- 本手法のセキュリティモデルは，ごく限られたセキュリティ課題にしか対応できないので，考えられる限りのセキュリティ課題に対応できるように拡張する．
- 本手法のネットワークモデルが扱うような，固定的なネットワーク構成やアプリケーション配置という前提は現実的ではない．そのような設定を変更可能にするような拡張が必要である．
- 計算コストのパラメータが，固定値として見積もることができない場合に，統計的に見積もることができれば適用できるような拡張を行うことにより，より現実的な手法となる．
- 移動経路が実行時に変化しうる，自律的なエージェントに対応できるような拡張を行う．
- エージェントが，認証に必要な情報を，実行前にあらかじめすべて取得済みである，という前提は，広域開放型ネットワークにおいては現実的でない．実際には，実行時にそのような情報を取得し，さらに取得した情報に応じてエージェントの動作が異なるような場合がある．また，エージェントが保有している情報が古いものであるなどのため，認証が失敗した場合の特別な処理を通じて，認証に必要な正しい情報を改めて取得することもありうる．そこで，これらのような状況を扱うための拡張を行う．

なお，以下に示すように，これらの拡張は，個々の内容は確立しており，個別には，本論文で提案した基本手法の枠組みに，自然に取り込むことが可能である．しかし，全体を統合することにより，1 つの手法として確立することについては，今後の課題である．

4.1 マルチエージェント対応

マルチエージェント，すなわち複数のエージェントを扱うことについては，以下のような対応を行う．

4.1.1 パターン体系や計算コストの扱いの変更

マルチエージェント化にともない，追加すべき動作プリミティブとして，エージェント複製，合流，およびエージェント間相互作用の 3 つがある．また，複数のエージェントが非同期に動作する可能性を考慮する必要がある．

このような課題に対しては，以下のような対応を行う．

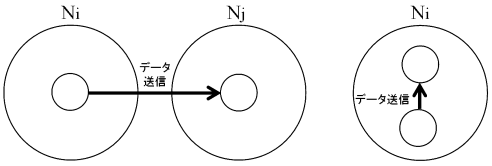

図 15 相互作用パターン例
Fig. 15 Examples of the interaction patterns.

図 16 複製パターンと合流パターン
Fig. 16 Clone pattern and join pattern.

- 各プリミティブを利用したパターンを追加する．ここで，エージェント間相互作用のパターンは，非エージェントアプリケーションも含めた他の種類の相互作用と同様なものが多い．そこで，同様なパターンは 1 つのパターンにまとめ，相互作用を行う対象によってさらに詳細に分類することにする．

  たとえば，**図 15** に示した 2 種類のパターンは，さらに次の 3 種類に分かれる．
  - 非エージェントアプリケーション間相互作用
  - エージェント・非エージェントアプリケーション間相互作用
  - エージェント間相互作用

  ここで，図 15 の 2 つ目のパターンで，矢印の向きがエージェントから非エージェントアプリケーションの場合が，図 6 のパターン P_2 を表すことになる．また，図 15 でエージェントどうしが相互作用を行う場合が，新たに追加されるパターンとなる．

  さらに，エージェントの複製と合流に関するパターンを追加する（**図 16**）．ここで，太線のエージェントは複製により新たに生成されたことを表し，破線のエージェントは合流に際し消滅したことを表す．また，複製と合流においては，各エージェントがコードとデータのうちどの部分を保有するかを考慮する必要がある．そのため適用にあたっては，後述のデータフローモデルも考慮する．
- 複数エージェントの場合は，非同期の動作を行うことがあるため，動作の表現，および合計計算時間の算出方法が変わる．

  まず動作の表現については，直線的な列ではな

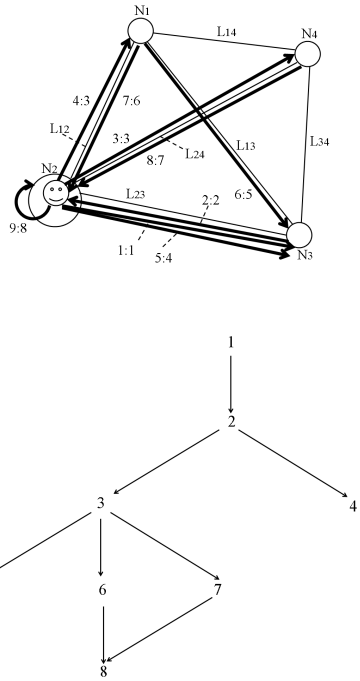

図 17 データフローモデル
Fig. 17 Dataflow model.

く半順序グラフになる．すなわち，非同期に実行する動作単位については，順序関係を規定しない．次に合計計算時間の算出は次のようになる．エージェントの複製が発生したある時点から，合流が発生したある時点までの計算時間は，その間のすべての動作スレッドの消費時間のうち,(その間に発生するたの複製・合流も考慮して）最も長いものになる．したがって，このような計算時間の算出のためのアルゴリズムを適用する必要がある．

4.1.2 計算・データモデルからデータフローモデルへの変更

前述のように，エージェントの複製・合流を扱うために，計算・データモデルをデータフローモデルに変更する．変更の詳細は以下のとおりである．

- プロセスフローをデータフローに変更する．それにともない，各フローでどのようなデータが転送されるかを明示する．そのため，転送されるデータのひとまとまりに番号をつけて区別し，各フローで「フロー番号：データ番号」のように示す．
- 相互作用の順序制約は，データ間の依存関係に変更される．形式上の変更点は，相互作用の番号がデータ番号に変わることである．一方意味的には，順序制約を表していたリンクは，データ間の依存関係を表すことになる．

図 17 に，データフローモデルの例を示す．本例において，フロー 3，4 では，同一のデータ 3 が転送される．またデータ 8 を転送するフロー 9 は，データ 6 と 7 を転送後でないと実行できない．

本例に対して，マルチエージェントに対応したシステム動作生成アルゴリズムを適用すると，まず依存関係のないデータのフローの組（3:3，4:3，5:4 の組など）に対して，複製パターンの適用により，各フローを担当するエージェントを複製することが可能になる．また，フロー 7:6，8:7，9:8 のように，複数のデータが前提となっているフローの組に対しては，合流パターンを適用する場合がある．

4.2 セキュリティモデルの変更

本論文で提案したセキュリティモデルでは，セキュアでないネットワーク上での盗聴防止と，信頼できないホストから来たエージェントやデータの認証のみを取り扱っている．しかし，取り扱うべきセキュリティ課題は他にも数多く存在する．ここでは，セキュリティモデルを変更することにより，さらに多くのセキュリティ課題を取り扱うことを検討する．

4.2.1 セキュリティ課題の詳細

本手法で取り扱うべき，モバイルエージェントに関するセキュリティ課題について，以下にまとめる．

- モバイルエージェントに関するセキュリティ課題は，次の 2 種類に分類できる．1 つ目はエージェントへの攻撃（他のエージェント，ホスト，あるいは第三者などその他の対象による）で，2 つ目はエージェントによるホストへの攻撃である．
- ホスト以外の対象（エージェントを含む）によるエージェントへの攻撃には，盗聴と改ざんがある．これに対しては，拡張前の本手法における，暗号化と認証により対応できる．
- ホストによるエージェントへの攻撃は，上述のその他のエージェントへの攻撃とは次の点で異なる．すなわち，エージェントがプラットホーム上で動作するためには，プラットホームがエージェントのデータやプログラムにアクセスする必要があるので，暗号化や認証によるアクセス制御を用いて攻撃を防ぐことができない．
- エージェントを含む対象によるホストへの攻撃に対しては，従来より認証によるアクセス制御が有効である．しかし，モバイルエージェントについては，エージェント自体が信頼できても，エージェントが危険なホストを通った後にやってきている可能性があるので，認証だけではセキュリティが確保できない．

そこで本節では，以上のようなセキュリティ課題にひととおり対応できるように，セキュリティモデルを変更して，手法を拡張することを試みる．

4.2.2 エージェントからのホストの保護

本論文では，保護領域内のホストから来たエージェントは信頼できるものとして，認証処理を行わなかった．しかし，信頼できるホストから来たエージェントでも，移動中に信頼できないホストを通っている場合には，実際には信頼できない．そのようなエージェントがホストに対して許可すべきでないアクセスを行うことを防止するために，移動履歴を記録して，各移動先ホストが確認する手法[14]などの技術が必要になる．

これらの技術を適用するためには，すべての移動において，移動直後にエージェントが信頼できるかどうかを確認する機構が必要となる．したがって，本手法においてエージェントからのホストの保護というセキュリティ課題を扱うには，必要なエージェント信頼性確認機構の計算コストを，計算コストモデルに採り入れ，計算コスト算出の際に，移動ごとに当該計算コストを加算することになる．

なお，ここで紹介した機構を導入するにあたっては，各ホストがどのような移動履歴を持ったエージェントを受け入れるか，という情報が，あらかじめ知られている必要があることに注意する．

4.2.3 ホストからのエージェントの保護

上で述べたのとは逆に，悪意のあるホストがエージェントに対して許可すべきでないアクセスを行うことに対応したいというセキュリティ課題が存在し，特に従来のクライアント・サーバ技術にはなかった，モバイルエージェント特有の課題として重視されている．

このような課題への対応技術の1つとして，エージェントが保持しているデータを，アクセス制御の単位ごとに分割し，それぞれ署名と暗号化を行うことにより，それらデータへの許可されないアクセスを防止するPRAC（Partial Result Authentication Codes）[18]手法がある．本手法では，各データ部分は，そのデータにアクセス可能なホストのみが解読できるように暗号化されるので，その他のホストが読むことはできない．またホストがデータの追加や変更を行う場合には，署名を義務づけるので，許可されないホストがデータの追加や変更を行うこともできない．

そこで，PRAC技術を，本論文で提案する開発手法に採り入れることを検討すると，以下のようになる．

- 各データへのアクセス権限を表すデータを，セキュリティモデルに追加する．本データでは，各ホスト，またはホストのグループが，各データに対し，どのようなアクセス権限（読み取り，書き込み，あるいは読み書き）が許されるかを明確にする．
- PRAC技術で利用されるセキュリティ機構（暗号化，復号化，署名，および認証）の計算コストを，計算コストモデルに追加する．
- 各動作候補の計算コストの算出においては，エージェントが移動する場合のみ，PRAC技術でのセキュリティコストを加算する．

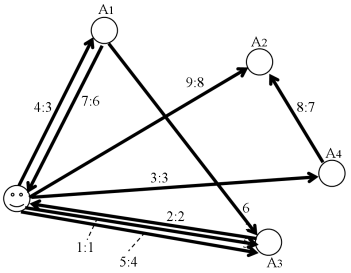

図 18 変更後のデータフローモデル
Fig. 18 Changed dataflow model.

4.3 変更可能なネットワークモデルへの拡張

本論文の手法におけるネットワークモデルでは，ネットワーク構成やアプリケーションの配置などは固定になっている．しかし，現実のシステムでは，そのような設定も一般に柔軟に変更可能である．したがって，変更可能なネットワークモデルを扱えるように手法を拡張することは重要である．

そこで，モデルを以下のように拡張する．

- ネットワークモデルは，表記としてはそのままだが，ホスト間のリンクの意味を，接続が可能である，ということを表すように変更する．
- 計算・データモデル，またはデータフローモデルのフローを，エージェントとホストではなく，エージェントおよびアプリケーションの間のフローであるように変更する．詳しくは，**図 18** に示すように，エージェントとアプリケーションを示す円の間を，フローの矢印で結んだものに変更する．さらに，どのホストにどのアプリケーションをインストール可能かを示すデータを追加する．
- エージェントの初期位置（ホスト），および各アプリケーションのインストール先ホストとして可能な場合を網羅的にリストアップし，それら各設定に対して動作候補生成アルゴリズムを実行する．

4.4 統計的な計算コストパラメータの利用

計算コストモデルにおいて，各パラメータがあらかじめ固定値として評価できる，という前提は現実的でなく，計算コストが統計的，あるいは近似的にしか評

価できないような応用領域への適用も重要と考える．

そのような方向への拡張として，たとえば，計算コストが統計的に与えられている，すなわち，コストの確率分布が与えられている場合は，全体のコストの計算の際に，統計理論に基づいてコストの和や積の分布を計算する．そして，コストを比較して最終的なエージェント動作を決定する際には，計算コストに対するより詳細な要求を検討して，やはり統計的理論に基づいて決定を行うことが必要となる．たとえば，ピーク時のコストをなるべくおさえたい，といった要求に対しては，コスト分布に対しある確率で高コストの部分を打ち切った場合の最大値を比較することとなる．

4.5 移動経路が実行時に変化しうる，自律的なエージェントへの対応

可能性のある移動経路が複数あり，実行時にどの経路が選ばれるかは，各時点での状況により変わりうる，といったような自律性は，モバイルエージェントの大きな特長の 1 つであるため，これを取り扱うことは重要である．

そのために，可能性のある移動経路が，網羅的に列挙可能な場合，次のような手法の拡張を行う．まず，そのような移動経路を列挙し，各経路候補が選択される確率が予測可能であれば，コストの確率分布を比較することにより，またそうでない場合でも，たとえばコスト最大の場合を比較することにより，比較的最適な動作を導出する．

4.6 認証情報の動的取得への対応

エージェントが，認証に必要な情報を，実行時に動的に取得し，その情報に応じて動作が変化しうるような場合への対応のために，まず取得しうる認証情報，およびその情報に応じて起こりうる動作を網羅的に列挙する．その後，4.5 節で示した，移動経路が変化しうるエージェントへの対応のための拡張と同様に，これらの情報の集合に対して，統計的処理やコスト最大の場合での比較などを行うことにより，比較的最適な動作を選択する．

ただし，このような拡張には，取得しうる認証情報，および起こりうる動作が網羅的に列挙可能であることが必要であり，そうでない場合への対応は今後の課題である．

4.7 拡張後の手法の適用範囲

本章で示した手法の拡張を行ったとしても，各項目にて個別に述べたように，なお適用範囲に制限がある．以下に，その制限をまとめて示す．

- 移動経路が実行時に変化しうる，自律的なエージェントに対応する拡張は，可能性のある移動経路が列挙可能な場合にのみ適用可能である．
- 認証情報が動的に取得される場合に対応する拡張は，取得しうる認証情報，および起こりうる動作が網羅的に列挙できる場合のみ適用可能である．
- エージェントからのホストの保護のための機構において，各ホストがどのような移動履歴を持ったエージェントを受け入れるか，という情報が，あらかじめ知られていることが必要である．

5. 関 連 研 究

本章では，本論文の手法と関連研究との比較を行う．

文献 1)，16)，22) をはじめ，エージェントデザインパターンの研究は多数存在する．しかし，これらの研究のほとんどのものはセキュリティ問題を考慮していない．文献 16) はセキュリティパターンを提案しているが，セキュリティと性能とのトレードオフを考慮していない．さらに，これらの研究には形式的枠組みに基づいているものがない．

モバイルエージェントのセキュリティに関する研究も数多くある[11),17),19),23)]．その中には，形式的枠組みに基づいているものもある．しかし，モバイルエージェントのセキュリティはそれ自身困難な課題であるため，性能の課題も考慮したものはほとんどない．一方本論文では，保護領域の概念と，暗号化・認証機構のみを考慮した単純なセキュリティモデルの扱いに絞っている．これにより，セキュリティと性能のトレードオフを考慮することが可能となっている．なお文献 19) では，本論文と同様に，パターンを用いてモバイルエージェントのセキュリティと性能のトレードオフを考慮して設計する手法を提案している．しかし，本文献のパターンは，アルゴリズムによって機械的に適用するのではなく，開発者が手動で適用することを意図している．したがって，パターンの適用は完全には自動化されず，また適用結果がセキュリティ要求を満たしているかどうかを厳密に検証できない．一方本論文の手法では，パターンの適用は，その正しさが厳密に保証されたアルゴリズムによって，完全に自動的に行うことができるので，開発効率が向上している．一方で，文献[19)] の手法は，開発者の裁量でシステムを設計する余地が多分にあり，柔軟性が高いという利点がある．

一般的な文脈でのソフトウェア工学におけるセキュリティについても，多くの研究者が取り組んでおり，形式的枠組みに基づくものも多い[6)]．しかし，このサマリー論文が示しているとおり（228–229 ページ），セキュリティ課題と他の課題とのトレードオフの解消を，ソフトウェア設計フェーズで考慮するような研究は，

特にモバイルコードやエージェントに関してはほとんどない．一方本論文では，そのような方向でのセキュリティ問題への解を提示している．

モバイルエージェントの計算コストについての研究もいくつかある[2),4),10)]．しかし，これらの研究ではセキュリティ課題についてはほとんど考慮していない．一方本論文では，単純な形式的枠組みに基づくモデルを構築し，さらにパターンを利用することにより，セキュリティ課題と計算コストの課題の両方を考慮している．

最後に，形式的枠組みに基づいたソフトウェアパターンの研究もいくつか行われている[7),9),13)]．しかし，これらの研究は，モバイルエージェントもセキュリティ課題も取り扱っていない．一方我々は，モバイルエージェント技術のセキュリティ課題の形式的枠組みによる取扱いにおいて，パターンを利用することが有用と考え，本論文で示したように，形式的枠組みに基づいたパターンを有効に活用している．

6. おわりに

本論文では，与えられた要求から，計算コストとセキュリティを考慮したモデルを構築し，パターンを組み合わせることによって，これらのモデルを満たすようにモバイルエージェントアプリケーションを設計する枠組みを提案した．このような枠組みにより，セキュリティと性能のトレードオフとを考慮しながら，アプリケーション開発を行うことができる．また，モデルとパターンは形式的枠組みに従っているので，パターンの組合せがモデルを厳密に満足することを保証できる．さらに，パターンの組合せはあるアルゴリズムによって自動的に導出されるので，セキュリティ要求が変化した際にも容易にエージェントの挙動を修正することができる．そして，実用的なシステム設計への適用可能性を広げるため，現実的な状況のモデルを扱えるような拡張についても検討した．

今後の課題として，さらに実用的なアプリケーションに適用できるような方向への発展を検討している．詳しくは次のとおりである．

- 2.4 節で示したように，本手法のアルゴリズムの計算量は，最悪の場合フロー数の指数関数のオーダとなる．したがってこのままでは実用性が低いので，計算量の削減を検討する必要がある．たとえば，本手法では最終的に計算コストを比較して動作を 1 つに絞るので，動作候補を生成する途中で，計算コスト最小にならないことが分かれば，その時点で手続き実行を打ち切るような，枝刈り処理などが考えられる．
- 本論文の手法を，実際のモバイルエージェントプラットホーム[20),21)] 上でのアプリケーション構築に適用して，現実的な有効性を検証する必要がある．
- 拡張後の手法も，可能性のある移動経路が列挙可能な場合にしか適用できない．移動経路が無限にあったり，移動の範囲があらかじめ特定できないような場合への対応が課題である．
- 4.6 節で示したように，取得しうる認証情報，および起こりうる動作が網羅的に列挙可能でない場合への対応は今後の課題である．
- エージェントからのホストの保護に対応する機構の導入にあたっては，4.2.2 項に示したような制限がある．しかし，現実の広域開放型ネットワークに適用する場合は，各ホストのエージェント受け入れ方針があらかじめ分かっているとは限らない．そのため，たとえばエージェントが信頼できないホストやネットワークを通るか否かの判断を行い，必要に応じて危険な部分を迂回する，といった場合のコストをも考慮する必要もありうる．そこで，そのような状況への対応も今後の課題である．
- 4 章で示した手法の拡張部分を，拡張前の手法に統合して，1 つの手法として確立する必要がある．
- 広域開放型ネットワーク環境においては，本手法が前提としている，実行中の環境の不変性は現実的でない．したがって，実行中の環境変化を扱うための拡張として，実行中に，現状の環境がモデルのとおりになっているかどうかを監視し，モデルから外れていることが検知されれば，モデルをその場で修正し，定常的環境向けの元の手法を適用して動作を変更する，といった手法を検討する必要がある．

  ただし，このような手法を実現するには，現状の環境とモデルとの整合性を，どのように監視するか，および動作変更をどのタイミングで行うか，といった課題を解決する必要がある．
- 3 章で使用したパラメータは，例題として単純化されている．しかし，実際の環境では，たとえば暗号化と復号化のコストは一般に大きく異なるなどのため，現実的な設定とはいい難い．したがって，実際の環境での実測値を使用するなどにより，現実的な設定で評価を行うことが必要である．

謝辞 研究の機会を与えてくださいました，(株) 東芝研究開発センターコンピュータ・ネットワークラボラトリー尾高敏則室長，ならびに，(株) 東芝研究開発

センター知識メディアラボラトリー住田一男室長には深く感謝致します.

参 考 文 献

1) Aridor, Y. and Lange, D.B.: Agent design patterns: Elements of agent application design, *Proc. Agents'98* (1998).
2) Baldi, M. and Picco, G.P.: Evaluating the tradeoffs of mobile code design paradigms in network management applications, *ICSE'98*, pp.146–155, IEEE (1998).
3) Bradshaw, J.: *Software Agents*, AAAI Press/ The MIT Press (1997).
4) Carzaniga, A., Picco, G.P. and Vigna, G.: Designing distributed applications with mobile code paradigms, *ICSE'97*, pp.22–33, ACM Press (May 1997).
5) Chess, D.M.: Security issues in mobile code systems, In Vigna [17], pp.1–14.
6) Devanbu, P.T. and Stubblebine, S.: Software engineering for security: A roadmap, *The future of Software Engineering*, Finkelstein, A.(Ed.), pp.225–240, ACM (2000).
7) Eden, A.H., Hirshfeld, Y. and Yehudai, A.: Multicast − observer ≠ typed message, *C++ Report* (October 1998).
8) Gamma, E., Helm, R., Johnson, R., Vlissides, J.（著），本位田真一，吉田和樹（監訳）：オブジェクト指向における再利用のためのデザインパターン，ソフトバンク (1995).
9) Geppert, B. and Rößler, F.: Generic engineering of communication protocols — current experince and future issues, *Proc. ICFEM'97*, IEEE (1997).
10) Ismail, L. and Hagimont, D.: A performance evaluation of the mobile agent paradigm, *Proc. OOPSLA'99*, pp.306–313, ACM Press (1999).
11) Karjoth, G., Lange, D.B. and Oshima, M.: A security model for aglets, *IEEE Internet Computing*, Vol.1, No.4, pp.68–77 (July/August 1998).
12) Kendall, E.A., Pathak, C.V., Krishna, P.V.M. and Suresh, C.B.: The layered agent pattern language, *Proc. PLoP'97* (1997).
13) Mikkonen, T.: Formalizing design patterns, *Proc. 20th International Conference on Software Engineering*, pp.115–124, IEEE Computer Society (1998).
14) Ordille, J.J.: When agents roam, who can you trust?, *Proc. 1st Conf. on Emerging Technologies and Applications in Communications* (1996).
15) Silva, A. and Delgado, J.: The agent pattern: A design pattern for dynamic and distributed applications, *3rd European Conference on Pattern Languages of Programming and Computing* (1998).
16) Tahara, Y., Ohsuga, A. and Honiden, S.: Agent system development method based on agent patterns, *Proc. ICSE'99*, pp.356–367, IEEE (1999).
17) Vigna, G.(Ed.): *Mobile Agents and Security*, LNCS 1419, Springer (1998).
18) Yee, B.S.: A sanctuary for mobile agents, *Foundations for Secure Mobile Code Workshop*, DARPA (1997), http://www.cs.nps.navy.mil/research/languages/statements/bsy.ps
19) Yoshioka, N., Tahara, Y., Ohsuga, A. and Honiden, S.: Security for mobile agents, *Agent-Oriented Software Engineering*, Ciancarini, P. and Wooldridge, M.(Ed.), Vol.1957 of *LNCS*, pp.223–234, Springer (2001).
20)（株）東芝，Bee-gent WWW ページ，http://www2.toshiba.co.jp/beegent/
21)（株）東芝，Plangent WWW ページ，http://www2.toshiba.co.jp/plangent/
22) 田原康之，大須賀昭彦，本位田真一：ビヘイビアパターンに基づくモバイルエージェントシステム開発手法，情報処理学会論文誌，Vol.40, No.12, pp.4319–4332 (1999).
23) 田原康之，大須賀昭彦，本位田真一：IPEditor 開発ツールと Mobile UNITY 言語の適用によるモバイルエージェントセキュリティの実現，情報処理学会論文誌，Vol.43, No.6, pp.1582–1597 (2002).

(平成 14 年 10 月 1 日受付)
(平成 15 年 4 月 3 日採録)

田原 康之（正会員）

1966 年生．1991 年東京大学大学院理学系研究科数学専攻修士課程修了．同年（株）東芝入社．1993 年〜1996 年情報処理振興事業協会に出向．1996 年〜1997 年英国 City 大学客員研究員．1997 年〜1998 年英国 Imperial College 客員研究員．現在（株）東芝研究開発センター知識メディアラボラトリーに所属．エージェント技術，およびソフトウェア工学等の研究に従事．日本ソフトウェア科学会会員．

吉岡 信和（正会員）

1993年富山大学工学部電子情報工学科卒業．1995年北陸先端科学技術大学院大学情報科学研究科博士前期課程修了．1998年同大学院大学情報科学研究科博士後期課程修了．博士（情報科学）．同年（株）東芝入社．エージェント技術の研究開発に従事．2002年より文部科学省国立情報学研究所産官学連携研究員，現在に至る．日本ソフトウェア科学会会員．

大須賀昭彦（正会員）

1958年生．1981年上智大学理工学部数学科卒業．同年（株）東芝入社．1985年～1989年（財）新世代コンピュータ技術開発機構（ICOT）に出向．現在（株）東芝研究開発センター知識メディアラボラトリー主任研究員．工学博士（早大）．2002年より電気通信大学大学院客員助教授ならびに大阪大学大学院非常勤講師兼任．主としてソフトウェアのためのフォーマルメソッド，エージェント技術の研究に従事．1986年度情報処理学会論文賞受賞．電子情報通信学会，日本ソフトウェア科学会，IEEE CS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（早大）．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．