推薦論文

パターンを用いたセキュアなモバイルエージェントシステム設計法

吉岡信和[†] 田原康之[†]
大須賀昭彦[††] 本位田真一[†,†††]

柔軟な分散システムを構築する技術としてモバイルエージェントが注目されてきているが，現在その開発方法論とセキュリティへの考慮が課題となっている．本論文では，セキュリティを考慮しながら，段階的にモバイルエージェントシステムを設計する方法論を提案する．この方法論は，セキュリティを考慮したパターンをコストの観点から適用するのが大きな特徴である．これにより，安全性の観点から適切な分散システムを構築が可能となる．

A Design Method for Secure Mobile Agent System Using Patterns

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

Though the agent technology is attracting more attention, mobile agents suffer from insufficiencies in the area of developmental methodology, and security. In this paper, we propose a methodology that supports the step by step development of mobile agent systems while ensuring consideration of security. The methodology revolves around the use of patterns with cost that take these issues into account. This allow us to develop appropriate distributed system in security point of view.

1. はじめに

近年，ネットワークコンピューティングのソフトウェア技術の1つとしてモバイルエージェントが注目されてきている．しかしながら，モバイルエージェント技術にもいくつかの課題があげられている．その中の代表的なものは，開発方法論の欠如とセキュリティへの考慮の欠如である．開発方法論がないと，実用レベルのシステムを構築するのが困難であり，セキュリティを考慮しないと実用運用の際に問題になる．特に，モバイルエージェントシステムは，サービスを遂行するための計算がシステム全体を移動するので，安全性の確保が非常に困難である．

そこで，我々はセキュリティを考慮しながら，段階的にモバイルエージェントシステムを設計する方法論を提案する．この方法論は，おおまかには以下の3つの特徴を持っている．まず第1に，セキュリティを考慮した具体的な設計方法を述べている．第2に，パターンの組合せにより適切なモデルを容易に構築することが可能である．パターンはもともとはオブジェクト指向設計/実装を容易にするための技術であるが，これはエージェントにも応用することができる．第3に，適切なモデルを選択するためにそれぞれのパターンにはコストを与えている．安全性を高めるためのコストは高く，不必要な暗号化や認証により実用的な性能が出なくなる可能性がある．そこで性能と安全性の両方のコストを考慮することにより，これらのトレードオフを客観的に議論することが可能になっている．

また，近年の急激な企業や行政の組織改革にともない，イントラネット/エクストラネット上のソフトウェアもこの変化にセキュリティ上問題なく安全かつ迅速に対応できることが求められている．本方法は，モバイルエージェントの構造に加え，組織情報やデータおよびハードウェアの管理情報をもとに安全なモデルを構築できるため，それらの変更に対しても柔軟に対応可能である．

† 国立情報学研究所
National Institute of Informatics
†† 株式会社東芝研究開発センター
Corporate Research & Development Center, Toshiba Corporation
††† 東京大学大学院情報理工学系研究科
The Graduate School of Information Science and Technology, The University of Tokyo

本論文の内容は 2000 年 8 月のオブジェクト指向 2000 シンポジウムにて報告され，SE 研究会前主査により情報処理学会論文誌への掲載が推薦された論文である．

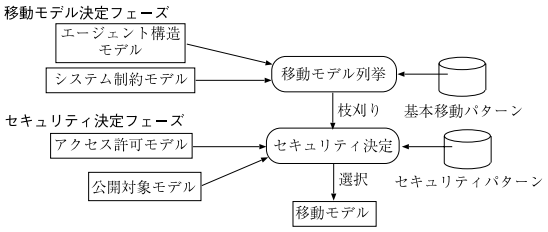

図 1 設計方法の概要
Fig. 1 Summary of process.

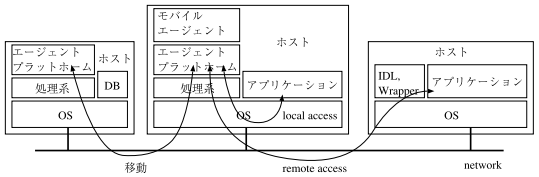

図 2 モバイルエージェントプラットホーム
Fig. 2 Mobile agent platform.

従来からモバイルエージェントシステムにセキュリティを考慮する研究はなされているが，開発プロセスにまで踏み込んだものはまだない．また，その実装方式はいくつか提案されているが，サービス全体を通してどのように実装すべきかの議論はなされていない．そのうえ，モバイルエージェントシステムの安全性についての明確な指針が示されていない．モバイルエージェントのパターンを扱った方法に Tahara らのパターン[1] があるが，セキュリティのためのパターンは提案されていない．加えて，パターンを適用するための客観的な基準を考えていない．また，性能やセキュリティなどの観点でコストを考えてパターンを選択し，その組合せでエージェントの振舞いを設計する方法論は新しい考え方である．

本論文は，また，開発環境/ツールやモバイルエージェントの移動に関する厳密な議論，性能や安全性の観点でシステムを評価するための基盤となる．

この論文の構成は以下のとおりである．2 章では設計法の概要，適用ドメインについて述べる．3 章から 5 章で設計法の詳細についてを述べる．3 章では本方法で用いるモデルについて，4 章ではパターンについて，5 章では設計プロセスについて説明する．続く 6 章では例を示してこの設計法の評価し，本方法について議論する．7 章では関連研究について言及し，最後の 8 章でこの論文をまとめる．

2. セキュリティを考慮した設計

2.1 設計の概要

図 1 が設計方法プロセスの概要である．本方法は，移動モデル決定フェーズとセキュリティ決定フェーズの 2 フェーズからなる．移動モデル決定フェーズでは，エージェントが移動してローカルアクセスを行うか，ネットワーク通信でリモートアクセスを行うかを表現した移動モデルを基本移動パターンの組合せで構築する．セキュリティ決定フェーズでは，このモデルにセキュリティパターンを適用することでセキュリティを付加する．移動モデルは複数導出できるが，移動モデル決定フェーズでは性能コストを考慮して絞り込みを行い，セキュリティ決定フェーズではそれに加え，危険度やオーバヘッドとの兼ね合いを考えて適切なモデルを選択する．

この方法論では，各フェーズで選択した移動モデルに対して，実装，テスト，運用が可能なので，段階的なシステムの洗練が可能である．また，この方法を用いる開発プロセスは，エージェントやシステムの各構成要素について単体テストがほぼ完了しており，結合/総合テストを行う段階にある下流工程やすでにあるレガシーシステムや分散オブジェクトを連携させる設計フェーズを想定している．

2.2 適用ドメイン

本論文で扱う設計法を適用しやすいドメインは，システム全体を把握しその振舞いを予測しやすいイントラネットや企業間エクトラネットの定型的な処理を行う分散システムである．このドメインには，B2B の一般的な企業間連携システムが含まれている．また，仮定しているモバイルエージェントのアーキテクチャは，**図 2** のように，エージェントはローカルアクセスまたはネットワーク通信によりホスト上のアプリケーションを利用可能であるとする．また，モバイルエージェントは，オブジェクトまたはサブルーチンの集まりで構成されているとする．

また，本論文の安全性は，**モバイルエージェントに対する攻撃**に関するものに限定し，ホストへの攻撃などについては考慮していない．また，ネットワークやホストにアクセス許可されている組織以外の人はそこをアクセス操作できず，許可されている組織の人でもデータの改ざんを行う可能性があることを仮定している．

3. モ デ ル

3.1 エージェント構造モデル

このモデルは，エージェントが持つオブジェクトまたはサブルーチンや，ホスト上のアプリケーションをノード（以下では単に計算と呼ぶ）とし，その間のデータフローをアークとするデータフロー図に，以下の情

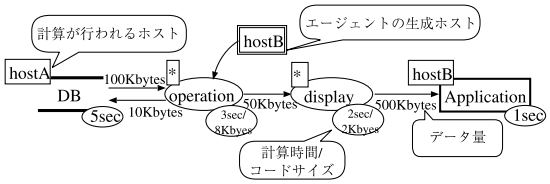

図 3 エージェント構造モデルの例
Fig. 3 Example of agent structure model.

報を付加している．

- **データ量**：アークごとに表記．
- **計算を行うホスト**：ノードの左上に表記．
  ただし計算を行うホストがシステム上のどのホストでもよい場合，“*” とする．
- **サービスが始まるホストと一番初めの計算**：ホスト名を書いた 2 重線からエージェントの一番初めの計算への矢印で表記．
- **計算時間**：各ノードの右下の丸の中に表記．
- **計算のコードサイズ**：計算時間の後に “/” に続いて表記．
  ただし，サービスが始まる前にあらかじめコードをロードするシステムではコードサイズは考慮しない．

計算時間やデータ量は，平均値や典型的な値とする．これらのデータは，通信量やエージェントの計算時間を予測するために必要であり，適切なモデルを選択するためのコスト計算に使われる．また，計算間のデータはエージェントが持つ中間データであり，ネットワーク上に流れる可能性がある．

図 3 にエージェント構造モデルの例を示す．この例では，エージェントは host B で生成され，DB は host A 上のデータベースであり，operation はどのホストでも実行可能であることを表している．以下では，このエージェントを例に本設計法を説明する．

DB の右下の時間は，SQL などの入力を与えて結果を取り出すまでの時間を表す．たとえば，図 3 の場合，DB から 100 Kbytes のデータを取り出すために 10 Kbytes の入力が必要で，入力から出力が得られるまで 5 秒かかることを表している．

3.2 システム制約モデル

システム制約モデルはシステム全体のハードウェア/ソフトウェア情報を表し，ネットワーク構成要素にエージェントの最大滞在時間とネットワークの通信速度を付加している．

前者の制約をつける理由は，リソースを制限して負荷の集中を防ぐためである．たとえば，頻繁に利用されるデータベースがあるホストでは，負荷を集中させないためにエージェントの滞在時間を短く設定し，データベースをアクセスするときだけ移動を許すようにする．

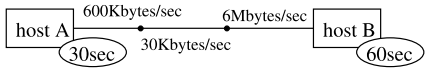

図 4 システム制約モデルの例
Fig. 4 Example of system restriction model.

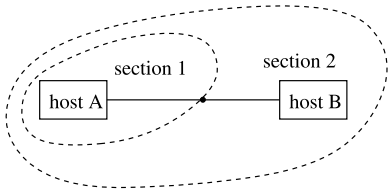

図 5 アクセス許可モデルの例
Fig. 5 Example of access control model.

図 4 にシステム制約モデルの例を示す．ここでは，host A ではエージェントは 30 秒以上滞在できないことを表している．また，host A と host B との間のネットワークは 3 回線からなり，途中に 30 Kbytes/sec の通信速度の回線があることを表している．

3.3 アクセス許可モデル

このモデルはネットワークやホストにどの組織がアクセスが許されているかを表現しており，ネットワーク構成図にアクセス可能な組織名を付加している．このモデルでは，組織ごとにアクセス可能なホストやネットワークを破線で囲み，組織間の包含関係を定義する．

図 5 にアクセス許可モデルの例を示す．ただし，section 1 $\subset$ section 2，すなわち，section 1 のメンバは，section 2 のメンバでもあるとする．この例では，host A とそれにつながるネットワークは組織 section 1 の人にのみアクセスが許可され，host B とそこにつながるネットワークは組織 section 2 の人によってアクセスが許可されていることを表している．ここで，hostA は section 1 以外の人は直接アクセスできないように管理されていると仮定している．

3.4 公開対象モデル

このモデルはエージェントが扱うデータをどの組織に公開してもよいかを表現したモデルであり，エージェントに関するデータフローに公開可能な組織名を付加している．このモデルのアークごとに，アクセス許可モデルを考慮し，そのデータを暗号化すべきかどうか決定する．**図 6** に公開対象モデルの例を示す．この例では，DB の入出力データは section 1 にしか公開しないことを表し，operation の出力は，3.3 節で定義

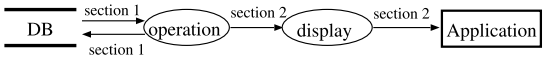

図 6 公開対象モデルの例
Fig. 6 Example of confidentiality model.

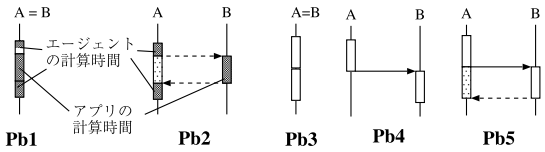

図 7 基本移動パターン
Fig. 7 Basic migration patterns.

した組織の包含関係より，section 2 のメンバでもある section 1 のメンバにも公開してもかまわないことを表している．

4. パターン

4.1 基本移動パターン

セキュリティを考慮しない移動パターンは，**図 7** に示す 5 つの基本移動パターンの組合せで構成する．Pb1, Pb2 は計算パターンであり，エージェント中の計算が行われる様子を表している．Pb3，Pb4，Pb5 は連結パターンであり，計算間のモバイルエージェントの移動の様子を表している．これらの計算パターンと連結パターンの組合せで全体の移動モデルを構築する☆．

図 7 中の縦線はホストを表し，下方向の時間軸となっている．また，縦線上の長方形は計算を表し，黒塗りは実際に計算が行われていることを表し，灰色は他のホストでおこなっているクローンの計算結果待ちのブロック状態を表す．Pb3，Pb4，Pb5 中の白い長方形は注目するパターンそのものではなく，組み合わせる前後のパターンの一部を表している．また，破線矢印はメッセージ通信を表し，実線矢印はモバイルエージェントの移動を表す．以下が各パターンの説明である．

- **Pb1**：計算を行うホストと，その計算に関連するアプリケーションや DB のあるホストが同じ場合や，関連するアプリケーションや DB がない場合に適用するパターンである．
- **Pb2**：計算に関連するアプリケーションや DB のあるホストとのデータのやりとりをメッセージ通信によって行う場合のパターンである．
- **Pb3**：計算を行ったホストと，次の計算が行われるホストが同じ場合のパターンである．
- **Pb4**：計算を行ったホストと，次の計算が行われるホストが違う場合に，モバイルエージェントが次の計算が行われるホストへ移動するパターンである．
- **Pb5**：計算を行ったホストと，次の計算が行われるホストが違う場合にモバイルエージェントの一部またはクローンが次の計算が行われるホストへ移動し，その計算が終わるまでもとのモバイルエージェントはブロックするパターンである．最低限以降の計算に必要になるデータとコードが移動し，計算の結果だけがもとのホストに返される．

☆ ここで，エージェントの一意性を保証するために，エージェントが複数に分かれてそれぞれが同時に違う計算を行わないとする．

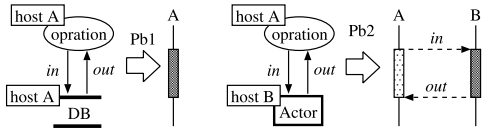

図 8 計算パターンの適用
Fig. 8 Application of patterns.

計算パターンの **Pb1** か **Pb2** のどちらを適用するかは，計算を行うホストをどこに割り当てるかで決まる（**図 8**）．

Pb4 と Pb5 のどちらを適用するかは，計算を行うホストを割り当てただけでは決定できず，性能や安全性を考えながら選択する必要がある．

4.2 セキュリティパターン

セキュリティパターンは，各基本移動パターンに対して，データの漏洩防止のための暗号化や改ざん防止のための署名を考慮したパターンである．

セキュリティを考慮しなければならない場面は，通信に使うネットワークに部外者がアクセスできるかどうか，通信先のホストに部外者がログインできるかどうか，また，適用した基本移動パターンによって 12 通りに分けられる．ここで，部外者とは，エージェントが扱うデータについて，その閲覧が許可されていない者を指す．**表 1** に，セキュリティパターンの適用状況を示した．部外者かどうかの判定は，扱うデータの公開対象とハードウェアのアクセス許可に依存して決まり，詳細は 4.3 節で述べる．また，具体的にどのようにパターンを選択するかについては，5.2 節で述べる．

図 9 がセキュリティパターンの一部である．セキュリティパターンの $P_{se}1$，$P_{se}2$，$P_{se}3$，$P_{se}4$ は Pb2 に，$P_{se}5$，$P_{se}6$，$P_{se}7$，$P_{se}8$ は Pb4 に，$P_{se}9$，$P_{se}10$，$P_{se}11$，$P_{se}12$ は Pb5 に，セキュリティを考慮したパターンである．図 9 中の黒い四角は，セキュリティのためのオーバヘッドを表している．この図の中で，丸印は部内者しかアクセスできない部分を表し，N.G. は，部外者もアクセスできる部分を表す．

一般に，システムの安全性を確保するために**表 2** の

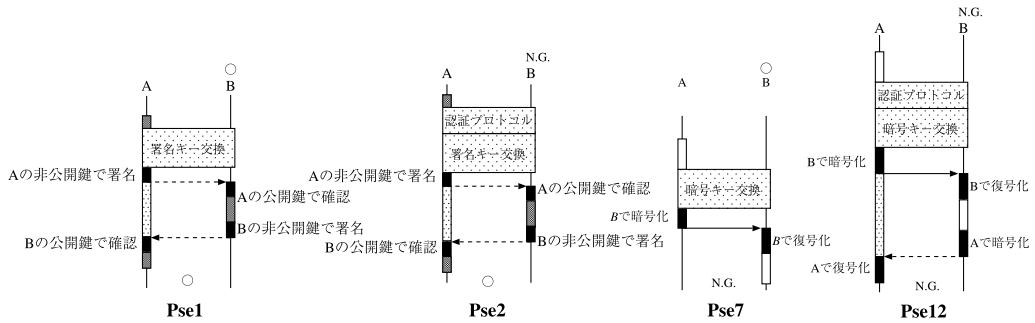

図 9 セキュリティパターン例
Fig. 9 Examples of security patterns.

表 1 セキュリティパターンの選択
Table 1 Security pattern selection.

基本移動パターン	ネットワーク	ホスト	セキュリティパターン
P_b2	部内	部内	$P_{se}1$
P_b2	部内	部外	$P_{se}2$
P_b2	部外	部内	$P_{se}3$
P_b2	部外	部外	$P_{se}4$
P_b3	部内	部内	$P_{se}5$
P_b3	部内	部外	$P_{se}6$
P_b3	部外	部内	$P_{se}7$
P_b3	部外	部外	$P_{se}8$
P_b4	部内	部内	$P_{se}9$
P_b4	部内	部外	$P_{se}10$
P_b4	部外	部内	$P_{se}11$
P_b4	部外	部外	$P_{se}12$

表 2 アタックへの対抗技術
Table 2 Countermeasures against attacks.

危険な場所	アタックの種類	対抗技術
ネットワーク	盗聴	暗号化
ネットワーク	改ざん	署名
ホスト	なりすまし	認証
ホスト	盗聴/改ざん	sandbox モデル

対抗技術が用いられる．

本論文では，この表中の最後の項目である，ホスト中の複数のプロセス（エージェント）をまたがった盗聴や改ざんなどは扱っていない．すなわち，このようなアタックに対する対抗はそれぞれのプラットフォームが行っていると仮定している．しかしながら，これらのアタックに対しては，セキュリティ決定フェーズで取り扱う評価式の危険性コストとして考慮することが可能である．

また，無意味なリクエストや多数のメッセージ送信によるシステム停止攻撃（DoS 攻撃）などは想定していない．これらは，別途，ホスト側のセキュリティとして考慮する必要がある．さらに，ネットワークの切断やパケットそのものが意図しないホストに送られるというアタックを想定していない．それらはネットワークに対するセキュリティとして考慮され，これらをアプリケーションレベルで考慮しなくてもよいことを想定している．

表 3 規定されるアタックの種類
Table 3 Kinds of attacks assumed.

場合	アタッカ	アタックされる場所	アタックの種類	対抗技術
(a)	部内者	ネットワーク	改ざん	署名
(b)	部外者	ネットワーク	盗聴/改ざん	暗号化
(c)	部内者	ホスト	改ざん	署名
(d)	部外者	ホスト	なりすまし	認証

上記のセキュリティパターンは，エージェントが扱うデータから，その部外者または部内者が行うアタックに対して防御するための振舞いである．本論文では，部内者にデータを盗聴されてもかわないというセキュリティポリシーを扱う．また，セキュリティを考慮する際には，(1) 部内者であってもネットワーク上に流れるデータを改ざんする可能性がある，(2) 通信は必ず指定したホストに届くというという仮定を置く．この場合，**表 3** のようなアタックが考えられる．

ここで，(a) の状況でも盗聴は可能であるが，前述のポリシーから，部内者のデータの盗聴はアタックとして考慮していない．また，(c) の場合は，部内者による改ざんを想定しており，部内者に正しい署名をできなくすることでこれを防御可能である．(d) の部外者のなりすましの場合，部外者へのデータの閲覧を防ぐためにデータそのものを部外者に渡さないように認証が必要となる．また，データを暗号化している場合でも，部外者に簡単に情報を渡さないためにも認証が必要である．

これらの場合にあてはまる箇所では，それぞれの対抗技術を付加したセキュリティーパターンを選択する．この判定は移動モデルが決まれば自動的に行え，適切なセキュリティパターンも自動的に決定できる．

ここで，部外者がアクセスできるネットワークやホ

ストを，それぞれ部外ネットワーク，部外ホストと呼び，それ以外を部内ネットワーク，部内ホストと呼ぶ．

ここであげたパターンは，部外ネットワークを利用する $P_{se}3$，$P_{se}4$，$P_{se}7$，$P_{se}8$，$P_{se}11$，$P_{se}12$ には暗号化を，部外ホストを利用する $P_{se}2$，$P_{se}4$，$P_{se}6$，$P_{se}8$，$P_{se}10$，$P_{se}12$ には認証のプロトコルを付加している．さらに部内ネットワークを利用する $P_{se}1$，$P_{se}2$，$P_{se}5$，$P_{se}6$，$P_{se}9$，$P_{se}10$ の場合にもネットワークに流すデータにはすべて署名を行っている．

以下が図 9 中のパターンの説明である．

$P_{se}1$ このパターンは，R_b2 のホスト B が部内，その間のネットワークが部内の場合に適用するパターンである．この場合，ネットワーク上でのデータの改ざんを防ぐために署名を行う．相手の署名を確認するために，まず署名確認のための鍵の交換を行っている．この鍵は，正しい通信相手のものを入手する必要がある．通常この鍵として公開鍵が用いられ，自分の公開鍵を正しい相手に配布し，その相手の公開鍵を自分が所持している状態にする．この正しい相手との鍵の交換は，あらかじめ物理的に配布しておく，アタックされないタイミングに配布する，セキュアな特別なネットワークを使うなどが考えられ，システムの実装に依存する．セッションごとに署名の鍵を交換する場合，あらかじめ決められた情報をもとに正しい相手を認証するプロセスが必要になる場合がある☆．鍵の交換のために認証が必要かどうかは実装に依存するので，本論文では，パターン中に特に明示しない．

このパターンでは，R_b2 より署名のための鍵の交換，データへの署名，署名の確認を行う部分が余計に時間がかかる．

$P_{se}2$ このパターンは，R_b2 のホスト B だけが部外である場合に適用するパターンである．この場合，ホスト上の部外者のなりすましによる情報の漏洩を防ぐためにホスト B 上の通信相手が正しいかどうか確認する必要がある．なりすましの方法としては，ホスト上で，偽の受信ポートを開くことを想定している．さらに，ネットワーク上の部内者による改ざんを防ぐためにメッセージは署名を行う必要がある．セキュリティの実装上，署名キーの交換に認証が必要な場合，認証プロトコルと鍵交換プロトコルは，1 つのプロトコルとして実行される．

このパターンでは，$P_{se}1$ より認証のための時間が余計にかかる．

$P_{se}7$ このパターンは，R_b4 の部外ネットワークに対して適用するパターンである．この場合，移動する前にコードや中間データを暗号化する必要がある．そのために，前もって鍵を交換する．この鍵は，署名キーと同様正しい相手と交換しておく必要がある．鍵の交換にはあらかじめ共有している鍵を使って秘密鍵（セッションキー）を交換する方法が考えられる．鍵を複数のエージェントで共通のものを使う場合☆☆，悪意を持ったエージェントに鍵が渡らないようにする必要がある．これらの鍵の交換も，実装によっては正しい相手を認証するプロトコルが含まれる場合がある．

$P_{se}12$ このパターンは，R_b5 の移動先のホスト B とネットワークが部外である場合に適用するパターンである．この場合，通信相手の認証と暗号化のための鍵の交換が必要である．このパターンのセキュリティに関する詳細を付録 A に載せた．

セキュリティパターンのうち，通信先のホストが部外であるパターン $P_{se}6$，$P_{se}8$，$P_{se}10$，$P_{se}12$ は，このホスト上で盗聴される可能性がある．そのため，安全性を重視するシステムでは，これらのパターンを用いないように計算の割り当て方の変更や，ホストの管理組織を変えるなどの入力モデルの変更が必要になる．この危険性は，次章で述べるモデルのコストとして形式的に表現が可能である．

4.3 部外者の判定

ホストやネットワークにアクセスする者が部外かどうか判定するには，データの公開度（公開対象モデル）とハードウェアの管理状況（アクセス許可モデル）を利用する．ここで，部外者とは，ネットワーク上を流すデータやホストに送るデータについて，そのデータの閲覧が許可されていない者を指す．

ネットワークやホストに部外者がアクセスするかどうかの判定は，以下のように定義☆☆☆できる．

定義 1： ネットワーク N を流れるデータの公開対象メンバの和集合を a，N をアクセスできるメンバの和集合を b とすると，$b-a \neq \phi$ が成り立つとき，かつ，そのときに限り N に部外者がアク

☆ すでに共有済みの鍵を使って署名キーを暗号化する場合，特に認証は必要ない．

☆☆ この場合，通信相手が限定できないが，通信先が間違っていなければシステム上問題とならない．

☆☆☆ この定義は，アクセス許可モデルが完全に守られており，許可されていない者がホストやネットワークにアクセスができないということを仮定している．

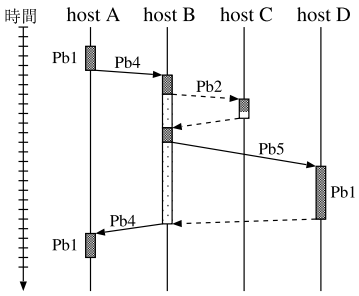

図 10 移動モデルの例
Fig. 10 Example of migration model.

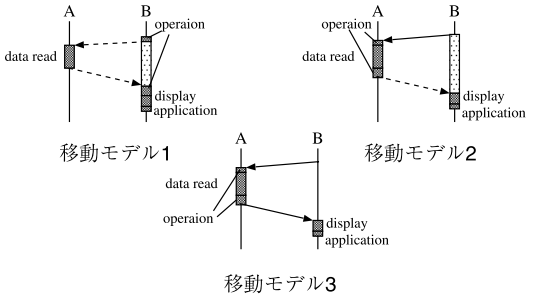

図 11 導出された基本移動モデル
Fig. 11 Derived basic models.

セス可能である．ここで，$b-a$ は，b から a の要素を取り除いた集合である．

定義 2： ホスト H に渡すデータの公開対象メンバの和集合を a，H にアクセスできるメンバの和集合を b とすると，$b-a \neq \phi$ が成り立つとき，かつ，そのときに限り H に部外者がアクセス可能である．

4.4 移動モデル

移動モデルは，移動パターンの組合せで構成するが，パターンとの違いは，計算時間が実時間になっていることとデータの転送時間が考慮されていることである．このとき考慮するモバイルエージェントのコードと中間データのサイズは最低限必要なもののみを考慮し，通信量を最適化することができる．**図 10** は移動モデルの例である．この例では，Pb1，Pb2，Pb1，Pb1 の順で計算パターンを適用し，それぞれを Pb4，Pb5，Pb4 のパターンで連結している．この図には時間軸（タイムスケール）が存在し，各ホスト上の縦軸は，それと対応している☆．

5. 設計プロセス

本論文で提案する設計方法は大きく分けて，移動モデル決定フェーズとセキュリティ決定フェーズで成り立っている．以下では各フェーズで何を決定をすべきかの詳細を述べる．

5.1 移動モデル決定フェーズ

このフェーズでは，まずエージェント構造モデル中の未確定部分へホストを割り当て，次にその割当てに従い基本移動パターンをあてはめて移動モデルを作成する．ここで，基本移動パターンは，まず計算パターンを割り当て，その間の連結パターンを組み合わせて全体を導出する．

移動モデルは複数考えられるが，このフェーズでは，計算時間，通信量を考え，明らかに効率の悪いモデルは枝刈りを行う．また，計算の割当てやパターンを適用する際に，全体のコストを予測しながら行うことで，適切なモデルを素早く求めることが可能になる．

3 章の例に対して導出された基本移動モデルを，**図 11** に示す．

5.2 セキュリティ決定フェーズ

このフェーズでは，移動モデル決定フェーズで導出された移動モデルに対してセキュリティパターンを適用し，そのモデルにセキュリティを付加する．どのセキュリティパターンを適用するかは，表 1 のように基本移動パターンに対し，公開モデルとアクセス許可モデルから自動的に決定できる．

ここで導出された移動モデルは，4.2 節で述べたように，鍵の交換，通信先の認証，署名/確認，暗号化/復号化に余計な時間がかかる．先のフェーズで複数のモデルが導出された場合，それらのモデルについて通信性能と部外ネットワークに流れるデータ量，セキュリティのためのオーバヘッドを考えて，適切な最終モデルの選択を行う．たとえば，以下の式でシステム性能や危険性を評価できる．

$$\text{システム性能} \equiv TT + TC + OH$$
$$\text{危険性}^{☆☆} \equiv DC + DM + DO$$

ここで TT，TC，OH，DC，DM，DO は，それぞれ合計計算時間，合計通信時間，セキュリティのためのオーバヘッド，危険なネットワーク上のデータ通信時間/ コード配送時間，危険なホスト上での計算時間である．ただし，通信時間には，エージェントの移動のための処理時間も含まれるとする．ここで，危険なデータ通信時間とは，公開したくない組織が管理する部外ネットワークに流すデータの通信時間のことで，

☆ ただし，紙面の関係上，これ以降の図にはタイムスケールを省略する．

☆☆ 危険性の式では単純化のためにすべてのアタックにおいて破られる確率が同じであると仮定している．確率が違う場合，重みを付ける必要がある．

表 4 セキュリティのオーバヘッドの例
Table 4 Examples of security overheads.

パターン	ホスト A の計算時間	ホスト B の計算時間
$P_{se}1$	$skey + sign(x) + check(y)$	$skey + check(x) + sign(y)$
$P_{se}2$	$auth + skey + sign(x) + check(y)$	$auth + skey + check(x) + sign(y)$
$P_{se}7$	$dkey + encode(x + code)$	$dkey + decode(x + code)$
$P_{se}12$	$auth + dkey + encode(x + code) + decode(y)$	$auth + dkey + decode(x + code) + encode(y)$

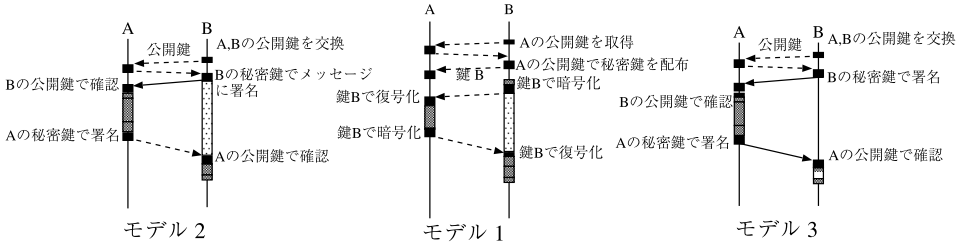

図 12 導出されたセキュリティ移動モデル
Fig. 12 Derived secure models.

危険なコード配送時間は，そのコードが扱うデータがそれを公開したくない組織の管理する部外ネットワークを流す場合のコードの配送時間である☆．最後の計算時間は，ホスト上での他のエージェントからのアタックの可能性を最小限にするために導入する．同じホスト上にある他のエージェントやプロセスからのアタックが行われない（プラットフォームの安全性が保たれている）という仮定がある場合，*DO* を考慮する必要がない．

総合的なコストの評価は，{ システム性能に対するメリット }−{ 危険性 }×{ セキュリティが破られる確率 } ×{ そのときのリスク } という評価式で可能であるが，メリットやリスクの数値化は通常困難である．

セキュリティのオーバヘッドは，パターンごとに以下のように計算できる．すなわち，*skey*，*sign*，*check*，*dkey*，*encode*，*decode*，*auth* はそれぞれ各ホストでの署名キー交換，署名，確認，暗号キー交換，暗号化，復号化，認証にかかる時間とすると，たとえば，$P_{se}1$，$P_{se}2$，$P_{se}7$，$P_{se}12$ のコストは，**表 4** のようになる．ただし，$P_{se}1$，$P_{se}2$ 中の x，y は，アプリケーションや DB への入力または出力サイズであり，$P_{sa}7$，$P_{sa}12$ 中の *code*，x，y は，それぞれ移動すべきコードサイズ，次の計算に必要なデータサイズ，次の計算の出力サイズである．

ここで，認証や鍵交換のための時間は，実際に用いるプロトコルによって異なる．たとえば，エージェントが利用するアプリケーションとお互いに公開鍵をあらかじめ共有している場合，認証のための鍵交換時間は 0 と見なす．また，この場合でも，効率と安全性を考慮し，暗号化キーは秘密鍵としてセッションごとに作成し，公開鍵によって安全にこれを交換するのが望ましい．認証プロトコルとしては，特別な認証局を立てて相手を認証する方法や，あるランダムな文字列を相手の秘密鍵によって暗号化してもらい，それに対応する公開鍵で復元できるかどうかで確かめる方法がある．

認証や暗号化の鍵の有効期限は，システムが保証する安全性によって変わってくる．移動した場合や，移動しない場合でも長くセッションが続く場合，新たにセッション鍵を交換することが望ましい．

図 11 にセキュリティを付加すると，**図 12** のようになる．ここで，暗号化/復号化時間がメッセージへの署名とその確認時間に比べて大幅にかかる場合，モデル 1 よりもモデル 2 やモデル 3 がセキュリティのためのオーバヘッドが少ないモデルとして選択できる．

6. 議　　論

この章では，例を通して本方法の有効性と有用性を検討し，次に本方法の十分性について検討する．

6.1 方法論の有効性

この節では，簡単な例題に対して，方法論なしでセキュリティを付加したモデルを構築する場合と，本方法に基づいて構築する場合比較する．

ここで用いた例題は，「ホスト A に存在するデータベースからデータを取り出して，それを入力としてホ

☆ エージェントが実際に何もデータを持っていなくても，そのエージェントが生成するデータが公開したくない場合，そのコードの配送は危険である．

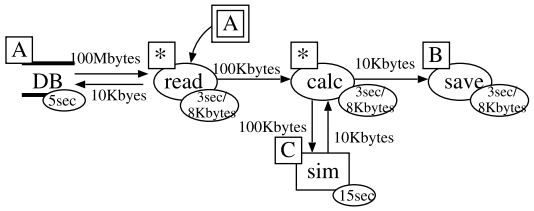

図 13 エージェント構造モデル
Fig. 13 Agent structure model.

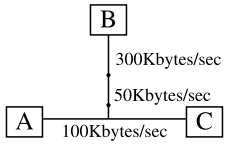

図 14 システム制約モデル
Fig. 14 System restriction model.

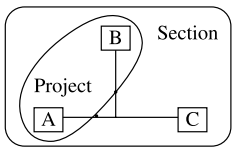

図 15 アクセス許可モデル
Fig. 15 Access control model.

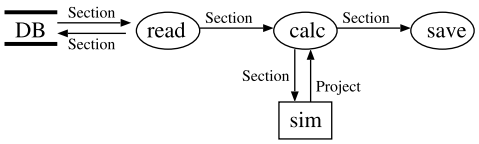

図 16 公開対象モデル
Fig. 16 Confidentiality model.

スト C にあるシミュレータを起動し，その結果をホスト B のファイルに格納する」システムである．このシステムの入力モデルを**図 13**，**図 14**，**図 15**，**図 16** に示す．ただし，Project ⊂ Section とする．

この例は，ある部署（Section）がいくつかの物理的に離れたオフィスで構成されており（図 14），その中でホスト A と B を所有する人で Project を構成する（図 15）ことを想定している．そして，ホスト A で設計情報が蓄えられ，ある製品に関する情報をホスト C で計算し，ホスト B に保存することを考える．この場合，図 13 に示しているとおり，シミュレータのライセンスの関係上，ホスト C でしか動作しないが，シミュレータの結果の生データには金銭的な情報が含まれるので，Project 外の人たちには公開したくないという状況を考える（図 16）．このように，部外のホストに固定されているソフトウェアを利用し機密情報を取り扱う状況は，ライセンスが固定されている場合

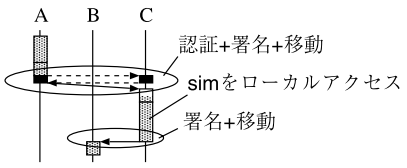

図 17 モデル A
Fig. 17 Model A.

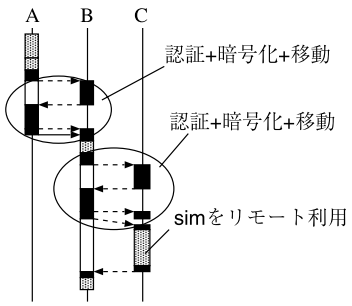

図 18 モデル B
Fig. 18 Model B.

のほかに，システムの一部を他社に委託している場合や，複数の企業をエクストラネットで連携する必要がある場合に起こりうる．この状況でも，適切にセキュリティを付加しなければならない．

方法論なしでモデルを設計しようとすると非常に多くの選択肢が考えられ，しかも，その選択肢のうちどれが正しくセキュリティが守られていてかつ効率が良いものかの判断がつきにくい．実際に，このような簡単な例ですら “read” と “calc” のオペレーションをホスト A，B，C のどこでやるのか，クローンを作るのかどうか，各データのどれを暗号化すればよいのか，“sim” を利用する前に認証が必要かどうかなど，その組合せにより非常に多くのモデルが考えられる．

図 17 中のモデル A が，本方法論で構築した最終モデルである．このモデルでは，DB にアクセスした後，sim をローカルアクセスするためにホスト C に移動している．このとき，sim の計算結果は，Project 内のメンバにのみ公開の情報であり，Project 以外が管理する部外ネットワークを通り，かつ，ホスト C も部外ホストであるので，$P_{se}8$ が適用される．このパターンは，移動先の認証，通信データの署名，移動のための時間を含む．このときの通信データは，計算 calc の入力値と，calc と save のコードの総和であるので，116 Kbytes となる．さらに，計算 save はホスト B で行う必要があるので，sim の計算の後，calc の出力と計算 save のコードを保持し，ホスト B へ移動している．

特にモデルを選択するための方針がない場合，**図 18**

表 5 セキュリティコスト
Table 5 Security costs.

Model	性能/TC+OH(sec)	危険性 (sec)
A	34.52/4.92	1.16
B	65.36/11.82	2.2

のモデル B のような効率の悪いモデルを実装してしまう可能性がある．このモデルでは，DB にアクセスした後，ホスト B に移動し，sim をリモートで利用している．また，セキュリティとして，すべてのアクセスに対して，認証と暗号化を行っている．

モデル A とモデル B のコストを比較してみると**表 5**☆のようになる．ただし，移動やセキュリティにかかる時間を以下のように見積もっている．

移動の前処理/後処理にかかる時間：0.1 秒
認証にかかる時間：1.6 秒
署名とその確認に時間：0.7 秒
暗号鍵の交換にかかる時間：1.0 秒
暗号と復号にかかる時間：1.0 秒

ここでは，認証プロトコルとして公開鍵を用いて，署名はあらかじめ交換しておいた公開鍵で行い，暗号化は移動するごとに生成した秘密鍵によって行った．

モデル A では，ホスト A からホスト C に移動する際の通信が危険な通信となる．このときのデータ量は，前述のとおり 116 Kbytes であり，ホスト A，C 間のネットワーク速度は図 14 から 100 Kbytes であるので，その通信時間は，1.16 秒となる．また，モデル B では，ホスト B からホスト C 上の sim をリモートでアクセスする際の通信が危険な通信となる．この通信は，計算 save のコードとその入力データを含み通信時間は 2.2 秒となる．

表 5 に示すとおり，モデル B では，本来必要のない部分にも認証を導入しているにもかかわらず我々の方法で導入したモデル A よりも，危険性が高くなっている．ここで危険性を時間の単位で評価しているが，その絶対値は重要ではなく，システムの性質を相対的に比較し，最終モデルの候補に順位付けするためにこの値を利用する．

また，性能に占める通信時間とセキュリティのオーバヘッドの総和（TC+OH）は，モデル A で 15%，モデル B で 29%にもなり，決して無視できる割合ではなく，最適な移動モデルとセキュリティの考慮が重要であるといえる．

6.2 実問題への適用可能性

本方法の実問題への適用可能性を調べるために，本方法を使って実問題である環境調和型製品（ECP: Environmentally Conscious Products）設計支援システムをモバイルエージェントで再設計し，これにセキュリティを付加した．

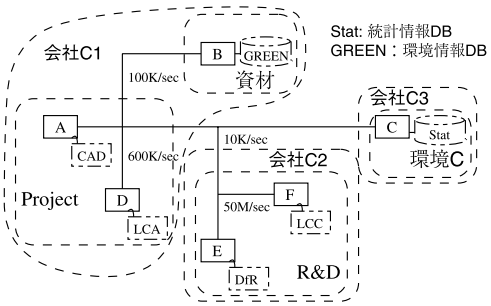

図 19 ECP 設計支援システムの概要
Fig. 19 An overview of ECP design support system.

ECP 設計支援システムとは，製品設計時に，その製品の材料・部品の調達，製品の製造，販売，使用，回収・解体，リサイクル・廃棄に至る製品の全ライフサイクルにわたる環境負荷を考慮するための支援システムである．環境負荷とは，製品が環境に与える影響を数値化したものであり，たとえば CO_2 排出量や，リサイクルのためのコストなどである．

構築したシステムは環境負荷を評価する機能をエージェントによって実現し，その構成要素として，環境影響度などやリサイクル性を算出する ECP 設計支援ツール群，既存の CAD や部品情報などが格納されている各種データベース群が含まれている．**図 19** がその概要である．このシステムでは，C1，C2，C3 の 3 つの会社に存在するシステムを連携することを想定している．具体的には，会社 C1 中の Project で CAD により設計中の製品に関して，資材部の部品情報（GREEN）と会社 C3 が公開している統計的な環境情報（Stat）を使って，有害物質排出量計算シミュレータ（LCA），リサイクル率計算ツール（DfR），リサイクルコスト計算ソフト（LCC）を呼び出し，さまざまな観点で環境負荷を計算する．DB や各種支援ツールは，複数の企業で管理されており☆☆，エージェントは必要に応じてこれらを利用する必要がある．この機能を実現するフローは，**図 20** のエージェント構造モデルとして表現できる．

このような企業間連携システムは，一般に下記のような特徴を持つ．

- さまざまな会社組織で管理されている DB やアプ

☆ ここでの計算式では DO を考慮していない．

☆☆ 会社 C2 は ASP として自社のツールを提供することを想定している．

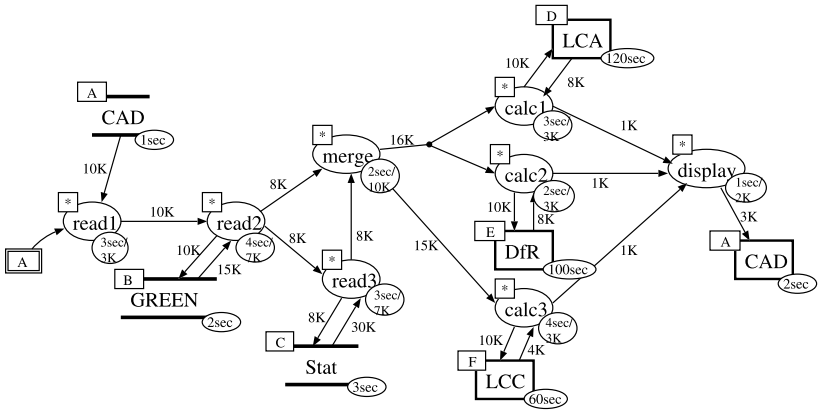

図 20 エージェント構造モデルの例
Fig. 20 An example of agent structure model.

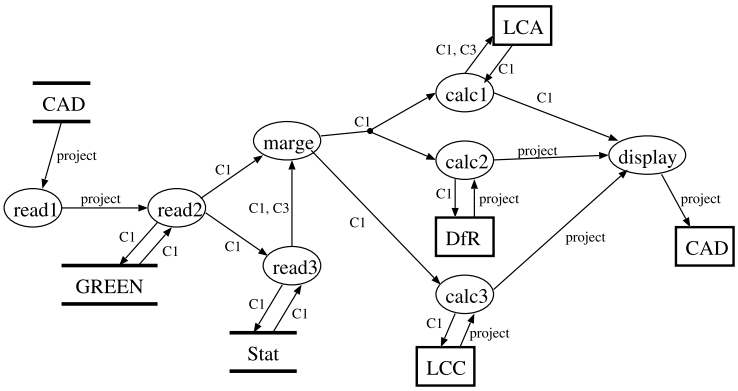

図 21 公開対象モデルの例
Fig. 21 An example of confidentiality model.

リケーションを連携する．

- 高速回線（イントラネット）と低速回線（エクストラネット）が混在している．
- 設計情報など特定の会社に重要な情報を利用して複数の会社のシステムを連携する．

1 項目と 3 項目の特徴により，このようなシステムは，セキュリティが重要な課題であることがいえる．具体的には，ソフトウェアやハードウェアの管理情報やシステムで扱うデータの公開情報をモデリングし，その情報をもとにセキュリティを考慮する必要がある．本方法では，ソフトウェアやハードウェアの管理情報をアクセス許可モデルとしてモデル化できる．また，システムで扱うデータの公開情報は，公開対象モデルとしてモデル化できる．ECP 設計支援システムの場合，公開対象モデルを，**図 21** のように設計できた．ここでは，システムの入出力データ，および金銭に関連する出力結果を project 外秘に設定している．

2 項目の特徴により，効率の良いエージェントの振舞いを設計することが重要になる．遅い回線に大量のデータを流してしまうと，全体のシステム効率に影響を及ぼすことになる．今回実装したシステムでは，実際の製品の一部分の設計情報のみを扱い，データ量は多くならなかったが，シミュレーションを行う場合，大量のデータを扱う可能性があり，また，企業間を結ぶ回線は企業内より千倍以上遅いことから，その通信効率が問題になることが予想できる．本方法では，基本移動パターンを組み合わせることで，最適な振舞いを構築可能である．また，このパターンは，アプリケーションや DB にアクセスするさまざまな部分に適用できる．具体的には，ECP 設計支援システムの場合，基本移動パターンを 10 カ所に適用でき，どのパターンを適用するかにより，エージェントのさまざまな振舞いを設計可能であった．たとえば，**図 22** や**図 23** が振舞いの候補である．図 22 の候補 (a) の場合は，エージェントは，Stat へのアクセスをクローンを作って行い，LCA へリモートでアクセスした後，ホスト E に

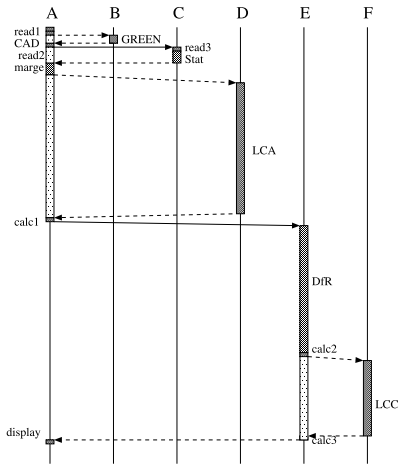

図 22 候補 (a)
Fig. 22 Candidate (a).

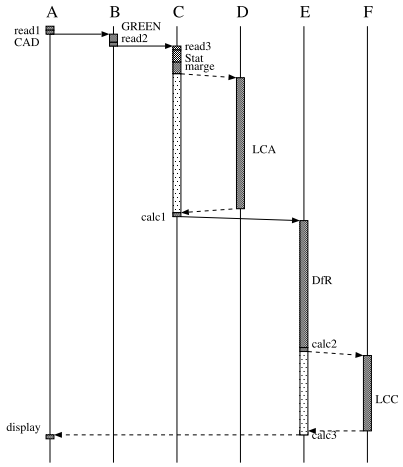

図 23 候補 (b)
Fig. 23 Candidate (b).

移動する．それに対して，図 23 の候補 (b) の場合は，ホスト B，C，E へ移動し，それぞれのデータベースを自らがローカルでアクセスする．

また，一般に，エクストラネットではイントラネットよりも強度の高いセキュリティが要求される．さらに，エクストラネットでは，回線の細く，よりセキュリティのオーバヘッドは高くなる．ECP 設計支援システムの場合，エクストラネットを使った認証は第 3 者の認証局を立てることを想定し，以下のようにセキュリティにかかる時間を設定した．

- 認証にかかる時間：5.0 秒
- 暗号キーの交換にかかる時間：1.5 秒

ただし，イントラネット内の処理時間は，6.1 節に示した時間を想定している．

表 6 性能の比較
Table 6 Comparision of performance.

候補	移動/通信時間	セキュリティ時間	全体性能
(a)	6.22 秒	26.4 秒	109.8 秒
(b)	8.2 秒	31.7 秒	117.1 秒

候補 (a) や候補 (b) にセキュリティを付加する場合，移動やメッセージにどのようなセキュリティを施すべきかは容易に判別できない．たとえば，ホスト C からホスト D にある LCA を利用する場合，扱うデータが機密性が高いものだと暗号化は必須となるが，機密性が低いものは暗号化する必要はなく，適切なセキュリティの確保には，システムの性質と実行環境の両方を熟知していなければならない．それに対し，本方法のセキュリティパターンは，ネットワークが部外かどうかを自動的に判定し，適切なセキュリティを施すことが可能である．さらに，このセキュリティパターンは，移動や通信を行う場面に，汎用的に適用できる．候補 (a) や候補 (b) の場合，それぞれ 6 カ所ある移動と通信にセキュリティパターンが適用可能であった．

最終モデルの全体コストは，個々のパターンのコストの総和である．**表 6** が，候補 (a) と候補 (b) の性能の比較である．候補 (a) の振舞いは，通信時間やセキュリティの時間に関し，候補 (b) よりも効率が良く，最終的にはこの候補をエージェントの振舞いとして選択できた．このように，候補 (a) や候補 (b) など，一見してどちらが適切な振舞いが分からない場合でも，コストを比較することにより，適切な振舞いを導出可能であった．また，これらの候補間の性能の差は全体の約 1 割にもなり，適切な振舞いを発見する重要性を確認できた．

実際のシステムを再設計する場合，すでに構築されている一部分には変更を加えず，他の部分のみ再設計を行うことがある．たとえば，ECP 設計支援システムの場合，DfR や LCC の利用部分はパターンを適用せず，ホスト E に移動してから既存のコードを呼び出す方法をとることが考えられる．このような場合，本方法では，既存のコードを 1 つの計算と見なし，その計算をホスト E で行うようにモデルを作ることにより，パターンを適用しない部分を隠蔽可能である．また，本方法では，パターンを適用した最終モデルのコストを相対的に比較することが可能であり，パターンを適用しない既存部分についての性能が正確に分からない場合も，最適なモデルを選択可能である．具体的には，DfR や LCC の正確な計算時間が分からなくても，相対的に候補 (a) が性能が良いことがいえるので，この候補を最終モデルとして設計することができる．

本方法を用いて実際の設計を行う際，公開対象モデルをいかにして設計するかが問題となった．これは，企業では報告書やメールなど形に残る情報に関して厳密な機密レベルを定めているが，システムの途中結果の一時データなどについては厳密な扱いを定義していないためである．しかしながら，企業間で通信を行う必要がある分散システムでは，第三者に一時データが漏洩してしまう可能性が高く，漏洩した場合のリスクを考え，その機密レベルを定める必要が今後必要になると考えている．公開対象モデルの設計を通して，それがシステムで使う一時データのモデリングの手法として有効な 1 つであることを確認できた．

6.3 モデルとプロセスの十分性

本論文で定義されたモデルには，他のさまざまな付加情報やパターンを考えることが可能である．しかし，重要なことはセキュリティを確保するためのコストの観点でパターンを適用し，モバイルエージェントシステムを構築することである．モデルに付加できる追加情報には，通信データ量やデータのアクセスの制限などシステムのリソースの制限が考えられる．しかし，どこまでシステムの制約を表現すべきかは，システムに依存する．

また，本論文ではエージェントの並列処理を考慮しなかったが，パターンとその適用の仕方を変更することでこれに対応可能である．具体的には，R5 を次の計算が終わる前にその次の計算を始めるように適用する．

実用的なシステムの構築のためには，ホストへの安全性を確保する必要がある．そのための技術は，本論文では言及しなかったが，既存の分散システムの技術がそのまま適用可能である．

6.4 セキュリティポリシーの変更への柔軟性

近年のイントラネットやエクストラネットの分散システムは，組織改革の変更にともなうセキュリティポリシーの変化に柔軟に対応できることが求められている．本方法では，セキュリティポリシーの変更は，アクセスモデルや公開対象モデル，セキュリティパターンの変更に対応し，こられの変更に対しても安全に，かつ，以下のように段階的に対応可能である．すなわち，まず，移動パターンを再利用し，セキュリティ決定フェーズだけをやり直すことで，安全性を確保した早急の対応がとれる．この場合，暗号化の処理や認証部分をエージェントがアプリケーションを利用するロジックと分離しておくことにより，エージェントの中身を変更せずにセキュリティの対応が可能である．具体的には，エージェントの移動やアプリケーションとの通信の API の初めの部分に，認証や暗号化の処理を行うように変更するだけでよい．オブジェクト指向言語でエージェントを実装している場合，これらの API を適宜拡張することで実現可能である．セキュリティ決定フェーズをやり直すだけで安全性は確保できるが，このモデルは性能面や危険性の点で最適ではない可能性がある．次の段階として，性能面も考えてより適切なシステムを構築したい場合，過去の候補モデルを再利用し，新しいモデルを構築する方法が考えられる．この方法では，あらかじめこれまで候補となったモデルをそのコスト計算式とともに保持しておき，この段階で，変化した部分を再計算し，過去の候補から最もコストの低いものを再選択する．再選択されたモデルは，環境の変化が少ない場合には，適切なものになるだろう．劇的に環境が変化し，エージェントの振舞いを大きく変えなければならない場合，最終的な段階としてすべてのフェーズをやり直す必要がある．しかし，この場合も，過去に候補となったモデルは，モデルの候補を計算する場合に再利用可能である．また，最後の 2 つの段階のいずれも，次のような実装を初めの構築時にあらかじめ施しておけば，その再実装の作業が軽減可能である．すなわち，エージェント構造モデル中の各状態をそのままオブジェクト指向言語でステートパターン[2] として実装しておけば，その状態変化を制御する Context クラスを変えるだけで再選択したパターンに対応可能である．より具体的には，状態変化の前後に，エージェントが移動するかどうかをテーブル参照することで決定するように実装しておけば，そのテーブルを書き換えるだけで対応可能である．

6.5 モバイルエージェントとセキュリティ

モバイルエージェントシステムと一般の分散システムでのセキュリティに関する違いは，前者の場合，移動するかメッセージ通信を行うかどうかでセキュリティ確保のためのオーバヘッドが大きく異なるので，どこのセキュリティを強化すべきかが後者よりも難しい点である．しかしながら，必要な部分に最適なセキュリティを施すことにより，モバイルエージェントシステムの方が少ないオーバヘッドで性能の良いシステムが構築できる可能性がある．本論文では，移動パターンとセキュリティパターンの組合せでエージェントを設計し，できあがったエージェントの性能，オーバヘッドをコストとして計算し，最適なシステムを構築する方法を述べている．

7. 関 連 研 究

GOF [2] がオブジェクト指向システム開発に有効なデザインパターンを発表して以来，モバイルエージェ

ントのためのパターンについて多くの研究がなされている．その 1 つに Aridor らによる研究[3] がある．彼らはデザインパターンをトラベリングパターン，タスクパターン，インターラクションパターンの 3 つに分類し，それぞれのパターンを示した．Kendall らは，文献 4) において，エージェントのためのレイアードアーキテキチャにパターンを応用した．彼女らはエージェントのそれぞれのレイアの構築にパターンが適用可能であることを示した．Silva らは，文献 5) において 1 つのエージェントに対したパターンを研究し，それらのパターンをコラボレーション図とクラス図から合成した．

Tolksdorf は，文献 6) において，コーディネーションモデル Linda に基づくエージェントモデル “Mobile Object Space（MOS）” のためのパターンを提案した．このパターンは，Pull，Push，Inde と Traveller の 4 つである．しかしながら，これらの研究ではパターンは個々に提案され，開発者に適切なパターンを選択する方針を与えていない．我々の手法では，パターンにコストを付加することにより，開発者に明確なパターン選択の方針を与えている．

Shaw[7] らによって提案されたソフトウェアアーキテクチャのパターンに質的な属性を考慮し，その属性によってアーキテクチャを評価する方法として，Klein らが Attributed-Based Arichitecture Stles（ABAS）[8] を提案している．この方法では，ソフトウェアの分析や設計を行う際に，性能などからアーキテクチャを検討するための指針を示している．この方法では，マルコフモデルや数学的な解析手法を使ってアーキテクチャを評価し，適切なアーキテクチャの選択する．ABAS と本研究との違いは，(1) パターンとしてモバイルエージェントのビヘイビアを扱っている点，(2) セキュリティを考慮している点，(2) パターンの粒度が異なる点である．以下にそれぞれの違いを述べる．

モバイルエージェントのビヘイビアを考えたとき，その自由度の高さから振舞いの候補数は非常に多くなる．そのため，粒度の大きなアーキテクチャとして，最適な振舞いをあらかじめ用意することはできない．ABAS では，スケジューリングアルゴリズムなどアーキテクチャ中に存在する質に関連する属性をパラメータ化しているが，エージェントのビヘイビアは自由度が高く，この観点でパラメータ化し評価することは困難である．そこで，本研究では，プリミティブなアクセスプロトコルを定義し，その組合せでエージェントの振舞いを最適化することを提案している．

セキュリティを考慮したパターンを作成する場合，その安全性を保証するために，セキュリティをモデル化する必要がある．本研究では，セキュリティのパターン定義とともに，組織構造のモデル化（アクセス許可モデル）とアプリケーションとの関係のモデル化 (公開対象モデル）を行い，部外/部内を区別し，どこを守るべきかのセキュリティポリシーをパターンの適用条件として扱っている．このようにセキュリティの取扱いを厳密に定義したパターンは，従来の研究になかった点である．

3 点目の違いには，パターンの粒度がある．ABAS はアーキテクチャレベルの粒度の大きなパターンを対象にしているのに対して，本論文の方法では，エージェントの振舞いの基本動作に対してパターンを定義し，その組合せで最適なビヘイビアを選択する方法をとっている．本パターンは，エージェントとアプリケーションや DB との通信部分や移動部分に適用できる汎用的なものであり，6.2 節で述べたようにアプリケーション連携プログラムのさまざまな局面に適用可能である．

8. おわりに

本論文では，コストを付加したパターンを用いてモバイルエージェントシステムを開発する手法を示した．この方法では，モバイルエージェントの移動のさせ方を決定するためにパターンを用いる．また，本方法ではセキュリティを考慮したモバイルエージェントシステムを構築可能であり，それぞれについてパターンが存在する．さらに，我々は適切なモデルを選択するための数値をもちいた方針を示した．これによって，客観的なモデルの比較が可能になった．

今後，特定なドメインに有効なより多くのパターンやノウハウの列挙と，この方法論の能力についての考察と拡張，そして，本方法に基づく開発環境を計算機上で実装，有効利用していきたい．

参考文献

1) Tahara, Y., Ohsuga, A. and Honiden, S.: Agent System Development Method Based on Agent Patterns, *21st International Conference on Software Engineering*, pp.356–367, ACM Press (1999).
2) Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: *Design Patterns*, Addison-Wesley (1995).
3) Aridor, Y. and Lange, D.B.: Agent Design Patterns: Elements of Agent Application Design, *Agents'98*, pp.108–115, ACM Press

(1998).
4) Kendall, E.A., Pathak, C.V., Krishna, P.V.M. and Suresh, C.B.: The Layered Agent Pattern Language, *PLoP'97* (1997).
5) Silva, A. and Delgado, J.: The Agent Pattern: A Design Pattern for Dynamic and Distributed Applications, *EuroPLoP'98* (1998).
6) Tolksdorf, R.: Coordination Patterns of Mobile Information Agents, *Cooperative Information Agents II*, pp.246–261, Springer Verlag (1998).
7) Shaw, M. and Garlan, D.: *Software Architecture: Perspectives on an Emerging Discipline*, Prentice Hall (1996).
8) Klein, M.H., Kazman, R., Bass, L., Carriere, J., Barbacci, M. and Lipson, H.: Attribute Based Architecture Styles, *WICSA1*, pp.225–243 (1999).

付　　録

A.1 セキュリティパターンの詳細例

以下は $P_{se}12$ に関するセキュリティの詳細である．他のパターンについても同様に説明が可能である．

パターン名 $P_{se}12$

適用条件

ネットワークが部外，かつ，通信先のホストが部外

想定するアタッカ

ネットワーク上のが部外者および部内者，または，通信先のホストにいる部外者および部内者

想定するアタック

- 部内者によるネットワーク上の改ざん
- 部外者によるネットワーク上の盗聴/改ざん
- 部内者のホスト上のなりすましによる改ざん
- 部外者のホスト上のなりすましによる情報取得/システムの停止

対抗手段

ネットワーク上のデータの暗号化，通信先の認証

実装上の前提

- 暗号化のための鍵が正しい通信相手と交換可能
- データの復号化時に，暗号化した後の改ざんが検出可能

パターンの適用結果

- 認証により正しい通信相手のみと通信が確立し，部外者への情報のホスト上の漏洩を防止
- 符合化により，改ざんを検出可能

考慮外のアタック　通信の切断/遅延

このパターンの限界　ネットワーク上の盗聴による暗号化されたデータの漏洩

(平成 14 年 5 月 10 日受付)
(平成 16 年 1 月 6 日採録)

推　薦　文

モバイルエージェントシステムのセキュリティという新しい問題を対象とした論文である．モバイルエージェント技術が実際的に役に立つためには，避けて通れない重要なテーマだが，技術的には難しい面が多くある．それに対し新規なアプローチを提案し，説得性のある結果を出している．論文としての構成も優れ，読者にとって有用性の高い論文と判断されるため，論文誌に推薦した．

（SE 研究会前主査　玉井哲雄）

吉岡　信和（正会員）

1971 年生．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，ソフトウェア工学の研究に従事．2002 年より文部科学省国立情報学研究所産学官連携研究員，現在に至る．日本ソフトウェア科学会会員．

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．2003 年より文部科学省国立情報学研究所科学研究支援員，現在に至る．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．

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（早稲田大学）．電気通信大学大学院客員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．