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

田原康之[†] 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環境が発展するのにともない，変化が激しく，多種多様な要求が与えられるような分散システムへの需要がますます高まりつつある．そのようなシステム構築のための基盤技術として，エージェントが期待されている．エージェントとは，自律性・移動性・知性・協調性・即応性といった特徴により，環境の変化や多種多様な要求に対応するソフトウェアである．しかし，エージェントシステムは開発方法論が整備されていないために，一般に普及するには至っていない．本論文では，ビヘイビアパターンと呼ばれる，エージェントの典型的な振舞いを示す概念に基づいた，モバイルエージェントシステムの開発手法を提案する．本手法により，開発者はエージェントの複雑な動作を，ビヘイビアパターンを組み合わせたアーキテクチャとして整理することにより，システム設計を効率的に行うことが可能となる．

Mobile Agent System Development Method Based on Behavior Patterns

YASUYUKI TAHARA,[†] AKIHIKO OHSUGA[†] and SHINICHI HONIDEN[††]

As wide-area open networks such as the Internet and intranets grow larger, agent technology is attracting more attention. 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 usual development methods of the agent systems are not sufficiently investigated, the technology is not yet widespread. This paper proposes a method of mobile agent system development based on behavior patterns that represent typical and recurring structures and behaviors of agents. The behaviors patterns are classified according to their appropriate architectural levels and the degree to which they depend on specific agent platforms. Our method enables developers to design agent systems efficiently since they can construct complicated system architectures and behaviors by dividing the design process into three architectural levels and applying the appropriate behaviors patterns.

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環境が発展するのにともない，変化が激しく，多種多様な要求が与えられるような分散システムへの需要がますます高まりつつある．そのようなシステム構築のための基盤技術として，エージェント[2]が期待されている．エージェントとは，自律性・移動性・知性・協調性・即応性といった特徴により，環境の変化や多種多様な要求に対応するソフトウェアである．特に，移動性を備えたいわゆるモバイルエージェントは，最も実用化が近いと期待されている．代表的なものとして，IBMのAglets[11]，および我々が開発したPlangent[14]がある．これにより，モバイルエージェントを適用した実用的なアプリケーション開発，すなわちモバイルエージェントシステム開発が現実のものとなりつつある．

さらに，モバイルエージェントの実システムへの適用が考えられるようになってきたため，開発支援技術の研究も出はじめている．たとえば，統合的なアーキテクチャ[7]，およびオブジェクト指向方法論を適用した開発手法[6]といったものがある．しかし，これらの従来手法では，開発効率，汎用性，拡張性，および理解容易性といった，開発手法に必要と考えられる要素において問題点が多く，実用的なものとは言い難い．また，これらの手法は移動性を持たない常駐エージェントの協調機能のみサポートしているので，実用性が期待されるモバイルエージェントシステム開発への適用は困難である．

一方，オブジェクト指向に代表される従来型のソフ

† 株式会社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
Computer & Network Systems Laboratories, Research and Development Center, Toshiba Corporation
†† 株式会社東芝官公システム事業部
Government & Public Corporation System Division, Toshiba Corporation

トウェア開発において，近年「デザインパターン」[3]，および「ソフトウェアアーキテクチャ」[4] といった概念が注目されている．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．このようなデザインパターンの概念は，ソフトウェア設計の段階にとどまらず，コーディングからプロジェクト管理まで幅広い範囲における「パターン」として拡張されている．またソフトウェアアーキテクチャは，データ構造やアルゴリズムといった範囲を越えて，システム全体の構造に関わるような，ソフトウェアの設計上の課題を表す概念である．現実的な大規模ソフトウェアを，再利用を推進することにより低コストで開発を行う場合には，これらの概念は不可欠なものとなりつつある．なお，「パターン」が目的，動機，および解決法など，項目がある程度明確に規定されたドキュメントを指すのに対し，「アーキテクチャ」は，文脈に応じて異なる意味を持つ．アーキテクチャの例としては，単なるシステムの構成要素間の関係の図から，モジュール間インタフェースの詳細な形式的記述までありうる．本論文では，パターンとしては後述のビヘイビアパターンを取り上げ，アーキテクチャといえば，エージェントシステムを構成する3階層の構造，および各階層の設計仕様を指す．

このような背景のもとで，モバイルエージェントシステム開発へのパターンの適用が検討されはじめている[1),5),9)]．しかし，これらの研究は，いずれも様々なパターンを個別に記述するのみであり，実際のシステム開発プロセスにおいて，各パターンをどのように位置づけるか，という観点に欠けている．そのため，これらのパターンの適用だけでは，体系的な開発手法の構成は困難である．

そこで本論文では，開放型ネットワーク環境での運用を前提としたモバイルエージェントシステムを，パターンを利用して効率的に開発するための手法を提案する．本手法では，まずシステムの構造を，モバイルエージェントプラットホームとの関係を考慮した3階層で構成する．そして，各階層における設計を容易にするために，階層ごとのパターンを用意する．なお本論文で扱うパターンは，エージェントの挙動の記述に関するものがほとんどであるので，ビヘイビアパターンと呼ぶことにする．そのうえで，ビヘイビアパターンを用いて，各階層を上位から下位へと設計する．なお，設計を行う前に，ビヘイビアパターンの適用条件を抽出するフェーズを用意する．したがって本手法は，これと各階層をそれぞれ設計するフェーズとの計4フェーズから構成される．本手法では，各ビヘイビアパターンは，開発プロセスにおいて適用すべきフェーズにより分類される．また上位レベルの設計は，特定のモバイルエージェントプラットホームによらないので，容易に再利用可能である．これにより，モバイルエージェントシステム開発に特有な問題を解決することができる．

本論文は，次のように構成される．まず2章では，モバイルエージェントシステム開発の問題点を整理する．3章では，本論文で提案する開発手法について詳述する．その中で，ビヘイビアパターンについて紹介する．4章では，関連研究との比較について述べる．5章では，結論と今後の課題について述べる．

2. モバイルエージェントシステム開発の問題点

本章では，モバイルエージェントシステム開発の問題点について述べる．まず，モバイルエージェントシステムは従来のシステムに比べてどのような特徴があるのかを説明する．そして，そのような特徴を活かした開発を行ううえで，従来手法はどのような問題点があるのかを明らかにする．

2.1 モバイルエージェントシステムの特徴

一般にエージェントと呼ばれるものは，非常に広い範囲にわたっている．少なくとも，モバイルエージェント，協調型エージェント，およびインタフェースエージェントの3種類のものがあてはまると考えられている．これらのうち，開放型ネットワークにおいて有効とされているのは前2者である．したがって，本論文で考察するモバイルエージェントシステムは，移動機能と協調機能を持つものとする．また，多くの実用的なモバイルエージェントシステムに共通の機能として，複製機能も持つと考える．ここで移動機能は，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環境で提供される情報リソースを，ネットワーク上を移動しながらそれらにアクセスすることにより，効率良く活用することを目的とした機能である．そして協調機能は，複数のエージェントが情報を交換することにより，1つのタスクをエージェント群で協力して解決したり，あるいはそれぞれ別のタスクを持つエージェントが，競合を解消しながら各タスクを実行したりする機能である．また複製機能は，自分と同じプログラムを持つ別のエージェントを生成する機能である．

我々は，このような移動，協調，および複製機能を持

ち，さらにプランニング機能により柔軟な動作が可能なモバイルエージェントプラットホームとして，Plangent[8),14)] を開発した．Plangent においては，ユーザはエージェントへの要求を，ゴールとして与える．するとエージェントは，プランニング機能により，ゴールを満足するプランを作成する．一般に作成したプランは，遠隔ノードに関する不確実な知識も利用して，移動動作も含んだものとなる．また，遠隔ノードにおいて，前記不確実知識が誤っていたり，時間経過とともに無効なものになっていたりしたために，プラン実行に失敗した場合，当該ノードで知識を新たに参照して再プランニングを行うことにより，そのような状況にも柔軟に対応することができる．

2.2 モバイルエージェントシステム開発の問題点

以上に述べたような高度な諸機能により，モバイルエージェント技術は，広域開放型ネットワーク環境における，分散システム開発のための基盤技術としておおいに期待されている．しかしその反面，以下のような理由により，効率的なシステム開発が困難であるのが現状である．

- 移動機能を持つモバイルエージェントは，その動きをとらえるのが，従来のソフトウェアに比べて困難である，という点がある．すなわち，エージェントがリモートホストに移動して作業を開始すると，移動先の環境に対応して動作することになるが，そのようなリモートホストの環境を把握することは容易ではなく，結果的にエージェントの動作の把握も困難となる．このため，モバイルエージェントシステムは理解容易性が低くなりがちであり，開発はアドホックなものとなり，信頼性や拡張性が乏しいシステムとなる．
- モバイルエージェント技術そのものが，登場してからまだあまり時間が経っていないため，現在もプラットホームが急速に進歩しているという状況である．そのため，特定のプラットホームの，しかも特定のバージョンの上に構築したシステムに対し，その後のプラットホームのバージョンアップに対応させたり，異なるプラットホーム上のシステムを連携させたりする，といったことが困難であった．

3. 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

本論文では，モバイルエージェントシステム開発におけるこのような問題へのソリューションとして，ビヘイビアパターンに基づく開発手法を提案する．本手法の概要は次のとおりである．

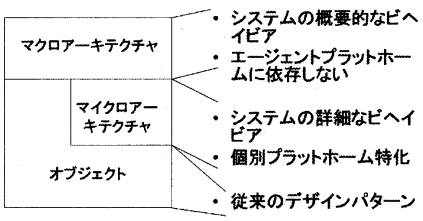

図 1 アーキテクチャ階層構造
Fig. 1 Layered architecture.

- 体系的な手法を確立する目的により，まずシステムの構造を**図 1** に示すような 3 階層で構成する．最上位層は，モバイルエージェントプラットホームに依存しない汎用性を持つ，概略のシステム構成，およびエージェントの動作を表すマクロアーキテクチャである．2 層目は，個別のプラットホームに特化した，詳細なシステム構成とエージェント動作を表すマイクロアーキテクチャである．3 層目は，以上 2 つのレベルの設計仕様に基づいて実装されたプログラムコードを表すオブジェクトレベルである．
- 各階層における設計を容易にするために，以下に示すような，階層ごとのビヘイビアパターンを用意する．
  - マクロアーキテクチャパターン
    マクロアーキテクチャ設計において適用するビヘイビアパターンである．したがって，モバイルエージェントプラットホームに依存しない汎用的なものが選ばれる．具体的には，移動パターンと協調プロトコルパターンの 2 種類に分かれる．
  - マイクロアーキテクチャパターン
    マイクロアーキテクチャ設計において適用するビヘイビアパターンである．したがって，個別のモバイルエージェントプラットホームに特化して，そのプラットホームの特長を活かすようなソリューションを提供するものが選ばれる．
  - OO デザインパターン
    Java や C++などの OO 言語により実装する際に適用するビヘイビアパターンである．GOF[3)] が代表的なものである．
- ビヘイビアパターンを用いて，各階層を上位から下位へと設計する．なお，設計を行う前に，ビヘイビアパターンの適用条件を抽出するフェーズを用意する．したがって本手法は，**図 2** に示すよう

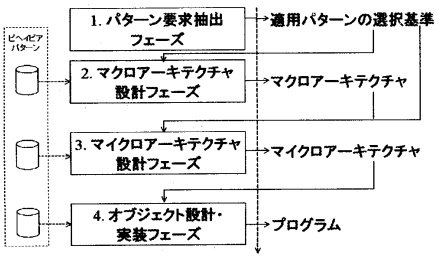

図 2 開発プロセス
Fig. 2 Development process.

に，これと各階層をそれぞれ設計するフェーズとの計 4 フェーズから構成される．

以下に示すように，本手法により，モバイルエージェントの課題が解決できる．

- ビヘイビアパターンという形で，モバイルエージェントの動きをビジュアル化することにより，システムの動作を容易に把握しながら開発を行うことができる．
- マクロアーキテクチャとマイクロアーキテクチャを切り出すことにより，アプリケーションの本質とプラットホームの詳細を分離できる．これにより，次のような効果が生ずる．まず，アプリケーションの本質，すなわち，システムの論理的構成が明確になり，この部分はプラットホームのバージョンアップなどの際にも再利用でき，そのような変更への対応が容易になる．また，プラットホームの詳細においては，そのプラットホームの特性を活かしたアプリケーションの実現が可能となる．
- 特にマクロアーキテクチャで実現できるアプリケーションの本質の記述は，プラットホームによらない汎用性を持つので，異種プラットホーム間の連携のための基盤として期待できる．

3.1 モバイルエージェントシステムアーキテクチャの階層構造

本手法においては，体系的な開発プロセスの検討のために，まずシステムアーキテクチャの階層構造を定義する．各階層の詳細は次のとおりである．

- マクロアーキテクチャ
  マクロアーキテクチャは，エージェントの基本機能である，移動機能と協調機能のみを用いて記述した，概略のシステムの構成，およびエージェントの動作の設計仕様を表す．これらの基本機能を持つモバイルエージェントプラットホームであれば，どのようなものでも利用できるという，汎用性を持つ．本階層は，マクロアーキテクチャパターンの適用により設計する．
- マイクロアーキテクチャ
  個別のモバイルエージェントプラットホームに特化した，システムの詳細な構成，および動作の設計仕様である．ただし，詳細度としては一部プログラムコードで実装された形となる．本階層では，マクロアーキテクチャのうち，当該プラットホームでサポートされる部分を詳細化し，外部アプリケーションとのインタフェースなどの個別に実装すべき部分は，後述のオブジェクトレベルの実装となる．本階層は，モバイルエージェントプラットホームごとに用意された，マイクロアーキテクチャパターンの適用により設計する．
- オブジェクトレベル
  マクロアーキテクチャ，およびマイクロアーキテクチャの仕様に従って実装されたプログラムコードである．本階層は，必要に応じて，GOF などの OO デザインパターンの適用や，OOA/OOD 手法を用いて設計・実装する．

本階層構造において，特にマクロアーキテクチャについては，エージェントプラットホーム独立性が重要であるので，抽象度が非常に高い構造となる．逆に，マイクロアーキテクチャについては，エージェントプラットホームに即した，非常に具体的な構造となるので，この両者は明確に区別できる．したがって，ビヘイビアパターンの明確な分類が可能となる．その代わり，各階層における具体的な設計・実装の支援においては，個々の具体的なビヘイビアパターンが実質的な役割を果たすことになる．したがって，それらが充実していなければ，本論文の方法論はあまり有用なものとならない可能性がある．そこで本論文では，可能な限り網羅的にビヘイビアパターンを抽出した．

マクロアーキテクチャは次のものから構成される．

- エージェントの典型的なネットワーク上での動作シナリオを，エージェント動作図として記述したもの．エージェント動作図は，分散システムの物理的構成（ホスト構成やネットワーク構成）と，その構成においてエージェントがどのように動作するのかを記述する．
- エージェント動作図に対し，エージェントの一般的な動作の全体を状態遷移図として記述したもの．状態遷移図では，エージェントが動作しているホストや，タスクにおけるステータスといった情報を状態ボックスとして表現し，エージェントの移動やローカルな動作をそれらの間の遷移矢印として表す．

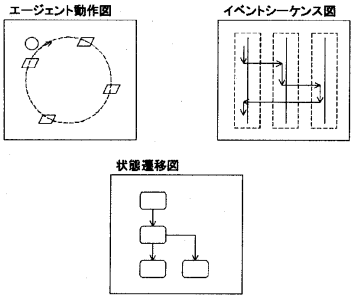

図 3 マクロアーキテクチャ記述
Fig. 3 Macroarchitecture description.

- 状態遷移図に対し，動作全体の時系列的な関係を，イベントシーケンス図として記述したもの．詳しくは，イベントシーケンス図は次の要素から構成される．
  - 各ホストにおいて，アプリケーションやエージェントの時系列的な動作の様子を，縦長の長方形で表記した，ホストライフボックス．
  - 各ホストでローカルに動作するアプリケーションや常駐エージェント，さらには移動の途中で一時的にホスト内で動作しているモバイルエージェントの時系列的な動作の様子を，ホストライフボックス内の縦方向の直線で表記した，ローカルライフライン．
  - アプリケーションやエージェント間の，ホスト内でのローカルな，あるいはホスト間でネットワークを介して行われる相互作用を，ローカルライフライン間の横方向の矢印で表記した，イベント矢印．
  - モバイルエージェントのホスト間移動を，ホストライフボックス間をまたがって，モバイルエージェントのローカルライフラインを結ぶ横方向の矢印で表記した，エージェント移動矢印．

これらのイメージ図を図 3 に示す．

マイクロアーキテクチャの構成は，各モバイルエージェントプラットホームごとに異なる．たとえば Aglets においては，エージェントは純粋な Java コードとして記述され，プラットホームとしてはクラスライブラリのみ提供されるので，そのマイクロアーキテクチャは，通常の OOA/OOD における設計仕様表記（たとえば UML），あるいは直接 Java コードにより記述される．また Plangent の場合は，エージェント動作についてはプラットホーム独自の Plangent スクリプトにより記述される．そしてエージェントと外界の相互作用については，最終的には Java で記述されるため，やはり Aglets と同様の表記となる．Aglets と Plangent それぞれの階層構造を，ビヘイビアパターンの観点からまとめたものを，図 4，5 に示す．

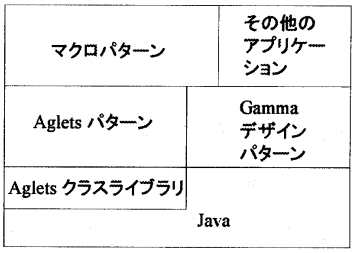

図 4 Aglets アーキテクチャ階層構造
Fig. 4 Layered architecture of Aglets.

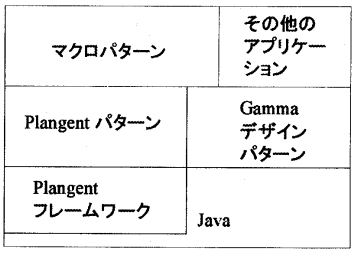

図 5 Plangent アーキテクチャ階層構造
Fig. 5 Layered architecture of Plangent.

3.2 ビヘイビアパターンカタログ

本論文では，開発プロセスにおいて，上述の各フェーズのうちマクロアーキテクチャおよびマイクロアーキテクチャ決定フェーズにおいて，それぞれに対応したビヘイビアパターンを適用することにより，モバイルエージェントシステム開発の効率化を図ることを提案している．ここでビヘイビアパターンとは，モバイルエージェントシステム開発における経験を定式化し，明示的に表現することにより，それらのモバイルエージェントシステム設計の再利用を容易にするものである（図 6）．

- 各階層は，それぞれに対応するビヘイビアパターンを具体化した構成要素を組み合わせて設計する．
- 下位の階層に対するビヘイビアパターンは，上位階層のビヘイビアパターンの構成要素を詳細化する．
- 下位の階層は，上位階層の設計仕様を詳細化することにより設計する．

以下本章では，各ビヘイビアパターンの詳細を一定のフォーマットで記述し，系統的に配列したパターンカタログとして紹介する．ビヘイビアパターンを記述

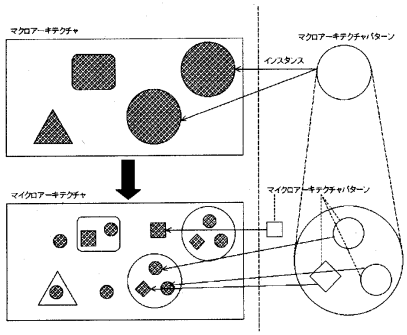

図 6 ビヘイビアパターンとアーキテクチャの関係
Fig. 6 Relationship between patterns and architectures.

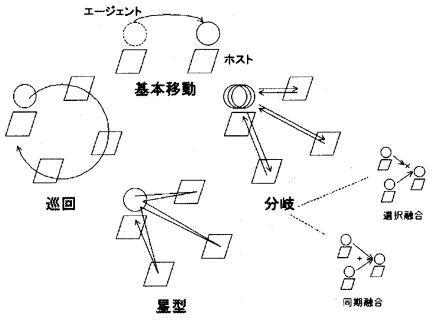

図 7 移動パターン
Fig. 7 Mobility patterns.

するフォーマットとしては，従来の OO デザインパターンと同様に，多数の項目から構成されるパターンテンプレートを用意する．本論文では，紙数の都合により，各ビヘイビアパターンの概要を示すにとどめ，テンプレートに基づくビヘイビアパターン記述については，付録 A.1 において例を示す．

3.2.1 マクロアーキテクチャパターン

マクロアーキテクチャパターンは，モバイルエージェントプラットホームに依存しない汎用的なものとして，移動パターンと協調プロトコルパターンの 2 種類のものを提供する．

移動パターン

モバイルエージェントが，ネットワークを移動しながら作業を行う場合に適用するビヘイビアパターンである．これらのビヘイビアパターンを利用することにより，ネットワークは必要なデータとエージェントの移動のみに使い，計算資源はローカルに利用することにより，効率向上を図ることができる．したがって，ネットワーク上で運搬すべきデータが少量で，ネットワークの帯域幅が狭い場合に有効である．

このような移動パターンは，基本移動パターンと，その他の移動パターンとに分類できる．

基本移動パターンは，あるホストから別のホストへの 1 回の移動のビヘイビアパターンである．本パターンは，移動時のエージェントの実行状態の継続方式により，さらに 3 種類に分かれる．すなわち，継続なし，浅い継続，および深い継続である．本パターンは，移動時のエージェントの実行状態の継続方式により，さらに 3 種類に分かれる．すなわち，継続なし，浅い継続，および深い継続である．これらは，エージェント移動時に，エージェントが保持する基本データ，および動作停止時の実行状態を表すデータのそれぞれを持ち運ぶか否かにより，次のように区別される．

- 継続なし：基本データ，実行状態データともに持ち運ばない．
- 浅い継続：基本データを持ち運び，実行状態データは持ち運ばない．
- 深い継続：基本データ，実行状態データともに持ち運ぶ．

これらのデータを持ち運ぶ場合，移動先では次のように取り扱う．基本データに関しては，移動して実行開始後，実行手続きにおいて通常どおりに取り扱う．また実行状態データについては，移動して実行開始時に，本データが表す実行状態から実行を再開するようにする．

その他の移動パターンでは，3 つ以上のホスト間をエージェントが移動する．このようなものは，さらに巡回，星形，および分岐の 3 種類に分かれる．また，これらの移動パターンを複数組み合わせた，複合移動パターンも考えられる（**図 7**）．その他の移動パターンでは，3 つ以上のホスト間をエージェントが移動する．このようなものは，さらに巡回，星形，および分岐の 3 種類に分かれる．また，これらの移動パターンを複数組み合わせた，複合移動パターンも考えられる（図 7）．

巡回パターンは，エージェントが必要な場所を順に巡り，各ノードで作業を行う．星形移動パターンは，エージェントがそれぞれ必要な場所と基点とを往復し，各移動先ノードで作業を行う．分岐パターンでは，エージェントは，自分の複製を利用したいサイトの数だけ生成する．その後各複製は，各サイトに移動して，各リソースにアクセスし，その後は親エージェントのサイトに帰還，消滅，あるいは常駐する．帰還する場合は，各エージェントが持ち帰った結果の取扱いに対し，同期融合と選択の 2 種類に分かれる．同期融合は，融合の際に獲得した情報を 1 つにまとめ，選択の場合

はどちらか1つの情報だけ残す．

これら種々の移動パターンを使い分けるための条件の概要は次のとおりである．詳細は，各ビヘイビアパターンの記述で説明する．

- 2ホスト間の接続には基本移動パターンのみ，3以上のホスト間の接続にはその他の移動パターン．
- 次に，ホストの安定性で分類する．安定したホストに対しては巡回パターンか星形移動パターン，不安定なホストに対しては，分岐パターンを適用する．
- 巡回か星形かについては，運搬データの量で判断する．少量ならば巡回，大量ならば星形を適用する．

協調プロトコルパターン

複数の常駐エージェントが，メッセージ交換を行うことにより，協調動作を行う場合に適用するビヘイビアパターンである．このタイプのビヘイビアパターンとしては，まずプロトコルにより様々なビヘイビアパターンがある．代表的なものは，契約ネットプロトコル，(非) 同期バックトラック，マルチステージネゴシエーションである．たとえば契約ネットプロトコルは次のようなものである．あるタスクを与えられたエージェントが，それを実行する能力を持たない場合に，他のエージェントに実行を依頼するためのプロトコル．まず複数のエージェントに，タスク実行の入札依頼を行う．受理した入札のなかからコストが最小のものを選んで落札とし，落札エージェントにタスク実行を依頼する．

これらのプロトコルは，アプリケーションへの要求に応じて決定する．たとえば，制約充足問題に対しては (非) 同期バックトラック (同期・非同期は，性能への要求を考慮して判断する)，負荷分散の要求に対しては契約ネットプロトコル，などとなる．また，メッセージ交換の方式や，複数エージェントの配置方式の違いにより，直接交渉，仲介，および派遣の3種類のビヘイビアパターンを考えることができる (**図8**)．直接交渉パターンは，協調プロトコルの実現において，メッセージ交換を，一般にはネットワークを介して行うビヘイビアパターンである．仲介パターンでは，まず常駐型エージェントとは別に，仲介エージェントと呼ぶ移動型エージェントを用意する．仲介エージェントは，サイト間を移動しながら，常駐型エージェントの協調を仲介する．さらに派遣パターンでは，ユーザが利用しているマシンから，リソースが配置されているマシンへエージェントを移動させる．エージェントはこの時点では移動型であるが，移動後は協調型エージェントとして，エージェント間 (したがってマシン間) でメッセージ交換を行い，協調動作を実現する．

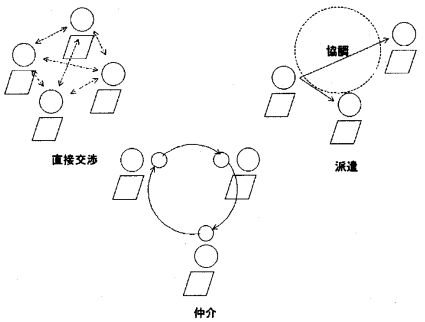

図8 協調プロトコルパターン (協調方式による分類)
Fig. 8 Cooperative protocol patterns (classified according to cooperation methods).

3.2.2 マイクロアーキテクチャパターン

マイクロアーキテクチャ設計において適用するビヘイビアパターンとして，個別のモバイルエージェントプラットホームに特化して，そのプラットホームの特長を活かすようなソリューションを提供するものを用意する．本論文では，プラットホームの例として，Aglets および Plangent を取り上げ，それぞれのマイクロアーキテクチャパターンについて紹介する．さらに，いくつかのプラットホームで共通に利用可能なビヘイビアパターンとして，セキュリティ・セーフティパターンを取り上げる．

Aglets パターン

ここでは Aridor & Lange[1] で紹介されている各ビヘイビアパターンの概要を説明する．Aglets の特徴として，エージェントは純粋な Java コードとして記述され，プラットホームとしてはクラスライブラリのみ提供される，という点があるので，そのビヘイビアパターンも，Java の特長をも活かしたものとなっている．Aglets パターンは，次の3種類に分類される．

- Traveling Patterns
  エージェントの移動に関するビヘイビアパターンである．巡回マクロアーキテクチャパターンの実装に用いることのできる Itinerary と，基本移動マクロアーキテクチャパターンの実装に用いることのできる Forwarding と Ticket がある．
- Task Patterns
  エージェントが一連のタスクを実行するのを実現するためのビヘイビアパターンである．Master-Slave と Plan がある．
- Interaction Patterns
  エージェントどうしの相互作用において適用するビ

ヘイビアパターンである．協調プロトコルパターンの実現などに利用できる．Meeting, Locker, Messenger, Facilitator, Organized Group がある．

Plangent パターン

すでに示したように，Plangent の大きな特徴は，プランニングに基づく知的動作という点である．したがって，Plangent のためのマイクロアーキテクチャパターンとしては，このような特徴を最大限に活用するようなものが適切であるといえる．以下に示すビヘイビアパターンは，そのような観点から抽出されたものである．

- 実行の基本動作
  エージェントの基本動作パターン例として，移動，知識更新，複製，操作，および認識があげられる．ここで，知識はプランニング機能に利用するものである．また複製においては，データまで同じエージェントを生成するクローンと，データとしてはサブゴールを与える子エージェントパターンの 2 種類がある．
- プラン生成・実行パターン
  Plangent のプランニング機能や再プランニング機能といった AI 機能を用いた動作は，知的動作パターンという形でまとめることができる．これらの機能は，実行すべき動作手順をプランとして生成し，その後生成したプランを実行するという，プラン生成・実行パターンとして定式化できる．

セキュリティ・セーフティパターン

エージェント，特にモバイルエージェントにおいては，エージェントやホストにおけるセキュリティやセーフティが重視される．そこで，セキュリティ・セーフティのサポートが十分でないプラットホームについては，これらの機能を実現するビヘイビアパターンが必要となる（**図 9** 参照）．

セキュリティパターンの例としては，移動エージェントが悪意のあるホストでのデータの改ざんを防ぐためのダミーデータ照合パターン[12]がある．このビヘイビアパターンでは，あらかじめエージェントにダミーデータを付加しておき，さらにエージェントのコピーを移動元に残しておいて，移動エージェントが戻ってきたときにダミーデータを照合する．

またセーフティパターンの例としては，次のようなものがある．エージェントが不安定なホストに移動する場合に，あらかじめそのエージェントの複製を移動元に残しておき，そのエージェントが移動先で動作を完了した際に，動作完了を示すメッセージを受けるようにしておく．さらに，移動先のホストに障害が発生し，エージェントが動作続行不能となった場合に対応するため，一定時間を過ぎて複製エージェントがメッセージを受け取らなかった場合に，移動処理を繰り返す，といった動作完了確認パターンなどが考えられる．

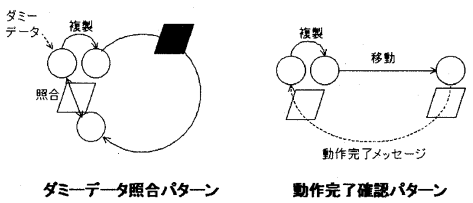

図 9 セキュリティ・セーフティパターン
Fig. 9 Security/safety patterns.

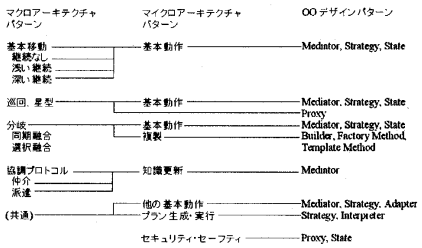

図 10 各ビヘイビアパターン間の関係
Fig. 10 Relationships of patterns.

3.2.3 各ビヘイビアパターン間の関係

3.3 節や図 6 でも示したように，各階層のビヘイビアパターン間には，開発プロセスに即した関係性が存在する．簡単にいえば，下位のビヘイビアパターンは上位のビヘイビアパターンを詳細に実現する際の構成要素となり，したがって，上位のビヘイビアパターンを適用して作成された概略設計の詳細化において，下位のビヘイビアパターンの適用が行われる．個々のビヘイビアパターン間の関係の例示として，Plangent のマイクロアーキテクチャパターンに関するものは**図 10** のようになる．図 10 において，「(共通)」というのは，いくつかのエージェントプラットホームに共通のマイクロアーキテクチャパターンということを意味する．

3.3 階層構造に基づく開発プロセス

以上の階層構造に基づいて開発プロセスを定義する．概要は次のとおりである．

- 上位の階層から順に，開発フェーズを割り振る．これにより，計 3 フェーズが定義できる．
- 各フェーズでは，ビヘイビアパターンを適用してアーキテクチャを設計する．
- ビヘイビアパターン適用のための条件を特定する

ために，これらのフェーズの前にパターン要求抽出フェーズを設定する．

以下，各フェーズについて詳述する．

3.3.1 パターン要求抽出フェーズ

本フェーズでは，以降の各アーキテクチャ設計フェーズに用いるビヘイビアパターンの適用条件を抽出する．ここでビヘイビアパターンの適用条件として，次のようなものがあげられる．

- モバイルエージェントシステムが稼動する環境において，利用可能な既存のインフラ，およびそれらの特性．

  ここでインフラとしてはハードウェアおよびソフトウェアの両方がある．ハードウェアに関しては，ネットワークを構成する計算機，およびネットワーク回線のそれぞれについて，型番や性能を抽出する．またソフトウェアに関しては，OSのような基盤レベルから，再利用するアプリケーションのような高位レベルまでを洗い出す．
- システムへの要求仕様における条件．

  システムへの要求仕様において，特に性能や取り扱うデータ量に関する条件を抽出する．

3.3.2 マクロアーキテクチャ設計フェーズ

本フェーズでは，前フェーズで得られたパターン適用条件にマッチしたマクロアーキテクチャパターンを具体化し，それを組み合わせることにより，与えられた要求仕様を満足するマクロアーキテクチャ設計仕様を作成する．

まずビヘイビアパターンの具体化の手順は次のとおりである．

(1) 前フェーズで洗い出したインフラの特性を整理し，適用条件がマッチするパターンを選択する．ただし，一般にインフラは，一群のホストやそのあいだの接続回線などに応じて，様々な特性を持つ部分から構成されるので，各部分ごとに異なるパターンを選択することになる．

(2) インフラの各部分に対し，選択したパターンを適用することにより，エージェント動作の概略をエージェント動作図として記述する．その後，それらの動作図を結合することにより，システム全体におけるエージェントの典型的動作の記述を作成する．

(3) 状態遷移図を作成するため，作成したエージェント動作図から，まずその動作図で示したエージェントの典型的動作を示す状態遷移図を作成する．その後，その状態遷移図を必要に応じて拡張・一般化することにより，一般的動作を示す状態遷移図とする．

(4) イベントシーケンス図を作成するため，作成した状態遷移図で示されるエージェント動作を，時間的な流れで記述する．

以上のようなプロセスを可能にするため，マクロアーキテクチャでは次の項目を明確にする必要がある．

- ビヘイビアパターンを構成する3種類の図表記に対し，マクロアーキテクチャ設計プロセスに沿って，順次一般化や時系列化により作成するためのガイドライン．
- ビヘイビアパターンを適用する，すなわち，与えられた問題への解となるように具体化するためのガイドライン．たとえば，どのようなパラメータをどこにあてはめるか，など．

3.3.3 マイクロアーキテクチャ設計フェーズ

本フェーズでは，前フェーズで得られたマクロアーキテクチャ設計仕様を詳細化することにより，マイクロアーキテクチャ設計仕様を作成する．詳しくは，図6でも示されているように，次の2つの方法を用いる．

- マクロアーキテクチャ設計仕様を構成する，マクロアーキテクチャパターンのインスタンスを，マイクロアーキテクチャパターンのインスタンスの組合せで詳細化する．
- システムにおいて，マクロアーキテクチャパターンのインスタンスの外部の部分を，マイクロアーキテクチャパターンを適用して設計する．

本フェーズでも，パターン適用条件が重要である．また，モバイルエージェントプラットホームの特徴をどう活かすかもポイントとなる．したがって，本フェーズで適用されるマイクロアーキテクチャパターンは，このような点を明示する．

ただし，マイクロアーキテクチャは，エージェントプラットホームに特化した部分の仕様であるため，マクロアーキテクチャの全体から得られるわけではない．マイクロアーキテクチャとして詳細化できない部分については，次のオブジェクトレベル設計・実装フェーズにおいて，オブジェクトレベルとして設計・実装を行う．

3.3.4 オブジェクトレベル設計・実装フェーズ

本フェーズでは，前フェーズで得られたマイクロアーキテクチャ設計仕様を詳細化することにより，オブジェクトレベルの設計を行い，実装まで進める．具体的には，マイクロアーキテクチャ設計仕様を構成する，マイクロアーキテクチャパターンのインスタンスを，OOデザインパターンを適用して実現し，システムとして組み合わせる．

4. 関連研究との比較

次に，関連研究との比較により，本手法の有効性を検討する．

従来よりデザインパターン[3]による設計が幅広く行われており，モバイルエージェントシステムの開発においても，そのような研究がある．

Aridor ら[1]は，エージェントのためのデザインパターンを，Traveling Pattern, Task Pattern, および Interaction Patterns の 3 種類に分けて，いくつか例示している．また，そのようなデザインパターンの利用例も示している．しかし，ここであげられているデザインパターンは，本論文における 2 階層の分類に関しては区別しておらず，利用例においても，どのような順序で適用を検討すべきか，といった観点が欠けている．したがって，デザインパターンの種類が増加していくと，適用の検討を行うのが困難になる．一方，本論文では，ビヘイビアパターンを例示するだけでなく，アーキテクチャレベルと詳細レベルに分類し，適用を検討する際の指針および順序を規定している．

また，Kendall ら[5]は，階層アーキテクチャを持つエージェントのためのデザインパターンについて考察している．ここでは，エージェントを構成する各階層に対し，それぞれ適用可能なデザインパターンを例示している．しかし，Kendall らがあげているデザインパターンは，個々のエージェントの実現に適用できるもののみである．すなわち，本論文における詳細エージェントパターン，あるいはそれよりも実装に近いレベルのデザインパターンしか検討しておらず，アーキテクチャレベルの設計については取り上げていない．たとえば，Kendall らは最も粒度の大きいデザインパターンとして，エージェントの移動を実現するためのデザインパターンを紹介している．一方，本論文では，現実的なエージェントシステムの開発においては，アーキテクチャレベルからビヘイビアパターンにより設計を支援する必要があるとの観点から，ビヘイビアパターンを整理した．

次に Silva ら[9]は，1 エージェントを中心にして，協調動作図（collaboration diagram）とクラス図を用いたパターンを提案している．しかし本パターンは，複数のエージェント間の相互作用，エージェントシステム全体の構成，およびエージェントのシステム全体における動作といった，実システム開発に必要な観点が欠けている．一方，本論文では，これらの観点を総合した開発プロセス，およびビヘイビアパターンを提示しているので，実際のシステム開発，特に設計段階全般の効率化を図ることができる．

5. おわりに

本論文では，開放型ネットワーク環境での運用を前提としたモバイルエージェントシステムを，効率的に開発するための手法を提案した．本手法は，ビヘイビアパターンと呼ばれる，エージェントの典型的な振舞いを示す概念に基づき，エージェントの複雑な動作を，ビヘイビアパターンを組み合わせたアーキテクチャとして整理することにより，システム設計を効率的に行うことが可能となる．

今後の課題としては，次のような項目をあげることができる．

- ビヘイビアパターンの抽出・検討
  本論文では，企業間異種アプリケーション接続例題をはじめとしたいくつかの例で，必要なビヘイビアパターンを抽出した．今後さらに Plangent の実適用を通じて，有用なビヘイビアパターンを抽出・検討していく．
- 体系的開発方法論への発展
  本論文では，3 フェーズから構成され，2 つのレベルのビヘイビアパターンを適用する設計プロセスを提案した．今後は要求分析・実装・保守・管理プロセスまで含めた，モバイルエージェントシステムの体系的開発方法論への発展を目指す．

謝辞 本研究の機会を与えてくださいました，コンピュータ・ネットワークラボラトリー相川健室長，ならびに有信睦弘・前 S&S 研究所所長には深く感謝いたします．

参考文献

1) Aridor, Y. and Lange, D.B.: Agent design patterns: Elements of agent application design, *Proc. Agents'98* (1998).
2) Bradshaw, J.: *Software Agents*. AAAI Press/ The MIT Press (1997).
3) Gamma, E., Helm, R., Johnson, R. and 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).
4) Garlan, D. and Shaw, M.: An introduction to software architecture, *Advances in Software Engineering and Knowledge Engineering*, Ambriola, V. and Tortora, G. (Eds), Vol.2 of Series on Software Engineering and Knowledge Engineering, pp.1–39, World Scientific Publishing Company (1993).
5) Kendall, E.A., Pathak, C.V., Krishna, P.V.M.

and Suresh, C.B.: The layered agent pattern language, *Proc. PLoP'97* (1997).
6) Kinny, D. and Georgeff, M.: Modelling and design of multi-agent systems, *Intelligent Agents III; Proc. 1996 Workshop on Agent Theories, Architectures, and Languages*, Müller, J.P., Wooldridge, M. and Jennings, N.R. (Eds) (1996).
7) Martin, D.L., Cheyer, A.J. and Moran, D.B.: Building distributed software systems with the open agent architecture. *Proc.3rd International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology* (1998).
8) Ohsuga, A., Nagai, Y., Irie, Y., Hattori, M. and Honiden, S.: Plangent: An approach to making mobile agents intelligent, *IEEE Internet Computing*, Vol.1, No.4, pp.50–57 (1997).
9) 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).
10) Tahara, Y., Ohsuga, A. and Honiden, S.: Agent system development method based on agent patterns, *Proc. 21st International Conference on Software Engineering*, pp.356–367, IEEE (1999).
11) IBM Corporation Tokyo Research Laboratory: IBM Aglets Software Development Kit – Home Page. `http://www.trl.ibm.co.jp/aglets/`.
12) 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`.
13) モーブレィ, T.J., マルボー, L.C.(著), 大谷真(監訳): *CORBA Design Pattern*—アーキテクチャからプログラミングまで, IDG コミュニケーションズ (1998).
14) 東芝: Plangent WWW ページ. `http://www2.toshiba.co.jp/plangent/`.

付　　録

A.1 パターンカタログの例：仲介パターン

本論文で採用するパターンテンプレートは，文献13)のものに基づいている．文献13)のテンプレートは，本論文が問題としているアーキテクチャレベルの問題意識に対応しているので，本論文のビヘイビアパターン記述に利用するのに適切であると考え，採用した．具体的には，本論文のパターンテンプレートは次の項目を含む．

- 名称：ビヘイビアパターンを識別するためだけでなく，その内容を簡潔に表現する名詞句．
- 概要：目的，動機，およびソリューション程度に絞った，ビヘイビアパターンの簡潔な説明．
- 目的：ビヘイビアパターンを適用することにより，解決を目指すべき問題の記述．
- 動機：ビヘイビアパターンが必要となった背景．すなわち，目的で記述した問題に対し，なぜビヘイビアパターンが必要か，またなぜ他のソリューションでは解決できないか，といった観点の説明．
- 適用可能性：ビヘイビアパターンが適用可能な条件の記述．
- ソリューション：問題に対するソリューションの詳細な記述．
- モデル図：ソリューションを図示するモデル図．
- 適用例：ビヘイビアパターンを具体的な問題に適用した例の詳細な記述．
- 結果：ビヘイビアパターンを適用した結果得られる利点，および場合によっては他のソリューションの方が適切であるような条件を示す注意事項の記述．
- 関連ビヘイビアパターン：ビヘイビアパターン適用の際に，他に考慮すべきビヘイビアパターンや，ビヘイビアパターン適用結果に対しその詳細部の実現時などに考慮すべきビヘイビアパターンなど，当該ビヘイビアパターンと関連が深いビヘイビアパターン．

テンプレートを用いたビヘイビアパターンの記述例として，仲介パターンの記述を以下に紹介する．

- 名称：仲介（Mediation）
- 概要：まず常駐型エージェントとは別に，仲介エージェントと呼ぶ移動型エージェントを用意する．仲介エージェントは，サイト間を移動しながら，常駐型エージェントの協調を仲介する．また，仲介エージェントの協調プロトコルを変更するだけで，システム全体の協調プロトコルの変更が可能である．
- 目的，適用可能性：帯域幅の狭いネットワークで，協調プロトコルを使用する．さらに，協調プロトコルの変更が考えられるような場合に特に有効である．
- 動機：ネットワーク上に分散して存在する異種アプリケーション群を接続したいという要求が高まっており，それに対して現在 CORBA などの分散オブジェクト技術が用いられている．また，分散オブジェクト技術における，アプリケーショ

ン相互接続のための協調処理の記述を簡略化することを目的として，直接交渉を行う協調エージェントを利用することも検討されている．しかしこれらの技術では，各アプリケーションでのローカルな処理と，接続のためのグローバルな処理とが混在して組み込まれるので，それぞれ個別に変更したいといった柔軟性に欠ける．そこで，後者のグローバルな処理は仲介エージェントに組み込み，ローカルな処理のみ組み込まれた常駐エージェントの存在するホスト間を移動させることにより，そのような柔軟性を実現する．

- ソリューション：本ビヘイビアパターンでは，ネットワークで接続された複数のホスト上で稼働する常駐エージェントと，それらの間を移動しながら，常駐エージェントとローカルに相互作用を行う，モバイル仲介エージェントとを用意する．
- モデル図：エージェント動作図テンプレート，および状態遷移図テンプレートでは，モバイル仲介エージェントの動作を中心に記述する．これは，常駐エージェントはモバイル仲介エージェントとの相互作用のみ行うのに対し，モバイル仲介エージェントは，相互作用と移動動作の両方を通じて，システム全体の連携動作を実現するからである．**図11，12**のように，モバイル仲介エージェントの動作は，ローカル相互作用，移動開始，移動終了の3つの状態を繰り返すものである．またローカル相互作用においては，モバイル仲介エージェントと，常駐エージェントとの相互作用を表現する状態遷移図を，下位の動作として記述する．
  またイベントシーケンス図テンプレート（**図13**）では，各ホストでの動作を分離して記述する．各ホストでの動作を表す点線長方形の内部には，各プロセスのライフラインと，それらの間のメッセージ矢印が記述される．プロセスとしては，そのホストで稼働している常駐エージェントとモバイル仲介エージェントの各プロセスを記述する．なおモバイル仲介エージェントはホスト間を移動するが，そのようなイベントは，ホスト間をまたがる点線矢印で表示する．また，移動イベント矢印は，移動前のホストでのエージェントプロセスの停止を示す，ライフラインの矢印の終端と，移動後のホストでのエージェントプロセスの開始を示す，ライフラインの矢印の始点とを結ぶ．
- 適用例：近年，EC，CALS，SCMといった，企業間異種アプリケーション接続が必要なシステム構築への要求が高まっている．このような要求に

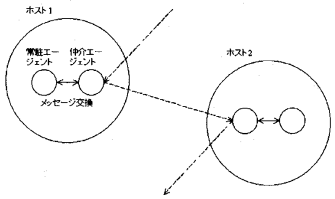

図11 仲介パターンのエージェント動作図テンプレート
Fig. 11 Agent behavior diagram template of the mediation pattern.

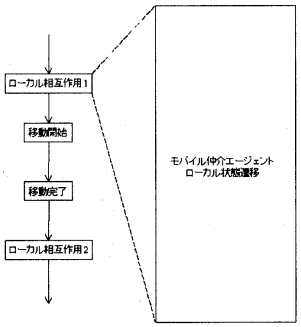

図12 仲介パターンの状態遷移図テンプレート
Fig. 12 State transition diagram template of the mediation pattern.

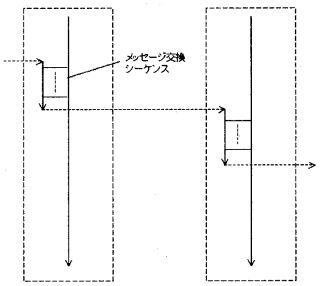

図13 仲介パターンのイベントシーケンス図テンプレート
Fig. 13 Event sequence chart template of the mediation pattern.

対しては，CORBAやJava RMIなどといった，分散オブジェクト技術が，最新技術として普及しつつある．しかし，これらの技術では，プラットホームベンダ間の相互連携や，変更に対する柔軟さといった点で問題がある．一方，本ビヘイビアパターンを適用することにより，上述の説明で示したように，これらの課題を解決できる．

まず，本例題の問題設定を，次のように考える．本例題では，3つの企業（A，B，C社とする）にそれぞれ1つずつホストがあり，それらがいわゆ

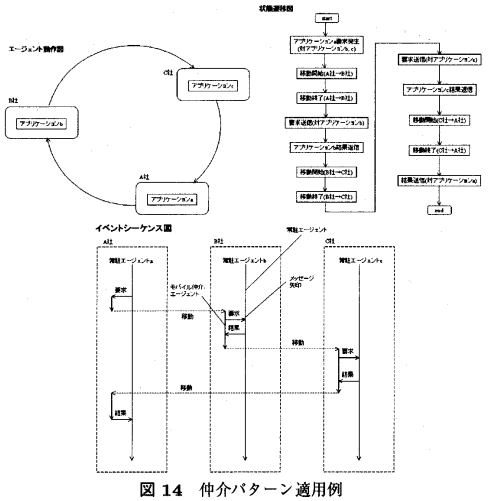

図 14 仲介パターン適用例
Fig. 14 Application example of the mediation pattern.

るエクストラネットで接続されている．これらのホストに，3つの異なるアプリケーション（それぞれa，b，cとする）が稼働している．このような状況で，アプリケーションaから，他の2アプリケーションに要求を出し，結果を返してもらう，というシステムを構築したいとする．

これに対し，まずパターン要求抽出フェーズは概略次のように進められる．本例題において，たとえば次のような条件が抽出される．

- 企業間はエクストラネットのため，トラフィック軽減が必要である．
- サービス要求先企業の変更が頻繁に必要となる．

これらの条件から，仲介パターンの適用が適切であると判断する．すなわち，各アプリケーションのインタフェースを実現する常駐エージェントを各ホストに用意し，さらにネットワークを移動して，これらの常駐エージェントの相互作用を仲介するモバイルエージェントを用意することにより，仲介パターンの適切な適用が可能となる．

次に，マクロアーキテクチャ設計フェーズにおいて仲介パターンを適用した結果を，**図14**に示す．

次に，Aglets および Plangent による実装例を次に示す．

- Aglets による実装例
  Aglets においては，マクロアーキテクチャの巡回パターンは，マイクロアーキテクチャ設計フェーズにおいて，マイクロアーキテクチャの Itinerary パターンで実現する（**図15**）．

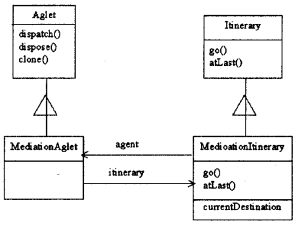

図 15 Aglets Itinerary パターンによる実現例
Fig. 15 Implementation example using the aglets itinerary pattern.

さらに，オブジェクト設計・実装フェーズにおいて実装されたコード例を以下に示す．

```
import com.ibm.aglet.*;
import com.ibm.aglet.util.*;

public class MediationItinerary {

  String[] destinations;
  String[] tasks;
  String currentDestination;
  Aglet aglet;

  public MediationItinerary(
          String[] destinations,
          String[] tasks,
          Aglet aglet) extends Itinerary {
    this.destinations = destinations;
    this.tasks = tasks;
    this.aglet = aglet;
  }

  public void go() {
    // 次の目的地に移動
  }

  public void atLast() {
  }

  public void executeTask() {
    // tasks の先頭のタスクを aglet に実行させ,
    // その後削除.
  }
}

public class MediationAglet extends Aglet {

  MediationItinerary itinerary;

  public void onCreation(
              String[] destinations,
              String[] tasks,
              Object init) {
    itinerary = new SimpleItinerary(
                destinations, tasks, this);
    itinerary.go();
  }

  public void run() {
```

```
    itinerary.executeTask();
  }

  public ... // 各タスクの実行メソッド
}
```

- Plangent による実装

  Plangent では，仲介パターンを，マイクロアーキテクチャ設計フェーズにおいて，次のように実現する．

  * 各移動に対しては，基本移動パターンを適用する．
  * 各ローカルでの動作には，その他の基本動作パターンを適用する．

  さらに，オブジェクト設計・実装フェーズにおいては次のようになる．

- 結果：本ビヘイビアパターンの適用により，アプリケーションを接続するためのグローバルな処理は，仲介エージェントが一括して管理するので，変更に対して柔軟に対応可能なシステムが構築できる．たとえば，
  - グローバルな協調プロトコルの変更においては，仲介エージェントに変更を加えるだけでよく，各アプリケーションや常駐エージェントの変更は不要である．
  - 逆にローカルな処理の変更においては，変更するアプリケーションの存在するホストの常駐エージェント，および仲介エージェントとそのエージェントやアプリケーションとの相互作用の部分のみの変更でよく，その他のアプリケーションや常駐エージェントの変更は不要である．
- 関連ビヘイビアパターン：仲介エージェントの移動において，各移動パターンの適用を検討する．また，協調方式に関しては，各協調プロトコルパターンの適用を検討する．

(平成 11 年 2 月 18 日受付)
(平成 11 年 10 月 7 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．1993～1996 年情報処理振興事業協会に出向．1996～1997 年英国 City 大学客員研究員．1997～1998 年英国 Imperial College 客員研究員．現在（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．

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．1985～1989 年（財）新世代コンピュータ技術開発機構に出向．現在（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に所属．工学博士．主としてソフトウェアのためのフォーマルメソッド，エージェント技術等の研究に従事．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 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．工学博士．同年（株）東芝入社．現在，同社官公システム事業部に所属．1989 年より早稲田大学非常勤講師を兼任．1996 年度より大阪大学大学院非常勤講師兼任．主としてエージェント技術，オブジェクト指向技術の研究に従事．1986 年度情報処理学会論文賞受賞．日本ソフトウェア科学会，IEEE 各会員．著訳書多数．