3S-02　移動エージェント相互運用の実現：基本アーキテクチャ

粂野 文洋 **　長 健太 *　長谷川 哲夫 *　大須賀 昭彦 *　中島 震 ***
株式会社東芝 *　株式会社三菱総合研究所 **　NEC***

1　はじめに

本稿では、異なる移動エージェント基盤間の相互運用を実現する相互運用ミドルウェアの基本アーキテクチャについて概説する。まず、本アーキテクチャの設計思想を説明した上で、アーキテクチャの構成要素およびその間の関係を概説し、移動エージェント基盤間の相互運用がどのように実現されるのかを明らかにする。また、本研究開発で実現するミドルウェアに基づく相互運用の実現例についても具体的なアプリケーションを用いて概説する。

2　相互運用ミドルウェア・基本アーキテクチャ

異種の移動エージェント基盤の間には、移動エージェントの構造や計算モデル、実装方式、実装言語など様々な違いがあり、エージェントが異なる移動エージェント基盤に直接移動する相互運用を実現するのは極めて困難である。そこで、本研究プロジェクトでは、エージェントが内部に持つ自分の意図情報を移動エージェント基盤に依存しない形に翻訳した後、移動先の移動エージェント基盤に移送し、その移動エージェント基盤用に逆翻訳し、送信先で移動エージェントを起動実行させるインカネーションエージェントを提案している。この仕組みにより、エージェントの直接の移動、すなわち物理的な移動が不可能な移動エージェント基盤間においても、論理的な移動を実現することが可能となる。

2.1　基本アーキテクチャの設計思想

インカネーションエージェントの各動作（意図やライフサイクル情報の抽出、意図の共通表現への翻訳、移動、移動先上のエージェントとしての処理継続）を実現するには、移動エージェント基盤間で通信されるエージェントの意図情報 (本稿ではこれをタスクと呼ぶ) に関して、大きく分けて次の 3 フェーズの処理を実装する必要がある。

抽出・翻訳フェーズ：　移動エージェント基盤からタスクを送出し、移動エージェント基盤に依存しない形式 (交換タスクと呼ぶ) に変換する。

移動フェーズ：　交換タスクを送信先の移動エージェント基盤に送信する。

挿入フェーズ：　送信先の移動エージェント基盤が受けとる前に交換タスクをその移動エージェント基盤が解釈実行できるタスクに変換し、そのタスクを移動エージェント基盤に処理を依頼する。

具体的な移動エージェント基盤間で各フェーズを実現するには、1) 移動エージェント基盤はどのような情報を、何をトリガにして、タスクとして送出するのか、2) 交換タスクの構成要素およびそれを表現する枠組み、3) 各移動エージェント基盤毎のタスクと交換タスクとの対応関係およびタスクの解釈実行方法、を規定する必要がある。各項目の具体的内容は、2) の交換タスクの枠組みを除いては、個々の移動エージェント基盤の特性、実現したい相互運用機能の内容、アプリケーションに深く依存する。そこで、相互運用ミドルウェアを実現するための基本アーキテクチャとしては、上記の各項目については規定をせず、各項目を実現したモジュールを柔軟に変更できるオープンアーキテクチャを目指す。そして、ミドルウェアが相互運用の目的や移動エージェント基盤の特性に合わせ、様々なバリエーションの相互運用形態を実現する実行基盤となることを目標とする。なお、OMG や FIPA などの標準化仕様を考慮し、交換タスクは FIPA ACL、通信基盤は CORBA/IIOP をベースとする。

2.2　相互運用ミドルウェアのシステム構成

相互運用ミドルウェアのシステム構成を図 1に示す。基本アーキテクチャは、接続アダプタ、移動エージェント通信チャネル、ミドルウェア通信基盤、ディレクトリから構成される。各構成要素について以下に説明する。

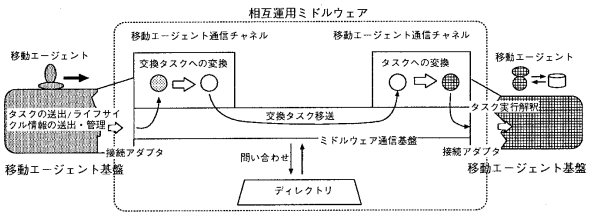

図 1: ミドルウェアのシステム構成

2.2.1　接続アダプタ

移動エージェント基盤からタスクやライフサイクル情報をミドルウェアに送出したり、ミドルウェアから送られてきたタスクを解釈実行するためのモジュールである。このモジュールは移動エージェント基盤およびミドルウェア間のインタフェースを確立する部分であり、移動エージェント基盤の拡張として実装される。

タスクには送信先の移動エージェント基盤が実行するアクション情報およびエージェントが移動先で実行するアクション情報が含まれている。ライフサイクル情報には、インカネーションエージェント起動直後の実行状態を表現した情報が含まれる。タスクの表現、定義、生成、管理、インカネーションエージェント起動の方法およびライフサイクル情報の生成・管理方法は、各エージェント移動エージェント基盤毎に規定される。

移動エージェント基盤接続アダプタは、移動エージェント基盤処理系本体またはエージェント自身から、タスク表現やライフサイクル情報の文字列（またはバイトコード）を受けとり、それを移動エージェント通信チャネルに送信する機能を最低限のタスク送出機能として持つ。一方、タスクを解釈実行する機能は、タスク内のアクション情報を

Interoperability Middleware for Mobile Agents: Basic Architecture
Fumihiro Kumeno, Kenta Cho, Tetsuo Hasegawa, Akihiko Ohsuga, Shin Nakajima
Mitsubishi Research Institute, Inc., Toshiba Corporation, NEC Corporation

解釈して、移動エージェント基盤の内部機能を呼び出す一種の wrapper 機能である。

これらの機能の実装にあたっては、移動エージェント基盤本体と接続アダプタの結合度を低くすることにより、移動エージェント基盤内部に極力手を加えずにアドオン式でミドルウェアと接続可能となることが期待できる。一方、移動エージェント基盤内部にも手を加え、より密に結合させることを前提とすれば、アプリケーションプログラマが移動エージェント基盤の違いを意識せずに移動エージェントアプリケーションを構築できるような高度な相互運用性を実現できる可能性もある。

2.2.2 移動エージェント通信チャネル

移動エージェント通信チャネルは1つの移動エージェント基盤に対応しており、移動エージェント基盤間のタスク通信中継を行なう。移動エージェント通信チャネルは中継するタスクを交換タスクに変換し、送信先の移動エージェント通信チャネルに移送する。交換タスクを受けとった移動エージェント通信チャネルは対応する移動エージェント基盤用のタスクに復号化し、その移動エージェント基盤の接続アダプタに送信する。

交換タスクは移動エージェント基盤に依存しないという意味で抽象的なアクション表現であるため、交換タスクへの翻訳／逆翻訳を通したタスク中継により、ミドルウェアに接続したあらゆる移動エージェント基盤間でのタスクのやりとりが可能となる。ただし、これを実現するには、各移動エージェント基盤毎に変換プログラムを実装し、それをミドルウェアに登録する必要がある。変換処理には文法的な変換と意味的な変換 (用語変換) がある。文法的な変換プログラムの実現には JavaCC[3] などのコンパイラ・コンパイラを利用し、用語変換はあらかじめ用意された用語変換表に基づく置換処理によって実現する。また、用語変換はアプリケーション毎 (もしくはアプリケーションドメイン毎) に用意する必要がある。

2.2.3 ディレクトリ

ディレクトリでは、各移動エージェント基盤の情報など、異なる移動エージェント基盤の間で必要となるグローバルな情報を登録、管理する。たとえば、移動エージェント通信チャネルが交換タスクの送信する場合、ディレクトリに問い合わせを行ない、送信先の移動エージェント通信チャネルのアドレス情報を特定する。また、このディレクトリ機能は OMG の MASIF 仕様の拡張となっており、各アプリケーション間の相互運用機能の実現に利用する文法変換や用語変換の情報もディレクトリで管理できる。

2.2.4 ミドルウェア通信基盤

移動エージェント基盤 ↔ 移動エージェント通信チャネル間および移動エージェント通信チャネル同士の通信はミドルウェア通信基盤を介して行なわる。ここで、移動エージェント基盤がベースとしている通信プロトコル (たとえば、RMI、Plain socket 通信) との変換プログラムを組み込むことができ、通信実装方式の違いが吸収される。また、このモジュールによる通信路がセキュアでない移動エージェントにとっての抜け道になるのを防ぐために、フィルタリング機能も導入する。

3 相互運用の実現例

異なる移動エージェント基盤上に同じ目的のアプリケーションが実装されていることを想定し、これらの間での相互運用を実現する例を与える。

3.1 オフィス情報共有システム

相互運用機能を実現するアプリケーション例として、移動エージェントによるオフィス情報の共有支援システムを示す。このアプリケーションは、複数のネットワークセグメントで構成されるイントラネット上において、社員が各自で公開しているデータ (スケジュールデータなど) を移動エージェントを用いてリアルタイムに収集し、スケジュール調整等の処理を行なうアプリケーションである。

3.2 相互運用機能の実現例

現在、二つの既存の移動エージェント基盤、Plangent[1] および Flage[2] 上でオフィス情報共有システムを実装している。Plangent は Java 上に構築された移動エージェント基盤であり、独自のスクリプト言語により移動エージェントを定義する。一方、Flage は Quintus Prolog ベースの移動エージェント基盤であり、こちらも独自の言語によりエージェント記述を行なう。各システムの機能仕様は同一であるが、それぞれの移動エージェント基盤特性に応じて独立に実装されており、収集するデータのフォーマットも別々に設計されている。

相互運用の実現イメージを図2に示す。Plangent のアプリケーション上で情報収集を行なう移動エージェントは、相互運用ミドルウェアを介して、Flage 上の社員公開データをも獲得でき、またその逆も可能となっている。

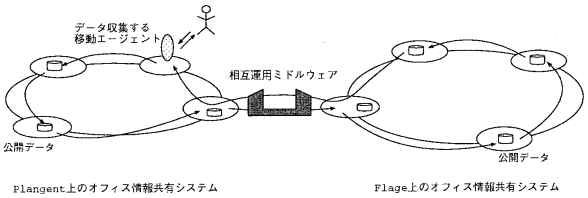

図 2: オフィス情報共有システムの相互運用

この相互運用の実現により、従来2つのアプリケーションごと行なわれていたオフィス情報のデータ収集・処理を相互に行なうことができ、より広範囲なアプリケーション適用が可能となる。

4 まとめ

本稿では相互運用ミドルウェアの基本アーキテクチャと相互運用機能の実現例について概説した。現在、相互運用ミドルウェアは Java で実装中であり、Plangent や Flage との接続実験を行なっている最中である。基本アーキテクチャの実装は、相互運用の目的や移動エージェント基盤の特性に合わせ、タスクの構成やその解釈実行方法などに組み変えることにより、様々なバリエーションの相互運用形態を実現する、実行環境としての機能を提供することを目標としている。そのためには、実装後の課題として、様々な移動エージェント基盤を本ミドルウェアに接続する実験を行ない、フィードバック＆改良を加えながら、数多くの移動エージェント基盤間の相互運用を実現するミドルウェアにしていく必要がある。

謝辞 本研究は、「IPA 次世代デジタル応用基盤技術開発事業」の一環として行なわれているものである。

参考文献

[1] http://www2.toshiba.co.jp/plangent
[2] http://www.ipa.go.jp/NEWSOFT/public/Flage
[3] http://www.metamata.com/JavaCC