エージェント――柔軟なシステムを目指して

Agent――Adding Flexibility to Computer System

戸　叶　　徹

要　約　エージェントとは，自律性，目的指向性，協調性をもったソフトウェアである．各研究機関・企業では，エージェントの研究・開発を活発に行っており，それにともなって，エージェントの標準化作業も始まっている．

上記のことに鑑み，本稿では以下のことを述べる．

・エージェントの概要
・エージェントシステムの例
・エージェント言語
・標準化動向
・弊社の取り組みの一例

Abstract　Agent is an autonomous, objective-oriented and collaborative software. Various research organizations and companies today are seriously engaged in the research and development of agent software. Consequently, the effort for standardization of agent is also underway. Taking into account the circumstances mentioned above, this article describes:

Outline of Agent
Examples of the Agent System
Agent Language
Standardization of Agent
Our Company's Agent System

1. はじめに

コンピュータの高機能化/低価格化やネットワークの発展により，コンピュータ資源の分散化が著しく進んでいる．それを背景に，クライアント/サーバコンピューティング環境は成熟期に入り，分散オブジェクト環境への展開が始まっている．しかし，今後ネットワークノード上に大量のオブジェクトが生成され，動作すると，オブジェクトの氾濫，その結果としてのシステムの不安定化，硬直化という事態に陥ることは容易に想像できる．

一方，次世代のソフトウェアとして注目されているエージェントは，システムに柔軟性を付与するものとして，ここ数年大きな注目を浴びている．国内外の研究機関・企業は，エージェントを積極的に取り入れようとしている．

EC の分野においても，広大なサイバースペース上の情報を如何に利用者へ提供できるかが大きな問題となっており，情報の利用者と提供者を結びつけるものとしてエージェントへの関心が高まっている．

本稿では，これらの現状に鑑み，まずエージェントの概論を述べ，次に代表的なエージェントシステム，エージェント言語を述べる．その次に，標準化団体である FIPA（Foundation for Intelligent Physical Agent）や OMG（Object Management Group）

が進めているエージェントの標準化活動を解説し，最後に弊社が開発に取り組んでいるエージェント技術を利用したシステムについて記述する．

2. エージェントの概念

2.1 エージェントの定義

現在，エージェントは非常に広い意味で使用されており，明確な定義がなされているわけではない．しかし，本稿では，「人間（または，ソフトウェアやハードウェア）の依頼に基づいて，依頼主に代わって，依頼主の目的とする処理を実行するソフトウェア」と定義することとする．その定義に基づくと，エージェントは以下のような性質を持つ必要がある．

1） 自律性

自らの状態を把握し，外部の状態や今までの経験に基づいて能動的に行動することができる性質である．エージェントが自律的に行動するためには，自分自身の知識を持ち，経験を貯え，情報を収集しなければならない．

2） 目的指向性

自己の目的のために自らの処理方法を決定して動作するという性質である．エージェントは他のエージェントやプログラムから依頼され，その処理を行うこともあるが，最終的には自己の目的を目指した動作をしなければならない．

3） 協調性

他のエージェントやユーザと協調して動作する性質である．他と協調することにより，複数のエージェントが共通の目標に向かって処理を行うというようなことも実現できる

2.2 エージェントシステムの種類とその応用分野

エージェント技術を適用したエージェントシステムは，様々な企業，研究機関で開発されつつあるが，その種類と応用分野は以下のタイプに分けることができる[*1]．

1） 擬人化エージェント

人間を擬したキャラクタを使用したインタフェースである．このようなエージェントは，顔や体を持ち，視覚，音声，表情，身振りなど，人間の会話に使用される複数の様態を使用し，人間と情報のやり取りを行う．代表的なものとして，TOSBURG II，ニューロベイビ，VISA などがある．インターネット上でオンラインショッピングを構築する際に，ユーザインタフェースに応用できる．

2） 代理人エージェント

ユーザから与えられた権限に基づき，ユーザの代理となって業務を達成する．ユーザの意思は，「パラメータ，キーワード」，「スクリプト」，「学習」などにより，エージェントに知らせる．代表例として，Quarterdeck 社[*2] の WebCompass があげられる．インターネット上の情報検索などに応用できる．

3） 他のエージェントと交渉能力を持つプログラム

独立した目標を持つ複数のエージェントが互いに協調しあうことによって，各々の目標を達成するエージェントシステムである．エージェントは互いに交渉して，相互の利益の最大化のための局所的協調動作や競合の解消に関する合意の

形成，説得を行う．電子的なオークションなどに応用できる．

4) 共通言語で相互作用するプログラム

知識の共有・再利用の研究から生まれたもので，もともと相互運用することを念頭におかずに構築されたソフトウェアをカプセル化して，共通言語を話す仮想知識ベースに仕立て上げて，相互運用性を獲得しようというものである．電子商取引におけるカタログ流通に応用することができる．

5) 移動スクリプト

ネットワーク内のある計算機から別の計算機へ移動して実行されるプログラムである．クライアントからサーバへ移動するアップロード型のものと，サーバからクライアントへ移動するダウンロード型のものがある．アップロード型として，General Magic 社[*3] の Telescript，IBM 社の Aglets[*4] などがあげられ，ダウンロード型として，Java[*5] の Applet をあげることができる．

2.3 エージェントがもたらす利点

前述の性質を満たすエージェントは，今までの情報技術では実現することが困難であった，以下のような利点をもたらす．

1) 柔軟性

今までのプログラムは，プログラム間の連携方法をアルゴリズムとして決めておかねばならない．しかし，エージェントは自分の目的を達成するために，自律的に特定のサービスを提供するプログラムを見つけて連携することができる．

2) 協調性

あらかじめ決められた共通言語を介して，エージェント間で要求や応答をすることによって，エージェント同士を協調させることができる．

3) モバイルコンピューティング

移動するエージェントを，サービスを提供しているコンピュータに送り込むことによって，利用者が端末を常時接続しておく必要がなくなる．

4) ネットワーク通信負荷の軽減

クライアント/サーバ環境においては，クライアントからサーバへリクエストを発行するたびにネットワーク上にメッセージが流れる．しかし，エージェントが移動することにより，目的の処理を適切な場所で行うことができる．その結果，ネットワーク上を流れるメッセージの量を削減することができる．

5) 保守容易性

必要時に必要なエージェントをシステムに追加することができるので，システムの機能追加を動的に行うことができる．

3. エージェントシステム

2 章で，エージェントの性質や特徴を述べたが，本章ではエージェントシステムを紹介する．エージェントシステムとは，エージェントが動作するための基盤となるシステムのことである．現在，エージェントシステムは，各社，各研究機関で様々な研究がおこなわれているが，ここでは，エージェント指向プログラミングの先駆けとなった Agent-0 と，現在もっとも注目されているプログラムミング言語である Java 言

語でエージェントを記述することができる JATLite を紹介する．

3.1 Agent-0[*6]

スタンフォード大学の Shoham は，OOP（Object Oriented Programming）を発展させた AOP（Agent Oriented Programming）を提唱し，AOP に基づく言語処理系である Agent-0 を考案した．

3.1.1 AOP

AOP は，OOP の概念を拡張したものである．Shoham は，AOP と OOP を表 1 のように比較した．

表 1 AOP と OOP の比較

	AOP	OOP
構成要素	エージェント	オブジェクト
構成要素の内部状態を定義するパラメータ	信念，責務，能力など	決まっていない
計算プロセス	メッセージ通信 応答メソッド	メッセージ通信 応答メソッド
メッセージ種別	通知，要求，提案，契約など	決まっていない
メソッドの制約	誠実性，一貫性など	なし

3.1.2 Agent-0 処理系

Agent-0 におけるエージェントとは，心的状態（Mental State）やコミットメントルールを持ち，外部からのメッセージによって信念（Belief）や責務（Obligation）を更新し，コミットメントルールによって処理を実行する．心的状態は信念と責務で表現され，能力は動作と起動条件で表現される(信念，責務，コミットメントルール，

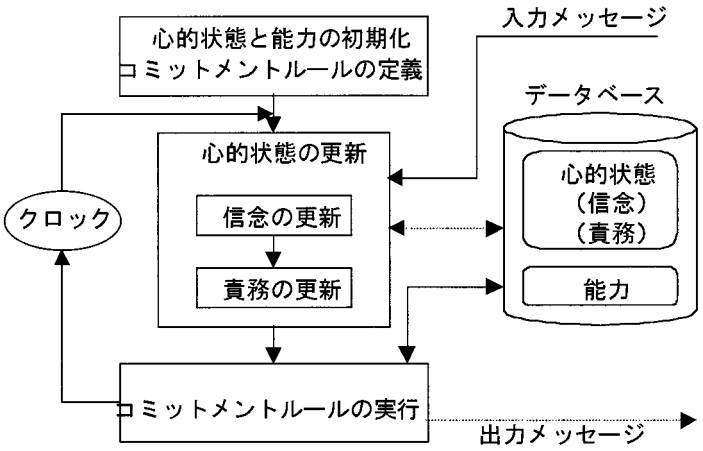

図 1 Agent-0 の処理系

動作については後述).

Agent-0 の処理系は，図 1 のようになる．

Agent-0 のエージェントは初期状態として，「心的状態と能力」と「コミットメントルールの定義」を与えられる．その後は，基本的に以下の処理を繰り返す．

- クロックに基づいて入力メッセージを受け入れる．
- 自分自身の心的状態を更新する．
- コミットメントルールを起動し，心的状態や能力を更新するとともに，他エージェントへメッセージを送る．

3.1.3 信念・責務・コミットメントルール

Agent-0 において，「信念」とは事実の集合であり，「責務」とはエージェントが達成すべき事柄と定義される．

信念は，B（T，X，F）と表現される．この意味は，「エージェント X が時刻 T において事柄 F が成り立つと信じている」ということである．責務は，OBL（T X Y F）と表現される．この意味は，「エージェント X がエージェント Y に対して時刻 T において事柄 F を達成する責務を負っている」という意味である．

この表現を応用すると，

B(10 : 00 X（Employee Tokano Unisys))

は，「Agent X が時刻 10 : 00 において，Tokano が Unisys のメンバであるという事実を信じている」という意味になり，

OBL(15 : 00 X Y（RESERVE flight＃1 2 seats))

は，「Agent X が時刻 15 : 00 において，Agent Y に対し flight＃1 の seat を 2 席予約する責務を負っている」という意味になる．

Agent-0 において，エージェントの実行処理はコミットメントルールというルールで記述される．コミットメントルールは，

（COMMIT＜MsgCond＞＜MntlCond＞(＜Agent＞ ＜Action＞))

という形で表される．MsgCond は，「あるエージェントがある時刻に受け取るメッセージの状況を示したもの」である．MntlCond は，「ある時刻におけるエージェントの心的状態で」ある．上式の意味は，「MsgCond のメッセージがあり，かつ MntlCond の状態であるときにおいて，指定された Agent に対する Action を実行する」ということを表現したものである．たとえば，

```
（COMMIT(？X REQUEST ?ACTION)
        (B（now（myfriend ?X)))
        (？X ?ACTION))
```

は，「あるエージェント X から何らかの動作 ACTION に関するメッセージを受け取ったとき，現在エージェント X が自分の友人だと信じているのならば依頼された動作 ACTION を実行する」という意味である（？が付与された記号は変数とみなされる).

3.1.4 動作プリミティブ

エージェント自身が独自に実行する動作は，動作プリミティブとして以下の 3 種類が定義されている．

1） 固有動作

（DO＜TIME＞＜Action＞）

と表される．

＜TIME＞で指定されている時刻に，＜Action＞で示されている動作を実行するという意味である．

2） 条件付き動作

（IF＜MntlCond＞＜Action＞）

と表される．

＜MntlCond＞で表される心的状態が成立した時に，＜Action＞で示される動作を実行するという意味である．

3） 動作抑制

（REFRAIN＜Action＞）

と表される．

＜Action＞で示される動作を抑制するという意味である．

3.1.5 通信プリミティブ

他のエージェントとの通信は，以下の 3 種類の通信プリミティブによって行われる．

1） 通　　知

（INFORM＜Time＞＜Agent＞＜Fact＞）

と表される．

＜Time＞で指定されている時刻に，＜Fact＞で指定された事実を，＜Agent＞で指定されたエージェントへ通知するという意味である．

2） 要　　求

（REQUEST ＜Time＞ ＜Agent＞ ＜Action＞

と表される．

＜Time＞で指定された時刻に，＜Action＞で指定された動作を，＜Agent＞で指定されたエージェントに要求するという意味である．

3） 要求の無効化

（UNREQUEST ＜Time＞ ＜Agent＞ ＜Action＞

と表される．

＜Time＞で指定された時刻に，＜Action＞で指定された動作を＜Agent＞で指定されたエージェントへ無効化要求をするという意味である．

3.2 JATLite[*7]

Stanford 大学はエージェント記述言語に Java 言語を使用したエージェントシステムである JATLite を開発している．JATLite は Java で記述されたクラスライブラリパッケージであり，インターネット上でエージェント同士が相互に通信できるエージェントを生成することができる．

3.2.1 JATLite のメッセージ通信

図 2 に示すように，JATLite のエージェント（図では，Java　Standalone/Applet と Agent Wrapper がエージェントである）では，名前とパスワードにより Agent Message Router に登録され，エージェントは Agent Messge Router 経由でメッセー

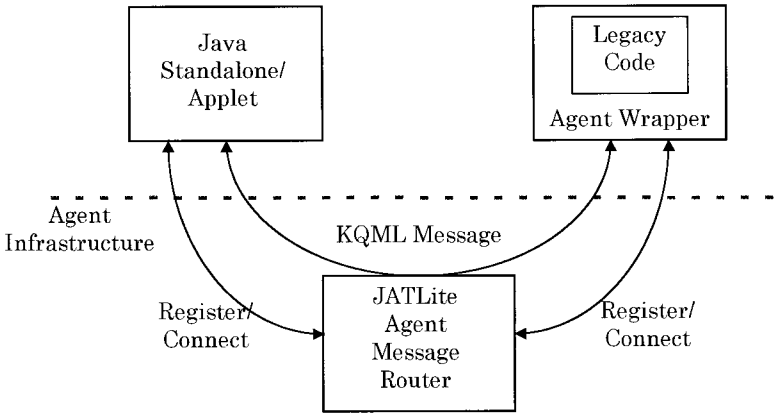

図 2　JATLite のメッセージ送受信

ジの交換を行う．

3.2.2　JATLite の構造

JATLite は，ユーザがエージェントを作成するための雛形（Template）を提供するものであり，知的なエージェントそのものを提供するわけではない．エージェントプログラム開発者は JATLite が提供する Java のクラスライブラリのパッケージを使用することにより，さまざまな言語やプロトコルが使用可能なエージェントを作成することができる．例えば，JATLite には，KQML（Knowledge Query and Manipulation Language）のパッケージが備わっており，エージェント間で KQML メッセージのやり取りができる．

図 3 のように，JATLite のパッケージは階層構造になっており，開発者はシステムの構築時に任意の階層を選び開発することができる（Abstract Layer はすべての層から利用できる）．

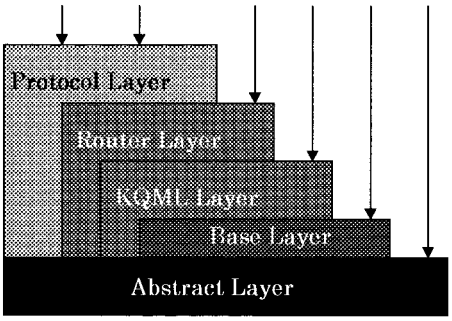

図 3　JATLite 階層図

各々の階層は以下の機能を有している．

1)　Abstract Layer

上位層で実装されるべきクラスを有した抽象的な層である．JATLite は TCP/IP での通信を想定しているが，この層を拡張することによって，他のプロトコル（例えば UDP）を使用することができる．

2） Base Layer

TCP/IP プロトコルによる，基本的な通信機能を提供する．この層を拡張することにより，例えば，ソケットからの入力をファイルに出力するようなことができるようになる．

3） KQML Layer

KQML（後述）メッセージの蓄積及び解釈機能を提供する．

4） Router Layer

エージェント名の登録やメッセージの経路制御，蓄積を行う．全てのエージェントはメッセージの送信，受信をルータ経由で行う．エージェントがルータから切断されたり，クラッシュしたりした場合，ルータはそのエージェント宛てのメッセージをため込み，エージェントが再接続したときにメッセージを送信する．

5） Protocol Layer

SMTP，FTP，POP 3，HTTP のようなさまざまなインターネットプロトコルをサポートする．エージェントが SMTP を使用してメッセージの交換場合や，FTP を使用して非常に長いデータを送る場合，Protocol Layer のクラスで提供されている関数を使用することができる．現バージョンは，SMTP と FTP が提供されている．

4. エージェント言語

エージェントが互いに協調して動作するためには，お互いが理解できる言語を取り決めなければならない．そのための言語として，様々な言語（Agent Communication Language）が考案されている．また，エージェントの知識を表現するための言語として，知識表現言語（Knowledge Interchange Format），共通の語彙概念を規定するオントロジーなどがある．以下にそれぞれを説明する．

4.1 エージェント間通信言語

エージェント間通信言語は，エージェント間のメッセージのやり取りを規定するための言語である．代表的な言語として，KQML（Knowledge Query and Manipulation Language）[*8] がある．KQML は，ARPA（Advanced Research Projects Agency）の知識共有プロジェクトで開発されたものであり，分散された知識システム間での知識の再利用，共有を目的に考えられた．

KQML はメッセージフォーマット兼エージェント間通信プロトコルであり，発話行為理論[*9] に基づいている．KQML メッセージは，動作を表す命令（Performative）とその引き数（キーワード/値のペア）およびメッセージ内容（Contents）から成る．命令と引数については予約語が定義されているが，メッセージ内容を記述する言語については規定しない．

各エージェントは，およびキーワードのサブセットのみを使用することもできるし，拡張定義することもできる．

たとえば,

(tell : sender agent 1 : receiver agent 2
: language prolog
: ontology Genealogy
: content " father (Taro, Jiro) ")　　(＊)

は,「Taro の父親は Jiro であることを agent 1 が agent 2 に通知する」という意味であり，知識表現言語（後述）は prolog で記述されており，オントロジ（後述）は Genealogy を使用している.

KQML の命令は表 2 に示すようなものがある.

表 2　KQML の命令抜粋

名前	意味
achieve	S は R にその物理環境で何かを達成することを要求する.
ask-all	S はすべての R に回答を要求する.
ask-if	S は R が知っているかどうかをたずねる.
ask-one	S は R のなかの一つに回答を要求する.
deny	S は performative が真実でないことを知らせる.
discard	S は R に以前の performative に対する回答を要求しない.
insert	S は R にデータの登録を要求する.
recommend-all	S はある performative に対して応答できる全てのエージェントの名前を要求する.
recruit-one	S はある performative に対して応答できるエージェントの名前を要求する.
sorry	S は R のメッセージを理解したが，回答ができない.
subscribe	S は performative に対する R レスポンスについて，将来変更があればそれを逐次通知してくれるように要求する.
tell	S の知識内で真であることを知らせる.
untell	S の知識内で偽であることを知らせる.

ここで，S は命令の送信者であり，R は命令の受信者である

4.2 知識表現言語

エージェント間通信言語では，エージェント間のメッセージを規定するのみで，メッセージの内容で表現する知識までは規定していない．そこで，エージェント間通信言語とは別にエージェントが持っている知識を表現する必要がある．KQML では，KQML メッセージ内の「language :」のところで，使用する知識表現言語を指定することになっている．SQL，lisp，prolog 等の任意の言語を使用することが可能である．KQML を開発したプロジェクトでは，知識表現言語として KIF（Knowledge Interchange Format)[*10] を開発した．また，エージェント標準化団体である FIPA では，新たな知識表現言語として SL（Semantic Language）を規定した.

4.3 オントロジ

KQML ではメッセージ交換の前提としてオントロジ（ontology)，すなわちエージ

ェント間のコミュニケーションに使用する語彙の定義範囲を指定する．オントロジは，ある分野知識（語彙・概念・動作など）を定義する語彙体系である．

エージェント同士がメッセージを交換する場合には，メッセージを送信するエージェントは，これからどのような分野について話し始めるかを，オントロジを指定することによって相手に通知する．メッセージを受け取った側は，指定されたオントロジに従ってメッセージを解釈し，処理を実行する．この意味でオントロジ名は，サービス内容を規定するものと捉えることができる．

4.1節の（＊）では，オントロジに“Genealogy”が指定されている．このオントロジでは，人には親子関係があり，(“father”)とは父親を求めていることや，“Taro”，“Jiro”は人の名前のインスタンスであることが定義されている．

代表的なオントロジとして，Stanford大学のOntolingua[*11]がある．

4.4 ファシリテータを利用したエージェント間通信

エージェント間で直接通信するためには，相手のエージェントを特定しなければならない．しかし，エージェントはどのエージェントがどのサービスを提供しているのかをすべて知っているわけではない．そこで，通常のエージェントシステムはファシリテータ（協調促進器）が導入される．ファシリテータは，エージェントのメッセージを転送したり，サービスを提供しているエージェントを紹介したりして，エージェントの間の協調を促進する．ファシリテータが存在したときのエージェント間の対話の例を以下に示す．

1） 直接通信（Point-to-Point型）（図4）

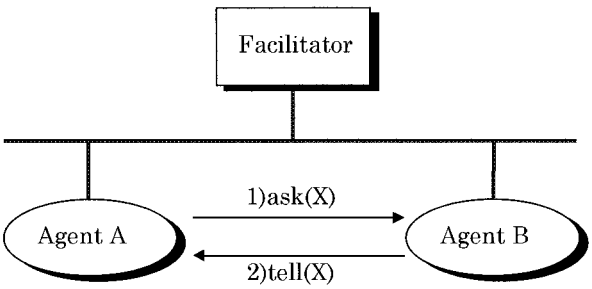

図 4 Point-to-Point型

お互いの存在を知っているエージェント間で直接通信を行うパターンである．

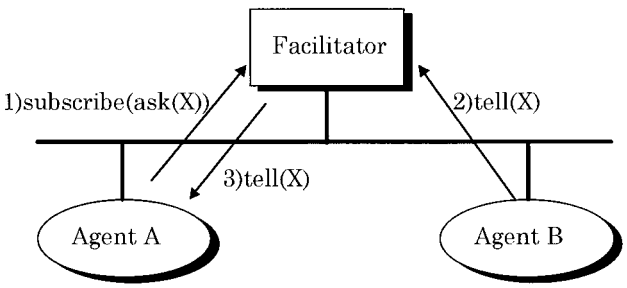

図 5 Subscribe型

2)　情報の仲介（Subscribe 型）（図 5）

ファシリテータがエージェント B からの情報の仲介役をする．多くのエージェントが同一情報を使用したいときに利用されるパターンである．

3)　メッセージの仲介（Broker 型）（図 6）

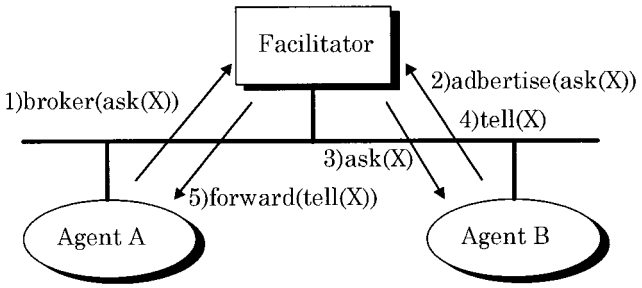

図 6　Broker 型

ファシリテータがエージェント A からの要求をエージェント B へ転送し，エージェント B からの回答をエージェント A へ返す．ファシリテータがブローカの役目を果たす．

4)　メッセージの転送（Recruit 型）（図 7）

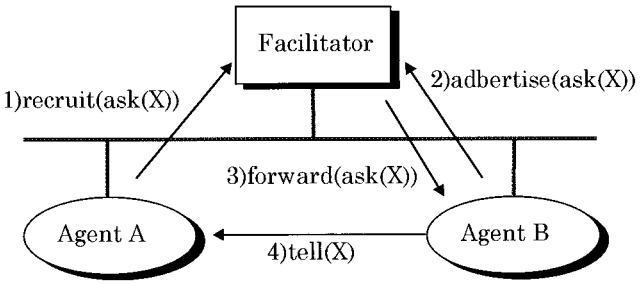

図 7　Recruit 型

ファシリテータはエージェント A から要求を受け取ると，エージェント B へエージェントの A へ回答を返すように要求を依頼する．

5)　エージェントの紹介（Recommend 型）（図 8）

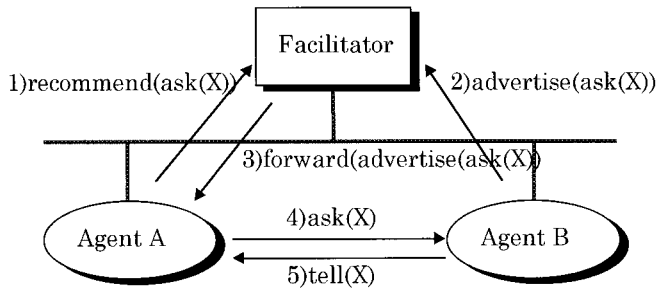

図 8　Recommend 型

ファシリテータがエージェント B の存在をエージェント A へ知らせる．

5. 標準化動向

5.1 FIPA

FIPA は，エージェント機能の業界標準化団体確立を目指して，世界各国の研究機関や通信・放送事業者，メーカーが集まって作った非営利の任意団体である．議長は，「MPEG」の標準化を導いたイタリアの通信分野の研究機関 CSELT（Centro Studi e Laboratori Telecomunicazioni）の Leonardo Chiariglione 氏である．

現在，第 1 次の標準である，FIPA 97 Specification Version 1.0（以下 FIPA 97 と呼ぶ）が規定されている[*12]．

5.1.1 FIPA の標準化内容

FIPA は，「エージェントとは，①他のエージェント，②外部ソフトウェア資源，③外部ハードウェア資源，④人間，と相互作用することによって，内部処理を実行し，目的の処理を実行するものである」としている．FIPA での標準化作業は，エージェントが外部との相互作用に基づいて活動できる環境を構築するとともに，エージェントの共用制を確保する技術の確立である．よって，エージェント自体の実装法については，標準化の対象外である．FIPA 97 で決められた内容は，Agent Management, Agent Communication, Agent Software Interaction, Application Design Test（Personal Travel Assistance, Personal Assistant, Audio/Video Entertainment and Broadcasting）である．

ここでは，FIPA 97 で策定された規約の中で，エージェントの管理に関する「エージェント管理」（Agent Management）と，エージェント間通信言語である「FIPA ACL」（FIPA Agent Communication Language）の概要について述べる．

5.1.2 FIPA エージェント参照モデル

FIPA 97 で規定されたエージェント管理のためのエージェントシステム参照モデルは，図 9 のようになる．

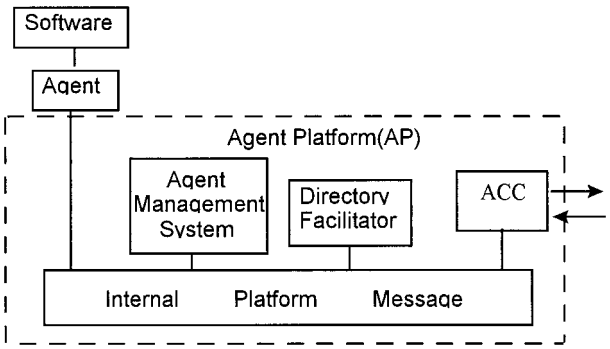

図 9 FIPA エージェントモデル

図 9 に記述された主要な用語を以下に解説する．

- Agent Platform（AP）

Agent が存在することのできる基盤を提供する．Agent は他の Agent と相互作用するために，AP に登録されなければならない．

- Agent

  Agent とは，AP 上で活動する最も重要な行為者であり，様々なサービスを提供する．外部のソフトウェア，人間へのアクセス手段や，他 AP への通信能力を持つ事ができる．Agent には，アイデンティティの概念が導入され，全世界で唯一に識別されるように名前付けされなければならない．このように全世界で一意につけられた名前を GUID（Globally Unique Identifier）と呼ぶ．

- Directory Facilitator（DF）

  AP 上において，必須のコンポーネントであり，Agent として実現される．Agent の Yello Page としての機能を提供する．

  Agent は，自身のサービスを DF に登録することもできるし，DF に問い合わせて，他の Agent によって登録されたサービスを見つけ出すこともできる．

  少なくとも一つの DF が AP に存在しなければならない．

- Agent Management System（AMS）

  AP 上において，必須のコンポーネントであり，Agent として実現される．一つの AP にただ一つの AMS が存在しなければならない．AMS は Agent のライフサイクル全般を管理（例えば，動的に Agent を AP へ登録，Agent の移動監視など）する．

- Agent Communication Channel（ACC）

  Agent（DF や AMS を含む）間の通信路である．ACC は AP 内の Agent 間メッセージ，AP をまたがった Agent 間メッセージの経路制御を行う．

  AP が FIPA 準拠であるための条件として，IIOP[*13] のサポートが義務付けられている．この仕様は，AP 間の相互運用性を確保するために最低限の規定である．

- Software

  Domain（後述）からアクセス可能な実行可能な命令の集合であり，Agent でないものと定義されている．

  例えば，Agent は以下のような理由で，Software にアクセスする．

  ① 新しいサービスを得る．
  ② 新しい通信プロトコルを得る．
  ③ 新しいセキュリティプロトコルやアルゴリズムを得る．
  ④ 新しい交渉プロトコルを得る．
  ⑤ 移動をサポートするツールを利用する．

5.1.3 Domain

Domain とは，エージェントやサービスを論理的にグループ分けしたものである．各々の Domain は DF を一つだけ持ち，その DF は Domain 内の情報を保持する．各エージェントは，複数の Domain に属することができる．

図 10 は二つの AP 上に三つの Domain を構築したときの図である．

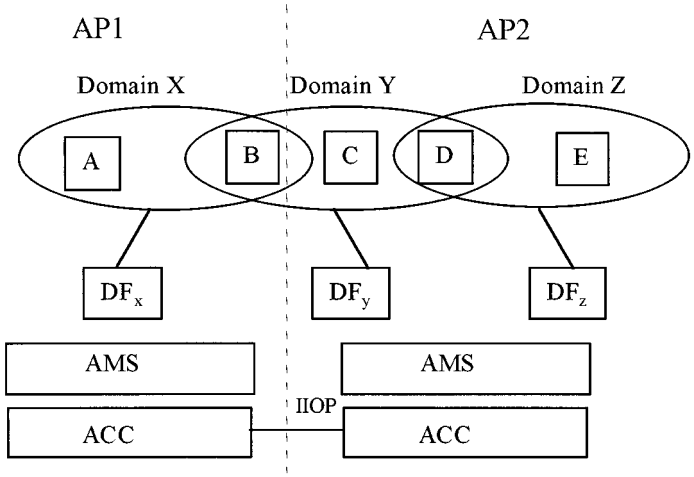

図 10　Agent Platform 概念図

5.1.4　エージェントのライフサイクル

FIPA では，Agent のライフサイクルを Agent Platform Life-Cycle と Domain Life-Cycle という，二つの観点から定義している．

1）　Agent Platform Life-Cycle

Agent Platform Life-Cycle とは，Agent を物理的なソフトウェアプロセスとして捕らえたライフサイクルモデルであり，図 11 のように表される．

Agent Platform Life-Cycle は，以下の特徴をもつ．

・AP bounded：Agent とは AP 内で物理的に管理されるので，Agent のライフサイクルは特定の AP 内に制限される．

・Application independent：このライフサイクルモデルは，いかなるアプリケーションシステムからも独立しており，エージェントの状態（State）と遷移（Transition）のみを定義する．

・Instance oriented：このライフサイクルモデルで述べられる Agent は，インスタンス（一意の名前を持ち，独立して実行する Agent）として扱う．

・Uniqueness：Agent は，ただ一つの AP 内でただ一つのライフサイクル状態を持つ．

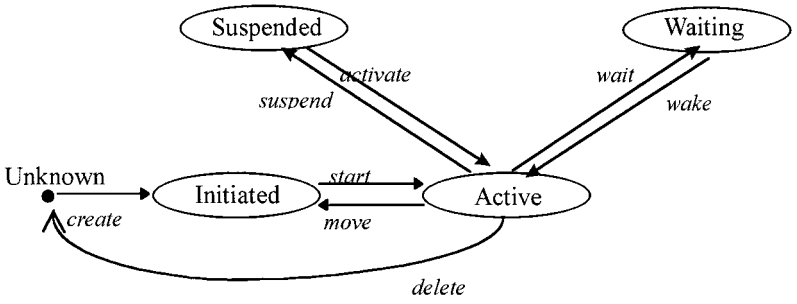

図 11　AP Life-Cycle

表 3　AP Life-Cycle の状態

状態	意味
Initiated	Agent が生成された，または他の AP から Agent が移動してきた状態．AP は，Agent を起動（もしくは再起動）させる前に，エージェントに与えられたパラメータや環境を初期化することができる．
Active	現在 Agent が AP 上で動作している状態
Suspended	Agent が停止している状態．Agent は AP や AMS によって，もしくは Agent 自身の要請によって停止する．この状態の時に，Agent に対してメッセージが送られた場合，AMS は，メッセージを送った Agent に送信が失敗した旨のレポートを返す．
Waiting	Agent があるイベントを待って休止している状態．この状態の時に，Agent に対してメッセージが送られた場合，メッセージは配送されるが Agent は即座に応答するとは限らない．

表 4　AP Life-Cycle の状態遷移

状態遷移	意味
Create	新しい Agent を生成（インスタンス化）させる．
Start	Agent を起動（または再起動）させる．
Suspend	AP や Agent 自身の依頼によって Agent を停止する．
Activate	停止している Agent を活性化させる．
Wait	Agent にあるイベントを待たせる．Suspend と違い，*wait* は AP によって起動されることはない．
Wake	Waiting 状態から Agent を目覚めさせる．AP によってのみ起動される
Delete	Agent を終了し，AP から Agent を消去する

AP Life-Cycle の状態遷移の意味は表 3，表 4 に示される．

2）Domain Life-Cycle

Domain Life-Cycle とは DF から見えるエージェントのライフサイクルモデルであり，図 12 のように表される．

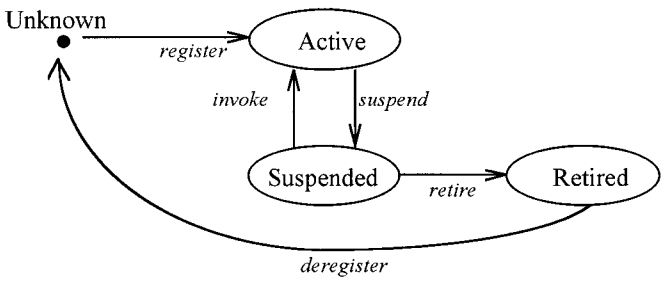

図 12　Domain Life-Cycle

Domain Life-Cycle は，以下の特徴をもつ．

・Domain centric：Agent は，その Agent が登録された Domain 内の DF によって認識され管理される．このライフサイクルモデルは，Domain 内の Agent の動作に焦点を当てる．Agent は，異なる Domain の中では異なった状態を持つことができる．

・Application Independent：このライフサイクルモデルはどんなアプリケーションシステムからも独立しており，エージェントの状態と遷移のみを定義する．

・Instance oriented：AP ライフサイクルと同様である．

AP Life-Cycle の状態遷移の意味は，表 5，表 6 に示す．

表 5 Domain Life-Cycle の状態

状態	意味
Suspend	Agent が DF に登録されたが，まだ，起動準備段階の状態．
Active	Agent が起動され，利用可能な状態．
Retired	Agent は登録を削除された，もしくは「退去した」と印がつけられ，もはや Domain 内で利用できない状態．

表 6 Domain Life-Cycle の状態遷移

状態遷移	意味
Register	Agent が DF に自分の名前やサービスを登録する．
Invoke	Agent が利用可能になったことを DF に知らせる．
Suspend	Agent が一時的に利用不可になったことを DF に知らせる．
Retire	Agent が永久に利用不可になったことを DF に知らせる．
Deregister	Agent が DF からエントリを削除するように，DF へ依頼する．

5.1.5 FIPA ACL

エージェント間の通信言語は，前述した KQML が一般的であったが，KQML より意味が明確なエージェント間言語が，FIPA により FIPA ACL として，新たに規定

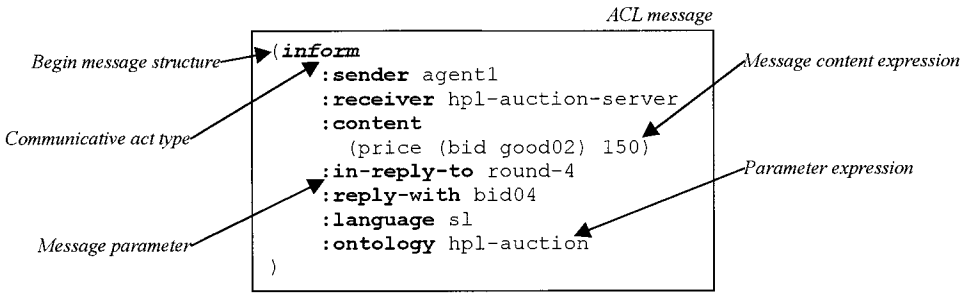

図 13 FIPA ACL メッセージ内容

された．FIPA ACL は S 式で表現され，図 13 のような構造となる．

メッセージの最初の部分は，Communicative Act が指定される．Communicative Act とは，メッセージの主要となる意味を定義する．その後にコロンで続くメッセージパラメータが続く．Communicative act は KQML の命令に相当し，その後に続くパラメタも KQML の引数と似ている．FIPA の仕様には，KQML から FIPA ACL への移行を容易にするために，既に KQML に精通している人への手引きを含んでいる．

5.1.6 FIPA の今後

FIPA は FIPA 98 のための Call for Proposals を発行し，提案を募集している．この Call for Proposal では，FIPA 97 から導き出された必要技術と，エージェントが有効に使用されると思われるアプリケーションについて述べられている．

必要技術としては，

- Human/Agent Interaction
- Agent Mobility
- Agent Security

などがあげられており，ターゲットアプリケーションとして，

- Manufacturing
- Electronic Commerce

などがあげられている．

5.2 OMG/MASIF[*14]

OMG では，CORBA 上のエージェントシステムをまず規定し，そのエージェントシステム間で，エージェントを移動するために必要なインタフェースを「Mobile Agent System Interoperability Facilities」（MASIF）として標準化した．この機能は，はじめは Mobile Agent Facility（MAF）と呼ばれ，エージェントシステム内部まで CORBA IDL で規定しようとしていた．しかし，その後方向が変わり，エージェントシステム間の相互運用性に重点がおかれ，名前も MAF から MASIF に変更された．GMD FOKUS，IBM が中心となり，Joint Submission[*15] を提出し，投票により，1998 年 3 月にこの Submission が承認された．

5.2.1 MASIF の概要

MASIF は，CORBA 上に構築されたエージェントやエージェントシステムの論理構成を規定し，エージェントが移動するために必要なエージェントシステム間の相互運用性を CORBA IDL として規定したものである．

MASIF が定義するエージェントシステムの論理構成は，図 14 のような論理構成をとる．

図 14 に示されている主要な用語は，MASIF では以下のように記述されている．

- Agent

  Agent とは，個人や組織の代わりとなって働く自律的なコンピュータプログラムである．Agent は，Stationary Agent と Mobile Agent に分けられる．Stationary Agent とは，起動された Agent System 上でのみ実行される Agent である．Agent が起動された Agent System 以外の場所の情報を得る必要があ

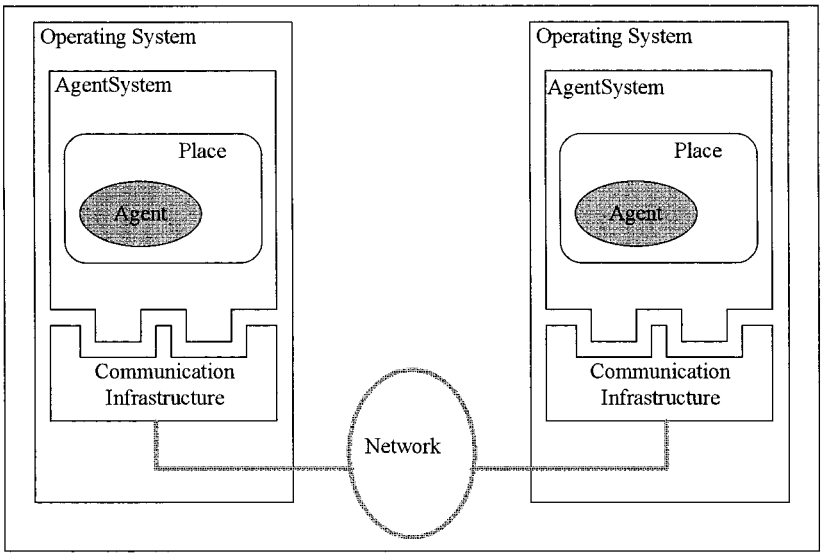

図 14　MAF エージェントシステム論理構成

る時や，他の場所の Agent と相互作用する必要が生じた場合は，一般的には RPC のような通信手段を使用する．Mobile Agent とは，起動された Agent System に拘束されないエージェントである．ネットワーク経由で，自分自身を他の Agent System 上へ移動させる．

- Agent System
  Agent の生成，解釈，実行，移動，終了を行うプラットフォームである．Agent System は，名前とアドレスによって一意に識別される．
- Place
  Agent が自分自身を移動させるとき，Place と呼ばれる実行環境の間を移動する．Place は Agent System 内に存在する Agent の実行エンジンである．Agent は，同一 Agent System 内の Place 間を移動することもできるし，異なる Agent System にある Place 間を移動することもできる．

5.2.2　MASIF のインタフェース

MASIF は，上記のエージェントモデルを，CORBA の IDL として規定している．この IDL は「MAFAgentSystem」インタフェースと「MAFFinder」インタフェースからなる．

MAFAgentSystem インタフェースは Agent の生成，移動，停止，終了など，Agent に関連したオペレーションを定義している．一方，MAFFinder インタフェースは，「Agent，Place，Agent System」の登録，登録削除，検索のために使用されるオペレーションを定義している（図 15）．

5.2.3　MASIF の現状

OMG の正式な文書とするために，OMG 内の RTF[*16] 内で Joint Submission の更新作業が行われ，その結果が公開されている[*17]．

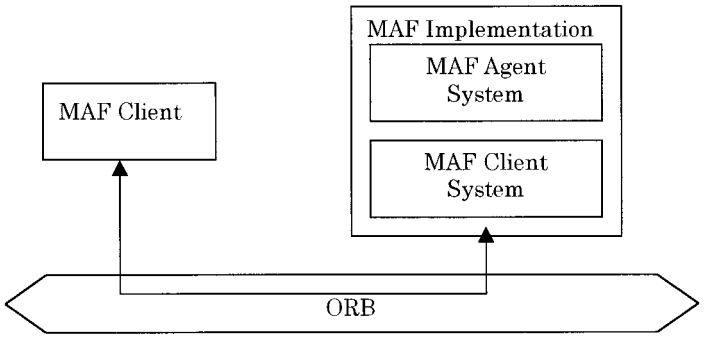

図 15　MAF（MASIF）と ORB の関係

6.　次世代電子図書館研究開発事業[*18]

ここでは，弊社のエージェントに関係したシステム案件の中の一つである，「次世代電子図書館システム研究開発事業」について概略を説明する．

次世代電子図書館システム研究開発事業とは，次世代情報化社会での情報流通の公共的基盤として機能する電子図書館システムの実現を目指して，その標準的なアーキテクチャや先進的な個別技術の研究開発を行うプロジェクトである．このプロジェクトは国からの委託を受けた情報処理振興事業協会（IPA）より，JIPDEC が再委託され実施するものである．

6.1　次世代電子図書館システムのアーキテクチャ

電子図書館のアーキテクチャ構造を図 16 に示す[*19]．エージェント基盤は，基本アーキテクチャを構成するサブアーキテクチャの一つであり，使い易く，柔軟で拡張性の富むシステムを構成するのに必須な要素として位置づけられる．

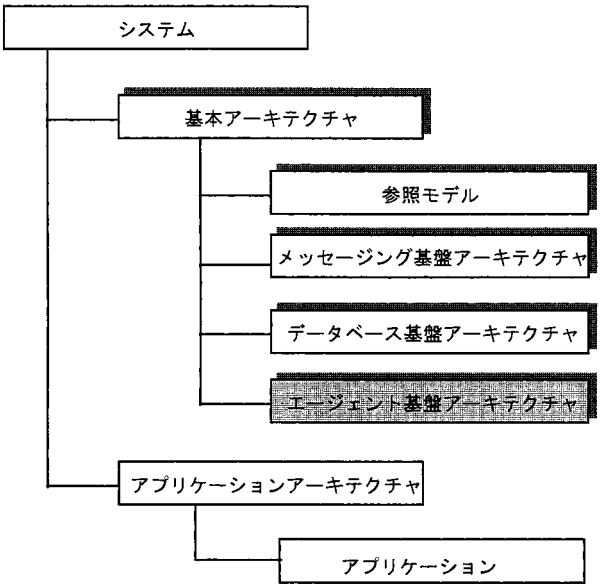

図 16　次世代電子図書館システムアーキテクチャ

また，電子図書館システムは，各サブアーキテクチャをもとにして，

- ユーザインタフェース機能を提供するオブジェクト群からなる表示層
- 業務ロジックを実現するオブジェクト群からなる機能層
- データベース機能を提供するオブジェクト群からなるデータ層

の 3 層構造からなる分散オブジェクト指向環境で構築される．

弊社は，エージェント基盤アーキテクチャを日立製作所殿と共同で開発しており，主に，エージェント基盤の中のサービスである，「モービルエージェント機能」の実装規約作成を担当している．

6.2 エージェント基盤の目的

エージェント基盤の目的は，エージェントの特長を生かして，

- 分散ネットワークに適したアプリケーション環境
- ソフトウェアの部品化・再利用を促進するための環境
- システム運用・システム拡張の負荷を削減するための環境
- ユーザに応じた処理の動的変更

を実現することである．

6.3 モービルエージェント

6.3.1 移動の定義

実行状態のプログラムを移動する場合には，二つの考え方がある．一つ目の考え方は，プログラムコード，データおよび実行時のスタック情報を丸ごと移動させる方法である．このタイプは，GeneralMagic 社の移動エージェントである，Telescript が実現している．二つ目の考え方は，プログラムコードおよびデータを移動し，スタック情報は移動させない，という方法である．この両者の優劣を一概に決めることはできないが，簡単にまとめると，前者は，後者に比べて実現できる機能性に優れるが，特殊な言語体系の開発が必要となる等の技術的な困難さを伴う．本研究のエージェント基盤では，エージェントをコーディングする言語として，既存のプログラミング言語（C＋＋または Java）を仮定している．C＋＋または Java 言語において，プログラムのスタック情報までの移動を実現するには，実装上かなりの労力がかかる．一方，利用者サイドである図書館アプリケーションを含む一般的なアプリケーションから見ると，あまり高度な『移動性』は必要ないといえる．つまり『他のノードに移動して処理を継続できる』機能があれば十分であろう．

したがって，本研究においての『移動性』とは，

- プログラムが処理ロジックとユーザデータを保持し，ネットワークを通して他の計算機上に移動すること

と定義した．すなわち二つ目の考え方をとることとした．このことにより，モービルエージェント機能を使用するプログラマは，以下の二つを意識しなければならない．

- 移動先の処理に必要となるデータを，プログラマの責任において不揮発性リソース（永続メモリ）に保存すること，および取り出すこと．
- プログラムのロジックで，エージェントの状態（エージェントのライフサイクル，エージェントが属するプレースやノード）を判断させ，その状態に応じた処理を記述すること．

6.3.2 モービルエージェント機能の定義

前節のことを踏まえ，モービルエージェント機能とは，

- エージェント基盤上で生成されるエージェントに，前節で定義した移動の能力を付加するサービス

と定義した．

6.3.3 モービルエージェント機能の利点

モービルエージェント機能は，特に，システム運用，システム拡張の負荷軽減に対して，重要な役割を果たす．

エージェントは，その自律性と移動性を生かすことにより，使用者に対して

- 定型作業の自動化
- システム構成や機能が頻繁に変わるような環境でのシステム構築
- 複数プログラムが機能を補完し合い，協調して処理を行うシステム
- インターネットに代表される異種分散広域ネットワークでのアプリケーション
- 個人に適応した動作をするアプリケーション構築

等の実現を支援する．

6.4 エージェント基盤論理構成

次世代電子図書館システム上のエージェント基盤は図 17 に示すような論理構成となる[*20]．

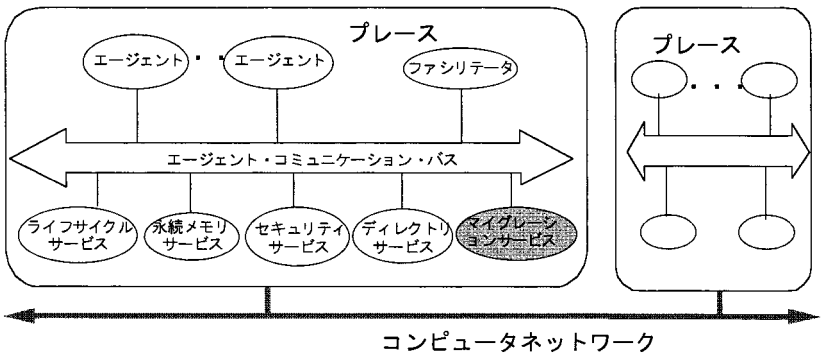

図 17　エージェント基盤論理構成

プレースとは，エージェントを実行するためのエンジン，スケジューリング機能，エージェント間通信機能，基本サービスを提供している論理的なエージェントの活動の場である．エージェントはプレース内に生成され，原則としてプレース内のサービスを使用し，他のエージェントと通信し，目的の処理を行う．

ここで，モービルエージェント機能は，「マイグレーションサービス」として実現される．マイグレーションサービスはエージェントを，エージェントの活動の場とする論理的なプレース間，及び計算機と一対一に対応づけられるノード間の移動を実現する．

6.5 移動モデル

マイグレーションサービスがエージェントを移動させる方法として，以下の二つの移動方法を定義した．

1) 同期型移動モデル

移動元のマイグレーションサービスと移動先のマイグレーションサービスが，同期を取り合って，エージェントの移動を実現する方法である（図 18）．

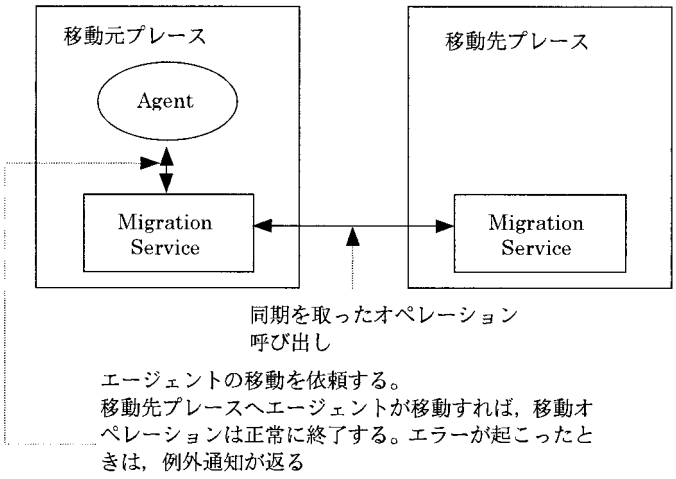

図 18 同期型移動モデル

同期型移動モデルでエージェントが移動する場合，エージェントはマイグレーションサービスへの移動依頼後に，移動が何らかの原因で失敗した時に例外通知を受け取ることができる．例外通知を受け取ることにより，エージェントはその後の処理を継続することができる．

2) 非同期型移動モデル

移動元のマイグレーションサービスと移動先のマイグレーションサービスが，同期を取り合うこと無く，エージェントの移動を実現する方法である（図 19）．

非同期型移動モデルでエージェントが移動する場合，エージェントはマイグレーションサービスへ移動を依頼した後に終了する．移動先で何らかの不具合により，エージェント生成に失敗した時のエラー処理をエージェントは行うことができないことを考慮する必要がある．

6.6 モデル実装のプロトコル

移動要求によって呼び出されるオペレーション及び転送対象データは，基本的に CORBA/IIOP によって移動先に運ばれる．CORBA/IIOP の同期/非同期通信の IDL 定義をすることにより，前述の両モデルを実現することができる．但し，CORBA/IIOP では，現在ファイアウォールを超えることができない．従って，ファイアウォールを超えた分散システム間で移動を行うプロトコルとして，SMTP を採用した．SMTP を使用することにより，インターネット上のほとんどのエージェントシステム間で，エージェントの移動が実現できる．

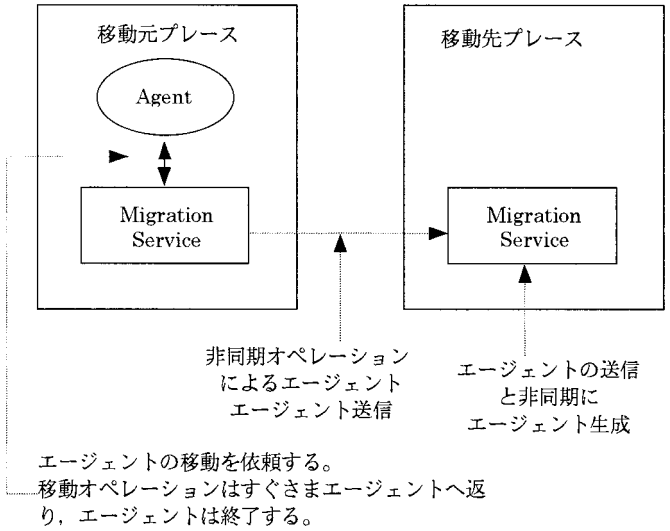

図 19　非同期型移動モデル

6.7　モービルエージェントの実装

エージェントは,エージェントプログラマが作成したエージェントプログラム部と,エージェントプログラマに対して各サービス（マイグレーションサービスを含む）のインタフェースを提供するエージェントライブラリ部からなる（図 20）.

プログラマがモービルエージェントを実装する場合，マイグレーションサービスライブラリを使用する方法と，CORBA オブジェクトであるマイグレーションサービスオブジェクトを使用する方法がある．しかし，通常プログラマは，ライブラリを使用してエージェントを作成することとした．なぜなら，ライブラリを使用してエージェントを作成すれば，将来基盤のオブジェクト環境が CORBA 以外に変わっても，エージェントプログラムを変更する必要がないからである．

6.7.1　モービルエージェントライブラリ

マイグレーションサービスライブラリは,

- マイグレーションサービス
- マイグレーションプラン

からなる.

マイグレーションサービスは，エージェントの移動要求を受けつけ，移動要求を解釈し，モービルエージェント基盤オブジェクト（CORBA オブジェクト）へ移動要求を伝達する.

マイグレーションプランは，エージェントの移動計画を保持するオブジェクトであり，エージェントの移動先をリスト構造として保持している．マイグレーションプランを使用することで，エージェントは移動の定型化及び管理を行うことができる（図 21）.

6.7.2　マイグレーションサービスオブジェクト

マイグレーションサービスオブジェクトは，通常 CORBA オブジェクトとして実

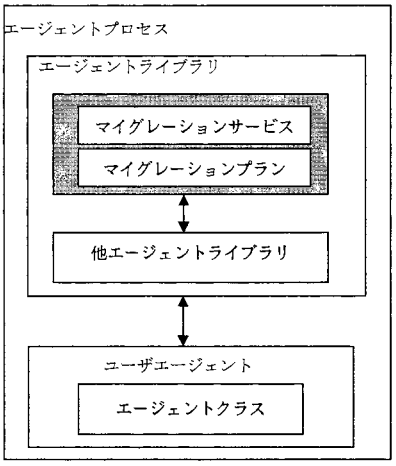

図 20　マイグレーションサービスライブラリ

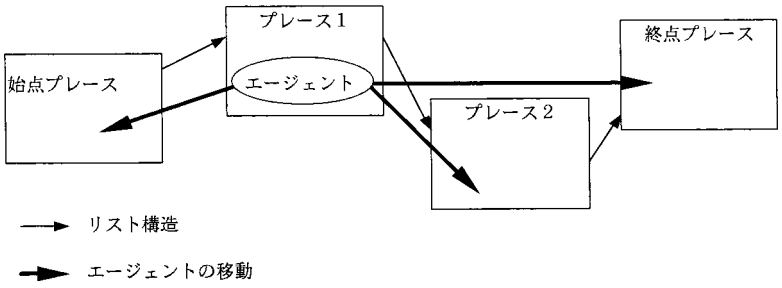

図 21　マイグレーションプラン

装されるオブジェクト群である．

エージェントの移動は，マイグレーションサービスオブジェクト間のメッセージ通信として実現される．

マイグレーションサービスオブジェクトの主な機能は，図 22 で示すように，以下の三つの機能から成り立つ．

- エージェントのクラス転送
- 永続メモリの転送
- エージェントクラスのローディング

なお，現在 OMG で標準化作業が進行中である，MASIF の機能もエージェントシステム間のゲートウェイ機能として，Joint Submission を元に盛り込んである．

6.8 今後の課題

今後の課題として以下のことがあげられる．

1）　セキュリティ

モービルエージェント機能とは，エージェントをエージェントが生成されたコンピュータ以外のコンピュータ内で実行させることである．よって，モービルエ

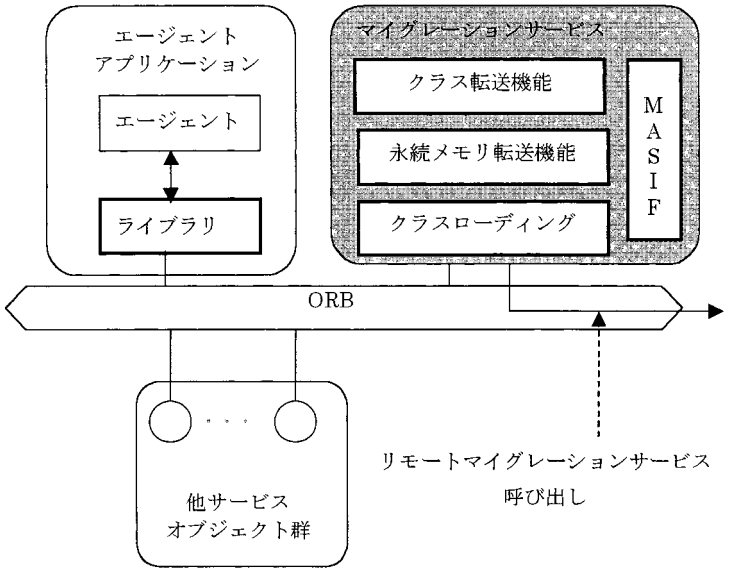

図 22　マイグレーションサービスオブジェクト

ージェントとコンピュータウィルスは，区別がつかなくなる可能性がある．今後は，エージェント基盤内にセキュリティサービスを構築し，不正なエージェントからエージェント基盤を守るための規約を制定する．

2)　運用管理

エージェント基盤内の統一的な運用管理を規定する必要がある．また，エージェントの移動により，エージェントの管理，追跡機能も規定する必要がある．

3)　標準化対応

モバイル端末からエージェントの移動が実現されると，エージェントの活用範囲が広がる．現在のエージェント基盤を軽量化し，モバイル端末のような小型の端末でもエージェントが利用できるような仕組みを検討していく予定である．また，既存の図書館サーバへの移動に関する実現可能性も検討項目としたい．

4)　エージェント活動環境の拡大

モバイル端末からエージェントの移動が実現されると，エージェントの活用範囲が広がる．現在のエージェント基盤を軽量化し，モバイル端末のような小型の端末でもエージェントが利用できるような仕組みを検討していく予定である．また，既存の図書館サーバへの移動に関する実現可能性も検討項目としたいと考えている．

7. おわりに

本稿ではエージェント技術の概論，エージェント間通信言語，エージェントの標準化動向，弊社におけるエージェントの取り組みについて紹介した．

エージェントは，まだ一般的ではなく課題も多い．たとえば，オブジェクトとしてどのようにエージェントシステムを実現するのか，エージェントシステムのセキュリ

ティ，信頼性をどのように確保するか，現実の情報システムへエージェントをどのように適用するかなどがあげられる．

今後は，FIPA や OMG での標準化動向に着目しつつ，前述の課題を克服し，エージェントシステムの構築を図っていきたい．

＊ 1　西田豊明 “ ネットワーク社会とエージェント ” 情報処理 38 巻 1 号 1997 年 1 月
＊ 2　http : //www.quarterdeck.com/
＊ 3　http : //www.genmagic.com/
＊ 4　http : //www.trl.ibm.com/aglets/index-j.html
＊ 5　http : //www.javasoft.com/
＊ 6　Shoham, Y.“ AGENT 0 A simple agent language and its interpreter, Proc. AAAI-91
＊ 7　http : //java.stanford.edu/
＊ 8　http : //www.cs.umbc.edu/kqml/
＊ 9　発話行為理論とは，メッセージとは行為，または通信行為であり，メッセージの送信によってある行為の発生が意図される，という考え方．
＊10　http : //www.cs.umbc.edu/kse/kif/
＊11　http : //ksl-web.stanford.edu/knowledge-sharing/ontolingua/ontolingua.html
＊12　http : //www.cselt.stet.it/fipa/spec/fipa 97/fipa 97.htm
＊13　Internet Inter-ORB Protocol : OMG で定められたオブジェクト間の通信プロトコル
＊14　Object Management Group “ Common Facilities RFP 3 ” 1995
http : //www.omg.org/library/schedule/Mobile Agents Facility RFP.htm
＊15　GMD FOKUS/IBM/Crystaliz, Inc./General Magic, Inc/The Open Group ” Joint Submission Mobile Agent System Interoperability Facilities ”, 1997
http : //www.omg.org/docs/orbos/97-10-05.pdf
＊16　Revision Task Force の略．採決された案を OMG の公式な文書にするために，更新作業を担当するクローズなタスクフォースのこと．
＊17　http : //www.omg/org/docs/orbos/98-03-09.pdf
＊18　http : //www.dlib.jipdec.or.jp/
＊19　財団法人日本情報処理開発協会次世代電子図書館システム研究開発事業論文集
＊20　財団法人日本情報処理開発協会次世代電子図書館システム研究開発事業論文集

参考文献

［1］ H. Dohi, M. Ishizuka,“ Visual Software Agent : A Realistic Face-to-Face Style Interface connected with WWW/Netscape,
http : //www.miv.t.u-tokyo.ac.jp/dohi/ijcai 97-ims/paper.html
［2］ N. Tosa“ Human-like Communication Character ”http : //www.mic.atr.co.jp/～tosa/
［3］ Y.Shoham,“ Agent-oriented Programming, Artificial Intelligence, Vol 60 ”
［4］ Y.Shoham,“ Agent−0 A simple agent language and its interpreter, Proc. AAAI-91
［5］ Robert H.Guttman, Alexandros G, Moukas, Pattie Maes ”Agent-mediated electronic Commerce : A Survey ”, http : //ecommerce.media.mit.edu/papers
［6］ FIPA,“ FIPA 97 Specification Part 1 Agent Management ” 1997.
［7］ FIPA,“ FIPA 97 Specification Part 2 Agent Communication Language ” 1997.
［8］ OMG“ Mobile Agent System Interoperability Facilities ”,
GMD FOKUS/IBM/Crystaliz, Inc/General Magic, Inc/The Open Group
OMG Documentat : orbos/97-10-05.
［9］ 竹林洋一 “ 音声自由対話システム TOSBURG II－ユーザ中心のマルチ・モーダルインタフェースの実現に向けて ”, 電子情報通信学会論文誌 Vol. J 77-D-II, No. 8.
［10］ 木下哲男, 菅原研次,“ エージェント指向コンピューティング ”, ソフト・リサーチ・センター.
［11］ 山崎重一郎, 津田博,“ Telescript 言語入門 ”, アスキー出版局.
［12］ S. Russell, P. Norvig, 古川康一（監訳）,“ エージェントアプローチ人工知能 ”, 共立出版社.
［13］ “ コンピュータソフトウェア Vol. 14 No. 4 July 1997 ”, 日本ソフトウェア科学会.
［14］ “ 情報処理 38 巻 1 号 1997 年 1 月 ”, 情報処理学会.
［15］ “ 次世代電子図書館システム研究開発事業論文集 ”, 財団法人日本情報処理開発協会.
［16］ “ 凝人化エージェントシステム（VSA）”, http : //www.miv.t.u-tokyo.ac.jp/vsa.html.

執筆者紹介　戸　叶　　徹（Toru Tokano）

1990 年東京理科大学理学部数学科卒業．同年日本ユニシス(株)入社．UNIX 系の通信関連プロダクトの開発/保守．NTT マルチメディア通信共同実験に従事．現在，新事業企画開発部市場開発室に所属．情報処理学会会員．