3F－5 複数の実装言語によるエージェントの記述のための アーキテクチャ

河辺岳人†、児玉英一郎‡、宮崎正俊‡

†株式会社ＳＲＡ東北　‡岩手県立大学ソフトウェア情報学部

1．はじめに

近年の計算機ネットワークの発達、およびその普及が進むにつれ、これを有効に利用する分散システムが求められている。このようなシステムを構築するための要素として、ソフトウェアエージェントが提案されている。

エージェントを利用してシステムを構築するにあたっては、これを実際に動かすためのプラットフォームとして、実装言語とその処理系、複数のエージェント間の協調実現の機構、協調プロトコルの記述・検証・再利用、機能に即した記述方式の利用など、様々な機能・性質が必要となってくる。

これらの要求を個別に解決するためのフレームワークや実行環境は従来から提案されていたが、依然、実際の応用開発者の要求としては

・エージェントに必要な要素を分離しておきたい
・それぞれを適切な言語で記述したい
・交渉プロトコルを再利用したい

といったものがある。

本研究においては、特に従来は単一の言語でしか記述できなかった問題点に着目し、応用開発者にとって開発のしやすいエージェントを記述・実装するためのアーキテクチャを提案し、実際にシステムを製作してその評価を行った。

2．エージェントの３層モデル

本研究では、単一のエージェントの内部構造を図１のように３層に分割する。

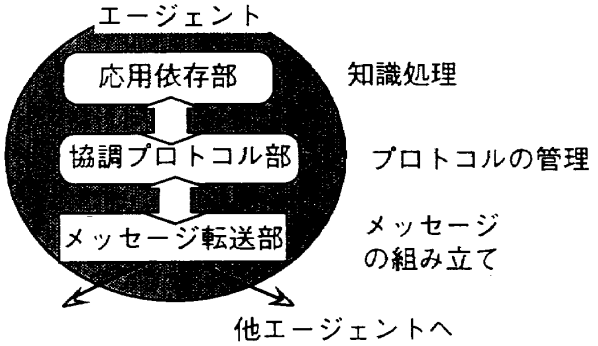

図 1.エージェントの３層モデル

エージェントの実現においては、内部の各機能のパラダイムの相違から、複数の言語を用いて記述したい場合がある。この３層は、こういった異なった言語を利用して記述したくなる境界に合わせて分割してある。

開発にあたっては、それぞれの部分を記述・実装する必要があるが、層を違えることにより、記述の分担や再利用が可能になる。以下にそれぞれの部分の機能について述べる。

2.1 各層の機能

2.1.1 応用依存部

応用領域に特化した部分の記述・処理を担当する。応用開発者によって記述される部分。通信に関する処理はいっさい行わず、下のプロトコル部によってすすめられる協調プロトコルの進行に応じて動作する。

ここの部分は、知識処理の得意な処理系(Prologなど)によって記述されることが多くなると思われる。

2.1.2 協調プロトコル部

エージェント間の協調プロトコル[HLP]を司る部分。プロトコル開発者によって記述される。完成したプロトコルは、プラットフォームを通じてプロトコル部品として応用依存部に提供される。

従来の分散協調エージェントの提案や実現例においては、交渉のためのプロトコルと、プロトコルの進め方を決定する知識・判断部が明確に分離されていないが、本アーキテクチャにおいてはここで明確に分離して記述することが可能であり、また要求されている。

協調プロトコル部の記述言語としては、プロトコルの記述や検証の得意な言語(LOTOSなど)を用いることが考えられる。

2.1.3 メッセージ転送部

分散環境上でのエージェント間のメッセージ交換を行う部分。メッセージの組み立てや送受信、名前解決、これに附随する自身の存在の管理も行う。

メッセージの送信要求がプロトコル部から降りてくると、これをネットワーク上に流せる形式に変換して宛先エージェントへ転送する。また、自分宛メッセージが到着すれば、対応するプロトコル部に対してこれを渡す。

メッセージ転送部自体の記述は、低レベルの通信メッセージの組み立てが可能な言語を使用することになる。メッセージ自体の形式としては、[KQML]などを利用することが考えられる。

2.2 プロトコル指向型インタフェース

エージェント間の通信機能としては、直接メッセージをやりとりするのではなく、協調プロトコルを

An Architecture for Multilanguage Implementation of an Agent
Taketo Kabe, Eiichiro Kodama, Masatoshi Miyazaki
Faculty of Software and Information Science,Iwate Prefectural University,Japan

単位とした通信を行う。これにより、協調プロトコルの分離記述や、応用依存部から協調プロトコルを再利用することが可能になる。

このようなプロトコル指向の概念はすでにAgenTalk[AgenTalk]において提案されているが、本研究では複数の言語による実装を考慮している点が異なる。

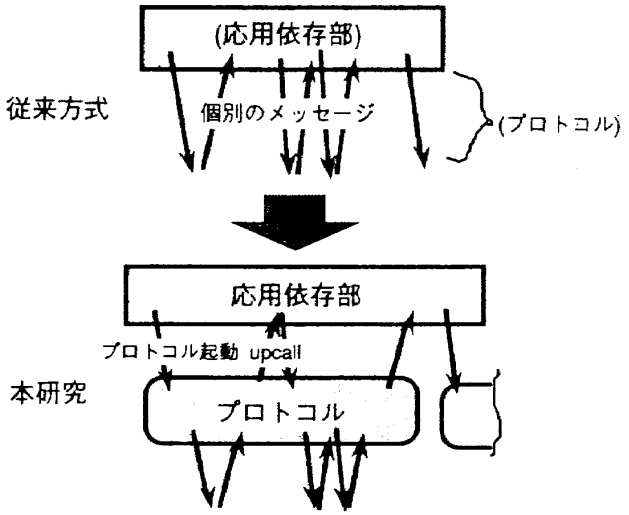

図2. プロトコル指向型インタフェース

応用依存部と協調プロトコル部、およびメッセージ転送部は、異なった言語により記述できることが本アーキテクチャの最も重要な点である。したがって、特定の記述言語に依存した方法によって各部を結合することは望ましくない。また、あまり汎用的な結合を許してしまうとインタフェースが肥大化し、インタフェース自体の製作が困難となる。このため、特に複雑になると思われる応用依存部と協調プロトコル部間で必要なインタフェースを整理し、より多くの言語が利用可能なようにする。

2.3 言語間インタフェース

言語が多くなればn対nの変換は非現実的となるため、どこかで1対nの変換を行うのが望ましい。このため図3のように、言語アダプタとエージェント内バスを用意する。

・エージェント内バス：言語に依存しない共通のAPIを提供する。また、バスを通る情報の宛先のうち、言語アダプタレベルでの振り分けを行う。

・言語アダプタ：共通APIと言語依存APIの変換を行う。また、情報の宛先のうち、同一言語内で異なる記述への振り分けも行う。

3．記述例

本研究のアーキテクチャに基づくエージェントの記述例を図4に示す。これは、[COOL]にて単一言語で書かれているエージェントの一部を、本アーキテクチャ用に書き直したもので、渾然一体となっていたプロトコル部と応用依存部を明確に分離し、異なった言語で記述することが可能となった。

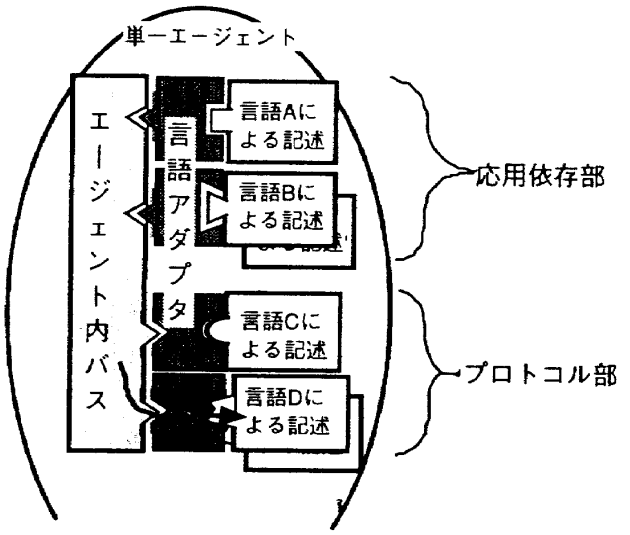

図3. 応用依存部⇔プロトコル部結合

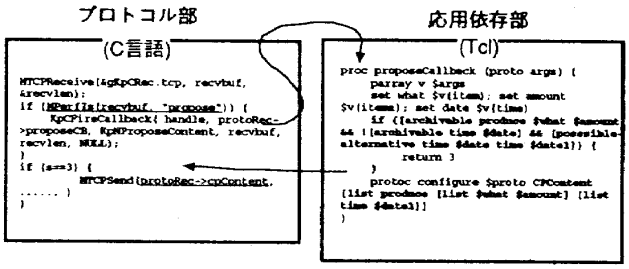

図4. 本研究による記述

4．まとめ

複数の言語を用いてエージェントを記述することが可能なエージェントのアーキテクチャを提案した。このため、エージェント内を3層に分割し、それを実現するための要素について述べた。

今後はプロトタイプ実装を整理し、実際に応用を開発してみる予定である。

謝辞

本研究の方向性を与えていただき、その後もご指導いただいた東北大学の白鳥則郎教授、および木下哲男助教授に感謝いたします。

参考文献

[AgentTalk] "AgentTalk: マルチエージェントシステムに置ける協調プロトコル記述", 桑原和宏,石田亨,大里延康, 電子情報通信学会技術研究報告 AI94-56(1995-01)

[HLP] "High-Level Protocols", Robert F. Sproull, Dan Cohen, Proceedings of the IEEE Vol.68 #11 Nov.1978

[KQML] "A Proposal for new KQML Specification", Yannis Labrou,Tim Finnin, http://www.cs.umbc.edu/~jklabrou/publications/tr9703.ps

[COOL] Mihai Barbuceanu and Mark S. Fox, "COOL: A Language for Describing Coordination in Multi Agent Systems", http://www.cs.umbc.edu/kqml/papers/kool.ps