2W－6

視覚化プログラミングによるエージェントシステム構築ツール*

青木 寛　丸尾 康博　木村 耕
電気通信大学 情報工学科

1　はじめに

近年システム開発において、コンポーネントウェアやエージェントなどが注目されている。コンポーネントウェアはソフトウェアの部品化に着目した技術であり、特にその視覚化プログラミングが注目されている。一方、エージェントは知的処理に着目したものであり、柔軟なシステム構成を可能にする。筆者らも昨年、エージェントによる部品管理システム Probe[1] を開発した。今回は、このときの開発経験を通じて、より汎用性の高いエージェントシステム構築ツール (以後、Ascot と呼ぶ) の必要性を感じた。

2　Ascot の構想

2.1　方針

エージェントシステムを構築する際に、メッセージに対するエージェントの動作の決定、エージェントの配置などが必要となる。これらを視覚的に行なうツールを作成する。また本ツールを試用し、ツールの有効性について検証する。

ツールを作成するにあたり、以下の点について考慮する。

- 視覚化を重視する。
- エンドユーザ指向のシステムとする。
- 柔軟で標準化に対応し得るものとする。

視覚化やエンドユーザ指向についてはコンポーネント技術を参考にして実現する。標準化についてはエージェントの標準化案である FIPA を利用することにした。

2.2　コンポーネントと視覚化

コンポーネントとは、オブジェクト指向に基づいたソフトウェア部品のことであり [2]、コンポーネント技術により、ソフトウェア部品を組み立てることでシステム開発ができるようになる。このため、従来のソースコードの再利用に比べると、工期が短縮できることが報告されている。

コンポーネント技術の適用である JavaBeans では、コンポーネント (Bean) はプロパティ、イベントなどにより特徴づけられる。プロパティは Bean の外観や動作を決定するのに使われ、変更することで Bean をカスタマイズできる。イベントは Bean の状態変化を通知し、複数の Bean の協調動作を可能にする。

これらの特性はエージェントの心的状態、メッセージ通信などと類似した特徴であり、Ascot を実装する上で参考にした。

2.3　FIPA 準拠のエージェント仕様

FIPA[1] は 1996 年に設立されたエージェントの標準化を目的とした非営利団体である。エージェントの活動の場となるプラットホームや、各プラットホームに存在しなければならない特殊なエージェント (ACC, AMS, DF)[2] 、それらの間でかわされる会話などについて規定している。

本ツールで扱うエージェントは FIPA 準拠のものとする。エージェントは FIPA で規定されたメッセージを交換し、内部で保持するルールに従って心的状態を変化させる。このルールをコミットメントルールといい、これによりエージェントの動作が決定される。コミットメントルールについては Agent-0[4] を参考にした。また、FIPA で定める特殊なエージェントは予め用意する。

エージェントが行なう言語処理の部分は外部オブジェクトとし、このプロセッサを複数登録することにより、エージェントに複数の言語処理能力を与えることができる。

3　Ascot の機能

本ツールは以下の機能を実現し、エージェントシステムを構築するための支援ツールとして機能する。

---

* Agent system construction tool by visual programming, Hiroshi Aoki, Yasuhiro Maruo, Koh Kimura, The University of Electro-Communications

[1] "Foundation for Intelligent Physical Agents", http://www.cselt.it/fipa/

[2] Agent Communication Channel, Agent Management System, Directory Facilitator

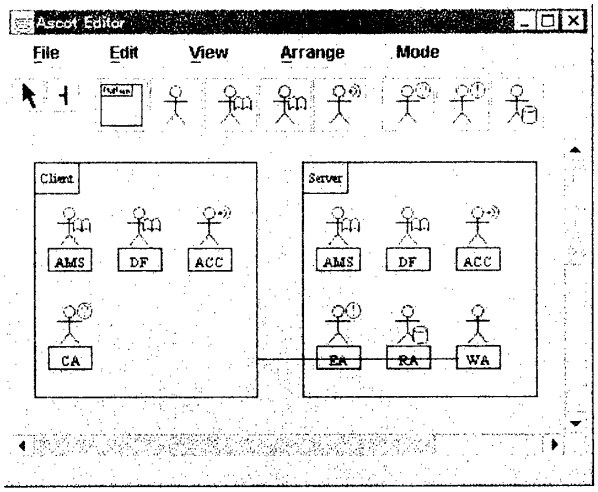

図 1: エディタによる編集例

- システムの設計 (エディタ)
- システムの模擬動作 (シミュレータ)
- システムの生成と出力 (ビルダ)

これらの機能について順に説明する。

3.1 エディタ

プラットホームに対応するオブジェクトを作成し、そこにエージェントを配置することで、システムを構築する。また、エージェントの保持する心的状態やコミットメントルールを編集することで、エージェント自身の知識や動作を指定する。

3.2 シミュレータ

シミュレータによりエージェントシステムの動作シミュレーションを行なうことができる。このときエージェントは Ascot 内でしか動作しないが、この機能を利用することで、構築するエージェントシステムの動作を早期に確認できる。

3.3 ビルダ

Ascot で作成したエージェントシステムを、Ascot とは独立して動作する形で出力する。移動エージェントをサポートするために、既存のエージェントライブラリである Odyssey[3] を利用する。Odyssey は place の概念を持っており、FIPA のエージェントモデルと近いので、これを利用することにした。

[3] http://www.genmagic.com/technology/odyssey.html

4 評価と考察

評価方法として、以下で記述したシステムを開発する際に、ツールを使用した場合と使用しなかった場合の工期を比較する。

> プラットホーム A に存在するエージェント a がプラットホーム B に移動し、B に存在するエージェント b から知識を獲得し、A に戻る。

実験結果を表 1 に示す。実験結果から、ツールを利用することにより開発工期が大幅に短縮された。特に、被験者αは使用したエージェントライブラリについての知識がなかったため、ツールを使用しなかった場合、ライブラリの学習に大きく時間を取られた。ツールを使用することで、エージェントライブラリの予備知識の有無によらず、短時間でシステムを開発できるようになった。

表 1: 実験結果

開発者	α	β
ツール不使用	8.1	3.5
ツール使用	0.38	0.67

単位は時間

5 おわりに

現在エージェントやプラットホームの間で結ばれる関連は一種類しか扱っていない。今後は他の関連の洗い出しと実装を行っていきたい。

参考文献

[1] 岩田真明, 青木寛, 山腰哲, 丸尾康博, 木村耕: “エージェントを用いたソフトウェア部品管理システム Probe の考案及び実現”, 情報処理学会, 第 56 回全国大会, 講演論文集 (分冊 1), pp.247-248 (1998).

[2] 青山幹雄: “コンポーネントウェア: 部品組み立て型ソフトウェア開発技術”, 情報処理学会, Vol.37, No.1, pp.77-79 (1996).

[3] 木下哲男, 菅原研次 著: “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 (1995).

[4] Yoav Shoham: “Agent-oriented programming”, Artificial Intelligence, Vol.60, pp.51-92 (1993).