パターンをもちいたエージェント技術の実現

2003MT103 立奏 将輝
指導教員　張 漢明

1　はじめに

エージェントソフトウェア [1] は，環境の変化に対してソフトウェアの動的な構成の変更をする特性がある．エージェントソフトウェアは，実行時に振る舞いや構造を変更する．動的な機構によって，開発者が意図しない動作をする可能性がある．

本研究の目的は，エージェントソフトウェアの概念で，静的な部分を分析して，デザインパターン [2] で実現し，考察することである．実行前にソフトウェアがどのような構造の変化をするのかを把握できるようになると考えた．

研究手順を以下に示す．

- エージェントソフトウェアの分析
- エージェントソフトウェアをパターンをもちいて設計実現
- エージェントソフトウェアをパターンで実現することの妥当性について考察

2　エージェントソフトウェア

オブジェクト指向の特性に，独自の新たな特性を加えたものがエージェントである．独自の新たな特性を以下に示す [1]．

- 適応性：環境の変化に対して，動的にエージェントの構成を追加，変更する
- 自律性：個々のエージェントが環境の変化に対して，動的に判断してプランニングする
- 協調性：エージェント同士が互いに連携して協調動作する

3　オブジェクト指向による設計実現

エージェントソフトウェアを，以下のようにオブジェクト指向で実現できると考えた．

- 適応性：そのときの状況によってエージェント自身の構成を変えるので，プロトタイプパターンをもちいる
- 自律性：動作があらかじめ決まっておらず，そのときのエージェントの状態にあった動作をするので，プロトタイプパターンとメディエータパターンをもちいる
- 協調性：ほかのエージェントを実行できるような機構を，エージェントに持たせる

実現にはオブジェクト指向プログラミング言語として一般的な Java[4] をもちいた．

3.1　適応性

プロトタイプパターン

適応性は，プロトタイプパターンをもちいて実現した．エージェントの構成を変える構造を**図 1** に示す．

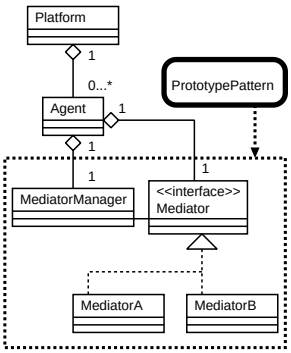

図 1　適応性

Mediator は，プラナーの役割りをしている．プロトタイプパターンをもちいることで，プラナーの切替えが容易になると考えた．

3.2　自律性

プロトタイプパターンとメディエータパターン

エージェントの動作に関する構成要素を生成するのに，プロトタイプパターンをもちいて実現した．エージェントの動作に関する構成要素同士の協調動作を実現するのに，メディエータパターンをもちいた．構造を**図 2** に示す．

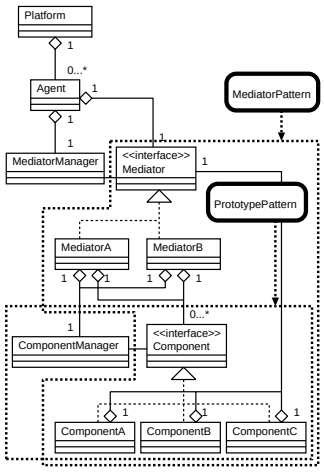

図 2　自律性

Component は，エージェントが動作する単位を表している．Component は，Mediator が生成されたとき，同時に生成される．Mediator は，Component を実行することができる．

Component は，自身の動作が終了したことを Mediator に通知する．通知を受けた Mediator は，メッセージに対応した動作をする．

3.3 協調性

複数のエージェント同士の協調動作を実現した例を**図 3** に示す．

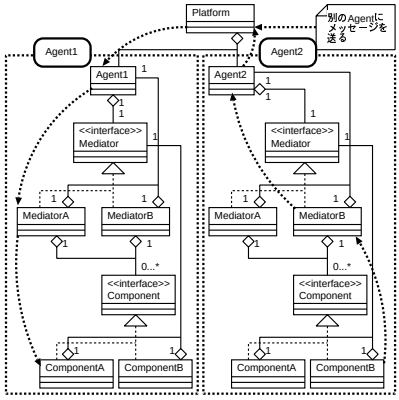

図 3　協調動作の例

Agent2 は，Agent1 の Component を実行することができる．Component は，Mediator，Agent，Platform の順にメッセージを送る．メッセージを受け取った Platform は Agent1，ComponentA の順にメッセージを送り，Component を実行する．

4 考察

実現したエージェントソフトウェアをもとに，構成要素の追加，変更と事前把握とエージェント指向言語との比較について考察する．

4.1 構成要素の追加，変更

実現したエージェントソフトウェアに対する，追加，変更について考察する．

プロトタイプパターンをもちいることで，Mediator を切替えやすくなると考えられる．例えば，**図 2** の Mediator を切替える場合，MediatorA を削除し，MediatorB を生成することで実現できる．

メディエータパターンをもちいることで，Component 同士の協調動作は，Mediator が管理するようになり，Component 同士は疎結合になると考えられる．例えば，**図 2** のエージェントに ComponentD を追加する場合，Mediator だけを変更すればよいので，追加，変更が容易になると考えられる．

4.2 事前把握

実現したエージェントソフトウェアでは，メディエータパターンをもちいることで，実行時の Component 同士の協調動作に関する記述を，Mediator が集中的に管理することができる．Mediator に局所的に記述されるので，実行前に挙動が把握しやすくなると考えられる．例えば，**図 2** のエージェントでは，実行前に MediatorA と MediatorB の記述を見ることで，Component 同士の協調動作を事前に把握しやすくなると考えられる．

4.3 エージェント指向言語と比較

実現したエージェントソフトウェアをエージェント指向言語 Flage[3] と比較し，考察する．Flage では，実行前にエージェントの構成を把握するのは難しいが，実現したエージェントソフトウェアは実行前にエージェントの構成を把握できる．Flage では，実行中に新たにプランを獲得することができるが，実現したエージェントソフトウェアは実行中に新たにプランを獲得することができない．

4.4 実現したエージェントソフトウェアの利点と欠点

実現したエージェントソフトウェアの利点は，実行前に実行時の動作や構造を把握できるようになったことである．欠点は，実行時に新たな構成要素を追加，変更できないので，記述した動作以外には対応できないことである．

5 おわりに

デザインパターンをもちいることによって，適応性と自律性を実現し，構成要素の追加，変更が容易になったか考察した．パターンで実現したエージェントソフトウェアは，新たにプランを獲得することはできないが，事前把握しやすいことを確認した．

今後の課題は，実現したエージェントソフトウェアをもとに，アスペクト指向をもちいて実現し，構成要素の追加，変更が容易になるか確認することである．本研究で実現したエージェントソフトウェアは，実行前に動作や構造を決定する必要があるので，構造が複雑になり，記述量が増加する．アスペクト指向をもちいて実現することで，構造が整理できるので，構成要素の追加，変更が容易になり，実行時に動作や構造が把握しやすくなると考えられる．

謝辞

本研究を進めるにあたり，熱心な御指導をいただいた野呂昌満教授，有益なアドバイスを下さった張漢明助教授，蜂巣吉成講師，大学院生の坂野将秀さん，久松康倫さん，水野耕太さん，安孫子正康さん，西山遼平さん，太田将吾さん，安江基規さんに深く感謝いたします．

参考文献

[1] 本位田真一，飯島正，大須賀昭彦, オブジェクト指向トラックエージェント技術, 共立出版,p.252,1999.

[2] E.Gamma,J.Vlissides,R.Helm,and R.Johnson, Design Patterns Elements of Reusable Object-Oriented Software,Addison-Wesley,p.249-325, 1995.

[3] Flage, Field oriented Langage for AGEnts, http://www.ipa.go.jp/archive/NEWSOFT/public/Flage/.

[4] Java, http://www.java.com/.