3F-8

# 多目的で柔軟なマルチエージェントシステムの設計 *

山崎 航† 溝口 文雄†

東京理科大学 理工学部 経営工学科‡

## 1 はじめに

マルチエージェントシステムは複数のエージェントどうしが協調することによって，複合的なタスクを遂行することを特徴とする．現在マルチエージェントシステムは様々なものが提案され，実用的な問題にも適用可能になりつつある．しかし，特にロボット等の物理的な機器を含むような環境内においては，機器は実際には様々なことを行なう能力があるのにもかかわらず，単一の目的のためだけに利用されてきた．また，このような環境ではエージェントの増減や，エージェントの状態の変化，環境の変化が頻繁に起きることが予想されるため，タスクの実現性を動的に検証する必要がある．本稿ではタスクの階層性を利用することによってこれらの問題に対応した，多目的で環境に柔軟なマルチエージェントシステム JMAC (Java Multi-Agent Component) の設計・実装について述べる．

## 2 設計方針

JMAC は，溝口らによって 並列論理型言語を用いて開発されたマルチエージェント記述言語 MRL[1] のアイデアを参考にしているが，実装は Java 言語によって行なった．JMAC は言語ではなく，必要な機能を提供するクラスライブラリから構成される．これは，構築するマルチエージェントシステムに対して，Java で書かれた既存のコンポーネントが利用できることを意味する．本稿で述べるライブラリは以下の基本的な機能を提供する．

**[エージェントの状態遷移]** ユーザはエージェントの状態遷移を記述することによって個々のエージェントの動作を定義する．状態はイベントによって遷移する．

**[タスクの割り当て]** 同種の能力をもつ複数のエージェントに対してタスクの分配を効率良く行なうためにの交渉を行なうプロトコルを提供する．

**[実行可能性の検証]** 環境の変化に対して柔軟であるために，自分の状態に加えて，他のエージェントの状態を考慮した実行可能性を検証することを可能とする

*Design of Multi-pourpose and Flexible Multi-agent System

†Wataru YAMAZAKI, Fumio MIZOGUCHI

‡Dept.of Industrial Admin. Faculty of Sci. and Tech. Science University of Tokyo

## 3 タスクの階層性とその利用

本章では，JMAC 用いる具体的な例として，プリントアウトした紙をユーザの元まで運搬する，配達タスクを用いる．このタスクは複数の移動ロボット，腕型ロボット，カメラロボット，プリンタ，センサなどをエージェントみなし，移動ロボットの運搬能力，移動ロボットの移動を補助するカメラの視覚能力，紙をつかむ腕型ロボットの能力などの複合的な能力であると見ることができる．図 1 は移動ロボットと腕型ロボットに関しての階層構造である．本章の残りの部分では，この例を用いて，前章で示した基本的な機能について述べる．

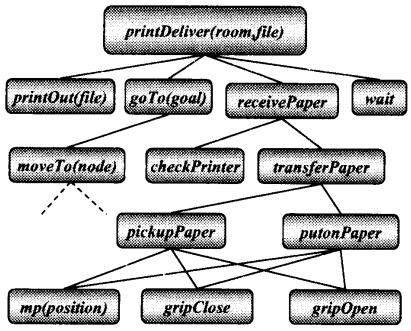


図 1: タスクの階層性

### 3.1 エージェントの状態と状態遷移

JMAC での各エージェントは，Task, Action, Negotiation の 3 つの状態を持つ．Task は，その他のエージェントからも見ることのできる public な状態を表し，そのエージェントにそのタスクを行なう能力があることを他のエージェントに宣言している．したがって，Task のとり得る状態数は，そのエージェントが実装しているタスクインタフェースの数と同じになる．

Action は基本的には Task を各エージェント内で分解した外から見えない private な状態を表す．Negotiation は交渉の状態を表し，requesting, acceptiong, null の 3 つの状態をとり得る．本ライブラリでは Task レベルの状態はインタフェースの抽象メソッドとして表す．例えばプリンタデリバリで一番最初にそのタスクを受けとるのはプリンタエージェントであるが，そのためにプリンタエージェントは以下のような構造を持ち，タスクが実行されると，タスクはスレッドとしてその他のスレッド

と並行に動作する．

```
public class PrinterAgent extends Agent
      implements PrintDelivable{
   public synchronized void printDeliverTask(..){
      PrintDeliverTaskThread tt;
      tt = new PrintDeliverTaskThread();
    }
}
```

Actionの状態はタスクのスレッドのrunメソッドから呼び出されるローカルなメソッドに相当する．イベントに対応したメソッドを呼び出すと，定義された遷移のマップにより，実際に呼び出すメソッドが決定される．遷移のマッピングにはFSM(Finite State Machine)を記述するためのスクリプト言語SMC[2]を参考にした．

### 3.2 タスクの割り当て

あるエージェントがタスクを単独では実行できない場合，その部分を他のエージェントに委譲する．このような場合には契約ネットプロトコル[3]を用いる．各エージェントは，実行可能なエージェントが複数存在する場合を考慮して実行に要するコストを計算して問い合わせ先に答える．問い合わせは登録されたエージェント群にブロードキャストされ，契約成立の後は一対一通信を行なう．本システムでは後述するように契約の際に自分の状態と，与えられたタスクから実行可能性の検証を行ない，実行が可能であった場合にのみ問い合わせに答える．実装にはブロードキャストをマルチスレッドを用いることにより，各エージェントの検証は並行に動作する．

### 3.3 実行可能性の検証

エージェントの数が増減し，多目的に利用されると，実際に問い合わせのあったタスクを実行できるどうかを検証する必要に迫られる．このような問題に対して，Raoらはモデルチェッキングを用いて動的にモデルを生成し，実現可能性を証明している[4]．しかしながら，各エージェントにおけるモデルの構築に時間がかかるという問題点がある．またMRLでは定理証明を用いることによって動的にモデルを生成している．これに対し本研究では，Taskレベルの可能性の検証を型とメソッドのマッチングによって行ない，Actionの検証をプロトコル解析等で利用されている全探索に基づく検証[5]を用いて可達性の解析を行なっている．通常，可達解析を行なう場合には，深さ優先探索を用いるが，本研究では，コストの計算を含める場合にはA*アルゴリズムを用いて最適解のコストを得ている．また，タスクの実行と同様に，サブタスクに分解した場合，サブタスクの検証はそれを行なうエージェントにまかせる．途中のサブタスクの検証に失敗した場合，それを用いる複合的なタスクの検証も失敗するため，結果としてタスク全体の検証を行なうことになる．実装ではすべてのエージェントにブロードキャストされ，型を確認し一致すれば検証を行なうスレッドを起動するメソッドを呼び出す．

```
public synchronized void requestAll
       (Method method, Object[] obj){
   RequestThread  rt = (RequestThread)obj[0];
   ...
   for(int i=0; i<agents.size(); i++){
     ((Agent)agents.get(i)).requested(method, obj);
   }
}
```

Taskの検証はそのエージェントがそのタスクを行なえる能力がそもそもあるかを判断し，Actionの検証は能力を現在の状態で利用できるかどうかを検証していることになる．このような二段階の検証により，すべての状態探索する場合に比べて探索のスペースは減少させていることがJMAC特徴である．

## 4 多目的な利用

タスクの階層的な性質は，いくつかの基本的な機能をエージェントに定義しておけば，それらを用いて，多様なタスクを行なうことができることを意味する．例えばプリンタデリバリで用いた移動ロボットの運搬能力は，ユーザからユーザへものを運ぶのに利用することができる．また，カメラロボットエージェントは，移動ロボットを誘導するだけでなく，監視カメラとしても利用することができる．このように，JMACを用いて構築されたマルチエージェントシステムは，エージェントに基本的な機能を定義することによって，それを用いた複合的な機能を構築しやすい．

## 5 おわりに

本稿では，タスクの階層性を利用することにより，基本的な機能をエージェントに定義することで，多目的なエージェントを定義し，また，契約ネットプロトコルと二段階の検証を行なうことによって，効果的に実行可能性を実現し，多目的で柔軟なマルチエージェントシステムJMACを設計した．また，実際にJMACを用いてエージェントシステムとその例を実装し有効性を示した．

## 参考文献


[1] F.Mizoguci, H.Nishiyama, H.Ohwada and H.Hiraishi, Smart office robot collaboration based of a multi-agent programming, Artificial Intelligence, Vol.114,1999

[2] http://www.objectmentor.com/

[3] R.Smith, The Contact Net Protocol: High-Level Communication and Contactrol in a Distributed Problem Solver, IEEE Transactions on Computers, Vol. C-29, No.12, 1980

[4] A.R.Rao and M.P.Georgeff, A model-theoretic approach to the verification of situated reasoning systems, IJCAI-93,1993

[5] G.J.Holzmann, Design and Validation of Computer Protocols, Prentice Hall, New Jersey, 1991.