連載解説◆◆◆「AI技術の産業応用」〔第9回〕

次世代生産システムへのエージェントによるアプローチ

An Approach for Future Manufacturing Systems Using Agents

和田 英彦
Hidehiko Wada
横河電機株式会社
Yokogawa Electric Corporation.
Hidehiko_Wada@yokogawa.co.jp

Keywords: mobile agent, autonomous agent, autonomous decentralized system, manufacturing system.

1. はじめに

製造業の生産形態や生産組織は，消費者のニーズの多様化や急速な変化，ワールドワイドでの競争を行なう必要性などにより，大きく変化していかなければならない[牧野 97, Shen 99]．生産環境の要求の急激な変化には，現在の生産システムでは対応しきれず，新たな要求に素早く対応できる柔軟性を持つことが，次世代の生産システムに強く求められている．ここでの柔軟性とは，例えば生産ラインの変更が容易に行なえることや，複数の製品を一つのラインで生産することができ，さらに生産する製品の種類を，製品の改良や特注品に対する個別の実験的な要求も含めて容易に変更できること，つまり変種変量生産が実現できることを意味している．

従来から使われている集中型でシーケンシャルな生産計画，スケジューリング，制御機構を持つ階層型/集中型のシステムでは，種々の生産形態や生産要求の変化に柔軟に対応するには不十分である．階層型/集中型のシステムは，生産システムの拡張性や構成変更の能力を制限しており，予期しない状況が発生した場合にも効率よく対応することができない．また，新しく発生した要求に合うようにシステムを改造する場合にも，多くのソフトウェアモジュールを変更する必要がある．さらに，一つの障害がシステム全体のシャットダウンへつながる場合があり，システム全体が脆弱な面を持っている．

生産システムに柔軟性を持たせるために，いくつかのアプローチが提案されており，自律分散システムの考え方もその一つである[Mori 98]．次世代生産システムに求められている生産物と生産ラインの柔軟性を実現するために，我々は個々の生産物と，生産ラインに設置された装置にそれぞれエージェントを対応させることにより，協調性および自律性を持たせた自律分散型のシステムの実現を目指している[星 98]．ターゲットとなる製品やワークの情報，たとえばその製品を作るための手順や製造データなどを生産物側のエージェントに持たせ，製品の加工は生産物側のエージェントから装置側のエージェントに指示を出すことにして各製品の加工を実現し，集中的な管理機構をシステムから取り除いている．

提案しているシステムの実現性を実証するために，我々は，実験的な加工機械制御システムを開発した[Wada 99]．開発したシステムは，カーネギーメロン大学（Carnegie Mellon University, CMU）のロボット工学研究所（Robotics Institute）の Shape Deposition Manufacturing（SDM）実験室[SDM Homepage]向けの加工機械制御システムである．

本稿では，2章で我々の提案している基本コンセプトについて述べる．3章では，CMUで開発したプロトタイプシステムのソフトウェアアーキテクチャと実装について説明する．4章では，開発したプロトタイプシステムについて議論し，関連研究と比較する．最後に，今後の課題についてまとめる．

2. 基本コンセプト

柔軟性を持つ生産システムを実現するために，階層型のシステムではなく，エージェントを用いたフラットな構造のシステムを筆者らは提案している．基本的な考え方を以下に説明する．

まず初めに，作ろうとする製品一つ一つに対応してエージェントを生成する．このエージェントを製品エージェント（Product Agent）と呼ぶ．製品の注文が入ると，その注文に対応して製品エージェントを生成すると考えてもよい．一方，それぞれの加工装置や加工機械に対応させて，装置エージェント（Machine Agent）と呼ぶエージェントを置く．装置エージェントは，個々のデバイスや装置を抽象化したオブジェクトである．我々は製品エージェントに，その製品エージェントが担当する製品の製造手順を与える．ここでは，製造手順のことを“レシピ”と呼ぶ．製品エージェントは，レシピで与えれらた手順に従って，必要な加工装置を選択して，その装置やコン

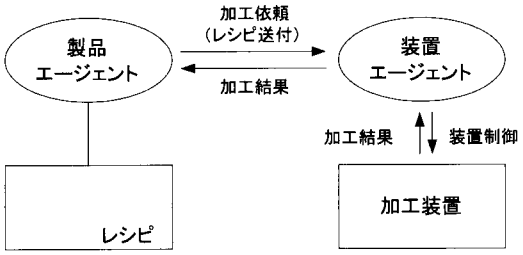

図1 基本コンセプト

トローラに具体的な加工手順を指示する．実際には，製品エージェントが加工装置へ直接指示を出すのではなく，製品エージェントは装置エージェントへ加工指示を出し，装置エージェントが装置やデバイスを制御する（図1）．

製品エージェントは，システムや周りの状況を判断しながら，その変化に動的かつ効率的に対応しながら，システム内を自律的に動作し，担当している製品を完成させる．また，製品エージェントや装置エージェントは，必要に応じて他のエージェントとネゴシエーションを行なってもよい．

上述のコンセプトを実現するために，製品エージェントを移動エージェント（mobile agent）とし，また装置エージェントを非移動型のエージェント（stationary agent）として考える．装置エージェントは，加工装置を制御しているコントローラやコントローラに接続しているコンピュータ上で動作する．一方，製品エージェントは移動型のエージェントであり，加工を依頼する装置に対応する装置エージェントが動作するコンピュータ上に，加工している製品やワークの動きを追う形で移動する．ターゲットとなる装置エージェントが動作しているコンピュータ上で，製品エージェントは装置エージェントに対して具体的な加工依頼，指示を送り，その加工処理の終了を装置エージェントが知らせるまで待つ．

提案しているアーキテクチャでは，加工指示は通常個々の製品エージェントから出される．製品を加工したり製造することに関しては，集中的に管理する部分は必要ではなく，提案しているシステムは，システム全体としてフラットな構造になっている．従って，製品や装置の数が増えた場合に，従来の階層型のシステムでは全体を管理する部分に負荷やメッセージが集中することがあるが，提案しているアーキテクチャでは，そのような負荷やメッセージの集中を避けることができる．

3. プロトタイプシステム

コンセプトを実証するために，我々はCMUのSDM実験室においてプロトタイプシステムを開発した．この章では，まず最初にSDM実験室の紹介を行ない，次にターゲットシステムのハードウェア構成を説明する．その後，コンセプトを実現したソフトウェアアーキテクチャと実装について述べる．

3·1 SDM実験室

Shape Deposition Manufacturing（SDM）実験室は，CMUのロボット工学研究所に属している．SDM実験室では，任意の複雑な構造や形状の物を，CADのモデルのデータから自動的にかつ素早く工作できるSFF（Solid Freeform Fabrication）プロセス*1を実現している．SDMの研究では，従来の技術では工作できなかった，複数の材料から構成される物や，中にセンサなどを埋め込むような物を工作する方法の研究に注力している．その応用範囲は，次世代の工作ツールから，個人が身につけるようなウェアラブルコンピュータ（wearable computer）などの広い範囲にわたっている．

SDM実験室では，工作物はパレットと呼ばれる金属製の板の上に作られていく．パレットは，パレット移動用のロボットによって，パレット受け取り機構を持った複数の加工装置にセットされる．現在のSDMテストベッドは，コンピュータ数値制御加工機械（CNC），材料堆積装置（Deposition Station），ショットピーナ（Shot-peening Station，材料の表面に硬い粒子をたたきつけて加工をする機械），洗浄機（Washer）の四つの装置から構成されている（図2）．

図2 SDM実験室概観

3·2 現状のシステム

SDM実験室のシステムは，長年にわたって学生による改良や拡張が行なわれてきている．そのようなad hocな拡張の結果，システム構成が複雑になってきており，システムの維持管理や新しい装置を付け加えることが難しくなっている．また，システム全体が一つのコントローラにより制御される典型的な集中型制御システムであり，

*1 SDM実験室の研究テーマの一つで，材料を堆積していく作業と堆積した材料を削る作業を繰り返すことで，自由度の高い形の物を加工するプロセスのこと．

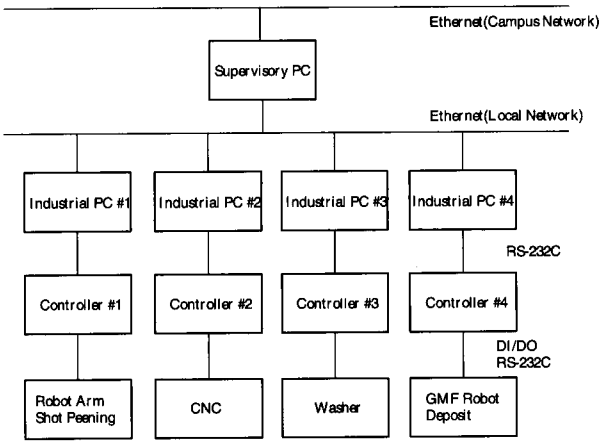

図 3 ハードウェア構成

二つの工作装置を同時に使用することができない，などいくつかの改善すべき点が発生している．

これらの問題点を解消するために，ハードウェア，ソフトウェアともに改めることになり，そこで我々の提案しているエージェントベースの制御システムを導入することになった．

3·3 ハードウェア構成

本システムでは 4 台のメインとなるプログラマブルコントローラと，2 台の補助のサブステーションを使用している．これらのメインコントローラはそれぞれ工業用パソコン（iPC）とシリアルライン（RS-232C）で接続されている．iPC は，イーサネットで相互に接続されている．それぞれのコントローラは加工装置にシリアルラインまたはディジタル入出力（DI/DO）接続されている．工業用 PC は，CPU が Pentium133MHz，メモリが 32MB の仕様である．

さらにオペレータが使用する管理用 PC（Supervisory PC）がある．システムのオペレータはシステムの監視や機械の操作を管理用 PC で行なう．管理用 PC は，CPU は PentiumPro 200MHz，メモリは 64MB の仕様である．図 3 にシステムのハードウェア構成を示す．

各 PC のオペレーティングシステムは Windows NT 4.0 である．ローカルネットワークは CMU のキャンパスネットワークから切り離されている．これは，本システムのそれぞれのノードが通信している間に外部からの影響を受けないようにするためである．

3·4 ソフトウェアアーキテクチャ

開発したプロトタイプシステムでは，2 章の「基本コンセプト」で説明した二つのエージェントの他に，一つのエージェントとソフトウェアモジュールを導入し，全部で四つのソフトウェアモジュールでシステムを構成している．

- 製品エージェント
- 装置エージェント

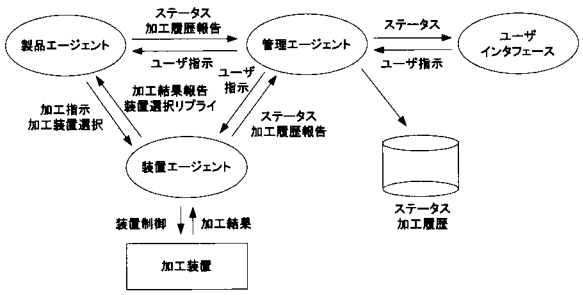

図 4 ソフトウェア構成

- 管理エージェント
- ユーザインタフェース

上記四つのソフトウェアモジュールの関係を図 4 に示す．以下に，それぞれのエージェント，モジュールについて説明する．

§1 製品エージェント

製品エージェントは，ターゲットとなる製品やワーク（本プロトタイプシステムの場合はパレットと考えてよい）と一緒にコンピュータやコントローラ上を移動する．製品エージェントは，“レシピ” と呼ぶ製品の加工手順を持っており，そのレシピに従ってターゲットの製品を加工していく．レシピの各行には，{機能名，コマンド}が対になって書かれている．製品エージェントは，レシピを 1 行ごとに処理していくことで，製造を完成させることができる．機能名には，各装置が実行できる作業名（たとえば洗浄機ならば WASH のように）が書かれていればよい．またコマンドには，工作機械への実際の加工指示が書かれている．

本システムで我々が使用しているレシピは，SDM 実験室で従来から使用されている形式をそのまま利用した．レシピの形式と例を図 5 に示す．

```
形式
<OP_NUM> <OP_NAME> <FUNCTION> <FILENAME>

<OP_NUM>:
        行番号

<OP_NAME>:
        PLACE|PICK|LOAD_FILE|RUN_FILE|DEL_FILE

<FUNCTION>:
        VMC6030|SPRAY|DRIP|WASH|PEEN|BLAST

例
001     PLACE           VMC6030
002     RUN_FILE        VMC6030  data.cnc
```

図 5 レシピ

<FUNCTION>は，本来の考え方からは，この行の作業を依頼するターゲットが持つべき機能名であるが，CMU のプロトタイプシステムの場合，従来のレシピの形式をそのまま利用したことと，一つの機能を実行できる装置が一つであることから，機能名が装置の指定となっている．<OP_NAME> は依頼する作業の内容を示している．

例えば，PLACEは指定する装置にパレットを置く作業であり，RUN_FILEは，指定した装置で指定したファイルで指示されている加工を行なう作業をすることである．<FILENAME>は，必要に応じて加工データなどのファイル名を指定する．そのファイルは加工に必要なデータや別の情報を持つ．

レシピは，各行が図5で示した形式になっており，それらが順番に並べられた形になっている．製品エージェントはレシピの各行を順に処理することで製品を完成できる．

製品エージェントは，レシピの各行に書かれている機能名を探索のキーとして，次に加工依頼を行なうことができる装置を探す．製品エージェントは，加工を引き受けることができる候補の中から一つの装置を選択し，コマンドに記述されている加工を指示する．実際には装置そのものではなく，装置エージェントが装置選択の要求や，加工指示を受け取る．加工装置や装置エージェントを選択するプロセスについては，4·3節でも考察する．

製品エージェントは，装置利用に関する排他制御を行なうために，一つまたは複数の装置エージェントを専有する．排他制御は，装置エージェントが発行するアクセス権を得ることで実現する．製品エージェントは，一つまたは複数の装置エージェントのアクセス権を同時に持つことができる．装置エージェントに加工依頼や指示を行なう場合には，まずターゲットとなる装置エージェントのアクセス権を取得した後に，実際の加工指示を送る．

§2 装置エージェント

一つの装置エージェントは，一つの工作機械に対応する．装置エージェントは，加工装置の制御に責任を持ち，また装置の状態も把握する．装置エージェントは，自分が担当する装置が実行できる機能を表す「機能名」を持つ．複数の装置エージェントが同じ機能名を持ってもよい．たとえば，同じ種類の加工装置や機械が複数台存在する場合である．先に述べたように装置エージェントの機能名はレシピの中で指定される．

装置エージェントは，多くても一つの製品エージェントに専有され，その製品エージェントからの指示で加工を実行する．装置エージェントは，自分が動作しているコンピュータに接続された装置やコントローラを制御し，それらの装置の状態を把握する．

装置エージェントの排他制御は，装置を利用できる製品エージェントにアクセス権を与えることで管理している．装置エージェントは，アクセス権を持っている製品エージェントからの指示によってのみ，装置を制御して加工を行なう．装置利用に関する，製品エージェントと装置エージェントのメッセージ交換の流れを図6に示す．

装置エージェントが，製品エージェントにアクセス権を与えている間，別の製品エージェントからの新しいアクセス権の要求はキューイングされる．アクセス権を持っている製品エージェントがそのアクセス権を装置エージェントに戻した後，装置エージェントは，新しい製品エージェントを選択して，アクセス権を与える．製品エージェントは，装置エージェントへのアクセス権のリクエストにプライオリティをつけて送ることができる．装置エージェントへのアクセス権の要求が複数キューイングされている場合，装置エージェントは一番高いプライオリティを持った要求を送った製品エージェントを選択し，アクセス権を与える．

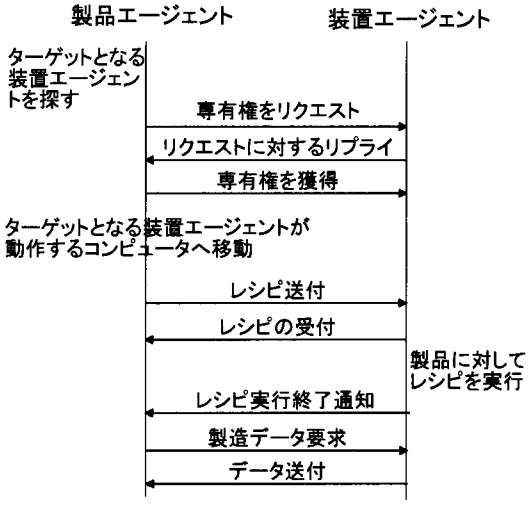

図6 メッセージの流れ

§3 管理エージェント

管理エージェントは，装置エージェントと製品エージェントを生成する．通常，装置エージェントはシステムスタートアップ時に生成され，製品エージェントは，個々の製品を実際に加工し始める時に生成される．製品エージェントの生成は，通常は生産順にシステムオペレータによって行なわれる．

また管理エージェントは，製品エージェントや装置エージェントから送られてくる製造データ，品質データ，製造実績を受け取り，データを永続的なストレージに保存する．また，管理エージェントは製品エージェントや装置エージェントの動きをモニタリングすることも行なう．

§4 ユーザインタフェースプログラム

オペレータの指示に従って，ユーザインタフェースプログラムは製品エージェントや装置エージェントへのオペレータの指示を知らせる．ユーザインタフェースプログラムは，システム内のエージェントや工作装置のステータスを表示する．

3·5 実　　装

我々は，インプリメンテーション言語としてJavaを使用し，またシリアルポートの制御にはJNI (Java Native Interface)*2を使っている．また，エージェントの移動と

*2 Javaからプラットフォーム固有のネイティブコードを呼ぶためのインタフェース．
http://java.sun.com/products/jdk/1.2/docs/guide/jni/

エージェント間のメッセージ通信には，IBM で開発された Aglets (alpha5) [Aglets Homepage, Lange 98] を使用している．また，管理エージェントとユーザインタフェース間の通信には HORB[HORB Homepage, Hirano 97] を使っている．Java の実行環境 (VM) は，当初 Sun Microsystems 社の JDK 1.1 ベースであったが，性能のために現在は Symantec 社の JIT コンパイラを使っている．コメントを除いたソースコードのライン数はおよそ 20,000 行であり，内部クラスを除いたクラス数は 117 である．また，ネットワークに接続されている計算機の時刻を同期させるためのプロトコルである NTP (Network Time Protocol) [Mills 91] をそれぞれの PC 上で使って，複数の PC 間の時刻の同期を図っている．

製品エージェントがターゲットとなる装置エージェントを探す場合，システムサイズが小さいことを考慮して，本プロトタイプシステム内の全てのコンピュータにメッセージが到達するブロードキャストメッセージを使用した．我々はこのプロトタイプシステムでは，スケーラビリティは考慮していない．それは，ターゲットとなるシステムが規模が限られている閉じた実験室のシステムであるため，ブロードキャストメッセージを使うので十分であると判断しているからである．

本プロトタイプシステムでは，自分が動作していることを知らせるハートビートメッセージ (または I'm alive message) を使ってシステムの可用性 (Availability) を向上させている．製品エージェントと装置エージェントが自身のステータスを，ハートビートメッセージとして管理エージェントに送り，管理エージェントがそのメッセージを受け取る．管理エージェントは，製品エージェントと装置エージェントの場所と状態を監視して，受け取ったステータスをエージェントのログとして保存する．製品エージェントや装置エージェントが止まった場合には，管理エージェントがそのことをオペレータに伝え，オペレータの承認の元に必要ならば製品エージェントを再生成して起動する．しかし，装置エージェントは必ずしも障害から回復できるとは限らない．それは，ハードウェアのステータスなどの必要な情報のいくつかを，ハードウェア的な制約などで得ることができない場合もあり，また，機械加工処理の場合には同じ処理を繰り返してできないこともあるからである．

3·6 性　　能

プロトタイプシステムでの実装を用いて性能測定を行なったので，いくつかの結果を示す．一つの評価はエージェントの移動に関するもので，もう一つの評価はレシピの実行に関するものである．

我々は，二つの Java の実行環境 (VM) を使用した．一つは，Sun Microsystems 社の JDK1.1.5 であり，もう一つは Symantec 社の JIT Ver3.0 である．Java の実行環境を変えて性能を測定したのは，これらの二つの実行環境の間で，システムの性能が大きく変わるということを認識をしていたからである．3·5 節で述べたように，実装の途中で Java の実行環境を JDK から JIT 環境に変更したのはこの性能の違いのためである．

表 1　測定環境

	CPU	Memory
Machine1	Pentium Pro 200MHz	64MB
Machine2	Pentium 166MHz	48MB

表 2　エージェントの移動に関する測定結果

Java VM	JDK 1.1.5	JIT Ver3.0
エージェントの移動時間	0.384(sec)	0.245(sec)

以下の測定は，CMU のプロトタイプシステムを使って実施したのではなく，社内の実験環境を使って実施した．実システムがすでに実運用に入っていたためであるが，実システムではなく実験環境を使うことは結果に影響を与えない．なぜならば，我々は機械加工時間を含めた実際のシステムの性能を測定するわけではなく，コンピュータシステム内での性能を測定することが目的だからである．

§1　エージェントの移動

この節では，エージェントが一つのマシンから別のマシンへ移動する時間の測定結果を示す．

測定環境を表 1 に示す．我々は二つのマシンを使い，それぞれのマシンはイーサネットで接続されている．この測定で使用したオペレーティングシステムは，Windows NT 4.0 である．ネットワークは，外部ネットワークからの影響や割り込みを避けるために，独立させている．

その結果を表 2 に示す．表 2 のそれぞれの結果は，クラスローディングなし*3に一つのコンピュータから別のコンピュータへ移動する場合の往復 (round trip) 時間である．この時間が小さいほど，エージェントの移動が速いということ意味する．それぞれの数字は 500 回の試行の平均値である．移動エージェントのサイズは数百バイトである．

この結果からエージェントの移動にかかる時間は 1 秒よりも短く，加工物の処理や搬送を行なう場合と比較して，十分に短い．本システムの場合は，この程度の時間ならば，エージェントの移動時間はボトルネックとなることはないと判断している．

§2　レシピの実行

レシピの実行性能に関しては，数行のレシピを実行する時間を測定した．この時間は，与えられたレシピでターゲットとなる製品を加工するために，システム (コンピュータ) 内で消費した時間を測定した．この測定のためのレシピは，

(1) PLACE WASH (パレットを洗浄機に置く)

*3　この場合は，移動先のコンピュータ上に移動エージェントが動作するために必要なクラスファイルが存在して，リモートの別のコンピュータからクラスファイルをロードしなくてもよいということ．

表3 レシピの実行に関する測定結果

Java VM	JDK 1.1.5	JIT Ver3.0
レシピ実行時間	27.78(sec)	16.35(sec)

(2) RUN-FILE WASH wash.dat (洗浄機の中にあるパレットを洗う)
(3) PICK WASH (パレットを洗浄機から取り出す)
というものである.

上記のレシピを処理する実行時間を測定した. 測定は表1内のMachine 1を使った. Javaの実行環境は, エージェントの移動の場合と同じで, 二種類の環境を使用した. また, 使用したオペレーティングシステムは, Windows NT4.0である.

測定結果を表3に示す. 表3の結果は, 5回実行した結果の平均値である. この簡単なレシピを実行する場合に, 二つのJava実行環境の間で10秒近い差があった. JDKを使用した場合は, バイトコードに対して完全にインタプリタとして動作するが, JITを使った場合には, 事前にJavaのバイトコードの一部をネイティブコードにコンパイルするため, このように実行速度に大きな差が出ている.

表3で示した結果は, レシピ実行の時間が速くないことを示している. この結果から, 厳密な時間制約をもつような工作機械やコントローラを制御するためには, 当時のエージェントやJavaの性能では十分でないことがわかった. それゆえ, 我々はエージェントが厳密な時間制約を守る必要がないような形で, このプロトタイプシステムを設計した. つまり, エージェントシステムや装置エージェントはコントローラ内のプログラムを起動すればよく, 起動されたコントローラ内のプログラムが時間制約を守りながら, 機械やデバイスをコントロールする構成にしている.

4. 議 論

4·1 システムの柔軟性

柔軟性は将来の生産システムを実現するための重要な課題であることは最初にも述べた. 我々が提案しているシステムでは, 将来の生産システムで求められている柔軟性を提供していると考えている. そこで,

- システム構成を変更すること
- 製品の生産順番を変更すること

の点に関する柔軟性について議論する.

最初にシステム構成の変更に関する柔軟性に関して考察する. 本システムでは製品エージェントは, 次に処理するレシピ行内の機能名を使って, 処理する工作機械を動的に選択する. CMUのプロトタイプシステムでは, まず, 製品エージェントはどの装置エージェントが指定したレシピを処理可能かを, ブロードキャストメッセージを使って問い合わせる. 問い合わされたレシピを処理可能な装置エージェントは, 自身が処理可能であることを製品エージェントに伝える.

開発したシステムでは, 装置エージェントは受け取ったレシピが処理できるか, できないかだけをリプライしているが, 先に述べたようにその他の付加情報, たとえば予想される処理時間や工作機械の空き情報などを付けることも可能である. 問い合わされたレシピを処理できない場合は, 装置エージェントはリプライを返さない.

製品エージェントは装置エージェントからのリプライを元に, どの装置エージェント, つまり工作機械を選択するかを決定する. 従って, 新しい工作機械が加えられたり, また既存の機械が保守や何らかの別の理由でたまたまサービスができない状況にあるなどの原因でシステム構成が変更されていたとしても, 製品エージェントはシステム構成の変更や機械の状況に対応して, 適切な工作機械を選択することができる. 製品エージェントは, どの工作機械が動作中であるか, 動作中でないかを知っている必要がなく, 装置エージェントからのリプライメッセージだけを使って, どの工作機械を使用するかを選択する. このように, 製品エージェントと装置エージェントは自律的に動作するので, システム構成が変化しても, エージェント自身で対応できる. また, 集中して管理する部分がなくても, システム構成が変化した場合も, システム全体としては動作を継続できる. 言いかえれば, 新しい工作機械をPlug-and-Play的に付け加えることができる. 集中管理部分がある場合には, 構成の変更や現在のステータスをその集中管理部分へ報告する必要がある.

次に製品の生産順序の変更に対する柔軟性について説明する. 本システムでは, ターゲットとする製品一つ一つが, 製品エージェントにマッピングされている. このため, 納期が迫っていて非常に急いで作らなければいけない特急ジョブの製品やワークがシステムに投入された場合にも, その急ぎの製品を担当している製品エージェントが他のエージェントと交渉などを行なって, 加工処理の順番を変更して, 製品を早く仕上げることも可能である.

開発したプロトタイプシステムでは, それぞれの製品エージェントに, 担当している製品の緊急度に応じた優先度を持たせている. 装置エージェントへの専有要求を出す時に, 製品エージェントはそのプライオリティも一緒に送る. 装置エージェントは, 次に専有を許可する製品エージェントを選択する際に, そのプライオリティが一番高い製品エージェントを選択している. つまり, 一番高い優先度をもつ製品エージェントが, 低いプライオリティを持つ製品エージェントに優先して専有する権利を得ることができる.

本方式では, 次の加工待ちをする製品やワークの中から一番プライオリティの高いものを優先して加工することが実現できる. しかし, 現在加工中の製品を加工途中で止めて, 優先度の高い製品の加工を新たに始めること

までは実現していない．加工を途中で中断する場合には，エージェント間の交渉や退避場所の確保などが必要になる．退避場所に関しては，退避場所を装置の一つとして考えることなどで対応可能あると考えているが，実現するためにはエージェント間の交渉が必要であり，エージェント間の交渉についてはまだ実現していない．

エージェントのインテリジェンスや交渉力を上げることによって，スケジュールや生産効率や優先度制御に関して他のエージェントと交渉することができるようになる．また，全体の生産効率を上げるためや納期を守るために，外部のスケジューリングシステムと協調することが必要となるかもしれない．これらの機能は現在のプロトタイプシステムではまだ実現されていない．今後の課題の一つである．

4·2 システム構成

十分な柔軟性を持った将来の生産システムを考えた場合に，集中型の管理部分へのメッセージの集中を避けるために，集中型の管理部分をなくし，サブシステム自身がそれぞれ独立に自律的に動作することは重要である．本システムでは，製品エージェントと装置エージェントがそれぞれ自律的に動作し，たとえ管理エージェントが存在しなくても，製品を加工する動作を継続することができる．製品エージェントと装置エージェントが，このシステムのメインとなる部分である．

また，製品エージェントをモーバイルエージェントにしたことで，製品エージェントがシステム内のコンピュータに分散して動作することになり，特定のコンピュータに負荷やメッセージが集中することを避けている．製品エージェントの移動性をなくすと，製品エージェントが特定のコンピュータ上に集まって動作することになり，作っている製品やワークの数が多くなると，そのコンピュータにメッセージが集中することになりスケーラビリティに欠けたシステム構成になる．また，製品エージェントが動作するコンピュータが止まってしまうと，システム全体に影響が及ぶことになる．

管理エージェントは，ターゲットとなる製品やワークを加工することに関しては付加的な部分である．つまり製品エージェントと装置エージェントさえ動作していれば，製品を作ることは可能である．一方，信頼性を増すために我々がCMUのシステムで採用した，管理エージェントが製品エージェントと装置エージェントを監視する方式は，スケーラビリティを確保できていない．よって，本方式をより大規模なシステムに適用する場合には，メッセージング方式を工夫するなど別のアプローチが必要である．

4·3 ターゲットとなる工作機械の選択

ここでは，製品エージェントが装置エージェントを選択する手順について考える．製品エージェントは，まず初めにレシピの最初の行を取り出す．製品エージェントは，そのレシピ行を実行できる装置を探すために，レシピ行内の機能名とコマンドを，装置エージェントへ送る．このメッセージは，全部の装置エージェントへブロードキャストするか，要求している機能名を持つ装置のグループへのグループ通信やマルチキャストを使って送られる．装置グループは，同じ機能名を持つ装置エージェントの集まりである．装置の数が多い場合や，スケーラビリティを要求される場合には，グループ通信を使う方が好ましい．CMUのシステムでは，前述のようにシステム規模が限られているために，ブロードキャストメッセージを使っている．

製品エージェントからのメッセージを受信した装置エージェントは，メッセージの内容から，製品エージェントの依頼を実行できるかどうかを判断する．その結果，自分が依頼された加工をできる場合には，実行可能であることを製品エージェントへリプライメッセージとして送る．

リプライメッセージを受信した製品エージェントは，リプライを返してきた装置エージェント（つまり加工装置）の中から，実際に加工を依頼をする装置エージェントを選択する．選択する基準は，それぞれの製品エージェントが持っている．たとえば，ある製品エージェントは早くリプライを返してきた装置を選択してもよい．CMUのシステムでは単純なこの方式を採っている．また別の製品エージェントは，装置の性能や利用可能性を考慮して選択するようにすることも可能である．

4·4 開発を振り返って

CMUでのプロトタイプシステムの開発では，実用に供するシステムを構築したわけだが，エージェントやJavaといった技術を，システム開発の場面で本格的には初めて使用したために苦労した点もあったので，その点にも触れておきたい．ただ，実際に開発したのが2年以上も前の時点なので，現在の感覚と多少ずれた部分があることをお断りしておく．

開発で一番苦労したことは，（当時の）Javaも含めたエージェントシステムの性能の問題である．開発したシステムではエージェント間でかなり頻繁にメッセージを交換している．開発がほとんど終り試験運用を続けている時に，システムが安定しないことがあった．いろいろと調査してみたが，なかなか原因を特定できない．そこで，Javaの実行環境をSun Microsystems社のJDKからSymantec社のJITに変えたところ，なかなか解決できなかった障害がピタッと止まってしまった．正確なところはわからないのだが，どうもJavaのスレッドの切り替えのスピードの問題などで必要なメッセージを受け取れていなかったようであった．

システムを構築する場合，開発環境は重要な要素を占める．特にCMUのシステムのように，システム内の独立したアクティビティをそれぞれエージェントとして扱

い，エージェント間でメッセージを（かなり頻繁に）交換するシステムを構築するような場合で，さらに今回の開発のように慣れていない環境を使用する場合には，デバッグ環境の充実は重要な課題となる．分散システムでかつマルチスレッドで動作するシステムをデバッグする環境は当時はもちろんだが，今でもあまり充実していないように筆者には思える．今後，解決すべき重要な問題の一つと考えている．

4·5 関連研究

生産システム内の個々の構成要素を活動主体として考えるアプローチは，ホロニック生産システム [Kriz 95] でも提案されており，[Bussmann 98] では，マルチエージェントを利用した実現方法を示している．またYAMS[Parunak 85] でもエージェントオリエンティッドな生産システムを提案している．しかし，[Kriz 95] や [Parunak 85] の論文では，従来の階層型のシステムの中で，マルチエージェントシステムを利用する方法について述べている．またエージェントを，スケジューリングを行なう部分に適用することを検討している．他にエージェントを生産システムに適用する研究事例もあるが，同様にスケジューリングの部分を対象としている研究が多い．一方，我々が提案している考え方やシステムは，装置を実際に制御する部分をターゲットとして考えており，エージェントを生産システムに適用しようとしている従来の研究とは違う部分をターゲットとしている．従って，従来の研究では触れられていなかった加工中の製品と加工装置を，どのようにエージェントシステムとして取り扱うかについては，新しい提案をしている．さらに従来から使われている階層型のシステムの中でマルチエージェントを適用していくのではなく，自律分散型のシステムを提案していることも従来の研究との違いであり，この方式により，将来の生産システムに必要不可欠な柔軟性をシステムに持たせることができている．

自律分散システムのコンセプトは [Ihara 84, Mori 93] でADSシステムとして提案されている．ADSシステムは，我々のシステムと同様にフレキシブルなシステムを目指している．それぞれのシステムの想定している階層が若干違うので，我々のアーキテクチャはADSアーキテクチャと共存することができる．たとえば，我々のシステムを，[Mori 93] で提案されているメッセージング機構を使って実現することができる．

5. まとめと今後の課題

我々は，本論文で移動エージェントを使った機械加工システムの基本コンセプトについて述べた．また，コンセプトだけではなく，提案したソフトウェアアーキテクチャに基づいて実際のシステムを構築して評価を行なった．以上の議論から，我々の提案しているコンセプトとアーキテクチャがフレキシブルな生産システムを構築する上で有用であることを示した．

しかし，今後の課題としていくつかの解決すべき問題がある．今回開発したCMUでのプロトタイプシステムでは，システムの独立したActivityを表現するためにエージェントを使っている．しかし本来のエージェントの特徴である知性や交渉力を有効には使っていない．たとえば，製品エージェントが目的を満たすために最適な経路や機械を選択できる方がよい．また，他のエージェントと交渉して，共有資源のデッドロックを回避しながら [櫻庭 00]，生産効率を最大にして，すべての製品が納期に間に合うようにできればよい．このようなことを実現してシステムの柔軟性を増すために，エージェントのインテリジェンスを向上させる必要がある．

さらに，我々はコンセプトを小さな実験システムで検証したが，実システムで実用になるかどうかを評価するために，提案したアーキテクチャをもっと大きなシステムに適用して，検証する必要がある．

なお，本文中の各商標はそれぞれ各社の商標である．

謝辞

本研究を進める機会を与え，また有益なコメントをいただいた永島 晃氏，星 哲夫氏をはじめITプロジェクトセンターエージェントグループのメンバに感謝致します．

また，プロトタイプシステム作成にあたり多くの御協力をいただいたCMUのLee Weiss博士とSDM Lab.のメンバにも深く感謝致します．

◇参考文献◇

[Aglets Homepage] Aglets Software Development Kit Homepage: http://www.trl.ibm.co.jp/aglets/index.html.

[Bussmann 98] S. Bussmann: An Agent-Oriented Architecture for Holonic Manufacturing Control, Proc. of the First International Workshop on IMS, pp.1–12 (1998).

[Hirano 97] S. Hirano: HORB: Distributed Execution of Java Programs, Worldwide Computing and Its Applications'97, Springer Lecture Notes in Computer Science 1274, pp.29–42 (1997).

[HORB Homepage] HORB Homepage: http://ring.etl.go.jp/openlab/horb.

[星 98] 星 哲夫: エージェントによる次世代生産システムの課題と展望, 人工知能学会研究会資料, SIG-J-9801-2, pp.5–10 (1998).

[Ihara 84] H. Ihara and K. Mori: Autonomous Decentralized Computer Control System, IEEE Computer, vol.17, no.8, pp.57-66 (1984).

[Kriz 95] D. Kriz: Holonic Manufacturing Systems: Case Study of an IMS Consortium, http://hms.ifw.uni-hannover.de/ (1995).

[Lange 98] D. Lange and M.Oshima: Programming and Deploying Java Mobil Agents with Aglets, Addison-Wesley (1998).

[牧野 97] 牧野 洋, 新井民夫: 自動組立システムの動向, 精密工学会誌, vol.63, no.11, pp.1503–1509 (1997).

[Mills 91] D.L. Mills: Internet Time Synchronization: the Network Time Protocol, IEEE Trans. Communications, vol.COM-39, no.10, pp.1482–1493 (1991).

[Mori 93] K. Mori: Autonomous Decentralized Systems:

Concept, Data Field Architecutre and Future Trends, Proc. of the 1st Internatinal Symposium on Autonomous Decentralized Systems, pp.28–34 (1993).

[Mori 98] K. Mori: Application in Rapidly Changing Environment, IEEE Computer, vol.31, no.4, pp.42–44(1998).

[Parunak 85] H.V.D. Parunak, B. Irish, J. Kindrich, and P. Lozo: Fractal Actors for Distributed Manufacturing Control, Proc. of the 2nd Conference on AI Applicaions, pp.653–660 (1985).

[櫻庭 00] 櫻庭祐一, 久保和也, 和田英彦, 星 哲夫: 移動エージェントを用いた分散システムのデッドロックの検出, 計測自動制御学会自律分散システムシンポジウム, pp.63–68 (2000).

[SDM Homepage] Shape Deposition Manufacturing Laboratory Homepage: http://www.cs.cmu.edu/~sdm.

[Shen 99] W. Shen and D.H.Norrie: Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey, Knowledge and Information Systems, an International Journal, vol.1, no.2, pp.129-156 (1999).

[Wada 99] H. Wada, Y. Sakuraba, M. Negishi, T. Yamakawa, K. Kubo, and Y. Kashiyama: A Machinery Control System Using Mobile Agents, Proc. of the 4th International Symposium on Autonomous Decentralized Systems, pp.124–131 (1999).

2000 年 5 月 9 日　受理

著 者 紹 介

和田　英彦

1984 年早稲田大学理工学部電気工学科卒業. 1986 年早稲田大学大学院理工学研究科電気工学専攻前期課程修了. 同年横河北辰電機 (株)(現在は横河電機 (株)) 入社. 現在, 同社 IT プロジェクトセンター所属. スケジューリングアルゴリズム, 分散システム, リアルタイムシステム, エージェントの生産システムへの適用などの研究開発に従事. 情報処理学会, 電子情報通信学会各会員.