グループウェア 20－5
（1996. 10. 3）

ワークフローシステム Flexflow のエージェントによる実装

中川路 充*、小笠原 章夫†、
松井 馨一郎*、垂水 浩幸†

*NEC ソフトウェア神戸　　†NEC

NEC のワークフローシステム Flexflow の実装方式について述べる。Flexflow は電子メールに乗って移動するエージェントを用いて実装されている。エージェントシステムとしての特徴には、OLE オートメーションとの連携、永続プロセスを Windows 上で実現したこと、黒板を用いたエージェント間通信等がある。また、ワークフローシステムとしての特徴には、着信・送信などのイベント駆動によるルール実行、フローの部品化、分岐したフローの合流機能、封筒メタファ等がある。

Flexflow --- A Workflow System Implemented with Agents

Mitsuru Nakakawaji*, Fumio Ogasahara†,
Kaichiro Matsui*, and Hiroyuki Tarumi†

* NEC Software Kobe, Ltd.　　†NEC Corp.

Flexflow is NEC's workflow system implemented with mobile agents transported by e-mail. Its remarkable features as an agent system include accessibility from the OLE automation, persistent processes on the Windows, and agent communication by a blackboard system. Its features as a workflow system include event-driven rule execution at the time of receiving or sending agents, workflow parts, joining flows, and the envelope metaphor.

1.はじめに

我々が開発しているFlexflowは、Microsoft Windows[1]上でLotus Notesと連携して動作し、以下の点を特徴とするワークフロー管理システムである。

- ワークフロー中に流れる帳票をネットワークエージェントとして実装することにより、以下のようなメリットを得ている。
  - 自動処理プログラムの転送／インストールが不要（自動的に行われる）。
  - 自動処理プログラムを自発的に起動させることができる。
- ワークフローの一部を部品として動的に展開する機能を持つ。
- エンドユーザがワークフローに従って流れてきた帳票に対して拒絶・回送ができる。
- 帳票にはOLE複合文書オブジェクトやOLEカスタムコントロールを貼り付けることができる。
- ワークフロー中に分岐･合流を含むことができ、合流においては合流する帳票が揃うまで待ち合わせすることができ、さらに合流した帳票を封筒にまとめることができる。

本稿では、Flexflowの実装の基盤となっているエージェントシステムについて述べ、次に実装している機能の特徴について説明し、最後に他のシステムとの比較検討を行う。

2.エージェントの実装

2.1.め組エージェント

Felxflowのワークフローシステムは、内部にエージェントを実装し、そのエージェントが活動することにより構成されている。このエージェントをめ組エージェント[2]と呼ぶ。ワークフロー上を流れる帳票もめ組エージェントとして実装される。

め組エージェントは、いわゆる移動型エージェント（ソフトウェアロボット）であり、電子メールシステムを通してシステム間を渡り歩く。また、システムのシャットダウンを越えて生き残る。

め組エージェントはユーザ毎に用意されるめ組エージェントプール上で活動する。め組エージェントは、め組エージェントプール間を電子メールに乗って移動する。この様子を図1に示す。

め組エージェントは、生まれたときからの使命を遂行する能動的側面と途中でユーザや他のエージェントからの要求を受け入れる受動的側面を持つ。前者としてミッションと呼ばれるプログラムを実行し、後者として公開インタフェースと呼ばれるインタフェースをユーザや他のエージェントに対して提供する。

エージェントは、ミッションに従って行動する。ミッションには他のユーザのめ組エージェントプールへの移動やユーザのアクションを待つための待機などを記述できる。例えば出張処理のワークフローを実行するエージェントの場合、1) 出張者が記入しおわるのを待ち、2) 部の予算管理者のところへ行き、3) 予算進捗率を取得し、4) 部長のところへ行き、... というミッションになる。

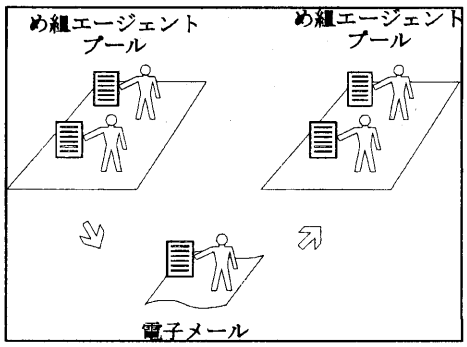

図1め組エージェントの環境

め組エージェントは、受動的にサービスを提供するインタフェースとして、自分自身にアクセスさせるためのオブジェクトを提供する。これを公開インタフェースと呼ぶ。公開インタフェースにアクセスすることにより、め組エージェントのプロパティにアクセスしたり、メソッドを起動したりすることができる。公開インタフェースには、ユーザがGUIを通してアクセス

---

[1] 本稿に記載の会社名・商品名は、一般に各社の商標または登録商標です。

[2] 「め組」はFlexflowの前身となったシステム[垂水95a]の名称であるが、開発中にはこの名称で呼ばれていたため、本稿ではその名前をそのまま用いている。

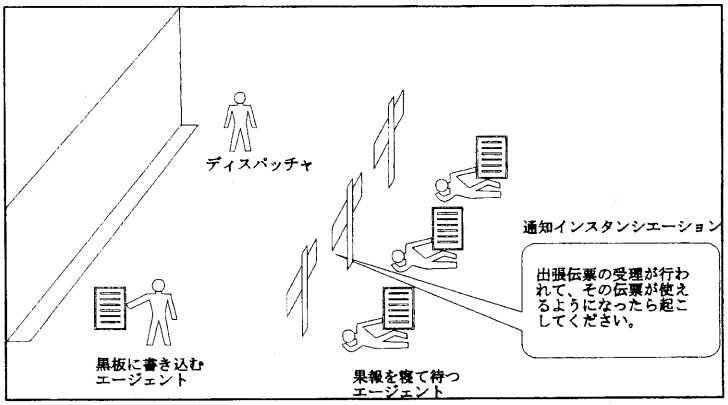

図２黒板システムの動作イメージ

できるだけでなく、他のアプリケーションがOLE オートメーションを使ってアクセスすることができる。これにより、他の OLE オートメーションクライアントとなれるアプリケーション(ex. Excel, MS-Word)からめ組エージェントを直接操作できる。また、一つのめ組エージェントが他のめ組エージェントを自動処理することも可能である。

2.2.ロングライフプロセス

ワークフローシステムを実装する場合、そのプログラムの起動は断続的になりやすい。たとえば、利用者がデータを入力した時に少しだけ動いて、メールの発信時にまた少しだけ動くというように小さなプログラムが何度も起動されることが多い。このとき、それらの起動毎のコンテキストをそのメールの状態遷移として明示的にファイルに保存する必要があり、プログラミングを難しくしている。

Flexflow ではこの問題を解決するためにロングライフプロセスという永続プロセス機構を提供している。そして、前述のミッションはロングライフプロセスとして記述される。

ロングライフプロセスでは、ll-sleep という関数を呼ぶことによって、現在の実行点のコンテキスト情報をファイルに保存できる。ll-sleep が呼ばれると、コンテキスト情報を保存後、そのプロセスは一旦終了するが、後で、ll-sleep を呼んだ場所から処理を再開することができる。したがって、ロングライフプロセスは、寝たり起きたりしながら処理を進めていく寿命が長いプロセスとみなすことができる。ロングライフプロセスは、シャットダウンやマシン間の移動を越えて生き残れるプロセスである。

2.3.黒板

め組エージェント間の通信は黒板を経由して行われる。黒板は各め組エージェントプールに一つずつ設置される。Flexflow の黒板は、黒板に置かれる情報を表現する「トークン」とその情報に反応して特定の処理を起動する「通知インスタンシエーション」の集合からなりる。つまり、二つのめ組エージェント間で通信を行う場合は、送信側が黒板にトークンを置き、受信側が黒板に通知インスタンシエーションを置くことになる。通知インスタンシエーションがトークンに反応して発火すると、受信側に発火したことが通知され、受信側が情報を受け取ることができる。

通常、め組エージェントは前述の「ロングライフプロセス」の ll-sleep を使って興味のある情報が入るまで、寝て待つ。送信側のエージェントがトークンを黒板に書き込むと黒板システム内のディスパッチャがそのトークンと登録されている通知インスタンシエーションを照合し、マッチするものがあれば、そのエージェントを起こす。このイメージを図２に示す。

以下の節では、トークンと通知インスタンシエーションについて説明し、その後、このような黒板システムによるエージェント間通信の特長について述べる。

2.3.1.トークン

トークンはLispのS式である。つまり、黒板に置きたい情報をシンボル、文字列、数値などのアトムをConsセルで構造化して表現したものである。トークンがLispのS式であることにより、簡単に印字表現（文字列による表現）を得ることができ、また印字表現から構造化された情報を復元することができる。たとえば、出張命令書が着信したことを表現するトークン以下のようなものとなる。

```
(:InformMailEvent :Receive ((:From "中川路") (:Form "出張命令書")))
```

2.3.2.通知インスタンシエーション

通知インスタンシエーションはif-then形式のルールである。そのif部はマッチング述語のリストから、then部は一個のアクション記述からなる。Flexflowの黒板システムでは、マッチング述語とアクション記述としていくつかの種類をサポートしているが、本稿では、簡単のため:ExclusiveMatchマッチング述語と:Goアクション記述のみを紹介する。

:ExclusiveMatchマッチング述語はトークンとの照合を行うための述語である。:ExclusiveMatchマッチング述語では引数にトークンと照合するためのパターンを指定する。パターンにはルール変数と呼ばれる変数を含むことができる。このパターンとトークンの照合は、Prologなどと同様の単一化(Unification)アルゴリズムによって行われ、単一化の結果、ルール変数の値が定まる。ここで確定したルール変数の値をアクション記述から参照したり、後でエージェントが自分の受信情報として参照することができる。:ExclusiveMatchマッチング述語とマッチングに成功したトークンは排他的に束縛され、他のインスタンシエーションとの照合には利用できなくなる。

通知インスタンシエーションのif部のすべてのマッチング述語について照合に成功するとその通知インスタンシエーションは発火する。発火した通知インスタンシエーションのthen部が:Goアクション記述である場合、黒板は引数に指定されたエージェントを起こす。起こされたエージェントは自分を起こした通知インスタンシエーションを識別することができ、その通知インスタンシエーションのルール変数の値を調べることができる。これを利用してエージェントは起動のパラメータを受け取る。

以下に:ExclusiveMatchと:Goアクション記述で構成される通知インスタンシエーションの例を示す。

```
If 部 :  ((:ExclusiveMatch ?TokenID
(:InformMailEvent :Receive ?Properties
)))
Then 部: (:Go
"c:¥¥Flexflow¥¥Data¥¥Mitsuru¥¥xxxx.
mga")
```

上記の例では、前節のトークンの例で示したトークンとの照合が成功する。このとき、?TokenIDにはマッチングに成功したトークンのIDが入り、?Propertiesには、((:From "中川路") (:Form "出張命令書"))というS式が入る。Then部には:Goアクション記述が記述されている。引数に指定されているのは、起こす対象となるエージェントのファイル名である。[1]

Flexflowの黒板によるエージェント間通信には以下の特長がある。

- 通信にLispのS式を使っているので、構造化された情報をやりとりできる。
- 通知インスタンシエーションではパターンマッチングを記述できるため、抽象的な表現が可能である。
- トークンと通知インスタンシエーションのどちらが先に登録されても、照合の結果が同一なので、エージェントの到着順序(受信側が先か、着信側が先か）に依存しない通信が可能。
- トークンを排他的に束縛できるので、処理の排他制御が可能。
- 通知インスタンシエーションに優先度が設定できるので、優先して起動する処理の設定が可能。

3.ワークフローの実装

前章で述べたエージェントシステムは汎用に設計されている。これを基盤層と呼んでいる。Flexflowでは、基盤層の機能を用いてワークフロー制御を行うため、さまざまな応用機能を用意している。これをワークフロー層と呼んでいる。この章では、ワークフロー層の機能として

[1] 説明を簡単にするため引数などを省いているので、実際のFlexflowの通知インスタンシエーションの形式は若干異なる。

のルール、フロー部品化、分岐・合流の機能がいかに実装されているかを示す。

3.1.ルール機能

Flexflow でのルール機能とは、ワークフローの定義者が特定のノードで実行する作業のうち自動化できるものをあらかじめ設定して置くことにより、ワークフローの実行時に自動的に作業が行われワークフロー実行者の負担を軽減するための機能である。

具体的な例としては、特定のノードで送信時に必要な項目への記入がなされているかのチェック、アンケートの集計ノードでの回答アンケートの集計作業等が考えられる。

ルール機能の実装は、特定のノードとタイミングの組み合わせに対するエージェントの動作についての定義機能と実行機能とからなる。ここでタイミングとは、エージェントのノードへの着信、拒絶、回送、送信といったエージェントの移動に関するものや、エージェントが持つボタンやテキストボックスなどの GUI 部品経由で発生するイベントの総称である。

ワークフローの定義者は、各ノード上で存在するエージェントに対してルールを設定することができる。設定内容は、新エージェントの生成、エージェント上の OLE オブジェクト間の値の転記や内容のチェック等である。

アンケートの例で送信時チェックの設定を行う場合、定義者はチェックを行うノードと当該エージェントを指定し、このエージェントの送信タイミングに対して、必須項目に相当する OLE オブジェクトの内容が空である場合に警告メッセージを出して送信を中断するルールを設定する。上記の定義が行われているワークフローでは、設定ノードでの実行者が必須項目に値を入れないで送信した時には送信時ルールによって警告がなされる。

3.2.フローの部品化

ワークフローを定義する際にフロー内の特定の部分を実行時に決定したい場合がある。例えば部門にまたがったワークフローの場合に他部門での作業の流れを当該部門で決定したい場合などである。

例として、部門をまたがるアンケートや稟議を行うシナリオ[1]において、業務の流れが個々の部門毎に異なる場合が考えられる。このような場合にはシナリオ上では回答部門・関連部門に当たる部分を抽象化(例: “部門内部長承認” という名前のみ指定)しておき、具体的なフローは各部門の窓口へ業務が渡されてから部門内のフロー部品を適用するというやり方でシナリオ定義の効率化を図ることが可能である。

部品化を行う場合、シナリオ以外に各部門の部品が定義されている必要があるが、この部品自体は他のシナリオで共用することが可能なのでシナリオ定義の負担を軽減できる。また、部門内での業務分担の変更があった場合は部品の手直しのみでシナリオの定義を修正する必要がないという利点もある。

Flexflow での部品の実装は以下のようになっている。各部門で部品を用意し、これを部門の環境サーバに登録する。各部門の窓口ノードでは、実行時にワークフローが到着した時点で部品を検索し、シナリオにダイナミックにバインディングする。(詳しくは[垂水 95b]も参照していただきたい。)

3.3.分岐と合流

Flexflow では一つのノードから複数のアークが出ている場合、フローはそのノードから分岐しているとみなす。Flexflow では、分岐しているアークすべてにエージェントを流す全分岐と、分岐しているアークのいずれか一つにエージェントを流す択一分岐をおこなうことができる。

全分岐では、一つのエージェントを分岐しているすべてのアークにコピーして流すことも可能であるが、複数のエージェントを別々に異なるアークに流すこともできる。

一方、択一分岐は、送り先の経路を実行時に選択して、エージェントを一つのアークに流す。経路を選択する方法は、エージェントの持つドキュメントのフィールドの値等によってエージェントが自動的に選択する方法と、ユーザが任意に選択する方法がある。

分岐とは逆に、複数の異なるアークが一つのノードに合流するフローが存在する場合、それ

[1] Flexflow では、ワークフロー定義のことをシナリオと呼んでいる。”ミッション”はシナリオ定義を基にして生成されるプログラムである。

ぞれのアークを流れてくるエージェントをそのノードで待ち合わせることができる。

待ち合わせは、すべてのアークからエージェントが着信した時に完了し、その際任意のプログラム(例えばアンケート集計プログラム)を実行することができる。待ち合わせの完了は、ユーザが指定することも可能であり、将来的には期限を指定して待ち合わせを完了することも検討している。

択一分岐を行ったフローの場合、同じノードから別れたすべてのフロー同士のみの合流をおこなうことができる。この場合、実際に複数のエージェントが合流するわけではなく、エージェントはただ一つしか流れない。この合流手法は、択一分岐によって別れたそれぞれのフローが再び同じフローを利用する場合に有効である。

待ち合わせたそれぞれのエージェントは、封筒エージェントによって一つにまとめられ、次のノードに送ることができる。

封筒エージェントは封筒のメタファとして、複数のエージェントをその中に格納した状態で別のノードに送信を行ったり、保存しておくことができる。封筒エージェントの中に格納しているエージェントは、「封筒を開封してとりだす」というインタフェースによって簡単に参照することができる。

封筒エージェントは待ち合わせをおこなったエージェントを集めるだけでなく、定義されたフローには含まれていない人とアドホックにエージェントをやりとりする場合にも使うことができる。封筒に入れている間は封筒内のめ組エージェントのミッションは凍結状態にあり、この間にアドホックな処理を行ったあとで元のフローに戻すことも可能である。

4.比較・考察

移動エージェントとしては Telescript [Telescript]などが著名であるが、Flexflow のめ組エージェントは Windows における OLE との連携が強化されている点が特徴になっている。

なお、め組エージェントは電子メールによって移動するが、現在のところ Lotus Notes のメールシステムを利用している。将来は他のメールシステムでも利用可能にしたいと考えている。また、データストアに関しても、Notes 以外のシステムとの連携が将来は可能である。

ワークフローを移動エージェントで実装する場合、トラッキングの手法が問題となるが、め組エージェントがイベント発生の度に指定された場所にログを出力することで対応する。

次に他のワークフローシステムとの比較であるが、Flexflow はフローの部品化によるダイナミックバインディングと、合流機能に特徴がある。例えば Staffware [戸叶 95]にも「同期」という合流機能があるが、本稿でいう全分岐に対する合流機能はない。

将来的には個人のスケジュール管理と連携したワークフローの動的計画機能[垂水 96]も導入する予定である。

参考文献

[垂水 95a] 垂水、田淵、吉府: ルールベースの電子メールによるワークフローの実現、情報処理学会論文誌、Vol.36, No.6, pp.1322-1331 (1995)

[垂水 95b] 垂水、金政、小笠原: ワークフロー技術とその応用、計測と制御、Vol.34, No.12, pp.932-936 (1995)

[Telescript] http://www.genmagic.com/Telescript

[戸叶 95] 戸叶: スタッフウェア、UNISYS TECHNOLOGY REVIEW, No.45, pp.35-47 (May 1995)

[垂水 96] 垂水、石黒、田淵、吉府、喜田、朝倉: ワークウェブシステムの実現、情報処理学会グループウェア研究会、GW-15-22 (1996)