ヒューマンインタフェース 55－9
グループウェア 7－9
（1994．7．15）

エージェントメールシステムのワークフロー制御への適用

松尾 朗　服部 真穂　橋本圭介　貫井春美

株式会社 東芝 研究開発センター
システム・ソフトウェア生産技術研究所

ソフトウェア開発は、複数の組織、複数の開発者の分業による共同開発が一般的である。このような形態での開発を支援するためには、複数の開発者が連携して業務を行う場合の手順(ワークフロー)を支援する仕組みが必要となってくる。

筆者らは、ソフトウェア共同開発支援システム D^2 (D-square:Distributed environment for software Distributed development) の中で開発したエージェントメールシステムのワークフロー制御への適用を試みた。適用にあたり、共同開発における共同作業の分析・モデル化を行い、そのモデルに沿った支援システムをエージェントメールをベースにして構築し、ワークフローの制御が可能であることを確認した。

本稿では、エージェントメールシステムの概要と、ワークフロー制御への適用可能性について述べ、その適用事例を紹介し、最後にワークフロー制御システムに関して考察する。

WorkFlow Control using Agent Mail System

Akira Matsuo,Maho Hattori,Keisuke Hashimoto,Harumi Nukui

Systems & Software Engineering Lab., R&D Center, Toshiba Corp.

For cooperative software development, it is important to coordinate the tasks of several developers. In D^2 (D-square:Distributed environment for software Distributed development), we consider whether it is possible to provide this coordination using the Agent Mail System, and try to apply WorkFlow Control (to coordinate several tasks) using the Agent Mail System to cooperative software development.

In this paper, we describe the following.

- Summary of the Agent Mail System, and the possibility of using it for WorkFlow control.
- An example of WorkFlow control using the Agent Mail System.
- Our notion of WorkFlow Control System.

1 はじめに

ソフトウェアの分散開発支援環境 D²† は、コミュニケーション支援を中心としたソフトウェア共同開発支援システムである。[1] D²の中で、非同期コミュニケーションを支援するエージェントメールシステム[2]を開発し、ソフトウェア開発を中心にこれを適用している。

エージェントメールシステムは、個人毎に記述したルール(エージェントスクリプト)に従って電子メールメッセージの自動処理を行うシステムである。

これは、エージェントスクリプトにより仕事の流れを記述することにより、個人の作業手順を制御することが可能であるので、一つのワークフロー制御と捉えることができる。

共同開発を支援するには、複数の作業者が連係して仕事を進める場合の手順、すなわちワークフローをサポートすることが重要である。

筆者らは、エージェントメールをベースにしたワークフロー制御システムを開発し、ソフトウェアの共同開発へ適用した。適用にあたり、適用対象部門における作業分析を行い、ワークフローを抽出し、作業モデルを作成した。エージェントのスクリプト(処理ルール)は抽出したフローに沿って記述し、支援システムを構築した。[3]

本稿では、エージェントメールシステムの概要とそのワークフロー制御への適用可能性について述べ、その適用事例について紹介する。最後に、ワークフロー制御への適用に関する考察と、ワークフロー制御システムへの拡張について述べる。

2 エージェントメールについて

2.1 システム概要

エージェントメールシステムは、電子メールによるメッセージ交換の際に発生する種々の処理をエージェントというユーザの代理プロセスが、自動処理するシステムである。

ここでは、エージェントを“ユーザの代わりに仕事を行うサービス機能”と定義しており、「代理人」や「行為人」といった意味で捉えている。

メッセージの着信時には、着信メッセージを発信者やサブジェクトによって分類したり、他ユーザへの転送、発信者への返信、メッセージデータ自身の加工や指定アプリケーションの起動などを行うことができる。またメッセージ発信時にも同様に、発信するメッセージの属性に応じて、ある決まった処理を送信データに施した後、発信するという作業も可能である。

システム構成を図1に示す。

どのようなメッセージに対してどのような処理を行うかは、その処理ルールを個人毎に持ち、各ユーザがカスタマイズを行う。メッセージの発着時にはエージェントがそのルールに沿った処理をバックグランドで行う。

現在エージェントメールシステムはUNIX電子メールをベースに構築している。

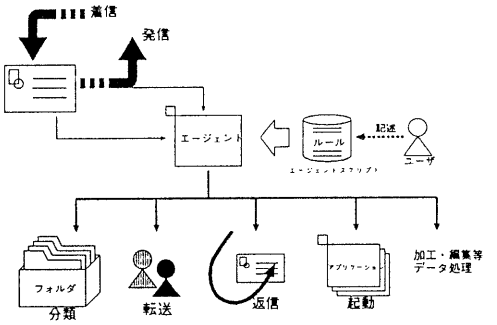

図 1: エージェントメールシステムの構成

2.2 ルール記述

メッセージの処理ルールは、エージェントスクリプトという専用の記述言語を用いて記述する。エージェントスクリプトは、基本的にIF-THEN形式で表現し、C言語ライクな文法を持つ。記述の中心はIF-THEN文になるが、その他処理によってはループ文やジャンプ文、変数の扱い、サブルーチンなどの記述が可能である。

IF文に処理メッセージの条件(誰から来た(または誰へ送る)メッセージか、サブジェクトに含まれるキーワードは何かなど)を記述し、THEN文にそのメッセージに対する処理内容(フォルダーに保管する、他ユーザに転送する、返信するなど)を記述する。

記述例(ルールの一部)を図2に示す。

```
if {
        from("matsuo") && subject("agentmail")
}
then {
        add_folder("+myjob");
        fow_mail("nukui,hattori,hasimoto");
        if {
                access_file("~/reply_file")
        }
        then {
                rep_mail("~/reply_file");
        }
}
```

図 2: スクリプト記述例

この記述例のルールは、

- 差出人が“matsuo”で、
- サブジェクトに“agentmail”というキーワードを含む

†D-square:Distributed environment for software Distributed development

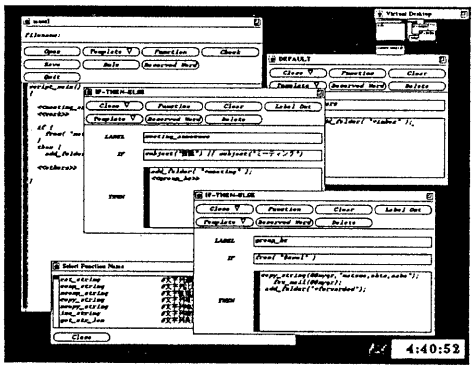

図 3: スクリプトエディタ

メッセージを受信した場合に、

- フォルダー “myjob” に保存し、
- “nukui”,“hattori”,“hasimoto” へ転送 (同報) する。
- さらに “~/reply_file” というファイルが存在すれば、そのファイルをメッセージとして差出人に送り返す。

という処理を行うための記述である。

ルール中の、**from**(メッセージの差出人をチェックする)、**subject**(メッセージのサブジェクトをチェックする)や、**add_folder**(メッセージを指定した自分のフォルダにしまう)、**fow_mail** (メッセージを指定したユーザに転送 (同報) する) などのような、メッセージ操作のための機能をエージェントのライブラリとして提供している。

さらに細かい処理が必要な場合は、文字列の検索、比較、置換などのテキスト編集の機能や、ファイルの作成、削除、行単位の編集などのプリミティブな機能のライブラリも用意している。

図3はルールを記述するためのエディタで、テンプレート形式で条件/処理を埋めてルールを書くことができる。デフォルトはどの条件にもマッチしない場合のメッセージの処理を記述するものである。また、エディタ上でスクリプトのシンタックスチェックを行うことも可能である。

3 ワークフロー制御への適用可能性

次に、エージェントメールシステムのワークフロー制御への適用可能性について検討する。

ワークフロー制御の機能要件として以下があげられる。

(1) 仕事の流れが記述できる。
  (a) 仕事全体のフロー
  (b) 個人単位のフロー

(2) データやメッセージを流すことができる。

(3) 各ジョブの終了を知ることができる。

(4) フローの流れを追跡し、状態を知ることができる。

この機能要件の中で、エージェントメールシステムによりどの程度サポートできるかについて検討する。

(1) 仕事の流れの記述に関して

エージェントメールシステムは、本来個人単位の処理手順の記述により自動処理を行うもので、(b) の個人単位のフローの記述は可能である。しかし、複数の人間の間の業務手順を記述する機能を持っていないため、(a) の仕事全体のフローを記述することはできない。

そこで図4のように、データ・メッセージの流れる各ノードに、共通な着信用/発信用のルールを各ユーザのスクリプトに追加する。

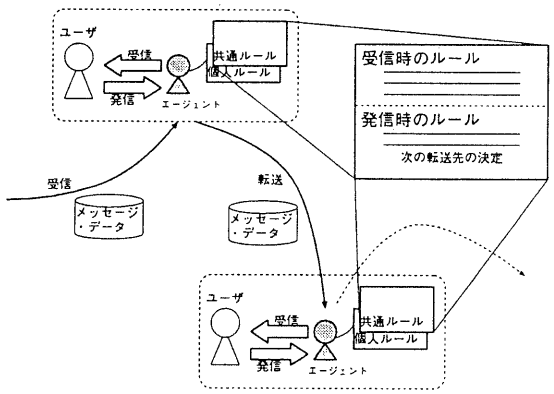

図 4: エージェントによるワークフロー制御

この共通ルールには、ルールの先頭にワークフローに沿って流すべきデータ・メッセージの属性を判別する条件が記述してあり、該当データの受信/発信時にはこの共通ルールが適用される。

共通ルールの処理内容は以下のようになる。

- 受信時の処理
  受け取ったデータの分類や必要ならば加工を行い、ユーザの目的にあった形式でデータを提供する。データの属性により、ユーザによる判断が不要で、定型的な処理が可能なものならば自動で処理を行い、続けて発信処理を行う場合もある。

- 発信時の処理
  データ・メッセージの属性を判断し、次の転送先 (ノード) を決定し発信する。次のノードはデータにより、固定になる場合や、転送ルートの情報がデータと一緒に送られ、そこから決定する場合がある。

ワークフローに関係のないデータ・メッセージの処理に

ついては、従来通りの各ユーザでカスタマイズしたルールが適用される。

(2) データの転送に関して

エージェントメールの持つ、自動転送、自動返送の機能を使用してデータ・メッセージの流れの制御が可能である。ただし、転送の場合の転送先は (1) の発信時のルールから決定する。

(3)、(4) に関しては、エージェントメールの機能では実現できないため、運用でカバーすることになるが、各ノードのエージェントが処理の履歴を記録し、その履歴を参照するなどの方法が考えられる。

以上より、前述の要件のすべてを完全に満たすことはできないが、複数のノード間のデータ・メッセージのフローを記述し、その間を円滑にデータ・メッセージを流すことが可能となるため、ワークフローに沿った制御ができるようになる。

また、電子メールベースのワークフローでは電子メールメッセージとして流れるデータに属性を持たせること (メッセージの構造化) が、メッセージの自動処理には必要であるが、エージェントメールシステムでは UNIX メールのヘッダを拡張してそれを実現している。

4 適用事例

ソフトウェア共同開発でのワークフロー制御へのエージェントメールの適用事例について述べる。

まず、対象業務に対して問題点を抽出し、支援範囲の決定、作業分析により支援対象業務のワークフロー化、支援システムの構築/適用というアプローチをとった。

4.1 問題点抽出

適用対象のソフトウェアの共同開発は、拠点が物理的に分散した形態である (図5)。ソフトウェアのシステム設計が終了すると、図のようにシステムをサブシステムに分割し、各開発拠点に分担して開発を行う。それぞれの開発拠点内でサブシステムの設計/製造を行い、単体テストが終了するとサブシステムを1箇所に持ちより組合せテストを行うという形態である。

ソースコードやドキュメント、ライブラリ等の成果物は各開発拠点で複製として保管されている。

現状の問題点をあげると以下のようになる。

(1) 拠点間では多くの成果物を複製として共有しており、それぞれの開発者がバージョンアップを繰り返す間に共有物のフェーズのずれが発生したり、仕様の変更の連絡が行き届かないまま作業を進めてしまうことがある。

(2) 拠点が物理的に分散しているため、DR や WT の間隔が長くなり仕様に対するフェーズのずれが大きくなる。

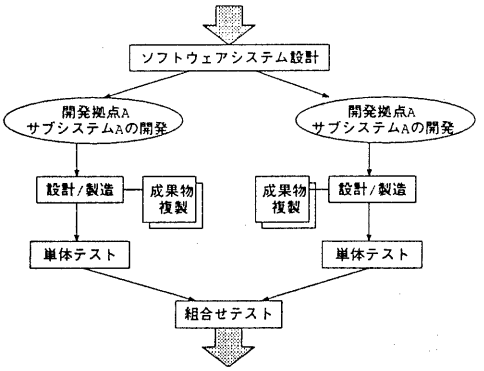

図 5: 共同開発形態

(3) 組合せテストや、実機を用いた総合テストの段階で (1)、(2) で述べた仕様のずれが影響し、サブシステム単位での成果物のインタフェースが合わなくなってしまう場合がある。

すなわち、共同開発のある拠点で、仕様変更によって成果物のバージョンアップが行われると、拠点間でフェーズのずれが発生し、それを吸収できないまま DR やテスト工程に入ることになり、そこで工程の後戻りが生じてしまうことになる。

このような拠点間のフェーズのずれ、およびそれによる後戻り作業の問題を解決するには、以下のように開発途中の成果物のバージョンを統一しながら、開発を進めることが重要である。

- 成果物を一元管理し、拠点間の成果物のフェーズを揃える。
- 修正や変更作業と同期をとって、各拠点 (作業者) へ通知を行い、連絡洩れを防ぐ。
- 変更履歴を管理し、修正や変更に関する情報を明確にする。

4.2 共同開発の作業分析

まず、修正や変更はどのようなトリガで発生し、修正の同期はどのようにとり、修正後の成果物はどのように管理されるかを整理していく。すなわち、それぞれの作業間の関係を明示的に表現するワークフローの考え方に基づき、修正・変更や、その他の理由により発生するバージョンアップに伴う作業の流れをパターン化し、作業モデルを作成する。

このように、共同開発作業をワークフローとして捉え、作業手順を明確化、パターン化し計算機による定型的処理として支援する。

対象業務は上記問題点の原因となる仕様変更の作業とし、これについて分析を行うことにした。
作業分析は次の4点に着目して行った。

(1) 修正を行うトリガとなる要因は何か

(2) 修正した成果物に関係するユーザにはどのような手順で通知を行うか

(3) 修正した成果物はどのような手順で登録を行うか

(4) 修正した成果物に関連する成果物はあるか。ある場合にはどのような手順で修正するか

ここから作業手順の基本パターンを抽出し、整理した(表1)。

1.修正のトリガ	(a)自発的に修正を行う (b)他からの連絡により差し替えを行う (c)他からの指示により修正を行う
2.通知手順	(a)修正に対するテストを行った後で通知する (b)修正に対するテストを行う前に通知する
3.登録手順	(a)修正に対するテストを行った後で登録する (b)差し替えだけのためテストを行わずに登録する
4.関連成果物の修正手順	(a)テストを行った後で関連する成果物の修正を行う (b)テストを行う前に関連する成果物の修正を行う

表 1: 修正に伴う作業の基本パターン

これらの基本パターンの組合せにより数パターンの仕様変更作業のワークフローが表現できる。実際の作業で発生する基本パターンの組合せによるワークフローを表2に示す。

これらの基本パターンを組み合わせた仕様変更時の作業フローの例を図6に示す。これは表2のワークフロー1にあたるもので、この例における手順は以下に述べる通りであり、[] 内の文字で表された基本パターンから構成される。

(1) 開発者が自分の拠点内で保持しているマスタファイルからソースコードを取り出し、修正を行う。[1-(a)]

(2) テスト確認後、マスタファイルに登録を行う。[3-(a)]

(3) テスト確認と同時に開発拠点に通知する。[2-(a)]

次に、抽出した基本パターンに基づいてこのソフトウェア共同開発における、ソースコードの仕様変更作業まわりの開発手順をモデル化した(図7)。このモデルは、表2中の1、8、9の3パターンのフローを含む。モデル化したフローに従って、コンピュータによる支援システムを構築する。

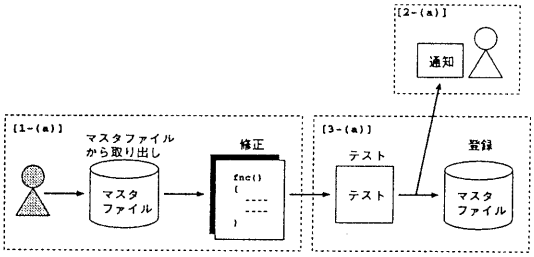

図 6: フローの組合せ例

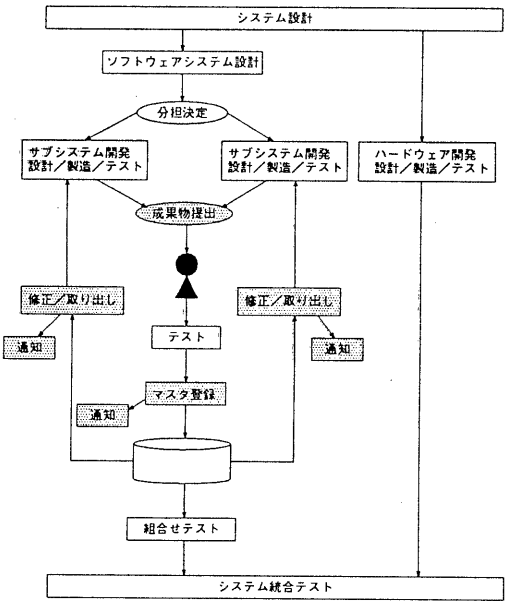

図 7: 開発モデル

4.3 支援システムの構築

4.3.1 機能要件

支援システムの機能要件として以下をあげる。

(1) 仕様変更と同期をとって通知させる。
仕様変更作業は他からのバージョンアップの指示に従って行う場合や、バグの発見によりその追跡過程で行われる場合など様々なタイミングが考えられる。

修正のトリガ	ワークフロー	パターン数
自発的修正	1=1(a)→2(a)→3(a) 2=1(a)→2(b)→3(a)	2通り
他からの連絡による修正	3=1(b)→2(b)→3(b) 4=1(b)→2(b)→3(b)→4(b)→2(a)→3(a) 5=1(b)→2(b)→3(b)→4(b)→2(b)→3(a) 6=1(b)→3(b)→4(b)→2(a)→3(a) 7=1(b)→3(b)→4(b)→2(b)→3(a)	5通り
他からの指示による修正	8=1(c)→2(a)→3(a) 9=1(c)→2(a)→3(a)→4(a)→2(a)→3(a) 10=1(c)→2(a)→3(a)→4(a)→2(b)→3(a) 11=1(c)→2(a)→3(a)→4(b)→2(a)→3(a) 12=1(c)→2(a)→3(a)→4(b)→2(b)→3(a) 13=1(c)→2(b)→3(a) 14=1(c)→2(b)→3(a)→4(a)→2(a)→3(a) 15=1(c)→2(b)→3(a)→4(a)→2(b)→3(a) 16=1(c)→2(b)→3(a)→4(b)→2(a)→3(a) 17=1(c)→2(b)→3(a)→4(b)→2(b)→3(a)	10通り

表 2: 修正に伴う作業のワークフロー

このように仕様変更のタイミングは様々で、その都度洩れのないように確実に通知を行う必要がある。

(2) 仕様変更の範囲により通知範囲も変更する。
変更を施した成果物によって影響を受ける他の開発者を割り出すことが可能である。変更した開発者がいちいち誰に通知を出すべきかを修正の度に意識せずに、通知範囲を自動的に決定する必要がある。

(3) 成果物は一元管理し、バージョンを統一する。
成果物自身のバージョンの管理はバージョン管理専用のツールを用いて行う。また統一のため成果物を一元管理し、そのマスタからの取り出し/登録作業を開発者にできるだけ負担(難しいオペレーションなど)をかけないようにする必要がある。またその際、仕様変更の記録を確実に記録することも重要である。

上記機能要件より、バージョン管理自体には既存のバージョン管理ツールを使用する。これらに電子メールのようなメッセージ交換システムをリンクさせ、通知の自動化を図り修正/変更などの作業と同期をとった通知を可能にする。この際作業フローを支援するために、メッセージの制御をワークフローに沿って行わせることが必要となる。

抽出したマスタからの取り出し/登録作業の手順から、それに同期して行うメールによる通知までのワークフローをエージェントスクリプトで記述して自動化を図る。

4.3.2 システム構成

システム構成を図8に示す。ユーザ(作業者)は、成果物に対して修正/変更が生じたときマスタから取り出して、修正/変更作業を行ったのち再び、マスタに対してバージョンアップ登録を行う、といったサイクルで作業を進める。

このシステムでは2つのエージェントをセットしている。マスタを管理しているサーバ用のエージェントと、ユーザ用のエージェントの2種類である。

このシステムでは、"取り出し" および "登録" 作業をサーバ用エージェントに対する "要求メール" という形で発行する。また、取り出したソースコードもメールで送られる。これらのデータやメッセージをエージェントがそれぞれのルールに従い判断し、処理する。

"要求メール" には、要求の種別(取り出し、登録)、取り出しのモード(修正、参照)、取り扱うターゲット(システム)名、などの情報をヘッダとして追加しメール中に埋め込み、属性を持たせている。

[サーバ用エージェント]
サーバ用エージェントは、作業者からのメールを受信すると、サブジェクトや追加されたその他のヘッダ情報より要求内容を解析し、それぞれの要求に従い、スクリプトに記述されたフローに従い処理する。成果物マスタファイルへのアクセスはバージョン管理ツールを通してエージェントが行い、取り出し/登録などを行う。バージョンアップなどが行われると、その成果物ごとに異なる通知先を決定し、通知のメールを送付する。このよう

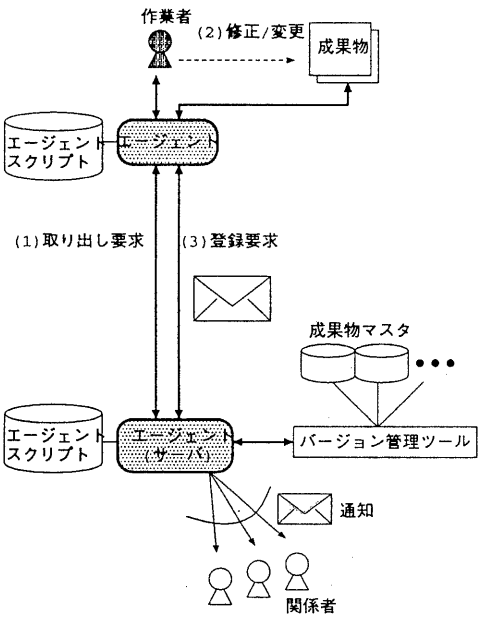

図 8: 支援システムの構成

な作業手順や通知先などの変更にはエージェントスクリプトの変更により柔軟に対応することが可能である。

[ユーザ用エージェント]
一方、ユーザは要求した作業結果 (取り出したソースコード、登録完了通知、他ユーザからの登録通知など) をすべて電子メールとして受け取ることになる。
作業者の手元には作業者用のエージェントが設定されている。デフォルトでは以下のメールの受信/発信時の処理ルールが共通ルールとして与えられ、その他のメールに対する処理ルールはユーザ個人のカスタマイズによって記述される。

- マスタから取り出したソースコードの受信
  取り出したソースコードを受信すると、メール本文の中からデータを取り出し、解凍して各作業者が指定した各自のワークディレクトリの下へ展開する。さらに自分自身にその旨を伝えるメッセージを送信する。作業者はメールボックス中にソースコードのデータを見ることはない。

- 登録用メッセージの発信
  修正後のソースコードをマスタに登録 (バージョンアップ) する場合、登録要求と同時にソースコードを送付するが、登録データをアーカイブしサーバ用エージェントに送付する。作業者はターゲットのシステム名を登録要求メール発信時に指定するだけでよい。

- (自分の) 登録完了の通知の受信
  登録が正常に完了すると、サーバ用エージェントから通知が送られる。その通知を受け取ると作業者のワークディレクトリ内のテンポラリなファイルを削除する。

以上述べたような構成でエージェントメールをベースにして支援システムを構築することができた。現在、このシステムは前述の作業分析の対象とした部門で実運用中である。バージョン管理の自動化や取り出し登録の通知、およびそれに付帯する作業の自動化により、不具合件数の減少、作業効率の向上などの成果を得ている。

5 結果

エージェントメールシステムは、本来ユーザ個人毎の作業手順を記述し、それをサポートするために開発されたシステムである。
今回、ソースコードのバージョン管理に関わる作業に関して作業分析を行い、

- バージョン管理自体の作業を行うセンターのエージェントの作業フローをエージェントスクリプトとして作成

- センタへの取り出し/登録作業を行うためのユーザのエージェントの共通作業フローのエージェントスクリプトを作成して配布

を行った。
これらのエージェントスクリプトによるエージェントの組合せにより、エージェントメールを適用したワークフローの制御が可能であることが確認できた。
この方式には以下のような問題点があげられる。

- 個人のルールの組合せでフロー制御を実現しているため、フロー全体のルールというものがなく、フロー全体を捉えにくい。従ってフローがどこまで進んでいるかなどの追跡を行うのが困難である。

- 各ユーザにルールを配布しているため、ユーザによるカスタマイズ性はあるが、その反面個々のルールの一貫性を保証してやる必要がある。

6 ワークフロー制御システムの考察

前節で述べた問題点の解決のために、D^2で研究を進めているワークフローの制御方式について以下に述べる。ワークフローを制御するためのシステムアーキテクチャとして図9のような、マネージャとエージェントからなるアーキテクチャを考えている。

マネージャは“グローバルルール”により業務全体のフローの制御を行い、各ユーザのノード上ではエージェントが“プライベートルール”とマネージャからの制御により動作する。

マネージャは、各エージェントに対してジョブを依頼し、エージェントは依頼されたジョブが終了すると終了通知をマネージャに返す。マネージャはその通知を受け取ったら、フローを進め次のエージェントに対して次のジョブを依頼する。

ルールの記述は、「Coordination Theory」[4]をベースにし、仕事のフローをシナリオとして定義する。すなわちグローバルルールにおいて、

- goal(仕事の目的、最終の成果物)
- role(フローに登場する役割)
- actor(各役割を担当する人、各役割に割り当てられる)
- activity(分割された仕事の単位、各役割に割り当てられる)
- output(各アクティビティのアウトプット、成果物)
- limit(各アクティビティの締切期限)
- flow(アクティビティの流れ)

を定義する。

フローに関しては、マネージャが管理するのはアクティビティ単位の流れであり、各アクティビティでの詳細の処理手順の制御は、割り当てられた役割の人のエージェントが行う。

各アクティビティの処理手順は、プライベートルールとして、グローバルルールから生成され、その雛型が各エージェントに配布される。プライベートルール中のフローに影響のないローカルな処理手順に関して、各ユーザによるカスタマイズを可能とする。

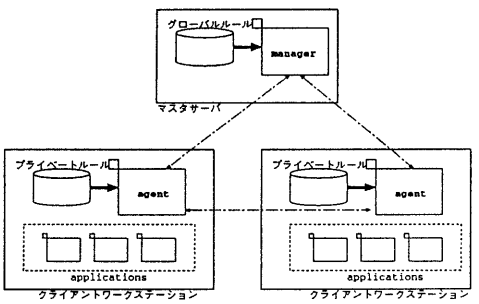

図 9: ワークフローシステム構成

このようにグループ全体の仕事の流れのルールと、個人環境におけるルール、その両方を融合する必要がある。

図中、マネージャ・エージェント、またエージェント同士の間の矢印は、それぞれの間の通信を示しており、ここでは、フローの制御、追跡などを行うため以下のような情報のやりとりを行う。

- ジョブの開始・中止要求/応答
- フロー状態、業務履歴の問い合わせ/結果
- 締切のフォロー
- データ (成果物) の転送

これらの情報の送受信のための専用のワークフロー制御用プロトコルを用意する。

また、ここで述べたエージェントはエージェントメールとは異なり、マネージャ・エージェントとの通信機能や、ルールの記述に関して拡張したものとなる。また電子メールなど特定のアプリケーションに依存しないインタフェースを有する。

7 おわりに

本稿では、D^{2}で開発したエージェントメールシステムの概要と、それを用いたワークフロー制御への適用可能性について述べ、その適用事例として社内でのソフトウェアの共同開発におけるバージョン管理支援システムを紹介した。

エージェントメールをプラットフォームにしたワークフローの制御が可能であることが確認できた。現在はフロー全体を管理するマネージャを導入したワークフロー制御システムへの拡張に関し研究・開発を継続している。ワークフロー制御ではフローをどのように記述するか、その記述性とカスタマイズ性がポイントになると考えている。

参考文献

[1] 貫井 ソフトウェア分散開発支援システム D^2、情報処理学会グループウェア研究グループ 92-GW-3 (1992)

[2] 松尾, 貫井, 中村 電子メールにおけるエージェントシステム, 第 43 回全国大会講演論文集 (5)pp.296-297(1991)

[3] 服部, 松尾, 貫井 エージェントメールを適用したバージョン管理支援システム, 第 47 回全国大会講演論文集 (5)pp.203-204(1993)

[4] Malone,T.W and Crowston,K “Coordination Theory” CSCW90 proceedings,pp.357-370(1990)