組織モデルに基づくワークフロー管理と例外処理

山口 真悟 ・ 守田 了 ・ 田中 稔

知能情報システム工学科

ワークフロー管理システムは，グループ作業を手順として記述し，その流れを管理するシステムである．本論文では，ワークフローにかかわる作業者の権限に着目して，作業者がその権限の範囲内で状況に応じた例外処理を行なうための機構について述べる．まずワークフローにかかわる作業者の権限の集合から組織モデルを構築する．組織モデルは作業者の権限を検査する機構を与える．本論文で提案する組織モデルの特徴は作業者の権限が組織構造にしたがって委譲できることであり，ワークフローの例外処理の柔軟性を高めている．さらに作業者同士の作業状況を伝え，作業者間のコミュニケーションを支援するためにアウェアネス支援機構を備えたユーザインターフェイスを開発した．提案する機構は作業手順だけでなく作業状況や組織情報を活用することによってワークフローの例外処理を柔軟に行なえることが特徴である．

Key Words : *workflow management, exception handling, organization, awareness*

1. はじめに

コンピュータネットワークの普及と電子メイルシステムの発展にあいまってワークフロー管理システムへの注目が高まっている．ワークフロー管理システムは，グループ作業を手順として記述し，その流れを管理するシステムである[1]．伝票処理を始め，複数の人がかかわるグループ作業の連携を自動化し，グループ作業全体の処理時間を短縮できる．さらにビジネス・プロセス・リエンジニアリングの分野にも積極的な利用が行なわれている．

ワークフロー管理は，あらかじめ定義された手順通りにグループ作業が進む場合はきわめて有効である．しかし実際のグループ作業には人間が介在するので，作業者が出張していたり作業の進行が大幅に遅れたりするといった例外が生じて，そのままでは手順通りに作業を進めることができなくなる場合がある．この例外に対処するため，進行中のワークフローを取り消し再実行する方式，あらかじめ例外処理を含んだワークフローを実行する方式が利用されている．しかし前者の方式はワークフローを再実行するので作業者の負担が大きく，後者の方式は前もって例外処理を記述するため柔軟性に乏しい．例外事象に満ちた現実世界においてどれだけ柔軟性をもたせられるかがワークフロー管理システムの大きな研究課題となっている[2]．

一方，グループ作業が円滑に行なわれるためには，作業手順だけでなく，作業主体の組織構造や誰がどのような作業を行なっているのかといった作業状況が相互に把握されている必要がある．現実のグループ作業ではこれらの情報にしたがって，作業者がその権限の範囲内で状況に応じた例外処理を行なうことが多く，そのような柔軟な解決を支援する機能はワークフロー管理システムの重要な要素となる．ところがワークフロー管理システムは，電子メイルなどの非同期通信を基盤として発展してきたため，作業状況をリアルタイムに伝達するようには設計されておらず，ワークフローの例外処理に作業状況と組織情報を組み合わせて活用した研究は少ない．

本論文では，ワークフローにかかわる作業者の権限に着目して，作業者がその権限の範囲内で状況に応じた例外処理を行なうための機構について述べる．まずワークフローにかかわる作業者の権限の集合から組織モデルを構築する．組織モデルは作業者の権限を検査する機構を与える．本論文で提案する組織モデルの特徴は作業者の権限が組織構造にしたがって委譲できることであり，ワークフローの例外処理の柔軟性を高めている．さらに作業者同士の作業状況を伝え，作業者間のコミュニケーションを支援するためにアウェアネス (awareness) 支援機構を備えたユーザインターフェイスを開発した．

以下，2. ではワークフローモデル OM-1 を取り上げて，ワークフローの例外処理について説明する．3. では権限関係からなる組織モデルとそれに基づく例外処理機構について述べる．4. では作業状況の共有を図ったアウェアネス支援機構について述べ，5. でその設計に基づいたシステムの実現と評価を示す．

2. ワークフロー管理と例外

ワークフローモデル OM-1(Office Model One)[3]を取り上げてワークフロー管理とその例外について説明する.

2.1 ワークフローモデル OM-1

OM-1 はオブジェクト指向の階層型意味ネットワークを用いてオフィス業務を記述するワークフローモデルである. 図 1に OM-1 による物品購入処理の記述例を示す. OM-1 の記述は処理構造と組織構造の 2 つの記述からなる. 図 1の上半分が組織構造の記述で, 下半分が処理構造の記述である. 処理構造はアクティビティ(activity)をノードに, アクティビティ間の順序関係を有向エッジに対応付けた有向グラフで記述される. 組織構造は役割・部署をノードに, 役割・部署間の包含関係をエッジに対応付けたグラフで記述される. そして処理構造と組織構造はアクティビティと役割・部署間の責任(responsibility)関係で結合される. 処理構造にはアクティビティの制御だけでなくデータの制御も記述できるが, 本論文では割愛した.

OM-1 の記述を用いたワークフロー管理の概略を簡単に説明する. 一つの記述は何回も実行されうる. たとえば, 物品購入処理は物品を購入する要求が発生するたびに実行される. OM-1 はクラス・インスタンスの概念を導入して処理の記述とその実行を区別している. 一回ごとの実行(ワークフローインスタンス)は一つの記述(ワークフロークラス)からインスタンス生成される. このとき作業者が組織構造の記述にしたがって処理構造のアクティビティに割り当てられる. 各作業者は割り当てられたアクティビティを順に実行していくことで, グループ作業が進んでいく. そしてワークフローインスタンスが終了条件を満たすとグループ作業は終了し, 同時にワークフローインスタンスも削除される.

2.2 ワークフローの例外

ワークフロー管理は, あらかじめ定義した記述通りにグループ作業が進む場合はきわめて有効である. しかし実際のグループ作業には人間が介在するので, そのままでは記述通りに作業を進めることができない場合がある. たとえば, 作業者が出張していたり作業の進行が大幅に遅れる場合がある. 現実のグループ作業では, このような例外が起こった場合, 各作業者がその権限の範囲内で例外を処理することが多い. このことから作業者の権限や組織構造は例外処理の枠組みも与えていると考えられる[4].

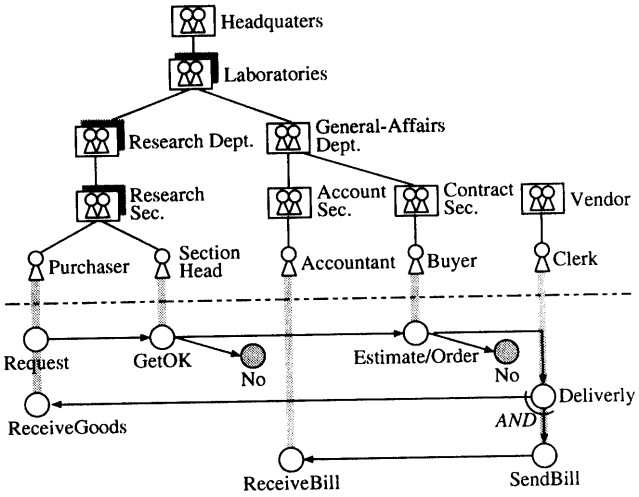


図 1 OM-1 による物品購入処理の記述例 (抜粋)

物品購入処理を例として組織構造にしたがった例外処理の概略を与える. 物品購入処理において役割 Purchaser の作業者はアクティビティRequest と ReceiveGoods の実行に責任がある. この作業者がアクティビティRequest を実行した後, 長期出張に出かけた場合, この出張はアクティビティReceiveGoods を実行できないという例外を引き起こす.

一方, 実世界ではそれぞれの役割に権限がある. 役割 Section Head の権限が Purchaser の権限を包含しているならば, アクティビティを代行することで例外を解決できる. このような例外処理が実現できれば, ワークフロー管理システムの柔軟性を高めることができる.

3. 組織モデルと例外処理機構

本章では, 作業者の権限の集合から組織モデルを構築し, それに基づいた例外処理機構について説明する.

3.1 権限関係の組織モデル

ワークフロー管理において作業者の権限は例外処理の枠組みを与える重要な概念である. OM-1 では作業者の組織構造をモデル化するが, 作業者の権限は取り扱っていない. そこで筆者らはワークフローにかかわる作業者の権限に着目し, 権限の集合から構成される組織モデルを開発した.

本モデルではアクティビティの実行許可を使って作業者の権限を表す. アクティビティの実行許可とは, あるアクティビティを実行するために作業者が満たしているべき条件である. アクティビティの実行許可を使って, 権限, 役割, 権限関係, 組織構造を以下のように定義する.

権限 権限はアクティビティの実行許可の集合である.

役割 役割は権限を持つ主体である. ある作業者 u の役割を $r(u)$ と表記する.

権限関係 役割間の権限の違いはアクティビティの実行許可の有無によって生じる. 権限関係 $\preceq$ はアクティビティの実行許可の包含関係で定義される.

$$r_i \subseteq r_j \Rightarrow r_i \preceq r_j$$

権限関係は半順序関係である.

組織構造 権限関係の定義された役割の集合を組織構造という.

あるアクティビティ a の実行許可を $p(a)$ と表記する. 役割 Purchaser がアクティビティ Request と ReciveGoods の実行許可を持つならば, その役割は

$$r_{Purchaser} = \{p(\text{Request}), p(\text{ReceiveGoods})\}$$

と表される. また役割 Section Head が

$$r_{SectionHead} = \{p(\text{Request}), p(\text{ReceiveGoods}), p(\text{GetOK})\}$$

とすると, 権限関係は

$$r_{Purchaser} \preceq r_{SectionHead}$$

と表される.

権限は組織構造に基づいて委譲できる. 組織構造に基づいた権限の委譲とは, 役割 r_A と役割 r_B が $r_A \preceq r_B$ を満たすとき, 上役 r_B の権限を一時的に下役 r_A に付与するものである. 権限関係は推移的であるので, 権限の委譲は直接の下役だけでなく間接的な下役 (つまり下役の下役など) にも適用できる. また, 委譲によって付与される権限には, 権限が委譲されてから剥奪されるまでという有効期間がある. この有効期間によって委譲された権限の行使は特定のアクティビティの実行に限定される.

3.2 組織モデルに基づく例外処理機構

例外処理にかかわる作業者の組織モデルを作成し, そのモデルに基づいた例外処理機構を示す.

現実のグループ作業では, ある作業者がアクティビティを実行できない例外が発生すると,

(1) 代理指名 指名人が代理人を指名する.

(2) 代理実行 代理人がアクティビティを実行する.

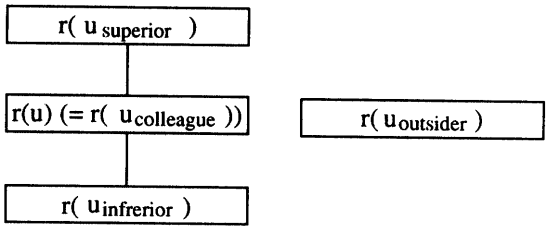


図 2 ある作業者を中心とした組織を単純化したモデル

という手順で例外を解決することが一般的である. その特徴は, 本来の作業者や代理人とは異なる作業者が代理人を指名でき, 作業状況に応じて代理人を変えることができることである.

一方, 代理指名の実行主体が作業者であるので, 越権行為が起こるおそれがある. 越権行為には権限関係に逆らった代理指名や権限のないアクティビティの代理実行が挙げられる. こうした越権行為を防止するには作業者の権限関係を検査する機構が必要であり, 代理指名と代理実行が権限関係を満たすための必要条件は以下のようになる.

代理指名の必要条件

$$(r_{principal} \preceq r_{appointer}) \wedge (r_{agent} \preceq r_{appointer})$$

代理実行の必要条件

$$(p(a) \in r_{agent}) \vee (r_{agent} \preceq r_{principal})$$

ここで, $r_{principal}$ はアクティビティを実行できない作業者の役割, r_{agent} は代理人の役割, $r_{appointer}$ は指名人の役割である. 代理指名の必要条件は, 指名人が被指名人と同役か, それらより上役であることを必要とする条件である. 代理実行の必要条件は, 代理人が代理実行するアクティビティの実行許可を持つか, 代理人がアクティビティを実行できない作業者の下役であることを必要とする条件である. 条件の後半部分は権限関係が $r_{agent} \preceq r_{principal}$ ならば, 代理人は権限の委譲を受けて $p(a) \in r_{agent}$ を満たすことを意味している. これらの条件は組織モデルと照合することで, 例外処理にかかわる作業者の権限を検査し, 越権行為を防止した例外処理機構を与えている.

具体的な組織モデルを使って例外処理機構の動作を説明する. ある作業者 u を中心とした組織を単純化したモデルは次のようになる.

上役の作業者 $u_{superior}$

権限関係: $r(u) \preceq r(u_{superior})$

同役の作業者 $u_{colleague}$

権限関係: $(r(u) \preceq r(u_{colleague})) \wedge (r(u) \succeq r(u_{colleague}))$

表1図3の組織における全ての作業者の組合せに対する代理指名と代理実行の可否．本来の作業者をuとする．表中の各要素は順序対であり，その第1成分が代理指名の可否，第2成分が代理実行の可否を記号(○は可，×は否，△は権限の委譲後に可)で示している．

指名人	代理人			
	$u_{colleague}$	$u_{superior}$	$u_{inferior}$	$u_{outsider}$
u	(○, ○)	(×, ○)	(○, △)	(×, ×)
$u_{colleague}$	(○, ○)	(×, ○)	(○, △)	(×, ×)
$u_{superior}$	(○, ○)	(○, ○)	(○, △)	(×, ×)
$u_{inferior}$	(×, ○)	(×, ○)	(×, △)	(×, ×)
$u_{outsider}$	(×, ○)	(×, ○)	(×, △)	(×, ×)

下役の作業者 $u_{inferior}$

権限関係: $r(u_{inferior}) \preceq r(u)$

部外の作業者 $u_{outsider}$

$u_{superior}$, $u_{colleague}$, $u_{inferior}$のいずれでもない

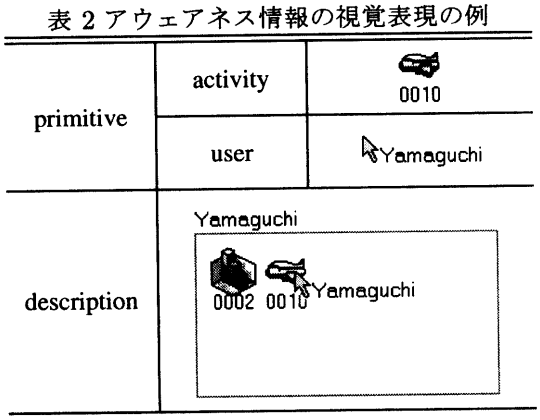
図2は，この組織を図示したグラフである．グラフのノードは役割を表し，エッジは役割間の権限関係を表している．役割$r(u_{outsider})$はそれ以外の役割と権限関係が無い．この組織における全ての作業者の組合せに対する代理指名と代理実行の可否を表1に示す．アクティビティを実行できない作業者をuとする．表中の各要素は順序対であり，その第1成分が代理指名の可否，第2成分が代理実行の可否を記号表記したものである．記号○は可，記号×は否，記号△は権限の委譲後に可であることを示している．たとえば，指名人が$u_{superior}$，代理人が$u_{colleague}$とすると，

- 代理指名にかかわる作業者の権限関係
  $(r(u) \preceq r(u_{superior})) \wedge (r(u_{colleague}) \preceq r(u_{superior}))$
- 代理実行にかかわる作業者の権限関係
  $r(u) \preceq r(u_{colleague})$

となり，代理指名と代理実行の必要条件を満たしているので，この例外処理が権限の範囲内であることが保証される．

例外処理の実行は，代理指名と代理実行が可の場合(表1中(○, ○)と(○, △))だけに限られる．それ以外の場合は越権行為である．例外処理機構は組織モデルの中である作業者を指定すると指名人や代理人になりうる作業者を見つけることが可能であり，作業者に候補を提示する機能などへの応用が考えられる．

表2アウェアネス情報の視覚表現の例



4. 例外処理とアウェアネス支援機構

グループ作業が円滑に行なわれるためには，誰がどこにいるのか，誰がどのような作業を行なっているのといった作業状況が作業者同士で相互に把握されている必要がある．一方，ワークフロー管理システムは電子メイルなどの非同期通信を基盤として発展してきた非同期型グループウェアであるため，作業状況をリアルタイムに伝達するようには設計されていない．そこで同期型グループウェアのアウェアネス支援技術を導入し，リアルタイムな作業状況の提供を図ったユーザインターフェイスを開発した．

アウェアネスとはXerox PARCのStultsらが主張している同期型グループウェアの概念の一つで，相手と陽にコミュニケーションしなくても相手が在籍しているか，取り込み中かなどの状況を把握できることである[5]．アウェアネス情報には様々なものがあるが，本研究では以下の3つを取り上げている．

- 作業者の存在
- 処理の進行状態(アクティビティが実行中か未処理か)
- 例外処理の方法

本アウェアネス支援機構はアウェアネス情報の通知に共有ウインドウ[6]を用いている．共有ウインドウは，ある利用者がその内部に文字や図形を書き込むと，それと全く同じものが他の利用者の共有ウインドウに表示される機能を備えたウインドウである．共有ウインドウはアイコンを使ってアウェアネス情報を表示する．

画面上の基本要素は，アクティビティを表わすアイコン(アクティビティアイコン)とそれを操作するカーソルである(表2)．アクティビティアイコンは，ワークフローのクラスを図案化した絵シンボルと，インスタンス番号を表示したラベルからなる．表2のアクティビティアイコンは，ワークフローのクラスが出張申請で

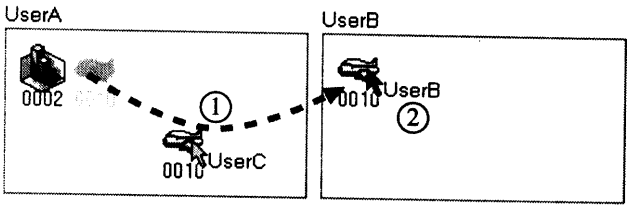


図 3 例外処理の画面操作

インスタンス番号が 10 番のアクティビティを表わしている．カーソルもアクティビティアイコンと同様に共有ウインドウで共有される視覚表現であり，他の作業者のカーソルと区別できるように作業者の名前を伴っている．作業者がアクティビティアイコンをクリックすると，外部アプリケーションが起動しアクティビティを実行できる (アクティビティを実行している間，その作業者の個人領域にはアクティビティアイコンをクリックした状態が表示されている)．

共有ウインドウは作業者の個人領域に分割されている．個人領域は実世界の個人作業机のメタファである．アクティビティアイコンはワークフローの進捗状況にしたがって各作業者の個人領域に配置される．したがって個人領域を見れば実行中のアクティビティや未実行のアクティビティが把握できる．たとえば，表 2の視覚表現から，作業者 Yamaguchi が出張申請クラス 10 番のアクティビティを実行中で，売上集計報告クラス 2 番のアクティビティを未実行であるというアウェアネス情報を知ることができる．

例外処理それ自体も重要なアウェアネス情報の一つである．ある作業者が次のような手順で画面操作を行なうと，他の作業者は誰がどのように例外を処理しているかを知ることができる．

操作①代理指名 指名人は代理人を指名するために，アクティビティを実行できなくなった作業者の個人領域から代理人の個人領域へアクティビティアイコンを移動する．

操作②代理実行 代理人はアクティビティを代理実行するために，ドラッグされたアクティビティアイコンをクリックする．

図 3は例外を処理している画面の一部である．作業者 UserC は作業者 UserA の代理人を UserB に指名するために，作業者 UserA の個人領域から UserB の個人領域へアイコンを移動する (操作①)．計算機は作業者が操作①を行なった直後に，3.2節で示した権限の検査機構を使って，代理指名と代理実行にかかわる作業者の権限を検査する．代理実行にかかわる作業者の権限が代理実行の前に検査できるのは，操作①が終わった時点で代理実行にかかわる作業者が決定するからである．その結果が越権行為の場合は，操作①をキャンセルし，その作業者に注意を促す．また権限の委譲が必要な場合には，作業者に委譲の許可を問い合わせる．代理指名と代理実行がともに権限の範囲内と判断された場合に限り，作業者 UserB はアクティビティを代理実行できる．以上の機構により作業者が行なった例外処理が権限の範囲内にあることが自動的に保証される。

5. 実現と評価

本論文で提案する例外処理機構を備えたワークフロー管理システムの実現について説明し，実現したシステムを用いた実験と評価を示す．

5.1 実現

本システムは WWW(World Wide Web) を構築基盤としたクライアント/サーバモデルで構成されている．作業者は WWW のブラウザで動作するクライアントを使ってサーバの機能を呼び出しワークフローの処理を行なう．サーバとクライアントは HTTP と TCP/IP で接続され，前者を (HTML 言語で記述した) データの制御に使い，後者をワークフローの制御と共有ウインドウの実現に使った．またサーバとクライアントの実装には C++と Java を用いた．

クライアントはアウェアネス支援機構を実現したユーザインターフェイスを提供する．クライアントの表示画面の例を図 4に示す．画面上には組織構造を表示した作業者選択ウインドウ (図 4Ⓐ) とアウェアネス情報を表示した共有ウインドウ (図 4Ⓑ)，アクティビティ実行ウインドウ (図 4Ⓒ) がある．作業者選択ウインドウは，作業者が自分の共有ウインドウに表示するアウェアネス情報の範囲を組織全体の中から選択するために使うウインドウである．作業者選択ウインドウは組織構造をフォルダアイコンの階層構造で表示し，作業者はフォルダアイコンをオープン・クローズする操作でアウェアネス情報の表示・非表示を選択できる．たとえば，図 4において，役割 Director と Staff のフォルダを閉じるとそれらのアウェアネス情報の表示を消すことができる．この選択的表示機能は共有ウインドウの画面サイズの問題[7]を軽減する一つの方法を示していると考えている．

5.2 実験と評価

アウェアネス支援と例外処理機構の有効性を検証するために，本機構を備えたシステムと備えないシステ

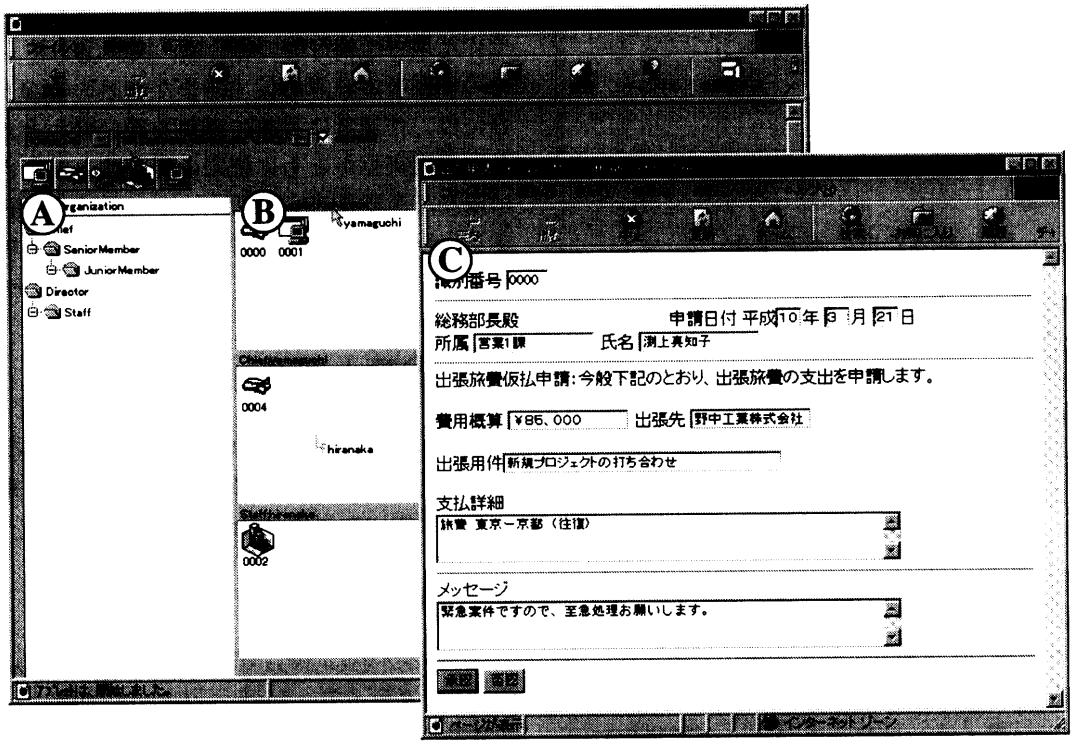


図 4 クライアント表示画面例

表 3 ワークフロー「WWW 利用統計の報告」の追跡結果．表中の要素は上段がアクティビティの開始時間，下段がアクティビティの終了時間を表している．また四角形で囲んである要素は例外処理が行なわれたことを示している．

(a) アウェアネス支援と例外処理の機構を備えたシステムの結果

実験日	Fill in	Approve	Fill in(差戻し)	Approve(差戻し)	Entry	Check
10/20(月)	10/20/17:44:45 10/20/17:45:39	10/20/17:46:59 10/20/17:48:02			10/20/18:12:47 10/20/18:14:08	10/20/19:57:41 10/20/19:58:12
10/21(火)	10/21/13:18:13 10/21/13:19:50	10/21/13:26:59 10/21/13:28:42			10/21/18:00:22 10/21/18:00:48	10/21/18:07:33 10/21/18:07:54
10/22(水)	10/22/13:23:07 10/22/13:24:59	10/22/15:16:21 10/22/15:16:52			10/22/17:45:45 10/22/17:46:03	10/22/18:31:28 10/22/18:31:45
10/23(木)	10/23/08:46:09 10/23/08:47:51	10/23/10:09:41 10/23/10:12:05			10/23/18:20:35 10/23/18:21:03 Ⓐ	10/23/18:28:38 10/23/18:29:09
10/24(金)	10/24/17:10:08 10/24/17:11:34	10/24/17:13:20 10/24/17:13:27	10/24/17:14:01 10/24/17:14:52	10/24/17:15:05 10/24/17:15:29	10/24/17:40:21 10/24/17:40:50	10/24/19:15:40 10/24/19:16:13

(b) アウェアネス支援と例外処理の機構を備えないシステムの結果

実験日	Fill in	Approve	Fill in(差戻し)	Approve(差戻し)	Entry	Check
10/13(月)	10/13/11:49:07 10/13/11:50:33	10/13/15:38:43 10/13/15:39:04	10/13/16:01:08 10/13/16:02:12	10/13/16:15:25 10/13/16:15:40	10/13/16:17:13 10/13/16:18:06	10/13/17:59:08 10/13/17:50:39
10/14(火)	10/14/13:19:29 10/14/13:22:20	10/14/15:21:51 10/14/15:23:09			10/14/16:34:40 10/14/16:35:57	10/14/16:42:05 10/14/16:42:44
10/15(水)	10/15/18:22:15 10/15/18:23:17	10/15/18:23:24 10/15/18:23:38	10/15/18:25:45 10/15/18:26:02	10/15/18:26:06 10/15/18:27:12	10/15/18:53:01 10/15/18:53:26	10/15/21:43:35 10/15/21:45:13
10/16(木)	10/16/13:37:07 10/16/13:39:11	10/16/14:20:00 10/16/14:20:24			10/17/13:58:59 10/17/13:59:42 Ⓑ	10/17/14:01:26 10/17/14:01:47
10/17(金)	10/17/13:53:33 10/17/13:56:42	10/17/16:10:59 10/17/16:11:27			10/17/17:59:15 10/17/17:59:55	10/17/18:04:29 10/17/18:04:51

ムの利用比較を行なった．後者のシステムは前者のシステムからアウェアネス支援と例外処理の機能を除去したものである．

実験はそれぞれのシステムで定期的に実行されるワークフローの追跡 (trace) を比較するという方法で行なった．追跡の対象は図 5に示した「WWW 利用統計の報告」というワークフローである*．このワークフローは研究室から総合情報処理センターへ 1 日 1 回 WWW サーバーの利用統計を報告するという作業を想

*実験期間中，追跡対象外のワークフローも実行された．

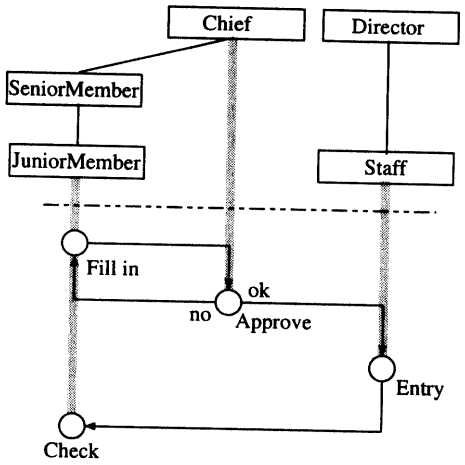


図 5 ワークフロー「WWW 利用統計の報告」の記述

定したものである．この作業にかかわる作業者は研究室とセンターのいずれかの組織に所属している．研究室に所属している作業者には 3 つの役割 JuniorMember, SeniorMember, Chief があって，その組織構造は $r_{JuniorMember} \preceq r_{SeniorMember} \preceq r_{Chief}$ である．またセンターに所属している作業者には 2 つの役割 Staff, Director があって，その組織構造は $r_{Staff} \preceq r_{Deirctor}$ である．作業は以下のような手順で行なわれる．研究室に所属する役割 JuniorMember の作業者は報告書を作成 (Fill in) し，その上役 Chief の承認 (Approve) を得た上でセンターに利用統計を報告する．アクティビティApprove で上役が否決すると報告書はアクティビティFill in へ差戻される．センターでは役割 Staff の作業者が報告書を確認した後，それをセンターの計算機に登録 (Entry) する．最後に報告者はセンターから報告書を受け取って，報告作業が無事完了したことを確認する (Check).

表 3にそれぞれのシステムに対する 5 日間の実験結果を示す．表 3(a) はアウェアネス支援と例外処理の機構を備えたシステムの結果であり，表 3(b) はその機構を除去したシステムの結果を示している．表中の各要素は上段がアクティビティの開始時間，下段がアクティビティの終了時間を表している．また四角形で囲んである要素はそのアクティビティが例外処理されたことを示している．比較実験の両方で役割 Staff の作業者がアクティビティEntry を実行できない例外が 1 回ずつ発生した (表 3Ⓐと Ⓑ)．例外処理機構を備えたシステムでは役割 Director の作業者が例外処理を行なったのに対し，その機構を備えないシステムでは例外を解決できず，役割 Staff の作業者が戻るまで作業を進めることができなかった．例外の解決に要した時間は前者のシステムが約 8 時間であったのに対し，後者のシステムでは約 24 時間かかった．また例外が起きていない 4 日間の平均処理時間を比較すると，本機構を備えたシステムが約 3 時間 35 分，本機構を備えないシステムが約 4 時間 40 分であった．この結果，本機構がワークフローの例外処理だけでなく通常処理にも有益であると考えられる．

次に関連する研究との比較を示す．これまでワークフローの例外に対処するため，全体を取り消し再実行する方式，あらかじめ記述した規則にしたがって例外処理を行なう方式が提案されている．たとえば，OM-1 を基にした COOKBOOK[3] は全体を取り消し再実行するシステムであり，Access の Application Gateway[8] はあらかじめ記述した規則にしたがって例外処理を行なうシステムである．本論文で提案する例外処理機構は，作業者が組織構造やアウェアネス情報を活用して例外処理を行なう点に特徴がある．従来の方式のように例外の発生場面を想定する必要はなく，組織モデルに基づいて幅広い例外に対処することが可能である．

ワークフロー管理に組織情報を取り入れたモデルやシステムが提案されている．FORC モデル[9] はワークフローが，伝票，組織，業務の流れ，文書の内容によって構成されるとするワークフローモデルである．また日立の Groupmax [10] は個人や組織に関する情報を管理するディレクトリサービスを提供している．しかし，いずれの組織情報も例外処理の枠組みを与えることを意図したものではない．

グループ作業にアウェアネス支援を試みる研究が行なわれている．Prinz らはグループ作業におけるチームアウェアネス (team awareness) (チームの他のメンバーの作業状況を知ること) の重要性を強調している[11]．協調型ハイパーメディアシステム VIEW Media[12] はメディアを利用する作業者間のコミュニケーションを支援するためにアウェアネス支援機構を提供している．また Communication Assist/Awareness[13] は IRC(Internet Relay Chat) を用いたアウェアネス支援機構を提供している．しかし，いずれもワークフロー管理にアウェアネス支援を応用する研究ではない．また DEC の LinkWorks や Olibetti の IBIsys は未実行のアクティビティを個人別に表示し，その実行を催促する機能を備えている[14] が，アウェアネスを提供するものではない．

6. おわりに

本論文では，ワークフローにかかわる作業者の権限に着目して，作業者がその権限の範囲内で状況に応じた例外処理を行なうための機構を提案した．その機構を与えるために必要なメカニズムとして，ワークフローにかかわる作業者の権限の集合から組織モデルを構築

し，それに基づく例外処理方式を明らかにした．また作業者同士の作業状況を伝え，作業者間のコミュニケーションを支援するためのアウェアネス支援機構を示した．そして提案した機構を実現するワークフロー管理システムの実装を行なった．実際にシステムを使った利用評価を行ない，良好な結果を得た．本論文で提案した機構はワークフローの例外処理に作業状況と組織情報を活用することが特徴であり，ワークフロー管理の柔軟性を高める重要な要素となることが期待できる．

謝辞：本研究の実験にご協力いただきました本学総合情報処理センターの久長 穣講師を始めとする皆様方に謝意を表します．

参考文献

1) 垂水: グループウェア・ワークフローの研究動向, 電子情報通信学会技術研究報告, KBSE97-30 (1997).
2) 石井: グループウェアのデザイン, 共立出版, pp. 42-44 (1994).
3) H. Ishii and M. Arita: Message-driven Groupware Design Based on Office Procedure Model: OM-1, 情報処理学会欧文論文誌, Vol. 14, No. 2 (1991).
4) 山口, 守田, 田中: 組織構造に基づく対話型ワークフロー管理方式, 電子情報通信学会技術研究報告, MVE96-63 (1997).
5) 石井: リアルタイムグループウェアのデザイン, 情報処理学会誌, Vol. 34, No. 8, pp. 1017-1027(1993).
6) J. C. Lauwers and K. A. Lantz: Collaboration Awareness in Support of Collaboration Transparency: Requirements for the Next Generation of Shared Window Systems, *Proc. CHI'90*, pp. 303–311 (1990).
7) S. J. Gibbs: LIZA: An Extensible Groupware Toolkit, *Proc. CHI'89*, pp. 29–35 (1989).
8) 鎌田: 電子メール拡張によるグループウェア実現技術, OPEN DESIGN, No. 9, CQ 出版, pp. 84–111 (1995).
9) 神谷, 瀧野, 山本: WWW を用いたワークフロー管理システムに関する変更容易性の評価, 電子情報通信学会技術研究報告, KBSE91-13 (1997).
10) 塚本: 日立の統合型グループウェア Groupmax の概要, 情報処理学会研究報告, GW-20-7 (1996).
11) W. Prinz and S. Kolvenbach: Support for Workflows in a Ministerial Environment, *Proc. of the ACM 1996 Conference on Computer Supported Cooperative Work*, pp. 199–208 (1996).
12) 藤田, 加用, 上林: 分散共同作業支援環境における仮想オフィスシステムの実現, 情報処理学会第 50 回全国大会, 2M-2 (1995).
13) 奥山, 福山, 岡田, 松本, 真鍋: オフィスコミュニケーション支援のためのアウェアネスサービス, 電子情報通信学会 1997 年総合大会, D-9-8 (1997).
14) 業務の連携を自動化，時間短縮と管理を実現．ワークフロー管理ソフトが日本でも利用可能に, 日経コンピュータ, 94 年 5 月 2 日号, pp. 57–67 (1994).

(1998. 5. 15 受理)

WORKFLOW MANAGEMENT AND EXCEPTION HANDLING BASED ON ORGANIZATION MODEL

Shingo YAMAGUCHI, Satoru MORITA and Minoru TANAKA

Workflow management system automates and manages procedures of group-work, however, it often happens that the procedures cannot be completed because of unpredictable situations. In this paper, we propose new mechanisms of workflow management, which allows users to handle exceptions based on an organization model. First, we presented the organization model to describe privileges. The model make a feature of privilege transfering and provide a facility for testing exception handlings. Next, we designed an user interface to grasp each other's progress. To accomplish this, the awareness support mechanism has been supported. These mechanisms provide flexibility for a workflow management and an exception handling in comparison with existing ones, because users can use organizational informations and runtime contexts to handle exceptions.