グループウェア 18－6
（1996. 6. 14）

# ユーザ経験の追加によるワークフロー機能拡張システム TORES

飯塚　京子 *1　　桧垣　博章 *2　　平川　豊 *1

*1 ＮＴＴソフトウェア研究所
*2 東京電機大理工学部

## 概要

ワークフローシステムによって、定型的な作業を計算機の支援で、効率をあげる試みがなされている。しかし、一見定型的と思われる作業でも、人間の臨機応変な対処を必要とする場合が多く、それが計算機による自動化を困難にしている。通常のワークフローシステムは不確定要因を計算機上で定義できないため、どうしても例外処理は発生する。

我々は、例外処理の発生を防ぐのではなく、例外処理の経験を持つ作業者が、同一作業を行う他の作業者に迅速かつスムーズに伝達でき、その経験がシステム上で通常作業と同等に扱えるようになればよいと考えた。そして、それを実現する方法として”逐次追加法”を提案し、”逐次追加法”を実現するシステムTORESを構築した。

# TORES: Task Order Evolution System

Kyoko Iiduka *1　　Hiroaki Higaki *2　　Yutaka Hirakawa *1

*1 NTT Software Laboratories
*2 Tokyo Denki University

## Abstract

In this article, we propose an incremental evolution methodology for work flow systems.

Work flow systems are support systems for procedural activities in real world. With existing systems, acceptable method of the work (task orders) should be defied before the usage of these systems. However, the task orders are often variable. It takes good many time for work flow systems to catch up with these changes.

To cope with this situation, the proposed methodology allows 'on-the-fly' tailoring of task orders.

We also introduce a implementation of the methodology, TORES(Task Order Evolution System).

## 1 はじめに

作業を計算機システムに乗せ、自動化(ないしは半自動化)する試みは古くから行なわれてきた。従来のウォータフォール設計手法では、作業の性質や関連する要因が明確なっており、それらが時間とともに変化しない場合は有効である。

しかし実際の作業では、人間が臨機応変に対応することで対処する場面が多々ある。これは、システム化に向けた作業の要因分析ができない部分が存在するためである。例えば通信販売の業務の中には、あらゆる顧客の要望や苦情などに対処する作業が含まれる。この作業には“顧客”という動作の予測ができない要因が含まれている。よって、この種の作業をシステム化するためには従来の方法では不十分である。

しかし、このような作業においても、システムによる支援を望む声が大きい。最近のPCの普及とともに、ワークフローシステム(WFS)を導入する企業が増加しているのはその証拠である[1]。

本稿ではWFSを例にとり、作業を支援するシステムの構築方法を考察する。まず、作業に対する計算機支援の方法論の検討を検討し、“逐次追加法”を提案する。次に、“逐次追加法”の実現システムとして、TORES(Task Order Evolution System)の実装と評価を行う。最後に、TORESの課題について述べる。

## 2 WFSにおける作業手順

### 2.1 問題点

WFSには“進捗管理”や“スケジュール管理”、“リソース管理”などの様々な機能がある。その中でも“進捗管理”機能は、サポートする作業の作業手順(通常ワークフローと呼ばれるもの)をシステム上で定義しており、その作業手順に従い進捗管理が行なわれる。本稿では、システム上で定義する作業手順に焦点をあてて考察する。

通信販売の受付業務において、顧客が支払方法としてクレジットカードを指定したときの作業手順を考えてみる。この場合、カードの期限が切れていたり、カード不正利用してたり、誤った番号を指定したりと、様々な事態が考えられる。カードの不正利用が頻発するようならば、それを防止するチェックが必要となるであろうし、チェックが厳しい場合は、顧客の不評を買うかもしれない。このような1例を見ても、完全な作業手順を作ることは難しいし、思った効果を発揮しない場合がある。

上記の例のように、実作業では作業手順通りに進む場合ばかりではなく、しばしば例外的な処理(例外処理)が発生する。例外処理ではシステムの支援が適用できないため、作業効率は著しく低下する。よって、迅速に例外処理事例を作業手順に反映することが望まれる。しかしこれでは、例外処理事例の要因を分析し、作業手順を再設計する必要に迫られるため、作業手順の保守には多大なコストがかかることになる。

### 2.2 逐次追加法

上記で説明したとうり、例外処理を無くすことは難しい。そこで我々は、例外処理におけるコストを削減する方法について考えてゆく。

WFSの運用形態を見ると、複数の作業者が同じ作業手順に従い作業を進めてゆく場合が多い。例えば、受付担当者は何人かいるが、みな同じ受付業務の作業手順通りに作業を進めている。

例外処理が発生した場合、例外処理に遭遇した作業者にはその対処方法がノウハウとして蓄積される。そして、もし他の作業者に例外処理ノウハウを容易に伝達できれば、同じ例外処理が再発生しても作業に混乱をきたすことはなくなるであろう。更に、蓄積されたノウハウが作業手順の中に手順として組み込まれれば、その例外処理は通常の作業手順と同じくシステム上で処理することが可能となるため、一段と作業効率を向上させることが可能であろう。

そこで我々は“逐次追加法”を提案し[2]、例外処理に関する問題の解決方法を考える。“逐次追加法”は、システム上で定義した作業手順に対して、ユーザが得た例外処理の経験を追加してゆくことで、作業手順を成長させる。つまり“逐次追加法”は、1人の作業者が得た例外処理経験を他の作業者に伝えると同時に、例外処理のための手順を作業手順に組み込むこと目的とする。

#### 2.2.1 作業

問題の簡略化のため、本稿で取り扱う作業は、一人の作業者が目的を達成するために行なわれる一連の作業を対象とする。一般的なWFSでは、複数の部門にまたがる作業を扱う。例えば通信販売業務では、顧客からの注文を受付る受付部門、会計を行なう会計部門、商品の配送を行なう配送部門など複数の部門に分れている。このような作業は部門ごとに作業者がおり、個々の作業者が分担する作業に分割して考える。例えば受付部門の作業では、顧客からの受注から各部門へ受注を伝達するまでの作業、受付部門

からの会計処理依頼から会計終了、受付部門からの配送依頼から商品の配送完了までの作業と分割する。

#### 2.2.2 逐次追加法の概要

逐次追加法の説明をすると、以下に示す方法により作業手順へ例外処理の経験を追加する。

1. ユーザが例外処理に遭遇した場合、一旦システム支援から離れ、自力で例外処理に対処する。
2. 例外処理を対処したユーザには、経験としてその例外処理の対処手順が得られる。(図 1 Step2)
3. ユーザは例外処理の対処手順をシステムへ投入する。(図 1 Step3)
4. システムはこの対処手順を既存の作業手順に組込む。(図 1 Step4)
5. 以後、この対処手順に対してシステム支援することができるようになり、同じ例外処理は 2 度と発生しない。(図 1 Step1)

これを繰り返すことで、徐々に作業手順にユーザの経験が蓄積され、例外処理を経験していないユーザにも経験を共有することができる。

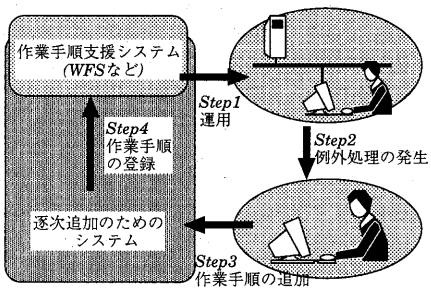


図 1: 逐次追加法の概要図

### 2.3 逐次追加法の要求条件

#### 2.3.1 前提条件

作業手順をシステム上に乗せるため、作業手順のモデル化と設計を行なう必要がある。我々は、ソフトウェアプロセスを記述するための以下の 3 つの側面から捉える手法 ([3], [4]) を参考に作業手順のモデルを考察する。

**機能的側面** 単位となる作業 (タスク) の機能を定義するフィールド

**動作的側面** タスクの実行順序 (つまり作業手順) を定義するフィールド

**構成的側面** リソースの配置を定義するフィールド

上記 3 つの側面に則して考えると、例外処理の対処手順を既存の作業手順に組込む場合、機能的側面においては新しいタスクの定義、動作的側面においてはタスクの実行順序の修正、構成的側面においてはリソース配置の修正と新たなリソースを追加する必要がある。

対処手順の組込みを考察するために上記の 3 点すべてを考慮する必要があるが、問題を単純化するために、最も基盤となるは動作的側面に注目する。そこで以降、追加に必要なタスクは全て用意されており、リソースの配置は固定であると仮定して考察してゆく。

#### 2.3.2 手順の追加方法

逐次追加法を効果的にするため、例外処理の対処手順を即座に蓄積させる必要がある。よって、既存の作業手順への組み込みは自動化したい。更に、逐次追加法では、システムに関する知識の乏しいエンドユーザが、対処手順をシステムへ入力する必要があり、システムへの入力は容易に行なえねばならない。

これらの要求に対して、我々は以下の 2 つの方法でアプローチする。

- 作業手順を“タスクの実行順序”で記述する。これにより、既存の作業手順への組込みが容易になる。
- 入力用のグラフエディタを用意し、作業手順をシンプルな図で表示し、ユーザは同じ図的表現で描いた対処手順を描き加える。図的表現を用いることで、比較的入力が容易に行なえるようになる。

以降、“タスクの実行順序”で記述した作業手順をワークフロー (WF)、対処手順を“追加 WF”、WF の図的表現を“WF 図”と呼ぶことにする。

#### 2.3.3 WF 図の要求条件

上記のアプローチだけでは、対処手順の入力は十分容易であるとは思えない。そこで、WF 図の表記方法に関する以下の 3 点を考る必要がある。

1. WF 図の上で、追加 WF の組込み箇所を容易に探索するためには何が必要か。
2. グラフエディタ上で容易に追加 WF を描き加えるために何が必要か。

3. 追加 WF を組込むと WF 図が大きくなるが、これに対する弊害は何か。またその解決するためには何が必要か。

この 3 点を考察するために、我々はソフトウェアのインストール作業と通信販売作業の WF 図と対処手順を記述する予備実験を行なった。予備実験の結果から、以下の要求条件が明らかになった。

1. 作業手順が、連結した 1 つの WF 図で記述できる

   理由：連結した 1 つの WF 図ならば、入力箇所の特定が容易であるため。

2. 1. 並行作業の記述が可能かつ容易である
   2. 分岐の追加が容易である

   理由：予備実験から、2.1) 追加 WF を既存の WF へ並行に追加する場合、2.2) 追加 WF を既存の WF から分岐させて既存の WF へ合流させる場合、以上の 2 つに分類できた。よって、WF 図がこの要求条件を満せば、追加 WF の入力は容易になる。

3. 閲覧性が良い

   理由：WF 図の規模が大きくなると WF 図を理解することが困難になり、入力が難しくなる。閲覧性に関する考察は後程行なう。

## 3 逐次追加システム TORES

### 3.1 TORES の機能

TORES(Task Order Evolution System) とは、“逐次追加法”を実現する WF 拡張システムである。本節では TORES の機能について説明してゆく。

#### 3.1.1 TORES システム構成

TORES の機構は WF 実行系と、WF 追加系に別れている (図 2参照)。

WF 実行系は、システム上に定義している WF に従い作業を進める。WF 実行系は、以下に示す 3 つの定義ファイルと、作業を実行する WF 実行プログラムからなる。

- タスク定義ファイル：WF の構成要素となるタスクの動作を定義する。
- リソース定義ファイル：各タスクが使用するリソースを定義する。
- WF 定義ファイル：タスクの実行順序 (つまり WF) を定義する。

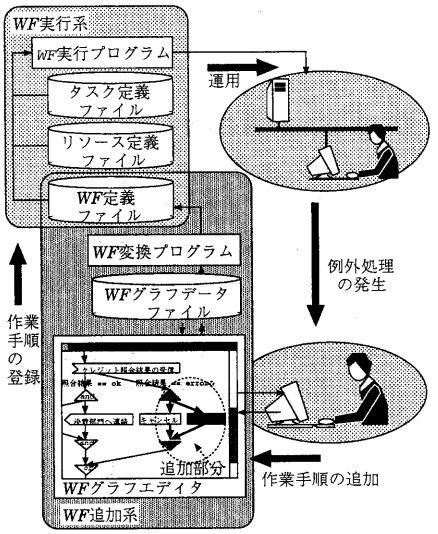


図 2: システム構成図

このサブシステムにより、WFS としての基本的な機能を実現する。WF 実行系を起動すると WF 実行プログラムは 3 つの定義ファイルを読み込み、定義ファイルに従ったタスクをユーザに提示する。この部分は、既存の WFS に置き変えることができる考えている。

これに対し WF 追加系は TORES の核となる。これは、WFS 運用中における追加 WF の組み込みを実現するためのサブシステムである。WF 追加系は、WF 変換プログラム、WF グラフデータファイル、WF グラフエディタからなる。ユーザから追加 WF の入力要求を受けると、WF グラフデータファイルに格納されているグラフ情報を WF グラフエディタに表示する。追加 WF の入力が終了すると、WF 変換プログラムによって WF グラフデータファイルから WF 定義ファイルへ変換する。つまり WF 追加系により、ユーザが入力した追加 WF を、即座に WF 定義ファイルに反映させることを可能にする。

#### 3.1.2 WF グラフエディタと TOREC

先に述べたとうり、TORES には WF を図的に表示編集するための WF グラフエディタを用意している。我々は、この WF グラフエディタ上で表示する WF 図として “TOREC(Task Order Expression Chart)” を提案した [5]。

TOREC は、7 種類のノードとアークを用いて WF を表現する (図 3参照)。各ノードの役割を以下に説明する。

- “task”

タスク定義ファイルに記述された作業を示す

- "and-branch" と "and-marge"
  並行作業の開始位置と待ち合わせ位置を示す

- "or-brunch" と "or-marge"
  分岐位置と分岐の合流を示す

- "send-massage" と "receive-message"
  他の作業者へのメッセージの送受信を示す

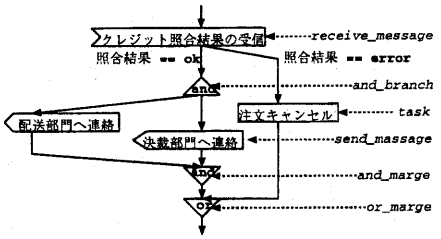


図 3: TOREC 記述例

TOREC は "and-branch" と "and-marge"、"or-branch" と "or-marge" を用いることで、WF 図を分断せずに並行な作業と分岐する作業の入力が容易に行なえる。

## 3.2 TOREC の評価

2.3.3 節で示した要求条件に関して、従来提案されている代表的な図的表記法と TOREC を比較することで評価する。

**SDL[6]** 通信ソフトウェアの下流仕様の記述言語。実行順序を問わない作業を並列に記述する場合は、複数の WF 図に分断される。

**MSC[7]** 通信ソフトウェアの上流仕様の記述言語。WF に分岐がある場合、分岐条件ごとに異なる WF 図に分断される。

**PTN(PetriNet)** 上記の条件全てを満たす。

**R-Net[8]** リアルタイムシステムでのメッセージの受信から送信までの動作記述する図。メッセージの送受信で WF 図は分断される。

要求条件 1,2 に関する比較結果を図 4に示す。比較の結果から、PTN と TOREC 以外の図的表記法は逐次追加に適さないことが明らかになった。

| | SDL | MSC | PTN | R_Net | TOREC |
|---|---|---|---|---|---|
| 1 WF図の枚数*) | × | × | ○ | × | ○ |
| 2.1 並行記述**) | △ | ○ | ○ | ○ | ○ |
| 2.2 分岐記述**) | ○ | △ | ○ | ○ | ○ |

*) ○は 1 枚の WF図で記述できる
**) △は並行/分岐すると WF図の枚数が増加

図 4: 記述性の評価表

### 3.2.1 WF 図の閲覧性評価

最後に、要求条件 3 で挙げた閲覧性に関する評価を行なう。本稿では、WF 図の閲覧性を "閲覧度＝タスクの数÷ WF 図を構成する全要素の個数" とする。閲覧度が大きければ WF 図の閲覧性は良いと言える。但し、ここで言うタスクとは WF の作業手順の単位となる作業のことを意味する。TOREC では、"task"、"send-massagge"、"receive-massage" といったノード 1 つが、1 つのタスクに相当する。

我々は、PTN と TOREC に対して通信販売の WF 図に 13 の追加 WF を組込む記述実験を行なった。記述実験の結果を図 5に示す。この結果から、PTN に比べて TOREC の閲覧性は高いことが判明した。

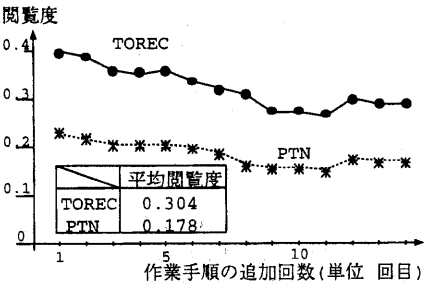


図 5: 閲覧度の比較グラフ

## 3.3 TORES の課題

今後の研究課題は "逐次追加法" の有効性を評価することである。このために、TORES を実作業へ適用し、TORES の適用評価を行なう必要がある。現在の TORES には、"逐次追加法" を実現する基本機能しか備えておらず、実作業への適用するための幾つかの課題がある。

### 3.3.1 WF 検証

追加 WF の組込みの結果、システムはユーザの要求通りに動作するとは限らない。特に TORES ではエントユーザが追加 WF の入力を行なうの

で、組込み時にミスを犯す危険性は高い。そこで、シュミレーションやWF構文解析を用いた検証機能が必要になると考えている。

### 3.3.2 分散環境化

基本的に現在のTORESは、複数の部門にまたがる作業には対応していない。このような作業には、部門ごとに作業を分割してWFを作成し、各WFを各部門で管理運用すればよいと考えている(図6参照)。この場合、管理するWFに対してのみ追加WFの組込めるものとする。

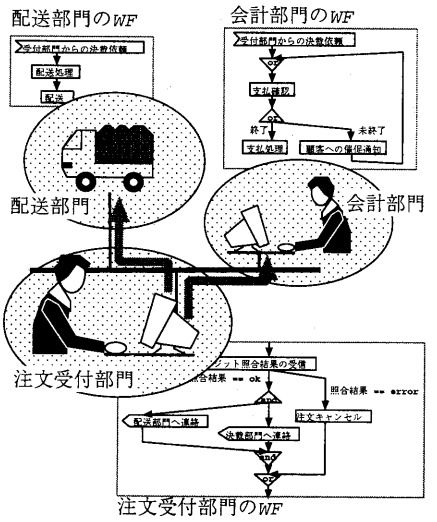


図 6: WFの運用例(通信販売業務)

ここで問題となるのは、組込みによって他部門で例外処理が発生する可能性である。つまり、部門間で作業の引継ぎがある場合、引き継ぎ元の作業が引き継ぎ先の作業へ影響を与えることである。

これに対する解決策として、以下の2つを考えている。

- 追加WFの中のメッセージ送受信記述に制限を与え、メッセージの整合性を保つ。
- 組込みの結果、作業に影響が現れる可能性のある部門に追加WFの内容を通知する。

### 3.3.3 WFの再設計

TORECの記述性が良くても、組込みを繰り返せばWFの閲覧性や整合性を維持することは難しくなると考えている。また、作業者の視点が正しいとは限らないため、グローバルな視点から見た修正が必要な場合もあるであろう。このような場合は、WFの設計者によってWFの再設計が必要だと考えている。

だが、WFにはユーザの経験が蓄積されているため、WFを解析すればユーザ経験が反映されたWFを設計することが可能である。よって、WFの再設計のために、WF追加で得られたWFの解析方法や環境を考察する必要があると考えている。

## 4 おわりに

我々は、WFSにおける例外処理の問題を、ユーザの例外処理経験を作業手順に蓄積することで解決すると考え、"逐次追加法"を提案してきた。本稿では、"逐次追加法"の説明と、"逐次追加法"を実現するシステムTORESについての説明してきた。

TORESの特徴は、シンプルなWF図を用いることで、追加WFの組込みが容易に行なえることである。更に我々は、TORES用のWF図としてTORECを提案し、他の図的表現と比較して評価を行なった。その結果、TORECは他の図的表現に比べてシンプルに作業手順を記述できることが判明した。

"逐次追加法"の評価は、TORESを実作業に適用し、適用データを分析することで行なえると考えている。実作業の適用に際してTORESには、検証機能、分散環境への適合、WF再設計のための方法論の確立、などの課題があることが判明した。

今後は、上記の課題の解決と、TORESの適用評価ならびに"逐次追加法"の評価を行なう予定である。

## 参考文献


[1] 河合, 土屋, "情報共有の窓が開く - 本格化するグループウェア," 日経コミュニケーション, No.206, pp.60–82, (1995).

[2] 飯塚, 桧垣, 平川, "逐次追加によるAP構築手法," 情報処理学会第50回全国大会(6), pp.215–216 (1995).

[3] Harel,D., Lachover,H., Naamad,A., Pnueli,A., Politi,M., Shtul-Trauring,A., "STATEMATE:A Working Enviroment for the Development of Complex Reactive Systems," Proc. of 10th ICSE, pp.396–406, (1988).

[4] Kellner,M.I., Hansen,G.A., "Software Process Modeling:A Case Study," Proc. of 22th Hawaii Conference, pp.175–188, (1989).

[5] 飯塚, 桧垣, 平川, "逐次追加型ワークフローにおける記述方式TOREC の提案," 情報処理学会第52回全国大会(6), pp.339–340 (1996).

[6] ITU:Z.100(1993), "CCITT Specification and Description Language(SDL)," ITU-T Jun. (1994).

[7] Z120(1993), "Message sequence chart(MSC)," ITU-T, (1994).

[8] M.Alford, "SREM at Age of Eight; The Distributed Computing Design System," IEEE computer, vol.18, no.4, pp.36–46, (1985).