逐次追加によるAP構築手法

7K－6

飯塚 京子　桧垣 博章　平川 豊
NTTソフトウェア研究所

1　はじめに

近年の計算機ネットワーク技術の発達によって、多数のユーザが複数のアプリケーションを連動させる大規模システムの構築、運用が進んでいる。このようなシステムでは、設計時点における完全なユーザ要求獲得は困難である。さらに運用中にユーザ要求が次第に変化していくため、それに迅速に追従する必要がある[1]。これらは従来のシステム構築、保守技術で解決することはできない。そこで本論文では、迅速に要求を反映することが可能なAP構築手法を提案する。本論文では、ワークフローシステムを適用対象として考察する。

2　ワークフローシステム

ワークフローシステムとは、伝票処理などの目的の定まった一連の作業を支援するシステムである。システム内には、目的を達成するための作業手順を示すフローが登録されており、ユーザはこのフローに従い作業を行なう。

2.1　問題点

システム運用時には、組織変更などシステム外の要因による変化が発生する。これにより新たな要求が生じ、システムに登録されたフローでは対処できない処理(例外処理)が起り、ユーザはシステム支援を受けることができなくなる。この問題を扱った従来のワークフローシステムには、事例ベース推論機構を備えたOM-1[2]がある。OM-1では例外処理事例を蓄積し、この事例を検索することで例外処理に対応できるシステムを実現した。

本論文ではOM-1と同様に、例外事例をシステムに追加することでこの問題に対処する。ただし、追加する例外事例を新規要求ととらえ、システム登録フローの再構築を行なう点がOM-1とは異なる。これにより、例外処理対応に限らず、新たな機能の追加を容易に行なうことが可能となる。

2.2　ワークフローのモデル

フローの構成単位となる作業をタスクと呼ぶ。タスクには作業内容を特定するラベルと、タスクの実行を開始する条件(開始条件)を付加する。この開始条件によりタスクの実行順序を表現する。例えば、ラベル[伝票起票]のタスクをラベル[決裁]のタスク終了後に実行させる場合、ラベル[決裁]のタスクの開始条件に{ラベル[伝票起票]のタスク終了}を記述すればよい。このようにタスクを組立てたものがフローである。

本論文で扱うフローは、最初に実行するタスク(開始タスク)と最後に実行するタスク(終了タスク)が1つに定まるものとする。開始タスクから終了タスクまでの間には、実行するタスクやタスクの実行順序が異なる、幾通りかの作業手順が存在する。そこで、開始タスクから終了タスクまでの、1通りの作業手順をパスと呼ぶことにする。

2.3　フロー逐次追加法

本論文は、フローの逐次追加によるAP構築手法を提案する(図1)。新規要求により例外処理が発生した場合、ユーザは例外処理の成功経験を1本のパスとしてシステムに入力する。これを追加パスと呼ぶ。システムはこれを登録フローから分岐するように結合する。これにより、次回の作業からは追加パスを利用することが可能となる。

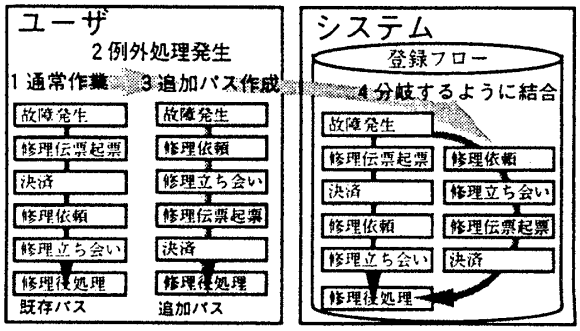

図1: フロー逐次追加システム

2.4　フロー逐次追加の課題

ワークフローシステムでは、作業の効率化、すなわち作業時間の短縮と作業量の縮小が求められる。作業の効率化は、並行なタスク群を抽出し、複数の作業者にタスクを分担させることで実現できる。例えば図1のフローで、ラベル[伝票起票],[決裁]のタスクとラベル[修理依頼],[修理立合い]のタスクが並行ならば、担当者は[修理依頼],[修理立合い]を他の作業者に任せることで、作業にかかる所要時間を短縮する。

システムの運用開始時に登録するフローに関しては、運用前に並行なタスク群を抽出し、作成することができる。しかし例外処理により作成される追加パスでは、ユーザが並行なタスク群に気づきパスを作成するとは限らない。このため追加パスでの並行なタスク群を抽出する必要があるが、上で述べたフロー逐次追加法はではできない。

並行なタスク群の抽出は、同じ例外処理で作成された、複数の追加パスのタスク実行順序を比較することで可能となる。例えば図1の既存パスと追加パスには並行なタスク群は存在しない。しかし、2つのパスを見比べると、2組のラベル[伝票起票],[決裁]と[修理依頼],[修理立合い]間には順序関係が存在せず、並行なタスク群とみなすことができる。ここで本論文では、構成するラベルの集合が一致する複数のパスを、新たな並行性を含む1本のパスへ変換する、コンパクションを提案する。

Constructon Method for New Version Systems by Flow Addition
Kyoko IIZUKA and Hiroaki HIGAKI and Yutaka HIRAKAWA
NTT Software Laboratories

2.4.1 コンパクション

本節ではコンパクション過程の概要を説明する (図2)。まず、全コンパクション対象パスの、全タスクの履歴を抽出する。パスPのタスクtの履歴とは、Pの作業手順で、tより順序が前のタスクのラベルの集合とする。例えば図2の左図を見ると、既存パスでラベル [決裁] のタスクの履歴は、{[故障発生],[伝票起票]} であり、追加パスでラベル [決裁] のタスクの履歴は、{[故障発生],[修理依頼],[修理の立合い],[伝票起票]} である。次に、同一ラベルのタスク同士で履歴を比較し、共通する履歴を抽出する。ラベル [決裁] の共通経歴は、{[故障発生],[伝票起票]} である。更に、各タスクの開始条件に、共通経歴の中のラベルのタスクの終了条件を積算し、新たな開始条件を作る。ラベル [起票] の新しい開始条件は、{[故障発生] 終了 and [伝票起票] 終了} となる。最後に、同じラベルを持つタスクを、新しい開始条件を持つ1つのタスクに集約する。このタスクを組立てると、図2の右図のような、コンパクション対象パスに含まれる全作業手順を含み、既存のパスには存在しなかった並行性含む1本のパスが生成される。

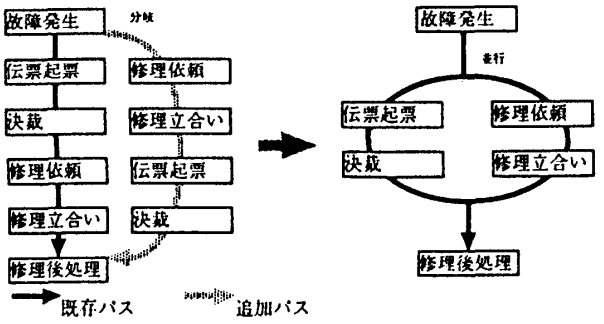

図 2: コンパクション例

2.4.2 拡張順序

コンパクションにより生成されたパスには、以前のパスには存在しない新たなタスクの実行順序を含む場合がある。例えば図3のコンパクション後の右図には、コンパクション前の左図にはない [伝票起票] → [担当者押印] → [予算管理者の伝票チェック] → [契約係の伝票チェック] → [課長押印] というタスクの実行順序が含まれている。このようなタスクの実行順序を、コンパクションにより生成された拡張順序と呼ぶことにする。

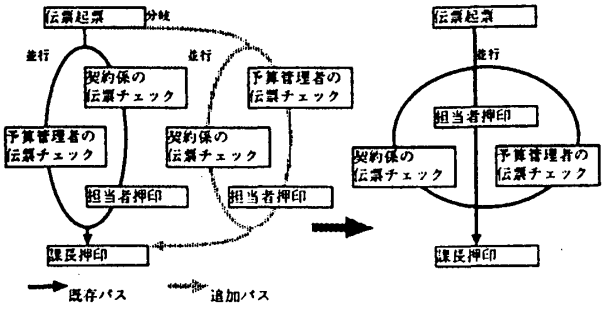

図 3: 拡張順序を持つコンパクション例

拡張順序にはユーザの潜在的な要求が含まれているが、全ての拡張順序がユーザに認められるわけではない。例えば、コンパクション前は担当者印が押される前に、契約係か予算管理者による伝票のチェックが行なわれていたが、拡張順序により誰からの伝票のチェックも受けることなく担当者印が押せるようになってしまう。従って拡張順序が存在する場合は、その妥当性を判定した後にフローの書換えを行なうこととする。拡張順序の妥当性はタスクやパスの意味に依存するため、システムによる判定の自動化は行なえない。そこでコンパクション結果をユーザに提示し、ユーザの許可が得られた場合はシステム登録フローをコンパクション結果に書換えることにする。

2.5 従来技術との比較

並行化に関する従来技術として、並列化コンパイラの研究がある[3]。これは、FortranやCなど逐次型言語で書かれたプログラムを自動的に並列プログラムに変換する技術である。通常のプログラムは、引数により引継がれる値が演算ごとに変化するため、引数の演算順序がソースプログラムと一致するように並列化を行なう。

ここでタスクを並列化の演算単位とし、作業の性質から生じるタスク間の依存関係を引数で表現すると、並列化コンパイラ技術をワークフローに適用できる。タスク間の依存関係の例として、図3のラベル [伝票の起票] と [課長押印] の2つのタスクのように伝票というリソースの制約から生じるものがある。しかし、ワークフローの対象とする世界はソースプログラムのようにシステム内で閉じていないため、適用作業ごとにフローの意味を保存するため記述すべき依存関係を特定する必要がある。これに対しコンパクションでは、依存関係を特定することなく並行化が可能である。ただし、拡張順序が発生するために完全な自動化は困難である。

このように、コンパクションと並列化コンパイラの技術には、それぞれ異なる長所と短所がある。この2つの方式を組み合わせることで、個別方式を採用するよりは容易にフローの意味変質を防止できる並行化が実行可能となる。

3 おわりに

本論文はシステムへの迅速な機能追加を実現するために、ワークフローシステムを適用例として、逐次フロー追加方法を提案した。更に、フロー追加時に並行性の抽出を行なうコンパクションを提案した。これにより、ユーザが気付かない並行なタスク群を抽出することが可能となる。

今後は本手法をネットワーク保守システムに適用し、コンパクションの効果を測定する。また、本論文では対象としていないループを含むフローでのコンパクション方法を確立する。

参考文献

1) 桧垣, 森保, 奥山, 平川, 市川, "システム進化支援環境 リセプティブプラットフォーム," 情報処理学会第47回全国大会, 6E-1, pp.201-202 (1993).
2) 大久保, 石井, "オフィスプロシジャ修正事例に基づくプランニング技法," 人工知能学会研究会資料, SIG-HICG-8801-4 (1988).
3) 本多, 岩田, 笠原 "Fortran プログラム粗粒度タスク間の並列性検出手法," 信学論 (D-1), J73-D-1, 12, pp.951-960 (1990).