Vol.2013-DPS-154 No.26
Vol.2013-CSEC-60 No.26
2013/3/14

ワークフロー実行におけるエラー対処プランの自動推薦システム

三品 拓也[1,a)]　勝野 恭治[1,b)]　串田 高幸[1,c)]

概要：企業や組織がある特定の目的を達成するために実行が必要な，タスクとその実行順序の定義をワークフローと呼ぶ．ワークフローをITシステムで管理できるワークフロー管理システム（WFMS）におけるワークフローのライフサイクルの中で，ワークフローの分析・更新の自動化は先行研究において未解決の課題となっている．そこで本論文では，タスク実行時に発生するエラーに対処するときに，多くのエラー対処プランの中から適切なものを優先的にユーザに提案する推薦システムのアーキテクチャとアルゴリズムを提案する．提案手法によって，ワークフローをより迅速に完了できるように更新することと同様の効果を得ることができる．

1. はじめに

企業や組織がある特定の目的を達成するためには，多くの主体が複数の小さな作業（タスク）を，適切な順序で実行する必要がある．ある目的を達成するための処理手順をビジネスプロセスと呼び，ビジネスプロセスを構成する複数のタスクを条件分岐で連結して，処理順序・処理内容・処理主体を明確化したものをワークフローと呼ぶ[*1]．ワークフローは，完了に至るまでの速度と完了するまでに消費する資源のいずれかもしくは両方が最小となるように，予め熟慮の上で定義しておく必要がある．作成されたワークフローは手順書や作業マニュアルといった形で記録され，複数のワークフローが目的・実行組織などの適切な単位でまとめて保管される．本論文ではこのようなワークフローの保管場所をワークフローレポジトリと呼ぶ．

一般的なワークフローのライフサイクルは，**表 1** に示す4段階からなる．この4段階のうち，作成段階の作業を支援する技術がワークフローマイニング [1] である．ワークフローマイニングは，機械的に記録されたタスクの実行結果（以下「実行ログ」と呼ぶ）からタスクの前置・後続関係を抽出してワークフローを構築する技術である．タスクの前置・後続関係以外にも，ワークフロー実行時にユーザが入力する，自然言語による実行時メモ（[2], [3], [4] ），電子メールのやりとり [5]，グループウェア上でのやりとり [6]，それらに類するアドホックな情報交換ツールの履歴 [7]，データベース上に記録されたトランザクションデータ [8] など，様々なデータを情報源としたワークフローマイニング技術が提案される．マイニング手法そのもののバリエーションとしては，あらかじめ各パターンを表現するオントロジーを定義してそのパターンを探索する手法 [9] を挙げることができる．

実際にワークフローを実行したときに事前には想定できなかった不都合が発見された場合は，その原因を分析してワークフローを更新する必要がある．分析・更新は繰り返し実施されるので，この作業を効率化すると改善の度合いは大きい．前述のワークフローマイニングはあくまで新規作成時に有効な技術であり，分析・更新については触れられていない．分析・更新を行う既存技術としては，抽出したマイニング結果が正しく実行ログを反映しているかどうかを確認する方法 [10] が挙げられる．また，ワークフローと実際の作業内容との乖離を修正する技術 [11] では，ある特定のワークフロー単体について，ワークフロー定義とワークフロー実行結果を比較して乖離がないかを確認することができる．これらの技術はある特定のワークフローについて分析を行うものであって，複数のワークフローで共通に現れる特徴を捉えることができないため，分析の効果に限りがある．ワークフロー同士の関係を発見する方法としては，ワークフローとワークフローの類似度を計算する技術（[12], [13], [14] など，[15] に手法比較）が挙げられる．こ

[1] IBM Research - Tokyo
5-6-52 Toyosu, Koto, Tokyo 135–8511, Japan
[a)] tmishina@jp.ibm.com
[b)] katsuno@jp.ibm.com
[c)] kushida@acm.org

[*1] 一般的な定義としては，例えば [1] においてビジネスプロセスとワークフローは同義語として扱われているが，本論文では処理順序が厳密に定義されたものに対して特にワークフローという用語を使用する．

表 1　ワークフローのライフサイクル

作成	現に存在しているビジネスプロセスの手順を，ワークフローとして明確に定義する
実行	定義したワークフローに従って作業を実施する（ひとつひとつの実施例をプロセスインスタンスと呼ぶ）
分析	実施結果を分析して非効率な手順が存在しないかを確認する
更新	非効率な手順が発見されたら，それを回避するように元のワークフローを書き換える

れらの技術を使えば，人手でワークフローを更新するにあたって参考となる類似のワークフローをワークフローレポジトリから探し出すことが容易になるものの，類似検索技術で更新そのものを行うことはできない．すなわち，ワークフローの更新作業を効果的に行う直接的な研究は先行研究では行われておらず，未だ解決できていない課題であると言える．

そこで本論文では，表 1 に示した 4 段階のうち，分析・更新段階について新たな提案を行う．具体的には，実行記録から，「エラー処理プラン」（2.3 節参照）と呼ばれる，ワークフローから独立してモジュール化されたタスクの利用状況を分析し，ワークフロー内のある特定のタスクでエラーが発生した場合に，そのタスクにおいて利用価値が高いエラー処理プランを推薦するためのアルゴリズムとアーキテクチャを提案する．この推薦システムは，ひとつのエラー処理プランを複数のワークフローの適切な位置に組み込む作業であるとみなすことができる．これによって，効率のよいワークフローパターンを，多くのワークフローにおいて援用することができるので，組織全体で作業内容の標準化されて業務効率が改善されることが期待される．

2. ワークフローと管理システム

本節では，提案手法を説明するために必要な，ワークフローとその管理システムについて詳細に定義する．

2.1 ワークフローの定義とライフサイクル

前節で述べたように，ワークフローとは，複数のタスクの順序・条件を定義して処理の流れを明確化したものである．

図 1 は，ワークフローとそれを構成するタスクの属性を示したものである．ワークフローは複数のタスクからなり，各タスクには，処理すべき内容（作業内容）と処理を行う担当者名，次に行うべきタスク（後続タスク定義）が記述される．ここで作業内容は，自然言語で書かれた人間への指示であることもあるし，外部のプログラムを自動起動するためのコマンドラインでもあり得る．カテゴリとは，ワークフローマネージャが，実施したいと思っている作業に相当するワークフローをワークフローリポジトリ（2.2 節の**図 3** で説明）から検索するときに使われるメタ情報であり，ワークフロー設計者がワークフロー作成時に付与するものである．カテゴリの付与は設計者に多少の手間を生じさせるが，ある組織が持っているビジネスプロセスの数（すなわちワークフローの数）は数十以上になることがふつうであり，何らかの形で検索のためのメタ情報は必要である．

表 1 で示したワークフローのライフサイクルは，**ワークフロー設計者**がビジネスプロセスに基づいて作成することから始まる．作成されたワークフローは**ワークフロー実施者**によって実行開始される．これをワークフローをインスタンス化する，と呼び，ひとつひとつのインスタンスをプロセスインスタンスと呼ぶ．ワークフローの各タスクは**処理担当者**によって実施される．ここで処理担当者とは，人である場合と機械（スクリプトや URL アクセス）である場合の両方があり得る．人手である場合は，タスクに記述されたロールに属するユーザの中から処理担当者を選択する．何度かプロセスインスタンスが実行されたワークフローは，ワークフロー設計者によって再度内容が見直され，必要であれば更新される．

ワークフローの具体例を**図 2** に示す．このワークフローの目的は「web アプリケーションサーバを構築する」ことであり，作業の具体的な内容に応じてどの作業をどの順序で実行すべきかが明確になっている．

2.2 ワークフロー管理システム

表 1 に示したワークフローのライフサイクルは，全て人手のみで実施可能である．実行段階を例に挙げれば，ワークフローマネージャが手順書からワークフローの定義を読み取り，次に行うべきタスクを理解し，担当者を決めて電話やメールで通知し，各担当者が作業を完了したかを確認し，次のタスクを決定して再度担当者へ通知する，という作業を繰り返すことで，人力でも実施することができる．しかし，複数のワークフローが同時並行で実行される状況下でこのような管理の全てを人間が行うのは大変な手間であるから，ワークフローを適切かつ効率的に実行するためには，何らかの IT システムの力を借りることが現実的である．ワークフロー管理システム（Workflow Management System; WFMS）とは，ワークフローの実行と，それに付随してワークフローの作成・更新をサポートする IT システムである．

本論文で想定するワークフロー管理システムを図 3 に示す．ワークフロー管理システムは，6 つのコンポーネントからなる．ワークフローレポジトリからワークフローを読み出して，ユーザに適切な通知を与えたりツールを自動実行することで定義通りにワークフローを実行する実行エン

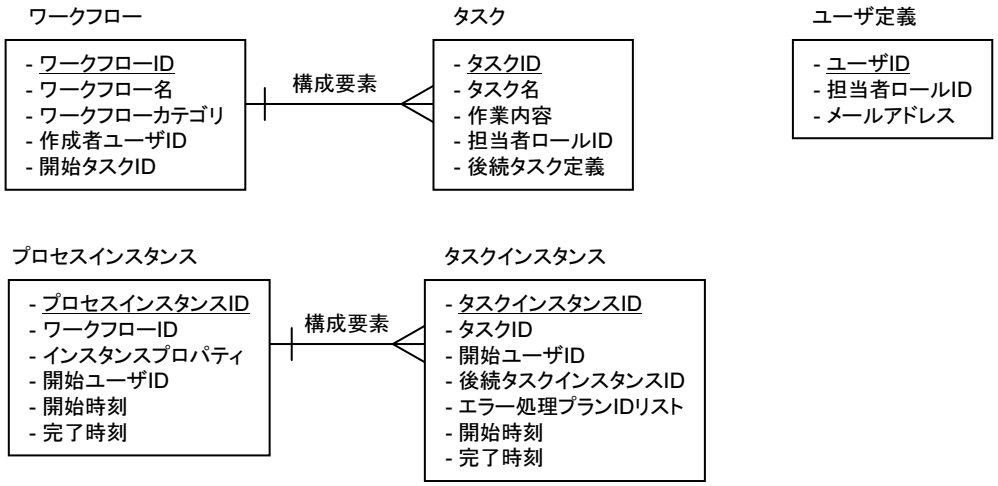

図 1 ワークフロー及びタスクの属性及び関係

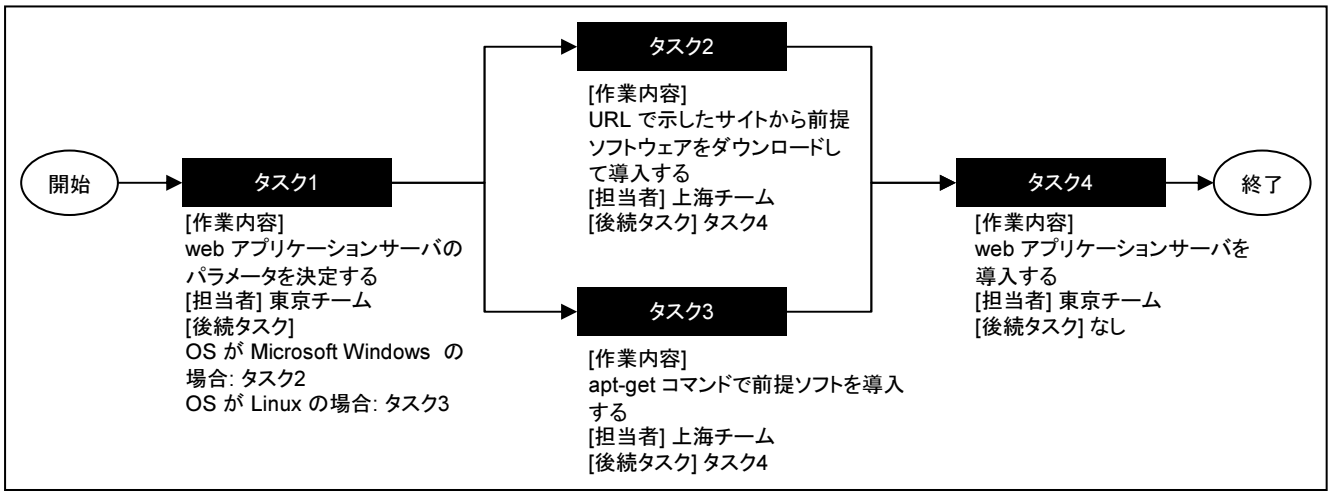

図 2 ワークフローの例

ジンを中心として，ワークフローを設計するためのワークフローエディタ，作成したワークフローを保管するワークフローレポジトリ，ユーザとのやりとりを行う UI (User Interface) 部分，ユーザ定義，ログがそれらに付随している．なお，点線囲い部分はこのあと 3.2 節（**図 7**）で拡張する部分である．

ワークフロー管理システム下におけるワークフローのライフサイクルは以下のようになる．

(1) ワークフロー設計者が，ワークフローエディタを使って既存のビジネスプロセスを新規ワークフローとして実装する．このとき，ワークフローにカテゴリを付与する．
(2) ワークフローマネージャが UI にアクセスし，ワークフロー名やカテゴリをキーにして，実行したい作業に応じたワークフローをワークフローリポジトリ内から探し，ワークフローの開始を指示してプロセスインスタンスを生成する．このとき，プロセスインスタンスに特有な情報（プロセスインスタンスプロパティ）を付与する．
(3) 実行エンジンが UI を呼び出し，担当者に対してメールでタスクを通知するか，プログラムを実行する．
(4) 担当者は通知されたタスクを実施し，作業が完了した場合は UI を使って実行エンジンに完了を通知する．担当者がプログラムの場合，実行結果の可否を実行エンジンに返す．
(5) 実行エンジンがワークフロー定義に従って次に実行すべきタスクを決定する（3 に戻って繰り返し）．もしも次に実行すべきタスクが存在しない場合はワークフローの完了を意味するので，実行エンジンがワークフローマネージャに完了を通知する
(6) ワークフロー設計者が，何度か実行されたワークフローについて，内容が適切かどうか分析し，必要に応じてワークフローを更新する

2.3 エラー処理プラン

2.2 節で掲げたワークフローのライフサイクルでは，タス

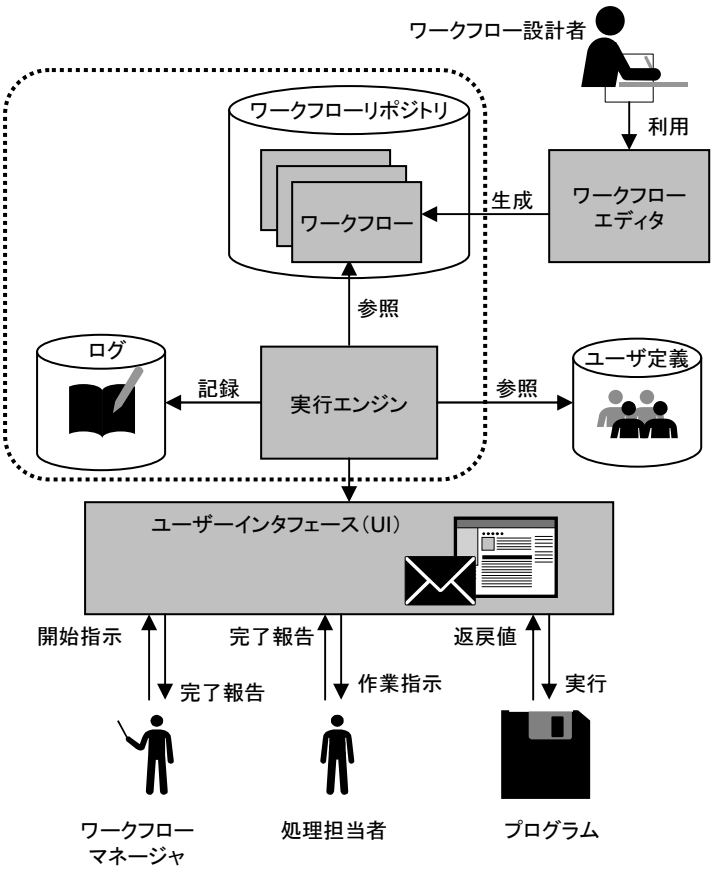

図 3 ワークフロー管理システムの構成

クは必ず「完了」することを前提としているが，実際には何らかの理由で完了に至らない可能性がある．例えば「web アプリケーションのインストーラをダウンロードする」というタスクがあって，指示内容として URL が記載されている場合，この URL がリンク切れを起こしていた場合はタスクを完了することができない．ワークフローを定義するにあたっては，このような例外的な事項が起こった場合にどう対処するのかという点に関しても明確に記述しておく必要がある．例えば，「URL がリンク切れであった場合は，ワークフロー設計者に連絡して正しい URL を受け取り，その URL を使ってタスクの実行を続行せよ」といった記述である．

WFMS の中には，このような問題発生時の対処手順を記述したエラー処理プランを，一般的なワークフローとは区別して管理できるものが存在する．例えば [16] では，あるワークフローの実行途中でエラーが発生した場合，ユーザは「サービスカタログ」と呼ばれる特別な指示書に登録された手順を参照して解決方法を見つけることができるしくみになっている．本論文ではこの特別な手順のことをエラー処理プランと呼ぶ．エラー処理プランは，プログラミング言語における例外処理に類似したものと考えることができ，発生頻度は稀であるが複数のワークフローにおいて利用可能なワークフローをモジュール化して切り出したもの

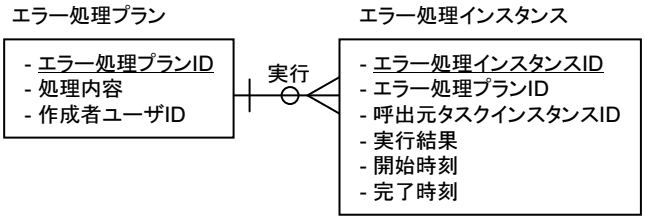

図 4 エラー処理プランとそのインスタンスの属性

と捉えることができる．本論文におけるエラー処理プランとそのインスタンスが持つ属性を**図 4** に示す．

エラー処理プランにはいくつかの特徴がある．まずライフサイクルの視点で見ると，エラー処理プランは，それを作るのが処理担当者であって，作るタイミングはエラーが発生した時点であって，その数はエラー処理の分だけ増加していく，という点が挙げられる．これはすなわち，ワークフロー設計者がビジネスプロセスをもとに一般的なワークフローを作る場合と異なり，カテゴリのようなメタ情報を付与する手間をかけられないことを意味する．エラー処理プランに与えられるメタ情報は，そのエラー処理プランがどのワークフローを実行している最中に発生したエラーを解決するために作られたのか，といった，明示的に入力する必要のないものに限定される．実行時の特徴としては，エラー処理プランは他のワークフローの途中で呼び出

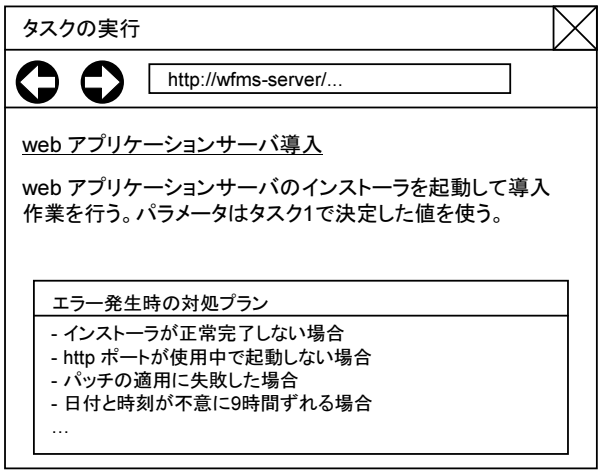

図 5 タスク指示画面におけるエラー処理プランの表示例

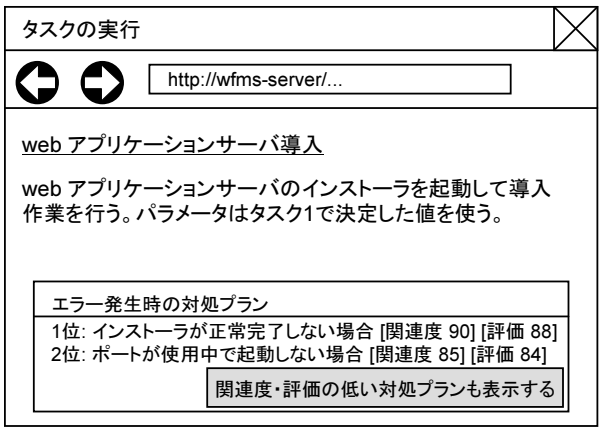

図 6 推薦機能付きエラー処理プラン選択画面を備えたタスク指示画面の例

されて，完了したときに元のワークフローに戻るという点が挙げられる．これは，一般的なワークフローがビジネスニーズに応じて作成され，それに基づくのプロセスインスタンスが完了したときにはなんらかの成果を得て一連の作業が終了することと対照的である．

2.4 エラー処理プラン活用の課題

エラー処理プランは，プログラミングを例に挙げれば，例外処理を切り出したモジュールであるとみなすことができる．ひとつのエラー処理プランを複数のタイミングで使えば，車輪の再発明をすることなく最良の解決方法を何度も繰り返し活用ができる反面，前節で述べたようにエラー処理プランの数は必然的に多くなるため，あるタスクで問題が発生した時にどのエラー処理プランを利用すべきなのかわからなくなる恐れが強い．**図 5** はエラー処理プランの提示例を示したものであり，ここで対処プランの数が多くなると，人手で適切なものを発見するのが徐々に困難になる．2.3 節で示したように，エラー処理プランはあるワークフローのあるタスクを処理しているときにエラーが発生した際に作られるので，エラーの発生元タスクの時点で利用できるのは自明であるが，同じワークフロー内の他のタスクや，別のワークフローにおいても同様に利用できるかは一意に決められない．あるタスクで問題が発生した時に使うべきエラー処理プランを決定する手段は，現状の WFMS[16] では，担当者がエラー処理プランの名称や，それに添えられた解説文を見て自力で判断する方法に限られており，エラー対処プランを作成した担当者が適切なプラン名称を与えなかった場合，使う側では判断が困難である．そこで，何らかの基準で利用価値の高いエラー処理プランを処理担当者に推薦する仕組みを用意し，適切なエラー対処プランが適切な箇所で再利用されるシステムが求められる．

3. エラー処理プランの推薦システム

3.1 提案システムの概要

本論文では，既存のワークフロー管理システムにおけるエラー処理プラン選択のユーザインタフェース（図 5）を，**図 6** のように拡張する．ここで各エラー処理プランの表示順序は，3.3 節で述べるアルゴリズムによって得られる関連度と効果によって決定される．表示されるエラー処理プランは一定以上の評価を得たものに限り，それ以外のエラー対処プランを隠蔽する．これによって，ユーザが確認すべきエラー対処プランの候補が少なくなり，選択が容易になる．

3.2 アーキテクチャ

提案する仕組みの全体アーキテクチャを図 7 に示す．図 3 に示したワークフロー管理システムの点線部分を改変し，タスクインスタンスにおいてエラー処理プランを選択する際に，推薦システムの結果を参照して UI に反映させる．

3.3 エラー処理プランの関連度・効果計算

あるエラー処理プラン e がタスクインスタンス τ に対してどの程度適合しているかを以下のように定義する．ただし，エラー処理プラン ID が e と等しいエラー処理インスタンスの集合（すなわち e の実行履歴の集合）を E とする．

3.3.1 関連度（カテゴリ合致性）計算

本論文では，「あるタスクインスタンスと関連性が高いのは，そのタスクインスタンスが属するワークフローのカテゴリと似たカテゴリのワークフローで使われたエラー処理プランである」という仮定に基づいて関連度を計算する．具体的には，τ のプロセスインスタンス ID から τ が属するワークフローを特定し，そのワークフローに付与された

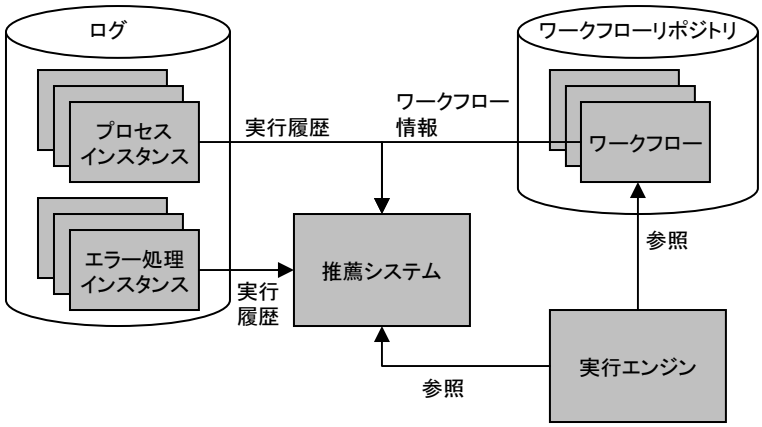

図 7 アーキテクチャ

カテゴリを $C^\tau = (C_1^\tau, C_2^\tau, ..., C_N^\tau)^T$ とすると，これと同様に，E の各要素属するカテゴリから E 全体のカテゴリの傾向を表す $\bar{C}$ についても以下のように定義できる．

$$\bar{C} = \frac{\sum_{e_i \in E} (C_1^\tau, C_2^\tau, ..., C_N^\tau)^T}{||E||} \quad (1)$$

ここから e の τ に対するカテゴリ合致度は，2 つのベクトル C^τ と $\bar{C}$ のなす角より

$$\cos\theta = \frac{C^\tau \cdot \bar{C}}{||C^\tau||\,||\bar{C}||} \quad (2)$$

で得られる．

3.3.2 効果指標計算

本論文では，以下のような評価値が大きなエラー処理プランを「効果が高い」とみなす．

エラー処理プランを実行したタスクインスタンスの 1 回当たりの平均処理時間

あるプロセスインスタンスにおいて e を実行した場合に，その呼出元となったタスクインスタンスの完了に費やした時間の平均 m_e^T と，e 以外のエラー処理プランを実行した場合に，その呼出元となったタスクインスタンスの完了に費やした時間の平均 $m_{\bar{e}}^T$ の差が大きい場合は優れたエラー処理プランであると言える．e 以外のエラー処理プランを実行したエラー処理インスタンスの集合を $\bar{E}$，あるエラー処理プラン e が実行されたタスクインスタンスの開始時刻と完了時刻をそれぞれ $t_s(e)$, $t_c(e)$ とすると，求める指標は

$$m_e^T - m_{\bar{e}}^T = \frac{\sum_{e \in \bar{E}} t_c(e) - t_s(e)}{||\bar{E}||} - \frac{\sum_{e \in E} t_c(e) - t_s(e)}{||E||} \quad (3)$$

で表すことができる．

エラー処理プランを実行したワークフローの平均所要時間

エラー処理プランを実行したタスクインスタンスの 1 回当たりの平均処理時間が最小であっても，そのエラー処理プランによって後続のタスクで別のエラーを発生させてワークフロー全体の完了時間を遅らせているものがあれば効果は低いと言える．そこで以下の式のように，エラー処理ログに記録されたタスクインスタンス ID から再帰的に後続タスクを全て検索し，エラー処理プランが実行されてからワークフローが完了するまでの 1 回の処理時間の平均値を取る．あるプロセスインスタンスにおいて e を実行した場合におけるワークフロー完了までの平均処理時間を m_e^P，e 以外のエラー処理プランを実行した場合におけるワークフロー完了までの平均処理時間を $m_{\bar{e}}^P$ とすると，この差が大きい場合に e は優れていると言える．e が実行されたプロセスインスタンスにおいて，e が実行されたタスクインスタンス及び後続インスタンスの集合を T，同様に e 以外のエラー処理プランが実行されたプロセスインスタンスにおいて，$\bar{e}$ が実行されたタスクインスタンス及び後続インスタンスの集合を T としたときのタスクインスタンス及び後続インスタンスの集合を $\bar{T}$ とし，あるタスクインスタンス τ の開始時刻と完了時刻をそれぞれ $t_s(\tau)$, $t_c(\tau)$ とすると，求める指標は

$$m_e^P - m_{\bar{e}}^P = \frac{\sum_{\tau \in \bar{T}} t_c(\tau) - t_s(\tau)}{||\bar{T}||} - \frac{\sum_{\tau \in T} t_c(\tau) - t_s(\tau)}{||T||} \quad (4)$$

である．

エラー処理プランによるエラー回復の成功・失敗

あるプロセスインスタンスにおいて e を実行することでエラーが解消された場合は効果が高く，e を実行しても引き続きエラーが解消しない場合は効果が低いと言える．エラーが解消したか否かはエラー処理インスタンスの実行結果属性から取得する．

エラー処理プランが利用された回数

e が利用された回数が多ければ，e は優れたエラー処理プランであると言える．e の利用回数はエラー処理インスタンスの数 $||E||$ である．

4. まとめ

本論文では，ワークフローリポジトリ内に保管されている複数のワークフローの実行記録を使って，ワークフローの実行時に最適なエラー処理プランを推薦するためのアーキテクチャとアルゴリズムを提案した．次の段階として，まず 3 節の推薦システムをプロトタイプとして実装する．このプロトタイプに，実際に業務で用いられているワークフロー管理システム上からワークフロー・エラー処理プラン・実行ログを与えて予備実験を行い，3.3 節で挙げた評価指標から具体的な一本の評価値を算出するためのパラメータを決定する．そののちこの推薦システムを実業務のワークフローシステムに組み込んで，エラー処理フローの利用回数の増加とプロセスインスタンスの所要時間減少を実現できるかの検証を行いたい．

参考文献

[1] van der Aalst, W. M. P., van Dongen, B. F., Herbst, J., Maruster, L., Schimm, G. and Weijters, A. J. M. M.: Workflow mining: a survey of issues and approaches, *Data and Knowledge Engineering*, Vol. 47, p. 2 (online), DOI: http://dx.doi.org/10.1016/S0169-023X(03)00066-1 (2003).

[2] Esposito, P. M., Vaz, M. A. A., Rodrigues, S. A. and de Souza, J. M.: MANA: Identifying and Mining Unstructured Business Processes, *Proceedings of the Workshop on Business Process Intelligence*, pp. 199–204 (2012).

[3] Qiao, M., Akkiraju, R. and Rembert, A. J.: Towards efficient business process clustering and retrieval: combining language modeling and structure matching, *Proceedings of the 9th international conference on Business process management*, Berlin, Heidelberg, Springer-Verlag, pp. 199–214 (online), available from ⟨http://dl.acm.org/citation.cfm?id=2040283.2040303⟩ (2011).

[4] Brander, S., Hinkelmann, K., Hu, B., Martin, A., Riss, U. V., Thönssen, B. and Witschel, H. F.: Refining process models through the analysis of informal work practice, *Proceedings of the 9th international conference on Business process management*, Berlin, Heidelberg, Springer-Verlag, pp. 116–131 (online), available from ⟨http://dl.acm.org/citation.cfm?id=2040283.2040298⟩ (2011).

[5] Sun, P., Tao, S., Yan, X., Anerousis, N. and Chen, Y.: Content-aware resolution sequence mining for ticket routing, *Proceedings of the 8th international conference on Business process management*, Berlin, Heidelberg, Springer-Verlag, pp. 243–259 (online), available from ⟨http://dl.acm.org/citation.cfm?id=1882061.1882085⟩ (2010).

[6] Ciccio, C. D., Mecella, M., Scannapieco, M. and Zardetto, D.: Groupware Mail Messages Analysis for Mining Collaborative Processes, *SEBD*, pp. 397–404 (2011).

[7] Martinho, D. and Silva, A.: A Recommendation Algorithm to Capture End-Users' Tacit Knowledge, *Business Process Management* (Barros, A., Gal, A. and Kindler, E., eds.), Lecture Notes in Computer Science, Vol. 7481, Springer Berlin Heidelberg, pp. 216–222（オンライン）, DOI: 10.1007/978-3-642-32885-5_17 (2012).

[8] Goel, S., Bhat, J. M. and Weber, B.: End-to-End Process Extraction in Process Unaware Systems, *Proceedings of the Workshop on Business Process Intelligence*, pp. 162–173 (2012).

[9] Ferreira, D. R., Alves, S. and Thom, L. H.: Ontology-Based Discovery of Workflow Activity Patterns, *Proceedings of the Workshop on Reuse in Business Process Management*, pp. 314–325 (2011).

[10] Fahland, D. and van der Aalst, W. M. P.: Repairing process models to reflect reality, *Proceedings of the 10th international conference on Business Process Management*, Berlin, Heidelberg, Springer-Verlag, pp. 229–245 (online), DOI: 10.1007/978-3-642-32885-5_19 (2012).

[11] Gambini, M., La Rosa, M., Migliorini, S. and Ter Hofstede, A. H. M.: Automated error correction of business process models, *Proceedings of the 9th international conference on Business process management*, Berlin, Heidelberg, Springer-Verlag, pp. 148–165 (online), available from ⟨http://dl.acm.org/citation.cfm?id=2040283.2040300⟩ (2011).

[12] Kunze, M., Weidlich, M. and Weske, M.: Behavioral similarity: a proper metric, *Proceedings of the 9th international conference on Business process management*, Berlin, Heidelberg, Springer-Verlag, pp. 166–181 (online), available from ⟨http://dl.acm.org/citation.cfm?id=2040283.2040301⟩ (2011).

[13] Pittke, F., Leopold, H., Mendling, J. and Tamm, G.: Enabling Reuse of Process Models through the Detection of Similar Process Parts, *Business Process Management Workshops*, pp. 586–597 (2012).

[14] Ekanayake, C. C., Dumas, M., García-Bañuelos, L., La Rosa, M. and ter Hofstede, A. H. M.: Approximate clone detection in repositories of business process models, *Proceedings of the 10th international conference on Business Process Management*, Berlin, Heidelberg, Springer-Verlag, pp. 302–318 (online), available from ⟨http://dx.doi.org/10.1007/978-3-642-32885-5_24⟩ (2012).

[15] Becker, M. and Laue, R.: Analysing Differences between Business Process Similarity Measures, *Business Process Management Workshops (2)*, pp. 39–49 (2011).

[16] Tivoli Service Request Manager. `http://www-06.ibm.com/software/jp/tivoli/products/tsd/`.