情報処理学会論文誌 Vol.59 No.4 1150–1160 (Apr. 2018)

システムログと組織情報を活用した業務フロー仕様復元

三部 良太[1,2,a)] 田中 匡史[1] 小林 伸悟[3] 小林 隆志[2]

受付日 2017年8月2日, 採録日 2018年1月15日

概要：昨今の業務システム開発では，既存システムが存在し，その既存システムの仕様の一部踏襲を求められることが多い．しかし，既存システムには最新の仕様書が残っていることが少なく，踏襲すべき仕様の特定が困難になっている．この課題に対して，ログなどを用いて操作の流れのフローを復元する取り組みはあるが，ユーザと仕様の議論をするには粒度が細かすぎて，上記の用途には適さない．本稿では，既存システムからの業務フロー仕様の復元を目的として，システムの操作ログから詳細な業務フローを復元する手法および操作したユーザの組織情報を用いて業務フローの粒度を抽象化する手法を提案する．操作ログの一連の操作をクラスタリングし，代表となる操作を抽出することで，業務の全体像の俯瞰に適した抽象化を実現する．提案手法を実際のシステムで適用した結果に基づき手法の有効性を議論する．

キーワード：業務フロー仕様復元，システムログ，組織情報，システムリプレース

Business Process Recovery Based on System Log and Organizational Structure Information

Ryota Mibe[1,2,a)] Tadashi Tanaka[1] Shingo Kobayashi[3] Takashi Kobayashi[2]

Received: August 2, 2017, Accepted: January 15, 2018

Abstract: In most cases of current enterprise system development, the requirement specifications should follow ̊typo2-1the ones of existing legacy systems. However, it is difficult to identify high level specifications such as business process steps from legacy and undocumented systems. In this paper, we propose a method to recover an abstract business process by using system logs and the organizational information of the operators using an existing legacy system. Our proposed method provides a hierarchical view based on a clustering technique to find abstract activities that consist of a series of operation. We also proposed a method to extract a representative operation in a cluster. We discuss the effectiveness of our method by looking at experimental results on a real system.

Keywords: business process recovery, system log, organizational structure, system modernization

1. はじめに

近年，企業の活動においてシステムが導入されている割合は約7割といわれている [1]．企業において業務の多くは変化しないため，新たに開発する仕様はすでに利用しているシステム（以下，既存システム）の仕様の一部を踏襲することが求められる．しかし，古くから運用している既存システムの仕様は明文化されていることが少ない [2]．そのため，既存システムの仕様の理解に多くの人的コストが必要となり，システム開発コストを圧迫することとなる．

この課題を解決するために，様々な仕様復元手法が提案されている．これらの手法は，ソースコードを入力とする静的解析に基づくものと，実行ログなどの実行時情報を入力とする動的解析に基づくものに大きく分類される．静的

[1] 株式会社日立製作所研究開発グループシステムイノベーションセンタ
Center for Technology Innovation, Systems Engineering, Research & Development Group Hitachi, Ltd., Yokohama, Kanagawa 244–0817, Japan
[2] 東京工業大学大学院情報理工学研究科計算工学専攻
Department of Computer Scinece, Tokyo Institute of Technology, Meguro, Tokyo 152–8550, Japan
[3] 日本エクス・クロン株式会社
Japan EXpert Clone Corporation, Shinjuku, Tokyo 160–0022, Japan
[a)] ryota.mibe.mu@hitachi.com

解析に基づく手法は，プログラムの静的構造と関数呼び出し関係などの依存関係に基づきドメインモデルやソフトウェアアーキテクチャの復元を行う [3], [4], [5], [6], [7]．これらは，ソースコードを用いることで網羅的に復元できる一方で，ライブラリやミドルウェアなどを利用するソフトウェアに対しては十分な復元ができないという課題がある．また，ワークフロー実現のために多数の機能を有するシステムに対しては，機能間の関係は静的に判断できない場合が多く，システムがどのように利用されるかを表す業務フローを復元することが困難である．

動的解析に基づく手法 [8], [9], [10], [11] は，システム利用における実行時情報に基づくため，業務フロー復元に必要な情報を取得することができる．

実行時情報は一般に膨大な量となるため蓄積および解析方法の効率化が課題となる．また，ログに含まれる操作を分析の単位とするため，粒度が細かすぎてしまう．このためシステムが実現する業務の全体像を把握することが困難となるため，ユーザとの仕様の議論で用いるには適さない．

本稿では，低コストで蓄積可能な運用時のシステムログに着目し，システムログとシステム利用者の組織情報を用いることで，業務の全体像を俯瞰する業務フローを復元する手法を提案する．運用時のシステムログを解析することにより，そのプロセスに関与するアクター（以下，ユーザ）の網羅的な実行時情報を取得し，それを分析することで抽象化したアクティビティを復元する．また，組織情報に基づいてアクティビティを階層化することにより全体像の俯瞰に適した抽象化を行う．さらに提案手法を実際のシステムに適用したケーススタディで有効性を評価する．

以降，2 章では関連研究について説明し，3 章では提案手法が想定するシステムとそのシステムログについて説明する．4 章では提案手法，5 章では提案手法を実現したプロトタイプについて述べる．6 章では，実際の業務アプリケーションに適用したケーススタディについて説明し，7 章では結論と今後の課題について述べる．

2. 関連研究

システムログからフローを復元する取り組みは van der Aalst によるビジネスプロセスマイニングの研究がある [12]．これは，ログに含まれる操作の系列から操作間の関係をビジネスモデルとして復元する．しかし，前章で述べた目的で業務を俯瞰するためには，ログに出現する操作の単位では細かすぎてしまい俯瞰することには適さない．

これに対して，ログデータ全体をいくつかに分類し，分類されたログごとにフロー図を生成する方法 [13] や，クラスタリング手法などを適用することで複雑さを軽減する技術 [14] が研究されている．例えば，ActiTraC [13] では，能動学習の考え方を応用し，ログをベクトル空間モデルに変換し，トレース間の距離と出現頻度をもとに選択的サンプリングを行うことで，ログデータのクラスタリングを行う．クラスタリングされたログデータを既存のビジネスモデルマイニングに適用することによって再現性を大きく低下させることなく，複雑さを軽減させている．また，Francescomarino らは Web アプリケーションを対象として，実行トレースから復元された詳細なフローを GUI 情報と繰返し構造に着目したクラスタリングを適用して抽象的なフロー表現する BPMN モデルの復元手法 [14] を提案している．

我々も，これまでにログデータに含まれるユーザに着目し，抽象化を行うアルゴリズム [15] を提案し，業務フロー生成工数が削減することを示した [16]．本稿で提案する手法では，システムログに加えて，ユーザとその組織情報を用いることで業務の実態に即した業務フローを復元する点が異なる．

3. 想定するシステムとシステムログ

3.1 想定するシステム

本研究で想定するシステムは，ユーザインタフェースをともない，多くのユーザが連携して行う業務を支援することを目的とした業務システムである．作業ごとに複数のユーザがそれぞれの担当範囲（ロール）を分担し，処理対象が引き回されることで処理が行われるシステムを想定している．これは，業務システムのカテゴリ [17], [18] ではチャネル系（お客様とのやり取りのインタフェースを担うシステム）や，バックオフィスまたは基幹系（社内の事務処理を行うシステム）でよく見られる．これらのシステムでは，ユーザによっては状況に応じて複数のロールを担当することがある．各ユーザは職務に応じた組織（営業部，在庫管理部など）に所属し，組織は階層的な組織構造を持つ．

3.2 システムログ

システムログはシステム運用時における利用状況を記録したものである．大規模システムでは，システムの障害時に状況を把握したり，サービスレベルの計測に用いられたりするために一定期間残しておくことが一般的である．システムログには次の (1)〜(4) の情報が含まれていることを想定する．**図 1** に (1)〜(4) の具体的な情報が含まれたシステムログの例を示す．

(1) `Time stamp`：この操作が行われた時刻を示す．
(2) `User ID`：この操作を行ったユーザを示す．後に述べるユーザ情報と対応付けるための情報でもある．
(3) `Action ID`：新規作成，登録，承認，修正など，行った操作の種類を示す．
(4) `Case ID`：案件番号，登録番号，受付番号，注文番号など，操作を行った対象（以降，案件と呼ぶ）を示す．

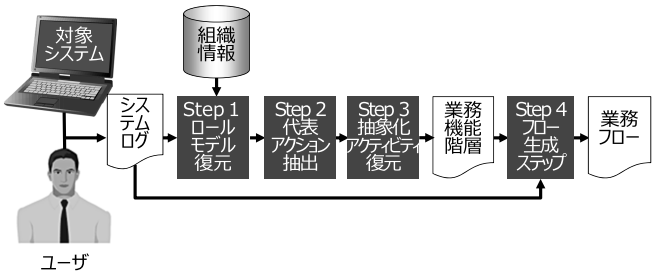

図 2 提案手法の概要
Fig. 2 Overview of the proposed method.

Time Stamp	User ID	Action ID	Case ID
7/1 10:00:00:00	佐藤	見積登録	00100
7/1 10:00:01:00	佐藤	登録エラー	00100
7/1 10:00:02:00	佐藤	見積登録	00100
7/1 10:00:05:00	佐藤	見積承認依頼	00100
7/1 10:00:06:00	小泉	見積登録	00101
7/1 10:00:06:00	小泉	見積承認依頼	00101
7/1 11:00:05:00	鈴木	承認依頼確認	00100
7/1 11:00:07:00	鈴木	見積承認	00100
7/1 11:00:05:00	鈴木	承認依頼確認	00101
7/1 11:00:07:00	鈴木	見積承認	00101
…	…	…	…

図 1 システムログの例
Fig. 1 Sample of system log.

3.3 組織情報

業務フローを階層化する際には「事業レベルから担当者が行う作業レベルまで，階層的に業務の流れを単純化して表現」[19] する．この階層を復元するために組織情報を用いる．組織情報は User ID で識別されるユーザの所属する組織構造（会社，事業部，部，課，担当など）やロールを示す情報である．組織情報は木構造で表現されることが多い．ユーザは必ず 1 つ以上の組織に属し，1 つ以上のロールを持っている．ユーザの担当は，システムのユーザ権限で定義されることも多く，この情報を利用してロールを復元する．

3.4 対象とする業務フロー

業務フローの復元にあたり，業務とは何かを定義する．本稿では，業務を「1 つの目的に対して特定のユーザや組織が行う一連の活動」と定義する．この定義は van der Aalst らが発表したプロセスマイニングマニフェスト [20] で示されている activity の定義（“a well-defined step in some process, related to a particular case”）よりも粒度が大きい概念である．上記の定義において「1 つの目的」がシステムログ上の Case ID に対応し，「ユーザや組織」が User ID と組織情報に対応する．本稿では，この性質を利用して抽象度の高い業務フローを復元する．

業務フローの抽象化は，複数のアクションから抽象化したアクティビティ（抽象化アクティビティ）を抽出することで抽象化が行われる．しかし，複数の組織にまたがる大規模な業務では，アクティビティレベルの抽象化だけでは全体を俯瞰するには不十分である．そこで，本稿では，抽象化アクティビティに加えてロール，組織も復元することでこのような大規模な業務も対象とする．

本稿で提案する技術の目的は，「大規模なシステムの仕様をマクロ的な視点から俯瞰して理解する」ことにある．この目的を達成するためには，下記の要件を定めた．

要件 (1) ビジネスプロセスに登場する多数のアクションを数個からなる抽象化アクティビティにグループ化する．

要件 (2) 抽象化アクティビティとアクションの関係が明確になっている．

要件 (3) 1 つの抽象化アクティビティに関係するアクション群は，業務の一部としてユーザが行うアクション列に対応する．

要件 (4) 抽象化したアクティビティや抽象化アクティビティ間の遷移は，極力抜けや漏れがないものにする．

なぜなら，要件 (1) は Cowan の「成人のワーキングメモリの容量は 3〜5 項目」という知見 [21] により，多くのアクションを含む業務フローを一度に理解することが困難であるためである．また，牧野の「ビジネスプロセス・フロー分析では，事業レベルから担当者が行う作業レベルまで，階層的に業務の流れを単純化して表現」というビジネスプロセス分析の一般的な手法 [19] にも則っている．要件 (2) と要件 (3) は全体を理解することが目的としているため，抽象化したフローで全体を俯瞰したあと最終的にはブレイクダウンしてアクションレベルでの理解も可能にするための要件である．要件 (4) は人が物事を理解するときには，与えられた情報から存在しないものを見つけるよりも，不要なものを見つける方が容易である．このために必要な情報をできるだけ復元するために必要な要件である．

4. 提案手法

4.1 提案手法の概要

図 2 に本稿で提案する業務フロー抽象化手法の概要を示す．提案手法は次のステップで構成される．入力情報からロールを復元する「Step1：ロールモデル復元ステップ」，復元したロール内のアクション群からそのロールを代表す

るアクションを抽出する「Step2：代表アクション抽出ステップ」，抽出した代表アクションに他のアクションを紐づける「Step3：抽象アクティビティ復元ステップ」，復元したアクティビティを用いて業務フローを描画する「Step4：業務フロー描画ステップ」である．

4.2 業務機能階層

本稿では，次の 4 つの階層を持つ業務機能階層を用いて抽象化を行う．組織における業務では，所属するメンバにそれぞれの役割（ロール）が与えられ，そのロールを果たすためにシステムを利用する．本稿では，システムのユーザにとって分かりやすい仕様を復元するために，ユーザになじみの深い「組織–ロール–アクティビティ–アクション」の階層に沿った抽象化を行う．これにより，粒度の細かい「アクション」から，粒度の粗い「組織」を段階的に抽象化することができ，全体像を容易に把握できるようになる．

(1) 組織：ユーザの組織．必要に応じて組織がさらに階層化されることもある．
(2) ロール：組織においてのユーザの役割．ユーザ権限に対応することが多い．
(3) アクティビティ：ロールにおけるユーザの複数のアクションを抽象化したもの．
(4) アクション：ログに現れるユーザの操作．

4.3 諸定義

まず，システムログ E をログエントリ e の集合として定義する．ログエントリ $e_i \in E$ は対象システムでのユーザ操作の記録に対応し次のように定義する：

$$e_i = (e.u \in U,\ e.t \in T,\ e.a \in A,\ e.w \in W) \tag{1}$$

ここで，$U = \{u_1, u_2, u_3, \cdots\}$ はユーザの集合，T はタイムスタンプの集合（システムにおいていつユーザがこの操作を実行したかを示す），$A = \{a_1, a_2, a_3, \cdots\}$ はアクションの集合（ユーザが行った操作の種類），$W = \{w_1, w_2, w_3, \cdots\}$ は案件の集合（操作の対象）を示す．

各ログエントリ e は，ユーザ $e.u$ が案件 $e.w$ に対して，対象システム上で時刻 $e.t$ にアクション $e.a$ を実行したことを表現する．ログエントリを案件単位で時系列順に集めることによって，その案件に対する全ユーザのアクションの履歴を得ることができる．この履歴をプロセスインスタンス（p_i）と呼ぶ：

$$p_i = \langle e_1, e_2, \cdots, e_n \rangle.\ (i < j \Rightarrow e_i.t < e_j.t,\ e.w = w_i) \tag{2}$$

業務機能階層の 1 つであり復元した業務フローの主要な要素となるアクティビティ $act \in ACT$ はアクションの集合として定義する．本研究では 1 つのアクションが複数のアクティビティに属することを許す（$ACT \subset 2^A$）．

ロール $r \in ROLE$ もアクションの集合として定義する（$ROLE \subset 2^A$）．ロールはアクティビティの上位概念であり，復元した業務フローのスイムレーンを構成する．アクティビティと異なり，1 つのアクションは必ず 1 つのロールに所属する．

組織 $org \in ORG$ は所属するユーザの集合として定義する（$ORG \subset 2^U$）．

4.4 Step1：ロールモデル復元ステップ

図 2 で示したロールモデル復元ステップでは，以下を実行することで，ロールとアクションの関係を復元する．

(1) システムログからプロセスインスタンスを抽出し，さらにユーザが切り替わる箇所で分割することで，ブロック（同一ユーザによる連続アクション）を特定する．
(2) 2 つのアクションの同一ブロックでの共起関係を用いて有向関係 R を求め，R の連結成分となるアクションの集合をロールとする．
(3) 組織情報を用いて，ロールの名称や上位の構造化を行う．

以下では各ステップの実行を詳しく説明する．

4.4.1 Step1-1：ブロックの特定

業務システムは，複数のロールのアクティビティによって 1 つの案件を処理していく．この一連の操作の列がプロセスインスタンスとなる．ロールモデル復元の最初のステップとして，プロセスインスタンスからアクティビティの候補となるブロックを特定する．

ブロック $b_{i,j} \in B$ はプロセスインスタンス p_i 上で連続する同一ユーザのログエントリ列と定義する：

$$b_{i,j} = \{e_k \in p_i | j \le k \le n,\ e_k.u = e_j.u,\ e_{n+1}.u \neq e_j.u\} \tag{3}$$

図 3 に図 1 のシステムログから抽出したプロセスインスタンスとブロックの例を示す．図において，縦軸はプロセスインスタンスを示し，横軸はアクションを示す．破線でくくられている箇所がブロックである．ブロックは，1 つの案件に対して 1 人のユーザが連続して行った一連のアクションを表している．大規模なビジネスアプリケーションでは，案件 c_{00100} のユーザ「佐藤」と c_{00104} のユーザ「小泉」のように，同じロールを持つ複数のユーザが同じアクションを行うことがある．このため本研究ではシステムログを 1 度プロセスインスタンスに分解した後に，さらにユーザ単位で分割したブロックを分析の基本単位として採用している．ここでいうブロックは，ビジネスプロセス・フロー分析 [19] においてフローを階層的に表現する際の最小単位であるリーフプロセスの候補と考えることもできる．

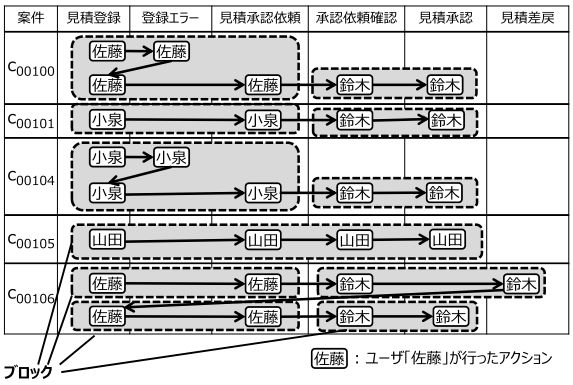

図 3 プロセスインスタンス，ブロックの例
Fig. 3 Example of process instances, block, role switching points.

同じロールを持つユーザは同じようなアクションを行う傾向があるため，ブロック単位で類似するアクションを集めることでロールとアクションの関係を推定できる．しかし，ブロックからロールを復元する際の問題として，案件 c_{00101}，c_{00104} のユーザ「小泉」のように必ずしも毎回同一のアクション群を行うわけではない．また，案件 c_{00105} のユーザ「山田」のように，複数のロールを担当するユーザがほかの案件でユーザが切り替わる部分を超えてアクションを行うこともある．この問題に対処するために次項で示す手法が必要となる．

4.4.2 Step1-2：ロールの復元

様々なアクションを含むブロックからロールと関係するアクション集合を特定するために，ブロックの中で同時に登場するアクション間の有向関係 R を定義する．

本研究ではこの関係に lift を用いる．lift はルールマイニングの分野でルールを構成する要素間の関係の尺度として用いられている [22], [23]．提案手法では，アクション a_i からアクション a_j への関係の重み $weight$ に lift 値を用い，次のように定義する．$weight(a_i \Rightarrow a_j) = lift(a_i \Rightarrow a_j)$ ここで，$aBlocks(a_i)$ はアクション a_i を含むブロックの集合を示す関数とする：

$$lift(a_i \Rightarrow a_j) = \frac{conf(a_i \Rightarrow a_j) \cdot |B|}{|aBlocks(a_i)|} \quad (4)$$

$$conf(a_i \Rightarrow a_j) = \frac{|aBlocks(a_i) \cap aBlocks(a_j)|}{|aBlocks(a_i)|} \quad (5)$$

$$aBlocks(a_i) = \{b \in B | \exists e \in b,\ e.a = a_i\} \quad (6)$$

関係 $a_i R a_j$ を，$weight(a_i \Rightarrow a_j) \geq MIN_{weight}$ となる 2 つのアクション間の関係とし，関係 R による連結成分をロール $r \in ROLE$ とする．つまり，r 内のアクション a_i は以下の性質を満たす：

$$\forall a_i \exists a_j \in ROLE.(a_i R a_j \vee a_j R a_i) \quad (7)$$

提案手法のロール復元手法は，同じユーザによって同時に行われるアクションは同じロールのアクションである傾向があることに着目している．一般にログに含まれるアクションの出現頻度には大きなばらつきがある．これに対応するため，lift 値を用いることでクラスタリングの際に頻度の大きいアクションの過度な影響を除去することができる．結果として，アクションのバリエーションを含むクラスタを構成することが可能になる．また，複数ロールを持つユーザの影響を低減させる効果もある．

組織	ロール	抽象化アクティビティ	アクション
	ロール1		見積登録 登録エラー 見積承認依頼
	ロール2		承認依頼確認 見積承認 見積差戻
	ロール3		見積受付 在庫確保依頼
	ロール4		在庫確保受付 在庫割り当て 割り当て失敗

図 4 ロールの推定
Fig. 4 Role assumption.

例えば，図 3 のプロセスインスタンス c_{00105} は 1 つのブロックに 4 つのアクションを含んでいる．しかし，他の案件では異なるユーザが行っているため，アクション「見積承認依頼」から「承認依頼確認」への lift 値が低くなる．一方，「見積登録」から「見積承認依頼」への lift 値は高くなる．したがって，**図 4** に示すロール推定結果では「見積承認依頼」と「承認依頼確認」は異なるロールに属し，「見積登録」と「見積承認依頼」は同じロールに属することになる．

4.4.3 Step1-3：組織情報の反映

大規模システムの業務フローの全体像を理解するためには，より高いレベルの抽象化を実現する必要がある．そのために，ユーザの情報だけでなく，組織情報（担当，課，部，事業部，アクセス権限など）を利用する．提案手法では，そのロール r に含まれるアクションを実行しているユーザが所属する組織に着目し，最も多くのユーザが所属する担当や組織 o_{max} をそのロールの組織に割り当てる：

$$g = \{e.u | a_i \in r, e.a = a_i\}, \quad (8)$$
$$\forall o_i \in ORG, |g \cap o_{max}| \geq |g \cap o_i|$$

まず，上記アルゴリズムで，アクセス権限の情報を使ってロール名を特定した後に，同じアルゴリズムを用いて，階層的に上位の組織名を特定していく．

図 4，**図 5** にこのステップを適用した例を示す．図 4 におけるロール 1〜4 は組織情報を用いて，アクションを実行しているユーザの多くが属するロール名が割り当てられる．つまり，図 5 の 4 列目のアクションを実行しているユーザ「佐藤」，「小泉」が承認依頼者，「鈴木」，「山田」が承認者のアクセス権限を持っていることから図 5 の 2 列目のようにロール名を割り当てられる．

組織	ロール	抽象化アクティビティ	アクション
営業部	承認依頼者		見積登録 登録エラー 見積承認依頼
	承認者		承認依頼確認 見積承認 見積差戻
在庫管理部	承認依頼者		見積受付 在庫確保依頼
	承認者		在庫確保受付 在庫割り当て 割り当て失敗

図 5 ロールモデル
Fig. 5 Role model.

4.5 Step2：代表アクション抽出ステップ

ロールモデルから抽象度の高い業務フローを復元するために，ロールを構成するアクションの中から代表的なアクションを見つけ出し，それをアクティビティとして業務フローの要素にする．

ビジネスアプリケーションのユーザは，「ログイン」「メニュー選択」のような一般的なアクションから操作を開始し，様々な業務的なアクションを行った後，「注文確認」や「承認依頼」などのロールに特徴的なアクションで終了する傾向がある．これは，一連のアクションを行った後に次のユーザへ引き渡す（もしくは引き渡さない）ことを最終判断するアクションであり，一連のアクションを代表すべきアクションであると考える．システム内に保存されている情報に対して，ユーザは一連のアクションにより様々な変更を行う．アクティビティを複数人が更新・参照する情報に対するトランザクションとしてとらえた場合，「コミット」「ロールバック」のトリガになるアクションが代表と見なされることとなる．この性質に着目し，ブロックの中で最後に行った割合を示す関数 $fRate(a_i)$ を次のように定義する：

$$fRate(a_i) = \frac{|fBlocks(a_i)|}{|aBlocks(a_i)|} \tag{9}$$

ここで $fBlocks(a)$ はアクション a_i が最後に登場するブロックの集合を返す関数であり，以下のように定義する：

$$\begin{aligned} &fBlocks(a) = \{b \in aBlocks(a) | \\ &\exists e_0 \in b, e_0.a = a, \forall e_k \in b\ e_0.t \geq e_k.t\} \end{aligned} \tag{10}$$

上記定義を用いて，Step2：代表アクションの抽出ステップでは，ロール r それぞれに対して，閾値 MIN_{final} 以上の割合で，ロール内の最後のアクションとなるアクション a_i を求め代表アクションとする．したがって，代表アクションはロールごとに複数抽出されうる：

$$mainAct(r) = \{a_i | fRate(a_i) \geq MIN_{final}, a_i \in r\} \tag{11}$$

4.6 Step3：抽象化アクティビティ復元

前節で抽出した代表アクションを用いて，抽象化アクティビティを復元する．抽象化アクティビティはアクションの集合として定義する．それぞれの代表アクション a_0 を含むブロックの中に含まれるアクションのうち同一ロールに含まれるアクションを抽象化アクティビティ act_{a_0} に加える．抽象化アクティビティ act_{a_0} は，その元となった代表アクティビティ a_0 で識別する：

$$act_{a_0} = \{e.a | b \in aBlocks(a_0), e \in b\} \tag{12}$$

この方法は 1 つのアクションが複数の抽象化アクティビティに属することを許すソフトクラスタリングである．**図 6** に復元した抽象化アクティビティの例を示す．抽象化アクティビティを構成するアクション集合は抽象化アクティビティを理解するうえで重要である．このため，抽象化アクティビティを構成するアクションで実際の実行履歴の断片であるブロックを再現することで，理解しやすくなる．一般に，複数のアクティビティを構成するブロックに共通して登場するアクションが存在する．そこで提案手法では，ハードクラスタリングではなくソフトクラスタリングを採用した．たとえば，営業部の承認者ロールにおいて，「承認依頼確認」は 2 つの抽象化アクティビティのメンバとなっている．

組織	ロール	抽象化アクティビティ	アクション
営業部	承認依頼者	見積承認依頼	見積登録 登録エラー 見積承認依頼
	承認者	見積承認	承認依頼確認 見積承認
		見積差戻	承認依頼確認 見積差戻
在庫管理部	承認依頼者	在庫確保依頼	見積受付 在庫確保依頼
	承認者	在庫割り当て	在庫確保受付 在庫割り当て
		割り当て失敗	在庫確保受付 割り当て失敗

図 6 復元した業務機能階層
Fig. 6 Recovered abstruct activities.

4.7 Step4：フロー生成

最後のステップでは，復元した情報から業務フロー図を描画する．業務フロー図の縦軸に組織情報とロールを配置し，スイムレーンにロールに属するアクションを配置する．次にそれぞれのプロセスインスタンスにおけるアクションの出現順にアクションを矢印の遷移で結んでプロットする．同じ遷移が登場した場合には重ねて描画する．**図 7** は，図 3 のプロセスインスタンスから復元したアクションの単位で業務フローを描画した例である．抽象度を上げる場合には，ロールの代表アクションのみを配置し，プロセスインスタンスの代表アクションだけに着目して，上記の描画を行う．**図 8** はアクティビティレベルの業務フローである．案件 c_{00100} でのロール「承認依頼者」のユーザ「佐藤」が実施した代表アクション「見積承認依頼」とロール「承認者」のユーザ「鈴木」が実施した代表アクション「見

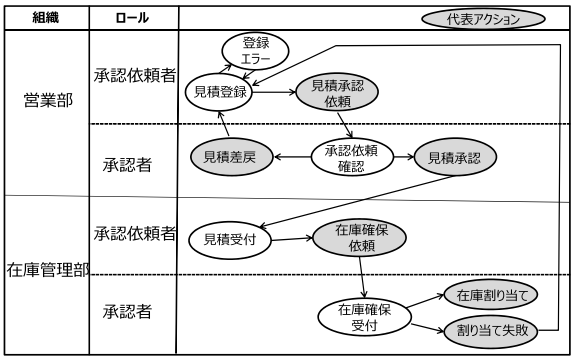

図 7 業務フロー（アクションレベル）
Fig. 7 Business process diagram (action level).

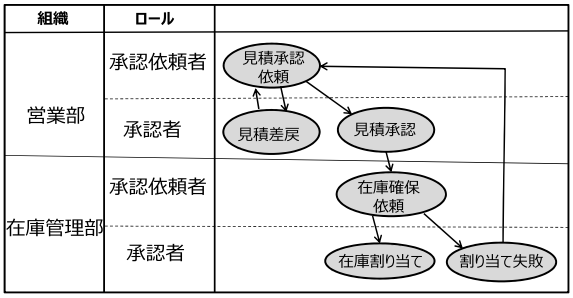

図 8 業務フロー（アクティビティレベル）
Fig. 8 Business process diagram (activity level).

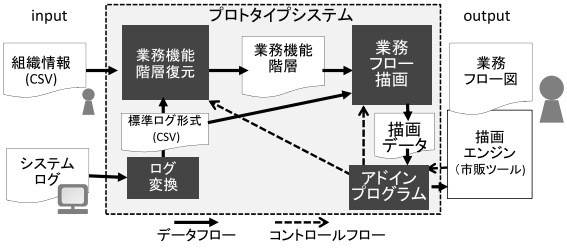

図 9 プロトタイプの概要
Fig. 9 Outline of our prototype system.

積承認」とが順番に遷移で結ばれている．

5. プロトタイプの実装

図 9 に開発したプロトタイプツールの概要を示す．ツールは 4 つのモジュール（業務機能階層復元，業務フロー描画，ログ変換，アドインプログラム）からなっている．業務フローの出力は，仕様書として利用することを考え，市販のオフィスツールの描画エンジンを利用した．それぞれのモジュールは次の特徴を持つ．業務機能階層復元は前章で示したロールモデル復元や抽象化アクティビティ復元を行う．業務フロー描画は前章で示した業務フロー復元のための描画データを作成する．ログ変換はシステム独自のフォーマットのログを図 1 で示した「標準ログ形式」に変換する．アドインプログラムは，市販のオフィスツールを GUI として用いるための，メニュー拡張やモジュールの起動，データ連携を行う．

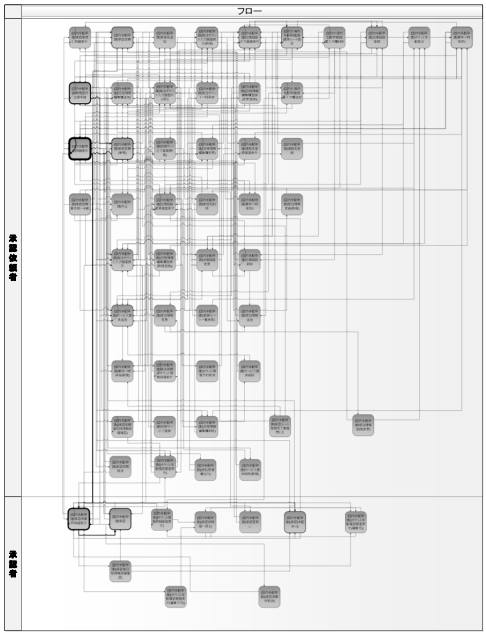

図 10 業務フロー（抽象化前）
Fig. 10 Business process (Before abstraction).

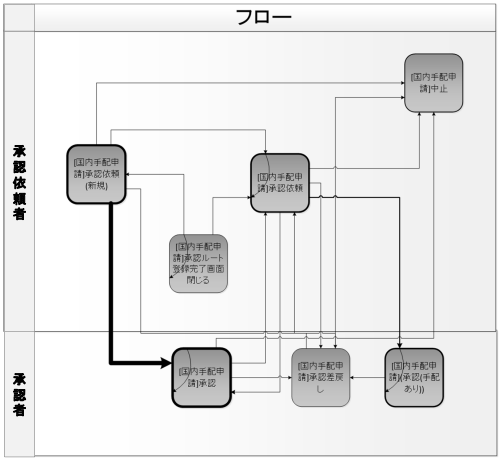

図 11 業務フロー（抽象化後）
Fig. 11 Business process (After abstraction).

このツールを利用する際には，あらかじめログをログ変換モジュールを使って，「標準ログ形式」に変換しておく．そして，オフィスツールのメニューから「標準ログ形式」のログファイルを選択して実行させることで，業務フローが描画される．次章で示すケーススタディでの業務フロー図（**図 10**，**図 11**）は本ツールのスクリーンショットである．

6. ケーススタディ

提案手法のケーススタディとして実際の業務アプリケーションに適用した．システムの実際のシステムログと，組織情報としてユーザ権限からユーザの役割の情報を入手し，前節のプロトタイプツールを利用して業務フローの復元を行った．

6.1 対象システム

対象システムは，出張申請・精算を行うシステムである．このシステムは，従業員が出張前の出張を申請し上長が承認，出張後の出張旅費を精算申請し承認する業務などを扱う．ユーザはこのシステムを Web ブラウザから利用する．ユーザのロールは承認依頼者（出張と旅費の精算申請を行う），承認者（申請の承認を行う）の 2 つである．従業員である承認依頼者が申請した内容を上長である承認者が承認するというロールを持つ．このシステムは 5 つサブシステムを持ち，それぞれ 5 つの業務（BP.A，BP.B，...，BP.E）に対応する．これらの業務は，出張の申請，出張旅費の計算と精算，住所や勤務地などの基本情報の登録，という企業の事務処理で行う典型的な業務を扱うシステムである．

これは，田中の分類 [18] でいうとバックオフィス系，鷹田らの分類 [17] でいうと基幹系と呼ばれるカテゴリのシステムである．提案の方式は，この分類に属するシステムだけでなく，チャネル系と呼ばれる顧客とのやりとりを行うシステムでも有効と考えている．しかし，チャネル系システムは顧客の個人情報を扱うことが多く，評価に使うことが難しいことから，評価の対象をバックオフィス系システムとした．

このシステムのシステムログと組織情報としてユーザ権限（ユーザとロールの関係の表）を用いてケーススタディを実施した．システムログは 3 カ月分のデータで合計 107 M バイトで，410,054 のログエントリ，3,969 のユーザ，247 のアクション，44,604 の案件を含んでいた．パラメータとして，$MIN_{weight} = 30$ を用い，$MIN_{final} = 0.3, 0.5, 0.75, 0.9$ で比較した．

6.2 評価方法

提案方式の評価は次の観点で行った．

(1) Step1 の評価：ロールモデル復元ステップで，アクションのロール分割が正しく行われているか．
(2) Step2 の評価：代表アクション抽出ステップで，代表アクションの抽出が正しく行われているか．
(3) Step3 の評価：抽象化アクティビティ復元ステップで，どの程度情報が圧縮され，必要な情報がどの程度復元されているか．
(4) Step4 の評価：フロー生成ステップで，復元した業務フローは開発者にとって違和感がないか．

6.3 評価結果

6.3.1 Step1 の評価：アクションのロール分割が正しく行われているか

表 1 に Step1 でロールに分割したアクションを対象システムの開発者が作成した操作マニュアルに記載されているロールと比較した結果を適合率と再現率で示す．それぞれの BP で承認依頼者（以後依頼者と表記），承認者の 2 つのロールを復元できた．BP.B のロール：承認者の 1 つのアクションを除き，操作マニュアルと同じ分割が行われた．正しく分割されていなかったアクションは，承認依頼の後に依頼した内容の詳細を確認するアクションであった．このアクションは他のロールと混在して使われるケースがあったため正しく分割できなかった．

表 1 ロールモデル復元結果
Table 1 Result of recovered role model.

BP	role	アクション	正解値	正解数	適合率	再現率
A	依頼者	47	47	47	1.00	1.00
A	承認者	10	10	10	1.00	1.00
B	依頼者	35	35	35	1.00	1.00
B	承認者	11	12	11	1.00	.917
C	依頼者	33	33	33	1.00	1.00
C	承認者	7	7	7	1.00	1.00
D	依頼者	36	36	36	1.00	1.00
D	承認者	8	8	8	1.00	1.00
E	依頼者	21	21	21	1.00	1.00
E	承認者	4	4	4	1.00	1.00
	計	212	213	212	1.00	.995

表 2 代表アクション抽出結果の評価
Table 2 Evaluation of main actions selection.

	終端率				頻度			
	.3	.5	.75	.9	.3	.5	.75	.9
復元数	10	9	7	4	5	4	1	1
正解	7	7	7	4	3	2	1	1
適合率	.70	.78	1.0	1.0	.60	.50	1.0	1.0
再現率	1.0	1.0	1.0	.57	.43	.29	.14	.14

6.3.2 Step2 の評価：代表アクションの抽出が正しく行われているか

表 2 に BP の中で最もアクション数が多い BP.A に対して，Step2 で抽出した代表アクション抽出の精度評価の結果を示す．パラメータ $MIN_{final} = 0.3, 0.5, 0.75, 0.9$ をそれぞれを用いて，復元した結果を評価した（終端率）．評価は抽出した代表アクションに対して，このシステムの開発者にロールを代表する作業として適切かどうかを評価してもらった．比較のため，ロールに対応する全ブロックの中で，アクションが登場する割合（頻度）の閾値によって抽出したアクションをあわせて評価した．

結果，終端率（fRate）の閾値 $MIN_{final} = 0.75$ で生成したものが適合率と再現率がともに 1.00 であり最も成績が良く，頻度の閾値で抽出したものと比較しても高い精度を得られた．抽出した 7 つの抽象化アクティビティに対して過不足がないことを対象システムの開発者が確認している．終端率が 0.75 よりも小さい場合には，本来他のアクティビティの中で途中段階でオプション的に登場するアクション「～書出力」「～削除」などが抽出されてしまい適合率が低下した．0.9 に設定した場合は，重要な作業まで取り除か

表 3 業務フロー復元圧縮率
Table 3 Result of recovered business process.

bpid	抽象化前	抽象化後	圧縮率
A	58	7	.899
B	84	6	.929
C	43	3	.931
D	44	4	.910
E	25	3	.880
計	59	6	.899

表 4 プロセスインスタンスカバー率
Table 4 Result of coverage of recovered business process.

	元データ		抽象化		再現率	
bp role	all	2+r	all	2+r	all	2+r
A 依頼	9,836	811	909	755	.092	.930
A 承認	906	801	867	786	.956	.981
B 依頼	2,507	535	509	403	.203	.753
B 承認	576	486	529	466	.918	.958
C 依頼	20,152	19,630	19,992	19,613	.992	.999
C 承認	28,900	19,707	28,897	19,707	.999	1.00
D 依頼	471	411	302	300	.641	.729
D 承認	540	421	540	421	1.00	1.00
E 依頼	1,620	1458	1,560	1,450	.962	.994
E 承認	2,448	1,458	2,447	1,457	.999	.999
のべ	67,956	45,718	56,552	45,358	.832	.992
合計	44,604	22,427	33,555	22,416	.752	.999
$\|E\|$	410,054	348,204	66,055	54,098		
$\|A\|$	247	245	22	22		

all：全プロセスインスタンス
$2+r$：2 以上のロールを持つプロセスインスタンス

れてしまい再現率が下がることも確認できた．また，頻度を閾値とした場合，多くの業務で共通して用いられる「一覧表示」など重要ではない作業が代表アクションとして抽出されてしまい，正しい結果にならないものがあった．

6.3.3 Step3 の評価：どの程度情報が圧縮され，必要な情報がどの程度復元されているか

表 3 に復元した抽象アクティビティによって，どの程度の情報が圧縮されたかを示す．それぞれのビジネスプロセス（BP）ごとに，抽象化前のアクションの数 A と抽象化後の抽象化アクティビティの数 B から，$(A-B)/A$ で圧縮率を算出している．平均して約 9 割の情報が圧縮されていることが分かる．

表 4 に復元した抽象化アクティビティが，元のプロセスインスタンスをどの程度カバーしているかを示す．抽象化アクティビティの代表アクションを含むプロセスインスタンスの全体に対する割合を再現率とした．表中の「依頼」はロール「承認依頼者」，「承認」はロール「承認者」を表している．取得したログには，取得期間の切れ目でプロセスインスタンスの前半，後半のログが取得できていないものや，ユーザが操作の途中で中断して別の作業を始めたものが含まれていた．これらは，プロセスとして開始から終了までがそろっていないため，プロセスインスタンスとして不完全なものになっている．このようなノイズを除去するために，1 つのロールしか含まれないプロセスインスタンスは除外して評価した．各評価の「$2+r$」列は，ノイズ除去を適用した場合の結果である．ノイズ除去後の再現率はのべ約 45,000 のプロセスインスタンスに対して 0.992 との高い精度で復元できていることが分かる．

再現率が 0.8 未満の BP.B，BP.D について調査を行った．これらは海外出張関係の業務であり，それぞれのトランザクションの期間が長いために案件のスタートがログ取得期間以前から始まっているものが含まれていた．このため，本来あるべき承認依頼プロセスがログに残っておらず，その後の承認プロセス以降がログに存在した．そして，一連の業務が終わった後に次の同様な業務を行う際の参考にするために，承認依頼者が過去の結果を参照する「明細表示」のアクションを行っていた．これが抽象化アクティビティとして認識されていないことが原因であることが分かった．

表 5 抽象化アクティビティの評価
Table 5 Evaluation of recovered abstract activities.

BPid	正解の遷移	不正解の遷移	適合率	再現率
A	15	0	1.00	1.00
B	15	2	.88	1.00
C	6	3	.67	1.00
D	7	2	.78	1.00
E	6	2	.75	1.00
計	49	9	.84	1.00

6.3.4 Step4 の評価：復元した業務フローは開発者にとって違和感がないか

復元した業務フローが正確にシステムの仕様を反映しているかどうかを確認するために，対象システムの開発者に内容を評価してもらった．図 10 と図 11 に，本方式による復元の例としてビジネスプロセス BP.A を復元した業務フローを示す．アクションレベルでは 57 個のアクションが複雑に関係しあうものであったが，アクティビティレベルでは 7 個の抽象化アクティビティにより表現できている．この業務フロー図を開発者が確認したところ，復元した抽象化アクティビティには過不足はないという評価を得た．さらに，業務フロー図上で復元した抽象化アクティビティ間の遷移に関して，仕様として正しいかどうかを評価してもらった．対象システムを最も理解している開発者に，復元した業務フロー図を提示し，生成した抽象化アクティビティのリストと遷移リストに○×をつけるワークシートを記入してもらった．**表 5** に開発者による評価結果を示す．評価の結果，84%の遷移が正しいという結果となった．不正解となった遷移は，「却下」した後「引き戻し」て「承認」するなど，終端率でフィルタリングした処理（この場合は「引き戻し」）を挟んだときに本来あるべき中間の処理

が取り除かれたために生じていた．

6.4 考察

6.4.1 業務フロー復元の精度について

ケーススタディによる評価により，以下のことが分かった．Step1 のロール分割の精度が 99.5%とほぼ復元できていると考える．

Step2 の代表アクション抽出については終端率 0.75 としたときが最も精度が高く，単なるブロックへの出現率と比較してもこの方式による代表アクション抽出は有効である．

Step3 の評価によって，元のログの情報からアクション数にして 89.9%の情報を圧縮でき，その圧縮した情報には全体の 99.2%のプロセスインスタンスの情報が含まれていることが分かり，情報の圧縮が効率的に行われているといえる．

Step4 の評価で，復元した業務フローが開発者の視点で見て，抽象化アクティビティのレベルでは問題なく，アクティビティ間の遷移で見ても 84%の適合率で復元できていることが分かり，全体として提案手法による業務フロー復元は有効であることがいえる．

3.4 節で示した要件に対しての考察を以下で述べる．

要件 (1) 表 3 でそれぞれの BP の 25〜84 あるアクションを 3〜7 に抽象化することができた．これによって，抽象化した業務フローを用いて，マクロ的に全体像を俯瞰することが可能になる．

要件 (2) 抽象化アクティビティとアクションの関係を業務機能階層で抽出することができた．これにより，抽象化アクティビティと元のアクションの対応をとることが可能になる．さらに，抽象化した業務フローで全体を俯瞰した後に，詳細レベルのフローを業務機能階層を手がかりに理解することで，段階的に詳細レベルまでの理解が可能になる．

要件 (3) ソフトクラスタリングを用いたことにより，それぞれの抽象化アクティビティに対応する意味のあるアクション群を再生でき，表 4 により全体の 99.2%のプロセスインスタンスの情報が含まれていた．これにより，抽象化アクティビティに関係するアクション群は，実際にプロセスインスタンスの断片として業務の中で実行されたものであることがいえる．

要件 (4) 開発者の評価により，抽象化アクティビティレベルでは適合率，再現率ともに 1.0，遷移レベルでは，適合率 0.84，再現率 1.0 となった．再現率を 1.0 に近づけるために，ソフトクラスタリングを採用したため，一定数の不要な遷移が描画されている．しかし，当初の目的である「大規模なシステムの仕様をマクロ的な視点から俯瞰して理解する」という意味では，遷移の情報が不足しているよりも，余分な情報を詳細レベルのフローで確認して除去するという方法で補完することができる．このため，要件は満たせたといえる．ハードクラスタリングを用いることで適合率は改善可能である．しかし，これにより，要件 (3) を満たせない影響は大きいので，本研究の目的には合致しない．

6.4.2 残された課題

前項の考察により，提案手法が有効であることを示したが，さらなる効率化のために以下の課題が残されている．

再現率優先の要件 (4) を満たすために，ソフトクラスタリングを採用したことにより遷移の適合率が 0.84 となり，16%の誤った遷移が復元されている．このため，ユーザ・開発者が全体像を誤った形で理解してしまう恐れがある．この課題を解決するために，遷移の頻度やロールの切れ目になるブロックにまたがる遷移をブロックの長さによってフィルタリングをすることが考えられる．これにより，ハードクラスタリングによって満たせなくなる要件 (3) を満たしたまま，この適合率を向上させることができる可能性がある．

今回 0.75 とした終端率の閾値 MIN_{final} の設定は，今回複数パターンで比較して良い結果を選んだが，実用上は何らかの方法で決める方法を検討する必要がある．これは，ログの特性を前もってスクリーニングすることによって，適切な閾値を算出できる可能性がある．

プロセスインスタンスのカバー率を低下させる原因となっている，ログ取得期間と案件のライフサイクルがずれるという問題に対して考慮する必要がある．これは，事前に途中開始や途中終了の案件をフィルタリングして分析するなどの前処理を行うことによって，プロセスインスタンスとしてふさわしくないデータをノイズとして除去し，より高いカバー率を得ることができる可能性がある．

「明細表示」のような複数のプロセスインスタンスにまたがる業務（過去のケースを参照して，別のケースを承認するなど）を扱うことができるようにプロセスインスタンスの考え方の拡張を行うことで，再現率を向上させることができる可能性がある．

7. おわりに

正確な仕様が残されていない既存の業務システムを理解するために，システムログと組織情報から業務フローを復元する方式を提案した．本方式は，複数のユーザによって行われる業務に対し，ユーザの切れ目に着目することによって，ロールや代表的なアクションを抽出し，抽象化されたアクティビティを復元することができる．

ケーススタディによる評価では，ロール復元精度が 99.5%，抽象化アクティビティの再現率が 99.2%であり，本手法の有効性を示すことができた．

参考文献

[1] 日本情報システムユーザ協会：企業 IT 動向調査 2015，日

本情報システムユーザ協会 (2016).
[2] Murphy, G.C., Notkin, D. and Sullivan, K.: Software reflection models: Bridging the gap between source and high-level models, *Proc. FSE'95*, pp.18–28 (1995).
[3] Rountev, A. and Connell, B.H.: Object Naming Analysis for Reverse-Engineered Sequence, *Proc. ICSE'05*, pp.254–263 (2005).
[4] Rabelo, L.A.P., do Prado, A.F., de Souza, W.L. and Pires, L.F.: An Approach to Business Process Recovery from Source Code, *Proc. ITNG'15*, pp.361–366 (2015).
[5] Ghose, A., Koliadis, G. and Chueng., A.: Process discovery from model and text artifacts, *Proc. SOPOSE'07*, pp.167–174 (2007).
[6] Paradauskas, B. and Laurikaitis., A.: Business knowledge extraction from legacy Information systems, *J. Info. Tech. and Control*, Vol.35, No.3, pp.214–221 (2006).
[7] Zou, Y., Lau, T.C., Knotogiannis, K., Tong, T. and MeKegney, R.: Model driven business process recovery, *Proc. WCRE'04*, pp.224–233 (2004).
[8] van der Aalst, W.: *Process Mining: Descovery, Conformance and Enhancement of Business Process*, Springer (2011).
[9] Francescomarino, C.D., Marchetto, A. and Tonella, P.: Cluster-based modularization of processes recovered from Web, *J. Softw.: Evol. and Proc.*, Vol.25, No.2, pp.113–138 (2010).
[10] Pérez-Castillo, R.: MARBLE: Modernization Approach for Recovering Business Processes from Legacy Information Systems, *Proc. ICSM'13*, pp.671–676 (2013).
[11] Leemans, M. and van der Aalst, W.: Process mining in software systems: Discovering real-life business transactions and process models from distributed systems, *Proc. MODELS'16*, pp. 44–53 (2016).
[12] van der Aalst, W.: *Process Mining—Data Science in Action*, 2nd edition, Springer (2016).
[13] Weerdt, J.D., vanden Broucke, S., Vanthienen, J. and Baesens, B.: Active Trace Clustering for Improved Process Discovery, *IEEE TKDE*, Vol.25, No.12, pp.2708–2720 (2013).
[14] Francescomarino, C.D., Marchetto, A. and Tonella, P.: Reverse Engineering of Business Processes, *Proc. CSMR'09*, pp.139–148 (2009).
[15] Mibe, R., Tanaka, T., Kobayashi, T. and Kobayashi, S.: Business Process Recovery based on System Log and Information of Organizational Structure, *Proc. SANER'17*, pp.531–535 (2017).
[16] 田中匡史，三部良太：業務フロー仕様生成技術の開発，信学技報 115(249)，電子情報通信学会 (2015).
[17] 薦田憲久，水野浩孝，赤津雅晴：ビジネス情報システム，コロナ社 (2005).
[18] 田中　淳：金融機関向けソリューション・システムの全体像と今後の取組み，UNISYS TECHNOLOGY REVIEW No.77, pp.3–11 (2003).
[19] 牧野友紀：ビジネス環境と実装システムを繋ぐ BPM と SOA，情報処理，Vol.46, No.1, pp.60–63 (2003).
[20] van der Aalst, W. et al.: Process Mining Manifest, *Proc. BPM 2011*, pp.169–194 (2012).
[21] Cowan, N.: *Working memory capacity*, Psychology Press (2005).
[22] Agrawal, R., Imielinski, T. and Swami, A.: Mining association rules between sets of items in large databases, *Proc. ACM-SIGMOD'93*, pp.207–216 (1993).
[23] Ling, C.X. and Li, C.: Data Mining for Direct Marketing: Problems and Solutions, *Proc. ACM SIGKDD'98*, pp.73–79 (1998).

三部 良太 （正会員）

1967 年生．1990 年電気通信大学計算機科学科卒業．1992 年東京工業大学大学院総合理工学研究科修士課程修了．同年（株）日立製作所入社．2016 年より東京工業大学大学院情報理工学研究科博士後期課程に社会人博士として再入学．システム生産技術に関する研究に従事．電気学会会員．

田中 匡史

1966 年生．1990 年東京大学基礎科学科第二卒業．1992 年東京大学大学院総合文化研究科修士課程修了．同年（株）日立製作所入社．システム生産技術に関する研究に従事．電子情報通信学会，ヒューマンインタフェース学会会員．

小林 伸悟

1989 年生．2011 年茨城大学理学部理学科物理学コース卒業．同年日本エクス・クロン株式会社入社．2013 年よりシステム生産技術に関する研究に従事．

小林 隆志 （正会員）

1974 年生．1997 年東京工業大学工学部情報工学科卒業．1999 年同大学大学院情報理工学研究科計算工学専攻修士課程修了．2004 年同専攻博士課程修了．博士（工学）．同大学助手，名古屋大学大学院情報科学研究科特任准教授，同研究科准教授を経て，2012 年より東京工業大学大学院情報理工学研究科准教授．現在，同大学情報理工学院准教授．ソフトウェア再利用技術，プログラム理解，リポジトリマイニング，Web サービス連携等の研究に従事．電子情報通信学会，日本ソフトウェア科学会，日本データベース学会，ACM，IEEE，IEEE-CS 各会員．