[解説]

プロセスマイニング・サーベイ (第 03 回: データ)

飯島 正 †, 田端 啓一 ‡, 斎藤　忍 ‡

1　はじめに

今回 (第 03 回) はプロセス発見の入力となるログデータに関連する話題について解説する．具体的には，ログに残すべき情報とその構造，標準化が進みつつあるデータ形式，プロセス発見の各種アルゴリズムの定量的評価に使用することのできるログ生成器，そして，ロギングのためのガイドラインについて取り上げる．

まず，第 2 節において，一般に情報システムがログを残す目的と，そこに含めるべき情報について解説する．プロセスマイニングは，その実施者の主体的な目的に応じて行われる．情報システムの解析目的に合わせて，プロセスマイニングのツールやアルゴリズムの選択も左右されるだけでなく，プロセスマイニングの入力データとして含めるべき情報も異なってくる．

続く第 3 節では，プロセスマイニングの文脈で使われるログデータの構造について解説し，ログ，トレース，イベントといった用語について確認する．ここでは，ログデータのためのメタデータに関して言及する．

さらに第 4 節では，プロセスマニングにおいて共通的に使われる主なデータ形式について解説する．データ形式には標準化が進められており，構造化されたデータ交換フォーマットとして，XML の利用が進んでいる．本稿では，手軽なデファクトスタンダードである CSV (Comma Separated Values) 形式に加えて，XML に基づく XES 形式と MXML 形式について解説する．XES は MXML の後継ともいえるフォーマットで国際標準化を目指しており，将来的には XES が有望である．しかし，本稿執筆時点では，ツールの一部に XES に未対応のものも残っており，データ形式間の相互変換ツールもあるとはいえ，MXML も根強く使われ続けている．ログは，(すべてのログがそうとは言い切れないが) 比較的長期に渡り，蓄積保管されるものが多いため，データ形式の安定した標準化と，拡張時の互換性の保持[1]は，今後のプロセスマイニング技術の普及と発展のために重要な課題である．

第 5 節では，評価用のログデータ生成ツールについて解説する．プロセス発見アルゴリズムの定量的な比較評価のためには，アルゴリズムへの入力とするログデータが必要である．それには，実データもさることながら，公平な評価のためには，何らかの性質を指定して生成したログデータが必要となる．前回のツール編においても，ツール試用のためのログデータの入手方法や生成方法に言及したが，今回は評価実験のための専用ログデータ生成ツールに関して解説する．

第 6 節では，ログデータの品質を確保するために，比較的最近，提唱されているロギングガイドラインを紹介する．このガイドラインは，一見すると，ごく当たり前のことを述べているだけのように見えてしまう傾向はある．しかし，組織内外で早い時点から合意を取り付けておかねばならない事項なので，こうしたガイドラインをまとめておき参照することは意義がある．

2　プロセスを意識した情報システム (PAIS) とログデータ

2.1　ロギングの目的

プロセスを意識した情報システム (PAIS; Process-Aware Information System) においては，そこで稼働されるビジネスプロセスが主要な要素である．

[1] 上位互換性ないし後方互換性．将来的にデータ形式が拡張されても，過去に蓄積された膨大なデータは，その保管時のまま使えること．もちろん，データ変換ができれば十分であるが，変換せずに使えるのであれば，それに越したことはない．

A Survey on Business Process Mining
— 03: Data —
Tadashi Iijima†, Keiichi Tabata‡, and Shinobu Saito‡
†Faculty of Science and Technology, Keio University
‡Nippon Telegraph and Telephone Corporation
† 慶應義塾大学・理工学部
‡ 日本電信電話株式会社
[解説] 2016 年 9 月 16 日受付.
ⓒ一般社団法人 情報システム学会

ビジネスプロセスに関する多くの情報が，情報システムの残すログ（運用記録）に残されている．

従来，ログを残す目的としては以下のようなものがあった．

(1) ログ監視による異常状態の検出/対処へのトリガー，原因究明の手がかり
(2) システム監査 (法令順守，組織内規定順守など）
(3) 性能改善 (ボトルネックの検出と解消，頻出経路への資源の優先割り当てなど)
(4) ディジタル･フォレンジックス (digital forensics) のための証拠

これらの目的から，当初は，人間（システム管理者）が読むためだったものから，計算機可読で自動処理 (監視や解析の自動化) が可能となるものへと変化してきている．

プロセスマイニングにおいてもログは基本的な入力データであり，当然ながらログの目的として，プロセスマイニングが含まれることとなる．プロセス発見（制御フロー発見）は，ログを入力データとして，プロセスモデルを生成し出力する．プロセス適合性検査では，モデルとログを入力として，モデルがログを満たすことを検査する．プロセス強化では，モデルとログを入力として，ログを満たすように強化されたモデルを出力する．

プロセス発見の目的はいくつかある．例えば，現実の運用結果であるログに基づいて，稼働中のプロセスを再認識することがある．情報システムの設計時の想定は，長く稼働している間に，成り立たなくなることがある．特に，最近は，複数のサブシステムやサービスを組み合わせて SoS(System of Systems) として構成され大規模化した情報システムも多い．その組み合わせは，必ずしも個々のサブシステムやサービスの設計時に想定されたものとは限らない．したがって，その情報システム全体を把握するにあたり，改めて，運用データであるログに基づいて再認識ことには価値はある．この目的では，稼働している情報システムが，これまで出力してきたログ情報だけではなく，マイニングを行う実行者がその目的に応じてログに残すべき必要な情報を選択し，その情報を抽出するプローブ (探査針) を後付けでアタッチしなければならない．

また，従来，手作業で行われていた作業を情報システム化へ移行する際に，如何にして，その作業プロセスをモデル化するかという問題も発生する．人間がきわめて柔軟に行っていた手作業から，情報システムで自動化することのできるプロセスモデルを抽出することは，必ずしもそう容易なことではない．Microsoft 社の表計算ソフトウェア Excel などでは，手作業での操作を記録し再利用できる自動記録マクロも，マイニングの対象となりうるログデータといえるかもしれない．

2.2 ログに含めるべき情報

ログの構造は次の第 2 節で述べるが，そこでイベント情報として含めることのできる付帯情報にバリエーションがある．

イベント (Event) とは，何らかの事象の発生に伴うシグナル，もしくは，その事象自体のことを指す．一般にイベントというと，それ次回は時間経過を伴わず一時点 (Timepoint) で発生し終了する原子的な (Atomic) ものであり，一つのタイムスタンプ (Time Stamp; 時刻印) を持つことができる．また，何らかのイベントが発生した場所，すなわちイベント発生源 (Event Source) も属性情報として持ちうる (特に区別する必要のない場合には意識しない)．そして，通常はイベントの種類を示すラベルも含まれる．さらにいえば，イベントには属性として 5W1H が付随するともいえる．すなわち,「いつ (When)」,「どこで (Where)」,「誰が (Who)」,「何を (What)」,「なぜ (Why)」,「どのように (How)」という情報が，各イベント毎に関与する．

ソフトウェア工学関連では，イベントという語は，いろいろな文脈で使われている，有限状態マシン (FSM, Finite State Machine) はイベントに応じて，状態遷移を引き起こす．グラフィカルユーザインタフェース (GUI, Graphical User Interface) では，利用者によるマウスクリックやキーボード打鍵などで発生するイベントに対して，事前に登録した何らかのイベントハンドラ（ないしコールバック関数) を暗黙的に起動 (Implicit invocate) するイベント駆動プログラミングが基本的である．組込みソフトウェア (Embedded Software) 分野では，例えばセンサーで検知されたデータの変化がイベントの発生を引き起こすことも多い．したがって，イベントに反応するようなリアクティブな組込みシステムでは，有限状態マシン (FSM) を使ったモデリングがしばしば行われる．そこでは，システムの外部 (利用者や外部環境) で発生するイベント (いわゆる外部イベ

ント) が扱われることが多いが，それと区別なくシステム内部で発生させた内部イベントも対象となりうる．

一方，プロセスマイニングでは，制御フロー，すなわち，タスクやアクティビティの実行の手順に関心を持つのが基本である[2]．この場合，タスクに関わるイベントだけが対象となる．したがって，タスクのライフサイクルに関するイベントのラベルとしては，一般的には，タスクの状態の変化に応じた,「スケジュール (schedule)」,「開始 (start)」,「取り下げ (withdraw)」,「中断 (suspend)」,「再開 (resume)」,「中止 (abort)」,「完了 (complete)」の 7 つのイベントタイプ (lifecycle:transition 属性) を想定することができる (図 1)[3]．

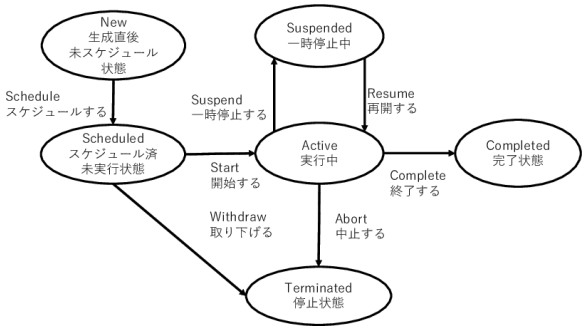

図 1: タスクのライフサイクルの状態遷移図 (文献 [2] を元に作成)

実際には，時間を伴って実行されるタスクを対象としているとはいえ，タスクの実行順序を制御する制御フローにのみ関心がある場合，実用上,「スケジュール」から「開始」，さらに「完了」までの三つのイベントを，一つの原子的イベントとみなすことがあり，そのために「タスク実行 (normal)」をイベントタイプに加えることもある．

「タスク実行 (normal)」タイプのイベントにタイムスタンプ情報を持たせるとすると,「スケジュール (schedule)」ないし「開始 (start)」のタイムスタンプで代表させることが多いと思われる[4]．以下の内容でも，タスクの実行順序にのみ関心がある場合,「タスク実行 (normal)」というイベントタイプのみを使うこともあれば (その場合，1 種類のイベントタイプしか使わないので，イベントタイプを明記する必要すらない)，タスクの「開始 (start)」と「完了 (complete)」のみを使うこともある．

タスクの実行順序にだけ関心がある場合，イベントタイプは「タスク実行 (normal)」だけで表現されることがあると述べたが，この場合には，イベント情報のうち (5W1H でいうところの)「何を (What)」は，タスク ID で十分である．しがって，単純にタスク ID の列 (シーケンス) で一つのケース (事案) の実行トレースを表現することも多い．テキスト形式の例題用ログデータでは，1 行に 1 つのトレースを表現する際に，単純にタスク ID を 1 個以上の空白文字で区切って並べることがある．また，行の先頭に (トレースに先立って)，ケース ID や頻度 (そのトレースが得られた回数) を記述することも多い．

「何を (What)」以外のイベント情報も与えられることがある．イベントの「いつ (When)」情報は，タイムスタンプとして与えることができる．システムの性能解析と改善のためにはタイムスタンプ情報は必要であろう．イベントの「どこで (Where)」情報は，必要に応じてイベントソースの ID で与えられる．タスクの実行者 (アクター) の情報，すなわち「誰が (Who)」情報も必要であれば与えることができる．タスクの実行の際の対象物，使用する資源，方法などは，イベントに付帯する「どのように (How)」情報として与えることができる．資源割り当ての解析と改善のためには，実行者 (これも広い意味で資源である) や使用する資源の情報もログに含める必要がある．このようにログに含めるべき情報は，実施する解析の目的によって決まるものである．

5W1H のうち，最後に残ったのが「なぜ (Why)」の情報である.「理由を説明する情報」の意味論は一般に取り扱いが難しくなると思われるが，意思決定プロセスなど，このような情報が活きるであろうマイニング対象も取り上げられつつある．

[2]実際には，必ずしも制御フローだけではなく，データやルール，組織構造といったパースペクティブ，および，それらの組合せに関しても，広くプロセスマイニングの活動としてとらえることがある．したがって，厳密にいえば，プロセスマイニングの対象は制御フローに限らない．

[3]「スケジュール」後に「開始」するまでの間に「割り当て (assign)」や「再割り当て (reassign)」というイベントを意識するライフサイクルモデルもある [1]

[4]対象システムの種類，および解析の目的によっては,「完了 (complete)」時のタイムスタンプで代表させたほうがよいこともあるかもしれない

3　ログの基本的なデータ構造

3.1　ログ

前回 (第 02 回) で紹介した ProM[3, 4, 5, 6, 7] を初めとするプロセスマイニングのツールでは，何らかの「ログ」を分析の対象としている．このログは，複数の「トレース」から成り，トレースは複数の「イベント」から成る．この階層構造を含め属性情報まで含めたメタデータを図示したものが図 2 である．この図は，後述する XES 形式のログデータのメタモデル [7] であるが，ログ，トレース，イベントの階層構造は，XES 形式に限らずプロセスマイニング一般で共通の概念である．このような構造は，ワークフローの概念から引き継がれたものである．

3.2　トレース

ワークフローの概念において，ある処理のまとまりを「ケース」と呼ぶ．あるケースが実際に実行され，その結果として出力された一連のログのことを，プロセスマイニングではトレースと呼ぶ．

3.3　イベント

システムが出力するログに記録された一つ一つの事象のことを，イベントと呼ぶ．一般に，イベントはタイムスタンプを持っており，どんなタスクが実行されたのかを示すものである．

4　ログのファイル形式

当然のことだが，システムが出力するログは，個々のシステムによって異なる形式である．よって，プロセスマイニングのツールに対して解析したいログを入力するためには，そのツールが理解可能なファイル形式への変換が必要となる．加えて，複数のツール間でログをやりとりするためには，共通のファイル形式が必要となる．そこで，このような用途で用いられているファイル形式のうち，ProM がサポートしている CSV 形式，XES 形式，MXML 形式を紹介する．

なお，この節では，企業内における購買システムを例に取って説明する．購買システムの概要を図 3 に示す．購入を希望する社員と，購買部員がこのシステムを利用するものとする．

図 4 は，ある 1 日に 2 人の社員が物品の購入を購買部員に対して依頼し，うち 1 人は依頼をキャンセルしたことを示すログの例である．この場合，社員 A による申請と，社員 B による申請の，2 つのトレースが得られることになる．

4.1　CSV 形式

プロセスマイニングのツールへログを入力するためのファイル形式で，最も一般的に利用されているものは，可搬性に優れる CSV(Comma Separated Values) 形式である．読者が自前のログ変換ツールを作成することになるとすれば，CSV 形式を利用する可能性を考慮に入れる必要がある．

CSV 形式は汎用のテキストファイルに過ぎないため，ログの意味論に踏み込んだ記録を行うことができない．データも構造化されておらず，それぞれのツールがログを解釈して構造化を行う必要がある．ログの意味論に踏み込むためには，入力時にカラムの意味を指定したり，予めカラムの名前を決めておいたりする必要がある．ProM や，前回 (第 02 回) の連載で紹介した Fluxicon 社の Disco では，CSV 形式の変換時 (ProM) および入力時 (Disco) にカラムの意味 (タイムスタンプ，ケースの識別子，タスクの種類，開始時刻，終了時刻) をユーザに指定させることができる．

図 4 に示したログはスペース区切りになっているため，容易に CSV 形式へと変換することが可能である．図 4 のログを ProM で入力可能な CSV 形式で表現すると，図 5 のようになる．ここで，トレースの識別子を表す “case” の列は，社員の名前を元に復元した．

図 5 に示した CSV 形式のログは，図 6 の通り，ProM 6.5 以降の “Convert CSV to XES” というプラグインによって，XES 形式に変換し，ProM で利用することが可能である．

図 5 に示した CSV 形式のログを，Heuristic Miner でマイニングした結果を，図 7 に示す．

4.2　XES 形式

ProM 6 がサポートしているネイティブの入力ファイル形式が XES(eXtensible Event Stream, エクセスと発音する) 形式 [7] である．XES 形式は，成長し続けるプロセスマイニングの分野において，従来のファイル形式に束縛されることにデメリットが生じたことから提案された．

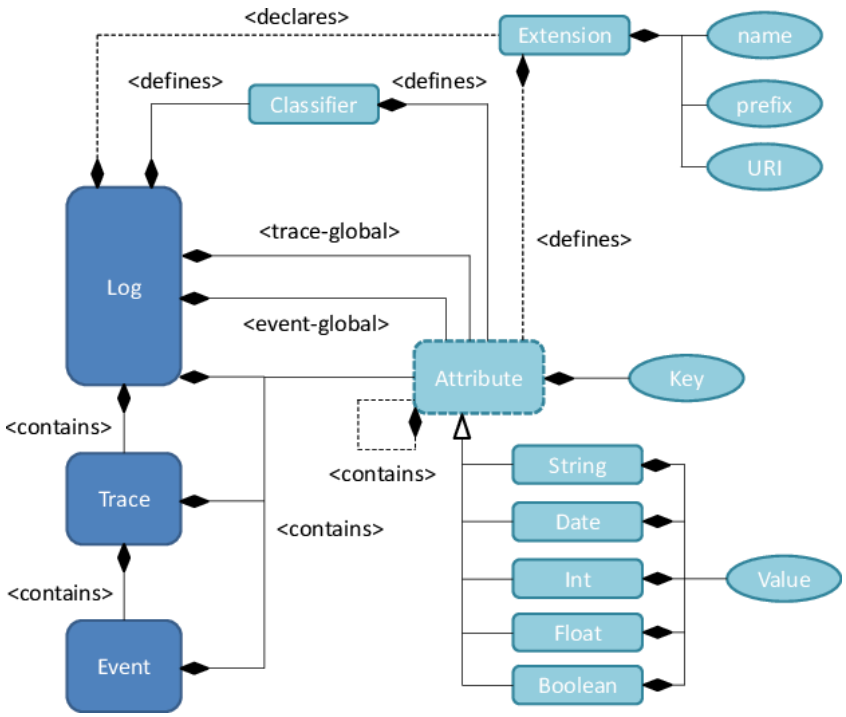

図 2: ログの構造 (後述する XES 形式のメタモデル)[7]

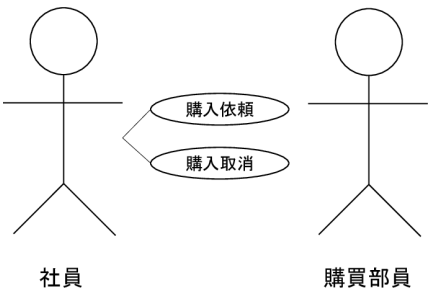

図 3: 購買システム

```
2016/04/04 10:41:10 購入依頼 社員 A 物品 B
2016/04/04 14:05:03 購入依頼 社員 D 物品 E
2016/04/04 16:39:49 購入取消 社員 A 物品 B
```

図 4: ログの例

XES 形式を用いることのメリットは，簡易さ，柔軟性，拡張性，表現力の 4 つである．簡易さは，ログの生成やパースが簡単であることを意味する．柔軟性は，どんな背景を持ったアプリケーションのログであっても対応できることを意味する．拡張性は，将来的に規格を拡張可能であることを意味する．表現力は，ログの情報量を欠落せずに保持できることを意味する．図 8 に，XES 形式の骨格を示す．

XES 形式は ProM 6 以降でサポートされているが，ProM のためだけに策定されたわけではない．XES 形式は，IEEE の標準となることを目指しており[5]，Eindhoven University of Technology により XES 標準が定められているほか，OpenXES[6]という Java によるリファレンス実装を持つ．ProM はこの OpenXES をライブラリとして利用している．

ところで，OpenXES を利用しているソフトウェアには，他にも，XESame[7]という，データベースからのログ収集ツールがある．実データを用いた大規模なプロセスマイニングでは，データベースからログ収集が必要となることがある [8]．な

[5] http://www.xes-standard.org/xesstandardproposal
[6] http://www.xes-standard.org/openxes/start
[7] http://www.processmining.org/xesame/start

```
"case","event","startTime","completeTime"
"0","購入依頼","2016/04/04 10:41:10","2016/04/04 10:41:10"
"1","購入依頼","2016/04/04 14:05:03","2016/04/04 14:05:03"
"0","購入取消","2016/04/04 16:39:49","2016/04/04 16:39:49"
```

図 5: CSV 形式のログの例

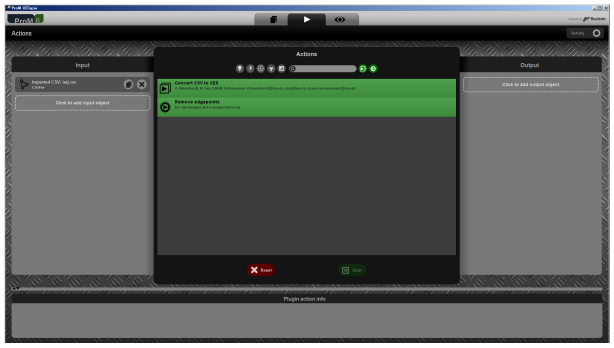

図 6: CSV 形式から XES 形式への変換

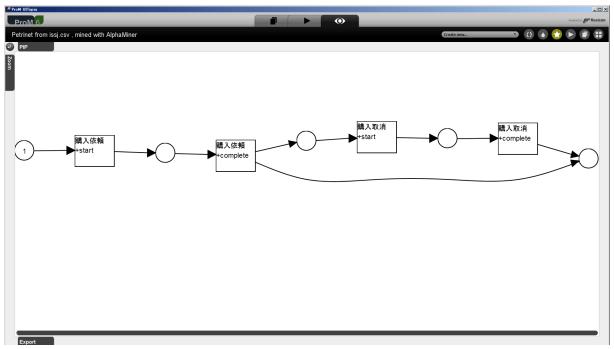

図 7: マイニング結果

```
<log>
  <extension/>
  <classifier/>
  <trace>
    <event>
    </event>
  </trace>
</log>
```

図 8: XES 形式の骨格

お，現在のところ，ProM Import Framework[8] では XES 形式に対応していない．

さて，前述の CSV 形式のデータは，XES 形式への変換後，図 9 の通り，ProM のエクスポート機能を使用して XES 形式での出力が可能である．XES 形式での表現は，図 10 のようになる．

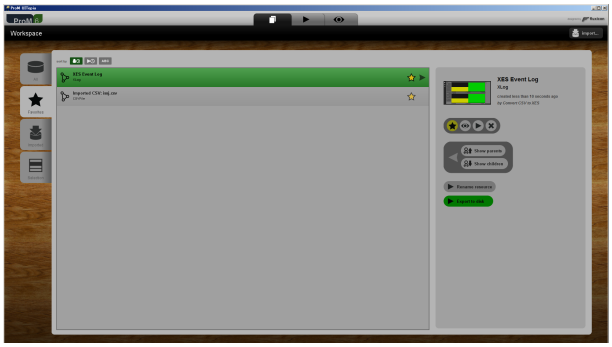

図 9: XES/MXML 形式でのエクスポート

このとき，イベントが start と complete に分割される点が，元のログと異なる．これは，ProM の流儀で，イベントの開始と終了を明確に分けて考えることに起因する．また，タイムスタンプにタイムゾーンが追加されている．

4.3 MXML 形式

ProM 5 まで ProM のネイティブな入力ファイル形式であったのが，MXML(Mining eXtensible Markup Language) 形式[9]である．MXML 形式は XML を利用したファイル形式である．コントロールフローを表現することに主眼を置いており，ワークフローのログを表現するためのシンプルな方法を提供している．MXML 形式の骨格は図 11 の通りである．

MXML 形式は ProM Import Framework という変換ツールで強力にサポートされており，Apache, SAP，People Soft といった著名なソフトウェアのログを MXML 形式に変換可能である．

[8] http://www.promtools.org/promimport/
[9] http://www.processmining.org/logs/mxml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- This file has been generated with the OpenXES library. It conforms -->
<!-- to the XML serialization of the XES standard for log storage and management. -->
<!-- XES standard version: 1.0 -->
<!-- OpenXES library version: 1.0RC7 -->
<!-- OpenXES is available from http://www.openxes.org/ -->
<log xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
  <extension name="Lifecycle" prefix="lifecycle"
                                uri="http://www.xes-standard.org/lifecycle.xesext"/>
  <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
  <extension name="Concept" prefix="concept"
                                uri="http://www.xes-standard.org/concept.xesext"/>
  <classifier name="Event Name" keys="concept:name"/>
  <classifier name="(Event Name AND Lifecycle transition)"
                                keys="concept:name lifecycle:transition"/>
  <string key="concept:name" value="issj.csv"/>
  <trace>
    <string key="concept:name" value="0"/>
    <event>
      <string key="concept:instance" value="0"/>
      <string key="lifecycle:transition" value="start"/>
      <date key="time:timestamp" value="2016-04-04T19:41:10.000+09:00"/>
      <string key="concept:name" value="購入依頼"/>
    </event>
    <event>
      <string key="concept:instance" value="0"/>
      <string key="lifecycle:transition" value="complete"/>
      <date key="time:timestamp" value="2016-04-04T19:41:10.000+09:00"/>
      <string key="concept:name" value="購入依頼"/>
    </event>
    <event>
      <string key="concept:instance" value="1"/>
      <string key="lifecycle:transition" value="start"/>
      <date key="time:timestamp" value="2016-04-05T01:39:49.000+09:00"/>
      <string key="concept:name" value="購入取消"/>
    </event>
    <event>
      <string key="concept:instance" value="1"/>
      <string key="lifecycle:transition" value="complete"/>
      <date key="time:timestamp" value="2016-04-05T01:39:49.000+09:00"/>
      <string key="concept:name" value="購入取消"/>
    </event>
  </trace>
  <trace>
    <string key="concept:name" value="1"/>
    <event>
      <string key="concept:instance" value="2"/>
      <string key="lifecycle:transition" value="start"/>
      <date key="time:timestamp" value="2016-04-04T23:05:03.000+09:00"/>
      <string key="concept:name" value="購入依頼"/>
    </event>
    <event>
      <string key="concept:instance" value="2"/>
      <string key="lifecycle:transition" value="complete"/>
      <date key="time:timestamp" value="2016-04-04T23:05:03.000+09:00"/>
      <string key="concept:name" value="購入依頼"/>
    </event>
  </trace>
</log>
```

図 10: XES 形式のログの例

```
<WorkflowLog>
  <Process>
    <ProcessInstance>
      <AuditTrailEntry>
      </AuditTrailEntry>
    </ProcessInstance>
  </Process>
</WorkflowLog>
```

図 11: MXML 形式の骨格

ProM 6 では，前述の XES 形式が新しい標準形式とされているものの，依然として MXML 形式もサポートされている．このため，MXML 形式と XES 形式のどちらを用いるかは，読者の利用する変換ツールがどちらをサポートしているかなどから，総合的に判断する必要がある．

前述の XES 形式のデータを，ProM を使って MXML 形式に変換すると，図 12 のようになる．トレース単位でデータが構造化されているのがわかる．

5　ツールによる評価実験用ログ生成

各種プロセス発見ツールを試すためのログデータの入手方法は，前回，解説した．その中で，モデルのシミュレーション実行によってログを生成する方法について，MiMo と CPN Tools に関して紹介したが，これらは，ログを生成するための専用ツールというわけではなかった．

ログはアルゴリズムを定量的に比較評価するための重要な入力データであり，公平な定量評価のためのログ生成器 (Log Generator) が開発されている．専用のログ生成器としては，確率自由文脈文法を利用した PLG (Process Log Generator) [9] およびその改訂版である PLG2 [10] がある．また，前回紹介した宣言的なプロセスモデル Declare のためのツール Declare Designer の拡張として Declare モデルからログを生成する機能 [11] などがある．本稿では，この二つについて簡単に紹介する．これらは，文法定義や時制論理による制約表現からのプロセス生成と，そのプロセスを使ったログ生成の二段階をたどることで，指定したタイプのログを生成する．

その他に，オントロジに基づくデータアクセス (ontology-based data access; OBDA) パラダイムに基づく OnTop システムを基礎とした ProM プラグイン (Ontology-Driven Extraction プラグイン) もある [12] が，詳しい紹介は省く．

5.1　前回紹介した MiMo と CPN Tools の場合

前回 (第 02 回) 解説した MiMo においては，そもそもマイニングアルゴリズム (α-アルゴリズム) 自体を WF-net モデルとして実装されている．そのアルゴリズムの WF-net モデルには，別の WF-net でモデル化されたプロセスモデルが同梱されており，これがアルゴリズムのデモンストレーション用のログ生成器に相当するものであった．

このプロセスモデルとマイニングアルゴリズム自体のモデルを直結することで，プロセスモデルが生成したトークンをマイニングアルゴリズムへ入力することができる．もちろん，両モデルを直結するのではなく，ログ生成器のモデルからログをファイルに保存し，マイニングアルゴリズムのモデルがそのファイルからログを読み込ませること（ファイル渡し）もできる．但し，これは決して専用のログ生成器というわけではなく，まさに，「構築したプロセスモデルをシミュレーション実行して，そのログを出力できる」ということに過ぎないが，この機能を使ってアルゴリズムの性能を調べることは可能である．

同じく前回 (第 02 回) は CPN Tools[10] を使ったログ生成も解説した．CPN Tools は，色付きペトリネット (Coloured Petri Net) によるモデリングのためのエディタ/シミュレータである．このために CPN モデルのインスクリプション中で使用可能な，関数型言語 ML の関数ライブラリが提供されている．インスクリプションとは，CPN モデル中のトランジションの出力アークに記述できるスクリプトである．そのスクリプトは，トランジションの発火に伴って遷移するトークンの情報を入力として動作するので，トークンの情報をログとしてファイルに書き出すことができる．したがって，生成したいログの性質を示すようなモデルを CPN で構築すれば，インスクリプションによってタスク (トランジションに対応する) 実行のログをファイルに残すことができる．

この場合も，専用のログ生成器というツールがあるわけではなく,「構築したモデルから，シミュレーション実行時の情報を出力でき，MXML 形式に適合したファイルにケース毎にログを書き出すことができる」ということに相当する．

[10] http://cpntools.org/

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- This file has been generated with the OpenXES library. It conforms -->
<!-- to the legacy MXML standard for log storage and management. -->
<!-- OpenXES library version: 1.0RC7 -->
<!-- OpenXES is available from http://www.xes-standard.org/ -->
<WorkflowLog>
  <Source program="XES MXML serialization" openxes.version="1.0RC7"/>
  <Process id="issj.csv" description="process with id issj.csv">
    <Data>
      <attribute name="concept:name">issj.csv</attribute>
    </Data>
    <ProcessInstance id="0" description="instance with id 0">
      <Data>
        <attribute name="concept:name">0</attribute>
      </Data>
      <AuditTrailEntry>
        <Data>
          <attribute name="concept:instance">0</attribute>
          <attribute name="lifecycle:transition">start</attribute>
          <attribute name="time:timestamp">2016-04-04T19:41:10+09:00</attribute>
          <attribute name="concept:name">購入依頼</attribute>
        </Data>
        <WorkflowModelElement>購入依頼</WorkflowModelElement>
        <EventType>start</EventType>
        <timestamp>2016-04-04T19:41:10+09:00</timestamp>
      </AuditTrailEntry>
      <AuditTrailEntry>
        <Data>
          <attribute name="concept:instance">0</attribute>
          <attribute name="lifecycle:transition">complete</attribute>
          <attribute name="time:timestamp">2016-04-04T19:41:10+09:00</attribute>
          <attribute name="concept:name">購入依頼</attribute>
        </Data>
        <WorkflowModelElement>購入依頼</WorkflowModelElement>
        <EventType>complete</EventType>
        <timestamp>2016-04-04T19:41:10+09:00</timestamp>
      </AuditTrailEntry>
      <AuditTrailEntry>
        <Data>
          <attribute name="concept:instance">1</attribute>
          <attribute name="lifecycle:transition">start</attribute>
          <attribute name="time:timestamp">2016-04-05T01:39:49+09:00</attribute>
          <attribute name="concept:name">購入取消</attribute>
        </Data>
        <WorkflowModelElement>購入取消</WorkflowModelElement>
        <EventType>start</EventType>
        <timestamp>2016-04-05T01:39:49+09:00</timestamp>
      </AuditTrailEntry>
      <AuditTrailEntry>
        <Data>
          <attribute name="concept:instance">1</attribute>
          <attribute name="lifecycle:transition">complete</attribute>
          <attribute name="time:timestamp">2016-04-05T01:39:49+09:00</attribute>
          <attribute name="concept:name">購入取消</attribute>
        </Data>
        <WorkflowModelElement>購入取消</WorkflowModelElement>
        <EventType>complete</EventType>
        <timestamp>2016-04-05T01:39:49+09:00</timestamp>
      </AuditTrailEntry>
    </ProcessInstance>
```

(次ページに続く)

(前ページからの続き)

```
    <ProcessInstance id="1" description="instance with id 1">
      <Data>
        <attribute name="concept:name">1</attribute>
      </Data>
      <AuditTrailEntry>
        <Data>
          <attribute name="concept:instance">2</attribute>
          <attribute name="lifecycle:transition">start</attribute>
          <attribute name="time:timestamp">2016-04-04T23:05:03+09:00</attribute>
          <attribute name="concept:name">購入依頼</attribute>
        </Data>
        <WorkflowModelElement>購入依頼</WorkflowModelElement>
        <EventType>start</EventType>
        <timestamp>2016-04-04T23:05:03+09:00</timestamp>
      </AuditTrailEntry>
      <AuditTrailEntry>
        <Data>
          <attribute name="concept:instance">2</attribute>
          <attribute name="lifecycle:transition">complete</attribute>
          <attribute name="time:timestamp">2016-04-04T23:05:03+09:00</attribute>
          <attribute name="concept:name">購入依頼</attribute>
        </Data>
        <WorkflowModelElement>購入依頼</WorkflowModelElement>
        <EventType>complete</EventType>
        <timestamp>2016-04-04T23:05:03+09:00</timestamp>
      </AuditTrailEntry>
    </ProcessInstance>
  </Process>
</WorkflowLog>
```

図 12: MXML 形式のログの例

5.2 PLG

PLG (Processes Logs Generator) は，確率的文脈自由文法 (Stochastic Context-Free Grammar; SCFG) を用いたログ生成ツールである．PLG には，すでにその後継の PLG2 が開発されており，2015 年 5 月 17 日に第 2.00 版がリリースされているが，本稿では，この PLG2 ではなく，旧版の PLG [9] を紹介する．

本稿執筆時点では，PLG(1.4Beta) は，Java のソースコードを含め，作者の WWW サイト[11]から入手可能である．このソフトウェアは，とても使いやすい GUI を備えているが，そのコマンドライン・インタフェース (CLI) 版 (第 0.1 版; PLGLib 1.2) も，やはり同じ WWW サイト[12] から 2 つの実行可能 jar ファイル (ProcessCreator.jar と LogCreator.jar) の形で入手でき，簡便に動作確認が可能である．

使用方法は容易であり，例を含め上記の WWW サイトに記載されているが，それに従い，簡単に解説する．手順は，2 つのステップに分けられ，最初のステップで指定したパラメータでプロセスモデルを生成し，次のステップで，そのモデルを使ってログ (mxml 形式) を生成する．コマンドライン版では，ProcessCreator.jar と LogCreator.jar のそれぞれの実行可能 Jar ファイルが各ステップに相当する．プロセスモデルを生成する際には以下のようなパラメータを，パラメータとして指定することができる (表 1).

GUI 版も，実行可能 jar ファイルで提供されており，ProcessLogGenerator.jar をダブルクリックするだけで実行することができる．パラメータの設定も容易であり，しかも，プロセスモデルを生成した時点で，ペトリネットモデルを図形的に表

[11] http://padova.processmining.it/sw/plg
[12] http://padova.processmining.it/sw/plgcli

表 1: プロセスモデル生成のためのパラメータ

and 分岐の最大数	整数
xor 分岐の最大数	整数
ループの出現確率	0〜100 までの整数
単一アクティビティの出現確率	0〜100 までの整数
シーケンスの出現確率	0〜100 までの整数
and 分岐の出現確率	0〜100 までの整数
xor 分岐の出現確率	0〜100 までの整数
最大の深さ	整数
プロセスの名前	文字列
モデルファイルを保存するパス	文字列

示できるため理解しやすい．ログの生成もこのソフトウェアの中でボタンをクリックし，パラメータを設定するだけで，実行可能である．ここでは，どのような設定からログを生成できるのかを，この GUI 版のパラメータ設定画面を示すことで紹介する．

図 13 は，GUI 版の PLG のパラメータ設定画面の一つである．ループ，単一アクティビティ，シーケンス，and 分岐，xor 分岐の出現確率を入力できる．

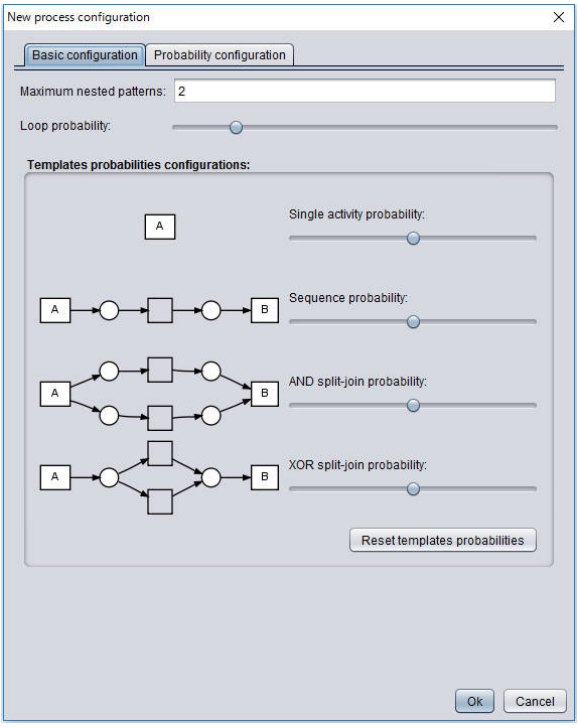

図 13: PLG のパラメータ設定画面 (1)

図 14 も，GUI 版の PLG のパラメータ設定画面の一つであり確率分布を入力する様子がわかる．一様分布，標準正規分布，ベータ分布がサポートされている．

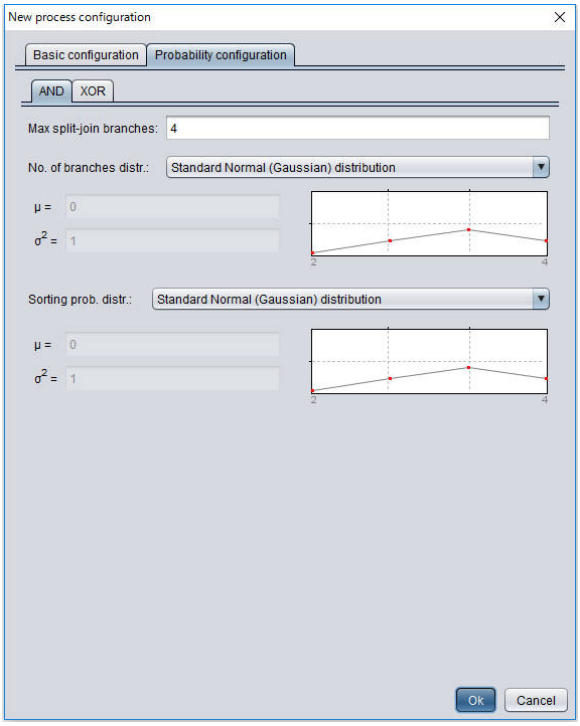

図 14: PLG のパラメータ設定画面 (2)

図 15 は生成されたプロセスモデルの依存グラフ (DG; Dependency Graph) 表現である．この表現では，分岐の種類は明示されていないが，各分岐で枝分かれの確率 (頻度) が表示されている．

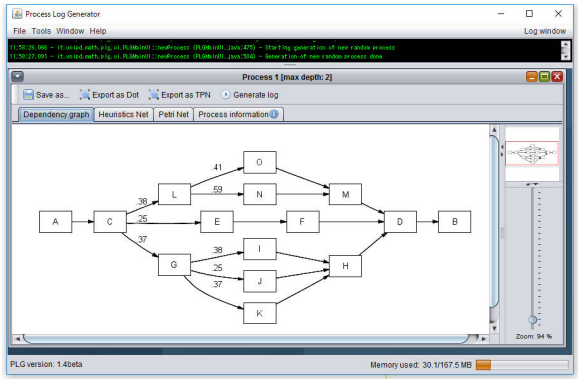

図 15: PLG が生成したプロセスモデル (依存グラフ)

図 16 は生成されたプロセスモデルのペトリネット表現である．この表現では，依存グラフ表現では明示されていなかった分岐の種類を読み取ることができる．

モデルの生成後に，そのモデルを使って，ログを生成することができる．ログは，zip 圧縮された MXML 形式のファイルとして出力される．

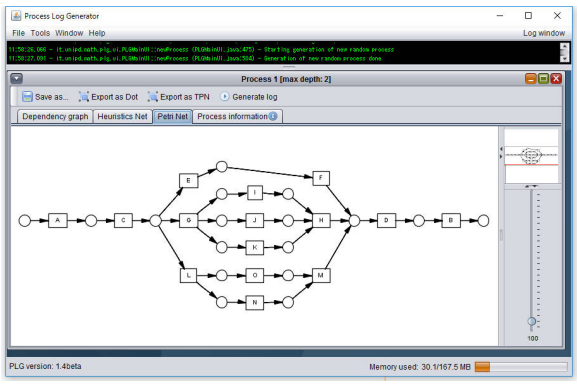

図 16: PLG が生成したプロセスモデル (ペトリネット)

5.3 Declare

Declare は，前回 (第 02 回)，詳しく紹介した宣言的プロセスモデルである．Declare モデルは，タスクの間の時間的な制約関係を，線形時制論理 LTL(Linear Temporal Logic) で与えることで，その制約を満たすプロセスを記述するプロセスモデルである．LTL 式を図形的な表現を使って定義することもできる．この Declare を使った合成的ログ生成ツール [11] も作られており[13]，タスク間の制約関係の集合とアクティビティ名の集合から，それを満たすログを生成することができる．

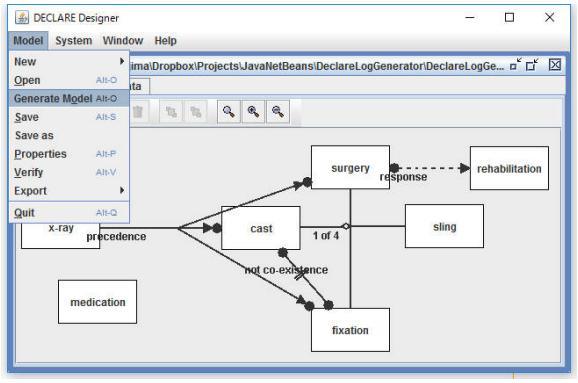

図 17: Declare Designer を拡張した Log Generator

そこでは，定義した Declare モデルを正規表現 (Regular Expression) に変換し，それを有限状態機械 (FSM) でシミュレートすることで，正規表現を満たすログデータを生成する．正規表現は対応する有限状態オートマトン (FSA; Finite State Automaton) で表示できる．そこで，文献 [11] に

[13] https://github.com/processmining/synthetic-log-generator

したがって，Declare 言語による幾つかのタスク間制約表現が対応するオートマトンを図示する (図 18〜21)．アルファベット Σ は，アクティビティ名の集合と対応づけられる．各制約表現の詳細ならびにここで例示しない制約表現に関しては，前回の解説 (第 02 回) を参照いただきたい．

図 18 は，制約表現 $Alt.Prec(b,d)$ から変換された有限状態オートマトンである．これは，

$$prec.(b,d) \wedge \Box(d \Rightarrow \bigcirc(prec.(b,d)))$$

という時制論理式で表現でき，「d が実行されるときには，その前に b が実行される」ということを意味する．

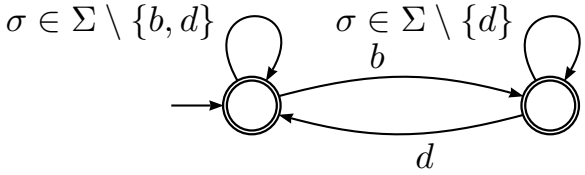

図 18: 制約表現 $Alt.Prec(b,d)$ に対応する FSA [11]

図 19 は，制約表現 $Response(f,g)$ から変換された有限状態オートマトンである．これは，

$$\Box(f \Rightarrow \Diamond g)$$

という時制論理式で表現でき，「f が実行されるときはいつでも，その後で g もいつかは実行されなければならない」ということを意味する．

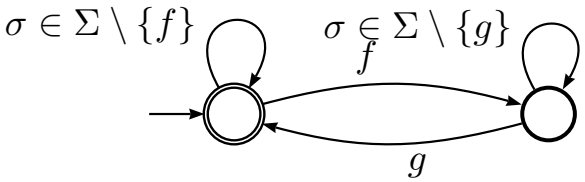

図 19: 制約表現 $Response(f,g)$ に対応する FSA [11]

図 20 は，制約表現 $Precedence(d,a)$ から変換された有限状態オートマトンである．これは，

$$(\neg a \sqcup d) \vee \Box(\neg a)$$

という時制論理式で表現でき，「a が実行されるなら，その前に d も実行される」ということを意味する．

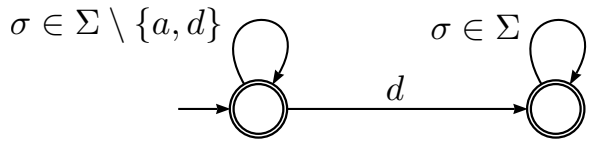

図 20: 制約表現 $Precedence(d, a)$ に対応する FSA [11]

図 21 は，制約表現 $Succession(a, h)$ から変換された有限状態オートマトンである．これは，

$$response(a, h) \wedge precedence(a, h)$$

という時制論理式で表現できる．すなわち，「(1) 各 a の実行後には少なくとも 1 回は h が実行され，(2) h が実行されるなら，先行して a も実行されなければならない」ということを意味する．

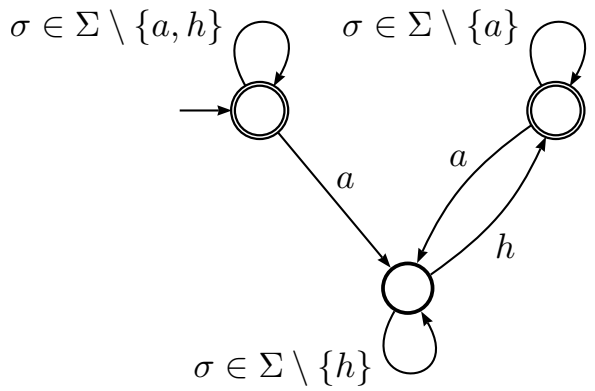

図 21: 制約表現 $Succession(a, h)$ に対応する FSA [11]

個々の制約表現ごとに変換された有限状態オートマトンの積をとり，その合成オートマトンが受理する文を生成することで，与えられた制約集合を満たす，ログデータの集合を得ることができる．

6 ロギングのガイドライン

第 01 回でデータマイニング・マニフェストを紹介したが，最近，ログの取得，すなわち，ロギングに関してもガイドライン (Guidelines for Logging) が提唱されている [13]．以下の (GL1) から (GL12) の 12 項目がある．一般的な感覚として，当たり前と思える項目も多いが，軽んじることなく事前に組織内・組織間で確実な合意を得ておくことが重要と思われる．そこで，こうしたガイドラインをまとめておくことには意義がある．

(GL1) 参照 (reference) と変数 (variable) の名前には，明確な意味論 (semantics) を持たせるべきである．

(GL2) 参照と変数の名前に関して，構造的かつ管理された集まり (collection) を持つべきである．

(GL3) 参照は安定的 (stable) であるべきである．

(GL4) 属性の値は可能な限り正確であるべきである．

(GL5) イベント発生や，その参照ないし属性に，不確定性 (uncertainty) があるのであれば，それを捕捉するべきである．

(GL6) イベントは，少なくとも半順序になっていなければならない．

(GL7) もし可能なら，イベントに関するトランザクション情報も保管すべきである．

(GL8) 定期的に，自動化された整合性と正当性のチェックを実施すべきである．

(GL9) イベントログの相互比較ができることは，時間が経ったものでも，異なるグループに属するものの間でも，様々なケースないしプロセスにわたって保証されるべきである．

(GL10) 解析プロセスの入力として使うイベントログ中のイベントを集約してはならない．

(GL11) イベントを取り除かず，出所 (provenance) を保証せよ．

(GL12) 意味のある相関を失なうことなくプライバシを保証せよ．

ログはプロセスの個々のインスタンスが実行されたときの重要な情報であり，後になって取得しなおそうとしてもできるものではない．失われた情報は，後になって補うことはできない．そこで，後々，必要となるであろう情報は，将来の利用を見越して保管しておく必要がある．

また，「異なる属性に同じ名前を使っている」，「同じ属性に異なる名前を使っている」といった曖昧さがあるログでは比較ができない．そこで，時間が経っても，異なるグループのプロセスであっても，参照や変数の名前にバラツキがないようにオントロジを体系化することは重要である．SA-MXML(Semantic Annotated MXML) は，MXML データ形式の拡張であり，ログ中の概念とオントロジとを結びつける modelReference 属性が追加されている [14, 15]．

さらに，保管されたログデータが独り歩きしてプライバシ上の問題を引き起こすことがあってはならない．データマイニング一般において，プ

ライバシ保護データマイニング (PPDM; Privacy Preserving Data Mining) 手法が提案されている．PPDM 手法では，機密保持のために複数組織に分散保持されたデータは，他組織に流出させないようにしたまま，データマイニングするといったタイプのものや，データを暗号化したまま計算するタイプのものがあるが，これらのタイプはマイニング手法自体を調整するものなので，ロギングのガイドラインで言及するには及ばない．ロギングのガイドラインとしては，適用の前後でマイニングの結果が変わらない範囲で，プライバシを侵害しないようにデータを加工し保管するタイプの手法が対象となる．例えば，ログデータの中に個人を特定できる情報が含まれていると，そこからプライバシを侵害する可能性があるので，データを他組織に渡す前には，匿名化 (anonymization) を施し，同じ対象であることは識別できても，個人を特定できないように，データを加工することなどが該当する．

情報システムにかかわる「人」の情報の重要性は，一層，重要性を増している．データマイニング技術の普及に伴い，情報システムの利用者に関する大規模な情報 (いわゆるビッグデータ) の蓄積は継続的に進められるようになっている．従来は要求されたタスクには直接的には必要のない情報を情報システムに取り入れることは，無駄を避け本質を明確化するという観点からは，むしろ抵抗感があったと思われる．しかし，最近は，直接関係がないとされていた個人の情報を情報システムに取り入れることで，新たな関係性を「発見」し，プロセスを最適化するために役立てることができるという期待も高まっている．他の情報システムのログデータなどの，他の情報源から関連するイベント情報をマージすることも考えられる [16]．情報システムの出力するログに多くの関連情報が結び付けられつつある中，プライバシの保護の観点を無視して通ることはできない．

7　おわりに

今回 (第 03 回) は，プロセス発見アルゴリズムを始めとして，これからさらに事例の蓄積が進むであろう適合性検査やプロセス強化のための各種アルゴリズムへの入力データとなるログデータについて解説した．

将来的な適合性検査やプロセスの強化に向けて，プロセスマイニング技術の実用化にあたっては，それに先立って適切なログデータの記録と保管が不可欠である．長期，かつ，広範囲にわたっての有効利用のためには，データ形式の標準化も重要な課題であり，第 4 節で紹介した XES 形式が有力である．

それとともに，ログデータとして残すべき情報に関して理解を深め，対象領域におけるドメインオントロジを整備し，ロギングのガイドラインをもとに関連組織内外での合意を取り付けることも必要である．むしろ技術的な観点以上に，こうした対象についての意味論的な理解や合意形成に関することこそが，プロセスマイニング技術の発展と普及のための鍵となるのかもしれない．

次回は，様々なプロセス発見アルゴリズムに関して解説する予定である．プロセス発見のアルゴリズムには，いろいろな背景を持ったものが提案されている．そうした各種アルゴリズムの体系的な整理を試みるとともに，その中から幾つかのアルゴリズムを選んで詳述する予定である．今回，プロセス発見アルゴリズムの定量的評価のための，専用の合成的ログデータ生成ツールに関しても紹介した．次回は，プロセス発見アルゴリズムの出力であるモデル評価のための考え方である，適合度 (fitness)，精度 (precision)，一般性 (generalization)，構造的単純さ (structural simplicity) などに関しても言及する予定である．入力したログデータと出力されるモデルの品質との関係を与える指標，出力されたモデル同士を比較する指標など，アルゴリズムの品質評価につながるメトリクスについても簡潔に紹介したい．

今回の執筆分担:
本稿の執筆分担は，第 1 節，第 2 節，第 5 節，第 6 節，および第 7 節を飯島が担当し，第 3 節，第 4 節を田端と斎藤が担当した．全体を通しての用語の統一は飯島が調整した．

文献

[1] W. van der Aalst, Process Mining: Discovery, Conformance and Enhancement of Business Processes, Springer, 2011,2016, http://www.springer.com/jp/book/9783642193446, http://www.springer.com/jp/book/9783662498507, http://www.processmining.org/book/start.

[2] W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters, "Workflow Mining: A Survey of Issues and Approaches," Data and Knowledge Engineering, Vol.47, No.2, pp.237–267, 2003, doi:10.1016/S0169-023X(03)00066-1, http://www.sciencedirect.com/science/article/pii/S0169023X03000661, http://is.ieis.tue.nl/staff/aweijters/WM_overzicht.pdf.

[3] B.F. Dongen, A.K.A. Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. Aalst, "The ProM Framework: A New Era in Process Mining Tool Support," Applications and Theory of Petri Nets 2005: 26th ICATPN 2005, (eds.) G. Ciardo, and P. Darondeau, LNCS-3536, pp.444–454, Springer, 2005, doi:10.1007/11494744_25, http://link.springer.com/chapter/10.1007/11494744_25, http://wwwis.win.tue.nl/~wvdaalst/publications/p264.pdf.

[4] W.M.P. Aalst, B.F. Dongen, C.W. Günther, R.S. Mans, A.K.A. Medeiros, A. Rozinat, V. Rubin, M. Song, H.M.W. Verbeek, and A.J.M.M. Weijters, "ProM 4.0: Comprehensive Support for Real Process Analysis," Petri Nets and Other Models of Concurrency : 28th ICATPN 2007, (eds.) J. Kleijn, and A. Yakovlev, LNCS-4546, pp.484–494, Springer, 2007, doi:10.1007/978-3-540-73094-1_28, http://link.springer.com/chapter/10.1007/978-3-540-73094-1_28, http://alexandria.tue.nl/openaccess/Metis205749.pdf.

[5] W. van der Aalst, B. van Dongen, C. Gunther, A. Rozinat, H. Verbeek, and A. Weijters, "ProM: The Process Mining Toolkit," Proceedings of the BPM 2009 Demonstration Track, (ed.) B.W. A.K. Alves de Medeiros, 2009, http://ceur-ws.org/Vol-489/paper3.pdf.

[6] H. Verbeek, J. Buijs, B. van Dongen, and W. van der Aalst, "ProM 6: The Process Mining Toolkit," Business Process Management Demonstration Track 2010, (ed.) M.L. Rosa, 2010, http://ceur-ws.org/Vol-615/paper13.pdf.

[7] H.M.W. Verbeek, J.C.A.M. Buijs, B.F. Dongen, and W.M.P. Aalst, "XES, XESame, and ProM 6," Information Systems Evolution: CAiSE Forum 2010, (eds.) P. Soffer, and E. Proper, LNBIP-72, pp.60–75, Springer, 2011, doi:10.1007/978-3-642-17722-4_5, http://link.springer.com/chapter/10.1007/978-3-642-17722-4_5.

[8] A. Nammakhunt, W. Romsaiyud, P. Porouhan, and W. Premchaiswadi, "Process mining: Converting data from MS-Access Database to MXML format," ICT and Knowledge Engineering (ICT Knowledge Engineering), 2012 10th International Conference on, pp.205–212, Nov 2012, doi:10.1109/ICTKE.2012.6408557, http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6408557.

[9] A. Burattin, and A. Sperduti, "PLG: A Framework for the Generation of Business Process Models and Their Execution Logs," Business Process Management Workshops: BPM 2010 International Workshops and Education Track, (eds.) M. zur Muehlen, and J. Su, LNBIP-66, pp.214-219, Springer, 2011, doi:10.1007/978-3-642-20511-8_20, http://link.springer.com/chapter/10.1007/978-3-642-20511-8_20, http://andrea.burattin.net/public-files/publications/2010-bpi.pdf.

[10] A. Burattin, "PLG2: Multiperspective Processes Randomization and Simulation for Online and Offline Settings," arXiv.org, 2015, http://arxiv.org/abs/1506.08415,http://arxiv.org/pdf/1506.08415v1.pdf.

[11] C. Ciccio, M.L. Bernardi, M. Cimitile, and F.M. Maggi, "Generating Event Logs Through the Simulation of Declare Models," Enterprise and Organizational Modeling and Simulation: 11th International Workshop, EOMAS 2015, (eds.) J. Barjis, R. Pergl, and E. Babkin, LNBIP-231, pp.20–36, 2015, doi:10.1007/978-3-319-24626-0_2,

http://link.springer.com/chapter/10.1007/978-3-319-24626-0_2.

[12] D. Calvanese, M. Montali, A. Syamsiyah, and W.M.P. van der Aalst, "Ontology-Driven Extraction of Event Logs from Relational Databases," the 11th Int. Workshop on Business Process Intelligence (BPI 2015), LNBIP-256, Springer, 2015, doi:10.1007/978-3-319-42887-1_12, http://link.springer.com/chapter/10.1007%2F978-3-319-42887-1_12, https://www.inf.unibz.it/~calvanese/papers/calv-mont-syam-aals-BPI-2015.pdf, https://www.inf.unibz.it/~calvanese/papers-html/BPI-2015.html.

[13] W.M.P. van der Aalst, "Extracting Event Data from Databases to Unleash Process Mining," BPM - Driving Innovation in a Digital World, (eds.) J. vom Brocke, and T. Schmiedel, pp.105–128, Springer, 2015, doi:10.1007/978-3-319-14430-6_8, http://link.springer.com/chapter/10.1007/978-3-319-14430-6_8, http://wwwis.win.tue.nl/~wvdaalst/publications/z7.pdf.

[14] A.K.A. de Medeiros, C. Pedrinaci, W.M.P. van der Aalst, J. Domingue, M. Song, A. Rozinat, B. Norton, and L. Cabral, "An Outlook on Semantic Business Process Mining and Monitoring," On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops, (eds.) R. Meersman, Z. Tari, and P. Herrero, LNCS-4806, pp.1244–1255, Springer, 2007, doi:10.1007/978-3-540-76890-6_52, http://link.springer.com/chapter/10.1007/978-3-540-76890-6_52, http://wwwis.win.tue.nl/~wvdaalst/publications/p424.pdf.

[15] A.K. Alves de Medeiros, and W.M.P. van der Aalst, "An Outlook on Semantic Business Process Mining and Monitoring," Advances in Web Semantics I: Ontologies, Web Services and Applied Semantic Web, (eds.) T.S. Dillon, E. Chang, R. Meersman, and K. Sycara, LNCS-4891, pp.35–80, Springer, 2009, doi:10.1007/978-3-540-89784-2_3, http://link.springer.com/chapter/10.1007%2F978-3-540-89784-2_3, http://wwwis.win.tue.nl/~wvdaalst/publications/p485.pdf.

[16] F. Mannhardt, M. de Leoni, and H.A. Reijers, "Extending Process Logs with Events from Supplementary Sources," Business Process Management Workshops: BPM 2014, (eds.) F. Fournier, and J. Mendling, LNBIP-202, pp.235–247, Springer, 2015, doi:10.1007/978-3-319-15895-2_21, http://link.springer.com/chapter/10.1007/978-3-319-15895-2_21, http://www.win.tue.nl/~mdeleoni/documenti/DAB14.pdf.

著者略歴

[1] 飯島 正 (いいじま ただし)
慶應義塾大学 理工学部 専任講師 (管理工学科 所属). 慶應義塾大学 理工学部 計測工学科卒業 (1986 年), 同大学院 理工学研究科 修士課程 修了 (1988 年), 同博士課程 単位取得退学 (1991 年). 1990 年より (株) 東芝勤務を経て, 1992 年より 現所属 (助手を経て現職). 博士 (工学). 情報システム学会 元理事 (2007–2008 年 大会担当理事, 2009–2013 年 理事).

[2] 田端 啓一 (たばた けいいち)
2012 年早稲田大学 大学院 修士課程修了, 同年, 日本電信電話 (株) に入社. 以来, ソフトウェアイノベーションセンタにて ソフトウェア工学の研究に従事. 専門分野: プログラム自動並列化, コンピュータアーキテクチャ, ソフトウェアテスト.

[3] 斎藤 忍 (さいとう しのぶ)
2001 年 慶應義塾大学 大学院 修士課程修了, 同年 NTT データに入社. 2015 年 日本電信電話株式会社に転籍. 現在, ソフトウェアイノベーションセンタに所属. 2016 年よりカリフォルニア大学アーバイン校 客員研究員. ソフトウェア工学, 要求工学に関する研究開発に従事. 2007 年 慶應義塾大学 大学院 博士課程修了. 博士 (工学).