知能と複雑系　132－20
(2003. 3. 14)

データマイニングシステム：MUSASHI

矢田 勝俊† 羽室 行信‡ 加藤 直樹* 鷲尾 隆** 元田浩**

†関西大学商学部 〒564-8680 大阪府吹田市山手町 3-3-35
‡大阪産業大学経営学部 〒574-8530 大阪府大東市中垣内 3-1-1
*京都大学大学院工学研究科 〒606-8501 京都府京都市左京区吉田本町
**大阪大学産業科学研究所 〒567-0047 大阪府茨木市美穂ヶ丘 8-1

E-mail: †yada@ipcku.kansai-u.ac.jp, ‡hamuro@adm.osaka-sandai.ac.jp, *naoki@archi.kyoto-u.ac.jp
**{washio, motoda}@sanken.osaka-u.ac.jp

あらまし 本稿では,我々が開発してきた知識発見を支援するMUSASHIと呼ばれるシステムについて紹介する.MUSASHIでは,小さな一つの機能に特化したコマンド群が提供されており,それらのコマンドを組み合わせることによって,XMLで記述された大規模データを処理し,KDDの全プロセスを効率的かつ効果的に支援することが可能となる.MUSASHIはRDBMSなどを導入せずに,大量データ(4ギガバイト2千万レコードを超える)を標準的なPCで処理することが可能である.我々はこれまでにMUSASHIを利用して,企業から提供された大規模なビジネスデータからの知識発見をおこなってきた.

キーワード データマイニング,知識発見,MUSASHI,XML, XMLテーブル

Data Mining System: MUSASHI

Katsutoshi YADA† Yukinobu HAMURO‡ Naoki KATOH*
Takashi WASHIO** and Hiroshi MOTODA**

†Faculty of Commerce, Kansai University 3-3-35 Yamate, Suita, Osaka, 564-8680 Japan
‡Faculty of Business Administration, Osaka Sangyo University 3-1-1 Nakagaito, Daito, Osaka, 567-8530 Japan
*Graduate School of Engineering, Kyoto University, Yoshida-Honmachi, Sakyo-ku, Kyoto, 606-8501 Japan
**Institute for the Scientific and Industrial Research, Osaka University 8-1 Mihogaoka, Ibaraki, Osaka, 567-0047 Japan

E-mail: †yada@ipcku.kansai-u.ac.jp, ‡hamuro@adm.osaka-sandai.ac.jp, *naoki@archi.kyoto-u.ac.jp,
**{washio, motoda}@sanken.osaka-u.ac.jp

Abstract In this paper we introduce a KDD system called MUSASHI developed by our group. It supports the entire KDD process efficiently and effectively based on data in XML format by providing a set of commands each of which performs a single function. MUSASHI can be used independently without other RDBMS technology or KDD software. We can efficiently handle large data (more than 4 gigabytes with 20 million records) on a standard PC. Using MUSASHI, we have succeeded in discovery of useful business knowledge based on a huge amount of real business data provided by several companies.

Keyword Data Mining, Knowledge Discovery, MUSASHI, XML, XML Table

1. はじめに

本稿の目的は,我々がこれまでに独自に開発を進めてきたデータマイニングもしくはデータベースからの知識発見を支える基礎技術,およびその思想について紹介することにある.

データマイニングを実施する際には,大量データを効率的かつ柔軟に操作するための仕組みが必要で,一般的には,リレーショナルデータベース,データウェアハウス,OLAPもしくは各ベンダーが独自に提供するマイニングアプリケーションなどの技術が用いられる.MUSASHIはこれらの諸技術を代替するものとして機能する.MUSASHIでは,いくつかのデータマイニング手法を実現するためのツールが実装されてはいるが,現時点ではオープンにしておらず,本稿におい

ても，そららの機能については解説していない．

MUSASHI は，XML で記述されたデータをその処理対象とする．XML（eXtensible Markup Language：拡張可能なマークアップ言語）とは，W3C(World Wide Web Consortium)にて策定されたデータ記述言語で，利用者が自由にタグを定義することができ，文書構造を柔軟に表現することができる．そのため，企業間のデータ交換用のデータ記述言語として注目されている．

MUSASHI が提案する考え方は非常にシンプルで，XML によって記述されたデータに対して，単一の機能を持つコマンドを組み合わせることによって多様な処理を実現するというものである．これは UNIX が伝統的に主張してきた考えでもあり，効果的な情報システムの構築に非常に有効に機能する．MUSASHI を使えば，十数万円の一般的な PC で，数百万件から数千万件規模のデータ処理も十分に可能である．

本稿の構成は以下の通りである．まず従来のデータマイニング上の問題点を主にシステムの側面から検討する．次に MUSASHI の特徴について述べ，扱うデータ構造，基本的なデータ処理の方法を説明する．そして実際のデータ処理を，事例をあげて詳細に説明し，今後の課題を述べる．

2. 従来のデータマイニング上の問題点

既存の業務システムの多くは基幹業務の遂行を目的にしており，データの中から有用なパターンやルールを発見することを目指して設計しているわけではない．したがって，データマイニングによって，経営に有用な知識を発見しようとすると，様々な問題点が浮かび上がってくる．ここでは，主に 3 つの問題を取り上げる．

2.1. データ消失

現実の経営データからの知識発見では，様々なデータ要求が出てくる．しかし，リレーショナルデータベースのように，業務システムで採用されているシステム技術は，こうしたデータ要求に応えられないことが多い．業務に最適化されたデータベースは，必要のないデータを記録せず，消失させてしまうからである．例えば，レジで精算中の顧客が買い忘れに気づき，売り場に戻った場合，割り込みキーを使って次の顧客の精算を行う．そこで，買い忘れが多い商品を見つけるために，割り込みキーのデータを利用しようとしても，業務データの中に，そうしたデータは存在しない．すでに消失してしまっているのだ．分析に必要なデータは事前に設計することはできない（もし，できるのなら分析する必要はないであろう）．したがって，データの消失を最小限に抑えるシステム技術が必要となる．

2.2. 変更・開発の自由度

データマイニングのプロセスでは，膨大なトライ＆エラーの中で，新しい属性を作りこんだり，新しいアルゴリズムを数多く実装する．過去に作ったプログラムを修正して使うこともあるが，これらの前処理は知識発見にかかる時間の極めて大きな部分を占める [2]．継続的に知識発見を行うためには，こうした新規のプログラム開発や既存のシステムの修正を，迅速，かつ低コストに実現しなければならない．リレーショナルデータベースを代表とする多くのデータ管理ソフトは，データベースへのアクセスを容易にするために，非常に扱いやすいとされる SQL に代表される言語インターフェースを備えている．しかしながら，その言語体系では不可能なもしくは非常に困難な処理要求もしばしば出される．例えば，顧客コードと来店日の表をもとにして，そこから顧客別来店間隔を求めることは非常に難しい．

2.3. 大規模かつ複雑なデータ処理

また，近年の情報機器の普及，業務の自動化によって，多岐にわたるデータが膨大に蓄積されており，そのデータサイズは従来と比較にならないほど，大きなものになっている．我々はこうした膨大なデータから有用な知識を発見しようとするのだが，その大規模かつ複雑なデータを効率的に処理する方法も必要になっている．

それでは，これらの問題をどのように解決することができるのだろうか？我々は MUSASHI というシステムアーキテクチャを開発することによって，上記の問題を解決し，現実のビジネスで効果的な知識発見に成功してきた．本稿では，我々が開発した MUSASHI の内容を詳細に検討する．

3. データマイニングシステム: MUSASHI

MUSASHI (Mining Utility and System Architecture for Scalable processing of Historical data)とは，大量データの効率的処理，継続的な知識発見プロセスを支えるシステムアーキテクチャである [6]．本章では MUSASHI の特徴について，システムの透明性，効率性，従順性の三つの観点から考察し，続いてそれらの実現方法について解説する．

3.1. MUSASHI が目指す 3 つの特徴

3.1.1. 透明性

知識発見を効果的に実施するためには，業務系システムをふくめて，システム全体がユーザにとって分かりやすいものでなくてはならない．上記で指摘したように，データの消失は知識発見にとって最大の問題になる．通常の業務系システムに用いられているリレー

ショナルデータベースなどの既存のシステム技術は，日々の定型業務で必要としないデータは，蓄積されず，捨て去られている．例えば，小売店のPOSシステムで更新される日々の売価変更データなどは，一般ユーザから見れば，その変更履歴が残っていて当然であるにも関わらず，データベース上では更新されてしまっており，その履歴が残っていないケースが多い．また，通常の業務システムで用いるデータベースの構造は複雑で，一般ユーザが理解できるものではない．そのために，新たなデータの蓄積をシステム部門に要求しても，うまく断られるケースも多い．

これらの問題は，すべてシステムが不透明であることが原因している．MUSASHIでは、全ての入力データを、ユーザに分かりやすい形式で蓄積することによってシステムの透明性を高めている。

3.1.2. 効率性

上述のような全ての入力データを蓄積すると膨大なデータを処理する必要が生じる．したがって，MUSASHIはデータ処理の効率性を追及しなければならないが，その効率性は知識発見プロセス全体にとっての効率性でなければならない．1つのデータ処理の効率性を追及しても，知識発見プロセス全体の効率化に必ずしも結びつかない．知識発見プロセス全体の効率化のためには，他の分析者のプログラムを容易に共有できるような仕組みなど移植性を優先する必要がある．一見、必要ないと思われるようなデータをも蓄積することは、非効率な印象を与えるが，経営における知識発見プロセス全体から見れば，効率的知識発見に貢献するものである．

3.1.3. 従順性

知識発見のための前処理はすべてにかかる時間の75%を占める [2]と言われる．そのプロセスは，ユーザー（分析者）のありとあらゆるデータ要求に応えるプロセスである．したがってデータマイニングシステムは，予期できない様々なデータを迅速に低コストで出力できることが不可欠である．すなわち，システムはユーザにとって従順でなければならない．これまで，企業における知識発見プロセスを観察する中で，ユーザが要求するデータ処理にシステムが応えることができない，もしくは数ヶ月もかかってしまうケースを多く見てきた．また入力されているはずのデータが保存されていないために，業務系システムを再設計し，ユーザを待たせるケースも多く見てきた．これらの問題は，知識発見の効率性，有効性に大きく影響するものである．

3.2. MUSASHIが扱うデータ構造

上記の特徴を実現するために，MUSASHIは以下で述べるような実装技術を採用した．通常，バイナリ形式でデータを定義すると，ユーザはそのデータが何を示しているのか，専用のプログラムを使わない限り簡単に確認することができない．

そこで我々はXMLによるデータの記述を採用した．伝統的なUNIXの考え方のようにASCIIフラットファイルでデータを表現するには限界がある．一つの表を記述することを考えても，各項目が何を意味するのかに関する情報（いわゆるデータ辞書）を入れ込むことは困難である．これが，近年XMLに注目が集まる理由の一つである．XML自体もASCIIテキストで記述される．しかしそこでは，データの内容を自由に記述できるタグを入れ込むことができる．

XMLを使うことによって，対象となる事象を非常に柔軟に，そしてわかりやすく記述することが可能になる．単に表構造のデータをXMLで表しただけではその効果がわかりにくいが，例えばPOSレジで行われる作業内容を記述する時には特にその効果は絶大である[3]．しかしながら一方で，XMLで記述するとタグを全てのデータに対して付与せねばならず，単純な表構造のデータに比べて，そのデータ量が飛躍的に増大してしまい，それとともに処理速度の大幅な低下を招いてしまう．そして表構造で表すことのできるデータについてはそのシンプルさも捨てがたい．そこで我々はXMLtableと呼ぶデータ構造を採用することにした．図1にXMLtableによって記述された購買履歴データが示されている

```
<?xml version="1.0" encoding="EUC-JP"?>
<xmltbl version="1.00">
<head>
<title>顧客別購買履歴データ</title>
<field no="1"><name>顧客コード</name></field>
<field no="2"><name>日付</name></field>
<field no="3"><name>商品</name></field>
<field no="4"><name>購買</name></field>
</head>
<body><![CDATA[
 01001    20011206    パン          1
 01001    20011206    牛乳          2
 01001    20011206    牛肉パック    1
 01002    20011015    ビール        6
 01002    20011015    缶チューハイ  6
 01003    20011015    ビール        1
 01003    20011206    牛乳          2
 01003    20011206    パン          1
                                    2

]]></body>
</xmltbl>
```

図 1 : XMLTable による購買履歴の記述

このデータを見て分かるように, XMLtable は完全な XML 文書である. そしてトップの要素<xmltbl>は, <head>と<body>の二つの要素をもつ. body タグには, 表形式のデータが記述され, 項目区切りとしてスペース, 行区切りとして改行が用いられる. この<body>要素内だけを見ると UNIX で標準的に用いられている表についてのデータ構造で, awk などのツール群が適用可能な構造となっている. そして head タグは, このデータに関する辞書として機能する. データのタイトルが<title>タグで示され, そしてそれぞれの項目に関する名前と位置情報が<field>タグによって記述される. <head>タグ内にはそれ以外にも様々な情報を格納することができるが紙面の都合上, ここでは省略する(詳しくは[1]を参照).

MUSASHI では, 純粋な XML と XMLtable を目的によって使い分ける. 例えば, 業務システムで発生するデータは全て純粋な XML として記述されるべきであるし, また多くのデータマイニングアルゴリズムを実装したコマンドは, 多様な情報を出力するために, XMLtable よりも純粋な XML として記述するほうが都合がよい. 一方で, 定型的なデータ加工を行う際には, 処理効率を高めるために XMLtable を利用する. このように MUSASHI は, 純粋な XML と XMLtable を用途によって使い分けることによって, 自由度の高い効率的なシステムの構築を目指している.

3.3. MUSASHI のデータ処理

MUSASHI は単一の機能に特化した小さなコマンド群を提供する. その特徴は, UNIX の伝統的な考え方[7]によって説明可能である.

通常のプログラムは, 驚くほど多くの機能を持っている. これは, 現在の UNIX のコマンド群も例外ではない. プログラムは, 多機能を目指すのではなく, 単機能だが高機能を目指すべきである. 1 つの機能に特化したコマンドを目指すことで, 小さなプログラムも可能になる. 現実には 1 つの機能に特化したコマンドを組み合わせることで, ほとんどの複雑な処理が可能であり, 多機能なプログラムを 1 から作るよりも, これらのコマンドの組み合わせでプログラムを実現した方が, はるかに開発効率, 柔軟性が高い.

すべての特化したコマンドは, テキストデータの標準入出力を前提としたシーケンシャルなファイルアクセスのみを扱い, パイプ機能を用いたシェルスクリプトとして記述され, 実行される [3][6]. 複雑なデータ要求に対しても, こうしたコマンドの組み合わせだけで対応できるため, 開発時間・コストが飛躍的に低減できる. 機能が単純であるために, ソースプログラムは平均で約 200 行程度であり, コマンドのメンテナンス, 高速化が容易である. 特に, 経営データからの知識発見に重要なメリットは次の 2 点である.

第 1 は, 開発, 修正への柔軟性が高くなることである. 膨大なトライ&エラーが必要な分析では, 新しいプログラムの開発, 既存システムの修正を迅速かつ低コストで行うことが求められる. MUSASHI は, コマンドの組み合わせだけでシステムを構築するため, 非常に柔軟性が高くなる. また, 分析には新しいアルゴリズムなど新技術を導入する必要性が生じる. その際にも, 新しいアルゴリズムをコマンドとして開発し, コマンド群に追加するだけで導入が可能であり, システムとの整合性などを考慮する必要性がない. 新しいデータマイニングのアルゴリズムもコーディング作業が終わった時点で導入が可能になる.

第 2 は, プログラムの理解が容易になることである. C 言語などでかかれたプログラムは, それを他人が解読することは容易なことではない. しかし, コマンドの組み合わせが記述されるシェルスクリプトは, 非常に単純なもので, 他人が解読することは比較的容易である. 従って, 過去に行った分析プログラムを再利用することが容易になり, すなわち過去のノウハウを共有することが可能になる.

コマンド名	機能	コマンド名	機能
xml2xt	xml → xmlTable 変換	xtsed	項目毎の文字列変換
xt2xml	xmlTable → xml 変換	xtnulto	Null 値変換
xt2txt	xmlTable → text 変換	xtagg	集計
txt2xt	text → xmlTable 変換	xtcount	行数計算
xtheader	ヘッダー情報の登録変更	xtslide	ある項目を一行ずらす
xtcut	項目の抜き出し	xtnumber	番号付け
xtsubstr	部分文字列の抜き出し	xtrand	乱数生成
xtcal	項目間演算	xtsep	ファイル分割
xtsel	行の条件選択	xtchgnum	数値を範囲で変換
xtuniq	重複行の単一化	xtchgstr	文字列の単純変換
xtselmst	ファイルによる行選択	xtbest	行番号による行選択
xtdelnul	Null 値行の削除	xtsort	並べ替え
xtproduct	直積演算	xtshare	シェア計算
xtjoin	単純結合	xtstatistics	基本統計量の集計
xtnjoin	自然結合	xtcat	ファイルの併合

表 1 : MUSASHI コマンド一覧

表１は，現在，MUSASHI が提供しているコマンド群の一例である．

4. MUSASHIにおけるデータ処理の実際

本章では，我々がこれまでに実施してきたデータマイニングのケースにおける MUSASHI の利用例を簡単に紹介しておく．ここでは，スーパーの販売データを用いたブランドスイッチに関する分析で用いたデータ処理について２つの事例を紹介する．

4.1. ブランドスウィッチ回数

メーカーにとって，他社の新製品の動向は自社の既存ブランドの売上に大きく影響を及ぼす可能性があり，できるかぎり早くに，その影響度合いを把握することが求められる．そこで，ここではある商品カテゴリーにおいて，ある期間中にどのブランドからどのブランドへのブランドスイッチが何回あったかを計算することを考える．

今，表２に示す顧客別の購買履歴データがあったとしよう．これは，ある商品カテゴリー，および，ある一定期間によって既に選択された後のデータで，どの顧客がいつどのブランドを購入したが記録されている．データ中の二人の顧客（01001 と 01002）は，それぞれ「020A→020B→020B→020A」，「020A→020B→020A→020B→020A」の順にブランドを購入していることがわかる．ここで，ブランドスイッチとは，ある一人の顧客の隣接する２回の購入について，あるブランドから他のブランドに商品を買い換えたことと定義する．ただし，以下では便宜上，同じ隣接する二つのブランドが同一であっても，それを同じブランドへのスイッチとして考えることにする．

ファイル名：rireki2.txt

顧客コード	日付	ブランド
01001	20011006	020A
01001	20011110	020B
01001	20011201	020B
01001	20011206	020A
01002	20011015	020A
01002	20011018	020B
01002	20011027	020A
01002	20011106	020B
01002	20011125	020A

表２：顧客別購買ブランドデータ

顧客 01001 は，期間中に４回来店し，020A→020B，020B→020B，020B→020A のブランドスイッチがそれぞれ一回ずつあったことになり，顧客 01002 は期間中に５回来店し，020A→020B，020B→020A のブランドスイッチがそれぞれ２回ずつあったことになる．そしてトータルとして 020A→020B が３回，020B→020B が１回，020B→020A が３回，それぞれブランドスイッチが起きた計算になる．

以上に示したブランド毎のスイッチ回数を MUSAHI で求めるには，図２に示すスクリプトを実行すればよい．

```
xtsort    -k 顧客コード,日付                          <rireki2.txt |
xtslide   -k 顧客コード -f ブランド -a 次回ブランド    |
xtcut     -f ブランド,次回ブランド                    |
xtcount   -k ブランド,次回ブランド -a スイッチ回数     >result4.txt
```

図２：ブランドスウィッチ回数を求めるスクリプト

一行目と二行目のコマンドは連携して一つの仕事をこなしている．まず xtsort によって表を顧客コード，日付の順番で並べ替えている．このことにより，ブランドコードが，各顧客の履歴データにおいて，その購買順序に整列されることになる．そして次の xtslide は非常にユニークな動作をする．ここでは顧客コードを単位にして(-k)，ブランドコードの項目を(-f)一行上にずらす（スライドさせる）のである．そしてずらした結果，新たな項目が一つできあがり，「次回ブランド」と命名されている．２行目までの処理によって，元のデータは表３のように加工されたことになる．

顧客コード	日付	ブランド	次回ブランド
01001	20011006	020A	020B
01001	20011110	020B	020B
01001	20011201	020B	020A
01002	20011015	020A	020B
01002	20011018	020B	020A
01002	20011027	020A	020B
01002	20011106	020B	020A

表３：xtslide までの途中経過

上記のデータを見ると，「次回ブランド」項目がちょうど「ブランド」項目と一行ずつずれていることが確認できる．またスクリプト内の元データの各顧客における最終行については，次回ブランドが存在しないために削除されることになる．

ブランド	次回ブランド	スイッチ回数
020A	020B	3
020B	020A	3
020B	020B	1

表４：ブランドスウィッチ回数

そして，ブランドスイッチの回数は「ブランド」と「次回ブランド」項目における値の組合せについて，

その行数をカウントすればもとまる．そこで，まずxtcutにより必要な項目（ブランド，次回ブランド）を抜き出し，次のxtcountにより，行数をカウントしている．カウントされた値を持つ項目は，「スイッチ回数」と命名されている．最終的に表4に見られる結果を得ることができる．

4.2. ブランドスウィッチパターン

我々は，ブランドスイッチの分析を行うにあたって，しばしば文字列解析手法を用いてきた[5,6]．この手法では，顧客ごとのブランドの購買パターンを文字列で表現し，そこから何らかのルールを発見しようというものである．そこで以下では，表2のデータから文字列解析手法が対象とするデータセットの一部を作成するスクリプトについて見ていく（図3）．

```
xtsort      -k 顧客コード,日付                    <rireki2.txt |
xtchgstr    -f ブランド:文字 -c020A:a,020B:b      |
xtcut       -f 顧客コード -f 文字                 |
xtpattern   -k 顧客コード -f 文字:ブランドパターン  >result5.txt
```

図3：ブランドスウィッチパターンを求めるスクリプト

このスクリプトにおいても，前のスクリプト同様，最初にxtsortによって表を顧客コード，日付の順番で並べ替えている．元データにおいてブランドコードは複数の文字列から構成されるが，文字列解析においては，一文字が一つのブランドを表すことを要求する．そこで，xtchgstrコマンドによって，ブランドコード020A,020Bをそれぞれ"a","b"に変換し，変換後の値を新たな項目「文字」として付け加えている．

そして次に縦に展開された顧客別のブランド「文字」を一つの文字列に連結するために，xtcutで必要な項目を抜き出したあと，xtpatternコマンドが実行される．xtpatternでは，各顧客コードについて(-k)，「文字」項目の値を全て連結し，ブランドパターンという名の項目名として出力する．最終的に表5に示される結果が出力される．

顧客コード	ブランドパターン
01001	abba
01002	ababa

表5：ブランドスウィッチパターン

5. むすび

最後に，我々のMUSASHIに対する基本的な開発・運用理念を示すことによって本稿の結語としたい．我々は，本稿でも見てきたように，リレーショナルデータベースやデータウェアハウスなどの巨大なソフトウェアを代替できるソフトウェアツールとしてMUSASHIの開発に携わってきた．

しかしながら，我々は既存のリレーショナルデータベース等の高度な技術を否定するものではない．ただ，企業が何らかのデータ管理を実現するシステムを構築するにあたっては，リレーショナルデータベースの導入以外に選択肢が無いに等しい現在，我々の提案する考え方が新たな選択肢として考慮され，柔軟性の高いシステムの構築を実現できるインフラとなることを望んでいる．

MUSASHIのプロジェクトはまさに始まったばかりで，ユーザインターフェースの整備や基幹系業務システムへの適用手法など課題も多い．しかし我々は，「基本となるインフラ技術は皆で知恵を出し合って作り，その組合せによる応用のノウハウは企業が独自に保有する」という理念のもと，MUSASHIにまつわる全ての基礎技術はオープンソースとして公開し[1]，企業における効率的かつ効果的な情報システムの一助を担うことのできるコミュニティの構築を目指すものである．

謝　辞

本研究は，平成14年度関西大学重点領域研究助成金，科学研究費補助金（特定領域研究(B)）「構造データからのアクティブマイニング」によって行った．

文　献

[1] Mining Utilities and Data Manipulation Commands for XML Table, http://www.adm.osaka-sandai.ac.jp/~musashi/index.html

[2] U.M.Fayyad, G.P.Shapiro, P.Smyth,"From Data Mining to Knowledge Discovery in Databases: An Overview," *Advances in Knowledge Discovery and Data Mining*, AAAI Press, 1996.

[3] Hamuro, Y., Katoh, N., Matsuda, N., and Yada, K., "Mining Pharmacy Data Helps to Make Profits," Data Mining and Knowledge Discovery, Vol. 2 Issue 4, pp.391-398, December 1998.

[4] Y. Hamuro, H. Kawata, N. Katoh and K. Yada, "A Machine Learning Algorithm for Analyzing String Patterns Helps to Discover Simple and Interpretable Business Rules from Purchase History," Progress in Discovery Science, LNAI 2281, Springer, 2002, pp.565-575.

[5] Y.Hamuro, N.Katoh, E.H.Ip, S.L.Cheung, K.Yada, Dicovery of interesting rules for purchase behavior using string pattern analysis, Proceedings of World Congress on Mass Customization and Personalization, 2001.

[6] Y. Hamuro, N. Katoh and K. Yada:"MUSASHI: Flexible and Efficient Data Preprocessing Tool for KDD based on XML", DCAP2002 Workshop held in conjunction with ICDM2002, 38-49, 2002.

[7] M.Gancarz,"The Unix Philosophy," Butterworth-Heinemann,1996.(芳尾桂監訳『UNIXという考え方』オーム社,2001.