機能実行履歴を用いたソフトウェア利用知識の共有

森崎修司[†] 門田暁人[†] 松本健一[†]
井上克郎[†,††] 鳥居宏次[†††]

ソフトウェア，特にアプリケーションソフトウェアの多くは，今日，非常に多機能となっているが，機能の利用知識の獲得コストが大きいため，実際に利用されている機能はそのごく一部にとどまっている．本稿では，ソフトウェアが提供する機能の利用知識の獲得コストを小さくすることを目的として，ユーザ間で機能実行履歴を共有する方式を提案する．提案方式では，ネットワークで結ばれた計算機を使用する複数のユーザから機能実行履歴を自動的に収集し，ユーザごと，ソフトウェアごとに整理したうえで，機能の実行頻度や時期等の情報を付加して，ユーザ間で相互参照可能とする．Microsoft Word 2000 を対象とした評価実験の結果，被験者 5 人のうち 4 人において利用知識の獲得が認められた．特に，知識獲得により初心者は作業効率が向上し，熟練者がその存在を想像さえしていなかった機能についても利用知識の獲得がなされた．

Sharing Usage Knowledge for Application Software Using Function Execution History

Shuuji Morisaki,[†] Akito Monden,[†] Ken-ichi Matsumoto,[†]
Katsuro Inoue[†,††] and Koji Torii[†††]

Most of software, especially application software provides us with various functions. Since the acquisition cost of the use knowledge of a function is large, the number of functions actually used is limited. This paper proposes system that shares a functional execution history among users for the purpose of making small acquisition cost of the use knowledge of the function which software offers. The proposed system can collect functional execution histories from users automatically through computer network. It also provides a mechanism for exchanging the use knowledge of a function among users by sharing functional execution histories. Four subjects were able to acquire use knowledge among five persons as a result of the evaluation experiment for Microsoft Word 2000. Efficiency of novice work improved by knowledge acquisition. The use knowledge could be acquired also about the function in which an expert does not know.

1. はじめに

ワープロソフト等の一般向けパッケージソフトウェアが提供する機能の数は増加の一途をたどっている．たとえば，Microsoft Word 2000 の場合，メニュー項目数，ツールバー上のショートカットボタン数，そして，コンテキストメニュー（マウスの右ボタンをクリックすると現れるメニュー）の項目数を合計するとその数は 1000 以上となる．1 つのメニュー項目や GUI ボタンがソフトウェアの 1 つの機能に対応しているとは限らないが，非常に多くの機能が提供されているといえる．

しかし，ソフトウェアの提供するすべての機能がユーザによって利用されているわけではない．たとえば，文献 10) では，UNIX のシェルの 1 つである tcsh が提供するコマンド編集機能のうち，実際にユーザが使う機能（利用機能）は全体の 50%にも満たないことが報告されている．我々が行った予備実験においても，A4 判で約 190 ページ（文字数約 10 万，図表数 72）のソフトウェア設計書を Microsoft Word 2000 で作成する作業において，利用された機能数（メニュー項目数と GUI ボタン数の合計）は約 70，すなわち，全体の約 7%であった[6),12)]．

† 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology
†† 大阪大学大学院基礎工学研究科
Graduate School of Engineering Science, Osaka University
††† 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

一般に，利用機能数はソフトウェアの使用時間とともに増加するが，ある程度まで増加すると頭打ちになることが知られている．たとえば，文献 11) の実験では，tcsh が提供する機能数の約 30%程度までしか利用機能数が増加しないことが確かめられている．しかも，利用すれば作業効率がより高くなる機能がソフトウェアによって提供されているにもかかわらず，それら機能の利用知識をユーザが積極的に獲得しようとするとは限らないことも観察されている[3]．

利用機能数が頭打ちとなる原因の 1 つは，新たな機能の利用知識を獲得するコスト（試行錯誤を行ったりオンラインヘルプを参照したりするための時間，工数，精神的負担）が高いことにある．ここで機能の利用知識には，単に機能の入出力関係だけでなく，機能が有効に働くコンテキスト，利用するための具体的な操作方法（対応するメニュー項目や GUI ボタンの場所や起動方法，前提操作等），も含まれるものとする．利用知識を獲得すればその機能は利用可能となり，ユーザの作業効率は高くなる．ただし，利用知識を持つ機能数が増加するにつれて，利用知識の獲得による作業効率の上昇幅は徐々に小さくなる．利用知識の獲得コストが一定であるとすると，作業効率の上昇幅がある値以下になれば，利用知識を新たに獲得する必要性をユーザは感じにくくなる．新たな機能を利用しなければ必要な作業が実行できない場合は別であるが，利用知識をすでに持つ機能やその組合せでユーザは作業を行うようになる[9]．

利用機能数が頭打ちとなるもう 1 つの原因は，ソフトウェアが提供する機能，あるいは，提供するかもしれない機能の存在自体をユーザが想像できるとは限らないことにある．特に，ソフトウェアの利用経験の少ないユーザには，存在自体を想像できない機能が多くある．たとえば，初心者ユーザが，いわゆるアンドゥ機能というものを想像し，ソフトウェアが提供していると期待するとは考えにくい．最近では，アプリケーションソフトウェアの多くがオンラインヘルプ機能を持ち，ヘルプに対する質問文検索やキーワード検索を可能としている．しかし，ソフトウェア本体の多機能化にともなって大規模化しているヘルプ記述の中から，存在すら想像できない機能を見つけ出すことは，多くのユーザにとって容易なことではない．

利用知識の獲得コストを小さくし，ユーザがその存在を想像できる機能の数を多くする 1 つの方法として，利用知識をユーザ間で共有することが考えられる．特に，その存在を想像できる機能がユーザによって異なれば，共有によってその数が大きく増えることが期待される．ただし，機能の利用知識は，いわゆる「暗黙知」の一種である．共有可能な形式とすることはもちろん，ユーザから知識の提供を受けること自体が容易ではない．また，機能数と同じかそれ以上の数の利用知識を共有するためには，個々のユーザにとって有用な利用知識を選択してユーザに提供する，あるいは，ユーザが容易に選択できるしくみが必要である．単にユーザ間で利用知識を相互参照できるだけでは，ソフトウェアの利用機能数が頭打ちになっているのと同様に，共有する利用知識もその利用数が頭打ちになる可能性がある．

本稿では，アプリケーションソフトウェアが提供する機能に関するユーザの利用知識が，ソフトウェアの機能実行履歴中に表れる（利用知識は機能実行履歴として有形化可能である）と仮定し，有形化した利用知識をユーザ間で共有する方式を提案する．

以降，2 章で提案方式を述べ，3 章でその提案方式に基づく試作システムを紹介する．4 章では評価実験，5 章では，実験結果について考察する．6 章で関連研究を紹介し，7 章でまとめる．

2. 提案方式

2.1 概　　要

提案方式では，ソフトウェアの利用知識はソフトウェア機能の実行履歴として有形化できるものとする．ここで，機能実行履歴とは，ソフトウェア利用時にユーザが実行した機能，具体的には選択されたメニュー項目や GUI ボタンの名称を，ユーザごと，ソフトウェアごとにその実行日時とともに時系列に並べたものである．機能実行履歴の例を**図 1** に示す．図 1 の例では，1 つの行が 1 つの機能実行に対応し，機能実行ごとに「実行日」,「実行時刻」,「メニュー項目名，あるいは，GUI ボタン名」が記録されている．特に，メニュー項目名については，メニュー階層内での位置が分かるように，トップメニューからの選択経路が付加されている．たとえば，図 1 に示す履歴の先頭行の場合，選択されたメニュー項目「フォント」は，トップメニューの「書式」のサブメニュー内に位置することが分かる．同様に，GUI ボタン名については，GUI

```
2000/02/03 18:50:41 書式->フォント(&F)
2000/02/03 18:50:45 書式設定->フォント サイズ(&F)
2000/02/03 18:50:48 標準->中央揃え(&C)
2000/02/03 18:51:16 ファイル->上書き保存(&S)
2000/02/03 18:51:23 ファイル->終了(&X)
```

図 1 機能実行履歴の例
Fig. 1 An example of function execution history.

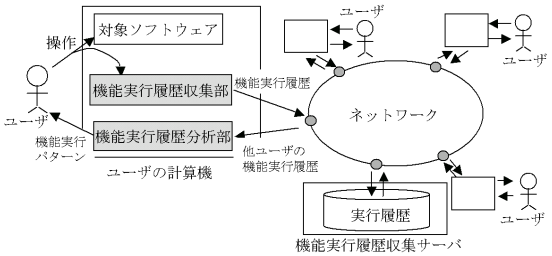

図 2 提案方式の概要
Fig. 2 Overview of proposed system.

ボタンが位置するツールバー名が付加される．たとえば図 1 に示す履歴の 3 行目の場合，選択された GUI ボタン「中央揃え」は,「標準」ツールバー上に位置することが分かる．

提案方式における「利用知識の共有」とは，機能実行履歴を複数のユーザから収集し，機能の実行頻度や頻出する特定の機能実行系列（機能実行パターン）等の情報を付加して，ユーザ間で相互参照可能にすることである．その概略を**図 2** に示す．利用知識を共有しようとするユーザは,「機能実行履歴収集サーバ」とネットワークで結ばれた計算機上でソフトウェアを利用する．すると，ユーザの計算機上で稼動する「機能実行履歴収集部」が機能実行履歴を収集し，ネットワークを介して「機能実行履歴収集サーバ」へと送出する．機能実行履歴の収集と送出は自動的，かつ，ユーザの作業を妨げないように行われる.「機能実行履歴収集サーバ」はユーザからの要求に応えて，蓄積した機能実行履歴をネットワーク経由でユーザに提供する．ユーザの計算機上で稼動する「実行履歴分析部」は，提供された機能実行履歴における各機能の実行頻度の算出，機能実行パターンの抽出等を行い，ユーザ間での利用知識の交換を支援する．以降では，機能実行履歴を収集する 3 つの方式，利用知識を獲得する 2 つの方式について述べる．

2.2 機能実行履歴の収集方式

ソフトウェアの機能実行履歴を収集する 3 つの方式を**図 3** に示す．各方式では，グラフィカルユーザインタフェース（GUI）を持つソフトウェアを前提とし，GUI ツールキットにより呼び出されたコード（メソッド）の履歴を機能実行履歴として収集できる．一般に，GUI に対するユーザの操作（キーボード入力，マウス操作等）の情報は，まず OS が受け取り，GUI ツールキットと呼ばれるミドルウェアを介してアプリケーションソフトウェアに伝えられる．GUI ツールキットは，ユーザの操作と実行されるメソッドとの対応を管理し，ユーザがある GUI 操作を行ったときに，その操作に対応したメソッドを呼び出す機能を持つ．GUI ツールキットの例としては，Microsoft Windows における MFC（Microsoft Foundation Class），UNIX における XLib，XToolKit，GTK（GNU Took Kit），Java における AWT（Abstract Window Toolkit），Swing 等があげられる．本稿では，GUI ツールキットにより呼び出された個々のメソッドがソフトウェアの各機能に対応すると見なし，メソッドの実行履歴を機能実行履歴として収集する．提案する 3 つの方式は，それぞれ適用範囲が異なるため，実装者は，対象ソフトウェアに応じた方式を選択することになる．

(C1) プラグイン方式
実行履歴収集部を，対象ソフトウェアのプラグインモジュールとして実装し，ユーザが機能実行するたびに呼び出されるようにする（図 3 (a)）．

対象ソフトウェアがプラグインモジュールを提供し，かつ，プラグインインタフェースからユーザの機能実行が監視できる場合に適用できる．適用できるアプリケーションの例としては，Microsoft Office（Word，Excel，PowerPoint 等），Adobe Photoshop，Illustrator 等があげられる．本方式の利点は，あらかじめ定められたインタフェースに従ってプラグインモジュールを作成すればよいので実装が容易である点と，ソースプログラムが公開されていない市販アプリケーションに適用できる点である．一方，欠点は，収集可能な機能実行履歴がプラグインインタフェースの仕様に依存するため，すべての実行機能を監視できるとは限らない点である．

(C2) 対象ソフトウェア変更方式
実行履歴収集部を，対象ソフトウェアに組み込み，機能実行のたびに呼び出されるようにする（図 3 (b)）．自作ソフトウェアや GNU のフリーソフトウェア等，ソースコードが入手できる場合に適用できる．本方式の利点は，ソフトウェアに含まれるすべての機能の実行履歴を記録するように実行履歴収集部を作成可能な点である．欠点は，多くの市販ソフトウェアのソースプログラムが公開されていない現状では，適用範囲が限定される点である．

(C3) ツールキット変更方式
実行履歴収集部を，GUI ツールキットに組み込む（図 3 (c)）．GUI ツールキットのソースリストが公開されている場合に適用できる．MFC や GTK 等をはじめとして，現在利用されている GUI ツールキットの多くはソースコードが公開されている．

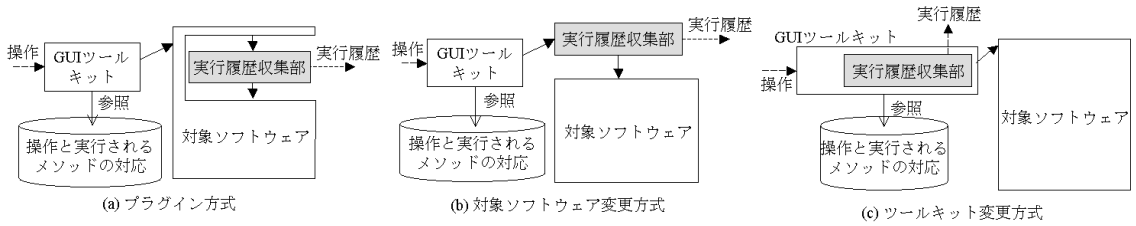

図 3 機能実行履歴の収集方式
Fig. 3 Architecture for collecting function execution history.

本方式の利点は，アプリケーションごとに実行履歴収集部を作成する必要がない点である．あるGUI ツールキットに対する実行履歴収集部は，そのGUI ツールキットを用いるすべてのアプリケーションから実行履歴を収集できる．欠点は，GUI ツールキットに対する深い知識が必要が必要であり，実装が必ずしも容易でない点である．

2.3 利用知識の共有方式

利用知識の共有によって各ユーザが自分にとって有用な情報を得るためには，どのユーザ間で共有を行うか，および，どの知識を共有すべきかを適切に選ぶことが重要となる．ユーザにとって有用な機能は，ソフトウェアを用いてユーザが行う作業の種類に依存する．たとえば，同じワードプロセッサを使う場合でも，年賀状を作成するユーザと論文を作成するユーザとでは，有用となる機能に違いがあると考えられる．そこで，2.3.1 項では，共有すべき知識の獲得方法についてまず述べる．

一方，共有すべき知識の提示方法もユーザが有用な知識を得るうえで重要である．2.3.2 項では，ソフトウェアの利用知識である機能実行履歴をどのように加工してユーザに提示するかについて述べる．

2.3.1 利用知識の獲得方法

利用知識を獲得する 2 通りの方法を提案する．

(E1) 同一タスク方式

知識共有を行いたいユーザが集まり，各人があらかじめ定められた同一タスクを行い，タスク遂行時の機能実行履歴を交換する．2 人のユーザ a，b が同一タスクを行った場合に実行された機能の集合を**図 4** に示す．図 4 中の四角形はソフトウェアが提供する機能の集合を，F_a はユーザ a が実行した機能の集合を，F_b はユーザ b が実行した機能の集合を，それぞれ表す．図に示されるように，一般に，ユーザはソフトウェアの提供する機能の一部のみを使う．また，複数のユーザが同一タスクを行ったとしても，各ユーザが同じ利用知識を持つとは限らないため，実行される機能に差が生じる．ユーザ a と b が知識共有を行う場合には，

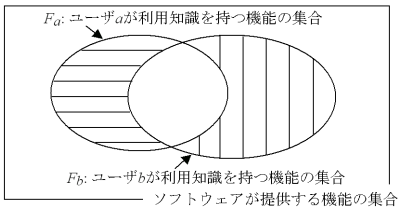

図 4 複数ユーザ間での機能実行履歴の共有
Fig. 4 Sharing function execution history among multiple users.

ユーザ a には $\overline{F_a} \cap F_b$ を，ユーザ b には $F_a \cap \overline{F_b}$ を提示することにより，効率良く実行履歴を共有することができる．

本方式は，同一組織内で普段から似た作業を行っているユーザ間で知識を共有したい場合に特に有効である．あらかじめ定めるタスクは，普段の行う作業を数多く含むように設定することが望ましい．

(E2) 類似度推定方式

互いに似た作業を行っているユーザを推定し，推定されたユーザ間で機能実行履歴を交換する．実行する機能の種類が似ているユーザは似た種類の作業を行っていると見なして，知識を共有すべきユーザとして推定する．たとえば，図 4 において，F_a と F_b の重なり部分が大きく，かつ，完全には一致していない場合である．本方式では，特定のタスクをユーザがわざわざ行う必要がない．

2.3.2 利用知識の提示方法

前項で述べたように，利用知識の共有とは，ユーザ自身が実行していない機能で，かつ，他のユーザが実行した機能をお互いに提示し合うことである．各ユーザは，提示された機能を（オンラインヘルプ等により）学習することで，利用知識を増やすことができる．ただし，提示された機能のすべてがそのユーザにとって有用であるとは限らない．有用である可能性の高い機能のみを選んで学習することが望ましい．また，提示された機能が数多くある場合にも，優先的に学習すべき機能を選べることが望ましい．したがって，学習すべき機能を決定するための情報をユーザに提示する必

要がある．

また，ソフトウェアによっては，実行された複数の機能の系列（機能実行パターン）が，ユーザにとって意味のある場合がある．個々の機能に関する情報だけでなく，機能実行パターンに関する情報も提示する必要がある．

以上の議論に基づき，以下では，ユーザに提示すべき情報を列挙する．

- 各機能の実行頻度と実行時期
  作業全般にわたって頻繁に使われている機能ほど，作業全般の効率を向上させるのに有用であると考え，優先的に学習すべきと見なせる場合がある．したがって，各機能の実行頻度に関する情報をユーザに提示することが重要となる．ただし，実行頻度の小さい機能でも，作業の初期や最後に一度だけ実行することが重要となる機能も存在するため，機能の実行時期に関する情報も提示する必要がある．たとえば，Microsoft Word における「文字/段落スタイルの設定」の機能は，文書作成作業の初期に使用することで作業効率を上げることが可能である．
- ユーザ間での機能の実行割合とユーザの特性
  より多くのユーザが使用している機能ほど，優先的に学習すべきと見なせる場合がある．したがって，機能を実行したユーザの数や割合を提示することが重要となる．ただし，1 人のユーザのみが実行した機能でも，そのユーザが熟練者であれば，優先的に学習すべきと見なせる場合がある．そこで，各機能を実行したユーザの特性（誰が実行したのか，初心者なのか熟練者なのか等）の情報も提示することが重要である．
- 機能実行パターン
  ソフトウェアによっては，機能実行パターンがユーザにとって有意味となる場合がある．たとえば，Microsoft Word における「目次作成」機能を利用するためには，あらかじめ「スタイル」の定義を行う必要があり,「スタイル」「目次作成」という 2 つの機能実行系列が揃ってはじめてユーザにとって役立つ情報になる．

上記の情報以外に，機能実行系列全体を閲覧できるようにしておくことも重要である．ユーザがある機能を使って作業を遂行できるためには，その機能の実行方法だけでなく，その機能を実行するべきコンテキストを知る必要がある．そこで，各機能が実行されたコンテキストを知るための手掛かりとして，各機能が実行された前後の機能を見られるようにしておくことが考えられる．

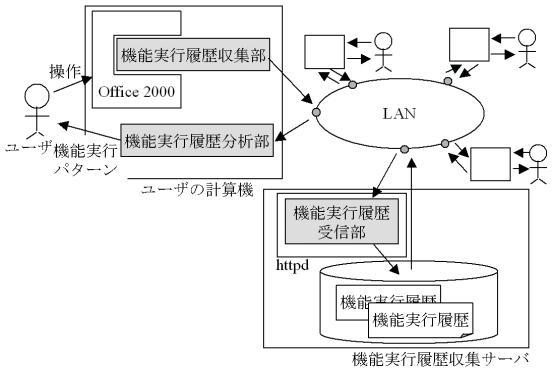

図 5 試作システムの構成
Fig. 5 Overview of prototype system.

3. 試作システム

3.1 概　　要

機能実行履歴収集部をプラグイン方式，獲得方法を同一タスク方式とするシステムを試作した．対象ソフトウェアは Microsoft Office 2000 である．**図 5** は試作システムの概要を示したものである．ユーザの計算機上には Office 2000 とともに機能実行履歴収集部が稼働する．機能実行履歴収集部はユーザの操作を監視し，ユーザの機能実行履歴を機能実行履歴収集サーバへ送出する．ユーザの計算機と機能実行履歴収集サーバはイーサネットで結ばれている．

ユーザが通常どおり対象ソフトウェアを操作すると機能実行履歴が蓄積される．ユーザ操作のうち，メニュー選択とツールバー上の GUI ボタンの押下による機能実行が機能実行履歴として収集される．収集された機能実行履歴は，自動的に機能実行履歴収集サーバに送出，蓄積される．ユーザは必要に応じて，機能実行履歴分析部から他ユーザ機能実行パターンを参照することができる．機能実行履歴そのものを参照することも可能である．

3.2 機能実行履歴収集・受信部

機能実行履歴収集部を Office 2000 が提供する add-in と呼ぶプラグインモジュールとして実装した．プラグインモジュールは，Visual Basic で記述され 150 行程度である．メニュー選択，ツールバー上の GUI ボタンの押下によるすべての機能実行のたびに，その内容がプラグインに通知される．ユーザの機能実行が通知されると，機能実行履歴収集部は，機能実行履歴を一時ファイルに蓄積し，一定量に達すると，実行履歴収集サーバに送出する．

機能実行履歴受信部は，実行履歴収集サーバに常駐

表 1 機能実行履歴と機能実行パターンの例
Table 1 An example of function execution pattern.

P_1	$P_{1,1}=\{1\}, P_{1,2}=\{2\}, P_{1,3}=\{3\}, P_{1,4}=\{4\}$
P_2	$P_{2,1}=\{1,2\}, P_{2,2}=\{1,4\}, P_{2,3}=\{2,4\}, P_{2,4}=\{3,1\}, P_{2,5}=\{4,3\}$
F_1	$F_{1,1}=\{1,2\}, F_{1,2}=\{1,0\}, F_{1,3}=\{1,1\}, F_{1,4}=\{1,1\}$
F_2	$F_{2,1}=\{1,0\}, F_{2,2}=\{0,1\}, F_{2,3}=\{1,0\}, F_{2,4}=\{0,1\}, F_{2,5}=\{1,1\}$

し，機能実行履歴収集部から送出された実行履歴をソフトウェアごと，ユーザごとに階層化し，テキストファイルとして蓄積しておく．これらのディレクトリは OS が提供するファイル共有機能を利用して，各ユーザの計算機から読み取り可能である．

3.3 機能実行履歴分析部

機能実行履歴分析部は，機能実行履歴収集サーバに蓄積された各ユーザの機能実行履歴を読み込み，共通に含まれる機能実行パターンを抽出するとともに，ユーザごとの出現頻度を調べる．なお，現時点では，機能実行パターンとは「機能実行履歴中の任意の部分系列」とする．したがって,「パターン」を切り出すための区切り子等は設定していない．

具体的には，すべてのユーザの機能実行履歴の集合 $\mathcal{O}$ から出現頻度 1 以上の機能実行パターンの集合 $\mathcal{P}$ を求め，各機能実行パターンのユーザごとの出現頻度 $\mathcal{F}$ を調べる．ただし，出現頻度を調べる機能実行パターンの最大長を設定する．

$o_{i,h}$ をユーザ i の機能実行履歴の h 番目の機能，$P_{j,k}$ を長さ j の 1 機能実行パターン，$f_{j,k,i}$ をユーザ i の機能実行履歴における $P_{j,k}$ の出現頻度とし，$\mathcal{O}$ から $\mathcal{P}$，$\mathcal{F}$ を，次のように求める．

すべてのユーザの機能実行履歴の集合 $\mathcal{O}$：

$\mathcal{O}=\{\mathcal{O}_1,\cdots,\mathcal{O}_i,\cdots,\mathcal{O}_t\}$ (t：ユーザ数).

ユーザ i の機能実行履歴 O_i：

$O_i=o_{i,1},\cdots,o_{i,n_i}$

(n_i：ユーザ i の機能実行履歴の長さ).

長さ j の機能実行パターンの集合 P_j：

$P_j=\{P_{j,1},\cdots,P_{j,k},\cdots,P_{j,l_j}\}$

(l_j：$\mathcal{O}\in P_j$ である P_j の数).

長さ j の機能実行パターンの集合 P_j のある要素 $P_{j,k}$：

$P_{j,k}=p_{j,k,1},\cdots,p_{j,k,j}$

(ただし，$1\le s\le n_i-j+1$ を満たす s に対して $p_1=o_{i,s},p_2=o_{i,s+1},\cdots,p_j=o_{i,s+j-1}$).

あるユーザ i の機能実行履歴 O_i に含まれる $P_{j,k}$ のおける出現頻度を $f_{j,k,i}$ $(1\le i\le t)$ とし，

$F_{j,k}=f_{j,k,1},\cdots,f_{j,k,i},\cdots,f_{j,k,t}$

長さ j の機能実行パターンの出現頻度の集合 F_j：

$F_j=\{F_{j,1},\cdots,F_{j,k},\cdots,F_{j,l}\}$.

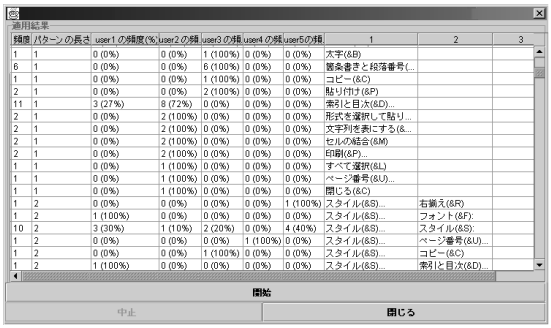

図 6 機能実行履歴分析部の出力画面
Fig. 6 Screenshot of function execution history analysis sub-system.

機能実行パターンの最大長を m として，機能実行パターンの集合 $\mathcal{P}$，機能実行パターンの出現頻度の集合 $\mathcal{F}$：

$\mathcal{P}=\{P_1,\cdots,P_j,\cdots,P_m\}$

$\mathcal{F}=\{F_1,\cdots,F_j,\cdots,F_m\}$

表 1 は抽出するパターンの数 m を 2 とし，2 人のユーザの機能実行履歴をそれぞれ $O_1=\{1,2,4,3\}$，$O_2=\{1,4,3,1\}$ とした場合の例である．

f_u が 0，もしくは，f_s $(1\le s\le t, s\ne u)$ よりも十分小さい $P_{j,k}$ をユーザ u に提示する．

$f_{j,k,u}$ $(\mathcal{F}\in f_u)$ の最低値を設けることにより（頻度が低い実行系列を枝刈することにより）$\mathcal{P}$ の総数を小さくすることも可能である．

図 6 は機能実行履歴分析部のスクリーンショットで，ユーザへの出力画面である．画面中央の表の各行が，分析部で抽出された機能実行パターンを表す．各パターンには次の項目が付加されている．

- 頻度（$\sum_{i=1}^{t} f_{j,k,i}$）：すべてのユーザの機能実行履歴における出現頻度
- パターンの長さ（j）：パターンを構成する機能数
- ユーザごとの出現頻度（$f_{j,k,1}\cdots f_{j,k,t}$）：カッコ内は頻度に対する割合

表のヘッダ部分をクリックすることで，ユーザは，これらの項目をキーとする機能実行パターンの並べ替えを行うことができる．たとえば，他のユーザと比較して自らの実行回数が少ない，あるいは，実行したことのない機能実行パターンをユーザが知りたければ，自らの機能実行履歴における出現頻度をキーとして機

能実行パターンを昇順に並べ替えればよい．ほかにも，画面に表示させる機能実行パターンの長さの最大値や出現頻度の最低値をユーザは指定可能である．

表の列のうち，頻度，シーケンスの長さ，各ユーザの頻度のそれぞれについて昇順，降順に並べ替えることが可能である．

また，ユーザは機能実行履歴そのもの（図1参照）をテキストエディタ等で閲覧することも可能である．機能実行履歴には機能実行ごとにその「実行日」と「実行時刻」が記録されており，機能の実行時期をユーザは容易に知ることができる．

4. 評価実験

4.1 概　　要

提案方式の有効性を確認することを目的として，3章で紹介した試作システムを用いた実験を行った．実験のあらましは次のとおりである．

- 被験者は奈良先端科学技術大学院大学の教官2人，修士課程学生3人の計5人である．被験者は互いに顔見知りであり，機能実行履歴の提供者名（被験者名）が明示されるだけで，被験者はその特性を知ることができる．なお，試作システムによる利用知識の共有の効果を明確にするため，被験者はすべて試作システムを初めて利用する者とした．
- 被験者に与えられたタスクは，プレーンテキストをある形式の文書に整形することである．タスクの詳細は4.2節で述べるが，タスク中には4つのサブタスクが暗に（被験者には事前に知らせずに）設定されており，それらタスクを効率良く実行できる4つの機能の利用知識が，提案方式により獲得されるかどうか評価する．
- 被験者はMicrosoft Word 2000を用いてタスクを実行する．Microsoft Word 2000は現在広く利用されているワープロソフトの1つである．また，1章で述べたとおり，多機能ではあるが，ユーザが実際に利用している機能はその一部にすぎない典型的なアプリケーションソフトウェアである．なお，各被験者のMicrosoft Wordの使用年数，主な作成ドキュメントを**表2**に示す．
- 機能実行履歴の収集，実行パターンの抽出，および，実行パターンを含む機能実行履歴の被験者への提示は，3章で紹介した試作システムを用いて行った．
- 実験終了時に，被験者にアンケートとインタビューを行い，4つのサブタスクを効率良く遂行できる4つの機能について，既知であったか，実験中に利用知識を獲得したか，サブタスク遂行のため利用したか，を確認した．

表2　被験者のWord使用経験
Table 2　Background of subjects' MS Word.

被験者	作成ドキュメント	使用年数
S1	レポート，論文1，卒論	1年
S2	レポート，案内状	2年
S3	論文 多数	3年
S4	レポート，案内状，論文1	3年
S5	論文 多数	5年

4.2 タ　ス　ク

被験者に与えるタスクは共通で，Microsoft Word 2000を用いて，（ファイルとして格納されている）プレーンテキストをある形式の文書に整形する作業である．被験者には，プレーンテキストのハードコピー，整形済み文書（でき上がり見本）のハードコピーが渡される．整形すべき部分やその具体的内容は，でき上がり見本のハードコピーに指示として書き込まれている．文書はあるシステムの解説書の一部であり日本語で書かれている．でき上がり時のサイズはA4判13ページ，約13,000文字からなる．図は含まれていない．

整形作業は次の4つの作業（サブタスク）T1〜T4から構成されている．

T1： 目次の作成 ··· 章，および，節見出しとページ番号の一覧を作成する．

T2： ページ番号の付加 ··· すべてのページのフッタ中央にページ番号を付加する．

T3： 表の作成（3カ所）··· タブで区切られた文字列を指示されたとおり罫線で囲み表を作成する．

T4： 章，および，節見出しのフォント変更（21カ所）··· 文書全体を通じて，章と節の見出し部分のフォントを「MSゴシック」に変更する（本文のフォントは，MS明朝）．

Microsoft Word 2000は，これら4つのサブタスクT1〜T4を簡単に遂行することのできる次の4つの機能F1〜F4を提供している．ただし，F1〜F4以外の機能を使用してT1〜T4を遂行することは可能であるが，一般に，作業時間とともに実行機能数と機能実行回数が増加する．また，いわゆる文字入力作業だけでT1〜T4を遂行することも可能である．たとえば，タスクT1（目次の作成）は，機能F1（索引と目次）を使用しなくても，ユーザが章，節を列挙し，それらのページ番号を調べることにより，文字入力のみで遂行することが可能である．その場合，実行機能数と機能実行回数は減少するが，一般に，打鍵数等は大幅に増加する（T1を遂行するためには，約2000文字の入力

表 3 被験者の作業時間，実行機能数，機能実行回数
Table 3 Task completion time, number of unique and total function executions.

	作業時間（分）				実行機能数	機能実行回数
被験者	フェーズ 1	フェーズ 2	フェーズ 3	合計	合計	合計
S1	23	10	35	68	24	99
S2	24	6	52	82	21	134
S3	24	7	27	58	36	139
S4	23	9	*50	*82	21	167
S5	24	5	18	47	27	87

* サブタスク T2 未完了

が余計に必要となる）ため，作業時間は（比較的大幅に）増加することになる．

F1：「挿入」→「索引と目次」··· 文書中の見出しの一覧を，指定した場所に挿入する．

F2：「挿入」→「ページ番号」··· 文書のヘッダやフッタにページ番号を挿入する．

F3：「罫線」→「変換」→「文字列を表にする」··· タブ等で区切られた文字列を表に変換する．

F4：「書式」→「スタイル」··· 文書内の特定の文字列に対して，フォント，フォントサイズ，行間隔，文字配置等を一括して変更する．

ここで，カッコで囲んだ文字列は，これら機能に対応するメニュー項目名であり，矢印はメニュー階層を表している．なお，被験者には，これらサブタスク T1〜T4 と機能 F1〜F4 の存在は事前に知らされていない．

4.3 実 験 手 順

実験は次の 3 つのフェーズで構成されている．

フェーズ 1： タスクの具体的内容が指示された文書（でき上がり見本のハードコピー）を閲覧してもらったうえで，タスクを個別に実行してもらう．閲覧とタスク実行を合わせた作業時間は 30 分とした．なお，この 30 分という作業時間は，当該タスクの完了に必要と思われる平均的な時間の半分となるよう，予備実験の結果に基づいて定めた．なお，被験者に対しては，次のような指示が出される．

- Microsoft Word 2000 の既知の機能をできるだけ利用し，効率良くタスクを実行するように．
- 必要があればオンラインヘルプを参照しても構わない．
- 実行が困難と思われる作業は後回しにしても構わない．

また，被験者の機能実行履歴は，試作システムによりリアルタイムに収集，蓄積される．

フェーズ 2： 被験者に，試作システムによる機能実行履歴の分析結果を閲覧してもらう．分析結果には，被験者自身の機能実行パターンやその頻度とともに，他の被験者の機能実行パターンやその頻度も含まれている．それらを比較することにより，タスク実行に役に立ちそうな機能の利用知識の獲得を試みてもらう．なお，必要であれば，他の被験者の機能実行履歴そのものを被験者は閲覧することができる．閲覧時間の制約は特に設けない．

フェーズ 3： 被験者にタスク実行を再開してもらい，その終了まで作業を続けてもらう．ただし，ある被験者がタスクを完了できない，あるいは，すでにタスクを完了した被験者と比べて非常に多くの時間をタスク完了までに要する，と考えられる場合，作業継続の意思の有無を本人に確認のうえ，タスク未完了ながら作業を終了させることもありうる．

なお，必要であれば，フェーズ 2 で閲覧した機能実行履歴やその分析結果を引き続き閲覧することができる．

4.4 結 果

試作システムの機能実行履歴収集による計算機への負荷は非常に小さく，被験者のタスク実行を妨げることはなかった．

被験者ごとのフェーズ 1 とフェーズ 3 における作業時間，実行機能数，機能実行回数を**表 3** に示す．なお，機能実行履歴を被験者（ユーザ）間で共有するフェーズ 2 の所要時間は被験者ごとに異なるが，おおよそ 5 分から 10 分であった．また，4 つのサブタスクの遂行に役立つ 4 つの機能について，各被験者がどのフェーズでその利用知識を獲得し，タスク遂行に利用したかを**図 7** に示す．図 7 において，□印は提案方式による利用知識の獲得を，4 ○印はオンラインヘルプによる利用知識の獲得を，△印は機能の実行を，それぞれ表す．△印のみの機能は，被験者にとってその利用知識が既知であったことを表す．また，□印，○印，△印のいずれもない機能は，利用知識の獲得もタスク遂行のための実行もなされなかったことを表す．

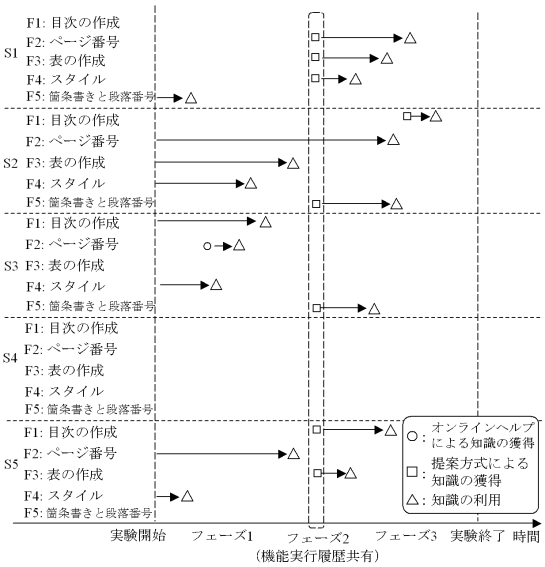

図 7 各被験者の利用知識の獲得時期
Fig. 7 Timimg of knowledge acquisition.

- 作業時間は最長が 82 分，最短が 47 分であった．ただし，被験者 S4 はサブタスク T2（ページ番号の付加）を正しく終えることができなかった．なお，機能実行履歴を閲覧するフェーズ 2 の所要時間は，5〜10 分であり，オンラインヘルプの一般的な参照時間等と同程度であった．
- 実験中に利用知識が獲得された機能はのべ 9 個で，そのうち，他の被験者の機能実行履歴を参照することで利用知識が獲得された機能はのべ 8 個，既存のオンラインヘルプ機能で利用知識が獲得された機能は 1 個のみであった．なお，試作システムを用いて他の被験者の機能実行履歴を参照することで利用知識が獲得された機能には，あらかじめ想定した機能 F1〜F4 以外の機能 F5（箇条書きと段落番号）が含まれている（被験者 S2 と S3 が利用知識を獲得した）．F5 を利用するとサブタスク T4 の遂行に必要な打鍵数は減少する．F4 を利用する方が，より簡単に T4 を遂行できるが，F1〜F4 と同様に「使用すると T1〜T4 を簡単に遂行できる」機能として，利用知識の獲得対象と見なした．

表 3 と図 7 の情報，さらに，実験後に被験者に対して行ったアンケートとインタビューの結果に基づいて，個々の被験者の特徴を簡単にまとめると次のようになる．

被験者 S1：5 人の被験者の中で最も初心者であり，4 つの機能 F1〜F4 いずれの利用知識も持っていなかった．しかし，フェーズ 2 において他の被験者の機能実行履歴を参照することで，3 つの機能 F2，F3，F4 の利用知識を獲得し，フェーズ 3 においてサブタスク T2，T3，T4 遂行のためにそれらを実行している．なお，機能 F2 が存在することを被験者 S1 は想像していたが，その具体的な実行方法は既知ではなかった．また，機能と F3，F4 については，その存在を想像さえしていなかった．

被験者 S2：3 つの機能 F2，F3，F4 の利用知識を持っていた．機能 F1 についてはその存在は想像していたが，他の被験者の機能実行履歴を参照することで，フェーズ 3 において具体的な実行方法（利用知識）を獲得している．

被験者 S3：2 つの機能 F1，F4 の利用知識を持っていた．機能 F2 については，フェーズ 1 においてオンラインヘルプを参照することで，その利用知識を獲得し，サブタスク T2 遂行のために実行している．また，あらかじめ想定した 4 つの機能 F1〜F4 とは別の「箇条書きと段落番号」機能の利用知識を，フェーズ 2 において他の被験者の機能実行履歴を参照することで獲得している．ただし，機能 F3 については，オンラインヘルプによっても，また，フェーズ 2 において他の被験者の機能実行履歴を参照することによっても，その利用知識は獲得できていない．

被験者 S4： 旧バージョンを含め，当該ソフトウェアの使用年数が 3 年と比較的長いにもかかわらず，初心者の S1 同様，4 つの機能 F1〜F4 いずれの利用知識も持っていなかった．オンラインヘルプによっても，また，フェーズ 2 において他の被験者の機能実行履歴を参照することによっても，それらの利用知識を獲得することはなく，実行も行っていない．

被験者 S5：旧バージョンを含め，当該ソフトウェアの使用年数が 5 年と 5 人の被験者の中で最も長いが，2 つの機能 F2，F4 の利用知識しか持っていなかった．ただし，フェーズ 2 において他の被験者の機能実行履歴を参照することで，残りの 2 つの機能 F1，F3 の利用知識を獲得し，サブタスク T1，T3 遂行のためにフェーズ 3 でそれらを実行している．なお，被験者 S5 にとって，機能 F1 は具体的な実行方法が既知でなかっただけであるが，機能 F3 はその存在を想像すらしていなかった．4 つの機能 F1〜F4 をすべて利用してサブタスクを実施した唯一の被験者である．

5. 考　　察

4 章で述べた評価実験は，提案方式の一部を対象としたもので，被験者数も少なく，その結果を一般化す

ることは適当でないかもしれない．しかし，得られた結果のいくつかは，ソフトウェア利用知識の共有，および，共有による作業効率の向上に提案方式が役立つ可能性のあることを示唆していると考える．たとえば，次の点である．

- 初心者（被験者 S1）であっても，利用知識の獲得が可能であった．しかも，知識の獲得により，その後の作業時間は短くなり，機能実行回数は熟練者（被験者 S5）と同程度まで少なくなった．提案方式は，初心者の作業効率の向上に寄与する可能性がある．
- ユーザが利用知識を獲得したのべ 8 個の機能のうち，ユーザがその存在を想像さえしていなかった機能がのべ 3 個もあった．そのうちの 1 つは，熟練者（被験者 S5）によって獲得されたものである．提案方式が，初心者だけでなく熟練者に対しても有効である可能性がある．
- 提案方式の有効性をより明確に評価するために，（被験者には知らされていなかったが）タスクには 4 つのサブタスクとそれらを簡単に行うための 4 つの機能があらかじめ設定（想定）されていた．しかし，それら 4 つの機能以外の機能についても，その利用知識を獲得した被験者が 2 人いた．このことは，タスク中にあらかじめ設定されていた機能のみが，提案方式による利用知識の獲得に特に適していたわけではないことを示している．また，提案方式を実際に利用する場合にあてはめてみると，ユーザに利用知識を獲得させる機能をあらかじめ明らかにしタスクを設定する，という必要が必ずしもないことになる．提案方式は，適用範囲が広く，実施コストも比較的小さい可能性がある．
- 機能実行履歴には，実行されたメニュー項目名だけでなく，メニュー階層内での位置が分かるよう，トップメニューから実行されたメニュー項目名までの選択経路が付加されている．タスク完了後の被験者へのインタビューでは，こうした情報が，機能の試用やオンラインヘルプの利用において非常に役立った，との感想を得た．すなわち，履歴中のある機能を試用したければ，ユーザは示された選択経路に従って実際にメニュー項目を選択するだけでよく，オンラインヘルプ上での検索では，表示されたメニュー項目名が非常に適切なキーワードとなった．オンラインヘルプ等の既存のユーザ支援機構と提案方式をうまく連携させることで，利用知識の獲得コストを大幅に小さくできる可能性がある．

一方，5 人の被験者のうち 1 人（被験者 S4）は，提案方式による利用知識の獲得は見られなかった．提案手法の有効性や適用範囲に制限のある可能性もある．利用知識が獲得されなかった主な原因と考えられる対策は次のとおりである．

- 被験者 S4 は，4 つのサブタスクのうち T3（表の作成）の遂行に作業時間のほとんどを費やし，他のサブタスクは満足に終えることもできなかった．与えられたタスクがユーザにとって負荷の高いものである場合，新たな機能の利用知識を獲得するよい動機付けになる反面，獲得のための時間的，精神的余裕が生まれない可能性もある．ソフトウェアに対するユーザの熟練度，機能面での興味，等を勘案し，ユーザにとって無理のないタスク設定が必要と考えられる．
- 被験者 S4 にとって，試作システムが提示する機能実行履歴は膨大なものであった．試作システムに使い慣れていないこともあり，5〜10 分という比較的短い閲覧時間では，利用知識を獲得するまでには至らなかった．同一タスク方式による知識獲得であっても，タスク内容やユーザ数によって，機能実行履歴は非常に大きくなる可能性がある．膨大な機能実行履歴をコンパクトに提示する方式，機能実行履歴のサイズに応じてその提示形態を変更する方式，ユーザの興味に合わせて提示内容を変更する（フィルタリングする）方式等についてより詳細な議論が必要と考えられる．

6. 関 連 研 究

ソフトウェアの利用知識を対象としたものではないが，個人の暗黙知を有形化し，グループで共有するシステムとしては Advice/Help on Demand が知られている[8]．業務ノウハウを文書化しグループ内での参照を可能にしている．ただし，業務ノウハウが自然語で記述されているため，本提案方式のように，ユーザの本来の作業を妨げずに，収集，分析することは容易ではない．

本提案手法と同様に，ソフトウェアユーザの行動やソフトウェアを利用した業務内容を記述する手段として，ソフトウェアの実行（操作）履歴を用いる研究は数多く報告されている[2),4)]．たとえば，森らは，X-window におけるマウス操作や打鍵等の詳細な操作履歴を収集，分析するツールを提案している[7)]．彼らのシステムを用いれば，操作履歴に基づいて，ユーザ操作の再現，正規表現に似た形式での操作履歴の分析が可能である．また，安部田は，個人情報管理システム

（PIM）の操作履歴を収集し，企業における業務知識を体系化するシステムを提案している[1)]．

ソフトウェアの実行（操作）履歴をネットワーク経由で収集するシステムはいくつか提案されている．Hilbert らは，ソフトウェア設計者の意図する操作系列と異なる操作をユーザが行うと，その詳細を電子メイルで設計者に通知するシステムを開発している[5)]．彼らのシステムを用いれば，ソフトウェア操作に対する設計者とユーザの認識のずれを知ることができ，ソフトウェア設計に役立てることができる．ただし，ユーザの操作と比較するための操作系列を設計者があらかじめシステムに登録しておく必要がある．本提案方式のように，実際の機能実行履歴の中から利用知識を抽出することはできず，ユーザ間での利用知識の共有を支援する機能も有していない．

7. おわりに

本稿では，ソフトウェア，特にアプリケーションソフトウェアが提供する機能の利用知識の獲得コストを小さくすることを目的として，ユーザ間で機能実行履歴を共有する方式を提案し，提案に基づく試作システムと評価実験の結果について述べた．評価実験では，被験者 5 人のうち 4 人において利用知識の獲得が認められ，初心者については，知識獲得により，作業効率の向上も確認された．また，熟練者でさえその存在を想像さえしていなかった機能についても利用知識の獲得がなされた．今回の評価実験では，タスクを 1 回行う間での知識獲得に注目したが，類似のタスクを複数回行って知識が獲得されていることを示す方法も考えられる．

現在の試作システムでは，機能実行パターンを「機能実行履歴中の任意の部分系列」とし，「パターン」を切り出すための区切り子等は特に設定していない．特定の機能を区切り子としたり，機能実行間隔に基づいて区切ったりすることで，ユーザの行動やメンタルモデル等により対応したパターン抽出が可能になるかもしれない．機能実行履歴の蓄積と分析を進め，「機能実行パターンの切り出し基準」についての議論を行う必要がある．また，機能実行履歴やその提供者の特性を獲得し，機能実行履歴参照者に提供する具体的方法についても検討の必要がある．特に，機能実行履歴を利用することで，提供者の特性に関するより多面的で詳細な情報を提供できる可能性がある．たとえば，機能実行履歴提供者の「対象ソフトウェアの使用経験」は，参照者がその機能実行履歴を参照するかどうか判断するうえで，重要な特性の 1 つと考えられる．履歴提供者自身に「使用年数」等を申告させるのが一般的な獲得方法であるが，機能実行履歴を利用すれば，「累積使用時間」はもちろんのこと，「出現頻度の高い機能実行パターン数」，「機能ごとの使用頻度（プロファイル）」といった指標を自動的に得ることも可能である．ここで，「出現頻度の高い機能実行パターン数」を「効率よく作業を行うために定型化された操作実行の数」と見なすことができれば，その数が多いほど「対象ソフトウェアに対する習熟度が高く」参照すべき機能実行履歴の提供者である，と考えることができる．また，「機能ごとの使用頻度（プロファイル）」は，「対象ソフトウェアの使用経験」をより詳細に表すものであり，対象ソフトウェアに対する習熟度，使用目的等を提供者と参照者との間で比較可能にすると考えられる．評価実験では明確にできなかったが，機能実行履歴を共有することで，他のユーザ（機能実行履歴提供者）と同様の操作ミスをユーザ（機能実行履歴参照者）がおかしてしまう可能性がある．また，ユーザが誤った利用知識を獲得する可能性もある．ただし，操作ミスはいわゆる「undo 機構」により容易に取り消すことができ，オンラインヘルプを活用することで知識獲得上の誤りは少なくすることができる．既存のユーザ支援機構と提案方式の連携について，さらなる議論が必要である．

提案方式は，3 つの機能実行履歴収集方式，および，2 つの利用知識獲得方式を含み，利用知識を共有する目的や状況によってそれらを使い分けることを前提としている．本稿で述べた評価実験では，それら組合せのうちの 1 つを試作システム上で実施したにすぎない．特に，知識の獲得方式の 1 つである「類似度推定方式」については，提案方式の適用範囲を広げる意味においても，より具体的な方式の検討と実験的評価が今後の課題である．

謝辞 実験に協力いただいた被験者の方々に感謝する．なお，本研究の一部は，新エネルギー・産業技術総合開発機構新規産業創造型提案公募事業，および，情報処理振興事業協会高度情報化支援ソフトウェアシーズ育成事業の援助によるものである．

参考文献

1) 安部田章：ユーザの操作履歴から業務知識を抽出する個人情報管理システム，情報処理学会技術報告，GW23-2, pp.7–12 (1997).
2) 赤池英夫，角田博保：X-Window 上の利用者行動分析システム，情報処理学会論文誌，Vol.33, No.5, pp.736–745 (1992).
3) Carroll, J.M. and Rosson, M.B.: Paradox of

the Active User, *Interfacing Thought: Cognitive Aspects of Human-computer interaction*, MIT Press (1987).
4) 海老名毅, 伊藤 昭：Xアプリケーションにおける可読性のある操作履歴の取得について，電子情報通信学会技術報告，HC94-87, pp.1–7 (1995).
5) Hilbert, D.M. and Redmiles, D.F.: An approach to large-scale collection of application usage data over the Internet, *Proc. 20th Intl. Conf. on Software Engineering*, pp.136–145 (1998).
6) 松本健一，森崎修司：フィールドテストにおけるソフトウェア操作履歴の収集と分析の支援，振興事業協会平成11年度高度情報化支援ソフトウェアシーズ育成事業報告書 (2000). http://tori.aist-nara.ac.jp/usageLog/
7) 森 孝弘，西田知博，斎藤明紀，都倉信樹：大量の GUI 操作履歴を分析するための走査・再生ツール，情報処理学会研究報告，HI-69-1, pp.1–8 (1996).
8) 中山康子，真鍋俊彦，竹林洋一：知識情報共有システム（Advice/Help on Demand）の開発と実践:知識ベースとノウハウベースの構築，情報処理学会論文誌，Vol.39, No.5, pp.1186–1191 (1998).
9) Reisberg, D.: Cognition, W.W. Norton & Company, New York, NY (1997).
10) 高田光男，小高知宏，小倉久和：UNIX シェルユーザのための適応型ユーザインタフェースの構築とその評価，第 50 回情報処理学会全国体会論文集（4），pp.331–332 (1995).
11) 高田光男，西野順二，小高知宏，小倉久和：UNIX 高機能シェルの行編集機能に対する適応型ヒューマンインタフェースの構築とその評価，情報処理学会論文誌，Vol.38, No.10, pp.1919–1927 (1997).
12) Torii, K., Matsumoto, K., Nakakoji, K., Takada, Y., Takada, S. and Shima, K.: Ginger2: An environment for CAESE (Computer-Aided Empirical Software Engineering), *IEEE Trans. Software Engineering*, Vol.25, No.4, pp.474–492 (1999).

(平成 12 年 3 月 21 日受付)
(平成 12 年 9 月 7 日採録)

森崎 修司

平成 8 年佐賀大学理工学部情報科学科卒業．平成 10 年奈良先端科学技術大学院大学博士前期課程修了．現在，同大学博士後期課程に在学中．分散オブジェクト，協調フィルタリングに興味を持つ．電子情報通信学会学生会員．

門田 暁人（正会員）

平成 6 年名古屋大学工学部電気学科卒業．平成 10 年奈良先端科学技術大学院大学博士後期課程修了．同年奈良先端科学技術大学院大学助手．博士（工学）．ソフトウェアメトリクスの研究に従事．日本ソフトウェア科学会会員．

松本 健一（正会員）

昭和 60 年大阪大学基礎工学部情報工学科卒業．平成元年同大学大学院博士課程中退．同年同大学基礎工学部情報工学科助手．平成 5 年奈良先端科学技術大学院大学助教授．工学博士．Computer-Aided Empirical Software Engineering（CAESE）環境，ソフトウェアメトリクス，ソフトウェアプロセス，視線インタフェースに関する研究に従事．電子情報通信学会，IEEE，ACM 各会員．

井上 克郎（正会員）

昭和 54 年大阪大学基礎工学部情報工学科卒業．昭和 59 年同大学大学院博士課程修了．同年同大学基礎工学部助手．昭和 59〜61 年ハワイ大学マノア校情報工学科助教授．平成元年大阪大学基礎工学部情報工学科講師．平成 3 年同学科助教授．平成 7 年同学科教授．工学博士．ソフトウェア工学の研究に従事．電子情報通信学会，日本ソフトウェア科学会，IEEE，ACM 各会員．

鳥居 宏次（正会員）

昭和 37 年大阪大学工学部通信工学科卒業．昭和 42 年同大学大学院博士課程修了．同年電気試験所（現電子技術総合研究所）入所．昭和 59 年大阪大学基礎工学部情報工学科教授．平成 4 年奈良先端科学技術大学院大学教授．現職は同大学副学長．工学博士．専門はソフトウェア工学．IEEE，ACM および情報処理学会の各フェロウ．