

## 手書きユーザインタフェースの評価を支援する環境の実現

二村尚, 中島一彰, 早川栄一, 並木美太郎, 高橋延匡

東京農工大学工学部電子情報工学科

本稿では、ウィンドウシステム“未”用APのユーザテストによるUI評価を支援する環境について述べる。“未”用APは、手書きユーザインタフェース(UI)によって紙にペンで記入する感覚で操作を行なうことが可能である。このUIの特徴を検証するためには、ユーザテストを実施し定量的なデータを収集して分析する必要がある。そこで筆者らはユーザテストによるUI評価を支援するために、ユーザの入力履歴と入力に対するAPの動作履歴、およびシステムとAP間の情報を収集する機構を実現した。今回は“未”用APである分散手書きKJ法を評価対象として、設定した評価項目に沿ってデータを収集し、データ分析ツールによってユーザの入力履歴を分析した。またAPの動作履歴や、システムとAP間のデータを用いることにより、APや評価方法に特化した分析が可能になることを明らかにした。

### **The Implementation of an Environment to Support the Evaluation of Handwriting User Interfaces**

Takashi NIMURA, Kazuaki NAKAJIMA, Eiichi HAYAKAWA  
Mitarou NAMIKI, and Nobumasa TAKAHASHI

**Department of Computer Science, Faculty of Technology,  
Tokyo University of Agriculture and Technology**

This paper describes an environment for evaluating the user interfaces (UI) of application programs (AP) on the HITSUJI window system with user-operated tests. Users of AP on The HITSUJI window system can operate as if writing on papers with a pen by the handwriting UI. The investigation of the characteristics of such UI needs to record and analysis quantitative data. A system which records a history of the user's input and the AP's reactions to the user's input is implemented for the purpose of supporting UI evaluation. In this paper, a user's input history of DISTRIBUTED HANDWRITING KJ-METHOD is analyzed with a tool. Finally, it was find that analysis is possible by using AP's action history or data among the system and the AP.

## 1. はじめに

現在のユーザインタフェース（以下 UI）の設計は、作業効率や操作の分りやすさなどを重要な方針として行なわれている。そのため、アプリケーションプログラム（以下 AP）の設計者は、UI の定性的な特徴と、その特徴を裏付ける定量的な評価結果を示す必要がある。

UI を定量的に評価する方法として、ユーザに AP を操作させてユーザの行動を調べるユーザテストが考えられる。このユーザテストによる評価において、どのようにユーザの行動を定量的データとして把握するのが問題となる。

ユーザの行動を保存して評価する研究としては、[1][2]などがある。これらの研究においては、キーボードもしくはマウスによる入力を対象としているので、入力デバイスによる入力履歴やウィンドウシステムが発行する情報など AP に手を加えずに保存できるデータを分析することによってユーザの行動を調査することができる。

しかしペン入力においては、認識系を用いるので再現性がない。そのため、ユーザの入力履歴から AP の動作を判断したり入力を再現することが難しい。この問題を解決するためには、システムや AP に手を加えることによって、より多くのデータを保存しなくてはならない。

そこで我々は、ユーザテストによる 手書き UI の評価支援を目的とした環境（UI 評価環境）を設計、実現した。

この UI 評価環境の特徴は、定量的な調査の対象であるユーザの操作履歴とユーザの入力に対する AP の動作履歴をユーザテスト中に自動的に保存できる点である。

本稿では UI 評価環境の設計、実現、システム評価について述べる。

## 2. ペンを用いた手書き UI とその評価

### 2.1 手書き UI の概要

筆者らが所属する東京農工大学工学部電子情報工学科高橋・並木研究室では、手書き UI の開発基盤としてウィンドウシステム“未”が実現された[3]。“未”では、表示一体型タブレットを使用することにより、紙にペンで記入する感覚で AP を操作できるので、ペンを利用した使いやすい UI の実現が可能となった。

このペンによる手書き入力を用いた“未”用 AP の一つとして、「分散手書き KJ 法」が実現された[2]。この AP は UI の面では次に挙げる特徴がある。

- ・ 図形、文字、記号などがペンだけで入力できる
- ・ AP の操作をジェスチャによって入力できる

ジェスチャ入力とは、作業ウィンドウ内に操作を示す記号を書き込み、その記号の種類を認識することによって AP の機能を実行するペン特有の入力方法である（図 1）。これにより分散手書き KJ 法では、作業中に入力デバイスの持替えや、メニューによる入力モードがなく、作業中の思考が妨げられることがない。

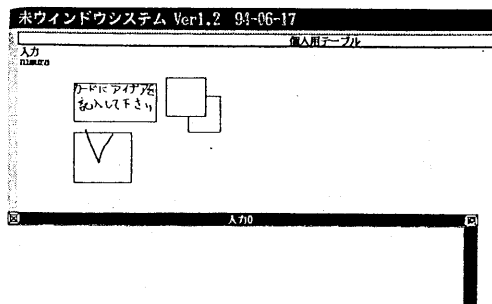


図 1 分散手書き KJ 法のジェスチャ入力（カード選択）

### 2.2 ユーザテストによる UI 評価

2.1 で述べた分散手書き KJ 法の UI の特徴を検証するためには、ユーザテストを行なってユーザの行動を調査し、UI の改良前後や異なる方式の UI を比較するといった評価が必要である。

ユーザの行動の調査項目を、分散手書き KJ 法の UI 調査を例に次に示す。

- ・ ジェスチャの種類別の認識率
- ・ ジェスチャ入力にかかる時間
- ・ ジェスチャの誤認識により余計に要する時間
- ・ キーボードとマウスで操作した場合のデバイスの持ち替えにかかる時間

このような調査を行なうためには、入力した時間や入力した場所などのユーザによる操作履歴を保存し、データを分析する必要がある。例えばカード選択のジェスチャ入力にかかる時間は、ユーザがカードを選択するために入力したペン入力の開始時刻と終了時刻を調べてその差分を計算する。

このように作業内容を判断して評価する場合、ユーザの操作履歴に加えて、ユーザの入力に対して AP がどのように動作したのかを示すデータが必要である。特にペン入力の場合は、入力したタブレットの座標列（筆点列とする）を認識系によ

って加工するために、筆点列や入力時刻などの入力データと、その入力の認識結果を保存する必要がある。

### 3. “未”用 AP の UI 評価環境の設計方針

#### 3.1 目的

2.2 で述べたように、ユーザテストによって収集したユーザの操作履歴とユーザの入力による AP の反応の履歴 (AP の動作履歴) を分析することにより、定量的なデータによって検証した結果として UI の特徴を主張できる。

この評価に必要なデータの記録を、実行画面のビデオ撮影などを利用して人手によって行なうと、ペンやマウスによる入力座標や、ウィンドウシステムによるジェスチャの認識結果など保存できないデータがある。さらにペンによる文字入力など、数 10 ms 単位で発生する入力時刻を評価者が記録することは現実的ではない。

そこで、ユーザの入力を分析するための定量的なデータと AP の動作履歴を自動的に収集することを目的として UI 評価環境を設計する。

#### 3.2 設計方針

3.1 の設計目的から UI 評価環境の設計方針を次に述べる。

##### (1) ユーザの入力履歴を保存する

ユーザが AP に対して入力デバイスを用いて行なった操作の履歴は、ユーザの行動を定量的に示すデータとして必要である。

##### (2) AP の動作履歴を保存する

入力に対する AP の動作履歴を保存し、ユーザが何をするためにその入力を行なったのかをユーザテスト後に判断できるようにする。

##### (3) データ保存のオーバーヘッドを小さくする

データを保存するためのオーバーヘッドは、テスト中の被験者の操作に影響を与えてしまう可能性がある。

##### (4) AP のソースコードの変更を最小限に抑える

データを保存する機構を実装するための AP のソースコードの変更量が多くなると、記述ミスによるバグが発生したり、評価者の手間が増加するといった問題がある。

##### (5) 保存するデータのサイズを抑える

ユーザテストによって収集するデータのサイズが大きくなると、データを格納するためのオーバーヘッドが大きくなる。特にペン入力の座標データのサイズは、その他の入力データと比較して大きいので、必要なデータを収集したうえで、そのサイズを抑える必要がある。

### 4. UI 評価環境の設計

#### 4.1 UI 評価環境によって保存するデータ

本節では、ユーザテストによって保存するデータの内容について述べる。

##### (1) ユーザの入力履歴

ユーザの入力履歴として必要なデータは、ユーザがどのような入力を、いつ、どこに行なったのかである。

ペン入力の場合、ペンの側面に付属されているスイッチの状態、ウィンドウの名称、ウィンドウの座標列  $(X_0, Y_0), \dots, (X_n, Y_n)$ 、入力開始された時刻と終了した時刻というデータを保存する。さらに、ペンによるジェスチャ入力やキーボードによる仮名漢字変換など、“未”内で入力データを加工する場合は、その結果も保存する。

##### (2) AP の動作履歴

AP の動作履歴として必要なデータは、入力に対して実行された AP の機能や AP の反応などである。これにより入力データが何をするためのものであったのかを判断することができる。

同じ入力でも AP の状態によって AP が異なる反応を示す場合があるので、入力に対して AP がどのような状態になり、どの機能が実行され、どのような反応を示したのかを保存する。

保存する具体的な内容は AP や評価方法に合わせて評価者が設定する。

#### 4.2 UI 評価環境の全体構成

UI 評価環境では、4.1 で示したデータの保存を自動的に行なう機構を提供する。この機構は、評価対象 AP の動作履歴を出力する関数を挿入してコンパイルし、評価データをファイルに保存する機構 (データロガー) を AP にリンクして使用する (図 2)。

この変更を加えた AP を使用してユーザテストを行ない、データをファイルに保存する (図 3)。

保存したデータは、テスト後にデータを分析するツールによって評価者が必要な計算や集計などの加工を行ない、評価者に提示する (図 4)。

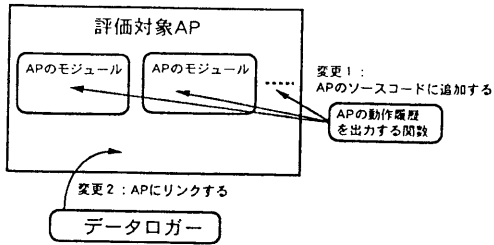


図2 評価対象 AP の変更

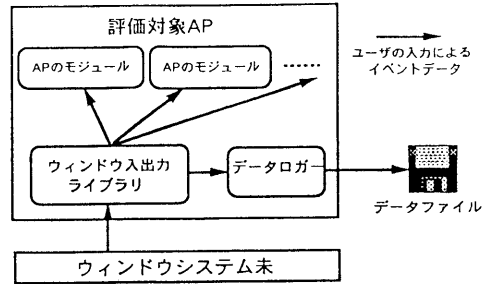


図5 ユーザの入力履歴を保存する機構

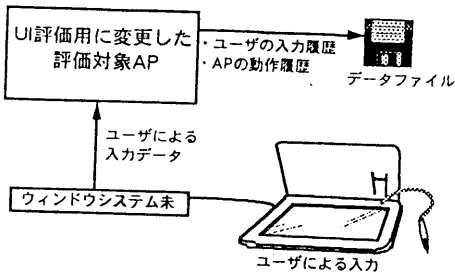


図3 ユーザテストによる UI評価データの保存

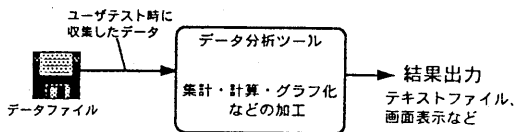


図4 保存したデータの分析

#### 4.3 ユーザの入力履歴を保存する機構

本節ではユーザの入力履歴を保存する機構について述べる。

データロガーでは“未”から AP に発行される入力データ（イベントデータ）の履歴をユーザの入力履歴として保存する。その理由は、イベントデータはユーザの入力を AP に伝えるデータであり、4.1(1) で述べた内容を含んでいるからである。この機構の構成を図5に示す。

イベントデータは“未”用 AP のウィンドウ入出力ライブラリを UI 評価用に改造して保存する。これにより、AP のソースコード自体に変更を加えずにイベントデータが保存できる。

しかし、イベントデータのメンバの中には、ID など数値で表現されるものもある。例えば、ウィンドウを一意に決定するウィンドウ ID などである。このような ID はシステムが動的に発行す

る値なので、ID がどのウィンドウを示すのかをテスト後に判断することはできない。このようなデータを、内容を示す文字列に変換して保存すると、変換のためのオーバヘッドや、ID と比較してデータサイズが大きくなるといった問題がある。

そこで、“未”用 AP のウィンドウ入出力ライブラリを改造して、ID が何を示すのかをテスト後に判断するためのデータを保存する。例えばウィンドウ ID の場合、AP がウィンドウの生成をシステムに要求するときにウィンドウの名称（文字列）を渡し、システムはウィンドウの名称に対する ID を AP に返すので、このウィンドウの名称と ID との対応を保存することによって ID を名称に変換することができる。

#### 4.4 AP の動作履歴を保存する機構

本節では、AP の動作履歴をファイルに保存する機構について述べる。

AP の動作履歴は、AP のソースコードに任意の文字列をデータロガーに出力する関数を挿入することによって保存する。ソース変更の具体例を図6に示す。

このような変更を加えることによって、AP の動作履歴を保存する（図7）。

ソースの変更は、ソースの読解や再コンパイルなどの手間が必要となってしまう。しかし、出力する文字列の内容や、どの部分に関数を挿入するべきかは、AP や調査項目に依存するために、この作業を自動化することはできない。したがって、評価者が評価内容に合わせて関数の挿入位置や出力する文字列の内容を設定する必要がある。

データロガーでは、受け取った文字列とともに、関数が実行された時刻を保存する。その理由は、AP が反応してから次にユーザが入力を行なうまでの時間を調べる場合や、逆にユーザが入力を行なってから AP が反応を示すまでの時間を調べる場合などに利用するためである。

```

/*ソースの一部(ジェスチャ入力処理)*/
switch(ジェスチャの種類){
case カード選択:
if(注目点にカードがある){
注目点のカードを拡大する();
APの動作履歴("カード選択");
}
else
APの動作履歴("カード選択失敗
(カードがない)");
}

```

図6 UI 評価用関数の追加 (下線部)

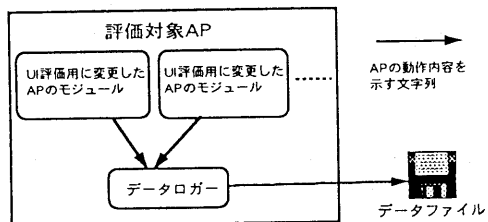


図7 ソースの変更による AP の動作履歴の保存

#### 4.5 AP の実行環境を示すデータの保存

ユーザテストでは、複数のユーザによるテストを実施しその統計をとって結果を示すことが考えられる。また、AP のバージョン、使用する入力デバイス、ユーザの経験など、異なる環境 (対象) でテストを行ない、結果としてどのような違いがあるのかを調べる場面も想定できる。したがって、ファイルに保存したデータが、どのような環境 (対象) において保存したものであるのかを示すデータとして使用者の氏名、データを保存した日時、AP 名をファイルに保存する。

#### 4.6 収集したデータの格納方法

データロガーは、ペンの応答性を落さないために UI 評価用データとして渡されたデータをバッファに格納し、バッファがオーバーフローした時点でデータセーブサーバがデータをファイルに保存する。

#### 4.7 データ分析ツール

データロガーによって保存したデータを分析す

るツールの設計について述べる。データ分析ツールは、AP や保存したデータの内容によって異なる計算、集計、結果出力を行なう必要があるため、評価内容に特化したツールとして作成する。

この全体構成を図8に示す。

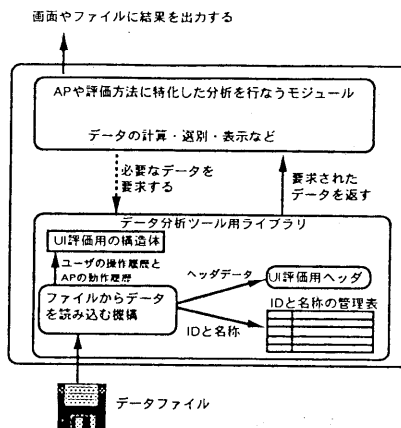


図8 データ分析ツール

データを加工する部分では、必要なデータを格納するための配列のポインタを UI 分析用ライブラリに渡す。ライブラリでは要求されたデータをファイルからロードするか、すでにロードして格納してある表を参照し、指定された配列にデータをコピーする。

次に図8の各機構について述べる。

##### (1) データ分析ツール用ライブラリ

保存したデータは、評価方法に特化したツールによって分析を行なう。しかし、異なる評価をおこなうたびにツールをはじめから作成するのは非常に手間がかかる。そこで、ツールの共通部分をライブラリとして評価者に提供し、ツールの作成を支援する。

データ分析用のライブラリとしては、保存したデータをファイルから読み込んで、UI 評価用に設定した型の構造体に格納する機構と、格納したデータを参照するための機構を提供する。

##### (2) 評価方法に特化した分析をする機構

ライブラリを利用してロードしたデータは、AP や評価方法に合わせて分析する。その際には、AP の動作履歴や、ウィンドウの名称など、AP に固有のデータを用いることにより、AP や評価方

法に合わせた結果の出力が可能となる。

このようなデータの利用方法を分散手書き KJ 法の UI 評価を例に示す。

#### 例1 カード選択からカード記入にかかる時間

ユーザの入力に対する AP の動作履歴が「カード選択」→「カード記入」と連続している場合、その入力履歴を参照して「カード選択」の入力が完了してから、「カード記入」の入力が開始されるまでの時間を計算する。

#### 例2 カード移動の距離

AP の動作履歴が「カード移動」の場合、その入力履歴を参照してイベントデータの筆点列の長さを計算する。

このように保存したデータを分析し、入力時間や入力座標などを評価方法にしたがって出力する AP 専用のツールを作成する。

### 5. “未”用 AP の UI 評価環境の実現

#### 5.1 UI 評価環境を実現した環境

UI 評価環境を実現した環境を次に示す。この環境は分散手書き KJ 法が実現されているものと同じである。

計算機 : ワークステーション 2050/32E  
(プロセッサ MC68030 25MHz)  
OS : OS/omicon V3[5]  
ウィンドウ  
システム : ペンウィンドウシステム “未”

#### 5.2 UI 評価環境を用いたユーザテスト

実現した環境を用いて、分散手書き KJ 法のユーザテストを行ない、データを収集、分析した。その方法と調査結果について次に述べる。

##### (1)ペンジェスチャの評価

参考文献[4]において、ジェスチャの入力時間とジェスチャの認識率を調査するために、UI 評価環境を用いてテストを行なった。テスト方法は 5 名の被験者に 10 回ずつジェスチャ入力をさせて、データを保存した。入力させるジェスチャの種類を決めて、異なる種類の認識が行なわれた場合を誤認識として集計した。結果を表 1 に示す。

表 1 の考察は、参考文献[4]に詳細が述べられている。

表 1 ジェスチャの評価

| 種類 | 時間<br>(秒) | 認識率(%) |    |   |
|----|-----------|--------|----|---|
|    |           | 正      | 誤  | 不 |
| 作成 | 1.41      | 92     | 8  | 0 |
| 選択 | 0.66      | 77     | 18 | 5 |
| 移動 | 0.48      | 97     | 0  | 3 |
| 困み | 1.48      | 99     | 0  | 1 |
| バツ | 1.69      | 91     | 6  | 3 |

#### (2)作業時間と作業間隔の調査

ユーザテスト用に設定した分散手書き KJ 法のテスト用タスクを被験者に入力させて、収集したデータを作業単位で分割し、作業時間と作業間隔を調査した。AP が設定したタスクから外れる動作をした場合、ジェスチャの誤認識が発生したものと考える。この調査目的の一つは、誤認識が発生した場合の時間的影響を調べることである。

この調査のために収集したデータの一部を図 9 に示す。図 9 では、実行された機能の名前と、作業時間、作業間隔を、収集したデータから抜き出して出力している。

```

時間/ 3583ms
機能:カード作成(無始入力)/入力時間:1120ms/ウィンドウ名:共有テーブル/
/作業間隔/ 3005ms
/機能:カード作成/入力時間:1120ms/ウィンドウ名:個人用テーブル/
/作業間隔/ 2539ms
/機能:カード選択/入力時間:336ms/ウィンドウ名:個人用テーブル/
/作業間隔/ 2637ms
/機能:カード記入/入力時間:4304ms/ウィンドウ名:入力0/
/作業間隔/ 1438ms
/機能:カード縮小/入力時間: 0ms/ウィンドウ名:入力0/
/作業間隔/ 1418ms
/機能:グルーピング/入力時間:1120ms/ウィンドウ名:個人用テーブル/
/作業間隔/ 1418ms

```

図 9 データ分析ツールによる出力例

集計した結果の一例を示す。ジェスチャの誤認識は、1 回発生すると、約 5.7 秒作業時間が長くなる。その時間の 82% (約 4.7 秒) が、誤認識が発生してから正しい入力を開始するまでの時間である。

#### 5.3 データロガーの実現

5.2 の評価で使用されたデータロガーを、次に示す手順で実現した。

- ①ウィンドウ入出力ライブラリの改造
- ②AP の動作履歴を出力するための変更
- ③ライブラリおよび AP の再コンパイル

①は AP に依存しないので、次に評価を行なう場合にこの作業を行なう必要はない。

②の作業によって保存できるようになったデータを次に示す。

- ・ジェスチャの認識結果
- ・実行された AP の機能

#### 5.4 データ分析ツールの実現

収集したデータを集計するために、必要なデータを選別、計算し、テキストファイルに出力するツールを作成した。

#### 5.5 “未”の改造

ユーザの入力を分析するために、ユーザが入力を行なった時刻の保存が必要である。しかしこれまでの“未”のバージョンでは、イベントデータに入力のタイムスタンプが含まれていなかった。この時刻を AP 層で測定すると、システムや認識系によるオーバーヘッドによって、ユーザが入力を行なった時刻とずれが生じる。そこで、“未”のイベント処理を行なう機構を改造してタイムスタンプが得られるようにした。

### 6. システム評価

本章では実現した UI 評価環境のシステム評価について述べる。

#### 6.1 データロガーの評価

##### (1) データロガーによる AP の変更

AP の動作履歴を保存する関数を、分散手書き KJ 法のソースコード (C 言語で約 11,000 行) に 48 行挿入することにより、AP の動作履歴を保存した。この作業は分散手書き KJ 法の設計者が行ない、約 1 時間かかった。

##### (2) データロガーの AP への実装

データロガーのソースコードは、C 言語で約 800 行である。このモジュールを“未”用 AP のウィンドウ入出力ライブラリに 6 行追加することによって組み込んだ。この UI 評価用に改造したウィンドウ入出力ライブラリを AP にリンクすることによってデータの保存を可能にした。

##### (3) データロガーによるオーバーヘッド

UI 評価用のデータをバッファに格納するためのオーバーヘッドは、最大 64 ms であるが、これはペンの筆点の数が、システムの規定する最大の 1024 点入力された場合であり、ほとんどの場合 32 ms 以内にデータをバッファに保存することができた。定性的には、データロガーを組み込んで

いない状態と比較してオーバーヘッドを感じることはなかった。

#### (4) 保存したデータのサイズ

5.2(2)のテストでは、テスト時間が約 10 ~ 15 分 (説明の時間は除く) であり、そのテストによって保存したデータのサイズは、21 ~ 41 Kbyte であった。

ペンのジェスチャ入力の場合、ジェスチャの種類によって異なるが、1 入力のデータサイズは約 150 ~ 640 byte で、筆点列の座標データはその 72 ~ 94 % を占める。

#### 6.2 データ分析ツールの評価

このツール専用に記述したソースは C 言語で約 500 行で、ファイルからデータを読み込んで、メモリに格納する部分はライブラリとして作成したモジュールを使用した。このライブラリは約 400 行で記述されている。

### 7. 考察

本章では、システム評価の結果をふまえて UI 評価環境の考察を述べる。

#### 7.1 データロガーの考察

本節では UI 評価に必要なデータを保存する機構であるデータロガーの考察について述べる。

##### (1) ユーザの入力履歴を保存する機構の考察

ユーザの入力履歴の保存は、“未”用 AP のウィンドウ入出力ライブラリを改造することによって実現したので、AP のソースコードを変更せずに実現できた。このライブラリは AP に依存する部分がないので、AP とライブラリを再リンクすることで“未”用のほかの AP に対しても同様にデータを保存することができる。

ウィンドウ入出力ライブラリの改造は、データロガーとオリジナルのソースをできるだけ分離して実現したので、ライブラリが更新された場合も、簡単に対応できる。

##### (2) AP の動作履歴を保存するための機構の考察

AP の変更作業は一時間以内で終了した。これは AP 設計者がこの作業を行なったので AP のソースコードを解説する必要がなかったからである。

AP の動作履歴は、保存するの内容が AP に強く依存するため、ソースに手を加えずに収集することはできない。例えば、実行された関数名や引

数を自動的に保存するような機構を提供しても、その関数内の条件分岐の結果などの内部情報を得ることはできない。

この作業は、“未”用 AP の開発を支援する環境[6]を利用したり、専用のエディターを用意するなど、作業を自動的にこなす方向ではなく、評価者による変更を手助けする方向で支援できる。

### (3) データロガーによるオーバヘッドの考察

入力履歴や AP の動作履歴をバッファに保存するためのオーバヘッドは、ユーザの行動に影響を与えることがなく、方針は達成できた。

### (4) 保存したデータサイズの考察

イベントの種類やウィンドウ名は、2 byte の ID として保存したので、文字列で保存する場合よりもサイズが小さく、さらに ID と文字列との対応を保存してあるので情報量としては変わらない。

またペンによる入力データはサイズが大きく、その 7 ~ 9 割が筆点列の座標データである。しかし、筆点列の座標はすべての評価で必要とは限らない。したがって、データのサイズを抑えるためには、収集するデータの内容をユーザテスト時に設定できるようにすることが有効である。

## 7.2 データ分析ツールの考察

本節では、保存したデータを評価方法に合わせて分析するツールの作成による考察を述べる。

### (1) データ分析ツール用ライブラリの考察

ファイルに保存したデータをロードする部分をライブラリとして作成した。ライブラリを利用することにより、AP に特化したツールの作成の作成を支援することができた。

このことから、他の評価方法や AP の評価を支援するツールを設計し、どの部分が共通部分として分離できるのかを検討し、共通部分をライブラリとして提供することによって、さらにツールの作成が容易になると考えられる。

### (2) AP の動作履歴を利用したデータ分析の考察

AP の動作履歴をデータ分析に利用することにより、次に示すような AP に依存する内容の評価が可能となった。

- ・実行された機能別に入力データを集計することが可能となった。
- ・まったく同じ入力に対してオブジェクトの場合などによって AP が異なる動作をした場合で

も区別して評価することが可能となった。

- ・入力に対して AP がなにも反応しなかった場合、その原因の特定が可能となった。例えばジェスチャの誤認識、もしくは認識不可能だったのか、正しく認識したが入力場所がずれていたのかを判断することができた。

## 8. おわりに

本稿では、“未”用 AP の UI 評価環境の設計、実現とその評価について述べた。本研究によって、ユーザテストによるユーザの入力履歴の保存を自動的にこなすことが可能となった。また AP の動作履歴などを保存することにより、ユーザの入力履歴の分析が容易になることが確認できた。

今後は、分散手書き KJ 法以外の AP に実現した UI 評価環境を使用し、保存するデータの再検討や保存したデータの分析ツール用のライブラリの拡充を行なう。

また、AP の変更の手間や、データセーブによるオーバヘッドなど、UI 評価環境による影響をできるだけ抑えるための方法を検討する。

その後、手書き UI の AP を評価するモデルを提案し、複数の AP に対して UI 評価環境を用いて評価を行ない、モデルの検証を行なう。

## 参考文献

- [1] 来住：X Window におけるユーザー操作記録の比較ツールの開発，情報処理学会ヒューマンインタフェース研究会資料 53-6，1994
- [2] 粕川，角田，赤池：X-Window 上での事象タイミングデータ測定法について，情報処理学会ヒューマンインタフェース研究会資料 33-4，1990
- [3] 早川，河又，宮島，加藤，並木，高橋：ペンインタフェース研究・開発のためのウィンドウシステム“未”(HITSUJI)の設計と開発，情報処理学会論文誌，Vol.36 No.4，1995
- [4] 中島，早川，並木，高橋：ペンインタフェースを用いた分散 KJ 法システム，プログラミングシンポジウム，1995
- [5] 岡野，横関，並木，高橋：並列処理用 OS カーネル“OmicronV3”の開発とハイパ OS による共有メモリ型マルチプロセッサへの実装，情報処理学会論文誌，Vol.32，No.5，1991
- [6] 平井，早川，並木，高橋：ペンウィンドウシステムのためのプログラム開発支援系，情報処理学会第 50 回全国大会 7J-7，1995