

X Window におけるユーザ操作記録の比較ツールの開発

来住伸子

津田塾大学数学科

グラフィカルユーザインタフェースの操作記録を収集して解析することにより、ユーザインタフェース設計の評価を行うツール群、SimUI を開発中である。その操作記録比較解析ツールは、想定した使用方法と異なる使い方をしている箇所を検出することが可能だが、それを再設計し試作した。Perl や Emacs などの既存のツールを利用することにより、高速化と柔軟な機能の追加変更が可能になった。実際のユーザの操作記録の解析にツールを試用してみた結果、作業終了時間などの通常使用される指標と、SimUI の操作記録比較解析ツールが行う多段差分生成では、異なる種類の評価結果が得られることが観察できた。

Design and Implementation of an Automatic Analysis Tool for User Operation Records on the X Window System

Nobuko Kishi

Department of Mathematics

Tsuda College

Tsuda-machi, Kodaira-shi, Tokyo 102, Japan

SimUI, tools for the graphical user interface design evaluation, are being developed for X Window applications. Its analysis tool is redesigned and implemented, which detects the differences between two sets of the recorded data of users' operations on mouse and keyboards. By using existing tools such as Gnu Emacs and Perl, the tool enabled faster and more flexible analysis. The validity of the tool's analysis result is evaluated by a classification of actual users' behavior. The classification result is then compared with other indicators such as the task completion time and several differences are observed.

1 はじめに

ユーザインターフェイスの使い易さの評価方法にはさまざまな手法があるが、実際のユーザにシステムを操作してもらい、その様子を観察するという方法が使われることが多い。この評価方法は、想定していなかったユーザの操作方法を発見することに有効である反面、観察者の主観で評価結果が異なったり、観察記録の分析に時間がかかるなどの欠点がある。そのため、操作記録の自動解析のための方法がいくつか提案されてきた。しかし、ポインティング操作の記録が主になるグラフィカルユーザインタフェースの操作記録の自動解析は必ずしも成功していない。

そこで、我々は、グラフィカルユーザインタフェースの操作記録から、観察者が注目しなければならない操作や設計上の問題点を発見するためのソフトウェアツール、SimUI を開発中である [3, 1, 2]。今のところ、再生時のデータ収集と名付けた手法によって、計算機で記録可能な操作記録は、ほとんど集めることができるようになった。また、X Window サーバの改造により、多くの X Window Application を評価の対象とすることができるようになった。現在、取り組んでいるのは、操作記録の解析である。すでに多段差分生成と名付けた手法を提案したが、この報告では、その手法およびその他の解析手法を、ユーザインタフェース設計者が手軽に使用できるようにする比較解析ツールの設計と作成について報告する。まず、SimUI の記録、再生、データ収集の仕組みを簡単に紹介し、次に、比較解析ツールの設計の方針と機能のいくつかを紹介する。さらに、多段差分生成をはじめとするツールによる解析結果で、操作の習熟の程度をどのように分類できるかを評価する実験を行なった。実験の考察の後、比較解析ツールの将来について考察する。

2 ユーザインタフェース評価ツール: SimUI

グラフィカルユーザインタフェースの操作記録の解析の難しい理由の一つは、ポインティング操作の記録を解釈して計算機の状態を推定することに手間がかかる点にある。そこで、ポインティング操作の記録だけでなく、計算機の状態についてもできるだけ多くの情報を集めることを考えた。しかし、ユーザが実際に操作をしている時に多量の記録をとると、計算機の応答が遅くなり、ユーザ操作を妨げることがある。そこで、ユーザが実際に操作している時は

必要最小限の入力データを記録し、その入力データを再生している時に出来るだけ多くのデータを収集することにした。

この再生時のデータ収集は、SimUI の各種ツールを次のような順序で使用して行なう。

1. ユーザ操作の記録をとる。これには、SimUI 用の X ディスプレイサーバを使用し、操作記録ツールがこのサーバに接続すると、マウスとキーボードからの入力の記録がとれる。
2. ユーザ操作を再生する。記録の時と同じように SimUI 用の X ディスプレイサーバを使用する。操作再生ツールがクライアントとして接続すると、このサーバは、マウスとキーボードからの読み込みを止め、再生ツールからの入力情報をマウスとキーボードからの入力として処理する。
3. 操作の対象となるアプリケーションと SimUI 用 Xt ライブラリをリンクし直す。これによって、Xt ライブラリからアプリケーションのコードが呼ばれた記録をとることが可能になる。
4. もう一度ユーザ操作を再生しながら、データ収集を行なう (図 1)。まず、データ収集ツールを起動しておく。次に、X プロトコル分析ツールを起動してから、X クライアントとなるアプリケーションを起動する。最後に操作再生ツールを起動すると、操作再生ツールは、再生時と同じように X サーバに入力データを送る。この時、アプリケーションがプロトコル分析ツールに、プロトコル分析ツールが X サーバに接続するというように設定されているので、X サーバとアプリケーションの間のプロトコルの交換をデータ収集ツールに集めることができる。さらに、SimUI 用 Xt ライブラリにより、コールバックや イベントハンドラが呼ばれた記録もデータ収集ツールに集めることができる。

図 2 は、この再生時のデータ収集によって集められたデータの一例である。各行の先頭の数字は、どのツールからデータ収集ツールに送られたかを次のように示している。

```
s> 再生ツールからデータ
sc> X プロトコル分析ツールによるサーバーから
    クライアントへのデータ
```

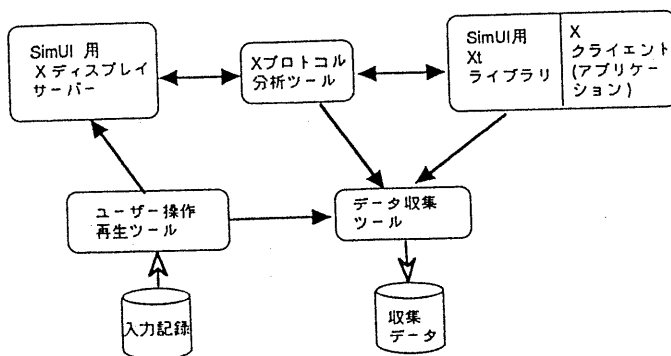


図 1: ユーザ操作の再生とデータ収集

cs> X プロトコル分析ツールによるクライアント
からサーバーへのデータ

xt>, ap> SimUI 用 Xt ライブラリによるデータ

多段差分生成は、このような二つのデータを比較して差を検出する際に、重要なデータが一致するよう、以下の処理を繰り返す繰り返す手法である。

1. 二つのデータから、それぞれ重要な記録を選ぶ。
2. 選んだ記録を差分生成ツール (UNIX の diff など) の入力とする。
3. 差分生成ツールの出力結果を解析し、互いに一致する記録に共通の番号付けを行なう。

以上の処理を、選び出す記録の重要度を下げながら繰り返し、最後には記録全体に対し差分生成を行なう。図 2 の中では、ap>handler がアプリケーションのイベントハンドラの呼びだしの記録なので重要度が高く、s>LOC はマウスの移動の記録なので重要度が低い。後述する解析記録ツールの表示 (図 3) では、左端のウィンドウが多段差分生成の結果を表示している。現在、3 回の差分生成を行なっており、行番号の表示の後の 3 文字が、各回の差分生成の結果を示している。

3 操作記録比較解析ツールの再設計

最初の SimUI 操作記録解析ツールは、市販の差分生成プログラムを繰り返して動かすことにより、多段差分生成を実現していた。しかし、この方法だと、次のような欠点があった。

- 使用した差分生成プログラムに提供されるフィルタの機能を利用していただけ、解析時間が非

常に長かった。

- 基準となる操作記録 (手本) に対し、差分をとるという手法なので、基準と操作の順序は多少異なるだけで、低い一致率であると判定していた。
- 現在の多段差分生成のフィルタが、自動比較に最適かどうか確定していない。また、すべての操作記録に同じフィルタが使用できるとは限らないが、フィルタの変更は手作業だった。

そこで、これらの欠点を改善し、ツールの実用性を高めるために、比較解析ツールを以下の目的で作成しなおすことにした。

- 差分生成の高速化
- 対話的な解析
- フィルタの柔軟な追加変更

新しい比較ツールは、Gnu Emacs, Perl, gdiff の 3 種の UNIX 上で入手しやすいツールを利用して以下のような構成にした。

Gnu Emacs

- ユーザ入力の解釈
- ウィンドウの管理
- Perl プログラムの起動

Perl

- 解析データの加工
- 正規表現フィルタの生成
- 解析データに対するフィルタの適用
- gdiff コマンドの起動
- gdiff コマンドの出力の解釈

gdiff

- 差分生成

```

s>LOC_Y_DELTA: 12388714251 mili value -27
s>LOC_X_DELTA: 12388714661 mili value 3
s>LOC_Y_DELTA: 12388714661 mili value -2
sc>EVENT: FocusOut detail: Nonlinear event: WIN2000038 mode: Normal
sc>EVENT: EnterNotify detail: NonlinearVirtual time: TIM49d7adaa root: WIN2b
sc>EVENT: FocusIn detail: Pointer event: WIN2000038 mode: Normal
sc>EVENT: FocusOut detail: Pointer event: WIN2000038 mode: Normal
ap>handler WID27728 index 0 proc ADRf76f0eac eventtype 7 //DispatchEvent2()
ap>handler WID27728 index 0 proc ADRf76f0eac eventtype 9 //DispatchEvent2()
ap>handler WID27728 index 0 proc ADRf76f0eac eventtype 10 //DispatchEvent2()
ap>handler WID27728 index 0 proc ADRf76f0eac eventtype 9 //DispatchEvent2()
s>LOC_X_DELTA: 12388715081 mili value -1
s>LOC_Y_DELTA: 12388715081 mili value 5
sc>EVENT: FocusIn detail: Nonlinear event: WIN2000038 mode: Normal
s>LOC_X_DELTA: 12388715501 mili value 1
sc>EVENT: LeaveNotify detail: Virtual time: TIM49d7add4 root: WIN2b event:

```

図 2: 再生時のデータ収集による操作記録例

そして、解析ツールを使用しながらでも、フィルタの追加修正が容易にできるようにするため、Emacs から Perl のプログラムを自由に使用できるようにした。例えば、任意の Perl プログラムに、操作ツールのカレントバッファ内の操作記録を入力として与え、その出力でカレントバッファの内容を置き換えるというプログラムは、Emacs Lisp では、以下のようになる。

```

(defun simui-perl (programe)
  (interactive "fEnter a perl program: ")
  (setq buffer-read-only nil)
  (goto-char (point-min))
  (call-process-region
    (point-min) (point-max)
    "perl"
    nil
    (current-buffer)
    ;; buffer name for the result
    nil
    ;; Update buffer infrequently
    (expand-file-name programe)
    ;; the name of perl program
  )
  (delete-region (point) (point-max))
)

```

つぎに、Perl で各種のフィルタを用意し、上述の Emacs Lisp プログラムを利用して、カレントバッファ内の操作記録を加工した。たとえば、Widget ID を正規化する（あるアプリケーションが X ツールキットを使用しはじめたときの、Widget ID を 1 として番号を付け直す）には、つぎのような Perl プログラムを使用した。

```

$wnum = 1;
while(<>){
  if(/WID([0-9a-f]+)/){
    if( $data{$1}){
      print('$',"WID",$data{$1}, '$');
    }else{
      print('$',"WID",$wnum, '$');
      $data{$1} = $wnum++;
    }
  }else{
    print $_;
  }
}

```

そして、頻繁に使用される Perl プログラムをよびだす Emacs Lisp 関数のいくつかを、Emacs の対話的なコマンドとして登録した。

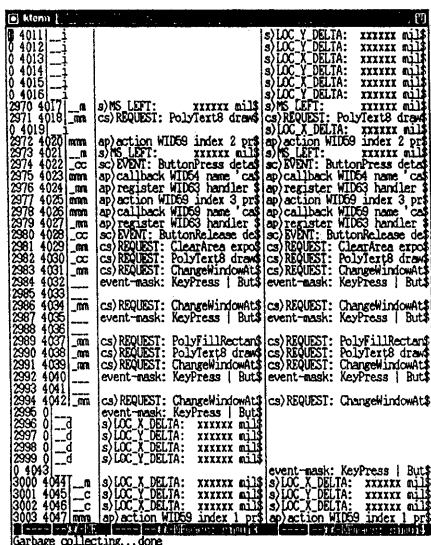


図 3: Emacs と Perl を利用した操作記録比較ツール

4 解析ツールの試用実験

作成した解析ツールの有効性を評価するために、簡単な実験を行なうことにした。解析ツールで調べられる評価基準によって、熟練者と初心者をつどのようにに区別できるかということを目とした。

まず、対象となるアプリケーションとしては、X Windows とともに標準的に配布される bitmap というプログラムにした(図 4)。bitmap は、簡単なドットパターン編集プログラムで、Athena Widget によって書かれている。課題とした作業は、黒い背景に白い星型を描くという作業である。このアプリケーションと課題にした理由は、次のような異なる種類の習熟が必要であると考えたからである。

- マウスによるポインティング操作
- アプリケーションプログラムとしての知識(反転や領域塗りつぶしコマンドの使い方)
- 幾何としての知識(どの順序で辺や領域を書くのが速いかという知識)

まず、すでに X Window 一般の使用経験のある 4 人の学生に、予告なしに、この課題を行なってもらった。次に、練習を好きなだけしてもらい、本人が初回より「上手に」(速くではない)できるようになっ

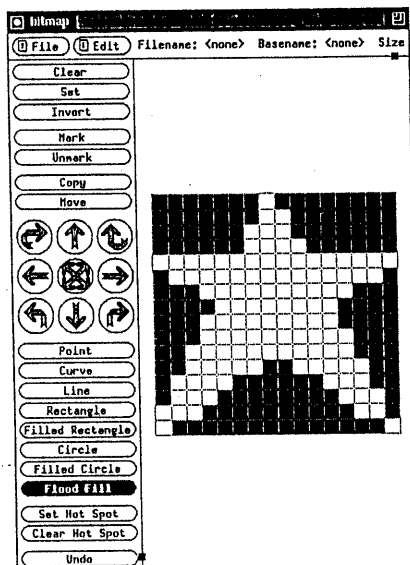


図 4: 評価実験に使用したアプリケーション bitmap

たと思えるようになった時点で、もう一度課題を行なってもらった。初回と 2 回目の比較結果は、表 1 のようになった。図の上 3 段は、多段差分生成の各回での一致率を示す。一致率とは、各回で対象となる記録の中で、互いに一致する記録の占める割合を示す。ユーザ A の 2 回目の操作記録を差分生成の基準とした。

どのユーザも作業時間を短縮しているの、速さという点ではどのユーザも習熟していることが分かる。しかし、実際に使用の様子を眺めてみると、習熟した要素が以下のように異なることが分かった。

- マウス操作が速くなった。
- マウス操作に誤りが少なくなった。
- 便利なコマンドの使用法を覚えた。
- 作業効率があがるよう、異なる図形の描き方をした。

どの要素で習熟したかが、各レベルの一致率の差とかなり関係がある。たとえば、ユーザ D は、2 回目でアプリケーションレベルの一致率を下げている。これは、他のユーザが使用していないコマンドも利用することにより星型を描いたためである。プロトコルレベルやサーバレベルの一致率は上がっているの、誤りではなく、星型の描き方としては、より

効率の高い方法であることが確認できる。

表1：操作記録の一致率と作業時間の変化

ユーザ A	初回	2 回目
アプリケーションレベル (%)	65.5	100
プロトコルレベル (%)	58.8	100
サーバレベル (%)	48.9	100
作業時間 (秒)	43.1	36.4

ユーザ B	初回	2 回目
アプリケーションレベル (%)	44.2	50.3
プロトコルレベル (%)	40.3	61.1
サーバレベル (%)	31.8	48.7
作業時間 (秒)	82.0	46.1

ユーザ C	初回	2 回目
アプリケーションレベル (%)	41.5	51.7
プロトコルレベル (%)	23.7	38.2
サーバレベル (%)	16.0	27.5
作業時間 (秒)	392.2	131.4

ユーザ D	初回	2 回目
アプリケーションレベル (%)	30.9	22.01
プロトコルレベル (%)	24.1	33.0
サーバレベル (%)	18.7	26.4
作業時間 (秒)	255.4	84.3

5 まとめ

今回の解析ツールの設計作成の目的とした3点のうち、多段差分生成の高速化は、一応達成できた。前回と使用機種や操作記録が異なるので、単純な比較はできないが、操作時間が2倍以上の記録が、3分の2ほどの早さで処理できるようになった。

対話的な解析は、Emacsの使用により、非常に容易できるようになった。以前の解析ツールにも2つの操作記録をタテに並べて同時にスクロールするという機能はあったが、テキストエディタの機能を備えている訳ではなかった。Emacsは、我々が普段使用しているエディタであることもあり、操作記録内の移動や探索が非常に手軽にできるようになった。

機能の追加変更も、EmacsとPerlの使用により非常に簡単になった。EmacsとPerlのユーザで

ない人にはあまり簡単でないとも言えるが、機能の追加が一区切りつけば、機能を固定できるようになると思われる。

以上の3点により、今まで非常に短い操作記録しか解析できなかったのが、10分程度の作業まで評価できるようになった点が、今回の再設計の最大の成果だと思う。試用実験の方は、実験をしつつ比較解析ツールに機能を追加するという手順になったために、結果の解析は不十分な点も多かった。この報告に解析の間に合わなかった実験データの解析もあり、

bitmapのユーザインタフェース設計上の問題点と、解析結果はどのように関連するのかについての、より詳しい考察が可能だと思う。

さらに、今回の比較解析ツールの作成と実験によって、解析ツールの今後として、以下の二方向があることが分かった。

1. 多くの操作記録の中から、問題となる操作記録を選び出す。
2. 数個の操作記録の中から、問題となる箇所を指摘する。

前者には、一つの記録に対してフィルタをかけて、重要と思われる情報を抜き出すという手法が適し、後者には、多段差分生成のような、少しでも重要と思われる差は検出するという手法が適している。両者に同時に使用できる解析ツールの作成が可能かどうかはまだ分からないが、今回作成した柔軟な機能追加変更の仕組みを利用しつつ、実験とツールの改善を繰り返す予定である。

参考文献

- [1] N.Kishi. SimUI: Graphical User Interface Evaluation Using Playback. Proceedings of the Sixteenth Annual International Computer Software and Applications Conference, COMP-SAC92. pp121-127, 1992.
- [2] 来住伸子, ユーザインタフェースデザイン評価支援ツール SimUI の試作. 情報処理学会プログラミング-言語・基礎・実践-研究会報告 93-PRG-10 pp89-96, 1993.
- [3] 来住伸子, ネットワーク環境におけるユーザ操作記録の解析手法. 情報処理学会ヒューマンインタフェース研究会報告 47-2, pp7-14, 1993.