プログラミング—言語・基礎・実践— 10−12
（１９９３．１．２９）

ユーザーインターフェイスデザイン評価支援ツール SimUIの試作

来住伸子
津田塾大学数学科

計算機資源の入手しやすさとともに、グラフィカルユーザーインターフェイス設計作成環境が整備されてきたが、十分に評価しないまま作成するために、使いにくいグラフィカルユーザーインターフェイスになりがちである。そこで、ユーザーインターフェイス設計を評価するためのツール、SimUI を作成することにした。SimUI は、再生時のデータ収集と多段差分生成により、想定した使用方法と異なる使い方をしている箇所を、ユーザーの操作記録から検出する。この報告では、SimUI作成の動機と主な機能を紹介した後、UNIX のネットワーク環境での上での実現方法とネットワーク機能の利用によって新しく可能になった機能を紹介する。

Graphical User Interface Design Evaluation Tool: SimUI

Nobuko Kishi

Department of Mathematics
Tsuda College
Tsuda-machi, Kodaira-shi, Tokyo 102, Japan

Usability evaluation is often neglected during software development processes, because it is still not considered as part of programming activities. We developed a set of tools, named SimUI, for evaluating graphical user interface designs. SimUI automatically detects the differences between two sets of the recorded data of users' operations on mouse and keyboards. Its two main techniques are data-gathering during playback and multi-step matching of recorded data. This report describes SimUI's main goals and functions, and then describes how these functions are implemented in the X Window environment. Furthermore it discusses the use of network communication for conducting usability tests in real end-user enviroments.

1 はじめに

ソフトウェアの使い易さへの関心の高まりとともに、ソフトウェア開発作成においてユーザーインターフェイス設計に多くの時間と労力が使われている。しかし、ソフトウェア作成者の経験や主観に頼る部分が多く、時間と労力を使えば、使い易いソフトウェアになるとは限らないのが現状である。

そこで、ユーザーインターフェイスの使い易さを評価するために、ユーザーの操作記録の収集および記録の比較参照を行なうツール、SimUI (SIMulated User Interaction analyzer) を開発することにした。ソフトウェア開発のテスト段階でデバッガやプロファイラを使用するように、ユーザービリティテストの段階でこのツールを使用することを目指している。

SimUI の最初の版は、OS/2 の環境で作成された。現在、UNIX のネットワーク環境と X ウィンドウ上に移行しつつある。この発表では、OS/2 版を元に SimUI 作成の動機と主な機能を簡単に紹介した後、UNIX 上での実現方法と UNIX 上で新しく可能になった機能を紹介する。

2 SimUI の動機および関連する研究

使い易くなるという言葉には大きく分けると二つの解釈が考えられる。一つは、今まで使用できなかった人が使用できるようになる — 能力 (competence) を持つという解釈、もう一つは、既に出来ることがより簡単に速くできるようになる — 作業能率 (performance) が上がるという解釈である。前者の解釈による使い易さの向上には、ユーザーの理解ということが大きな役割を持つ。後者の解釈による使い易さの向上には、予想した使い方と異なる使い方をユーザーがするかどうかを確認が重要になる。なぜなら、作業能率を上げるには、作業分析などによりユーザーの使い方を調べておき、その使い方に合わせたデザインをするからである。もし、ユーザーインターフェイスの設計者の意図と異なる使い方をユーザーがすると、作業能率が予想を下回る可能性が高く、デザインに何らかの問題があることになる。

一方、使用できなかった人を使用できるようにするためには、ユーザーの操作記録を観察して、ユーザーがなぜ使用出来なかったかを推定する作業が必要である。たとえば、ユーザーの手が止まった時、

- 計算機の現在の状態が分からない。
- 次に何をすればいいかが分からない。
- 次のコマンドが思い出せない。
- キーの位置を探している。

のどれかを判断することが必要になる。

これは、ユーザーが計算機を対話的に使用するには、高次レベルの情報処理と低次レベルの情報処理の両方が必要で、ユーザーが計算機の理想的な使い方をしているときには, 状況に合ったレベルでの情報処理を行っているはずだという考え方に基づいている [7]。たとえば、複雑な作業には、つぎのような段階を経ていると考えられる。

1. 画面上のイメージを見る。
2. イメージの中から情報を認識する。
3. 認識した情報と既に持っている知識を照合して計算機の状態を確認する。
4. 既に持っているプランを照合してつぎに必要なコマンドやオブジェクトを決定する。
5. コマンドやオブジェクトを操作に変換する。
6. 操作を実行する

一方、簡単な作業には低次レベルの段階だけを使用し、画面を見たとたんに手を動かす、というようにユーザーは振舞う。

したがって、問題点およびその原因を探す時、上記の各段階のどこで誤りが起きたかを見つけ出すという作業が重要になる。このとき、観察者は主観によって、ユーザーの頭の中で何が起きているかを、推定して判断していることになる。そこで、観察者の主観の代わりに、ユーザーの頭の中で何が起きているかを推定する方法として、計算機の方の状態の記録からユーザーの状態を推定することが考えられた。この考え方の最初の試みとしては、キーの打鍵時間から、キーボードのキーの配置やコマンド名の問題点を見つけ出すという研究があげられる [1, 3]。しかし、最近のグラフィカルユーザーインターフェイスの普及とともに、単純にキーやマウスの位置と時刻の記録だけから、ユーザーの状態を推定することは難しくなってきた。

3 SimUI の機能

SimUI では、グラフィカルユーザーインターフェイスに対するユーザー操作記録の解析の困難さを解

消するために、ユーザーの入力情報だけでなく、オペレーティングシステムとウィンドウシステム間、およびウィンドウシステムとアプリケーション間の情報も記録することによって、ユーザーの状態を推定することにした。そして、その実現のために、以下の2点の技術を使用した。

1. ユーザー操作の再生を利用したデータの収集 [4]
2. 多段差分生成によるデータの比較参照 [5]

これらは OS/2 の下で既に実現されている [6]。第一点は、ユーザーが実際の操作をしている時にはユーザーの入力情報のみを記録しておき、その後、入力情報を使用してユーザー操作を再生しながら計算機の状態のデータの収集を行なうことを指す。第二点は、収集したデータの中の情報から、

- 基準となる使い方と異なる使い方をしている.
- 基準となる使い方と一致しているが, 異なる速度で使用している。

という箇所を探し出すための手法である。これは、ユーザーと、ユーザーインターフェイスの設計者自身または熟練者が、同じ作業したときのデータを比較することによって行なう。二つの操作記録を再生した時に収集したデータ中の、各レコードを重要度によって分類し、重要度の高い順に比較照合することによって、予想外のユーザーの行動に対応する操作記録を差分として検出する。 さらに、この差分結果を視覚化することによって、ユーザーが予想外の使い方をした箇所を観察者に分かりやすく表示する。図 1 では、差分があるかどうかを、右上の三個の円内の色に対応させ、その下のメーターは速度の相対的な差を表示している。

4 UNIX 版 SimUI の実現

UNIX 版の SimUI は OS/2 版の移植ではなく、新しく作成中のツールで、OS/2 版と同様の機能を実現しつつ、ネットワーク環境の利用による新しい機能の追加を目指している。UNIX 版の作成によって、再生時のデータ収集という手法が、特定の OS に依存した手法ではないということと、ネットワーク機能の利用によって、実際のユーザー環境の中でのデータ収集と解析が簡単になるということが分かってきた。

4.1 UNIX 版 SimUI の構成

UNIX 版の SimUI の再生時のデータ収集は以下のツールによって実現されている。

- SimUI 用 X ディスプレイサーバー
- 操作記録ツール
- 操作再生ツール
- X プロトコル分析ツール
- SimUI 用 Xt ライブラリ
- データ収集ツール

ユーザー操作を記録するには、まず, SimUI 用の X ディスプレイサーバーを起動する。このサーバーは通常は普通の X サーバーだが、操作記録ツールがクライエントとして接続してくると、マウスとキーボードからの入力情報を操作記録ツールに送る (図 2)。

ユーザー操作を再生するには、記録の時と同じように SimUI 用の X ディスプレイサーバーを起動する。操作再生ツールがクライエントとして接続してくると、このサーバーは、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思って処理するという動作をする (図 3)。

データ収集には、まず、データ収集ツールを起動する。その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 サーバーに入力データを送るとともに、データ収集ツールにも接続して入力データを送る (図 4)。

次に、X クライエントとなるアプリケーションを起動する前に X プロトコル分析ツールを起動しておく。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 サーバーに接続するというように設定すると、X サーバー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、X ツールキットイントリシクス （Xt ライブラリ）を使用したアプリケーションの場合、SimUI 用 Xt ライブラリとリンクし直すことによって、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もデータ収集ツールに集めることができる。

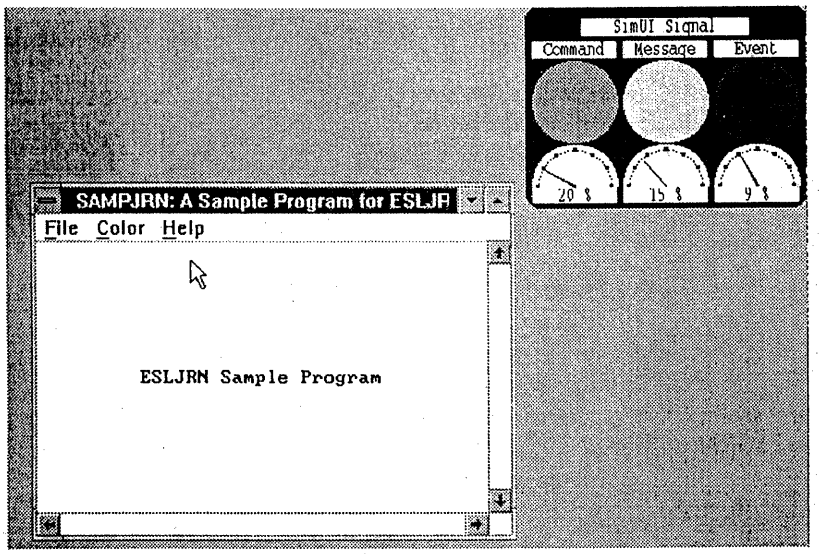

図 1: SimUI の差分視覚化ツール

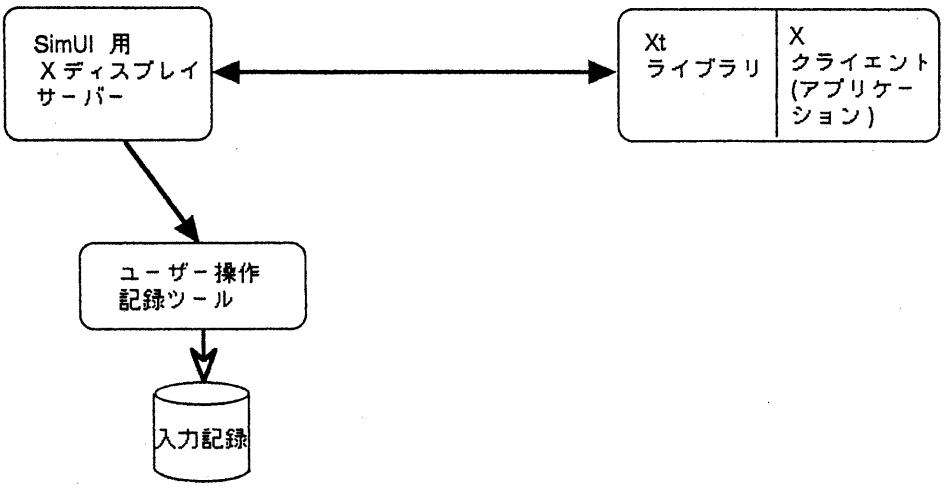

図 2: ユーザー操作の記録

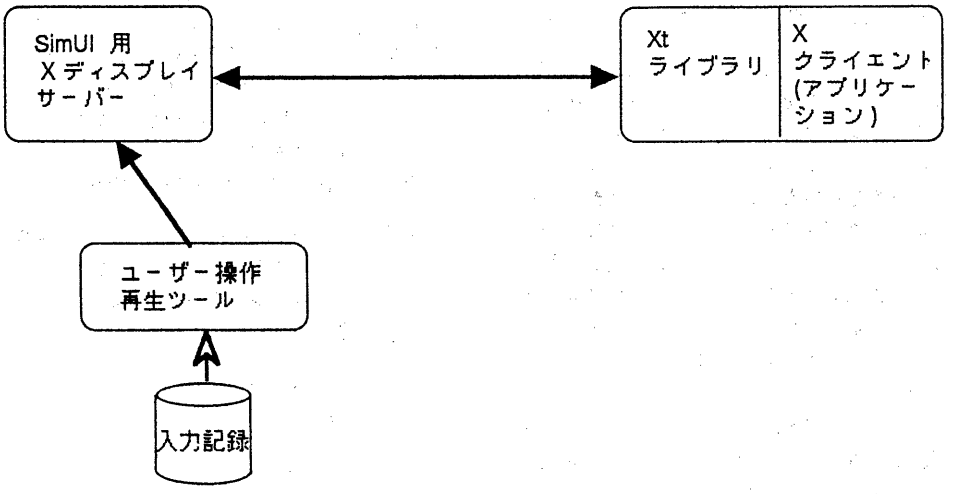

図 3: ユーザー操作の再生

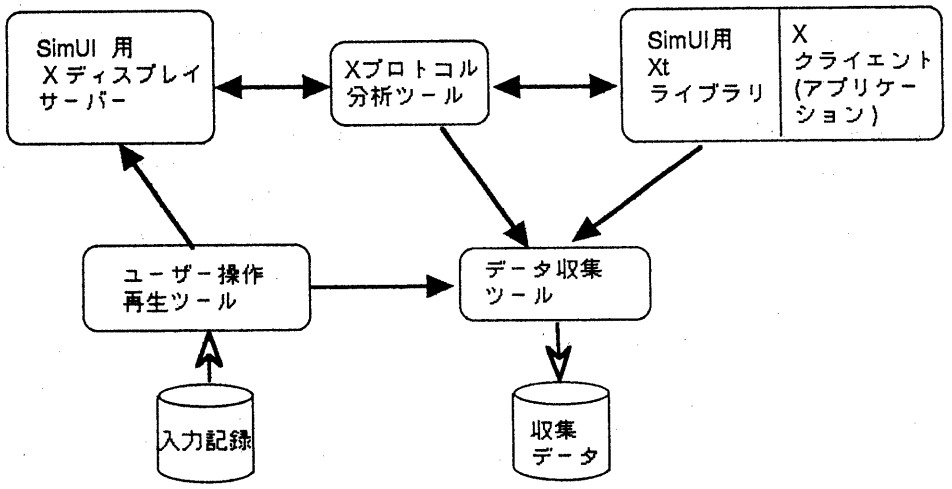

図 4: ユーザー操作の再生とデータ収集

4.2 SimUI 用 X ディスプレイサーバー

ユーザー操作の記録と再生を行うために，X サーバーのソースコードを変更した．ユーザー操作の記録を集めるには、デバイスドライバやストリームモジュールなど、ウィンドウシステムではない部分の変更によっても可能だが，ネットワーク機能の使用および安定した再生のためにサーバー部分を変更することにした．また、サーバー部分の変更を最小限にし、データ収集などの機能は別のクライエントやサーバーにすることによって、ツールの拡張性や保守性が高まるという研究が既にあったことも参考になった [2]。

X ウィンドウの機能はネットワークプロトコルとして規定されているので，X サーバーの構造は，それが動く計算機ごとに異なる．この SimUI 用の X サーバーには、X ウィンドウとして配布されるテープに含まれる Sun 用の標準的なソースコードを使用した．しかし、この ソースコードは Sun のハードウェアや SunOS に大きく依存している訳ではなく，次のような機能をもつワークステーション上に移植されたサーバーの代表的な例としてあげられているものである [8, 9]。

- ビットマップディスプレイ
- 4.3BSD 系の UNIX
- ソフトウェアカーソル
- I/O シグナルを出す入力デバイス

したがって、Sun でなくとも、このような機能をもつワークステーション上の UNIX と X ウィンドウの元なら、SimUI は簡単に移植できるはずである。

上記のワークステーションの場合、X サーバーは次のような方法で、各クライエントからのリクエストとユーザーからの入力イベントのスケジュール処理を行なう。まず、X サーバー自体は一つのプロセスとして動作している．複数のクライエントからのリクエストを一定の時間ずつ順番に処理しつつ、合間に、X ウィンドウの入力イベントキューを処理して必要ならクライエントに知らせるという動作を繰り返す。

一方、入力デバイスを実際にユーザーが動かしたことをオペレーティングシステムがを検知すると、シグナルハンドラが呼ばれる。このシグナルハンドラは X サーバーの一部だが、ワークステーションによって異なる部分に含まれる。このシグナルハンドラは、ハードウェアに依存した入力イベントキューから、X ウィンドウの入力イベントキューにイベントを変換して移動し、必要ならばマウスポインタの表示位置を移動する。

SimUI 用には主に次の 2 点を変更した。

- SimUI 用に定めておいたアドレスに接続してきたクライエントがいたら、そのクライエントからのリクエストに基づいて，SimUI 用の状態変数を
  - 通常モード
  - 記録モード
  - 再生モード

  の間で切替える。
- シグナルハンドラの中で、上述の状態変数をチェックし、モードに応じて、次のように処理する。入力デバイスのキューから
  - 通常モードのときは、通常の X サーバーと同じ処理をし、入力デバイスのイベントキューの情報を X ウィンドウのイベントキューに移す。
  - 記録モードの時は、通常モードの時と同じ処理をし、さらに入力デバイスからのイベント情報を SimUI の記録ツールに送る。
  - 再生モードの時は、SimUI の再生ツールからうけとったイベント情報を X ウィンドウのイベントキューに移す。

X サーバーはもともと、リクエスト処理のスケジューリングを調整できる構造になっている。サーバーのソースコードを全部読まなくても、必要な部分の構造を理解出来れば、これらの機能を組み込むのは、それほど難しくない。

4.3 SimUI 用 X プロトコル記録ツール

このツールは、X サーバーに対してはクライエントとして接続し、X クライエントにはサーバーとして接続することにより、X サーバーと X クライエントの間のイベントとリクエストの交換を記録する。これは、X ウィンドウソースコードとともに配布されているツール、`xscope` を次のように改造したものある。

- 標準出力に表示されていたプロトコルの解読結果をSimUIのデータ収集ツールにネットワーク経由で送るようにした。
- 一つのイベントまたはリクエストが一つのレコードになるように、プロトコルの解読結果を短縮した。
- X11R4向きに書かれていた部分をX11R5向きに修正した

4.4 SimUI用のXtライブラリ

Xlibレベルの Xウィンドウの振舞いは Xプロトコル分析ツールで記録することができるが、Xのアプリケーション開発には 各種の Widget を使うことが多く、Xlibレベルの振舞いからアプリケーションの動作を理解することが難しい。そこで、Xtライブラリのソースコードに手を加え、Coreクラスの基本メソッド、Initialize や Exposeが呼び出されたり、Callback 関数や Event ハンドラが呼び出されたときも記録することにした。

5 ネットワーク環境で可能になった機能

ユーザー操作の再生時のデータ収集を実現するために、OS/2版では、セマフォや共有メモリとなどによってプロセス間通信を行なっていたが、UNIX版では、ネットワーク機能、正確にはソケットでプロセス間通信を行なうようにした。これは、Xウィンドウとの同期をとるためにしたことだが、ネットワーク機能の利用によって、次のような機能の実現が可能になった。

ユーザー操作の高速再生 ユーザー操作の通常の再生の時は、ある入力イベントと次の入力イベントとの間に、記録したときと同じもしくはそれ以上の時間をおくという方法によって、安定した再生を行なう。この方法だと、再生には、記録より時間が長くかかるし、ユーザーが何もせずにじっとしている時が長い作業の時など、観察が大変である。そこで、データ収集だけが目的の再生の時は、X サーバーとクライエントの間にプロトコルが流れていなければ、ある入力イベントと次の入力イベントの間は一定の最大時間しかおかないいう方法によって再生の高速化が可能になった。

また、ネットワーク環境の利用によって実現可能だが、セキュリティとプライバシ上の問題のため、実際には実現していない機能としては、次のものがある。

実際のユーザーが使用している環境でのデータ収集 SimUI 用の X ディスプレイサーバーは、通常は普通の X サーバーと全く同じなので、実際のユーザーが通常の作業に使用することが可能である。そして、ユーザー操作の記録が必要な時に、ネットワーク上のどこかで記録ツールを立ちあげればユーザーに気づかれずに入力記録を集めることができる。

複数のユーザーのデータ収集の省力化 開発中のあるアプリケーションについて、出来るだけ多くのユーザー操作記録をとりたいという時がある。その場合、操作記録ツールを起動するというコードをアプリケーションに組み込むことによって、アプリケーションの操作記録を自動的に集めることが可能になる。UNIXのように、マルチユーザーのオペレーティングシステムの場合、だれがアプリケーションを起動したかがユーザー ID で簡単に同定できるので、収集したデータの管理も簡単にできることになる。

6 現状と今後の方向

現在, UNIX 版の SimUI はデータ収集ツールの試作を終え, データ収集を試験的に行ないつつ、以下の2点について検証している段階である。

1. 再生やデータ収集が安定して行なえているか。
2. 多段差分生成を効率良く、および精度良く行なうには、どんなデータが必要もしくは不要か。

残るデータ解析ツールは OS に依存する部分が元々少ないので、OS/2 版とほとんど同じ方法で多段差分生成の実現が可能だと考えている。ただし、UNIX版で収集されるデータに含まれる情報が、旧版とかなり異なるため、どのデータをどの段階でフィルタするかということは、決定し直さなくてはいけない。そのためにも、上述の検証を行ないつつ、いろいろな状況でのデータの特徴を調べている最中である。

また、現段階で気づいた問題点の一つとして、Xtライブラリと Athena Widget を使用したアプリケーションが、大学には意外に少ないという点がある。現在は、Xt の使い方を紹介した本に載っている例題プログラムで、SimUI 用の Xt ライブラリによるデータ収集のテストを行なっている。しかし、実用的なアプリケーションに SimUI が使用できるよ

うになるには、商業的に使用されていることの多いOpenLookやMotifのWidgetに対応するか、ツールキットレベルのデータには頼らず、Xプロトコルと Xlib レベルでのデータのみでの解析の可能性を調べるかの道を選ぶ必要があると考えている。

7 まとめ

計算機資源の入手のしやすさとともに、グラフィカルユーザーインターフェイス設計作成環境が整備されてきたが、グラフィカルユーザーインターフェイスの評価のための環境は、まだほとんど整備されていない。人と時間の制約から十分に評価しないまま設計されたために、せっかくのグラフィカルユーザーインターフェイスもなぜか使いにくいということが多い。従来不十分に行なわれていた評価も、計算機の能力を利用すれば、手軽に行なえるようになるということをSimUIによって示したと思う。

また、今回の UNIX 版の作成によって、

- 再生時のデータ収集という手法が特定のOSに制約された手法ではなく、標準的なOSで標準的なウィンドウシステムであれば、実現可能である。
- ネットワーク機能の利用により、ユーザーの操作記録の大量のデータ収集と解析が便利になる。

ということが分かった。

SimUIをさらに実用的なツールとすることによって、ユーザーインターフェイスの評価という作業が、ソフトウェア開発において標準的な一段階として行なわれるようにし、使い易いユーザーインターフェイスを実現することに役立てていきたい。

参考文献

[1] S.K.Card, T.P.Moran and A.Newell. The Keystroke-Level Model for User Performance Time with Interactive Systems. Communications of ACM, Vol. 23, No.7, pp396–410, 1980.

[2] 赤池英夫, 粕川正充, 角田博保. X-Window 上での利用者行動分析ツール. 情報処理学会第32回プログラミング・シンポジウム報告集, pp121–130, 1991.

[3] 木村泉. 打鍵レベル模型について –概説と問題提起–. 情報処理学会計算機システムのヒューマンインターフェース – モデル・評価・展望 – シンポジウム論文集, pp11–24, 1988.

[4] 来住伸子, ユーザー操作の再生機能を利用したUI評価の試み. 情報処理学会第33回プログラミング・シンポジウム報告集, pp135–143, 1992.

[5] 来住伸子, 多段差分生成によるユーザー操作記録の自動比較. 情報処理学会ヒューマンインターフェイス研究会報告 41-1, pp77–84, 1992.

[6] N.Kishi. SimUI: Graphical User Interface Evaluation Using Playback. Proceedings of the Sixteenth Annual International Computer Software and Applications Conference, COMPSAC92. pp121–127, 1992.

[7] D.Norman. *The Psychology of Everyday Things.* Basic Books, New York, 1988.

[8] S. Angebranndt et al. Definition of the Porting Layer for the X V11 Sample Server. Included in the X window source tape, 1991.

[9] D. Rosenthal et al. Godzilla's Guide to Porting the X V11 Sample Server. Included in the X window source tape, 1992.