2 種類の対話制御情報を利用したユーザ操作履歴獲得技術とその検証

The Technology for the Acquisition of User's Operations Log Utilizing Two Types of Dialogue Control Information

長崎　　等・東　　基衞（早稲田大学）
Hitoshi NAGASAKI・Motoei AZUMA (Waseda University)

概要

ユーザ操作履歴情報を用いてユーザの特性を明らかにするために、文法レベルと意味レベルの異なる 2 種類のユーザ操作履歴を同時に獲得できる仕組み開発した。またその仕組みを組み込んだソフトウェアも開発した。この仕組みは我々が適応型ユーザインタフェースの実現に向けて考案した 2 種類の対話制御部分を持つ概念モデルをベースとしている。2 種類の対話制御に利用される情報を用いることによって、文法レベルと意味レベルの 2 種類の操作履歴情報を獲得することが可能となった。さらにこの仕組みを組み込んだソフトウェアを使用することによって、ユーザナビゲーションなどの研究が容易に可能となった。また本ソフトウェアを用いて実験を行うことによってこの方法の実用性の検証を行うことが出来た。

キーワード：適応型ユーザインタフェース、操作履歴、ユーザインタフェース評価

Abstract

In order to recognize personal characteristics of a user by using the operations log, we developed a mechanism that acquires two different types (syntactic level and semantic level) of user's operation log. We also developed the software in which the mechanism is embedded. This mechanism is based on the user interface (UI) model, which we designed for implementing the adaptive user interface systems. The UI model has two types of dialog control component. With the use of the information which is used to control dialog, the mechanism can acquire these two different types of user's operations log. The software allows the study of user navigations. By the experiments with the software, we could verify the effectiveness of the mechanism.

Keyword: Adaptive User Interfaces, User Operations Log, Evaluation of User Interfaces

目次

1. はじめに
2. システム概念モデル
   2.1　操作履歴利用の要求
   2.2　概念モデルのコンセプト
   2.3　概念モデルの概要
   2.4　概念モデルの動作
   2.5　実行可能なモデルへの拡張
   2.6　モデルの特徴
3. 2 つの異なるレベルのユーザ操作履歴
   3.1　従来の操作履歴獲得手法
   3.2　センサーによる操作履歴の取得
   3.3　ユーザイベントレベルでの履歴
   3.4　メッセージレベルでの履歴
   3.5　メッセージの分類
4. Object Designer
   4.1　Object Designer の概要
   4.2　Object Designer の構造
   4.3　Object Designer の動作
   4.4　履歴のデータ構造
5. Object Designer を用いた実験
   5.1　実験の目的
   5.2　実験の概要
   5.3　実際の結果
6. 有用性の検証
   6.1　概略
   6.2　データの有用性の検証
   6.3　システム概念モデルの検証
   6.4　Object Designer を用いた応用研究
7. おわりに

1. はじめに

進化適応型分散情報システム DAISY（Distributed Adaptive Information SYstems）プロジェクト[1]の一環として、我々が開発した 2 種類の対話制御機構を持つユーザインタフェースモデルを基にユーザ操作履歴を獲得するための技術とその技術を組み込んだオブジェクトモデリングツール Object Designer を開発した。この研究の背景には、ユーザインタフェースの適応やユーザナビゲーションなどのサポートを行うためには、その動的な変更の指標として個々のユーザの特性を知ることが必要であり、それらを自動的に取得するためにユーザの操作履歴を利用したいという要求があることが挙げられる。

この要求を満たすために本研究では適応型ユーザインタフェースのために我々が考案した 2 種類の対話制御部分を持つ概念モデルを利用することとした。このモデルを前提にして実際に稼働可能なアーキテクチャモデルを考案し、ユーザの履歴を獲得する仕組みを作成した。この仕組みにより、本ソフトウェアにおいては文法レベルと意味レベルの 2 つの異なる操作履歴が同時に獲得可能となった。また本ソフトウェアはユーザ履歴を利用して様々な研究を行うためのフレームワークとして利用されている。

本論文ではまず Object Designer に用いたアーキテクチャモデルについて詳細に言及した後、操作履歴を取得するための構造及び獲得可能なデータの説明をおこなう。また開発したソフトウェアを用いて行なった検証実験について言及し、このモデル及び獲得したデータの有用性について検証する。さらに本ソフトウェアを用いて行った他の応用研究についても触れる。

2. システム概念モデル

2.1 操作履歴利用の要求

ユーザインタフェースの適応やユーザナビゲーションなどといったリアルタイム型の適応型ユーザサポートを行うためには、何らかの形でユーザの現在の状態を把握することが必要であり、そういった際に一番利用しやすいのがユーザが操作した履歴である。またそれ以外にも認知工学的な研究やユーザインタフェースのレビューなどの作業を行う上でもユーザの振る舞いを記録できることは分析に不可欠である。

一般にユーザがソフトウェアを使って業務を行うとき、その業務を遂行するのに必要な個々のタスクを実行するのであるが､これは実際にはソフトウェアのもつ機能を利用して､そのタスクを行うということができる。

Norman の 7 段階モデル[2]や GOMS モデル[3]に近い考え方でユーザのタスクの実行を考えてみると、まず「目標状態」の想起が必要となる。次に「その状態に導く行為」を

想起できなければならない。その次に「行為の具体的な手順」を想起する必要がある。

「その状態に導く行為」はユーザの行う操作の意味的なものであり、「行為の具体的な手順」はユーザの操作の系列で表される。その系列は機能の文法の系列でもある。

従来の操作履歴は一般的には、システムが生成するイベントをフックして、その情報を元に操作履歴として記録するものが多く、したがって文法レベルの操作履歴を残すものが多い。しかしながら文法レベルの操作履歴からユーザの行った意味を再構築することはかなり難しく、手間のかかる作業である。逆に意味的なレベルでの操作履歴はユーザのタスクの手順等を知るには有効であるが、どういった操作方法を用いてタスクを行ったかという情報を得ることは出来ない。

このように様々な場合においては、意味的な情報を必要とする場合が多いが、適応やユーザナビゲーションといった場合には両方の情報を必要とすることも多い。

つまり文法レベルの情報だけでなく意味的な情報の両者が揃うことによって様々な分析を有効的に行うことができる

2.2　概念モデルのコンセプト

このモデルは適応型ユーザインタフェースを実現するために考案したもの [4]、[5] であり、ユーザインタフェースの動的な変更を可能とする機能が必要である。この機能に関して要求されるのは大きく View の部分の変更機能と Control の部分の変更機能の 2 つに分類できる。

View の部分については「部品レイアウト」、「対話部品」、「部品の外観」の 3 つの変更を実現することであり、これらは UI を構成するオブジェクトの構成データを書き換えることによって UI を構成する部品に対する変更が実現される。もう 1 つの Control の部分は「イベント処理レベル」、「タスク処理レベル」の 2 つのレベルにおける対話の遷移の変更を実現することであり、これらはオブジェクトのメソッドの置き換えによる操作プロセスの変更によって実現される。

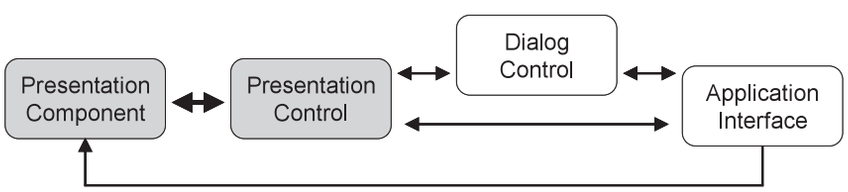

図 1：概念モデル

2.3　概念モデルの概要

図 1 のモデルは Seeheim モデル [6] を拡張したもので、4 つの構成要素からなるモデル

である。これらは互いに独立しているが、Presentation Component と Presentation Control との関係は他の関係と比較して結合度が高い。また、Presentation Component 及び 1 つないし複数の Presentation Component と対応している Presentation Control から構成される塊を Interaction Object と呼ぶこととする。

このモデルでは Seeheim モデルと違い Presentation Control 及び Dialogue Control の 2 つの制御部分を持つ。Presentation Control はユーザのイベントの処理を直接行い、Dialogue Control はタスクレベルでの対話制御を行う。また Presentation Control が Presentation Component と頻繁にコミュニケーションを行うことを想定しているのに比べ、Dialogue Control は Seeheim モデルと同様にトークンレベルのコミュニケーションしか想定していない。以下にモデルの各要素の概要を示す。

Presentation Component

この部分は Seeheim モデルと同様に論理的デバイスレベルを扱う。つまり字句レベルのフィードバックを含む論理的デバイスの制御と画面の外観やレイアウトを扱う。

具体的には Window、Frame などのインスタンスがこの Presentation Component に相当する。

Presentation Control

Seeheim モデルにおける Dialogue Control の機能の一部をこの部分で制御する。

論理デバイスの切換等が行われない状態においてその制御を行う。つまり Presentation Components のインスタンスの変更を行わないレベルでの対話を制御する。1 つの Presentation Component に対して複数の Presentation Control が制御を分担している場合もある。ここで扱う制御はイベントモデルで比較的容易に記述できるものである。

具体的には Window、Frame などの振る舞いに関してここで制御を行う。

Dialogue Control

Presentation Components のインスタンスの変更を伴うレベルでの対話を制御する。この部分の制御は状態遷移図などで比較的容易に記述できるものである。

具体的にはタスクレベルでの切り替えや、1 つの Window、Frame 等の Interaction Object で処理できない場合、Dialogue Control がその処理を行う。

Application Interfaces

この部分を通して、Presentation Control や Dialogue Control が実際の Application Components 群に処理を依頼する。

2.4　概念モデルの動作

基本的な動作としては、まず直接、ユーザのイベントを受け取るのは Presentation Component であり、受け取ったイベントを対応する Presentation Control に送付する。イベントを受け取った Presentation Control はそのイベントの処理を行う。イベントの処理の結果または途中で Interaction Object の範囲内だけで処理できない場合や Interaction Object の役割が終了した場合に Dialogue Control にメッセージを送付する。Dialog Control は対話進行に際して必要な処理があれば Application Interface に処理の依頼を行う。また、Presentation Control も Interaction Object 内での処理においてアプリケーション本体に処理を依頼したい場合に、Application Interface に処理の依頼を行う。

2.5　実行可能なモデルへの拡張

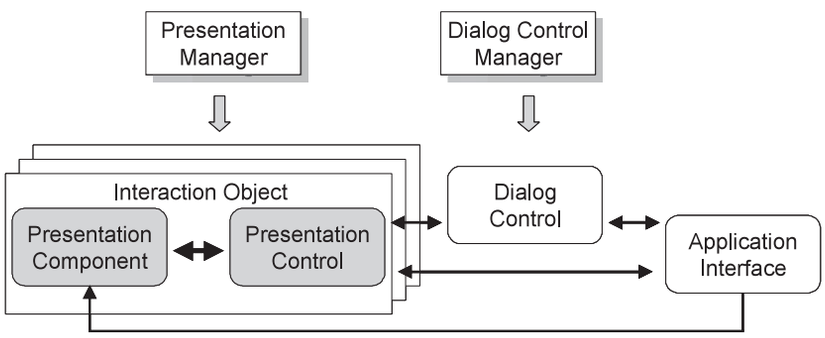

図 2：実行可能なモデル

概念モデルに存在する 4 つのコンポーネントだけでは実際に稼働することができないので、これらのコンポーネントの生成、管理を行うコンポーネントである Presentation Manager 及び Dialogue Control Manager を追加した。

基本的な各 Manager の役割は以下の 4 つである。

(1) 各コンポーネントの生成・消去

(2) 通信の仲介

(3) 適応の実現

(4) 整合性の管理

「各コンポーネントの生成・消去」については、全ての Presentation Component、Presentation Control 及び Dialogue Control は各 Manager によって管理されており、その生成及び消去を各 Manager がおこなう。

「通信の仲介」については各オブジェクトのインスタンスが通信相手を知らないためそ

の仲介をして通信を行わせるものである。

「適応の実現」と「整合性の管理」は適応のための機能である。「適応の実現」については適応する際に変更する部分は各 Manager によってその置き換えが行われる。「整合性の管理」に関しては、「各コンポーネントの生成・消去」、「適応の実現」の役割によって全ての Presentation Component、Presentation Control 及び Dialogue Control の管理を各 Manager がおこなうため、適応による変更に対する整合性のチェックを各 Manager が行う。

2.6 モデルの特徴

本モデルの最大の特徴は前述のように Presentation Control と Dialogue Control という 2 つの制御部分を持つことである。このことにより以下の利点をもつ。

(1) Presentation Control が存在することによって、Event の処理などの頻繁に行われる通信は Presentation Component と Presentation Control の間で行われるため、Seeheim モデルの持つ「各部分が頻繁にコミュニケーションを行う必要がある直接操作には向かない」という問題に対処している。
(2) 各部分の通信を記録することで分類された形で操作履歴を記録することができる。具体的にはユーザの行動をアクション、メソッド、ユーザ操作要素の 3 段階のレベル [7] で捉えて、履歴を記録する。本モデルでは 2 つの制御部分及び Presentation Components からそれに近い形で操作履歴を取得できる。
(3) 独立性が高いため個々の部分が単独で変更可能である。それによりユーザインタフェースの様々なレベルでの単独での動的変更が可能となる

3. 2 つの異なるレベルのユーザ操作履歴

3.1 従来の操作履歴獲得手法

従来の操作履歴は一般的には、システムが生成するイベントをフックして、その情報を元に履歴を記録するものが多く、分析を工夫することによって利用している。来住はユーザモデルを Prolog で記述し、操作履歴が想定した使い方と一致するかどうかで分析を行っている [8]。また岡田らは操作履歴をもとに、共通対話パターンを抽出し、グラフ表現することによって GUI の評価を行うツールを提案している [9]。これらの研究からもわかるように操作履歴からユーザの行った意味を再構築することはかなり難しく、手間のかかる作業となる。また分析できる範囲も非常に限られたものになる。

3.2　センサーによる操作履歴の取得

図３のようにシステムアーキテクチャモデルの各コンポーネントに相当するオブジェクト同士の通信を通信機構にセンサーを埋め込むことによってフックし、メッセージの内容を記録することが可能である。

またセンサーを個々のオブジェクトに埋め込むために必要のないオブジェクトのメッセージは取得しなくても良い。

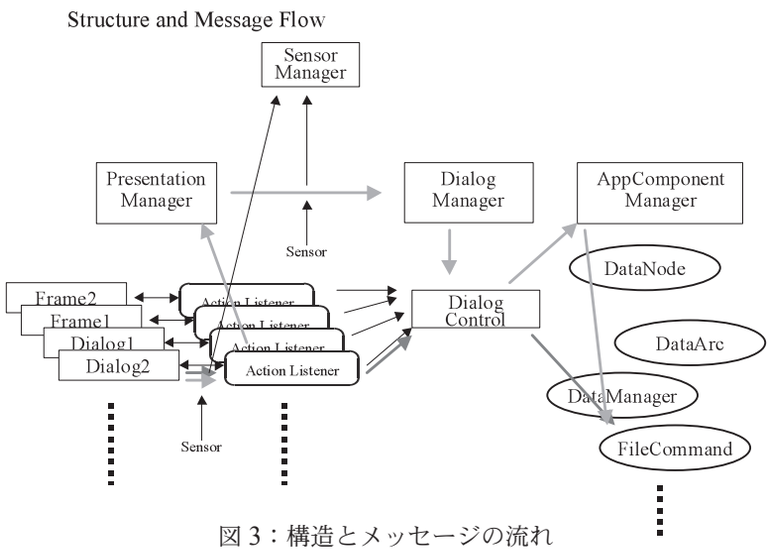

図３：構造とメッセージの流れ

3.3　ユーザイベントレベルでの履歴

Presentation Component と Presentation Control 間の通信は Presentation Control がユーザイベントレベルでの制御を行っているため、ユーザイベントレベルでの通信となる。そのためここにセンサーを埋め込むことによってユーザイベントレベルでの履歴を記録することが可能となる。

この履歴は従来の通常のソフトウェアでユーザイベントをフックすることによって得られた履歴とほとんど同じものである。

3.4　メッセージレベルでの履歴

Presentation Control と Dialogue Control 間の通信は Dialogue Control が疑似タスクであるメッセージレベルでの制御を行っているため、同様にメッセージレベルでの履歴を記録することが可能となる。

この履歴は Presentation Components のインスタンス間およびその変更を伴うレベルでの対話を制御した際の制御用メッセージである。具体的には Interaction Object の切り替えや、１つの Window、Frame 等の Interaction Object で処理できない場合に制御のため

に Dialogue Control に送られる通信や Application Components から Application Interfaces を通して Dialogue Control に送られる通信を記録したものである。

3.5 メッセージの分類

我々はタスクを以下のように人間プロセスとソフトウェアプロセスから構成されるものと捉えている。

Task ＝ Human Process ＋ Software Process

前節で述べた履歴に記録されるメッセージはこの定義に沿って大きく 2 つに分類することができる。1 つは Human Process のうちソフトウェアを用いて行うものの実現をサポートするためのメッセージであり、もう 1 つは Software Process を実現するためのメッセージである。前者は Interaction Object をユーザが操作することが直接の要因となったメッセージであり、後者はソフトウェアの機能を実現するための内部処理をおこなうためのメッセージである。前者のメッセージを User Driven Message (UDM)、後者のメッセージを System Driven Message（SDM）と呼ぶこととする。

4. Object Designer

4.1 Object Designer の概要

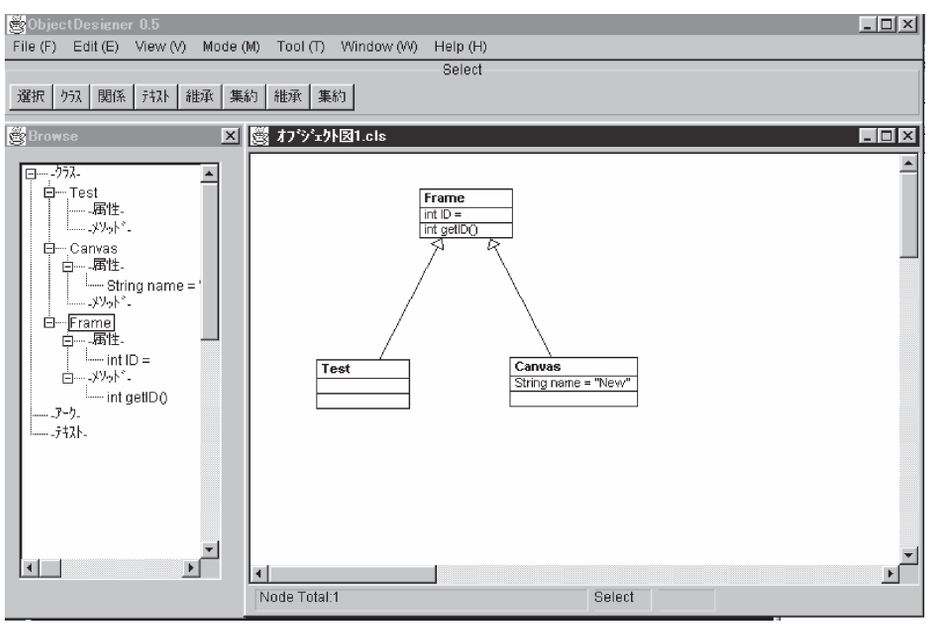

図 4：Object Designer

Object Designer（図 4）は 2 章で提案したシステム概念モデルに基づいて作成したオブ

ジェクトモデリングツールである。

本ソフトウェアは Java 1.1 を用いて作成されている。このソフトウェアには前述の操作履歴獲得機構が組み込んであり、操作履歴ファイルを出力することができる。このソフトウェアの作成目的は 2 種類の操作履歴の獲得とその操作履歴の有用性の検証および操作履歴を利用した応用研究をサポートすることにある。

4.2　Object Designer の構造

図 3 に示したのが実際の Object Designer の構造である。Java 1.1 の Delegation Event Model を用いて Action Listener に送られるイベントを Presentation Component と Presentation Control 間の通信とみなしている。またそれ以外の個々のコンポーネント間のメッセージの通信には専用の通信機構を用意し、各 Manager が Broker として通信の仲介をして Dialogue Control にメッセージを送り、その後 Dialogue Control がその制御ルールに従って、メッセージを受信可能なオブジェクトを管理する Manager にメッセージを送る。受け取った Manager は管理下にある受信可能なオブジェクトに対してメッセージを送るというメッセージ伝達機構を作成し利用している。

4.3　Object Designer の動作

4.4　履歴のデータ構造

Object Designer は以下の 2 種類の直接取得したデータを履歴として残すことができる。

（1）ユーザイベントレベル

　　Time, Event, Location, Key Modifiers, Click Count

（2）メッセージレベル

　　Time, Message ID (, Source)

（1）の項目値に関しては、Time の他は現在のところ Java のイベントクラスに依存した情報である。（2）の項目値に関しては現在は Time とメッセージの内容を区別するための ID だけであるが、今後のためにメッセージの発信元を示す Source フィールドが用意されている。

5.　Object Designer を用いた実験

5.1　実験の目的

本実験の主目的は提案する機構により得られたデータが有用なものであるかどうか検証

することである。また実験により同時にシステムアーキテクチャモデルが有効であるかどうか検証することできた。

5.2 実験の概要

本実験では簡単なオブジェクト図の清書作業を Object Designer を用いて 8 名の被験者に行ってもらった。実験の際には簡単なリファレンスマニュアルを見てもらうこととした。

5.3 実際の結果

実験の際に得られたデータは以下の 3 種類である。

(1) ユーザ操作履歴

(2) タスク履歴

(3) オブジェクト図のデータ

	A	B	C	F
1	TIME	DATA	経過時間	Message
2	885810772240	1111	0.0	SETCONFIGUSERNAME
3	885810772730	1116	0.5	GETCONFIGPAPERSIZE
4	885810803220	MOUSE_RELEASED,(206,4),mods=16,clickCount=1	31.0	#N/A
5	885810803330	100	31.1	SELECTMODE
6	885810803330	MOUSE_CLICKED,(206,4),mods=16,clickCount=1	31.1	#N/A
7	885810873520	MOUSE_RELEASED,(201,109),mods=16,clickCount=1	101.3	#N/A
8	885810876380	101	104.1	NODEMODE
9	885810876430	Node	104.2	#N/A
10	885810880720	MOUSE_RELEASED,(177,121),mods=16,clickCount=1	108.5	#N/A
11	885810895710	101	123.5	NODEMODE
12	885810895710	Node	123.5	#N/A
13	885810896370	101	124.1	NODEMODE
14	885810896370	Node	124.1	#N/A
15	885810898350	MOUSE_RELEASED,(81,47),mods=16,clickCount=1	126.1	#N/A
16	885810898400	1	126.2	#N/A
17	885810898400	2	126.2	GETVIEWARCS
18	885810898400	10	126.2	ADDDATANODE
19	885810898400	1118	126.2	GETCONFIGWIZARD
20	885810899230	30	127.0	GETONENODE
21	885810899340	800	127.1	CLASSWIZARD
22	885810899340	100	127.1	SELECTMODE
23	885810899340	MOUSE_CLICKED,(81,47),mods=16,clickCount=1	127.1	#N/A
24	885810899390		127.2	#N/A
25	885810899390	MOUSE_RELEASED,(0,0),mods=16,clickCount=2	127.2	#N/A
26	885810899560	30	127.3	GETONENODE
27	885810899830	300	127.6	CLASSDIALOG
28	885810909990	301	137.8	CDDESTROY

図 5：獲得したユーザ操作履歴

6. 有用性の検証

6.1 概略

2.6 のモデルの特長で述べたように提案したシステム概念モデルは 2 つの大きな特徴を持つ。そのうち（1）の「Seeheim モデルの持つ『各部分が頻繁にコミュニケーションを行う必要がある直接操作には向かない』という問題に対処している」、ことについては各

コンポーネント間でやり取りされる通信を分析することによって示すことができる。また (2) の「2 つの対話制御部の情報から異なるレベルの操作履歴を獲得できる」ことに対するメリットについてはデータの有用性の検証によってその有用性が示す。

6.2　データの有用性の検証

以下の項目では Object Designer で得られた履歴データからどのような分析が可能かをまず示し、その次にその分析を実際の実験データに当てはめるとどうなるかを示すことによって、それらの分析が可能であることを示す。またその分析結果の利用方法についても言及する。

UserDrivenMessage

		A	B	C	D	E	F	G	H
*	ARCDIALOG	20	2	2	1	17	29	14	0
*	CLASSDIALOG	14	7	12	19	10	4	4	19
*	CLASSWIZARD	4	7	2	12	23	10	9	5
*	CLASSWIZARD1	4	7	2	8	21	9	9	4
*	CLASSWIZARD2	6	7	2	9	22	9	9	4
*	CLASSWIZARD3	9	7	2	8	26	9	9	4
*	CLASSWIZARD4	7	7	2	7	24	9	9	4
*	CONFIGDIALOG	0	0	0	0	1	0	0	0
*									
*	COPYACTION	3	0	1	0	1	0	0	0
*	CPACTION	1	0	0	0	6	0	0	0
*	DELETEACTION	0	1	4	3	15	4	1	0
*	PASTEACTION	1	0	0	0	2	0	0	0
*									
*	LINK2MODE	8	3	3	8	5	5	2	9
*	LINK3MODE	5	4	5	5	16	12	0	0
*	LINK4MODE	1	0	2	3	0	2	0	0
*	LINK5MODE	0	0	0	1	0	2	0	0
*	LINKMODE	11	2	6	10	6	18	14	0
*	NODEMODE	6	10	17	16	34	11	11	0
*	SELECTMODE	139	52	67	58	359	227	110	22
*	TEXTMODE	0	0	4	1	0	0	1	0
*									
	PRINTALL	1	0	0	1	1	1	0	0
	SAVE	1	2	1	1	2	1	0	3
	SAVEFILEDIALOG	1	2	1	1	2	1	0	3
	(空白)	0	0	0	0	0	0	0	0
	総計	4197	2597	2118	1677	8618	3263	1902	552
	UserDriven	242	120	135	172	593	363	202	77
	SystemDriven	301	95	138	191	361	177	151	116

SystemDrivenMessage

		A	B	C	D	E	F	G	H
0	ADDDATAARC	20	7	10	15	19	15	9	0
0	ADDDATANODE	15	7	9	12	25	10	9	5
0	DELETEDATAARC	7	1	4	0	9	0	1	0
0	DELETEDATANODE	0	0	2	0	13	0	0	0
0	UPDATEDARC	18	2	2	1	15	9	10	0
0	UPDATEDNODE	15	8	13	21	25	11	13	18
0	UPDATEVARC	18	2	2	1	15	9	10	0
0	UPDATEVNODE	15	8	13	21	25	11	13	18
0	UPDATETREE	11	0	0	0	2	0	0	0
0	SETCONFIGUSERNAME	1	1	1	1	1	1	1	1
0	SETCONFIGWIZARD	0	0	1	0	0	0	0	0
0	BUFFERNODES	12	0	0	0	6	0	0	0
0	PASTENODES	11	0	0	0	2	0	0	0
0	CDDESTROY	14	7	12	19	10	4	4	18
0	GETCONFIGPAPERSIZE	2	1	1	2	2	2	1	1
0	GETCONFIGUSERNAME	1	1	1	1	2	2	2	5
0	GETCONFIGWIZARD	4	7	10	12	24	10	9	0
0	GETONEARC	20	2	2	1	17	29	14	0
0	GETONENODE	18	14	14	31	33	14	13	24
0	GETVIEWARCS	99	27	41	53	116	50	42	26
0	#N/A	3654	2382	1845	1314	7664	2723	1549	359
	DRAG	2675	1892	1204	747	5299	1693	966	63
	Mes Total	543	215	273	363	954	540	353	193
	Event-DRAG	979	490	641	567	2365	1030	583	296

図 6：ユーザ操作履歴の集計表

6.2.1　回帰直線による異常値の検出と原因の追及

ユーザ操作履歴によって得られたデータのうち図 6 のような各タイプの通信の総計などを用いて相関分析を行うことによって異常値を検出し、その原因を追及することによってユーザの特徴を把握することができたり、ユーザインタフェースの改善を行うことができたりする。

例えば図 6 のデータについて標準残差をプロットしたのが図 7 である。

この標準残差をみると、標準残差が－ 2 付近の値が見受けられる。この値は異常値であると言えるので、この原因を見るために被験者のユーザイベントレベルの内訳を分析する。そこでこの被験者が他の被験者よりマウスをドラッグした割合が高いと言うことが判明し、その理由を探っていくことによって原因を追及することができる。

またある程度典型的な原因が究明できた分析については、その回帰直線をシステムに組み込むことによってナビゲーションや適応型ユーザインタフェースを実現することも可能である。

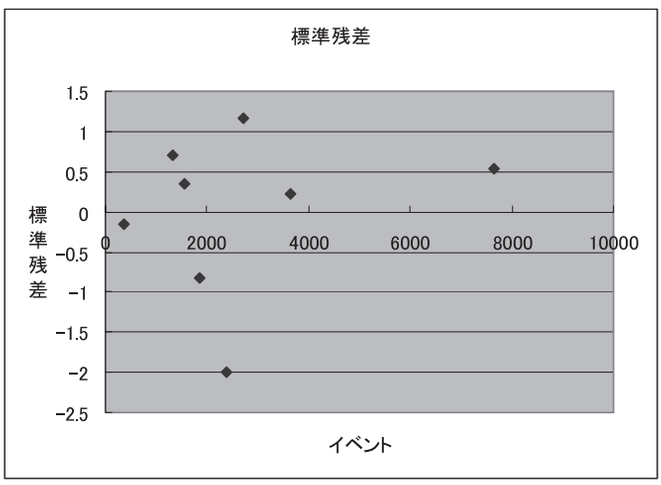

図 7：イベントとメッセージの関係の標準残差

6.2.2　直間比率による効率性

標準残差が 0 から離れた値があった場合、正の場合だと UDM によってポストされる SDM の割合が高いので、ユーザの操作がシステムの状態の変化に及ぼす割合が高いと言える。換言すれば少ない操作数でタスクの遂行を行っている可能性があると言える。また逆に負の値の場合には操作がタスクの遂行に結びついていない可能性があると言える。

図 8 は実験データを元にいくつかのこの分析には不必要なメッセージを除いた User Driven Message を横軸に同様の処理をした System Driven Message を縦軸にとった図である。

実線で描かれた直線はこのデータに対する最小二乗法による回帰直線であり、破線は y 切片を 0 としたときの回帰直線である。実線の方は

$y = 1.1616\,x + 48.683$

で表すことができ、その相関係数も 0.8431 でこのときの 1％有意水準が 0.834 であるから有意であるといえる。しかしながら、y 切片である 48.683 を理論的に説明できない。そこで理論的には User Driven Message があってはじめて、System Driven Message がポストされるので y 切片を 0 として破線の方の回帰直線を求めた。この直線は

$y = 1.666x$

で表され、相関係数は 0.7318 となる。これは 5％有意水準が 0.707 であるから 5％有意であるといえる。このことから User Driven Message と System Driven Message の関係

は 1：1.666 であるといえる。

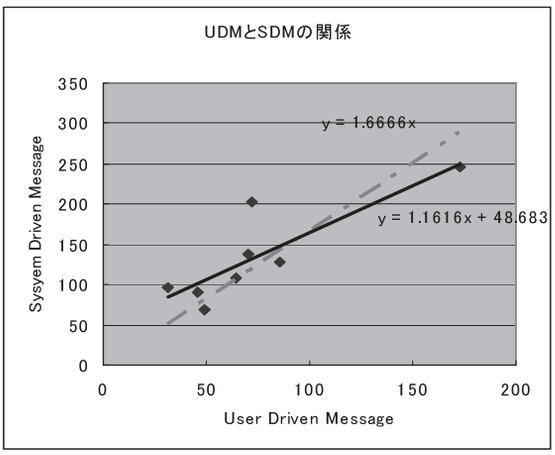

図 8：UDM と SDM の関係

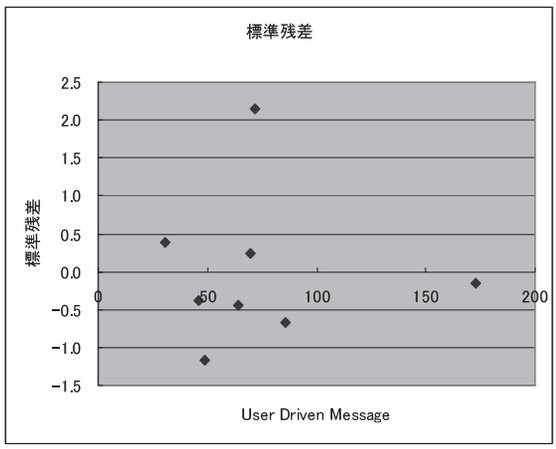

図 9：UDM と SDM の標準残差

そこで、標準残差をプロットしたのが図 9 である。

図 9 を見ると標準残差が 2 を超える値や－1 を下回る値が見受けられる。これらの被験者についてデータを詳しく見ていくと 2 を越える被験者は Copy & Paste を多用し、操作を省略していることが分かる。この被験者の効率性は良いといえる。また－1 を下回る被験者はウィザードを使った操作の割合が高いことがわかる。この操作は UDM が 4 に対して SDM が 1 しか起こらないもので、このことから効率性が悪いとは言えないが、この部分にユーザインタフェースもしくはユーザの操作などに問題がある可能性は高い。

6.2.3　2 種類のメッセージの比較による失敗操作の検出

前節では UDM 全体と SDM 全体との比率で効率性について検討したが、UDM の一部とそれによってポストされる SDM の比較をすることによって検討することもできる。例えば Copy や Paste を行うための操作による UDM とその UDM によってポストされる SDM を見ることによって、Copy の操作はやっているのに実際には Copy されていない。Paste の操作はやっているのに実際には Paste されていない。などの状態を把握することができる。前者の原因としては Copy する対象を選択していない。後者の原因としては Paste の前に Copy の作業を行っていない。等が考えられる。それに対してナビゲーションなどの対策をとることが可能である。

6.3　システム概念モデルの検証

図 10 は各被験者が実験を行った際におこなわれたコンポーネント間の通信を横軸にユーザイベントレベルの通信数を、横軸にメッセージレベルの通信数をとって散布図を描

いたものである。

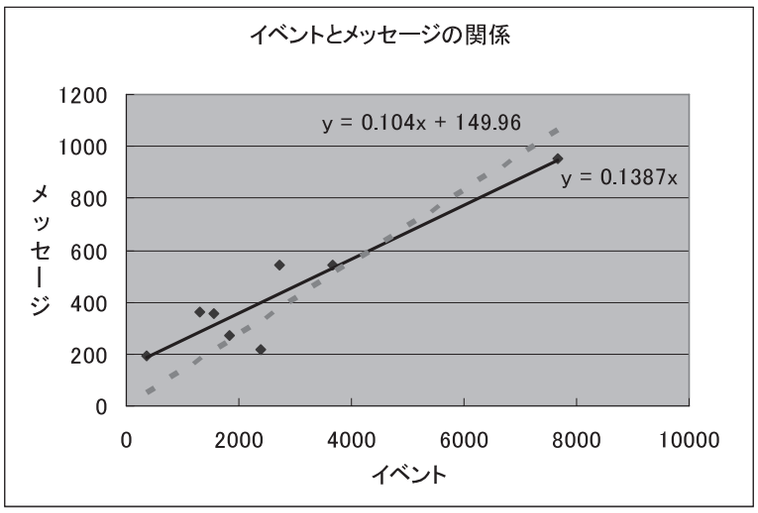

図 10：イベントとメッセージの関係

実線で描かれた直線はこのデータに対する最小二乗法による回帰直線であり、破線は y 切片を 0 としたときの回帰直線である。実線の方は

$y = 0.1404\,x + 149.96$

で表すことが出来き、その相関係数も 0.9320 でこのときの 1％有意水準が 0.834 であるから有意であるといえる。しかしながら、y 切片である 149.96 を理論的に説明できない。そこで理論的には Object Designer を操作せずに使用することのみで生じるメッセージレベルでの通信は考えられないので、y 切片を 0 として破線の方の回帰直線を求めた。この直線は

$y = 0.1387x$

で表され、相関係数は 0.8444 となる。これも 1％有意であるといえる。

このことから、ユーザイベントレベルでの通信とメッセージレベルでの通信の比率は 1:0.1384 つまり、メッセージレベルでの通信 1 に対して約 7.2 倍のユーザイベントレベルでの通信が行われていることとなる。このことより Seeheim モデルのようなモデルによって作成した場合には、Dialogue Control に現在の 7 倍以上の負担がかかり、各部分が Token レベルでのやり取りしか想定していない場合、遅延を起こす原因となりうる。

つまり今回提案したモデルの特長の 1 つである。Seeheim モデルの持つ「各部分が頻繁にコミュニケーションを行う必要がある直接操作には向かない」という問題に対処している。ということが示されたと言える。

6.4　Object Designer を用いた応用研究

以下の様な研究が Object Designer を利用した応用研究として早稲田大学理工学部経営システム工学科東研究室で研究されている。

(1) ソフトウェア学習支援システムの研究

特定のグループ内においてはメンバー間に共通したタスクがあるため、共通の「学習すべきソフトウェアの操作方法」が存在する。したがって、メンバー間で操作方法に関する知識を共有することは業務上有効である。ソフトウェアの操作方法に関する知識共有の現状と問題点について分析し、その問題点を解決するシステムを提案し、その検証実験を行った。

(2) タスク実施支援システムの研究[10]

あるタスクを実施しているユーザの操作履歴を取得、分析することによってその実施に合わせた適切な情報を提示し、ナビゲーションを行う方法を提案し実装した。

(3) タスク実施のための最適プロセス共有化の研究

グループ内での効率の良いタスク実施プロセスを共有化することで、ユーザの効率的なタスク実施を支援する方法を提案する。そのために本研究では、タスク実施プロセスを実際に構築してシステムが認識する為のプロセス認識技法を提案し、その技法に基づいた支援システムを実装した。

7. おわりに

実験での分析結果や他の応用研究の結果からこのシステムによって得られた履歴は、イベントのみで得られる履歴に対してより多くの情報を提供し、タスクの特定を容易にしたと言える。また 2 タイプのメッセージを比較することにより今まで得られなかった失敗操作なども取得可能となった。また、現在は Interaction Object 内だけで処理できるものに関してはメッセージを送らないことになっているが、今後は、自分自身に対して処理依頼メッセージを送るように変更したほうがより正確なデータがとれると考えられる。

これらのことから本研究は一定の成果をあげたといえる。しかしながら本システムは JAVA を用いているとはいえ、本システムを開発した時とは違い、現在ではアプリケーションのフロントエンドは Web のブラウザを利用したものも多くなっている。今後はそういったより幅広く利用される環境下で使えるように考える必要があると思える。本研究の成果を生かしながら新たな展開を模索していきたい。

参考文献

[1] 東基衞，野中誠，木村泰己，長崎等，“分散・適応型システム実現のフレームワークと目標”，『情報処理学会第 53 回全国大会講演論文集』，情報処理学会，1996 年，pp.4-251-252

[2] Norman, D. A., Draper, S. W., Eds. User Centered System Design, Lawrence Erlbaum, 1986 年

[3] Card S. K. et al., The Psychology of Human Computer Interaction, Lawrence Erlbaum, 1983 年

[4] 長崎等，東基衞，“適応型ユーザインタフェースを実現させるためのシステムアーキテクチャ”，『情報処理学会第 53 回全国大会講演論文集』，情報処理学会，1996 年，pp.5-181-182

[5] 長崎等，東基衞，“情報システムにおけるユーザインタフェースの適応を実現させるためのユーザインタフェースモデル”，『日本経営システム学会誌　Vol.16 No.2』，日本経営システム学会，2000 年

[6] Green, M.，“A Survey of Three Dialog Model”, ACM Transaction on Graphics 5(3), ACM, 1986 年，pp.244-275

[7] Nagasaki, H., Azuma, M., AMethodology for Assessing User’s Skill Grade to Implement Adaptive User Interface Systems”, Proc. of the First IEEE International Conference on Cognitive Informatics (ICCI’02), IEEE Computer Society, 2002, pp.280-287

[8] 来住伸子，“GOMS モデルを利用した入出力記録自動解析の試み”，『情報処理学会ヒューマンインタフェース研究会報告 64-5』，情報処理学会，1996 年，pp.25-30

[9] 岡田秀彦，旭敏之，井関治，“使いやすさ評価ツール「GUI テスタ」の提案”，『情報処理学会ヒューマンインタフェース研究会報告 59-13』，情報処理学会，1995 年，pp.87-94

[10] 山田季史，福原綾介，長崎等，東基衞，“タスク実施支援システムの研究”，『情報処理学会第 56 回全国大会講演論文集』，情報処理学会，1998 年，pp.4-71-72