ヒューマンインタフェース 96－9
モバイルコンピューティングと
ワイヤレス通信 19－9
（2001．11．16）

インタラクション記述言語 CHIOPA
— デザイラブルなシステムの構築に向けて —

神谷年洋[*1],中小路久美代[*1*2*3],高嶋章雄[*1*2]
*1: 科学技術振興事業団さきがけ 21（ポスドク活用型）
*2: 奈良先端科学技術大学院大学 情報科学研究科
*3: (株)SRA 先端技術研究所
e-mail: {kamiya, kumiyo, akio-ta}@is.aist-nara.ac.jp

概要

日常業務における知的創造活動を支援する対話的アプリケーションシステムには，機能を充足する必要と共に，使い心地の良さ(デザイラビリティ)が求められる．本稿では，デザイラブルなシステムの構築に向けて，システムのインタラクションの分析やデザインを目的とした，インタラクション記述言語 CHIOPA を提案する．提案した言語を用いて粒度の異なる 2 つのインタラクションを記述した例を用いて，アプローチの特徴を述べる．

CHIOPA: A Language for Human-Computer Interactions
— Toward Desirable Computer Systems —

Toshihiro Kamiya[*1], Kumiyo Nakakouji[*1*2*3], Akio Takashima[*1*2]
*1: PRESTO, Japan Science and Technology Corp.
*2: Grad. School of Information Science, naist-ccc
*3: SRA Key Technology Laboratory Inc.

Abstract

Interactive systems that support intellectual creative tasks require not only to be *useful* and *usable* by satisfying their targeted functionalities, but also to be *desirable* for individual users by providing appropriate interaction styles. This paper proposes a descriptive formal language, CHIOPA (Computer-Human Interaction on Objects, Properties, and Actions), with which developers can analyze and design interaction between human and interactive computer systems. We illustrate characteristics of the language by using two examples of interactions at two different levels of granularity.

1. はじめに

文章作成やコミュニケーション，情報検索といった我々の日常業務における多くの知的創造活動が，コンピュータツールを用いて行なわれるようになった．人間はこれらのシステムと対話的にやりとりをしながら，目的とするタスクを遂行する．従来これらのシステムには，利用者が目的としているタスクを適切に行えるよう支援すること，すなわち「機能的な要求を充足し useful かつ usable であること」が必要とされてきた[8]．我々はこれに加えて，システムを使う個々の人間にとって「デザイラブル(desirable; 使い心地がよい)」であることも重要な要件であると主張してきた[9]．

システムが知的創造活動を支援するための認知ツールとして作用するためには，人間の思考を阻害したりまた思考のパタンの変更を強要したりするものであってはならない．個々人やタスクの状況に合った，デザイラブルなインタラクションを提供するシステムが望まれる．

デザイラブルなインタラクションを考慮するためには，人間(利用者)とシステムとが，そのシステムが備える UI(ユーザーインターフェイス)を通してインタラクションを行う系としてモデル化する必要がある．その上で，システムあるいはシステムが提供する表層的な UI だけではなく，利用者をも含めたインタラクションの表現が必要である．そして表現されたインタラクションを比較，評価，合成，検証するためにはそれらのインタラクションの表現が何らかの形式性を有していることが必要となる．

本稿では，デザイラビリティに関する議論を目的として，インタラクションの分析，デザインなどを可能にする，インタラクション記述言語 CHIOPA を提案する．CHIOPA 言語によって，利用者が記述されたインタラクションを選択したり，実装された UI が特定のインタラクションに適合しているかを判定したりすることが可能となる．

2. 機能充足性からデザイラビリティへ

対話的コンピュータシステムを機能充足性の観点から評価する場合には，UI はシステムが果たす機能との関連において議論されてきた．例えば，UI が必要な情報を利用者に提示しているか，UI が情報を提示するタイミングはシステムが果たすべき仕事に必要とされる時間的制約を満足しているか，UI の個々の状態は到達可能か，といった論点である．

これに対し，対話的コンピュータシステムをデザイラビリティの観点から評価する場合には，UI をシステムと人間(利用者)との間のインタラクションに関して議論することが必要である．すなわち，タスクを遂行するにあたり必要となる機能が提供されていることのみではなく，個々の利用者が「好ましいと思う」やりとりを経てそのタスクを遂行できるか否かを議論する必要がある．本稿では UI を比較的静的なオブジェクトの並び(e.g. メニューの配置)や機能指向のインタフェースウィジェット(e.g. ダイアログボックス)を指すものとする．これに対しインタラクションは，一般に時間的幅を伴う，人間にとって意味のある人間とシステムとの一連のやりとりを指すものとし UI と区別する．

例えば，HCI デザインにおいて重要とされている直接操作性(Direct Manipulation) という概念を例にとる．Shneiderman[12] によれば，以下の 3 つの基準を満たすとき，その UI は直接操作性を提供しているという．

- 注目しているオブジェクトの連続的な表現であること
- 複雑な文法ではなく，物理的なアクションやラベルつきボタンの押し下げであること
- 迅速な，インクリメンタルな，元に戻すことが可能な操作であり，注目しているオブジェクトに対する効果が即座に見えること

ある UI が直接操作性を備えているかどうかを判断するためには，「利用者の操作とそれに対するシステムの反応が，時間的制約を満たし，かつ，オブジェクトの変化に連続性を持ちながら，繰り返し起きる」ことを記述できる必要がある．

ところが，これらの概念を実際に実装しようとすると，どのオブジェクトとどの操作に直接性をマッピングさせるべきなのか，どの粒度でインクリメンタルとするのか，どんなフィードバックを返せば利用者が直接性を認識するのかといったいくつものデザインの選択肢から一つを選択する必要がある．どの選択肢を選択すればデザイラブルなシステムになるのかは，個々の利用者やタスクの状況に依存する．例えばアイコンをドラッグ・アンド・ドロップで移動させる際に，移動の軌跡の表現にも様々な形態がある(図 8 参照)．利用者のマウス移動操作に対してシステムがどのように反応を返すことをその利用者が好ましいと思うか，これをデザイラブルなインタラクションと呼ぶ．どのインタラクションを利用者が好むのかを比較，検証するためには，こういったインタラクションを表現する必要がある．

User Action	Feedback	System state
Click on (icon) (Apply process)	Highlight icon	Access application

	if (process is being made)
	{
	if (time < S)
	{ show process is finished }
	elsif ((time > M) or (unknown_time))
	{ show process is stopped }
	else { show file size, speed, time }
	}
	else { process is being hung }
		Process is made
		
		Process is finished
		
		Process is stopped
		Processing
		
		Process is hung

図 1 XUAN のプログレス表示記述例

3. インタラクションの表現

人間と対話的コンピュータシステムとのインタラクションを表現する方法には，大きく分けて，(1)例示によるもの，すなわち，インタラクションのインスタンスの集合によるものと，(2)形式的に一般化することによるもの，という2種類が考えられる．

前者のアプローチには，例えば，ビデオによって人間がコンピュータシステムを使用しているところを撮影する，といった方法がある．後者のアプローチには，オブジェクトやプロパティ，関係によって，人間がコンピュータシステムを使用している状況を記述するものがある．本研究は後者のアプローチを採る．

後者のアプローチの利点は，表現を操作することが可能になるという点にある．これにより，インタラクションの分析において，例えば 2 つのコンピュータシステム間のインタラクションの違いを「計算」したり，あるいは，あるコンピュータシステムが一貫したインタラクションを持っているかを検証したりできる．曖昧さや冗長性を調べる際にも表現を用いることができる．あるコンピュータシステムが曖昧な UI を持ち，特定の操作系列が 2 つ以上の解釈を持つ場合，同じ操作系列に対する異なる反応を持つようなインタラクション表現が存在することになる．あるコンピュータシステムが同じ機能を呼び出す 2 種類の操作を持つ場合には，そのインタラクションの表現も 2 通り存在することになる．

また，インタラクションの合成においては，インタラクションの記述から，必要とされる UI を特定したり，あるいは，表現を操作することでインタラクションを変形したりすることができる．

既存の UI やインタラクションの表現のための表記法には，機能的な要求の充足の議論を目的としているものが多い．

HighTime，SMIL[14] は，ハイパーメディアを実装するための SGML や XML をベースとした言語であり，記述される内容は基本的に，「コンテンツを再生するタイミング」である．メディアを利用する人間の動作は記述されないため，インタラクションの記述はできない．文献[7] では，Temporal Logic をインタラクションの記述に用いているが，「利用者の操作に対して，システムは何秒以内に機能しなければならない」というシステムの機能的な制約条件を書くことを目的としている．

UAN[3]，XUAN[2]，PUAN[1]，は，システムとそれを利用する人間のインタラクションを記述する記法である．記述の中身は，「利用者が何かをすると，システムはこれこれのフィードバックを返し，システムの状態はこのように変化する」というものである．例えば，図 1 は，文献[1] から引用した，XUAN を用いてプロセスの進捗を表示する例である．システムの「利用者に対するフィードバック」，「機能」，「状態」が混在して記述され，純粋にインタラクションを記述することができ

```
H:SyncronizedMoveBByA :=
  [ System.focus == windowA ]
  H:MouseButtonDownOnA,
  <>,
  H:MouseDrag,/c
  (C:UpdateWindowA
  & C:UpdateWindowB);
H:MouseButtonDownOnA :=
  [ Mouse.pointer.isIn(windowA)]
  && Mouse.button.state == "up" ]
  H:{ Mouse.button.state = "down"; };
```

図 2 CHIOPA 言語による記述の例

ていない．例では，状態と表示の両方の意味で「process」が用いられている．また，プロセスに要した時間が計測できない(「unknown_time」)のは機能(実装)の詳細であるが，インタラクションの記述に紛れ込んでいる．Multiparty Grammar[11] は，人間とコンピュータのインタラクションを，文法に従って記述する記法である．時間的な制約を書くことができないため，適用範囲が限られている．

形式的仕様に基づく GUI(Graphical User Interface)開発環境 GUISE[10] においては，UI の状態遷移の到達可能性と合成が目的とされている．UI の検証が目的であるため，人間の動作は記述されない．ユーザータスクの形式的記述に基づくインタラクティブシステム設計法[4] は，UI 部品よりも抽象度の高い記述である，タスク(例えば，「本の発送先を指定する」)の形式的記述から UI を合成することを目的としている．人間の動作は，抽象度の高いタスクの記述において，システムの入力として暗黙のうちに記述される．いずれの研究においても，重点はコンピュータシステムの開発にあり，インタラクションの比較や，個々の利用者によるインタラクションの選択といった利用法は目的とされていない．インタラクションを(機能や UI を記述せずに)記述することを目的としている研究として，Multiparty Grammar[11] がある．時間的な制約を書くことができないため，適用範囲が限られている．

インタラクションを言語ではなく，状態遷移として書き下す試みもなされている[5] [6] ．状態遷移図を使ったアプローチでは，任意のアクションの系列が等しく状態遷移図中のひとつの状態として表現されるが，インタラクションとして望ましいかどうかは，別の判定手法が必要である．

これらの考察に基づき本研究では新たな言語 CHIOPA を提案する．

4. インタラクション記述言語 CHIOPA

本稿では，GUI によって利用者と対話を行うコンピュータシステムと利用者のインタラクションを記述するための CHIOPA (Computer-Human Interaction on Objects, Properties, and Actions) Language を提案する．CHIOPA 言語の前提は，以下の 3 点である．

- 人間(H)とコンピュータ(C)との間のインタラクションは，H と C が共通に知覚可能なオブジェクトおよびオブジェクトのプロパティを，H または C が変化させること(以下「アクション／発話」)の時間的系列によって表現される(表現されなければならない)．
- H と C は共同でひとつのインタラクションを成立させる．したがって，ひとつのインタラクションは H のアクションと C のアクションを

```
〈インタラクション〉 ::= 〈アクション記号〉 “:=” 〈定義式〉 “;” ;
〈アクション記号〉 ::= 〈発話者〉 “:” 〈アクションの名称〉 ;
〈定義式〉 ::= 〈繰返定義式〉 “&” 〈繰返定義式〉
  | 〈繰返定義式〉 “|” 〈繰返定義式〉 | 〈繰返定義式〉 ;
〈繰返定義式〉 ::= 〈素定義式〉 “*”
  | 〈素定義式〉 “+/d” | 〈素定義式〉 “*/d”
  | 〈素定義式〉 “,” 〈繰返定義式〉
  | 〈素定義式〉 “,/c” 〈繰返定義式〉 | 〈素定義式〉 ;
〈素定義式〉 ::= “(” 〈定義式〉 “)”
  | “<” “(” 〈時間属性条件〉 “)” 〈定義式〉 “>”
  | “<” 〈条件〉 〈定義式〉 “>”
  | 〈条件〉 〈アクション〉 | 〈アクション〉 〈条件〉
  | 〈アクション〉 ;
〈アクション〉 ::= “<>” | “null” | 〈アクション記号〉
  | 〈発話者〉 “:” 〈アクション定義〉 | 〈発話者〉 “:”
〈アクションの名称〉 〈アクション定義〉 ;
〈アクション定義〉 ::= “{” 〈属性アクション〉 “}” ;
〈条件〉 ::= “[” 〈属性条件式〉 “]” ;
〈発話者〉 ::= 〈名称〉 ;
〈アクションの名称〉 ::= 〈名称〉 ;
```

図 3 CHIOPA 言語の LR 文法部分の構文定義

```
〈時間属性条件〉 ::= 〈開始時間〉 “,” 〈終了時間〉 ;
〈属性条件式〉 ::= 〈オブジェクトのプロパティの条件式〉 ;
〈属性アクション〉 ::= 〈オブジェクトに対する操作〉 ;
オブジェクトとプロパティは，Java 言語のインスタンスとインスタンス変数の表記(“オブジェクト.プロパティ”)を借用する．
オブジェクトに対する操作は，代入やメソッド呼び出しの表記を借用する．
開始時間，終了時間は数値を値とする式の表記を借用する．
```

図 4 CHIOPA 言語の属性部分の構文定義

含む.

- 有効なインタラクション(アクションの系列)には明確なパタンが存在し，それは文法によって定義される言語として捉えることができる.

言語の設計にあたっては，コンピュータや人間の内部状態とインタラクションを混同することなく，インタラクションだけを記述できること，人間を明示的に記述することにより複数の人間が関与するインタラクションを記述可能とすること，インタラクションの抽象化や異なった粒度を扱えること，実際のインタラクションのインスタンスが記述されたインタラクションであるかを検証可能であること，などに注意を払っている．結果として，CHIOPA 言語は，WIMP(ウィンドウ，アイコン，メニュー，マウスポインタ)などのオブジェクトに対するアクションの系列でインタラクションを記述する，属性文法(LR-attributed grammar)を基本として，時間的制約や並列性を表現するための拡張を行ったものとなった.

図 2 は CHIOPA 言語の記述例である．この例では，ウィンドウ A をドラッグ&ドロップで移動させると，ウィンドウ B も同時に移動する，というインタラクションを表現している．人間とコンピュータのアクションは，「H:」や「C:」をアクションの前につけることで区別されている．H: SyncronizedMoveBByA がこの記述で定義されるインタラクションであり，「:=」の右辺がその内容を表している．この右辺で用いられているアクション H: MouseButtonDownOnA は，2 つ目の定義式で定義され，このようにして，CHIOPA 言語は段階的な詳細化を意図した構文規則となっている.

4.1 文法

CHIOPA 言語は「属性」つき「LR 文法」を用い，LR 文法部分でアクション，アクションの並列性，アクションの間の経過時間を表現し，文法の属性部分でアクションの時間的制約やアクションとオブジェクトの関係(どのアクションがどのオブジェクトやどのプロパティを参照したり変更したりするか)を表現するとしている．「属性」に関する記述は，<時間属性条件>，<属性条件式>，<属性アクション>の 3 つの構文の中に現れる.

CHIOPA 言語の構文定義を図 3，図 4 に示す．終端記号となる，演算子およびアクションの説明を，それぞれ，表 1，表 2 に示す.

4.2 記述例 プログレスダイアログ

演算子を説明するための例として，プログレスダイアログの例を説明する．図 5 はプログレスダイアログの画面上の表示を示す．画面上に見えているオブジェクト dlg(ダイアログ自体)，

表 1 CHIOPA 言語の演算子

演算子	説明
A & B	A と B が共に起きる(時間的な前後は問わない)
A | B	A か B のいずれかが起きる
A *	A が 0 回以上繰り返される
A +	A が1回以上繰り返される
A , B	A が起きた後「即座」に B が起きる．「即座」とは，一般に 100 ミリ秒以内[13]
A ,/c B	A が起きた後「即座」に B が起き，A と B の間に一貫した因果関係がある.
<(M, N)A>	M 以上 N 以内の経過時間の内に A が起きる
<[C] A>	A が起きる．その間じゅう，C が満たされている
[C] A	C を満たしている時に A が起きる(前条件)
A [C]	A が起き，完了した時点で C が満たされる(後条件)

表 2 CHIOPA 言語のアクション

アクション	説明
s : a	発話者 s が a という名称のアクションを行う
s : { … }	発話者 s が…で記述されるオブジェクトの変更を行う
s : a { … }	発話者 s が…で記述されるオブジェクトの変更を行う．このアクションの名称を a とする
<>	0 以上任意時間の経過を表す
null	何もせず，時間経過もなし

dlg.cancelBtn(ダイアログ上のキャンセルボタン), dlg.progress(ダイアログ上のプログレスバー)を基にして, インタラクションは以下のように記述される.

図 6 は以下のようなインタラクションを記述している.

1. C(コンピュータ)がプログレスダイアログを生成し(行 2),
2. キャンセルボタンが押されないまま進捗が既定の値(D)に達する(行3～11)か,
3. あるいは, 途中で H(人間)がキャンセルボタンを押す(行 12～16)と,
4. ダイアログを閉じる(行 16).

この例では, 演算子「|」と「*」を用いて, アクションの選択と繰り返しが記述されている.

インタラクションは, 人間(利用者)に見えるオブジェクトである, に対するアクションとして, システムの機能に言及することなく, 表現されている.

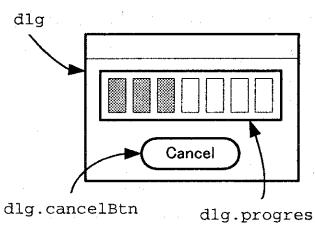

図 5 プログレスダイアログを構成する 3 つのオブジェクト

```
1  C:ModalProgressDlg :=
2  C:OpenProgressDlg,
3  (
4    < [ dlg.cancelBtn.state == "up" ]
5      ( <>,
6        [ dlg.progress.state < D ]
7        C:UpdateProgress
8      )*,
9      [ dlg.progress.state == D ]
10     C:{ dlg.result = "done"; }
11   >
12 | <>,
13   H:PushCancelBtn,
14   C:{ dlg.result = "canceled"; }
15 ),
16 C:CloseProgressDlg;
```

図 6 プログレスダイアログのインタラクション記述

4.3 記述例 ドラッグ・アンド・ドロップ

プログレスダイアログよりも小さな粒度のインタラクションとして, また同一の機能に割り当てることが可能な異なるインタラクションの記述の例として, ドラッグ・アンド・ドロップでアイコンを移動させるインタラクションを記述する. 図 7 は画面上に出現する 2 つのオブジェクトマウスポインタ(Mouse.pointer)とアイコン(icon)である.

ドラッグ・アンド・ドロップの操作に対して, システムは例えば図 8 に示す 3 つの異なるフィードバックを返すことができるとする. (c)では, アイコンの移動の軌跡が画面上に表示されるため, 軌跡のオブジェクト(icon.trace)が必要となる. さらに, 今回は, ドラッグ・アンド・ドロップの操作(アイコンの上でマウスのボタンを押し下げ, ボタンを押し下げたままマウスポインタを移動させ, そのあとマウスのボタンを離す)を記述するために, マウスのボタン(Mouse.button)もオブジェクトとする.

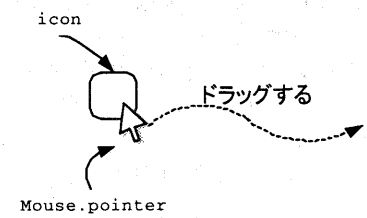

図 7 マウスポインタとアイコン

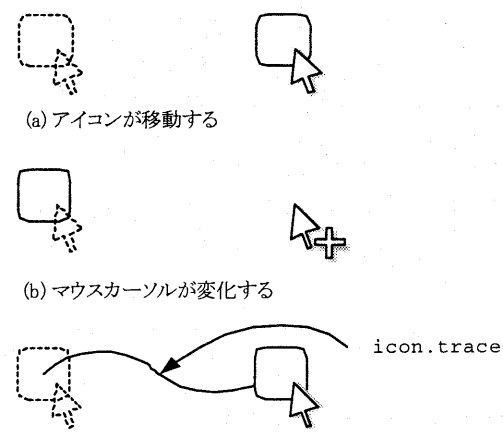

図 8 ドラッグ・アンド・ドロップの 3 種のフィードバック

```
H:MoveIconByDnD :=
  [ Mouse.pointer.isOn(icon)
    && Mouse.button.sate == "up" ]
  H:{ Mouse.button.state = "down"; },
  C:HoldIconRes,
  < [ Mouse.button.state == "down" ]
    (
      H:MoveMouse,
      C:DraggingRes
    )*
  >
  H:{  Mouse.button.state  =  "up";  },
C:ReleaseIconRes;
```

図 9 ドラッグ・アンド・ドロップのインタラクション記述の共通部分

```
        H:MoveIconByDnD :=
     [ Mouse.pointer.isOn(icon)
   && Mouse.button.sate == "up" ]
    H:ClickMouse, C:HoldIconRes,
< [ Mouse.button.state == "up" ]
                ( <>,
            H:MoveMouse,
            C:DraggingRes
                )*
                >
            H:ClickMouse,
          C:ReleaseIconRes;
```

図 10 クリック 2 回で移動させるインタラクション

図 9〜図 13 はこれらのインタラクションの記述である. 図 9 は 3 つのインタラクションの共通部分の記述であり, 図 11, 図 12, 図 13 の固有部分と組み合わせることで, それぞれ, (a), (b), (c) のインタラクションを表現する. 共通部分には人間のアクションの詳細が, 固有の部分にはコンピュータのアクション(画面上でのフィードバック)が記述されている.

次に, この記述を利用して, ドラッグ・アンド・ドロップとよく似たインタラクションを記述してみる. 将棋のゲームなどでよく見られる, 「移動させる駒をクリックしてから, 移動先のマスをクリックする」とインタラクションを考える. ドラッグ・アンド・ドロップではドラッグ操作で行ったものが, クリック 2 回で行われている. ドラッグ・アンド・ドロップの記述例を, クリック 2 回の移動操作に変更するには, 図 9 のインタラクション共通部分の(すなわち, 人間のアクションの)記述を, 図 10 のものに修正すればよい.

この例では, ドラッグ・アンド・ドロップのインタラクションを, 人間のアクションとコンピュータのアクションに分割することで, 記述の構造化を行った. これら人間のアクション(2 種類)の記述は, コンピュータのアクション(3 種類)の記述と, 自由に組み合わせることが可能である. このような構造

```
C:HoldIconRes := C:{ offset = icon.position - Mouse.pointer.position; };
C:DraggingRes := C:{ icon.position = Mouse.pointer.position + offset; };
C:ReleaseIconRes := null;
```

図 11 ドラッグ・アンド・ドロップのインタラクション記述の(a)に固有の部分

```
C:HoldIconRes :=
  C:{ Mouse.pointer.appearance = "holding";
    offset = icon.position - Mouse.pointer.position;
  };
C:DraggingRes := null;
C:ReleaseIconRes :=
  C:{ icon.position = Mouse.pointer.position + offset;
    Mouse.pointer.appearance = "normal";
  };
```

図 12 ドラッグ・アンド・ドロップのインタラクション記述の(b)に固有の部分

```
C:HoldIconRes :=
  C:{ icon.trace = icon.createTrace();
    offset = icon.position - Mouse.pointer.position;
  };
C:DraggingRes :=
  C:{ icon.position = Mouse.pointer.position + offset;
    icon.trace.addPoint(icon.position);
  };
C:ReleaseIconRes := C:{ icon.trace.finish(); };
```

図 13 ドラッグ・アンド・ドロップのインタラクション記述の(c)に固有の部分

化は，インタラクションの選択を可能にするシステムの構築に利用できると考えられる．

また，この例においては，プログレスダイアログの例では記述に現れなかったマウスが，インタラクションの記述に現れている．これは，CHIOPA 言語による記述が，記述されるインタラクションに応じて，記述の粒度を柔軟に選択できることを意味している．

5. まとめ

本稿では，インタラクションを記述する言語 CHIOPA を提案した．CHIOPA 言語は，人間やコンピュータの内部状態ではなく真にインタラクションを記述すること，時間的制約が記述できること，記述の粒度を柔軟に選択できることを特徴とする．CHIOPA 言語による記述の例では，プログレスダイアログの例と，ドラッグ・アンド・ドロップのインタラクションの例を示した．ドラッグ・アンド・ドロップの例では，構造化された記述によって，インタラクションの選択を可能とするしコンピュータシステムの構築に，CHIOPA 言語が利用できる可能性を示した．

参考文献

[1] M. Du and D. England: "Temporal Patterns for Complex Interaction Design", *Proc. DSVIS 2001*, 2001.

[2] P. Gray, D. England, and S. McGowan: "XUAN: Exchanging the UAN to capture temporal relationships among actions", *Proc. HCI '94*, pp.26-49. 1994.

[3] H. R. Hartson: "Temporal Aspects of tasks in the User Action Notation", Human Computer Interaction, 7(92), pp.1-45. 1992.

[4] 池田瑞穂, 高田喜朗, 関浩之: "ユーザタスクの形式的記述に基づくインタラクティブシステム設計法の提案", *情報処理学会研究報告, SE122*, pp.25-32, 1999-3.

[5] R. J. K. Jacob: "A Specification Language for Direct-Manipulation User Interfaces", *ACM Trans. Graphics, Vol. 5, No. 4*, pp.283-317. 1986.

[6] R. J. K. Jacob, L. Deligiannidis, and S. Morrison: "A Software Model and Specification Language for Non-WIMP User Interfaces", *ACM Trans. Computer-Human Interaction, Vol. 6, No. 1*, pp.1-46. 1999.

[7] C.W. Johnson and M.D. Harrison: "Using Temporal Logic To Support The Specification And Prototyping Of Interactive Control Systems", *International Journal Of Man-Machine Studies*, Vol. 36, pp. 357-385. 1992.

[8] T. K. Landauer: *The Trouble withComputers, MIT Press*, Cambridge, MA. 1995.

[9] 中小路久美代, 神谷年洋, 高嶋章雄, 山本恭裕: "ヒューマンコンピュータインタラクションの表現形態とインタラクティブビューア", ヒューマンインタフェースシンポジウム 2001, ヒューマンインタフェース学会, Osaka, Japan, pp.155-158, October, 2001.

[10] 尾崎寛和, 渋谷 雄, 辻野嘉弘, "形式的仕様に基づく GUI 開発環境の試作", *情報処理学会研究会報告, 2001-HI-92*, pp.55-62. 2001-1.

[11] B. Shneiderman: "Multiparty Grammars and Related Features for Defining Interactive Systems", *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 12, No. 2, pp.148-154. 1982.

[12] B. Shneiderman: *Designing the user interface: Strategies for effective human-computer interaction*, Addison-Wesley, 1992.

[13] B. Shneiderman and P. Maes, "Direct Manipulation vs. Interface Agents", ACM Interactions, 4(6), pp. 42-61. 1997.

[14] W3C Synchronized Multimedia, http://www.w3.org/AudioVideo/.