ヒューマンインタフェース 77－5
（1998. 3. 13）

タスク指向インタラクション設計に基づくUIアーキテクチャ設計法の検討

伊澤 謙一 † 徳田 佳一 † 村松 成治 †† 田中 康 †† 李 殷碩 ‡ 白鳥 則郎 †

† 東北大学電気通信研究所
情報科学研究科
仙台市青葉区片平 2-1-1

†† ソニー株式会社
ヒューマンインタフェースラボ
東京都品川区北品川 4-7-35

‡ 韓国成均館大学
工科大学情報工学科
京畿道水原市長安区泉川洞 300

近年のユーザインタフェース (UI) に対する要求の多様化に対し，そのようなユーザ要求を十分反映したUIを効果的に設計することの必要性が高まっている．本稿では，著者らが提案している記述法TID (Task oriented Interaction Description technique) によるインタラクション設計結果に基づく，UIアーキテクチャ設計支援手法を提案する．まず，UIアーキテクチャを表現するためのモデルとして，TIO(Task oriented Interaction Object) を提案する．次に，TIOを用いたUIアーキテクチャ設計を支援するために，TID記述におけるタスク関係に基づいたTIOネットワークの導出法を提案する．

Design Methods for UI Architecture based on Task oriented Interarcion Design

Ken'ichi Isawa † Yoshiichi Tokuda † Seiji Muramatsu †† Yasushi Tanaka ††
Eun-Seok Lee ‡ Norio Shiratori †

† Research Institute of Electrical Communication / Graduate School of Information Sciences, Tohoku University, Sendai, Japan

†† Human interface laboratory, Sony CO. Tokyo, Japan

‡ Department of Information Engineering, Faculty of Engineering, SKKU(Sung-Kyun-Kwan University), Seoul, Korea

With the diversity of requirements for user interfaces(UIs), it is essential to design effectively UIs which reflect such user requirements. In this paper, we propose methods to support UI architecture design based on the interaction design specification described by our proposed TID, Task oriented Interaction Description technique. In order to represent an UI architecture, TIO, Task oriented Interaction Object, is introduced as a model. A method of constructing a TIO network structure based on task relationship of TID specifications is also added to support UI architecture design.

1 はじめに

1.1 背景

近年のユーザインタフェース (UI) に対する要求の多様化に伴い，ユーザ要求を十分反映したUIを効果的に設計することの必要性が高まっている．そのため，ユーザが要求するタスクを，効果的に遂行可能なインタラクションを実現可能なUIのアーキテクチャを構成するための設計支援法が必要である．

本稿では，そのための設計支援手法として，タスク指向UIアーキテクチャモデルと，それに基づくUIアーキテクチャの構成手法を提案する．

1.2 UI設計プロセス

タスク要求の獲得及びその分析に基づきUI設計が行われる．本稿では，UI設計プロセスをインタラクション設計ステージとUIアーキテクチャ設計ステージに大別する．

(1) インタラクション設計ステージ インタラクション設計ステージでは，タスク要求に基づきユーザとUIシステム間のインタラクションを設計する．著者らはインタラクション設計に対し，タスク指向インタラクション記述法 TID(Task oriented Interaction Description technique)[1] [2] を提案している．TIDでは，タスクという観点からインタラクションを記述する．タスクはいくつかのサブタスクによって構成され，サブタスクとそれらの間の関係を定義することでタスクを記述する．

(2)UIアーキテクチャ設計ステージ TIDに従って記述されたインタラクション設計結果をもとに，UIアーキテクチャを設計する．UIアーキテクチャ設計ステージでは，インタラクションを実現するUIオブジェクトを具体化し，更にオブジェクト同士の制御・制約を定義することで，システム全体の振舞いを設計する．

1.3 目的

本稿では，UIアーキテクチャ設計に焦点を当てる．インタラクション設計結果を十分にUIアーキテクチャ設計に反映するためには，以下の2つが必要であると考える．

1. UIアーキテクチャを表現するモデル
2. モデルに基づく効果的なUIアーキテクチャ設計支援手法

2節では，UIアーキテクチャ記述モデルTIO(Task oriented Interaction Object) を，3節では，TIOに基づくUIアーキテクチャの構成法を提案する．

2 TIO

TIOは，UIアーキテクチャを構成するコンポーネントである．UIアーキテクチャ全体は，TIOコンポーネントのネットワークで表現される．

2.1 TIOの内部モデル

TIOの基本的なアーキテクチャを図1に示す．TIOの動作は，(1) 外部動作レベル，(2) 内部動作レベルの2つのレベルで記述できる．外部動作レベルでは，TIOの外部的な振舞いのみを考慮し，内部動作レベルでは，内部状態を考慮した記述を行う．TIOの動作は，形式記述技法 LOTOS[3] を用いて記述される．

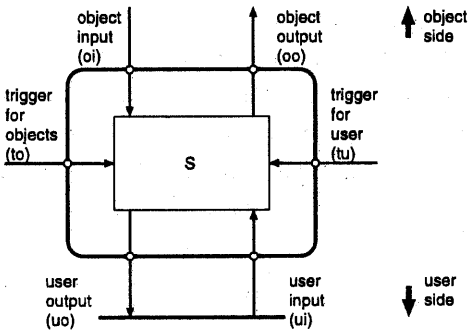

図 1: TIOの基本アーキテクチャ

(1) 外部動作レベル 外部動作レベル記述では，TIO内部をブラックボックスとみなし，TIO外部に対する振舞いのみを以下のように記述する．

```
process tio[ui,uo,to,tu,oo,oi] : noexit : =
    (oi; tio[ui,uo,to,tu,oo,oi]
    [] tu; uo; tio[ui,uo,to,tu,oo,oi]
    [] ui; uo; tio[ui,uo,to,tu,oo,oi]
    [] to; oo; tio[ui,uo,to,tu,oo,oi]
    )
endproc
```

(2) 内部動作レベル 内部動作レベルにおいては，TIOの内部状態に踏みこんだ仕様記述を行う．

```
process tio[ui,uo,to,tu,oo,oi](S : state_of_tio) :
 noexit : =
    (oi?oid:object_input_data;
            tio[ui,uo,to,tu,oo,oi](update(oi,S))
   []tu?True; oi?oid:object_input_data;
            uo!visualize(update(oi,S));
            tio[ui,uo,to,tu,oo,oi](update(oi,S))
   []ui?uid:user_input_data;
            tio[ui,uo,to,tu,oo,oi](update(ui,S))
   []to?True; oo!translate(S);
```

```
            tio[ui,uo,to,tu,oo,oi](S)
    )
endproc
```

この記述は以下のような動作を規定している.

1. ユーザからの操作を user input(ui) で受けとり, そのデータを S に蓄積し, 結果を user ouput(uo) へフィードバックする.
2. trigger for objects(to) でトリガを受けとった時, S で蓄積していた入力データを object output(oo) へ出力する. この時, データは oo で指定された形式に変換される.
3. 他のオブジェクトからの入力を object input(oi) で受けとり, S に蓄積する.
4. trigger for user(tu) でトリガを受けとった時, S で蓄積していたデータを, ユーザに提供する形式で uo へ出力する.

2.2 関連研究

文献 [4] 及び [5] の "interactor" は, TIO と同様に, UI アーキテクチャをインタラクションオブジェクトという単位で捉えたものである. interactor の概念に基づく具体的なモデルとして, CNUCE モデル [4] と YORK モデル [5] がある. これらのモデルと TIO との比較を表 1に示す. TIO モデルは, その特徴として, CNUCE モデルの動作指向な記述能力と, YORK モデルの状態指向な記述能力を併せ持つ.

	CNUCE	YORK	TIO
記述言語	LOTOS	Z	LOTOS
記述内容	動作指向	状態指向	動作指向 + 状態指向

表 1: UI アーキテクチャモデルの比較

3 TIO による UI アーキテクチャの構成

TID で記述されたインタラクション設計仕様 (TID 仕様) に基づき, TIO を用いた UI アーキテクチャを構成する. TID 仕様は, タスクを基本単位とした階層構造で記述されており, この階層を利用した TIO ネットワークの構成手法を提案する. まず, タスク階層における最下層の個々のタスクに, TIO を割り付ける. 次に, TID 仕様で記述されたそれらのタスク間の関係に基づき, TIO 間の制御構造を設計する. これを反復的に繰り返すことにより, TIO を用いた UI 全体のアーキテクチャを導出できる. 以下では, TID のタスク関係に基づく, TIO 間の制御構造の導出方法について説明する.

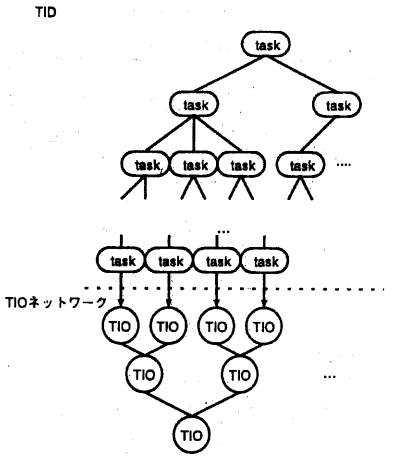

図 2: TID から TIO オブジェクトネットワークへ

3.1 タスク関係に基づく TIO 導出規則

TID におけるシーケンシャル及びパラレル関係子に対し, TIO 間の制御構造の導出規則を与える.

3.1.1 シーケンシャル関係子

タスク間のシーケンシャル関係から, 図 3に示す TIO ネットワークを導出する. 関係子毎に, 導出される TIO ネットワークの特徴を示す.

(1)sequence X:=A >> B

タスク A を実行する TIO A のみが, まずゲート A-tu におけるトリガによりインタラクション可能になる. タスク A の終了はゲート A-oo2 を通じて伝達され, タスク B を実行する TIO B は, ゲート B-tu でそれを受けてインタラクション可能になる. タスク B の終了時に, TIO B はそれをゲート B-oo から出力する.

(2)interrupt X:=A [> B

TIO A, B 各々が, ゲート A-tu1, B-tu2 でのトリガによってインタラクション可能になる. TIO A でのインタラクションが終了した場合, それを示す出力がゲート A-oo2 から TIO B のゲート B-tu1 に入り, TIO B は無効になる. 一方, TIO B でのインタラクションが開始した場合, それを示す出力がゲート B-oo2 から TIO A のゲート A-tu2 に入り, TIO A のインタラクションが無効になる.

(3)skippable X:= A [>> B

interrupt の構成に倣う. ただし, TIO A でのインタラクションが終了した後も, TIO B の動作は可

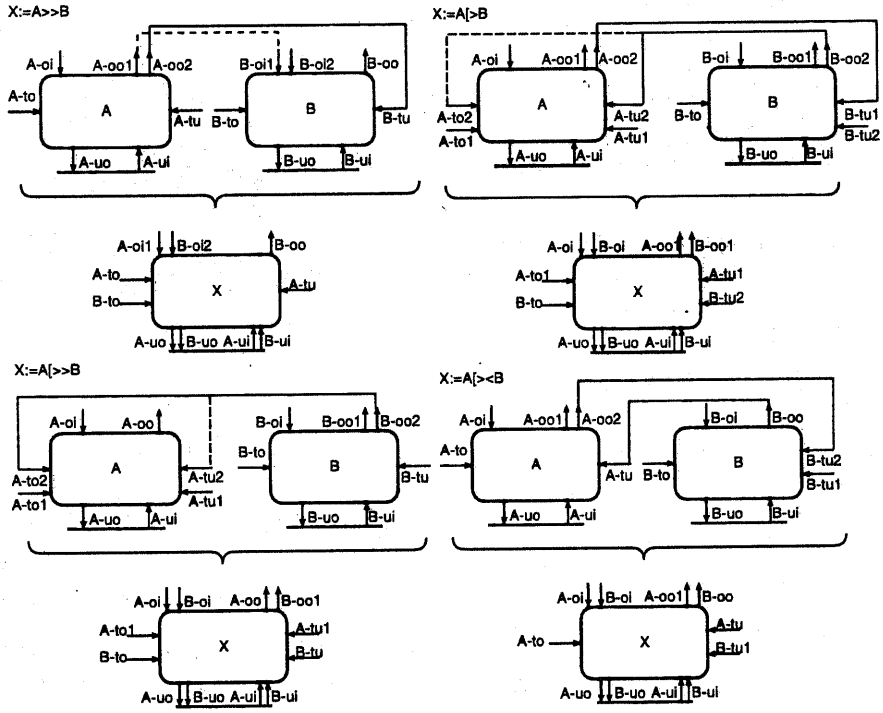

図 3: シーケンシャル関係子の TIO による表現

能であるため，TIO A から TIO B のゲート B-tu への接続が存在しない．

(4)modal interleave X:= A [>< B

TIO A でのインタラクションが開始されると同時に，ゲート A-oo2 を通して TIO B でのインタラクションが可能になる．TIO A でのインタラクションが終了した場合は，ゲート A-oo2 を通して TIO B でのインタラクションが無効になる．

3.1.2 パラレル関係子

パラレル関係の場合，各サブタスクを実行する TIO の他に，それぞれの出力を制御する役目の TIO を設ける．導出される TIO の構成は図 4の通りで，P の動作は，各関係子ごとに異なる．

(1)choice X:= | (A,B)

A, B いずれかの TIO で開始トリガが発生した場合，他の TIO のインタラクションを無効にする．

(2)parallel first X:= | F | (A,B)

全ての TIO が同時に実行可能である．A, B いずれかの TIO で終了トリガが発生した時点で，全ての TIO のインタラクションを無効にする．

(3)parallel last X:= | L | (A,B)

全ての TIO が同時に実行可能である．全ての TIO での終了トリガを P が受けとった時点で，全ての TIO のインタラクションを無効にする．

(4)order independence X:= >< (A,B)

A, B いずれかの TIO の開始トリガを受けとった場合，他の TIO のインタラクションを無効にする．インタラクションが終了した時点で，それ以外の全ての TIO のインタラクションを再び有効にする．

3.2 適用例

TIO による UI アーキテクチャ構成の具体例として，ファイル指定タスク（図 5参照）を考える．ファイル指定タスクは，TID を用いて図 6のように記述され，これに基づき構成された TIO を用いた UI アーキテクチャ設計の一部を図 7と図 8に示す．図 6の「ファイル選択」タスクにおいて，そのサブタスク「ファイル・フォルダ移動」と「ファイル決定」は interrupt

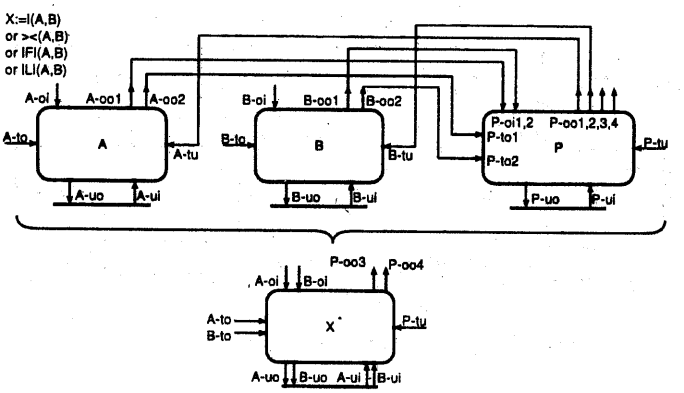

図 4: パラレル関係子の一般的な表現

ファイル指定	:=	ファイル選択 [>キャンセル
ファイル選択	:=	ファイル・フォルダ移動* [>ファイル決定
ファイル・フォルダ移動	:=	I (ファイル移動, フォルダ移動)

図 6: ファイル指定タスクの TID 仕様 (一部)

関係であるので, 図 3に示す導出規則に従い, 図 7の TIO ネットワークが構成される. また, 「ファイル・フォルダ移動」タスクにおいて, そのサブタスク「ファイル移動」と「フォルダ移動」は choice 関係であるので, 図 4に示す導出規則に従い, 図 8の TIO ネットワークが構成される.

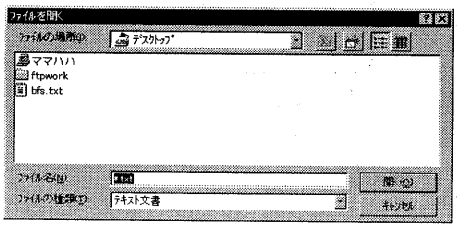

図 5: ファイル指定タスクを実現する UI イメージ

4 まとめ

本稿では, インタラクション設計結果に基づく効果的な UI アーキテクチャ設計手法を提案した. 具体的には, UI アーキテクチャ記述モデル TIO(Task oriented Interaction Object) を提案し, TIO に基づく設計支援法を構成した. 今後の課題として, TIO による UI アーキテクチャ設計結果に基づく, 検証支援や実装支援の検討がある.

参考文献

[1] 石山 啓太郎, 徳田 佳一, 鈴木 主真, 田中 康, 李 殷碩, 白鳥 則郎: インタラクション設計法 TID を用いた分析法の検討, 情処研報 HI77-4, 情報処理学会 (1998).

[2] Tokuda,Y.,Lee,E.S. and Shiratori,N.:Synthetic and Analytic Methods for User-Computer Interaction Design,Proc.of the 12th International Conference on Information Networking,IEEE Computer Society,pp.726-729,(1998).

[3] ISO.:Information Processing Systems — Open Systems Interconnection - LOTOS - A formal description technique based on the temporal ordering of observational behaviour, ISO/IEC 8807(1989).

[4] Paterno',F.:A Methodology to design interactive systems based on interactors,Technical Report WP7,ESPRIT BRA 7040 Amodeus-2(1993).

[5] Duke,D.J. and Harrison,M.D.:Abstract Interaction Objects, Computer Graphics Forum,NCC BlackWell,Vol.12,N.3,pp.25-36.

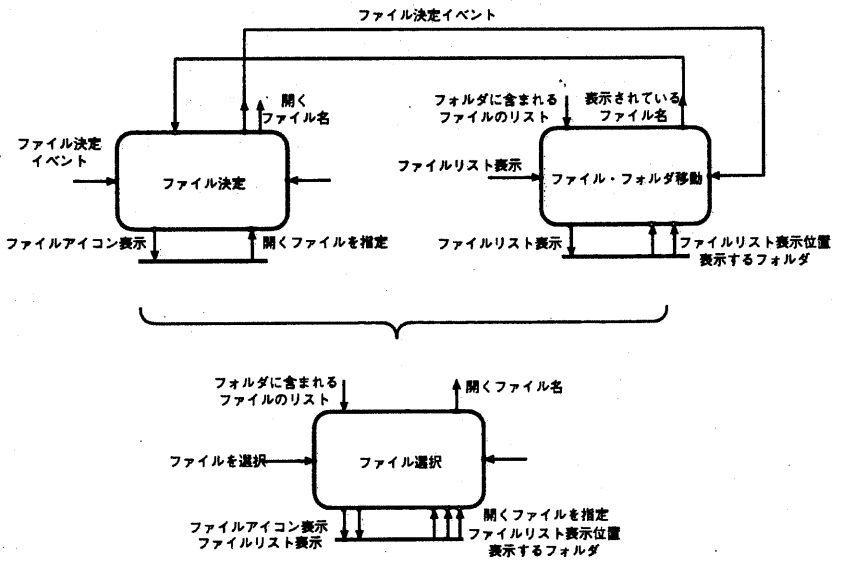

図 7: 「ファイル選択」タスクの TIO 構成

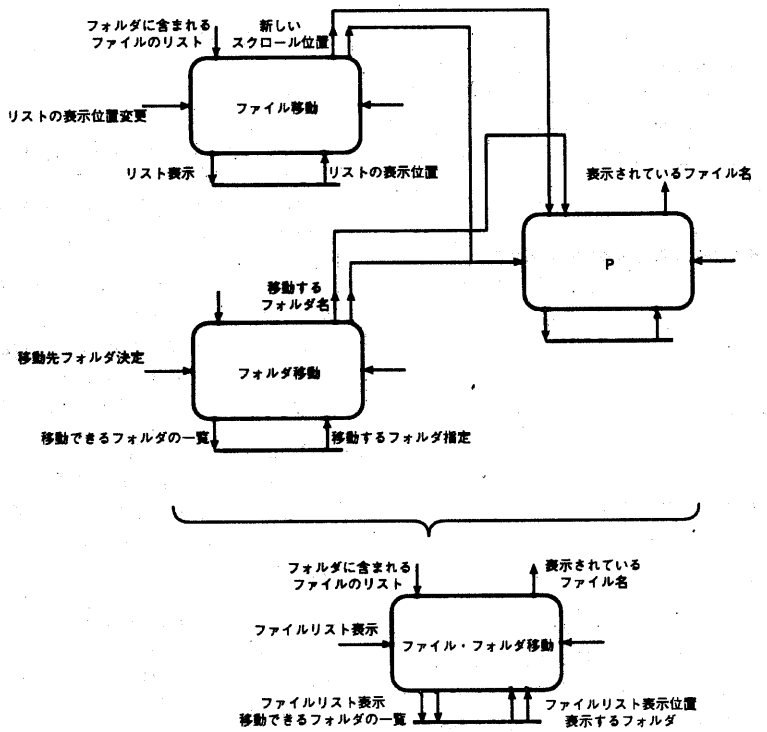

図 8: 「ファイル・フォルダ移動」タスクの TIO 構成