

インタラクション設計記法TIDを用いた分析法の検討

石山 啓太郎[†] 徳田 佳一[†] 鈴木 主真[‡] 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

[‡]ソニー株式会社
ヒューマンインターフェースラボ

[§]韓国成均館大学
工科大学情報工学科

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に反映した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、ユーザ/タスク中心の方法論をもとに、実際のインタラクション設計を効果的に支援するための設計・分析手法を提案する。まず、インタラクション設計記述法TID(Task oriented Interaction Description technique)による設計手法を提案する。次に、TIDに基づくインタラクション設計結果を用いた分析手法として、ユーザ操作回数予測について述べる。

Analytic Methods using Task oriented Interaction Description Technique

Keitaro Ishiyama [†], Yoshiichi Tokuda [†], Suma Suzuki [‡], Yasushi Tanaka [‡]
, Eun-Seok Lee [§] and Norio Shiratori [†]

[†] Research Institute of Electrical Communication
Graduate School of Information Sciences, Tohoku University

[‡] Human interface laboratory, Sony CO.

[§] Dept. of Information Engineering, Faculty of Engineering
SKKU(Sung-Kyun-Kwan University)

With the diversity of requirements for user interfaces(UIs), it is essential to design effectively UIs which reflect such user requirements. In this paper, based on user/task oriented methodologies, we provide design and analytical methods for supporting a practical interaction design. We propose a design method using TID, Task oriented Interaction Description technique, and also provide a analytic method for predicting the amount of user operations based on TID specifications.

1.はじめに

近年のコンピュータユーザの増加に伴い、ユーザインタフェース(UI)に対する要求が増大かつ多様化しており、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。これに対し従来研究においては、アプリケーション機能ありきの設計を改善すべく、ユーザやタスク視点に基づく設計方法論[2][3]が提案されている。

本稿では、このようなユーザ/タスク中心の方法論をもとに、実際の UI 設計作業、特にインタラクション設計を効果的に支援するための設計・分析手法を提案する。インタラクション設計は、ユーザのタスクに対する要求を十分反映した UI を実現する上で、非常に重要な役割を担うと考えており、まず、インタラクション設計記述法 TID(Task oriented Interaction Description technique)による設計手法を提案する。次に、TID に基づくインタラクション設計結果を用いた分析手法として、ユーザ操作回数予測について述べる。

2.TID

2.1 タスク定義

TID においてインタラクションは、タスクを基本単位として記述され、またそのサブタスク間に存在する関係の記述により、タスクの詳細が設計される。それゆえ、UI システム全体のインタラクション設計は、タスクをサブタスクとそれらの関係により定義し、それを繰り返すことで階層的に記述可能である。TID におけるタスクは、以下のように定義される。

$TASK := \langle ST, OP, AF \rangle$ (1)

$AF := \langle action_1, \dots, action_i, \dots, action_N \rangle$ (2)

定義式(1)の左辺 TASK は被定義タスク名、右辺の ST は TASK を構成するサブタスクの集合、OP は ST 間のタスク関係、AF は TASK に対するインタラクションを表すアクションフローである。定義式(2)の右辺は、 $action_i$ の実行順序である。個々の $action_i$ はユーザアクション(UA)、あ

るいは、それに伴うフィードバック(F)及びアフオーダーダンス(A)のいずれかである。ただし、UA は AF に高々一つのみ含まれるものとする。

2.2 関係子

タスク関係を簡潔かつ正確に記述するため、TID では、複数の既存システムのインタラクション設計結果の分析により、インタラクション設計に必要なタスク関係を抽出、整理し、それらを正確に表現可能な以下のタスク関係子を提供する。この結果、関係子の複雑な組み合わせによる技巧的な表現を削減することができ、設計者の知的負荷の軽減が期待できる。

2.2.1 シーケンシャル関係子

- Sequence : $A \gg B$
タスク A の終了後、タスク B が実行可能な関係を表す。
- Skippable : $A [\gg B$
タスク A のサブタスクをスキップして、タスク B を実行可能な関係を表す。ただし、タスク A が終了した場合はタスク B が実行可能な状態になる。
- Interrupt : $A [> B$
タスク A の実行途中において、タスク B が割り込んで実行可能な関係を表す。ただし、タスク A が終了した場合は $A [> B$ 全体が終了し、タスク B は実行不可能となる。
- Modal Interleave : $A [> < B$
タスク A が終了するまでの間、タスク B がモーダルに並列実行可能な関係を表す。

2.2.2 パラレル関係子

- Order Independence :
サブタスク群 A_i が任意順序で実行可だが、互いに割り込み不可である関係を表す。つまり、あるサブタスク A_i の実行終了後、他のサブタスク A_j が実行可能となる関係である。
- Choice : $| ((condition_i) A_i)$
サブタスク群 A_i のうちいずれかが、タスク開始時に選択可能である関係を表す。ただし、サブタスクの前に condition が定義されている場

合には、その条件が満たされる場合にのみ、そのサブタスクが選択可能である。

- **Parallel First : |F|(A_i)**
サブタスク群 A_i が同時に実行可能な関係で、サブタスク内の遂行順序は、そのサブタスクの定義に従うが、サブタスク間では特定の順序関係を持たない。サブタスク群 A_i のいずれかのサブタスクが終了した時点で、タスク |F|(A_i) 全体が終了する。
- **Parallel Last : |L|(A_i)**
Parallel First 関係とほぼ同じであるが、全てのサブタスク A_i が終了した時点でタスク |L|(A_i) 全体が終了する点が異なる。

2.2.3 再帰関係子

繰り返し実行される回数に応じて、次のような再帰関係子を用いる。

- A* : 0 回以上の任意回
- A+ : 1 回以上の任意回
- Aⁿ : n 回 (定数回)
- A% : 0 または 1 回

3. TID を用いた設計

3.1 インタクション設計手法

TID を用いたインタクション設計プロセスを図1のようにモデル化する。まず、タスク分析の結果から、ユーザ・ゴール構造を定義する。それに基づき、UI 機能設計、UI 機能構造設計、UI 機能実行手順設計を行う。

(1) ユーザ・ゴール構造定義

タスク分析により抽出されたタスクをもとに、そのタスクにおいて達成すべきゴールを階層的に記述する。これをユーザ・ゴール構造と呼び、図2に示すような構造を持つ。各ゴールは、それを達成するためのサブゴールの集合として定義され、詳細化される。

(2) UI 機能設計

ユーザ・ゴール構造をもとに UI システムが提供する UI 機能を設計する。この際に、ユーザ・ゴール構造とゴールを達成可能な UI 機能との関係を調べる。ゴールと UI 機能は一対一に対

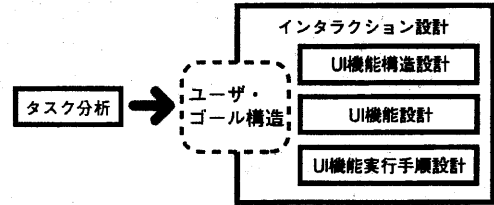


図1 インタクション設計プロセス

応する必要はないが、すべてのゴールを UI 機能のいずれか、もしくは複数の UI 機能の組み合わせで達成可能なように設計する必要がある。

(3) UI 機能構造設計

UI 機能構造設計は、ユーザが個々の UI 機能にアクセスするための手順を設計する。UI 機能構造は、ルートノードとして UI システム全体、リーフノードとして UI 機能を持つ階層構造として記述される。この UI 機能構造には、UI システムが提供する全 UI 機能が構造化され、TID を用いて記述される。

(4) UI 機能実行手順設計

UI 機能実行手順設計は、個々の UI 機能を実行するためのユーザ手順、つまり UI 機能の詳細を TID を用いて設計する。

3.2 設計例

上記の設計法に基づき、TID をグラフィカルに記述するための TID-Editor の再設計を行った。図3に TID-Editor のユーザ・ゴール構造の一部を示す。ユーザ・ゴール構造に基づき、ユーザに提供されている 25 個の UI 機能を抽出し、TID

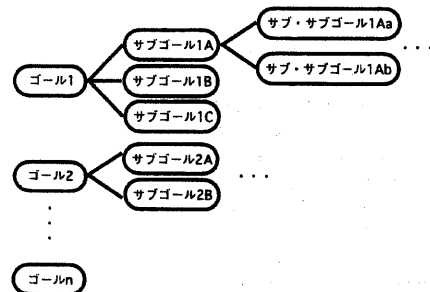


図2 ユーザ・ゴール構造

による記述を行った。記述の結果、TID仕様における使用関係子数は25であった。UI機能構造とUI機能実行手順のTID仕様の一部を図4に示す。

4. TIDを用いた分析法

既存の代表的なUI分析法としてGOMS法[4]があり、分析項目として、ユーザの操作回数予測、実行時間、操作に必要な学習時間などを提供する[5]。本稿では、TIDを用いた分析法について検討し、その分析項目としてユーザの操作回数の予測について述べる。

4.1 ユーザ操作回数の予測

インタラクション設計時に、任意のタスクに対するユーザの操作回数の予測を提供することにより、異なる設計案に関する比較検討が可能になる。しかしながらTIDでは、GOMSモデルと異なり、記述されたタスクが複雑な関係をもつので、ユーザの操作数を単純に算出することは出来ない。そこで、以下の算出手順により、ユーザ操作数の算出を行う。

4.1.1 テストゴールの決定

まず、ユーザ操作数の予測対象とするタスクのユーザ・ゴールを、テスト・ゴールTGとして設定する。

$$TG := \langle UG_1, UG_2, \dots, UG_k \rangle \quad (3)$$

$$UG_k := \langle UF_i, SP_i \rangle \quad (4)$$

$$SP_i := \langle P_j, V_j \rangle \quad (5)$$

TGはユーザ・ゴール構造におけるゴールUG_k

| ユーザ・ゴール構造 |
|-------------------|
| 新規に記述を行う |
|ノードの追加 |
|ノードの作成 |
|ノード名のラベル付け |
|関係を決定 |
|シーケンシャル関係子定義 |
|パラレル関係子定義 |
|再帰関係子定義 |
| 既存の記述の変更を行う |
|ノードの削除 |
|ノードの移動 |
|ノードの再定義 |

図3 TID-Editorにおけるユーザ・ゴール構造

の列として定義される。各UG_kは、そのゴールを達成するために必要なUI機能群UF_iと、各々のUF_iにおいて設定されるべきパラメータと値のセットSP_iで定義される。なお、SPに関しては、分析中に随時追加可能である。

4.1.2 TID部分構造の抽出

テスト・ゴールTGにおける各UGを達成するために必要のない部分をTID仕様から取り除くことにより、ユーザ操作回数の予測に必要なTID部分構造を以下のように抽出する。

- (1) | : UGに関係のないサブタスクを削除。
- (2) [> , [: UGに沿って >>>へ展開。
- (3) [< : モデルに割り込まれるタスクの割り込み回数を決定。
- (4) | F | : ユーザは最適な操作を行うと仮定し、|に置換。
- (5) 再帰関係子 : UGに沿って、その実行回を決定。(全てnの形式へ変換)

| |
|--|
| <p>●UI機能構造</p> <p>TIDエディタ:=(ファイル関連操作、TIDノード定義操作、TID関係子定義操作、編集関連操作、表示関連操作、作業補助操作)</p> <p>ファイル関連操作:=(新規作成、読み込み、保存、別名保存、テキスト出力)</p> <p>TIDノード定義:=(ノード作成、リンク作成、ノード選択)</p> <p>TID関係子定義操作:=(再帰パラレル関係子定義、シーケンシャル関係子定義)</p> <p>編集関連操作:=(カット、コピー、ペースト、削除、取り消し、全子ノード選択)</p> <p>表示関連操作:=(整理、宿対、最大化、スクロールバー表示)</p> |
| <p>●UI機能実行手順:ノード作成</p> <p>TIDノード定義:=(ノード作成、リンク作成、ノード選択)</p> <p>A:カテゴリーの表示</p> <p>UA:TIDノード定義を選択</p> <p>F:TIDノード定義に属する機能表示</p> <p>ノード作成:=ノードの作成>>ノード名入力[>>>決定</p> <p>A:TIDノード定義に属する機能表示</p> <p>UA:ノード作成を選択</p> <p>F:ノード作成モードであることをユーザに表示</p> |
| <p>●UI機能実行手順:リンク作成</p> <p>TIDノード定義:=(ノード作成、リンク作成、ノード選択)</p> <p>A:カテゴリーの表示</p> <p>UA:TIDノード定義を選択</p> <p>F:TIDノード定義に属する機能表示</p> <p>リンク作成:=リンク作成>>関係子決定</p> <p>A:TIDノード定義に属する機能表示</p> <p>UA:ノード作成を選択</p> <p>F:ノード作成モードであることをユーザに表示</p> |

図4 TIDによるインタラクション記述例

4.1.3 ユーザ操作数の算出

TID 部分構造に対し、ユーザアクション UA をユーザの基本操作単位として、ユーザ操作数を算出する。

[1] UG 毎の操作回数

UG 毎の操作数をボトムアップ的に算出する。TID の定義に従い、一階層上の親タスク遂行に必要な操作数を順次求めていく。n 個のサブタスクを持つ親タスクの操作数を N_x 、各サブタスクの操作数を N_i とすると、OP の種類により親タスクの操作数 N_x の求め方は以下のように分類される。

(1) OP := >>, |L|, >< の場合

$$N_x = \frac{\sum_{i=1}^n N_i}{n}$$

(2) OP := | の場合

各サブタスクの操作数の平均をとる。

$$N_x = \frac{\sum_{i=1}^n N_i}{n}$$

(4) OP := [= < の場合

予め決定した割り込み回数 m を用いる。

$$N_x = N_1 + m \times N_2$$

(4) OP := *, %, n, + の場合

予め決定した再帰回数 m との積をとる。

$$N_x = m \times N_1$$

[2] TG 全体の操作数

各 UG_i の操作数を N_i とすると、n 個の UG を持つ TG における全操作数 N_{TG} は、以下のよう
に算出される。

$$N_{TG} = \sum_{i=1}^n N_i \quad (6)$$

[3] ユーザ操作数予測の制約

本予測の制約として、算出されたユーザの操作回数は相対的な比較においてのみ有効な値であることが挙げられる。また、比較にあたっては比較対象の TID 記述の詳細化のレベルを揃えておくことが必要である。

4.2 分析例

3.2 節と同様の例題 (TID-Editor の再設計) に対し、本分析法を適用した。この再設計では、エディタのユーザテストにより抽出された問題点の解決のため、ユーザ操作回数予測に基づく分析及び代替設計案の検討が行われた。以下では問題点の一つである「シーケンシャル関係をもつ複数ノードの作成」タスクにおける操作性改善を例に説明する。

[1] 既存設計の分析

まず、テスト・ゴールを以下のように設定した。

TG(シーケンシャル関係をもつ複数ノードの作成)

:= <UG₁(ノード追加), UG₂(ノード追加),

UG₃(シーケンシャル関係子定義)>

UG₁ := <{UF₁(ノード作成), SP₁(ノード名≠ノード A)}>

UG₂ := <{UF₁(ノード作成), SP₁(ノード名≠ノード B)}>

UG₃ := <{UF₁(リンク作成), {SP₁(前ノード≠ノード A),

SP₂(後ノード≠ノード B), SP₃(関係子≠>>)}>

次に、各 UG に対応する TID 記述の抽出を行

| |
|---|
| ●UI機能構造 |
| TIDエディタ:=(ファイル関連操作, ノード作成, リンク作成, 選択, TID関係子定義操作, 編集関連操作, 表示関連操作, 作業補助操作) |
| ●UI機能実行手順:ノード作成 |
| ノード作成:=ノードの作成 >> ノード名入力 [>> 決定 |
| ●UI機能実行手順:リンク作成 |
| リンク作成:=リンクを作成 >> 関係子決定 |

(a) 代替設計案 A

| |
|--|
| ●UI機能構造 |
| TIDエディタ:=(ファイル関連操作, TIDノード定義操作, TID関係子定義操作, 操作モード選択操作, 編集関連操作, 表示関連操作, 作業補助操作) |
| 操作モード選択操作:=(ノード作成選択, リンク作成選択, 選択選択) |
| ●UI機能実行手順:操作モード選択操作 |
| 操作モード選択操作:=(ノード作成選択, リンク作成選択, 選択選択) |
| A:カテゴリ表示 |
| UA:操作モード選択操作 |
| F:カテゴリに属する操作表示 |
| ●UI機能実行手順:ノード作成(condition:ノード作成モード) |
| ノード作成:=ノードの作成 >> ノード名入力 [>> 決定 |
| ●UI機能実行手順:リンク作成(condition:リンク作成モード) |
| リンク作成:=リンクを作成 >> 関係子決定 |

(b) 代替設計案 B

図 5 TID による代替設計案

った。UG₁,UG₂に対して、図4のUI機能“ノード作成”が用いられる。抽出時の規則に従い、“ノード作成”タスクにおける“ノード名入力”と“決定”の間の関係子[>>]を>>>に置換し、また、“TIDノード定義”タスクにおいて選択されないのサブタスク“リンク作成”、“ノード選択”を取り除いた。抽出されたUG₁,UG₂のTID部分構造を図6に示す。

抽出したTID部分構造に基づき操作数を算出した。最下層でのタスク“ノードの作成”、“ノード名入力”、“決定”にはそれぞれ操作数1であるUAが割り当てられている。これらをサブタスクを持つ親ノード“ノード作成”の操作数は、>>>関係子で関係付けられているので、各サブタスクの操作数の総和3と、このタスク自身のUA1つを足した4となる。また、“ノード作成”をサブタスクとして持つ“TIDノード定義”は、サブタスクの操作数4とそれ自身の操作数1を足した5となり、UG₁,UG₂の操作数はそれぞれ5と算出された。同様の手順でUG₃についても同様の手順でその操作数を算出すると4となり、(6)式より設定したTGのユーザの操作数は14となった。

[2] 代替設計案の検討

TIDで新たに設計した代替設計案(図5(a), (b))に対し、[1]と同様に操作回数の予測を行った。代替案Aは、“ノード作成”、“リンク作成”の実行に対し直接ユーザアクションを与え、UI機能へアクセスする手順を省いたものである。[1]と同様のテスト・ゴールTGに沿ってユーザ操作回数の算出を行った結果、操作数は8となった。

代替案Bでは、操作モードを導入し、各操作を

| |
|-----------------------------|
| ノード作成 |
| TIDノード定義:=ノード作成 |
| A:カテゴリーの表示 |
| UA:TIDノード定義を選択 |
| F:TIDノード定義に属する機能表示 |
| ノード作成:=ノードの作成>>>ノード名入力>>>決定 |
| A:TIDノード定義に属する機能表示 |
| UA:ノード作成を選択 |
| F:ノード作成モードであることをユーザに表示 |

図6 TGに対して抽出されたTID記述

行う前にモードを選択し、次にモードを再選択するまでその操作を続行できる。この設計案の場合、ユーザ操作数は12と算出された。

既存のTID-Editorプロトタイプは操作数14と比較して、代替案A,Bともにユーザ操作数は減少しており、ユーザ操作数の観点から、代替案のインタラクション設計は問題解決に対し有効であると予想可能である。以上のようにTID仕様に基づき、ユーザ操作数を予測する方法を提供したことにより、異なる設計案の比較検討が容易に可能となる。

5.まとめ

本稿では、タスク指向インタラクション記述言語TIDを用いた設計手法と、その記述結果からのユーザ操作回数予測方法を提案した。今後の課題として、他の分析項目における分析法の検討がある。

参考文献

- [1]Y.Tokuda, E.S.Lee, N.Shiratori"Synthetic and Analytic Method for User-Computer InteractionDesign"Proceedings the 12th International Conference on Information Networking,pp.726-729,1998.
- [2]Lewis,C. and Rieman,J. "Task-Centered User Interface Design: A practical Introduction" University of Colorado,1993.
- [3]Norman,D. and Draper,S.W. "User Centered System Design" New perspectives on Human-Computer Interaction,Lawrence Erlbaum Associates,pp.526,1986.
- [4]Card,S.K., Moran,T.P., and Newell,A. "The Psychology of Human Computer Interaction" Lawrence Erlbaum Associates,pp.469,1983.
- [5]B.E.Hohn, D.E.Kieras "The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast"ACM Transaction on Computer-Human Interaction ,Vol.3,No.4, pp.320,1996.