ヒューマンインタフェース 50－2
（1993. 9. 17）

機能列および機能設定値の個人適応インタフェースモデルの提案

炭野 重雄　　宮武 孝文　　上田 博唯
sumino@crl.hitachi.co.jp　miyatake@crl.hitachi.co.jp　ueda@crl.hitachi.co.jp

（株）日立製作所 中央研究所

ユーザの操作に応じてそのユーザに適応するインタフェースモデルを提案する。ユーザの操作には複数の機能に渡る操作と機能内の操作が存在する。前者に対しては、機能列の使用頻度を重み付きペトリネットの重みとして蓄え、頻繁に用いられる一連の機能列を抽出し、マクロ機能として登録する機能列の個人適応方式を提案する。後者に対しては、設定値の組合せの使用頻度をハイパーキューブの要素として蓄え、頻繁に用いられる設定値の組合せを抽出する。その組合せを新規ボタンに登録する設定値の個人適応方式を提案する。本方式を用いることで各ユーザに応じたインタフェースを構成でき、操作手数を減少することができる。

A Proposal of an Adaptive Interface Model for Functional Sequence and Functional Parameters

Shigeo Sumino　Takafumi Miyatake　Hirotada Ueda

Central Research Laboratory, Hitachi, Ltd
1-280, Higashi-koigakubo, Kokubunji-shi, Tokyo 185, Japan

We propose an adaptive interface model based on the observation of user's operations. User's operations can be classified into two types : operations through two or more functions and operations in a closed function. In the former, we store the usage frequency of each functional sequence as a weight of weighted petrinet, and assign new macro command to high frequency functions. In the latter, we store the usage frequency of combination of functional parameter as a element of hyper-cube, and assign new button to high frequency parameters. By this inferface model, the interface adapting to user can be constituted and the number of user's operations can be reduced.

1 はじめに

コンピュータやそれを組み込んだ製品の設計者は、技術を追求するあまり、本当にユーザが必要としているニーズをないがしろにして設計を行なう傾向がある。その結果、コンピュータを設計している人間と使用しているユーザとの意識の相違が益々大きくなり、ユーザにとって使い勝手の良くないインタフェースを持つコンピュータが生産されるようになった。

このような設計者とユーザとの意識の相違から生じた使い勝手の悪いインタフェースを、よりユーザにやさしいものとするヒューマン・コンピュータ・インタラクション(HCI)が研究されるようになった。これは認知心理学に基づいてユーザとコンピュータとの関わり合いを研究する分野である。

通産省のFRIEND21プロジェクトでは、HCIとしてメタウェアとエージェンシーモデルの2本立てで、ユーザの操作に応じて変化するインタフェースの研究を進めてきている[6]。同プロジェクトの研究の一環として、文献[5]においてユーザの嗜好に応じて変化し、使い勝手を向上させる個人適応インタフェースを提案した。具体的には、“対象物の指定”と“機能の選択”との組合せによって成されるユーザの操作系列の中から、そのユーザが頻繁に用いる機能列をマクロ機能として登録する個人適応化モデルを提案した。

しかし、同モデルでは処理可能な対象物が一つであるとか、動的なモデル記述ができないなどの問題があった。そこで本報告では、モデルのベースをペトリネットに置き換えることにより、モデルの拡張を図る。また、ユーザ操作のきめ細かい支援を行なうため、機能内の操作に関する個人適応方式も提案する。

2章において、インタフェースを個人に適応させるための機能列の個人適応と設定値の個人適応の切り分けについて、3章において、機能列の個人適応について、4章において、設定値の個人適応について示す。5章において関連研究との比較を行ない、6章においてまとめと今後の課題について示す。

2 二段階個人適応方式の提案

あるアプリケーション・ソフトウェアを用いている時のユーザの操作には、複数の機能に渡る操作と機能内の操作とが存在する。例えば、LaTeXで文書を作成する場合、まずLaTeXの予約語を埋め込んだテキストファイルdoc.texをLaTeXにかけて、中間ファイルであるdviファイルdoc.dviを作成する。それをモニタに表示、もしくは、dvi2psというフィルタでPostScriptファイルに変換してプリンタから出力する。具体的な操作として、まず(1)latex doc.texとキーボードで入力してdoc.dviを作成し、次に(2)dvi2ps doc.dvi > doc.psと入力してdoc.psを作成する操作が必要となる。また、dviファイルをモニタに表示するプレビュアxdviは、起動時にxdvi -topmargin 0.5 -s 3 doc.dviなど様々な設定値を設定することができ、機能内の操作が必要となる。

ユーザは意図したタスクを行なう場合、抽象的な概念から実操作にブレークダウンして作業を行なう。その時の実操作とは上記した複数の機能に渡る操作や機能内の操作である。それらの操作のうち頻繁に用いるものを抽出して、複数の操作を一回で行なえるようにするとタスクの短縮が図れる。以下、頻繁に用いられる操作を抽出してタスクの短縮を図る個人適応方式について、複数の機能に渡る操作と機能内の操作に分けて提案する。

2.1 機能列の個人適応

複数の機能に渡る操作に対しては、ユーザによる“対象物の指定”と“機能の選択”とから構成される機能列を考察の対象とする。前述の例では、コマンドとして入力したlatex, dvi2psが機能であり、文書データであるdoc.tex, doc.dviなどが対象物である。機能にはキーボードから入力するコマンドだけではなく、マウスでGUIをクリックして起動するツールも含まれる。このような“対象物の指定”と“機能の選択”から成るユーザの操作は、ユーザの嗜好、年齢、成育した教育環境、そのアプリケーションに対する習熟度など、枚挙にいとまがないほどの様々な要因によって変化に富んだものとなる。しかし、その変化に富んだ操作であっても、使い込んでいるうちにそのユーザにとって習い性と成った操作系列、すなわち、頻繁に用いる操作系列というものが存在する。その頻繁に用いられる機能列を抽出し、マクロ機能(短縮機能)として登録することによってタスクの短縮を図る。これを機能列の個人適応と呼ぶ。

2.2 設定値の個人適応

機能内の操作に対しては、ある機能に設定すべき設定値が複数個存在する場合を考察の対象とする。機能列の

個人適応と同様、設定値の選択の仕方にもユーザの嗜好が反映される。そのユーザの嗜好が反映した頻繁に用いられる設定値の組合せを抽出し、新規のボタンに登録もしくはデフォルト値とすることでタスクの短縮を図る。これを設定値の個人適応と呼ぶ。

2.3 本報告におけるユーザ操作のイメージ

本報告で想定している操作のイメージを図2.1に示す。

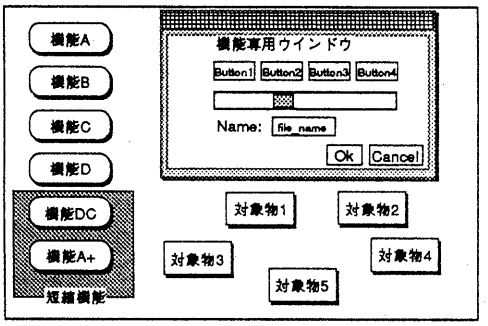

図 2.1: 操作のイメージ

図2.1において、ユーザはマウスによって(1)対象物を指定、(2)指定した対象物に施す機能を選択、(3)選択した機能の専用ウインドウにおいて設定値などを選択して対象物を加工、という操作を行なって目的のタスクを達成する。ここでの機能は、現在存在するMacintoshやWindowsのアプリケーションよりも粒度の小さいものを意識している。

例えば、(1)獲得、(2)分割、(3)加工、(4)並べ変えによって成される動画像の編集は、カットと呼ばれる対象物を各々が設定値を持った様々な機能によって加工して作品を完成させていく。

3 機能列の個人適応方式

3.1 重み付きペトリネットによる個人適応化モデルの定義

ユーザの操作において頻繁に用いられる機能列を抽出し、短縮機能として登録するため、重み付きペトリネットを用いた個人適応化モデル $M = (P, T; F, W)$ の要素を以下のように対応させる。ここで、P, T, F, W は各々重み付きペトリネットにおけるプレース、トランジション、アーク、重みを表す。

(a) トークン T_0：ある1つのプレースに存在し、現在起動している機能を表す。ユーザの対象物の指定と機能の選択との組合せに応じたトランジションが発火することによって、トークンがプレース間を移動する。

(b) プレース P：アプリケーション・ソフトウェアにおいて、ある機能を起動してから実行を終了するまでの状態に対応させる。

(c) トランジション T：ユーザが指定した対象物と選択した機能との組合せが実行可能なものかを判定するための起動条件に対応させる。実行可能な場合発火してトークンを移動させ、その結果トークンが移動したプレースに対応する機能が起動する。対象物の指定には、指定する個数も含まれる。従って、同一の機能を選択しても、指定する対象物の個数が異なれば、異なるトランジションが発火する。

(d) 入出力アーク F：トークンの通り道を表し、そのアークの方向で機能の実行順序に対応させる。

(e) 重み W：あるトランジションが発火したという条件下で、次に発火可能なトランジションの発火する確率に対応させる。

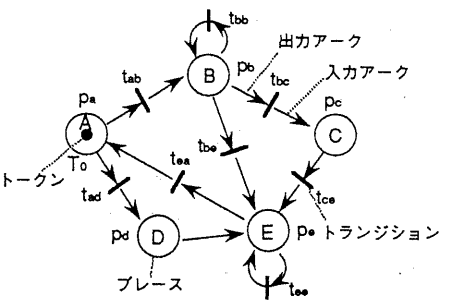

図 3.1: 重み付きペトリネットを用いた個人適応化モデル

図3.1を用いて上記の定義を説明する。図3.1において、現在の状態を表すトークン T_0 がプレース p_a に存在しているとき、すなわち、プレース p_a に対応する機能Aを起動終了後、ユーザが対象物 α を指定し、それに施す機能Bを選択した場合、α とBとの組合せに対応するトランジション t_{ab} が発火する。その発火によって、トークン T_0 がプレース p_a からプレース p_b に移動し、t_{ab} の発火の際指定された対象物 α を入力として、p_b に対応する機能Bが起動する。機能Bの終了後、次に発

火可能なトランジション t_{bb}, t_{bc}, t_{be} が発火するか、もしくは、p_b の出力アークがトランジションなしで別のプレースと接続していない限り、T_0 は p_b に留まる。

通常のペトリネットとは異なり、プレース間を接続するアークに必ずしもトランジションが存在している訳ではない。存在していない場合はトランジションの発火なしにプレース間をトークンが移動する。例えば、図3.1において、トークン T_0 がプレース p_a に存在する場合にトランジション t_{ad} が発火すると、T_0 が p_d, p_e と順次移動する。これは、トークンが通過したプレースに割り当てられた機能が順次起動することに対応する。

3.2 重みの保持方法

重みとトランジションを1対1に対応させると、そのトランジションと接続している2つのプレース間の接続関係は抽出可能である。すなわち、ある機能を起動した後、その次に起動する機能の確率は抽出可能である。しかし、どのような操作を経てその機能を起動したかというユーザの操作コンテクストまでは抽出できない。よって、よりユーザの操作コンテクストを反映した機能列を抽出可能とするため、連続して発火する2つのトランジションに対して重みを1つ対応させる。これはマルコフ連鎖に相当し、より詳細なトランジション間の発火の順序関係、すなわち、よりユーザの操作意図を反映した機能の起動を抽出可能となる [5]。

本モデルにおける重み付きペトリネットの重みを保持するため、関連する以下のデータを1つの構造体として管理する。以降、この構造体を**重み構造体**と呼ぶ。

(a) **重み** weight — あるトランジション t_i が発火したという条件下で、次にトランジション t_j が発火する確率 (≤ 1.0)。
(b) **トランジション t_i が発火することによって起動する機能の名称** pre_func
(c) **トランジション t_j が発火することによって起動する機能の名称** next_func

重み構造体の (b), (c) の要素は、更新すべき重みを特定するためと機能を起動する際の制御のために用いる。

3.3 重みの更新条件

重みの更新は、ユーザの意図が含まれると思われる以下の2つの場合に限り行なう。トランジションが発火する度に重みを更新するのではなく、ユーザの操作コンテクストに応じて重みの更新を行なうことで、ユーザの意図を反映した機能列の抽出を可能とするためである。

(A) 対象物を固定して複数の機能を順次施した場合 : 例えば、テキストファイル doc.tex を LaTeX にかけて中間ファイルである dvi ファイル doc.dvi を作成。その dvi ファイルを dvi2ps というフィルタを通して ps ファイル doc.ps にし出力するというパイプライン的な操作。ペトリネットでの定義に従うと、連続して発火した2つのトランジションによって特定される重み構造体の pre_func と next_func とが異なり、かつ、それらのトランジションに割り当てられた対象物が pre_func の入出力関係にある場合。

図3.2において、t_{ab} と t_{bc} によって特定される重み構造体の pre_func が B、next_func が C であり、t_{ab} で指定した対象物 α と t_{bc} で指定した対象物 α' とが機能 B の入出力の関係になっている場合、その重み構造体の重みを更新する。

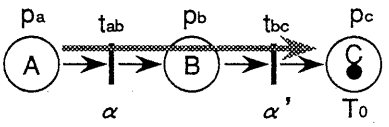

図 3.2: 重みの更新例 (その 1)

(B) 機能を固定して複数の対象物全てにその機能を施した場合 : 例えば、doc1.tex, doc2.tex, doc3.tex というテキストファイルに対して、doc1.tex, doc2.tex, doc3.tex の順で LaTeX をかけるという操作。ペトリネットでの定義に従うと、連続して発火した2つのトランジションによって特定される重み構造体の pre_func と next_func とが同一、かつ、それらのトランジションに割り当てられた対象物の個数が同一の場合。

図3.3において、t_{de} と t_{ee} によって特定される重み構造体の pre_func が E、next_func が E であり、t_{de} で指定した対象物 β_i と t_{ee} で指定した対象物 β_{i+1} とが意味的に連続になっている場合、その重み構造体の重みを更新する。

3.4 重みが閾値を越えた場合の処理

上記のようにして更新した重みがあらかじめ定めておいた閾値を越えた場合、その重みが属する重み構造体の

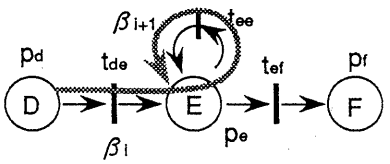

図 3.3: 重みの更新例 (その 2)

pre_func に該当する全てのプレースをコピーし、かつ、新たなトランジション、アーク、重み構造体を生成する。以下に詳細を示す。

(A) 対象物を固定して複数の機能を順次施したことによって閾値を越えた場合 : 図 3.4は対象物を固定して複数の機能 B, C を順次施したことによって、該当する重みがあらかじめ定めておいた閾値を越えた場合の短縮機能の生成方法を示したものである。この場合は、その重みが属する重み構造体の pre_func に該当するプレース p_b をコピーしてプレース p'_b とし、かつ、新たなトランジション t_{abc}、p_a と t_{abc} とを接続するアーク $\overrightarrow{p_a t_{abc}}$、$t_{abc}$ と p'_b とを接続するアーク $\overrightarrow{t_{abc} p'_b}$、$p'_b$ と p_c とを接続するアーク $\overrightarrow{p'_b p_c}$ を生成する。

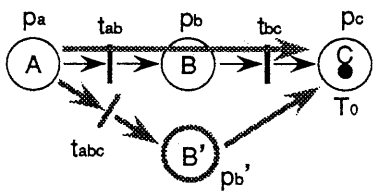

図 3.4: 短縮機能の生成 (その 1)

この処理によって、トークン T_0 が p'_b に移動した場合、トランジションが発火しなくても p_c に自動的に移動するようになる。すなわち、対象物の指定と機能の選択によって B′ が起動し、その機能を終了すると、対象物の指定と機能の選択という操作なしに C が起動する。その際、B′ が出力した対象物を C の入力として自動指定する。例えば、前述の LaTeX で文書を作成する場合では、テキストファイル doc.tex に対して機能である LaTeX と dvi2ps を順次起動して、ps ファイル doc.ps を生成することができる。

(B) 機能を固定して複数の対象物全てにその機能を施したことによって閾値を越えた場合 : 図 3.5は機能 E を固定して複数の対象物全てにその機能を施したことによって、該当する重みがあらかじめ定めておいた閾値を越えた場合の短縮機能の生成方法を示したものである。この場合は、その重みが属する重み構造体の pre_func に該当するプレース p_e をコピーしてプレース p'_e とし、かつ、新たなトランジション t_{dee}、および、p_d と t_{dee} とを接続するアーク $\overrightarrow{p_d t_{dee}}$、$t_{dee}$ と p'_e とを接続するアーク $\overrightarrow{t_{dee} p'_e}$、$p'_e$ と p_e とを接続するアーク $\overrightarrow{p'_e p_e}$、$p'_e$ を自己ループするアーク $\overrightarrow{p'_e p'_e}$ を生成する。

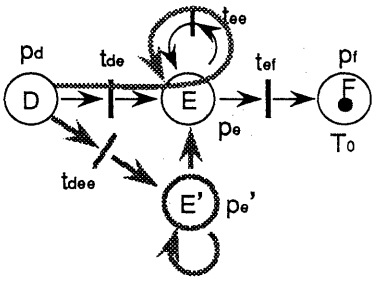

図 3.5: 短縮機能の生成 (その 2)

この処理によって、T_0 が p'_e に移動した場合、自己ループのアーク $\overrightarrow{p'_e p'_e}$ によってトランジションが発火することなく、自動的に再び p'_e に移動する。すなわち、ユーザの対象物の指定と機能の選択という操作なしに E′ が連続起動する。その際、処理可能な対象物が無くなるまで E′ が起動する毎に自動的に対象物を更新する。例えば、テキストファイル doc1.tex, doc2.tex, doc3.tex が存在した場合、対象物である doc1.tex を指定し機能である LaTeX を選択すると、対象物であるテキストファイルを doc2.tex, doc3.tex と更新しながら連続的に処理することができる。

4 設定値の個人適応方式

4.1 設定値の個人適応の定義

ユーザインタフェースにおける機能には、対話的に値を設定しなければならないものものが存在する。その設定値とは、例えば以下のようなものである。

(a) あらかじめ用意してある相互に排他的に働く複数の設定項目の中から 1 つを選択するもの。例えば、図 4.1に示した、ラジオボタンとして働く P_1, P_2, P_3, P_4 の中から 1 つ選択して設定するもの。

(b) 連続的に変更する動作パラメタ。例えば、図 4.1に示した R のようなスライダーをマウスでドラッグすることによって操作するもの。

(c) キーボードによって入力する文字列。例えば、図 4.1に示した S のようなキーボードから入力するもの。

(d) サンプリングによって設定する値

(e) その他の方法で設定する値

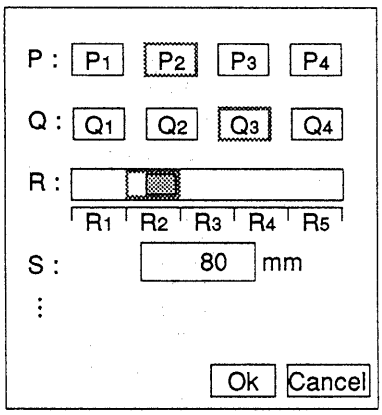

図 4.1: 設定値の種類

これらの設定しなければならない値が多数存在し、それらが相互に影響し合う場合、非常に煩わしい操作となり、意図した設定に調整するためには多大な時間を費やすことになる。その設定値の組合せには、機能列の個人適応と同様、多分に個人の嗜好が反映される。ユーザの嗜好を反映した頻繁に用いる設定値の組合せを抽出し、ボタンに登録することで、ユーザのタスクを短縮することができる。

そこで、ある機能における設定において、

- (a)のようなラジオボタンの場合、相互に排他的に働く設定項目を 1 つの選択項目の集合として扱う。図 4.1では、排他的に働く設定項目の集合 $\{P_1, P_2, P_3, P_4\}$ と、$\{Q_1, Q_2, Q_3, Q_4\}$ を各々 1 つの選択項目の集合とする。
- (b)のようなスライダーの場合、設定可能な値の範囲を適当な間隔で区分けし、区分けした範囲毎に設定項目を割り当てる。これらの設定可能な値に対応する複数の設定項目を 1 つの集合と見なす。図 4.1では、スライダ R の設定可能な値を 5 つに区切り、各々に設定項目を割り当てた $\{R_1, R_2, R_3, R_4, R_5\}$ を 1 つの選択項目の集合とする。
- (c)のようなキーボード入力の場合、入力する文字列が数字のときのみ (b)と同様に、設定可能な値を適当な間隔で区分けをし、区分けした範囲毎に設定項目を割り当てる。これらの設定可能な値に対応する複数の選択項目を 1 つの集合と見なす。

として、各設定項目の集合において 1 つずつ選択される設定項目の組合せのうち、ユーザが頻繁に用いるものを抽出し、登録、もしくは、デフォルト値とすることを**設定値の個人適応**と定義する。なお、前記 (d)、(e)に関しては今後の検討課題とした。

4.2 設定値の個人適応の実現方式

ある機能 f において、上記の定義で示した各々 N 個, M 個, L 個の設定項目によって構成される設定項目の集合 $P = \{P_i\}_{i=1,...,N}$, $Q = \{Q_j\}_{j=1,...,M}$, $R = \{R_k\}_{k=1,...,L}$ が存在する場合を例に用いて、設定値の個人適応の実現方式を示す。

(1) 各設定項目の集合の中から 1 つずつ設定項目を選択するときの全ての組合せの使用頻度を記憶する 3 次元の使用頻度行列 $\mathrm{S} = \{S_{ijk} = [P_i, Q_j, R_k]\}_{ijk}$ を用意する。

(2) ある機能 f における設定に応じて、以下のように使用頻度行列の値 $\{S_{ijk}\}_{i=1,...,N;j=1,...,M;k=1,...,L}$ を更新する。

- 設定項目 P_i が選択された場合、使用頻度行列の値 $\{S_{ijk}\}$ のうち i を固定した全ての値 $\{S_{ijk}\}_{j=1,...,M;k=1,...,L}$ を 1 加算する。
- 設定項目 Q_j が選択された場合、使用頻度行列の値 $\{S_{ijk}\}$ のうち j を固定した全ての値 $\{S_{ijk}\}_{i=1,...,N;k=1,...,L}$ を 1 加算する。
- 設定項目 R_k が選択された場合、使用頻度行列の値 $\{S_{ijk}\}$ のうち k を固定した全ての値 $\{S_{ijk}\}_{i=1,...,N;j=1,...,M}$ を 1 加算する。

よって、設定項目 P_I, Q_J, R_K が同時に選択された場合、S_{IJK} は 3 加算されることになる (図 4.2)。

(3) S_{ijk} があらかじめ定めておいた閾値を越えた場合、それに対応する設定項目の組合せ $[P_i, Q_j, R_k]$ を機能 f における設定値の組合せとして新規ボタンに登録、または各設定項目のデフォルト値とする。

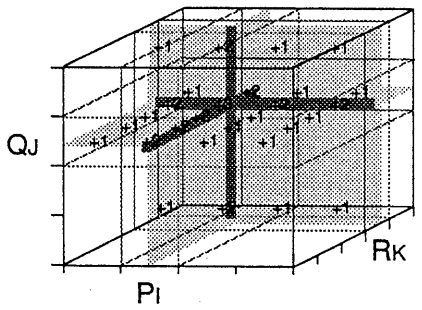

図 4.2: P_I, Q_J, R_K が同時に選択されたときの S_{IJK} の加算値

設定項目 P_I, Q_J, R_K が同時に選択されたとき、ユーザがその設定項目 P_I, Q_J, R_K の全ての因果関係を完全に把握している場合は、その選択の組合せに対応する S_{IJK} の値1つのみを加算すれば頻繁に用いる設定項目の組合せを抽出可能である。

しかし、設定値が多数存在する場合、ユーザは部分的には理解していても、全ての設定項目の因果関係を把握して設定を行なっているとは限らない。よって、設定項目の中に意識して選択していないものも含まれ、それらは設定の度に異なる可能性がある。設定の組合せに対応する1つの値のみ更新する方法では、そのような意識的に行なっていない曖昧な設定を考慮していないため、ユーザの嗜好を反映した組合せに対して値を更新しづらく、ユーザの嗜好に応じたものを抽出できない。

それに対し、$(M\times L+N\times L+N\times M-M-N-L+1)$ 個の S_{ijk} の値を更新して、有効ではない設定項目を含めた関係する全ての設定値の組合せに“1票投票(voting)”する方法では、分布的に値を更新するため、ユーザの意図した設定値の組合せに対しても値を更新することができる。また、前述の更新方式の説明から分かるように、設定値の組合せに応じて、+1, +2, +3と S_{ijk} の加算量が適切に変化するため、上記した因果関係のある項目 P_I, Q_J, R_K に対応する S_{IJK} だけを更新する方法の性質も包含しており、よりユーザの嗜好を反映した設定値の組合せを抽出可能である。

この更新方法は設定値の組合せのカテゴリーを抽出しているとも見なせる。設定の曖昧さという距離に応じて加算値を変更し、分布的に値を更新するので、その分布値に従って領域に分けると、設定値の組合せをカテゴリー分けすることに相当する。

以上、設定項目の集合が3つの場合について説明したが、ある機能において m 個の設定項目の集合が存在する場合、使用頻度行列Sは m 次元空間 (m 次元のハイパーキューブ空間)になり、その要素を各設定項目の選択に応じて加算することになる。

5 関連研究との比較

本報告で述べた重み付きペトリネットを用いた個人適応化モデルは、ユーザの操作のモデルであるタスクモデルをペトリネットで記述し、操作に応じてアークを生成して変形を加えるというものである。典型的なユーザのモデルを用意しそれを個々のユーザに応じて変形を加え、よりそのユーザにやさしいインタフェースを提供するという Rich の考察 [4] を具体化したものと見なせる。

また、本モデルはユーザの操作コンテクストを重要視し、それに応じた適応を施さなければならないとする Croft の概念 [1] も利用している。具体的には、対象物と機能との組合せがユーザの操作意図を表していると思われる条件を満たしていない限り、ペトリネットの重みを更新しない、というコンテクスト依存処理を行なっている。

その他に関連する研究として、木構造とメニューの選択頻度を用いて頻繁に用いられる選択肢をシステムが抽出し、メニューを変更するという方式 [3] と、頻繁に用いられる機能列をマクロコマンドとして登録する Eager [2] とが存在する。前者はアイディアの検証に重点が置かれており、機構としては木構造を選択頻度に応じて動的に変形し、対応するメニュー形式を変更するというものである。Eager は Macintosh の HyperCard 上で動作する実験システムであり、操作履歴を1次元的に管理し、ある操作が行なわれるとそれに類似したものを常に過去に遡って探索する。類似したものが存在した場合、探索のキーに用いた操作とそれに類似する操作との間の機能列をマクロコマンド化する。すなわち、機能列が類似した操作の繰り返しによって構成されている場合、その機能列の短縮を行なう。

メニューを変更する方式 [3] は、ユーザの操作に応じてモデルを動的に変更しているが、メニューの選択に着目しているだけで対象物に対しては考慮していない。本モデルは、機能と対象物の係わりを考慮して処理を行なっているので、よりユーザの操作意図を反映したモデルの変形を行なえるという点で優れている。また、Eager は、現在のところ機能列が繰り返しになっていない場合や機能列に誤った操作や遊びの操作が含まれた場合には

対応できない。本モデルでは機能列を2次元的にとらえ、選択頻度を用いて、それらの操作に適応可能という点で優れている。但し、Eagerは操作履歴をLisp言語で記述し、抽象度の高い処理を行なっているため、その処理いかんではより幅広い操作に適応できる可能性がある。

6 むすび

本報告では、ユーザのタスクを短縮するための二段階の個人適応方式について提案した。具体的には、あるアプリケーション・ソフトウェアを用いている時のユーザの操作を、複数の機能に渡る操作と機能内の操作と見なし、各々の操作に適した個人適応方式を採ることでタスクの短縮を図る。

複数の機能に渡る操作に対しては、ユーザが頻繁に用いる一連の機能列を抽出して、短縮機能として登録することでタスクの短縮を図る。それを機能列の個人適応と定義し、重み付きペトリネットを用いた実現方式を提案した。機能内の操作に対しては、ユーザが頻繁に用いる設定値の組合せを抽出して、その組合せをボタンに登録することでタスクの短縮を図る。それを設定値の個人適応と定義し、ハイパーキューブを用いた実現方式を提案した。

このような切口で個人適応をとらえ、二段階に渡り適応可能なインタフェースは過去に存在しなかった。現在のプロトタイプでは動画像の編集支援を用いているが、本報告の2つの適応方式はそれに限定されるものではなく、幅広い分野に適応可能と考える。また、本方式はノービスユーザがシステムをカスタマイズする際の障壁を低くするという面でも有用と考える。

今後の課題として、以下のものが列挙できる。

(1) **短縮機能の選択肢のユーザへの提示方法** — システム内部の機能の記述方法は複雑でもシステム自身が解釈可能であれば問題ない。しかし、複雑な記述をそのまま提示したのではユーザは理解できないので、ユーザが直観的に理解可能な形式に変換して提示する必要がある。現在、既存の機能の選択肢の外見(アイコン)を縮小し、並べて表示することで短縮機能の選択肢を表しているが、それだけでは記述に限界があり何らかの方策が必要である。

(2) **設定値の個人適応の実現** — 設定値の個人適応に関しては机上検討のみであるので、実際に実現してその機構の有用性を検証する必要がある。

(3) **個人適応機構の検証** — 機能列の個人適応メカニズムをほぼ理解しているユーザにはその機構の有用性を確認できた。次の段階として、機能列の個人適応と設定値の個人適応に関して全く知識のないユーザに、あらかじめ決めておいたタスクを与えた時、それら個人適応機構の有無によって使い勝手が向上するか否か検証する必要がある。

(4) **Undoが行なわれた操作の重み付けへの反映** — Undoがある決まった機能列の実行時に行なわれた場合、それはユーザにとっては意味を持つ操作となる。Undoが頻繁に行なわれた場合の重みへの反映方法を考案する必要がある。

上記の課題を踏まえ、作成したプロトタイプシステムで評価実験を行ない、ヒューマンインタフェース技術としての個人適応技術の有効性を検証してゆく予定である。

なお、本報告の一部は通産省のFRIEND21プロジェクトの一環として実施されたものである。

参考文献

[1] W. B. Croft. The role of context and adaptation in user interfaces. *Int. J. Man-Machine Studies*, (21):283–292, 1984.

[2] A. Cypher. EAGER : Programming Repetitive Tasks by Example. In *Proc. of CHI'91, Conf. on Human Factors in Computing Systems*, pp. 33–39. ACM, 1991.

[3] S. Greenberg and I. H. Witten. Adaptive personalized interfaces — a question of viability. *Behaviour and Information Technology*, 4(1):31–45, 1985.

[4] E. Rich. Users are individuals : individualizing user models. *Int. J. Man-Machine Studies*, (18):199–214, 1983.

[5] 炭野, 宮武, 上田. 動画像編集支援における個人適応化モデルの検討. 信学技報 HC92-46, pp. 63–70. 電子情報通信学会, 1992.

[6] (財) パーソナル情報環境協会. 第1回FRIEND21成果発表会予稿集, 1989.