ヒューマンインタフェース 53−16
（1994. 3. 11）

個人適応型のユーザーインターフェイス環境

塚本 高史　　中島 修
山田 雅高　　三宅 芳雄

中京大学 情報科学部

ymiyake@sccs.chukyo-u.ac.jp

使いやすい情報機器を実現するための基本的な方向の一つは、ユーザー個人個人が持つ特性や、ユーザーの回りの固有な状況にユーザーインターフェイスを適応させていく適応型のインターフェイスを実現していくことである。このような方向への試みとして、ユーザーのコマンド使用の履歴を利用して、インターラクティブにマクロを作成することを可能にするマクロ作成支援システムとユーザーがシステムについてより多くのことを知るためにシステムがユーザーに情報を提示する能動的な学習援助システムを実現し、実際に使ってその有効性を試し、今後の適応型インターフェイスの方向を検討した。

An Adaptive Interface Environment

Interactive Macro System and Active User Guidance

Takashi Tsukamoto　Osamu Nakashima
Masataka Yamada　Yoshio Miyake

Chukyo University
School of Computer and Cognitive Sciences

ymiyake@sccs.chukyo-u.ac.jp

One of the important directions in userinterface research and development is to make userinterface adaptive to the user and to the his work environment. We have developed and tested two types of adaptive user interface constructed on the user command records. One is the system which helps users to construt macro commands interactively by making use of his command history. The other is an active user guidance system which presents users commands which are judged to be unknown to the user, based on his command history. The future direction of the adaptive user interface is also discussed.

1 始めに

使いやすい情報機器を実現するための基本的な方向の一つとして、ユーザー個人個人が持つ特性や、ユーザーの回りの固有な状況にユーザーインターフェイスを適応させていくことがある。ユーザーインターフェイスの個人性への適応の形態は多様であるが[1]、ここでは、我々のインターフェイス実験環境の中で行っている二つの個人適応インターフェイスの実験的な試みを紹介する。

一つは、いわゆるマクロ機能をより使いやすいものにするために、ユーザーのコマンド使用の履歴を利用してインターラクティブにマクロを作成することを可能にするマクロ作成支援システムである。もう一つはユーザーの使用履歴に基づいて、ユーザーがシステムについてより多くのことを知るためにシステムがユーザーに情報を提示する能動的な学習支援システムである。

2 インターラクティブなマクロ作成インターフェイス

ここでマクロ機能とは、システムが提供している機能をユーザーが組み合わせて一つの機能を構成し、それをシステムに登録し、繰り返し使うことができるようにする機能を言うことにする。

マクロ機能が有用であるのは、ユーザーの要求する機能がユーザーに応じて固有であり、そのような機能を基本機能から新たに作り出すことによって、ユーザーの固有の要求に柔軟に応じることができるからである。ユーザーが構成するマクロに相当する機能をシステムがあらかじめ用意しておくことができれば、マクロ機能は必要ないが、さまざまのユーザーの固有な要求にすべて答えるようにシステムが機能を用意しておくことは難しい。また、仮にそのような機能を用意することができたとしても、ユーザーが必要としている機能をたくさんの機能の中から探し出すことは容易ではなく、むしろ、ユーザーが必要としている機能を簡単にマクロとして構成できた方が効率が良い場合も少なくないだろう。

マクロ機能が特定のプログラム言語を用いた本格的なプログラムと区別されるのはその簡便性である。マクロ機能を用いるのに、新しいプログラム言語を学ぶ必要はない。マクロはユーザーが既に知っているシステムの機能を組み合わせることによって構成できるために、ユーザーはシステムの使用の自然な延長上に比較的容易にシステムの機能を自分で拡張することができる。また、普通のプログラミング言語に較べて、より直接的でインターラクティブに使えるため、簡便に使える場合が多い。

マクロ機能によって、ユーザーは情報環境をユーザーの固有な状況に即応させていくことができるが、その変化自体はユーザーが自分でマクロを構成し作り出していくことによって実現しなければならない。現在、マクロを構成するためのユーザーの負担は、プログラムを構成すほどではないが、決して少なくはない。将来の課題は情報機器が必要なマクロを自動的に構成していくことを可能にすることである。しかし、ユーザーの意図や仕事の状況の情報をもとに必要なマクロを自動的に構成することは、高度な知能が要求され、簡単には実現できない。当面は、マクロ構成の認知過程に適合したインターフェイスを実現し、自動化可能な部分を自動化していくことにより、ユーザーの負担が少ないマクロ作成機能を実現することが現実的な目標になる。

以下では、従来のマクロ機能のインターフェイスの問題点を考察し、それを改善するための方策を考察する。

2.1 マクロ機能のインターフェイスの問題点

マクロ機能は簡便に使え、多くのユーザーにとって有用なものであるはずである。マクロ機能を持つシステムも少なくない。しかし、実際には、マクロ機能が使える情報環境であっても、あまり使用されていない場合が多い。例えば、Emacsエディターは多くのユーザーによって使われているシステムだが、そこに存在しているキイボードマクロと呼ばれるマクロ機能は必ずしも広く使われてはいない。Emacs のキイボードマクロはユーザーがキイボードから打ち込んだキイの列を登録しそれを繰り返すことができるようにした簡便な機能であり、新たに学習することは最小限でマクロを構成することができる機能である。筆者の一人はEmacsを長く使用しており、 時々、キイボードマクロを使うものの、それほど多用しない理由は、使う際の心理的負担が必ずしも小さくないからだと感じている。このような使用体験を基にして、心理的負担の増加につながるインターフェイスの問題点を検討してみると、以下のようなものがあるだろう。

- あらかじめ計画しなければならないことの負担
- エラーの回復の困難さ
- 構成されたマクロの明示性のなさ、編集の困難さ

Emacsのキイボードマクロは数個のコマンド（マクロの設定の開始、終了、実行、繰り返し）を知れば使えるが、マクロを作成するためには、まず設定の開始を指示し、それからマクロの本体を構成するキイの列を打ち込まなければならない。このため意図的にマクロを設定しようと計画しなければマクロは設定できない。また、マクロの設定中にシステムのエラーシグナルが発生するような間違いをするとマクロの設定は失敗に終わる。キイの数が多いほど、その間にエラーが起きる確率は増し、しかもやり直しの負担は大きくなるから、長いマクロを設定することは難しくなり、その心理的な負担も大きい。エラーの回復とも関係するが、構成されたマクロが必ずしも明示的でなく、マクロ自体の編集が簡単にはできないことも、使いにくさの原因の一つになっている。

Emacs のキイボードマクロのインターフェイスの心理的な負担が少なくないのは全体として、インターラクティブな性格が弱いことにその一因があると考えられる。普段のEmacsの中での作業は、働きかけた結果生じた状況を見ながら判断し、次の行為を決定し、また誤りを修正しながら、さきに進むというインターラクティブな過程として行われることが多い。これに対して、キイボードマクロの設定は、予め、マクロを設定するという計画性を要求され、しかも設定中に誤りを修正することが難しい。キイボードマクロの構成は予め一定の計画性が要求されるという点では、従来のプログラムの作成に近いところがある。一方で、構成したマクロが明示的でなく、その修正、試行が簡単にできないことも、キイボードマクロがインターラクティブでないものになっている一因であろう。

2.2 インターラクティブマクロ

マクロをもっと身近なものにし、使いやすくする一つの方向はマクロをよりインターラクティブなものにすることであり、本研究ではそのようなインターラクティブな性格の強いマクロのインターフェイスが持つべき要件を検討し、そのプロトタイプを構成し実際に使用する中で実験している。このプロトタイプをここは「インターラクティブマクロ」と呼ぶことにする。

マクロの作成をよりインターラクティブな

ものにする一つの方向は、ユーザーが自分のそれまでに行った操作を後から好きな時に振り返り、それを編集することによってマクロを構成することができるようにすることである。こうすることによって、ユーザーがあらかじめ計画しなければならないことの負担を減らし、マクロの作成が有効だと考えた時点で、自分のそれまでの作業を編集してマクロを作成することができるようになる。また、そのような編集ができるためには、ユーザーがそれまでにどのような作業を行ったかを明示し、容易に編集し、試すことができるようなインターフェイスが必要になる。

構成したマクロの再利用を可能にすることも、マクロの有用性を増す上で重要である。マクロはユーザーの仕事の固有な状況に対応して構成されたものであり、必ずしも再利用が保証されるわけではない。この点では何度も繰り返し使うことを想定して作られる場合が多い本格的なプログラムとは異なる。しかし、中には後で利用できるものも出てくる可能性がある。このような状況では特にユーザーの負担が軽く簡単にマクロを保存し、それを取り出すことができるようなインターフェイスが必要になる。 Gnu-Emacsのキイボードマクロも、保存し再利用を可能にすることはできるが、必ずしもその手続きは簡単ではない。構成したマクロは自動的にメニューから使えるなどのインターフェイスがあることが、再利用を促進するために必要である。

2.3 インターラクティブマクロのシステム構成

インターラクティブマクロの実験システムは日本語使用を可能にした Nepoch(いわゆるボタン機能を始めとしてグラフィックユーザーインターフェイス関係の機能を強化したGnu-Emacs版に日本語使用を可能にしたもの)の上に構成された。我々は現実の自然な使用の中でユーザーインターフェイスを研究する目的でNepochの上に、ユーザーの使用履歴を記録したり、メニューの使用が可能であるようなインターフェイス実験用の環境を構成してきた。この実験環境では、ユーザーのシステム使用の履歴がどのようなコマンドを使用したかというレベルで取り出せるようになっており、 インターラクティブマクロはこのコマンド使用の履歴に基づいてEmacs-Lispで実現されている。

マクロを作成しようとするユーザーは、メニュー、コマンド、またはコマンドに対応するキイによって、マクロ作成のウィンドウを開く。次に、そこに表示されているその時点までのユーザーのコマンド使用の履歴の中から、マクロにしたい部分をマウスなどで指定し、それだけの操作で、マクロの実行が可能である。さらにマクロに名前をつけて登録することもできる。表示された履歴を必要に応じて編集することも、普通のEmacsの編集作業と同様に可能である。さらに登録したマクロは自動的にメニューの中から、参照できそこで実行することもできる。ただし、マクロのメニューからの実行は繰り返しが指定できないこともあって、動作の確認の使用以外にはそれほど有用ではない。図１に、マクロ使用の例をマクロ作成のためのコマンド履歴を表示するウィンドウを中心にして示す。

```
83667 (egg-self-insert-command . 83667)
3944 (minibuffer-complete-word . 3944)
3135(exit-minibuffer . 3135)
 (dired . 796)
 (create-screen . 104)
 (execute-extended-command . 988)
 (minibuffer-complete-and-exit . 973)
 (manual-entry . 24)
 (delete-window . 130)
 (scroll-up . 5985)
 (beginning-of-buffer . 693)
 (buffer-menu . 1947)
```

図 1－A

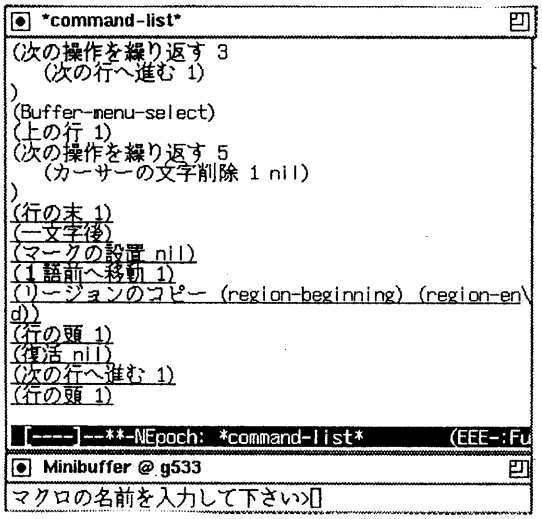

図　1－B

インターラクティブマクロの使用例

図1－Aは行末の数を先頭にコピーする課題を示す。図1－Bはマクロを指定するウィンドウの様子を示す。

2.4 インターラクティブマクロの評価

インターラクティブマクロはまだ試験版が実現されている段階であり、本格的な評価が行なわれたわけではない。しかし、これまでの筆者らの日常の仕事に使う中での主観的評価では、キイボードマクロに較べてマクロ作成が気楽にできるようになった。マクロ作成の開始をあらかじめ指定せずに後からできるようになったことの心理的な負担の軽減は少なくないものを感じる。短い簡単なマクロを使う時は増井、大和田[2]の報告しているダイナミックマクロが便利であるが、長いマクロを構成する時、特に、マクロを見ながら間違いを修正することができる点が使いやすい。全体に、キイボードマクロの機能と較べて、改善される点はあっても、特に大きく失うものはないために、インターラクティブマクロを実際に使って仕事をしている。また、キイボードマクロに較べて、直接Emacs-lispのコードを実行するために実行のスピードが早い点も実際の使用の中では現実的なメリットの一つである。

Emacsのユーザーではあるが、キイボードマクロを使ったことのないユーザー3人に対して、インターラクティブマクロとキイボードマクロと較べた小規模な実験的評価も行った。キイボードマクロマクロに較べて、試行をそのままマクロにできることなどの点で、おおむね好評であったが、コマンド使用の履歴の表示の中から、マクロにする部分を選びにくいという意見もあった。

2.5 インターラクティブマクロの今後の方向

インターラクティブマクロはユーザーの固有な状況に適合する環境をユーザー自身が作っていかなければならない。システムがもっと自動的に必要なマクロを構成していくようにしていくのが今後の基本的な方向である。ユーザーがマクロにしたい「意味のある」まとまりを表示したコマンド列の中で推定し示唆するメカニズムを付け加えていく予定である。

3. システム機能の学習支援

個人適応型のインターフェイスの実験的な試みの一環として、我々はユーザーがシステムの機能を学習することを支援するために、ユーザーのシステム使用の履歴に基づき、システムの側から能動的に有用な機能を提示するインターフェイスの検討を行っている。

前述のインターラクティブマクロは個人の状況に会わせてシステムの方を変化させる例であったが、反対に、ユーザー自身がシステムを理解し、システムを使いなれるというよ

うに、ユーザー自身が情報環境に適応し変化していくことも、情報環境が使いやすいものになっていくためには、必要なことである。将来、知能化の技術が進むことによって、システムが必要な機能を必要な場面でユーザーに代わって、自動的に発動させることができるようになれば、ユーザーがシステムを学ぶ負担は減るだろう。しかし、これはユーザーの意図や課題状況を理解する高度な知能をシステムが持つことが要求され、簡単には実現できない。当面はユーザー自身がまずそのような機能を学び、その機能が有効な場面でユーザーが機能を発動させることができることが必要である。

3.1 学習の停滞

情報機器をより便利なものにしようとして、情報機器がますます多くの機能を持つようになってきているが、ユーザーが学びきれずに、多くの有用な機能がユーザーに使われないままになるという問題が生じている。[3][4]。このことは、多様な機能を一つの環境の中に統合していこうとする統合化の傾向を考えれば、ますます問題になる可能性が高い。

多くの機能を情報機器が持つようになってくればくるほど、ユーザーが情報機器の持つ全ての機能をまず初めに全部学んでから情報機器を使いはじめることは現実的でなくなる。ユーザーは自分の仕事に応じてシステムを使えるところから使い始め、次第に高度な機能を学び、システムをより活用していくことができれるようになればよい。しかし、実際には、ユーザーがシステムを使いつつ次第に高度な機能を学習していくことはそう簡単ではない。現実には、システムが有用な機能を持っているにも関わらず、その存在さえも知らずに簡単な機能のみを使うだけでシステムを使い続ける場合が多い。ここではこの問題を「学習の停滞」と呼ぶことにする。

学習を促進させるための一つの方向はユーザーの能動的な学習を容易にすることであり、そのためにはマニュアルを整備し必要な情報を手にいれやすいようにすることなどが必要である。しかし、以下のような観察からも、ユーザーの側からの能動的な学習に期待するだけでは、学習の停滞を解消するのに必ずしも十分ではないと考えられる。我々はユーザーインターフェイスの実験環境として、メニューを使うことのできる Emacs環境を構成し、学生に提供している。このメニュー環境の狙いの一つは、ユーザーにメニューの形でEmacs の機能情報を提供し、学習を促進させることである。しかし、メニューを自分から調べ自分にとって有用な機能を探索していくユーザーは多くはない。また探索する場合も必要に応じて探索しているために、そこからユーザーが知らない機能を新たに学習することは少ない。実際、メニューを常時使っているユーザーでも、メニューに常に表示されている機能に気がつきもしないという経験が、よく報告される。

ユーザーが新しい機能を学習する契機はマニュアルなどを自分から読んで学ぶという能動的なものばかりではなく、人から教えられるなど受動的なものもある[4]。このことは、システムがユーザー積極的に働きかけ学習のきっかけを作ることが学習の停滞を防ぐ一つの方策になる可能性を示唆している。特に、ユーザーがまだ知らない機能をシステムの側からユーザーに学習しやすいタイミングで、積極的に提供することが効果的である可能性がある。このような考えを基本にして、我々は以下のような能動的な機能提示システムをユーザーインターフェイスの実験環境の一部として作成した。

3.2 能動的機能提示システム

我々の能動的機能提示システムのポイントはユーザーが新しい機能を学習するために、システムから積極的に働きかけることである。しかし、ユーザーに対するシステムからの能動的な働きかけが、ユーザーの仕事の邪魔になるようでは、学習支援システムとして受け入れられないであろう。一方で能動的であるという条件と、ユーザーの仕事の邪魔にならないという条件を満足させるために、適当なタイミングで情報を提示する必要がある。ここでは、ユーザーがシステムをしばらく使用していない状態だと判断される時に、システムがユーザーに対してシステムの機能についての情報をディスプレイの中央に目だつように提示することにした。

ユーザーがシステムの機能に関する能動的な情報提供を有効なものだと感じるためには、提供される情報がユーザーにとって実際に有効なものであることが重要である。能動的機能提示システムでは、支援の対象となるユーザーのシステムの使用履歴の情報を用いて、ユーザーがまだ学習していないと考えられる機能を調べ提示している。また、ユーザーのシステム使用の習熟度を考慮し、初心者用からと熟達者用までの機能を段階別に整理し、ユーザーの習熟度に応じて提示するコマンドを決めた。

3.3 コマンド使用記録の検討

どのような機能をユーザーに提示することが効果があるかを決定するための参考に、前記のNepoch上のユーザーインターフェイス実験環境を使っている数人のユーザーのコマンド使用の記録を検討した。そのうち二人のユーザーについては、半年以上にわたる使用の記録であり、キイストロークの数では約500,000になる。これらのユーザーは Emacsについての知識が豊富なユーザーではあるが、それでもコマンド使用にはかなり偏りがあり、Emacsの標準的なコマンドで使っていないものが多数あった。

3.4 能動的機能提示システムの構成

能動的学習支援の実験システムは、前記のインターラクティブマクロと同様に、Nepochのインターフェイス実験用の環境の中で作成された。この実験用環境を使っているユーザーはコマンド使用の記録が全て残されているので、それを使ってユーザーの使ったコマンドとその頻度情報が集計されたファイルを構成し、その情報に基づいて、ユーザーがまだ知らなくて、ユーザーのレベルに適合すると考えられる機能が選択され、その中から数個の機能がランダムに選ばれ提示される。提示のタイミングは、ユーザーの本来の作業を妨害しないために、休憩中であるときを狙うが、キイボードからの入力が１分以上ないときに、ユーザーが休憩中であると推定した。
図２にシステムが提示する機能の例を示す。

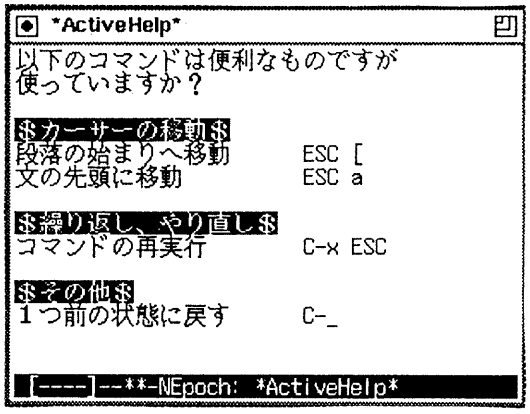

図２　機能提示のウィンドウ

3.5 能動的機能提示システムの評価

作成された能動的機能提示システムは数人のユーザーによって試験的に使われているのみであり、インフォーマルな使用感の報告が得られているだけである。その一人は文章作成、Electric-mail、Newsなどに常時Emacsを使っているユーザーであるが、いままで知らなかった機能をこれによって学ぶことができたと肯定的な評価を報告している。

一方、否定的な評価もある。ユーザーの一人はEmacsをよく使用するが限られた機能だけしか使わない初心者である。その初心者の感想では、知らない機能がいろいろ提示されるけれど、自分は見て参考にすることはないという否定的なものであった。

3.6 能動的機能提示システムの問題点と今後の方向

能動的機能提示システムがユーザーによっては一定の効果を持つ場合があることは分かったが、初心者に対して効果をあげることが難しいことも分かった。特に、システムの機能を学ぶのに、新たな概念的な理解を必要とする場合、機能の名前を提示するだけでは、十分ではない。分かりやすい説明を加えたり、アニメーションによる機能のデモを同時に提示するなど、ユーザーの注意を引き、分かりやすい提示にする必要がある。

4 おわりに

ここで報告した二つの個人適応システムは、現在のところ、適応の形態も全く異なり、ユーザーの履歴を用いているという点を除いて、共通点は多くない。しかし、将来的には個人情報に基づく、統合的なインターフェイス環境として統合されていかなければならないと考えている。その方向を目指して、より知的な機能を持たせるように改善していく予定である。

参考文献

[1] D. Hartmut, U. Malinowski, T. Kuhme and M. Schneider-Hufschmidt.
"State of the Art in Adaptive User Interface."
In M.Schneider-Hufschmidt et al (Ed.), Adaptivee User Interface. pp. 1-48.
Amsterdam:North-Holland. 1993.

[2] 増井俊之、大和田誠 :「操作の繰り返しによるマクロの自動生成」
情報処理学会研究会報告 Vol 93.
ヒューマンインターフェイス研究会 93-HI-48. pp 65-72. 1993.

[3] Carroll, J.M., and Rosson, M.B. :"Paradox of the
active user", in Carroll, J.M. (Ed.), Interfacing
Thought : Cognitive Aspects of Human-Computer Interaction,
Cambridge, MIT Press. 1987.

[4] 角田真二、三宅芳雄: 「ワードプロセッサーの使用における学習の停滞」
第６回ヒューマンインターフェイスシンポジウム論文集 pp. 79-84. 1990