インターフェイスの街角 (12)
適応型インターフェイス

増井俊之

いろいろな機械が使えば使うほど便利になればよいと思う人は多いのではないでしょうか。最近のかな漢字変換システムの多くは、学習機能によってユーザーの要求に近い変換をするように作られています。このように、システムが個々のユーザーの癖や特徴をなんらかの方法で検出し、徐々にユーザーに適応していくインターフェイスを**適応型 (adaptive) インターフェイス**と呼びます。

機械を特定の用途やユーザー向けに使いやすくするために、キー割当てなどを変えるカスタマイズや、マクロ定義、少量のプログラミングでルーチンワークを処理するエンドユーザー・プログラミングなどの手法がよく使われています。しかし、これらはいずれもユーザーによる作業を必要とします。このようなことをしなくても、機械が自動的に使いやすくなれば、それに越したことはありません。

最近、Web ページやニュース配送などの〝パーソナル化〟(personalization) がよく話題になります。パーソナル化により、自分の好みに応じたニュースを受け取ったりすることができるようになりますが、嗜好などをあらかじめ知らせなくても、自分の欲しいニュースが自動的に送られてくるようになれば便利です。適応型インターフェイスは、こういったことを実現する手法です。

適応型インターフェイスの種類

適応型インターフェイスという言葉は、さまざまな意味で使われています。パラメータを多少変更するだけのものからシステムの動作を完全に変えるものまで、各種の方式がありますが、大きく次の 2 つに分類できます。

- 繰返し操作の支援
- ユーザーの特性への対処

繰返し操作の支援

ユーザーが類似した操作を何度も実行するような場合、システムはそれを検出し、下記のような方法でその作業を支援することができます。

- マクロ作成支援

  似た操作が何度も繰り返される場合は、それらをマクロとして登録しておくと便利です。そのために、ユーザーの操作列からマクロ定義可能な共通部分をとりだすシステムが提案されています。3 月号で予測インターフェイスの一例として紹介した Dynamic Macro[10] では、テキストエディタでユーザーが〝繰返し実行キー〟を押すと、システムがそれまでの操作履歴から繰返しパターンを検出してマクロとして登録/実行します。Allen Cypher が Apple で開発した Eager[2] では、Macintosh の HyperCard でユーザーが似た操作を繰り返した場合、それを検出して、以後もその操作を繰り返すかどうかをユーザーに問い合わせます。

  あらかじめ用意されたアプリケーション知識を利用し、さらに高レベルの繰返し操作を検出するシステムもあります。UIDE[12] は、アプリケーション知識を表現する階層的データ構造を参照することにより、ユーザー操作列のなかでマクロ化可能な部分を抽出します。また、Flexcel[13] は、MS Excel の知識をもつエキスパート・システムがユーザーの Excel に対する操作をモニターし、マクロとして定義可能な操作列を検出します。
- 選択候補の並替え

  候補をリストにして提示するメニュー型インターフェイスや、かな漢字変換システムのように候補を順番に提示するシステムでは、よく使われる候補をプルダウン・メ

ニューの上部に配置したり、候補列の最初のほうにもってきたりします。Split Menus[11] は、プルダウン・メニューの項目のうち、よく使われるものを自動的に検出し、ほかの項目と分離してメニューの最上部に表示して選択しやすくするものです。カナダの University of Calgary の Saul Greenberg らは、階層的電話帳メニューにおいて、よく選択される名前がメニューの階層の上のほうに出現するようにした場合の効果について報告しています [4]。Kühme の Action Prompting システム [7] は、現在のコンテキストとユーザーの操作履歴をもとに、次に選択されそうなメニュー項目のリストを生成し、通常のメニューと別の位置に常時表示するようになっています。

- 予測される操作の提示
  3 月号でも紹介した Reactive Keyboard システム [3] は、UNIX のシェルのようなコマンド言語インターフェイスで、操作履歴の統計情報にもとづいてユーザーの次の操作を予測、提示します。
- 知的繰返し作業の支援
  MIT の Pattie Maes は、メモリベースの機械学習機構を用いた枠組みを提案しています。これは、多くの実例をもとに〝エージェント〟に学習させ、スケジュール調整や電子メールの仕分けを自動的に実行できるようにするものです [8]。

ユーザーの特性への対処

繰返し作業への適応は一時的なものですが、ユーザーの各種の特性にシステムを適応させるための手法も提案されています。

- 典型的ユーザーモデルとのマッチング
  ユーザーの知識や技能レベルに応じてオンラインヘルプやインターフェイスを変化させる試みがおこなわれています。Benyon が提案したシステム [1] では、ユーザーの知識や能力を判断してコマンド言語インターフェイスとメニューを使うインターフェイスとを切り替えます。UIDE では、アプリケーション知識を示すデータ構造の部分構造がユーザーの知識を示しているという考えにもとづき、ユーザーが実行したことのない操作に対して重点的にガイダンスをおこなう仕組みになっています。Flexcel では、操作履歴からユーザーの性質を判断し、そのユーザーに適応した動作をするようになっています。
- パラメータの調整
  文字認識や音声認識などの分野では、個人の書いた文字や声などの特徴パラメータをシステムが抽出する方式がよく用いられています。
- ユーザーの嗜好の抽出
  ユーザーの嗜好や感性のような数値化しにくい特性についても、適応のための枠組みが提案されています。前述の Maes の学習エージェントの考え方にもとづくスケジュール調整システム [6] では、エージェントは会議の時間の調整を通じてユーザーの時間の好みや会議の相手の重要度を学習していきます。私自身、個人の美的感性を表現するようなグラフ配置の評価関数を、遺伝的プログラミングの手法を用いて抽出する仕組みを提案しています [9]。NEC の神場知成氏による ANATAGONOMY システム [5][1]は、ブラウザ上に表示される電子ニュースをユーザーが読む様子からユーザーの嗜好を判断し、徐々にそれに合わせたニュースを表示するシステムです。ユーザーの記事に対する興味は、マウスクリックやウィンドウサイズなどから判断します。

問題点

適応型インターフェイスが目指す理想は高いのですが、かな漢字変換の学習機能のような比較的簡単な例を除くと成功例は少ないようです。以下に、適応型インターフェイスがうまくいかない理由をいくつか挙げてみます。

- 〝バンド幅〟問題
  適応型インターフェイス・システムはユーザーの操作履歴をもとに各種の判断をおこなうため、ユーザーとシステムのあいだのやりとりが少ないときは有効な判断ができません。たとえば、自動車を運転しているユーザー（運転者）がハンドルをわずかに右に切った場合、ハンドルを回した理由をシステム（自動車）が解析するのはほぼ不可能なので、適応も不可能です。
- モデルの問題
  適応型インターフェイスの構築にはアプリケーションのモデルとユーザーのモデルがよく使われますが、モデ

1 http://www.labs.nec.co.jp/freesoft/ANATAGONOMY/

ルの構築、使用にはいろいろな問題があります。アプリケーション・モデルは詳細なアプリケーション知識をもつ必要がありますが、複雑なアプリケーションの知識を効果的に表現するのは難しく、アプリケーションの作成によけいな手間がかかります。ユーザーモデルでは、ユーザーの各種の特性を数値表現した値の組がよく使われます。しかし、このようなデータの適応に対する有効性や正当性には疑問があります。モデルを作成するために、ユーザーへの質問に対する回答や操作履歴がよく使われますが、両者ともに信用できるとはかぎりません。

- 適応に対する不安感

  ユーザーが知らないあいだにシステムの適応が実行されるとユーザーは不安を感じることが多いので、このような仕様は避けるべきです。とはいえ、ユーザーに確認を求めてから適応動作をおこなったとしても、ユーザーはシステムの挙動や操作性が以前と変わることに抵抗を覚えるかもしれません。システムが適応しようとした理由をユーザーが理解できなければ不安感が増大しますし、理解していたとしても、ユーザーがその仕組みを操作できなければ苛立たしく感じるかもしれません。

- 適応の有効性への疑問

  方式が優れていても、適応の効果が小さければ無意味です。ユーザーの特性や状況に応じてシステムの動作を変えることが本当に有効な場合にのみ適応型インターフェイスを採用する意義があります。適応などしなくても、誰にでも使いやすいインターフェイスがあればそれが一番よいのですから、適応型インターフェイスそのものを疑問視する考えもあります。

- 人間の適応力との関係

  適応型インターフェイスをもつシステムは多くありませんが、世の中の多くの機器は、使えば使うほど使いやすくなります。機械自体が適応型でなくても、人間は日常的に使うことによって機器に適応してしまうからです。相当扱いにくい機械でもなんとなく使われているのは、人間の適応能力がきわめて高いからでしょう。

  したがって、人間が簡単に適応でき、そのことで大きな問題が生じないのなら、機械は非適応的（固定）にしておいて人間に適応させるほうが現実的です。たとえば、自動車のハンドルやブレーキの操作が運転者に適応するようになっていたとしても、ほかの車に乗るときにまごつくかもしれませんし、そもそも特定の個人に適応してしまった車は他人には運転しにくくなります。このようなケースでは、機械を適応的にするより、標準化して差異を減らすほうがよく、時間をかけても人間が機械に適応するほうが長期的には得策と考えられます。

以上に述べたように、深い推論が必要な適応型インターフェイスの構築はかなり難しいのが実状です。したがって、適応的な機械を開発するより、根本的に使いやすくしたり、明示的にカスタマイズしたり、あるいは人間が機械に適応するほうが都合がよい場合も多いと思われます。

しかし、自分だけが使う計算機で、カスタマイズが難しかったり、特殊な使い方をすることが多かったり、単純な適応手法が十分な効果を発揮する場合などには適応型インターフェイスが有効かもしれません。たとえば、SKK[14]のような単純なかな漢字変換システムでは、直前に変換した漢字が次回の変換で候補の先頭に提示されます。これはごく単純な手法ですが、よく使う漢字が表示されやすくなるので適応手法として効果的です。

超整理 Link 集

適応型インターフェイスの例として、SKK と同様な単純な適応手法を用いたプログラムを紹介します。

気に入った Web ページへのリンクを並べた〝リンク集〟ページを作って活用している人も多いと思います。毎日のように発見するおもしろいページをリンク集やブックマークに登録する場合、一般には新しく追加したリンクがリストの先頭に置かれます。この場合、それ以外のリンクは 1 つ後ろに移動するため、新しいけれどもあまり重要でないリンクが、旧いけれどもよく使われるリンクよりも上位に位置してしまうことがあります。重要なものを別のところに保存したり、不要なものをこまめに削除すればよいのですが、こういった URL の整理はけっこう面倒です。〝超整理法〟のように、アクセスしたリンクがつねに先頭に移動するようにしておけば、もうすこし使いやすくなるのではないでしょうか。これを繰り返していくと、結果的にユーザーの使用形態に適したリンク集ページができると考えられます。ここでは、一見普通のリンク集ページにみえますが、選択するたびに順番を変える手法を紹介します。

図 1　最初の画面

図 2　MindStorms をクリックしたあとの画面

図 3　リンク集データ hotlist.html

```
<br> Pilot拡張製品
<a href="http://www.qualcomm.com/pdQ/index.html">Pilot電話「pdQ」</a>,
<a href="http://www.symbol.com/palm/index.html">レーザースキャナ付SPT1500</a>
<br> ビルゲイツ
<a href="http://www.sannet.ne.jp/userpage/manatee/secretdiary/main/">ゲイツ秘密の日記</a>,
<a href="http://www.asahi-net.or.jp/~FV6N-TNSK/gates/index.html">頑張れゲイツ君</a>
<br> LEGOの
<a href="http://www.legomindstorms.com">MindStorms</a>
<br> Linux
<a href="http://www.debian.or.jp/index.html.ja">Debian</a>,
<a href="http://www.linux.or.jp/">Japan Linux Users Group</a>
<br> ズーミングシステム
David Foxの<a href="http://www.cat.nyu.edu/fox/tab/index.html">Tabula Rasa</a>,
商用の<a href="http://www.merzcom.com/">MerzScope</a>
```

ブラウザの最初の画面は、図 1 のようになっています。ここで ``MindStorms''をクリックすると、普通の Web ページと同じようにリンク先の LEGO のサイトにジャンプしますが、戻ってからページを再読込みすると画面が変化し、選択した MindStorms が先頭に表示されます（図 2）。封筒を用いた超整理法と同様、最近見たページはすべてリンク集の上のほうに表示され、旧いリンクも時間順に並ぶので、まったく整理しなくてもユーザーの仕事や興味に応じたリンク集ができます。

このような Web ページは、ブラウザのローカルマシンでは簡単には作れないので、サーバーマシン上で SSI (Server-Side Include) と CGI を用いて実現します。

超整理ページ

最初に、リンク集を表示するページを HTML で作成します。

```
<html>
<head>
<title>超整理Hotlist</title>
</head>
<body>
<table width=100% bgcolor=red>
<td><center><font color=white>Hot Links!
</font></center>
</td></table>
<!--#exec cmd="ssi-bin/hotlist.ssi" -->
</body>
</html>
```

リンク集データと SSI

リンク集は、HTML ファイル hotlist.html（図 3）に格納します。行頭の
を区切りと考えます。

リンク集の本体である hotlist.ssi は、リスト 1 のような Perl プログラムで、hotlist.html の〈a href=...〉の部分を ``hotlist.cgi''に置き換えます（$CGIDIR と $HOTLIST は、環境に合わせて書き換えてください）。

hotlist.ssi を実行すると URL が置き換えられ、ブラウザには図 4 のような HTML が渡されます。

CGI 処理

上記のリンクをクリックすると hotlist.cgi（リスト 2）が起動され、HTTP の Location 機能によって目的のリンク先にジャンプします。同時に hotlist.html を書き換えて、選択されたリンクを含むブロックが先頭にくるよう

リスト 1　hotlist.ssi プログラム

```
#!/usr/local/bin/perl
$CGIDIR = 'http://bird/~masui/cgi-bin';
$HOTLIST = '/user/masui/public_html/hotlist.html';

$c = 1;
open(hotlist,$HOTLIST);
while(<hotlist>){
  1 while(s/<a href=[^>]*>/SpEcIaL/i);
  $c++ while(s/SpEcIaL/<a href=$CGIDIR\/hotlist.cgi\?$c>/);
  print;
}
close(hotlist)
```

図 4　hotlist.ssi から渡される HTML

```
<html>
<head>
<title>超整理Hotlist</title>
</head>
<body>
<table width=100% bgcolor=red>
<td><center><font color=white>Hot Links!</font></center>
</td></table>
<br> Pilot拡張製品
<a href=http://bird/~masui/cgi-bin/hotlist.cgi?1>Pilot電話「pdQ」</a>,
<a href=http://bird/~masui/cgi-bin/hotlist.cgi?2>レーザースキャナ付SPT1500</a>
<br> ビルゲイツ
<a href=http://bird/~masui/cgi-bin/hotlist.cgi?3>ゲイツ秘密の日記</a>,
<a href=http://bird/~masui/cgi-bin/hotlist.cgi?4>頑張れゲイツ君</a>
<br> LEGOの
<a href=http://bird/~masui/cgi-bin/hotlist.cgi?5>MindStorms</a>
<br> Linux
<a href=http://bird/~masui/cgi-bin/hotlist.cgi?6>Debian</a>,
<a href=http://bird/~masui/cgi-bin/hotlist.cgi?7>Japan Linux Users Group</a>
<br> ズーミングシステム
David Foxの<a href=http://bird/~masui/cgi-bin/hotlist.cgi?8>Tabula Rasa</a>,
商用の<a href=http://bird/~masui/cgi-bin/hotlist.cgi?9>MerzScope</a>
</body>
</html>
```

にします。

たとえば、図 1 の状態で〝MindStorms〟をクリックすると、〝5〟を引数として hotlist.cgi が起動されるので、MindStorms へのリンクを含むブロックが先頭に移動し、ブラウザ上での表示が図 2 のように変わります。

おわりに

適応型インターフェイスは効果的な運用が難しいのでインターフェイスの万能薬にはなりにくいのですが、システムの機能が思いがけず自分の嗜好に適応していればちょっと嬉しくなることもあるでしょう。システムのスパイス的機能として用意しておくとよいかもしれません。

（ますい・としゆき　ソニー CSL）

[参考文献]

[1] Devid Benyon and Dianne Murray, Developing adaptive systems to fit individual aptitudes, In *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, pp.115–121, ACM Press, January 1993

[2] Allen Cypher, Eager: Programming repetitive tasks by example, In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'91)*, pp.33–39, Addison-Wesley, April 1991

[3] John J. Darragh, Ian H. Witten and Mark L. James, The Reactive Keyboard: A predictive typing aid, In

リスト 2　hotlist.cgi

```
#!/usr/local/bin/perl
$HOTLIST = '/user/masui/public_html/hotlist.html';
$urlno = shift;
# hotlist.htmlの読出しと選択URLの検出
open(hotlist,$HOTLIST);
for($lines=0;<hotlist>;$lines++){
  if(/^<br>/i){
    $bgnline[++$blk] = $lines;
  }
  $line[$lines] = $_;
  $blk[$lines] = $blk;
  while(s/<a href=([^>]*)>//i){
    if(++$c == $urlno){
      $bgnblk = $blk;
      $url = $1;
      last;
    }
  }
}
$bgnline[++$blk] = $lines;
close(hotlist);
# ブロックの並替え
for($i=0;$i<$lines;$i++){
  $s = $line[$i];
  $blk = $blk[$i];
  if($blk == 0 || $blk > $bgnblk){
    $nline[$i] = $s
  }
  elsif($blk == $bgnblk){
    $nline[$bgnline[1]+$i-$bgnline ⇒
    [$bgnblk]] = $s
  }
  else {
    $nline[$bgnline[$bgnblk+1]+$i-$bgnline ⇒
    [$bgnblk]] = $s;
  }
}
open(hotlist,"> $HOTLIST");
for($i=0;$i<$lines;$i++){
  print hotlist $nline[$i];
}
close(hotlist);
$url =~ s/^"//;
$url =~ s/"$//;
# 指定されたURLへのジャンプ指定
print <<EOF
HTTP/1.0 200 OK
Location: $url

EOF
```

（誌面の都合上、⇒ で折り返しています）

IEEE Computer, Vol.23, No.11, pp.41–49, November 1990

[4] Saul Greenberg and Ian H. Witten, Adaptive personalized interfaces - a question of viability, In *Behaviour and Information Technology*, Vol.4, No.1, pp.31–35, 1984

[5] Tomonari Kamba, Hidekazu Sakagami and Yoshiyuki Koseki, Anatagonomy: A personalized newspaper on the world wide web, In *International Journal of Human-Computer Studies*, Vol.46, No.6, pp.789–803, 1997

[6] Robyn Kozierok and Pattie Maes, A learning interface agent for scheduling meetings, In *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, pp.81–88, ACM Press, January 1993

[7] Thomas Kühme, A user-centered approach to adaptive interfaces, *Knowledge-Based Systems*, Vol.6, No.4, pp.239–248, December 1993

[8] Pattie Maes, Learning interface agents, In *Proceedings of the 1994 Friend21 International Symposium on Next Generation Human Interface*, February 1994

[9] Toshiyuki Masui, Evolutionary learning of graph layout constraints from examples, In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'94)*, pp.103–108, ACM Press, November 1994

[10] Toshiyuki Masui and Ken Nakayama, Repeat and predict – two keys to efficient text editing, In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'94)*, pp.118–123, Addison-Wesley, April 1994

[11] Andrew Sears and Ben Shneiderman, Split Menus: Effectively using selection frequency to organize menus, In *ACM Transactions on Computer-Human Interaction*, Vol.1, No.1, pp.27–51, March 1994

[12] Piyawadee Sukaviriya and James D. Foley, Supporting adaptive interfaces in a knowledge-based user interface environment, In *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, pp.107–113, ACM Press, January 1993

[13] Chritoph G. Thomas and Mete Krogsaeter, An adaptive environment for the user interface of Excel, In *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, pp.123–130, ACM Press, January 1993

[14] 佐藤雅彦「かな漢字変換システム SKK」、bit 1991 年 4 月号、pp.793–802