Desigining from Both Sides of the Screen: 3章

福井眞吾

2004.9.13

1　3章 無理強いしない:考える手間に気を配ろう

□ 表示するものは控えめに

□ よく使う機能を目立たせ、あまり使わない機能は隠せ

□ フィードバックをユーザに与えよう。処理が進んでいることを知らせよう

□ Preference 設定はすくなくしよう。よく考えられたデフォールトを用意しよう

□ 慣習に従おう (たとえそれがあなたの理想とするデザインと違っていても)

□ Widgetless Features を探求しよう

□ クリックを減らす方法として、個々の機能すべてにボタンを用意するやり方が考えられる。

□ これなら何でも 1 クリックでできる。でもこれは、「ボタンを選ぶ」という「考える動作」をユーザに課してしまう。肉体的な負担を減らすだけではなく、思考の負担も減らさなければならない。

□ 考えるためのエネルギーをユーザの本来の仕事に集中できるようにしよう。

□ 最近の TiVo、Replay、UltimateTV のような PVR(Personal Video Recorder) 製品は、思考の負担を減らすように作られている。「いつどのチャンネルを録画しますか」と聞く代わりに「どの番組を録画しますか」と聞いてくれる。連続物の番組の場合、1 回分だけ録画することも全ての回を録画することも指定できる。放送時間や局が変わっても気にする必要はない。番組が放送中止になった週は録画しない。PVR は、ユーザが本当に気にしなければならないこと (どの番組を見るか) だけ考えさせ、重要でない技術的な側面 (いつどのチャンネル) を気にしなくてよいようにしてくれる。

□ たいていの場合、これほど劇的に思考の負担を減らすことはできないが、いろいろな場面で少しづつ減らすことはできる。このことを、Steve Krug は著書「Don't Make Me Think!」の中で、『本来の作業を中断してインタフェースについて「考える」こと』と呼んでいる。この「考える」を最小にする方法を議論する。

2　表示する物はひかえめに

□ クリックと同様に、画面に並べる物のことも考えよう。メニューやボタンがちょっと増えるぐらいなんてことないと思うかもしれないが、「クリックの増加」と同様「画面がごちゃつく」ことも問題である。物が少なければ「選ぶ」作業が楽になる。

□ 例:Fig. 3.1 にはニューが A と B に 2 セットあるが、表の中から「Orange, Edinburgh, Cherry, Gold」を選ぶのはどちらが速いか? SetB のほうがずっと楽。SetA では、目的とする要素が含まれる可能性のあるカテゴリが複数あると探す手間が増える。特にソフトウェアのメニューの場合は、カテゴリー分けのあいまい性が高いので、この問題がより顕著になる。UI の他の部品も「少ない方がよい」というルールが成り立つ。画面を眺めるのにほんの少ししか時間がかからないとしても、「見つける」という動作によて、本来の仕事の流れが中断される。

□ アプリケーションが高機能になるとより多くの部品 (ボタン、メニュー、チェックボックスなど) が必要になると思うかもしれない。しかし、少ない部品で実現できれば、ユーザは求める機能を楽に見つけられるし、そうなればあなたが一生懸命作った機能を使ってもらえる可能性も高まる。

□ Fig3.1 は極端な例だと思う? Fig3.2 の Outlook2000 Calendar の例では、メインメニュー 7 個、メニュー選択肢 69 個 (内 16 個はサブメニュー)、サブメニュー選択肢 68 個もある。メニューを選ぶと、42 個のポップアップウィンドウが開き、それらはさらに子供のポップアップウィンドウを開く。画面上部にはタスクボタンが 12 個、アプリケーション呼び出しボタンが 6 個、フォルダ表示部、予定欄、今月、来月の概要、タスクパッドがある!

□ これに対して、Palm の DateBook 画面はシンプル。メインメニュー 3 個、メニュー選択肢 18 個、サブメニュー

なし、ポップアップウィンドウ 6 個。画面上にはボタン 3 個と、choice box2 個、今日の予定画面だけ。

□ 2 種類のカレンダーソフトを使い比べてみると、Outlook はいろいろなことができるが、Palm の方がずっと効果的だとわかる。ユーザは脅かされることとなく、よく使う機能はすぐに、あまり使わない機能も楽にみつけられる。

□ Web サイト、特に、ホームページ (トップページ) は、この原則を守るのに苦労している。皆が、トップページにリンクを置こうとするからだ。でもそうすると自分のところに来てくれるチャンスは相対的に少なくなる。

□ 画面上における要素の数は極く限られていると考えるようにすれば、機能を追加することに注意深くなる。機能を追加することによる UI のコストを考えるようになれば、あなたはよいインタラクションデザイナーのように思考できるようになる。

3 よく行う作業は見せ、あまりやらない作業は隠す

□ 「機能を減らす」以外に、画面のごちゃごちゃを減らすのにもっとも効果的なのは、あまり使われない機能を隠すことである。よく使われる機能だけを画面に配置すれば、初めての人も怖がらずにそのソフト使い始められる。ユーザは、強い動機付けがないと、どのような機能があるか探索したりはしない。だから、画面をごちゃごちゃにすることなく、よく使われる機能を明快に示すと同時に、他に機能があることをほのめかすことができればいいんだが。

□ 良い例を 1 つ。航空会社の窓口でもっとも多く行われることは、「旅行の計画 (発着日時、値段、予約)」と「運行状態 (今日のフライトの発着時刻) の確認」である。他のこと (割引商品の調査、なくなった荷物の捜索、勤め口、Frequent Flyer 会員申し込みなど) はずっと頻度が少ない。Web サイトをデザインするなら、最初の 2 つをすぐに見つけて作業をはじめられるようにしたいだろう。

□ 航空会社の Web サイトを比較してみよう。Fig.3.4 のユナイティッド航空の Web サイトでは、「計画」と「運行状態確認」と、「マイレージ確認」が目立つように配置されていて、クリックで画面遷移をすることなく、直接これらの情報を調べることができる。一方、あまり使われない機能は、「クリック」によって画面遷移してアクセスするようになっている。

□ 他の航空会社のページはずっと出来が悪い。例えば、Fig3.5 のアメリカン航空のページでは、情報がごちゃまぜだ。「アメリカン航空のニュース」と「旅行の計画」がトップに配置されていて、しかもそれらがあまり明確に区分されていない。あなたは「運行状態」を調べる場所をすぐに見つけられますか。ページの残りの部分はあいまいな名前のプルダウンがいっぱいあり、やりたいことの場所をみつけるのは大変。例えば、マイレージを確認するのはどれ?

□ Fig.3.6 のルフトハンザのページでは、「計画」部分は目立つようにオレンジ色の枠に囲まれて画面右側に配置されているが、中央上部は相互に関係のない機能がならんでいる。このページで「運行状態」をすぐに見つけられますか?

□ 「よく使う機能を手近に」という原理は、小さなデバイスではより重要である。これらはユーザが外出時に他の作業をしている最中に用いられる。だから、必要なことをすぐやれるようでなければならない。例えば、携帯電話が着信したときは「電話に出る」がすぐ出来る必要がある。どのボタンを押しても電話に出られる機種がある。これはいい。しかし、電話を切るには専用のボタンを探す必要がある。これは通話中に「切る」以外の他の作業をする可能性があるからだ。Nokia6162 では、フラップを開いて電話に出られ、閉じれば電話を切れる。何の努力もいらない。これはいい! 携帯電話をしまうときは必ずフラップを閉めるんだから。

□ Nokia の携帯電話では、Short Message(簡易メール) 操作画面のソフトキー (キーの意味が画面に表示される) のデフォルト設定が秀逸である。左ソフトキーを繰り返し押すことで、メールを次々に読むことができる。すでにメールを 1 通読んで保存してある状態で新しいメッセージが来ると、左ソフトキーは「Read」に変わる。押すとメッセージリストが表示され新着メールが選択された状態になる。ここで左ソフトキーを再度押すとメッセージを読める。読み終わると「Options」に変わり、デフォルトは「次のメッセージを読む」になっているのでソフトボタンを再度押せばよい。

□ 「自動車を運転するとき、よく使う機能は道路から目を離さずに操作できるべきだ!」と主張する友人がいる。Fig.3.9 は彼のアイディアを図にしたものだ。手を置くホームポジションをダッシュボード上に用意しておく。そこは手探りですぐ場所がわかること。そこに手を置いてラジオを選曲したり温度調節をする。操作に対して音声フィードバックがあるとなおよい。

4 フィードバックを与える:処理の進み具合を示す

□ 人間同士の会話では、相槌や仕草で「ちゃんと聞いているか」や「どれくらい理解できているか」を相手に伝える。これらがないと、会話の中身から、言葉のやりとりへ意識がそれてしまう。コンピュータも同様だ。通常、ユーザの要求にそのまま従うことでこれをおこなう。例えば、文字入力や削除は即座に画面に反映される。ユーザの要求を即座に実行できないときや結果が見て分からないものの場合、執事がするように「要求を伺いました(かしこまりました)」と返答すべきである。そうしないとユーザが再び要求を出し、プログラムは混乱に陥る。

□ よく用いられる方法は、作業に時間がかかるときに「タスクバー」を表示することである。別の例として、描画に時間がかかる場合に、最終結果をまとめて表示するのではなく、描画している途中状態を見せる方法がある。この方がユーザは処理が速く感じる。Web で画像よりも先にテキストを表示するのも同様である。また、粗い画像を最初に表示し、だんだん解像度を上げるというのもこの例である。

□ 時間がかかる作業は中断できるようにしておくべきである。もともと望んでいない作業を待つことほど時間が長く感じられることはない。読む気のない Web ページのダウンロードを停止する「Stop」ボタンは「神様からの贈り物」のように感じられる。よくできた音声認識システムでは、システムがしゃべっている最中にコマンドを投入することができる。

□ 要求を即座に実行できるけれども、これといったフィードバックが存在しないケースもある。この場合「あなたの要求をちゃんと実行しました」ということをユーザに知らせる必要がある。

□ 例:通話を終わるとき、回線が切れたことを明確に示す手段がない。⇒ビープを鳴らしたり、通常の画面表示に戻すことでこれを表す。ボリューム調節の際、目盛りが変わるだけではなく音を実際に鳴らすシステムもある。

□ 複数のステップがある処理をユーザに行わせるときも、フィードバックが役立つ。例:チェックアウト (精算) やアンケートへの回答など。フィードバックにより、ペースをコントロールでき、全体でどれぐらいかかり、今どこまできているかが分かる。

□ Fig3.10 は 800.com の例:支払い方法の選択、見直し、レシート受け取りができることがわかる。

□ Fig.11 は Harris Poll のアンケート画面:プログレスバーがある。1 ステップでどれぐらい進むかわからない点はよくないが、でもなにもないよりはずっといい。

□ 音声を用いたインタフェースでは、目に見える形でフィードバックは行えないが、会話のテクニックを使える。

□ 例:Schwab の電話による株価応答システム:

- User:「Sun の株価はいくら?」
- Computer:「その名前に該当する商品は 7 つあります。いつでも証券のフルネームをおっしゃっていただいてもかまいません。これから読み上げるリストを聞いて、しぼりこんでください。」

□ リストはアルファベット順ではなく、よく利用される順になっている点もよい。「もう一度証券のフルネームを行ってください」と言う代わりに、可能性のある証券を示すところがよい。さらに、途中いつでも言えるようになっている点が優れている。少し改良するとすれば、「『Sun』に該当する」というように、ユーザの入力を復唱するとよい。

□ 「ボタンがちゃんと押されましたよ」というフィードバックも重要。例えば、Canon のデジカメでは、画像のページめくりのたびに「クリック音」を発する。似た写真が並んでいるとき役に立つ。Palm もクリック音を出す。研究によると、音によるフィードバックを付けることでボタンのサイズを小さくできることがわかっている。

□ 音声フィードバックの他の例:Panasonic のコードレス電話は、充電台に乗せると音と光でちゃんと正しく置かれたことを知らせてくれる。

□ 音によるフィードバックは、情報を得るための動作をユーザが自発的にしなくても良い点がすばらしい。ユーザはこれにすぐなれるので、異常があるとすぐ気づく。音と光という 2 種類同時という点もよい。耳や目が不自由な人もいるし、普通の人でもノイズで聞こえない場合や、物に隠れて見えない場合があるので。

□ Tivo:音を多用している。

□ ボタンごとに違う音色を出す。

□ 内容に沿った音を出す。

- リストの最後までいってさらにスクロールしようとするとティンパニーの低い音がする。いかにも底についたと感じられる。
- 早送りのスピードを変える際、速くなるほど高い音程で「バ・ブー」と鳴る。同時に、1 個〜3 個の右向

き三角マークをディスプレイに表示し、カーソルがだんだん速く動く。

5 「Preference」はなるべく少なく、よく考えられたデフォルト設定を用意せよ

□ 「Preference」や「Option」でUIをカスタマイズできるソフトが多い。これはおそらく、開発者メンバー同士が「たいていの人はこれをのぞむ」ということに合意できなかったからである。だから個々のユーザに選択させることにしたのである。

□ Preferenceは、ユーザの好みでソフトの動作を自由にカスタマイズできるというすばらしいものだが、大変重い精神的な労力をユーザに課してしまう。本当は、ユーザは自分の作業をやりたいだけなのに。

□ Preferenceは、ユーザの常識に反することにもなる。多くの技術分野では、個々のユーザのニーズに合わせて変えるなんてことはできない。(例:スイッチを倒すだけで左利き用に変わる缶切りなんて作れない。別のを買うしかない。) だから、ユーザは「動きを変えられる」なんて期待していない。

□ ユーザは、「アプリケーションとUIは別個のもので、UIを好きなようにカスタマイズできる」なんて思っていない。色や形といった見栄えを変更することはあっても、「ソフトの振る舞い」を変更しようとは思わない。

□ ユーザは、ひどい目に会うたびに、すこしづつ我慢の限界に近づく。プログラムの設定を簡単に変えられるということを知っている専門家ですら、問題に不平をいいながらつきあい、我慢できなくなってようやくPreference設定を変えようとする。(Ellenは、Officeアプリケーションがファイルをセーブ/オーピンするデフォルトフォルダの設定をかえるのに2年かかった。一度も使わない「My Document」がいつも出てきて悩まされていた。彼女はデフォルトを変えられるのは知っていたが、変えるためにその時やっている作業を放り出して、Preference内の設定場所を探すのは、大変大きな努力が必要だった。)

□ 「ユーザはだれもシステムの動作を変えるためにPreferenceを設定するなんてことはない」と思うべきである。だからデフォルト値の設定は注意深く行うこと。よくある一般的なケースを考えよう。特定の人が望むかもしれないという特殊ケースは考えるな。

□ Fig.3.12はICQの何十とあるPreferenceシートの1例。めったにこれを変えたりしない。例:Display Email notification on Contact List if Email is assigned to user" (fukui:よくわからん)。メールに気づかないとしても、その問題を解決するためにPreferenceを見ようなんて思わない。ICPのPreference設定には山と項目があるが、それを用意するためのプログラムの手間を正当化する理由なんて存在しない。

□ 動作に関するPreference設定をゼロにすることを目指すべきだ。それができないとしても、このような心掛けは、設計者を「Preferenceにするかどうかよく考える」ように仕向けてくれる。。

6 慣習に従おう (たとえそれがあなたの理想とするデザインと違っていても)

□ 通常、他のアプリケーションも動作しているプラットフォーム上に自分のシステムを構築する。プラットフォームはユーザとのインタラクションになんらかの慣習 (しきたり、作法) をもっている。それに従うのが一番よい。

□ ユーザは、アプリケーションを使ううちに慣習を覚え、その動作を期待するようにもなる。

- 例:Webページの上部や左にカテゴリがならんでいる。
- 例:携帯電話ユーザは、左ソフトボタンで「選択」し、右ソフトボタンで「戻る」
- 例:Palmでは画面上に機能がならんでいるので、ユーザはめったにメニューを見ない

□ ユーザは、プラットフォームとアプリケーションをあまり区別できない。あなたのアプリケーションの動きが他と違っていると、アプリケーションのせいだとは思わず、その装置 (PC,PDA,携帯電話) がデザインの一貫性を欠いていると思う。

□ 例:Windowsでは、Cnt-C(コピー),Cnt-X(切り取り), Cnt-V(貼り付け)。自分のアプリケーションでCnt-X(マーク付け)としたら、ユーザは混乱してCnt-Xを使わなくなる。

7 「Widgetless Features」を目指そう

□ 「Widgetなし」というのは、我々が目指すべき理想である。「Widget」はボタン、スクロールバー、メニュー

といった画面上の部品。これなしにはやっていけないが、「機能が必要なときに利用できるようにする」ことで、Widget を使って機能を呼び出すことを不要にできるケースがある。

- 例:名前の補完機能 (数文字タイプするとそれにマッチする候補を示す機能)
- 例:入力場所を最初のテキストフィールドに自動的に置く

□ Widgetless Feature は、画面をごちゃごちゃにしないことによって思考の負担を最小にし、クリックを減らすことによって肉体的な負担を減らす。

□ 他の例:デスクトップソフトの auto-scroll 機能:ドロップ先の Window でマウスカーソルをボーダー付近に持ってくると自動的にスクロールする。

□ Garmin の eMap という GPS 端末:

□ キーボードがないのでテキスト入力にはアルファベットをぐるぐるまわして選択しなければならないが、通りの名前 (注:street name:日本での「町名」に相当) を入力するときは、可能な選択肢だけがでてくる。

□ Fig3.13 「WALLE」の次にくる文字の候補は「R」なので、↓を押すとこれが最初にでてきて (A〜Q はでてこない)、「WELLER」にマッチする候補が表示される。もう一度↓を押すと次候補の「s」が出る。

□ Netscape と IE では、アプリケーションのダウンロード&インストール操作が簡単になっている。通常はダウンロード後にそのファイルを見つけ出してダブルクリックする操作が必要なのだが、これらのソフトでは、ダウンロード後に「インストールします」に同意するだけでよい。

□ Yahoo!Messenger では、ソフト起動時にバージョンアップするかどうか聞いてくれる。Yes なら自動的にプログラムを閉じて、新版をダウンロードし、インストールして、先ほどと同じ場所にプログラムを起動してくれる。

□ Microsoft Word のスペルチェック機能も Widgetless の一例。従来のスペルチェックでは、その機能を別途呼び出し、別ウィンドウで置き換えや辞書登録といった操作をおこなっていた。Word のスペルチェックでは、文字を入力すると同時に認識できない単語に赤い波線がつく (Fig3.14)。直接修正することも、マウス右クリックで候補を表示することもできる。そのつづりが正しければ、赤い波線を無視することもできるし、辞書に登録することもできる。とても簡単に使えるし、すぐ修正するか後回しにするかを自由に選べる。

□ Word は「hte」を自動的に「the」に直してくれる。Word は typo の辞書をもっている。自分で辞書に追加することも可能。typo の自動修正を拒否することも簡単。Backspace で消して書き直すだけでよい (fukui:本当ですか?)。

□ Canon のデジカメには、複数枚の写真をつなぎ合わせてパノラマ写真を作れる機能がある。1 枚撮るととその右 1/3 が画面に残り次の写真をうまく重ねて撮れる。撮影時の設定も覚えているので同じ設定で撮れる。

□ Widgetless Feature のやりすぎには注意しよう。間違って解釈してユーザの望まないことをやってしまうより、なにもしない方がずっとよい。ユーザが望まないことをしてもその害が少なく、たいていの人のたいていの場合に手間が省ける機能を追い求めよう。

□ やり過ぎの例:MS Office Assistant の paper clip の help system

□ ドキュメントを「Dear」で書き始めると、目をぎらつかせて登場し、「手紙を書くのを手伝いましょう」押し売りしてくる。この機能はとても嫌われていたので、ライフルの照準に Paper Clip が描かれた T シャツが販売されもした。MS はついにこの機能を削除した。

□ ひかえめにしよう。「お手伝いしましょうか?」と言ってユーザの仕事の流れを中断するのではなく、ユーザに聞かなくても正しい動作ができる場合を見つけよう。