.NET Architecture Forum

真の“使いやすさ”を目指して

ユーザーインターフェイス論序説

第7回 実装に向けた方向性の検討

マイクロソフト株式会社
デベロッパーマーケティング本部テクノロジーエバンジェリズム部
デベロッパーエバンジェリスト
西谷 亮 *NISHIYA, Ryo*

Technology Tools

- ❑ Visual Basic .NET
- ❑ Visual C# .NET
- ❑ SQL Server 2000
- ❑ Oracle 9i
- ❑ Access 2002
- ❑ ASP.NET
- ❑ Internet Information Services
- ❑ Other:

Level

★★☆☆☆

Samples

はじめに

前回は、UIのユーザビリティを数値化する、という点に着目して話をしました。使用感を数値化することで、成果物の比較とその結果は認識しやすくなります。しかし実際は、利用者の主観や経験によって、その結果は大きく左右されがちです。つまり、ユーザビリティの数値化は、評価手法としては有効であっても完全なものではないことを認識しておかなければならないのです。

すなわち、ユーザビリティラボのような実際の検証作業と、数値的なデータを包括的に判断しなければ、本当に使いやすいものを開発し提供してゆくことはできない、と言うこともできるでしょう。

今回は、“利用しやすさ”を数値化するもうひとつの方法を紹介した上で、マイクロソフトも採用しているユーザーインターフェイスデザインの手法をご紹介することにします。

使用感は部品点数に依存する？

前回紹介したユーザビリティの数値化は、アプリケーションの画面上での操作に要する時間を測定するという手法でした。今回は操作の前段階、つまり画面デザインのフェーズでその操作性を客観的に判断する方法です。

まず、「ヒックの法則」を紹介します。

$$T=\log_2(x+1)$$

ここで、Tはある特定の操作を決断するまでの時間、xは選択肢の数を表わします。

つまり、この式によって、画面上の選択肢が操作にどのような影響を与えるのかを検討することができます。

たとえば、画面上に選択肢が多ければ多いほど選択に時間を費やし、結果的に操作性の低下を招いてしまうということがわかります。

一般的に選択肢が少なければ、迷うことはありません。簡潔に作業ができることは直感的にもわかります。当然、その逆もまったく同様です。この式は、

直感的にわかる、いわば常識的なことを法則という形で証明したものであると考えればいいでしょう。

前回紹介していた“ウォルホールドによるフィッツの法則の改良”と組み合わせて判断すると、

「画面上の各要素は理解しやすいように操作範囲（許容誤差）を大きめに設け、直感的に理解できるような工夫をすることで操作性の向上は可能だが、部品点数や選択肢を増やしたのでは、ユーザビリティの向上は望めない」

ということになります。

これらのことから、使いやすい画面とは、行なおうとしている作業を明示的に、できる限りわかりやすく配置し、必要な操作のための要素は許容誤差を設け、そして、それらの選択肢は最小限度にとどめる、ということになるでしょう。

さらに突き詰めると、その画面の中では、可能な限りひとつの作業で完結するようにし、必要な作業を明示することで次の操作をナビゲーションするようなものが望ましいと結論付けられます。

アプリケーションそのものがどのような場面で使用されるのか、どのようなスキルセットを持ったユーザーが利用するのか、その目的は……など、さまざまな要因によって結論は変わってくるかもしれません。しかし、基本的なデザインの方向性は、意外にも同一といえるかもしれません。

帰納的ユーザーインターフェイスガイドライン

これまで解説してきたさまざまな法則や研究成果などから、方向性として考えられているものに「帰納的ユーザーインターフェイスガイドライン（Inductive User Interface Design Guideline）」というものがあります。このガイドラインはマイクロソフトの製品の中でも一部取り入れられています。

ここでは、実際にこのガイドラインの中でどのようなことを目標として掲げられ実装を進めているのかを検討してみることにします。

そもそもIUIとは？

この“Inductive User Interface”という考え方はその名が意味しているとおり、帰納的なアプローチによるユーザーインターフェイスデザインの実現を目標としています。

つまり、システムやアプリケーションなどにおいて、個々の特殊な事情や命題の集まりを検討し、その中から共通する性質や関係を取り出すことからはじまります。次に、これらの要素を検討し、一般的な命題や法則を導き出し、その内容を画面デザインへ反映させてゆくことになります。

「帰納的」といっている部分で理解しにくい部分もあるかと思います。そこで、ここでは問題の認識とその解決の流れからIUIを紹介しましょう。

共通的な問題に認識

ソフトウェアにおいて共通する問題を考えてみます。

開発者や設計者という立場に立つと忘れがちなのが、ユーザーにおける以下の問題でしょう。

「ソフトウェアは難しい」

利用者の立場に立って、または使用経験がないアプリケーションに向かったとき、あなたはどう思うでしょう。どこから手をつければよいのか迷うはずです。しかし、それでも「たいていのソフトウェアは、利用しやすいようにデザインされているはずだから、たいしたことはないはずだ」という気持ちもあるかもしれません。

開発者のみなさんや多数のアプリケーションを触ったことがある方であれば、必ずしもそうでないことは容易におわかりになるでしょう。

さまざまな立場によってソフトウェアへの向き合い方は異なります。初めてのケースであれば、前述のような心理が働くかもしれません。これらを整理すると、以下のような状況が考えられます。

・たいていのソフトウェアは、利用しやすいようにデザインされていると考える
・実際には、論理的なデザインが行なわれていないケース