.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3回 いつ誰のために、UIは存在するのか

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 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

これまで、さまざまなアプリケーション環境がもつ潜在的な問題点について解説してきました。操作性の低下やシステム自身が求めている利便性を追及しようとすると、犠牲になる部分が生まれてくるという点は、その議論の中心でありました。

では、ユーザーインターフェイスを取り上げたときに、開発者が気をつけなければならないこととは、いったい何なのでしょうか。

立場による思惑の違い

一般的に開発者の視点で「生産性の向上」が語られるとき、そこでは「開発生産性の向上」を意味するケースが多いはずです。必要とされるシステムを、より短期間でより簡単に構築することができれば、アプリケーションを利用するエンドユーザーは、ビジネスチャンスを逃すことなく、新しいシステムを投入できるというわけです。

もちろん、この考え方は間違いではありません。むしろ大正解です。必要なものを必要なときに利用できることは非常に重要なことです。Visual Studio .NET 2003をはじめとする開発環境が提供される際、“開発生産性の向上”は、よく挙げられるメリットです。

しかし、忘れてはならないのが、

「生産性」を語るときは「開発生産性」だけを念頭においてはいけない

ということです。実は、ここに開発者とシステムを利用するエンドユーザーの、思惑の違いとそれに起因する最終的なゴールの違いが存在しています。

開発者はシステムの実装などに関する知識は長けています。もちろん開発作業そのものに関しても専門的テクニックを駆使することになるでしょう。そういった環境で、ユーザーインターフェイスをデザインし、ビジネスロジックを構築し、システムの稼動を確認すると、“使いやすいアプリケーションが完成した”と思うはずです。

そこでもう一度考えなければならないのは、「誰にとって使いやすいのか」ということです。

開発者自身は、実装やデザインを進めているのだから、新しいアプリケーションに触れる機会も多く、理解をしているはずなので、使いやすいのは当然でしょう。後述しますが、開発に関わっている以上、そのアプリケーションやシステムに対してよく認識しているため、エンドユーザーの立場に立つことが難しくなっているのです。

「何かをしたい」と思ったとき、実装やデザインを担当した開発者は、その方法を十分に理解できているため、なんのつまずきもなく完了させることができるはずです。

しかし、何も知らずにはじめてそのシステムに直面した場合、本当に同じように作業を進めることができるでしょうか。何も知らない状態であるエンドユーザーの視点や、純粋に利用する側の立場に立つことができていないのではないか、という点を認識しておかなければならないのです。

ここで、もう一度「生産性」というキーワードに話を戻します。

「生産性の向上」というキーワードは、エンドユーザーの立場からは何を連想させるでしょうか。

・作業時間の短縮
・導入時間／トレーニングの削減
・システム導入に伴なう収益率向上

といった答えが返ってくるのではないでしょうか。

つまり、システムに求めるものは、最終的にその利用によってもたらされる効率化や収益率の向上などだということがわかります。開発者が求めているような「迅速な開発」や「容易な開発」といったキーワードは、あくまでシステムを構築する側の立場における考え方であり、エンドユーザーにおける考え方とは異なるのです。さまざまな事例インタビューなどを読まれている方であれば、その違いに気付くはずです。

両者の思惑の違いをとらえると、潜在的な問題点が見えてきます。「生産性」というキーワードだけでも、その立場によって思惑も異なれば、その言葉の持つゴールも微妙に異なるわけです。

では、その両者の思惑の相違を具体例を使って検証してみることにしましょう。

開発者にとっての生産性

開発者にとっての生産性を語る際、先にも紹介したように「迅速な開発」や「容易な開発」を挙げることが多くあります。そこでVisual Studio .NETにおける開発生産性の向上について考えてみましょう。

ここでは2つの具体例を紹介します。先日開催された「.NET Developers Conference 2003－PDC Highlight－」の基調講演の中でも紹介しているので、ご覧になった方もいるかもしれません。

ひとつ目は、セキュアな通信の元でWebサービスを実現しようとしたときに必要となるコード量を、現在のVisual Studio .NETおよびWeb Services Enhancementsを利用した場合と、将来登場する次期バージョンのVisual Studio “Whidbey”を利用した場合で比較したものです（リスト1～3）。

もうひとつは、同様に印刷を実現するためのコードが現在と未来においてどのように変化するかを示したものです

リスト1：Visual Studio .NET（C#）のコード

```
class HelloService {

 [WebMethod]
 public String Hello(String Greeting) {

  X509CertificateCollection collection = new
    X509CertificateCollection();
  IntPtr blob = Marshal.AllocHGlobal( _
    Marshal.SizeOf(typeof(CRYPTOAPI_BLOB)));
  IntPtr data = (IntPtr)((int)blob +
    Marshal.SizeOf(typeof(CRYPTOAPI_BLOB)));
...
  SeqAckRange range = new SeqAcknRange(id, low, high );
  SeqAckRange[] ranges = { range };
  ReliableQueue.ProcessAcks( ranges );
...
  hr = pITxDispenser->BeginTransaction (NULL,
       ISOLATIONLEVEL_SERIALIZABLE, 0, pITxOptions,
       &pITransaction);
...
  return Greeting;
 }
}
```