実際に作りこんでわかるWebアプリケーション

.NETによるビジネスアプリケーション開発講座

第1回 カタログと購入コンポーネントの作成

日本ユニシス株式会社
.NETビジネスディベロプメント
尾島 良司 *OJIMA, Ryoji*

Technology Tools

- ☑ Visual Basic .NET
- ☑ Visual C# .NET
- ☑ SQL Server 2000
- ❑ Oracle 9i
- ❑ Access 2002
- ☑ ASP.NET
- ☑ Internet Information Services
- ☑ Other:
  - Visio for Enterprise Architect
  - Visual SourceSafe
  - NUnit

Level

★ ★ ★ ★ ★

Samples

・この記事で取り上げたソースコードおよびサンプルプログラムは、付録CD-ROMの\DOTNET\TANKIディレクトリに収録しています。

\DNSTORE
サンプルプログラムとソースコード

はじめに

Standish Group International, Incの調査によれば、2000年に実施されたソフトウェア開発プロジェクトで「成功」したものは28%しかないそうです[注1]。残りの数値は「問題を抱えている」が49%で「失敗」が23%。

このままではプログラマに明るい未来はありません。なんとかして成功率を上げなければ！　そのために必要なのは何か？「優れた道具を使う」。なるほど.NET FrameworkやVisual Studio .NETは非常に便利です。「技術力を付ける」。本誌の読者なら“言わずもがな”。

「では、優れた道具を高い技術力で使えばそれで十分なのだな？」

否。それだけでは満足のゆく成功率にはなりません。ビジネスアプリケーション開発は特殊な世界であり、特殊な考え方やプロセス、そして環境が必要なのです。

「その特殊性を記したものがいわゆる方法論ではないのか？　いくつも方法論を読んだけど抽象的すぎて現場では使えなかったぞ」

おっしゃる通り。現場には現場の論理があります。方法論は“雲上人”が書いたもの。雲の上の法則をそのまま地上に適用するのは無理があります。現場には現場の、プログラマにはプログラマのやり方が必要なのです。

というわけで、本連載では、サンプルアプリケーションの作成を通して、日本ユニシスの職業プログラマが実際にビジネスアプリケーションを構築する際のやり方を書いてゆきます。

作成するアプリケーションはずばり「ECサイト」。作成したソースコードは付録CD-ROMに収録しています。ソースを見て、ぜひ実際に動かしてみてください。

注1）Standish Groupのレポートにおける「成功」とは，納期と予算をまもり、予定された機能群の過半数を実装したという意味です。

プログラマのための開発方法論

プログラミングをしたくてうずうずしているみなさんには申し訳ないのですが、最初に「開発方法論」について簡単に触れさせてください。

ここでは、次のような項目について順に説明します。

- **プロセス**
- **アーキテクチャ**
- **開発環境**

どんな事柄でも準備作業は必要なのです。開発のための準備作業として「プロセス」と「アーキテクチャ」、そして「開発環境」について考え、具体的な形にしてみます。

プロセス

いきなりですが、ちょっとぶちかましてみましょう。

ドキュメントの精度を高めれば高めるほど、プログラムは不正確になってしまう

プログラマなら誰でも知っているように、ソースコード以外の方法でプログラムを詳細に記述／検証することは不可能です。

計画を詳細に定めれば定めるほど、納期とコストを守れる可能性が減ってしまう

プログラミングのような複雑な作業が計画通りに進むことはありえません。狂いの生じた初期計画への固執は納期とコストに悪影響を及ぼしてしまいます。

そもそも、何かを完全に決定することはできない

ビジネスの要求は日々変わってゆきます。今決定できることは今の要求に基づくことだけ。その決定は明日の要求には応えられません。明日には要求そのものが変わってしまうのですから。

つまり、世間に流布しているウォーターフォール型の開発プロセスは、ビジネスアプリケーション開発では機能しないのです。

もっと軽いプロセスを採用しなければなりません。「事前に作成するドキュメントはプログラミングに必要なものに限定し、計画や要求やソースコードを適宜修正することこそが正しい」としてしまいましょう。

このような“軽い”開発方法は「アジャイルメソッド」と呼ばれています。eXtreme Programming（紛らわしいことにXPと省略されます）がその代表。本連載もアジャイルでゆくことにしましょう。

eXtreme Programmingは、どういうやり方で作業を進めてゆけば良いのかを「プラクティス」という形で具体的にまとめています。今回の開発に適用したプラクティスを以下に箇条書きしておきましょう。

Practice | 小規模なリリース

プロジェクトは1～3週間という短い単位でのリリースを反復して進んでゆきます。リリースより先のことは大まかにしか定義しません。詳細な計画を立ててもどうせ無駄になってしまうのですから。

Practice | シンプルな設計

あとで必要になると考えて何かを作っても、それが必要になることは多分ありません。ですから、今必要な部分を満たす最もシンプルな設計にとどめます。

Practice | テスティング

eXtreme Programmingでは、テストを自動化して、開発のあらゆる段階でテストを行ないます。また、テストコードはテスト対象コードを作成する前に作成します。

Practice | リファクタリング

リファクタリングとは、機能を変更せずにソースコードを洗練させてゆく技法です。完全なソースコードをいきなり書ける人は存在しないので、作った後にソースコードを洗練させる必要があるのです。