世界はオブジェクトの海に浮かぶ

.NET Frameworkで楽しむオブジェクト指向

第18回 帰ってきた“有限状態機械生成機”

επιστημη
えぴすてーめー

昔書いた記事

先日編集部からDVDが届きました。dotNET MagazineとWindows Developer Magazineのバックナンバーを丸ごと収録した、定期購読者用の特製DVD「the BackNumbers」です。早速読ませていただいてます。自分の書いたのを……。

僕の書くものは連載とはいえほとんどが一発読みきりだし、ふとした思い付きをネタにガリガリ書いたりするものだからいつごろどんなのを書いたのか、情けないことに自分でもほとんど憶えちゃいないんです。昔書いたヤツを読み返してみると、どうにも不完全燃焼と申しましょうか。ツッコミが足りないヤツが少なからず見受けられます。いや、ほんとお恥ずかしい限り。

少しずつでもフォローせなあかんかなーと反省しつつ過去の記事へと遡っていくと、dotNET Magazineでの連載「επιστημηのオブジェクト指向的夜話」の第2・3回[注1]に行き当たりました。ここでのお題は状態遷移表。デザインパターンで言うところのState Patternでアプリケーションをこしらえる試みです。つらつらと読み返してみるに、今思えば書き足りてないと思われるツッコミどころ満載の内容でした……。

んなわけで今回はdotNET Magazine 2004年9月号に掲載した「有限状態機械生成機」のフォローアップを行ないます。

有限状態機械「Finite State Machine」

なにせ2年も前のネタ（大元のアイデアは20世紀から引きずってます）です。ざっくりおさらいしておかないと。いただいたDVDから当時の書き出しを引用します。まずは次ページの囲みをご覧ください。

当時の記事ではこんな書き出しに続いてXMLで表現された有限状態機械（FSM：Finite State Machine）をXMLパーサーで読み出し、FSMの動きを再現するC#コードを生成させるって段取りになってます。当時僕はC#に手を染めて間もない頃で、なん

注1）dotNET Magazine 2004年8月号「第2回：ModelとViewを分離する」、2004年9月号「第3回：有限状態機械生成機」。

レベル >>> Level

1 2 3 4 5

言語 >>> Language

- C#

ツール >>> Tool

- Visual Studio 2005 Professional

サンプル >>> Sample

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、**http://www.shoeisha.com/mag/windev/**からダウンロード可能です。

dotNET Magazine 2004年9月号「επιστημηのオブジェクト指向的夜話」より

オブジェクト指向の教材として、少し前に簡単なコードをC++/Javaで書きました。お題は“ゲート”です。近所の大きな病院の駐車場にあるもので、見舞い客の車がこのゲートを通過します。病院の受付の横にコインの自動販売機があって、見舞い客はこのコインを買い、駐車場を出るときにこのコインをスロットに投入すればゲートが開き、車が通過するとゲートが閉じるからくりになっています（実際にはもっと複雑なのでしょうが）。ゲートには2つの状態（state：ステート）、

・Open：開いている
・Close：閉じている

があり、ゲートに対して発生する事象（event：イベント）は、

・Coin：コインが投入される
・Pass：車が通過する

があります。ゲートの動きはこの状態と事象の組み合わせで表現できます。ある状態（s）において事象（e）が発生したとき、状態と事象の組み合わせに対応した動作（action）を行なって新たな状態に遷移します（図1）。

このように、有限の状態数を持っていて、それぞれの状態で発生する事象（event）に応じた動作（action）と新たな状態（state）への遷移を行なうからくりを“Finite State Machine（有限状態機械）”といいます。ゲートの有限状態機械に「gate_model」と名前を付け、今流行り（？）のXMLで書き表わしたのがリスト1（gate.xml）です。十数行のテキストとなりました。

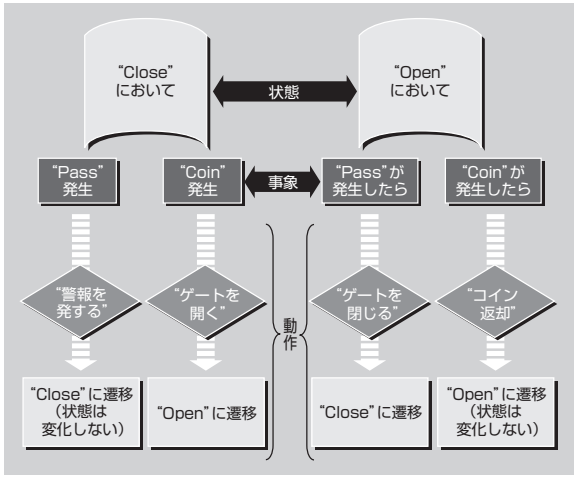

図1：状態と事象と動作の関係

リスト1：XML化したgate_model

```
<?xml version = "1.0" encoding="shift_jis" ?>

<fsm namespace="gate_model">
  <state name="Close">
    <event name="Coin" transition="Open">ゲートを開く</event>
    <event name="Pass" transition="Close">警報を発する</event>
  </state>
  <state name="Open">
    <event name="Coin" transition="Open">コインを返却</event>
    <event name="Pass" transition="Close">ゲートを閉じる</event>
  </state>
</fsm>
```

とも不恰好なC#コードを恥ずかしげもなく誌面に載せておりました。今ならもっと気の効いたわかりやすいコードが書けるわけでして、いやまったくお恥ずかしい限り。2年も経っちまってて恐縮ではありますが、XMLによるFSMからコードを生成する“有限状態機械生成機”SMC（State Map Compiler）をリライトさせていただきます。

StatePatternのちゃんとした実装

「事象（event）が発生したとき、それに応じた動作（action）と次に遷移すべき状態」を状態ごとにきちんと記述するため、リスト1のXMLを少しばかり修正したのがリスト2です。リスト1では動作を単にテキストで記述しただけだったのをaction属性としました。また、遷移を伴わない（状態に変化のない）場合はtransition属性を省略するように改めました。

こうしてFSM（有限状態機械）記述の中に現われるeventタグのaction属性をすべてピックアップすれば、機械の動作に伴うアクションのみを抽出したinterfaceを作ることができます。リスト2に現われるaction属性は、

・Unlock：ゲートを開く
・Lock：ゲートを閉じる
・Alarm：警報を発する
・Return：コインを返却する