ソフトウェアに対するパタンについて

01ca0125 鈴木 藍
2002年 5月 28日

目 次

概 要

オブジェクト指向に関するデザインパタンとそれに付随する言葉を調べ、まとめた。また、GoF の 23 のデザインパタンについて、使った事がありそうなパタンを調べた。

レポートの目的

有名なデザインパタンについて調べ、それを知識として身につける。また、オブジェクト指向という方法論について改めて振り返る。

1 はじめに

1.1 レポートの内容

ソフトウェアに対するパタン、アーキテクチャについて調べたが、アーキテクチャの種類とデザインパタンそのもの、 デザインパタンのカタログに触れたのは初めてだったので、文献の内容を記憶する意味でほぼそのままをレポートにした。内容については自分の考察などを適宜付け加えながらまとめてある。
デザインパタンはどのような言語、あるいはプログラミングにも存在するものであるが、今回はオブジェクト指向プログラミングを行う上で登場するパタンについて調べた。また、授業で触れた MVC についても調べてみた。

2 オブジェクト指向設計

2.1 オブジェクト指向設計を行うために

オブジェクト指向という方法論のメリットは「物」という人間的な認識の単位でソフトウェアを構築できるところにある。現実世界にあるものを、計算機上で写像し実行させる事は設計をする上でもとても有益なことであるが、実際計算機に必要とされている物は、現実世界「そのもの」ではなく、現実世界をあらわして問題解決を行うソフトウェアという一種の「機械」である。産業の中でのソフトウェア開発はこの「機械」の機構や これを構成する部品の設計を戦略的に行う必要が有る。現在のソフトウェア開発における設計には、これらを効率良く行うための設計手段が不可欠であると言える。

3 オブジェクト指向設計を行う上での基本的な概念

3.1 アーキテクチャ

もともとは建築の用語で、建築物が作られる際にその建築がとるべき典型的な構造、装飾を示すものである。さらにソフトウェアの世界では、単にソフトウェアの典型的構造にのみ焦点を当てているわけではなく、その振る舞いについても規定しており、どのようなデータが流れ、どのようにプロセスが起動し、どのモジュールが処理を行っていくのかという、典型的な振る舞いの規定を定める事によってシステムは信頼できるものになる。

3.2 フレームワーク

3.2.1 フレームワークとは

フレームワークは、アーキテクチャにくらべ、実際の具体物が存在するという点でより明確にとらえることが出来る。通常は、「特定のアーキテクチャに基づき、実装された半完成のシステム」を指し、「アーキテクチャの実例」とも言う事が出来る。

3.2.2 フレームワークの具体的な意味

アーキテクチャに基づいてソフトウェアを作る際に、一からすべてを構築していったのでは非常に時間がかかるので、特定のアーキテクチャに基づいて実装されたできあいの半完成システムに対し、プラグインとして作成した (またはどこからか入手した) 特定のソフトウェア部品をはめ込むことで、望みのシステムを手速く、安定したものとして作ることが出来る。フレームワークの代表例としては MVC ライブラリや、最近では Applet の仕組みも一種のフレームワークと言う事が出来、開発者は Applet のサブクラスを作成 (もしくは入手) するだけで、WWW ブラウザを通じ、サーバから動的に Applet を受け取ってクライアントとして実行させていくことが可能になる。

3.2.3 フローズンスポットとホットスポット

フレームワークを設計する上で重要なことは、どの部分をプラグインできるようにし、どの部分を出来合いのものとして固定化するかということにあり、この選択を間違えると手の込んだだけの使えないフレームワークになってしまう。メタパターン (フレームワークを構成するためのデザインパターン) を定義した Wolfgang Pree は、フローズンスポットとホットスポットという言葉で、このことを表現しており、フレームワークを作成する際にドメイン分析の結果、固定化しており、今後の変更が起こらないと思われる部分は、強固な流れを規定するフローズンスポットとなり、一方で新たな要求の発生が予想され、その変更に対し柔軟に対処できるようプラグを用意した部分がホットスポットであると定義している。
しかし、ソフトウェア開発を行う上で、どの部分がフローズンスポットであり、どの部分がホットスポットであるかの区別は自動的にできるものではなく、この分割を行っていくには、開発者側の都合を考えただけでは不完全であり、ユーザから得るアプリケーションのドメインに関する知識が不可欠である。ユーザの要求の中で変化しやすい部分とそうでない部分を見定めて「分けていく」ことが「分析」という作業になる。

3.3 デザインパタン

フレームワークのホットスポットとフローズンスポットにおいて、変化しやすい部分をホットスポットとして実装コードの中に適切に表現していくには、単に分析ができているだけでは不十分である。実際にフローズンスポット、ホットスポットをコードとして組み上げていくにはそのための道具が必要になり、その道具がデザインパタンである。

4　アーキテクチャの例

4.1　MVC とは

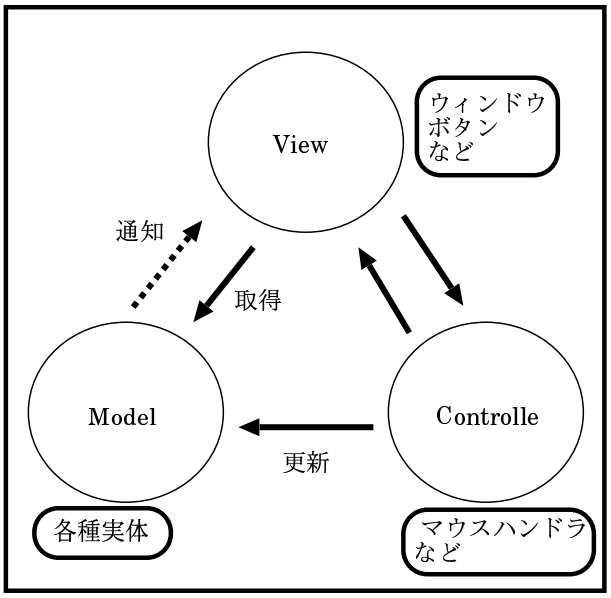
MVC は、対話型のアプリケーションを作成するためのアーキテクチャである。Smalltalk やその派生などの環境が、この MVC と呼ばれるアーキテクチャを用い、高度な GUI を実現している。

4.2　MVC の意味

MVC はそれぞれ、Model、View、Controller を意味しており、Smalltalk などの環境であれば、M はコンピュータが扱うデータ (一種の Model)、V はディスプレイ (一種の View)、C はマウス (一種の Controller) という様に分ける事が出来る。
また、MVC では、対話的なアプリケーションを実現するため、アプリケーションの構成要素を、処理、表示、入力の 3 つ組みに分割することに決めており、この分割が MVC アーキテクチャの基本的な構造を規定している。
Model の状態を表示する部分は View となり、ディスプレイ上に表示されるウィンドウや、その中のウィジェット群が View の候補になる。Controller は、マウスやキーボードからの入力をイベントとして受け取る役割を受け持ち、OS からのマウスやキーボードのクリックといったプリミティブなイベントの中から、自分に関係のあるものを選択、処理し、適切な形に変換してモデルに伝達する。

4.3　それぞれの役割

以下は Model-View-Controller の 3 つ組みの関係を表した図である。



Controller が、OS からイベントを受け取り、Model にメッセージを送り、Model の状態を更新する。Model の状態変化は View に通知され、通知された View は、Model の最新の値を取得し、再

表示を行うという流れが基本的になる。3 つ組は、それぞれ独立しているのが望ましく、特にモデルとビューの切り離しは非常に重要であり、「View や Controller は Model を知っているが、Model はその他の要素については何も知らない」という状態になるのが理想とされている。

5　デザインパタン

5.1　デザインパタンとは

デザインパターンは、もともとフレームワークの作成過程の中でなるべくソフトウェアを再利用性が高く、効率も良く、また柔軟なものとしたいという開発者の要求から、自然発生的に生じてきたものである。実装としてのフレームワークは、特定の言語によるクラス定義とその関連から構成されており、これには繰り返し特定のパタンが現れて出ている。

5.2　デザインパタンの成立ち

80 年代の後半ごろから、このパタンを精製して整理することで、後のフレームワーク作りの構築要素として役立てることができるのではないかと設計者は徐々に気づき始めた。1987 年、OOPSLA というオブジェクト指向に関するカンファレンスで、Kent Beck、Ward Cunningham の二人によって”Using Pattern Languages for Object-Oriented Programs”という記念的な論文が発表される。この論文は、建築家である、Christopher Alexander が、クライアントの要求にかなった建築を作り出すために抽出した「パタン」の考えをソフトウェアにも応用できるのではと示唆したものであった。

5.3　パタンという考え方

Alexander は、「パタン」という建築の過程で生じてくる様々な問題や、それに対する典型的な解決の方法を、誰もがわかりやすいパタンとし、カタログ化して示すことにより、設計者が自由にパタンを選択して、より要求にかなった建築の手がかりとしていくという考え方を提唱した。ソフトウェアも、要求をかなえるための成果物を構築しなければならないという点において、建築の世界と非常に似た部分がある。Kent Beck と Ward Cunningham は、GUI の構築について幾つかの「パタン」をカタログ化して提示することで、MVC の全てを理解しなくても、一定の基準を維持したアプリケーションを作成できるのではないかと主張した。

5.4　パタンの種類

パターンは、扱う問題のスケールや種類によっていくつかに分類できる。
・ アーキテクチャパタン
・ デザインパタン
・ イディオム

5.4.1 アーキテクチャパタン

アーキテクチャパターンは、システム全体を支配するような大きな問題解決のために特定のアーキテクチャの適用方法を説明したものである。MVC などが例にあがる。

5.4.2 デザインパタン

デザインパタンは、クラス間の関係や相互作用について示したものであり、フレームワークの構築要素として役立つ。

5.4.3 イディオム

特定のオブジェクト指向で有効な実装のルールやスタイルをパタンとして扱ったものである。意味としては「慣用句」や、個人語で何度も現れるパタンなどにあたる。

5.5 現在のデザインパタン

GoF (Gang of Four) - Erich Gammma, Richard Helm, Ralph Johnson, John Vlissides ら四人による”Design Pattern: Elements of Reusable Object-Oriented Software” において、23 のデザインパタンが提唱されている。しかし、現在はこれらに限らず企業などが独自に考案したパタンも公開されている。

6 23 種のデザインパタン

今回は、生成 、 構造 、 振る舞い 、 に含まれる 23 のデザインパタンを見て、Java でプログラミングをした時に遭遇したと思われるものを調べた。Abstract Factory から Chain of Responsibility までは実際に遭遇したと思われるが、それ以降は興味深いパタンだったので調べてみた。また、実際の実装に関しての記述は、言語に依存した内容になってしまうので省いてある。

6.1 Abstract Factory

実装に依存する部分を意識せずに、複数の部品群を生成するためのパタン。上位クラスではインスタンスを生成せず、実際にオブジェクトを作るのはサブクラスになる。今の所、Template Method と Factory Method との見分けが付いていない。

6.2 Factory Method

インスタンス作成をサブクラスにまかせて抽象クラスを生成するパタン。抽象クラスはインスタンス化されるサブクラスを事前に想定していない。学校の OOP の授業で C++ を教わっているが、メンバ関数 に virtual という宣言をすることで いずれかのインスタンスがオーバーライドすることを想定している。どのようなインスタンスかは想定していないので、この部分は Factory Method になると思うが、これも Template Method と見分けがつかない。

6.3 Template Method

基本的な振る舞いの記述は抽象クラスにテンプレートとして定義し、目的によって内容の記述を変えなければならない部分をサブクラスで定義する。Java などのオブジェクト指向言語では当り前のように使われていると思う。パタンとして上がっているのを見て改めて認識した。

6.4 Mediator

Web の文献によって記述が違っていたが、判断ロジックをオブジェクトに集中させたりあるオブジェクトに複数のオブジェクトの関連を記述するパタン。普段から行っている方法に思える。

6.5 Prototype

インスタンスから、さらにインスタンスを生成するパタン。Java で プログラムを書いていた時はあまり良い方法だとは思っていなかったが、普通に行われているようである。よく見ると自分自身をそのままの状態で生成しているが、なかなかこの風景が興味深い。

6.6 State

状態遷移のためのパタン。個々の状態をクラスとして表す。また、状態をサブクラス毎に記述するため、コードの可読性が高まるだけでなく状態の追加も容易になる。クラスそのものを状態として表した事はないが、クラスの属性の値を状態として、その状態を返すメソッドを作成することを普通にしている。わざわざクラスで状態を表す状況がまだわからない。

6.7 Chain of Responsibility

オブジェクトを数珠つなぎにし、あるメッセージを解釈できるオブジェクトに突き当たるまで、そのメッセージを横流しにするパタン。 複雑な switch 文を記述せずに、解釈できるオブジェクトにあたった時にはそのなかに処理を記述し、 可読性を上げる事が出来る。

6.8 Interpreter

言語の文法表現を解釈するためのオブジェクト構成を提供するパタン。問題解決のために新たな言語を作る。今回みた中で一番興味深いパタンである。このような事が 23 個の中の一つにあげられていることがとても意外だった。

6.9 Observer

複数の関連し合うオブジェクトの状態の一貫性を保つためのパタン。状態の基準となるオブジェクトを Subject と呼び、Subject と関連をもつオブジェクトを Observer と呼ぶ。1 人の Subject を多数の Observer たちが監視するので、通常 Observer は複数になる。一つの Observer に変化があると、Subject へ通知し、Subject は、その変化を関連する全ての Observer へ通知する。クライア

ント (Observer) がサーバ (Subject) に通知をしている形が クライアントサーバの様に思える。

7　これからのデザインパタンへの取り組み方

今回、デザインパタンを調べた上で、自分がこれからデザインパタンにどう取り組むかをまとめた。

7.1　デザインパタンを見て

23 種の中で、いくつかは以前に見た事があるような形をしていたがそれ以外のパタンについては、そのパタンの使われる状況の予測がしにくかった。また、これらを覚えるだけでは身につかないことは分かった。

7.2　どのように学習してゆくか

まず、すべてのパタンの枠組を一通り簡単に覚えておくことが大切だと思った。もし、ある問題に遭遇した時に、使えるパタンをすぐに出せる様な状況にしておけばこれらが活用できると思う。ただ、使うだけではなく、実際に自分で考えて本当にそれが適切であるかどうかを見定める事も忘れてはいけないと思った。

まとめ・結論

デザインパタンを学習する事は、自分のもっとも良い言い回し (イディオム) に早く気が付くための近道であると思った。また、これまで多くの技術者が培った知識や経験を使う事はよい作法を身につける近道でもあると思う。

感想

初めてデザインパタンについて調べてみたが、意外なパタンについての記述も多かった。Web で Smalltalker 以外のデザインパタンの文献はどれを参考にしたら良いかよくわからなかったので Happy Squeaking!! の引用がほとんどになったが、デザインパタンの本を書いているサイトも参考にしてみた。パタンについてはまだ一通り覚えるだけにして、しばらくは (学生の間だけは) 車輪の再開発を続けたいと思う。

参考文献

[1] Smalltalk イディオム
青木 淳 著
1997 年 2 月 25 日 (第 1 版)

[2] Happy Squeaking!!
http://www.ogis-ri.co.jp/otc/hiroba/technical/Squeak/
2002 年 5 月 27 日 参照

[3] UML 入門
山田 正樹 著
2002 年 3 月 11 日 初版

[4] Java 言語で学ぶデザインパターン入門
http://www.hyuki.com/dp/
2002 年 5 月 27 日 参照

[5] Skelton of GOF's Design Pattern
http://www11.u-page.so-net.ne.jp/tk9/ys_oota/mdp/
2002 年 5 月 27 日 参照

[6] デザインパターンの活用について
http://www.dmz.hitachi-sk.co.jp/Java/Tech/pattern/apply.html
2002 年 5 月 27 日 参照