連載　オブジェクト指向と哲学
第 47 回　オブジェクト指向を分析する - 目的と手段

河合 昭男
http://www1.u-netsurf.ne.jp/~Kawai

高品質のソフトウェア製品をより効率よく開発できる技術を求めて、オブジェクト指向のパラダイムは優れている面が多いことが認められてきました。オブジェクト指向には様々な概念や用語が登場します。本質はどこにあるのでしょう？今回はそれらを目的と手段という視点で考えたいと思います。

‥　オブジェクト指向っぽさ

UML のシーケンス図はいかにもオブジェクト指向っぽいと感じますが、アクティビティ図はあまりオブジェクト指向っぽいとは感じません。ここにヒントがあります。クラス図は微妙です。概念モデルとしてのクラス図はデータベースの ER 図とほとんど変わりませんが、操作が入るとオブジェクト指向の基本的モデルとなります。

筆者の感性で UML の主要なダイアグラムをオブジェクト指向っぽさで分けると、

・オブジェクト指向っぽいと感ずるもの：
- シーケンス図（相互作用図）
- ステートマシン図
- クラス図（操作の含まれるもの）：

・オブジェクト指向っぽくないと感ずるもの：
- アクティビティ図
- ユースケース図
- クラス図（操作の含まれないもの）

シーケンス図はオブジェクト指向という呼称の原点となった Smalltalk のメッセージパッシングのモデルを表現します。ステートマシン図はひとつのオブジェクトのイベントによる状態変化を表すものですが、メッセージもイベントもそれを受けて処理を行うのはクラスの操作です。つまりオブジェクト指向っぽいと感ずるのはすべてクラスの操作の存在によるものです。

一方、操作の働きが前面にでてこないものにはオブジェクト指向っぽさが感じられません。

・　部品化と再利用

オブジェクト指向に期待される技術のひとつはモジュール部品の再利用です。古典的名著「人月の神話[1]」を振り返ってみます。当時オブジェクト指向の技術は段階的に様々な側面が提唱されていましたが、ブルックスは特にパルナスの提唱するモジュール化の技術に注目しました。

--

だが実は、オブジェクト指向プログラミングで実現がもっとも期待されているのは、第一段階に由来するものだ。すなわち、モジュールのカプセル化（隠蔽）と、それに加えて、再利用のためにデザインされテスト済みの事前作成済みのモジュールライブラリまたはクラスに関するアイデアである。[1]

--

部品化は高品質・高生産性のための基本技術です。しかし当初パルナスの考え方には否定的な見方をしていました。後になって「情報隠蔽に関して、パルナスは正しく、私は誤っていた」と考え直しています。

--

プログラマは自分の担当以外のシステム部分の開発に関する詳細にさらされるより遮断されている方が効率的だと言う。ここでは、すべてのインタフェースが完全かつ正確に定義されていることを前提にしている。[1]

--

このパルナスの考え方を 1975 年版では「失敗のレシピ」としたが、20 周年記念版（1995 年版）では正しかったと訂正しています。

インタフェースを公開し実装を隠蔽することはオブジェクト指向の基本技術です。これは設計レベルの技術ですが、開発プロセスにも有効です。プロジェクトのチームは、インタフェースを利用する側と内部を実装する側が、開発作業を独立に並行して進めることができます。

・・　自律分散協調

オブジェクト指向という呼称の原点となった Smalltalk はメッセージパッシングでシステムを動かすことに特徴があります。これは基本技術というより、そのようにシステムを構築したいと

いう目的です。

アラン・ケイが object oriented という呼び方を提唱したそのあとで、C++を開発したビャーネ・ストロヴストルップ（Bjarne Stroustrup）はオブジェクト指向プログラミング言語の 3 つの特徴を再定義しました。

--

C++の創始者は、Smalltalk のような論理美の追求よりも、現用としての実用性を重視した。そのため、C++の再定義した「オブジェクト指向」はこれらの問題を全てクリアにし、既存言語の拡張としてオブジェクト指向機能を実装できることでブレイクスルーを迎え急速に普及する。これによりメッセージ送信という考え方はやや軽視されるようになり、オブジェクト指向とは C++の再定義したものと広く認知されるようになった。[2]

--

筆者は自律分散協調モデルやメッセージパッシングにオブジェクト指向の本質を感じます。UML では相互作用モデルです。メッセージパッシングは目的であり、C++の再定義は目的達成のための手段です。

カプセル化

オブジェクト指向を手段と目的で捉えるなら、3 つの特徴 － カプセル化・継承・多態性は手段であり、部品化・再利用と自律分散協調モデルは目的です。2 つの目的に最も貢献しているのはカプセル化です。これがオブジェクト指向の本質です。

UI のオブジェクト指向操作という目的もカプセル化という手段で実現されています。プログラム内部で行うオブジェクトの操作呼び出しと、ユーザーによるマウスのクリック操作によるオブジェクトの操作呼び出しの仕組みは同じです。

カプセル化にオブジェクト指向の本質がある。それによりモジュールの部品化・再利用ができ、メッセージパッシングのシステムが構築できる。継承や多態性は偶有的なものか？

カプセル化という概念には、クラスという型に属性と操作が一体化されていること、そこからオブジェクトが生成できることが含まれている。

アリストテレスはものの本質はそのものの中に埋め込まれていて取り出すことはできないと考えた － つまり形相と質料はカプセル化されている。それを生成する型というものはない。プラトンはイデアからものが生成されると考え、本質はイデアにあると考えた。イデアは型に近いよう

です。

本質はイデアにあるという考え方は、プログラミング言語の仕組みと似ています。クラスから生成されたインスタンスにはメモリ領域が割り当てられ、そこに属性の値を保持するが、操作の実装はクラスで共有され、個々には保持しない。操作は同じなので個々が内蔵する必要はありません。

ものの存在をオブジェクト指向で捉え、形相を操作の実装と考えるならアリストテレス方式よりプラトン方式のほうが効率的です。

まとめ

今回考えたことを整理します。

・オブジェクト指向の本質はカプセル化にある

・カプセル化は次のような目的に手段として貢献する
- 部品化・再利用
- 自律分散協調モデル（メセージパッシングのモデル）
- オブジェクト指向操作

以　上

【参考書籍とサイト】

[1]フレデリック・P・ブルックス,Jr、人月の神話、1996、アジソン・ウェスレイ
[2]http://ja.wikipedia.org/wiki/オブジェクト指向プログラミング