ソフトウェア工学 120−22
（1998. 7. 24）

オブジェクト指向による再利用性向上に対する要因評価

深澤 良彰
早稲田大学 理工学部

オブジェクト指向の進化とともに、各種の再利用技法が提唱されてきている。本稿では、これらが、これまでの手続き型パラダイムにおける再利用技法と何が異なるのか、本当に異なるのかについて述べる。特に、オブジェクト指向によって、再利用性が高くなった何が原因なのか、オブジェクト指向の何によって何が得られたのか、何が期待外れだったのかについて分析を試みる。

Factor Analysis of Reusability Improvement in Object-Orientation

Yoshiaki Fukazawa
School of Science and Engineering, Waseda University

As the expansion of object-oriented software development technologies, various kinds of reuse technologies have been proposed and practically used. In this paper, I describe which reuse technologies are truly different between function-orientation and object-orientation.

1 はじめに

オブジェクト指向ソフトウェア技術が注目されはじめた当初から、高い再利用性の達成に対して大きな期待が寄せられてきた。初期において、再利用性の向上に寄与すると考えられていた要因としては、次が挙げられる。

（1）これまでの分析・設計・実装手法における主導概念であった「機能中心」に比べて、オブジェクト指向が掲げている「モノ中心」の方が、アプリケーションに対する依存度が弱く、再利用の可能性が高まる。

（2）継承を用いたホワイトボックス型の再利用によって、クラス単位での高い再利用性が期待される。

しかし、これらは、いずれも希望的観測あるいは楽観的予想に過ぎなかったことが判明してきた。たとえば、上記（1）に対しては、機能中心の方法論においてもうまくモジュール分けを行なえば、高い再利用性を得ることができると同様に、うまくクラス分けすれば、高い再利用性が得られるという、方法論に依存する再利用性の向上ではなく、個人の資質による再利用性の向上であることがわかってきている。また、（2）に対しては、逆に継承そのものが再利用性を妨げる要因となっているとの指摘もある。

2 オブジェクト指向固有の再利用

1章で述べたような、ただ楽観的な期待をするだけでなく、より積極的に再利用性を高めるための提案・実用化が試みられているものに次が挙げられる。

2.1 クラスライブラリ

頻繁に使用されるクラスの集合体をライブラリとして用意しておき、これを高いレベルのデータ型が与えられているものとして利用していこうとする考え方である。従来のサブルーチンライブラリのオブジェクト指向版に相当する。直観的には、汎用のデータ型（スタック型や木構造型など）を実現したクラスライブラリが考えられる。これらは実際に用いられてきているが、より大きな効果を発揮しているのは、グラフィカル・ユーザ・インタフェイス (GUI) を実現するためのクラスライブラリ (MFC、AWT など) である。このようなクラスライブラリが多用される理由は、グラフィカル・コンポーネントが、クラスとしての性質をよく満たしており、かつ、GUI の実装がこれまで面倒な作業であったことによると考えられる。

2.2 デザインパターン

すでに開発された多くのオブジェクト指向システムを分析してみると、その中に、クラスやオブジェクトの定型的な使用法があることがわかる。このようなパターンにおけるオブジェクト間での役割りの分担や制御のやりとりなどを記述したものが、デザインパターンである。特定のアプリケーションに依存していないという意味で、汎用性の高いパターン集を目指しているものもあれば、ビジネス固有の情報を含み、よりアプリケーションに近いパターンを提唱しているものもある。また、汎用性を直接的には目指さずに、固有のプログラミング内で利用することを目的に、気軽にパターンとして登録し、利用していこうという試みもある。

ソフトウェア開発における各種のノウハウも再利用の重要な対象である。しかし、これまでソフトウェア開発においては、知識は個人的な所有物であり、これが書籍として出版され、ノウハウの再利用が広く行わ

れることは少なかった。例外的に、従来型のプログラミングにおいて、プログラミングパターン、クリシェなどと命名された定石的なコードの集合が定義され、特にAI技術の一部として研究されてきた。

デザインパターンが、現在のように注目を浴びている最大の理由は、パターンの一覧を掲載した書籍が発売されたことであろう (たとえば [1])。これまで、パターンがあることはわかっていても、それが開示されることはなかった。逆に考えれば、オブジェクト指向であることの必然性は何であったのだろうか。

2.3 フレームワーク

フレームワークは、「抽象クラスと具象クラスの集合であり、それらの間のインターフェイスを定めたものである。そして、サブシステムを単位とした設計に役立つもの」と定義される。「半完成のソフトウェアアーキテクチャ」と考えることもできる。

フレームワークを再利用する場合には、個々のクラスを再利用するのではなく、クラスの階層構造全体を再利用する。そして、自分の要求に合わせて、必要なクラスだけを変更したり、新たなクラスを付加したりする。この考え方は、そのアプリケーションに固有な部分の中に再利用可能な部品を組み入れるという通常の再利用の逆である。

これまでのソフトウェアの分類の中で、このフレームワークに最も近い概念は、パッケージ・ソフトウェアと呼ばれてきたものであると考えられる。パッケージ・ソフトウェアにおいては、あるシステム固有の部分を、予め与えられたパラメータの設定によって実現する。しかし、フレームワークにおいては、仮想関数として未定義のまま用意されている部分に対して、任意の関数を与えることができる。これは、動的束縛機能が効果を発揮する例でもある。

2.4 ソフトウェアアーキテクチャ

同じような性質をもったアプリケーションの構造は同じようになることが考えられ、このアプリケーションの構造を参照モデルとして再利用することにより、優れた設計を容易に行うことができるようになることが期待される。このような背景から生まれてきたのが、ソフトウェアアーキテクチャである。

直観的には、ソフトウェアアーキテクチャとは、ソフトウェアの構成要素とそれらの間の関係を示すものである。現時点では、一般的に受け入れられている明確な定義は存在していないが、「ソフトウェアアーキテクチャとは、ソフトウェアに含まれる複数のサブシステムやコンポーネントおよびそれらの間の関係の記述である。サブシステムやコンポーネントは、システムの機能面や非機能面の特性を表すさまざまな視点から記述される。ソフトウェアアーキテクチャは、設計行為の成果物である。」などと定義されてきている。

このようなアーキテクチャの議論は、決して新しいものではなく、モジュールの切り出し方などを中心に、さまざまな研究／実用化が行われてきている。その中で、今日特に脚光を浴びるようになった理由として以下を挙げることができる。

・アーキテクチャミスマッチの問題視
・構成要素間のインタフェイスの標準化の普及
・ソフトウェア基本構造の重視

2.5 ドメイン分析

同種のシステム全体の対象領域を明確に定義し、この対象領域に固有な機能、性質、知識などを包含したモデルを構築しておき、これをもとに、ソフトウェアを開発するすることが考えられる。この時に、同種のシステムの共通部分として考慮の対象としたアプリケーションの範囲をドメインと呼び、その領域をモデル化することをドメイン分析と呼ぶ。ドメイン分析の結果得られたドメインモデルは、そのドメインに属する新しいソフトウェアの開発の際に、ひな型として用いられる。また、ドメインに共通な機能を抽出し、それを部品として実現することにより、効果的な部品化も達成できる。開発プロセスは、類似したアプリケーションを抽象化することによりドメインモデルを構築し、これを具体化することにより、目的とするアプリケーションを開発することとなる。

ドメイン分析そのものは、オブジェクト指向を前提としているものではない。しかし、ドメインの記述にオブジェクト指向が向いていることは事実であり、双方の技術が相補的に適用された事例と考えられる。

2.6 コンポーネントウェア

ソフトウェアについても、ハードウェアと同じように部品を接続するだけで、自分の要求するシステムが構築できないものであろうかと考えるのは自然なことである。これを、オブジェクト指向の概念に基づいて作成されたソフトウェア部品を対象に考えようとすることが、コンポーネントウェアの発想である。この実現のために作成されたソフトウェア部品、ソフトウェア部品を組み合わせて開発されたソフトウェア、これを実現するための環境などが、広くコンポーネントウェアと呼ばれる [2]。

コンポーネントウェアが実用化されてきている最大の要因は、インタフェースの明確化および実装からの分離である。これこそが、分散オブジェクト指向などにもつながるオブジェクト指向普及のキーである。

3 おわりに

これまでの再利用技術に比べて、オブジェクト指向が勝っている点は、単なる掛け声だけでなく、再利用の仕組みを積極的に構築し、実例を収集／分析してきていることである。今後もこの傾向が続くことを祈り、また、一方では、「機能中心」の世界へのフィードバックが行なわれていくことを期待する。

参考文献

[1] E.Gamma et. al., 本位田真一他監訳, "オブジェクト指向における再利用のためのデザインパターン", ソフトバンク (1987).

[2] 深澤良彰, "コンポーネントウェアにおける方法論とツールとの協調関係", 情報処理学会 ウインターワークショップ・イン・松山予稿集, pp.47-48 (1996).