オブジェクト指向設計におけるソフトウェア・アーキテクチャの位置付け

3K−4

梶原 清彦
NTTソフトウェア研究所

1 はじめに

オブジェクト指向は部品化や再利用に適していると一般的には言われているが、それは抽象データや継承のような部品レベルでの議論である。本論文では、オブジェクト指向設計における、部品より大きな再利用の枠組を示し、さらなる生産性向上の可能性を検討する。

本論文は、まず、ソフトウェアの生産性を向上させる方法である再利用について述べ、次にオブジェクト指向設計で再利用を行なうためのモデルとプロセスを説明し、それらがOODで有効であることを示す。

2 生産性の向上

生産性を向上させる方法として、過去の蓄積をそのまま利用する方法と（以後、部品利用と記す）、本質的な部分だけを抽出し、それをもとに実際に利用するものを作り出す方法（再利用という言葉では誤認する可能性があるので、本論文では再生と記す）がある。部品とは制約の変更や言語に依存したチューニングなどは行なってもよいが、基本的にはデータ構造やアルゴリズムを変更しないで利用する設計単位であるのに対して、再生とは設計単位の抽象的な役割定義から、実際に利用する具体的な新しい設計単位を生成することを意味する。

個々の部品の利用では生産性の向上はわずかなものであるが、外部組織が提供するプラットホームのような一般化／構造化された部品群（例えば、実行環境や言語、第三者が製品などとして提供する部品）の利用は生産性を向上させる。しかし、一般的でない特殊用途の部品（その多くは対象領域に依存する部品である）は自分達で作り蓄積しなければならないが、適切な範囲に限定し、十分に一般化しないと利用しにくい部品となってしまう。また、異なる考え方に基づく部品では、適切に組み合わせて利用するのが難しくなる。それゆえ、対象領域向きの部品群の作成はコスト的に問題があるうえ、部品間の関連が固定化されてしまうという利用上の問題もある。いずれにしても、部品だけでは大幅な生産性の改善は望めないので、さらなる生産性の向上には異なるアプローチが必要である。

単純な部品より大きな設計単位を再利用すること、問題領域依存の設計単位を再利用すること、固定的に関連付けられない設計単位群を作成することにより、さらなる生産性の向上がはかれる。そのためには、設計単位の特長にもとづいて設計単位を再利用する再生が必須である。しかし、現状の多くのOOD[1][2]は、継承によるクラス単位の再生が可能なだけで、再生のための十分な枠組を与えていない。

以後の章では、OODで再生を行なうための再生対象群を表現するモデルと、再生をOODの一部として行なうためのプロセスを考察する。

3 再生のためのモデル

OODでの再生を支援するために、2つのモデル：ソフトウェア・アーキテクチャとオブジェクト・フレームワークを考える。どちらも一般的に用いられている用語であるが、明確な定義があるわけではない。そこで、本章では、再生の立場から定義し直す。

ソフトウェア・アーキテクチャの目的は、対象システムの捉え方を統一することである。捉え方を統一する必要性は、同一の問題に対して異なる人達が独立に考えた場合には、大きな単位で共通化を図ることは困難だからである。オブジェクト・フレームワークの目的は具体的な再生対象を表現することである。ソフトウェア・アーキテクチャとオブジェクト・フレームワークは設計結果を作り出すためのテンプレートである。ゆえに、作成されるソフトウェアの設計結果は、いずれかのOOD手法によって表現される。

●ソフトウェア・アーキテクチャ

ソフトウェアの全体的構成を表現するためのモデルであり、サブシステムとレイヤを規定する。このモデルで表現される情報はサブシステム、レイヤ、サブシステム／レイヤの役割、サブシステム／レイヤ間の関連であり、図と各サブシステム／レイヤを説明する文章で表現されるのが一般的である。具体

The Role of Software Architecture in Object-Oriented Design
Kiyohiko KAJIHARA
NTT Software Laboratories

例として、分散システムのためのアーキテクチャであるANSA(Advanced Networked Systems Architecture)[3] などがある。

●オブジェクト・フレームワーク

特定のソフトウェア・アーキテクチャの各サブシステム、各レイヤ毎に、どのようなオブジェクトが必要になるかを表現するためモデルであり、再生されるクラスの集合を規定する。このモデルで表現される情報は各サブシステム／各レイヤ毎のオブジェクトの分類／役割／特性、オブジェクト間の関係などであり、適当なＯＯＤでのクラスの表現方式を利用すればよいと思われる。具体例として、Managed Object [4]などが挙げられる。

実際のソフトウェアの開発では、上位のレイヤは再生クラスから生成されるオブジェクトと全く新規に作成するオブジェクトから構成され、下位レイヤは再生クラスから生成されるオブジェクトと部品、また一部の特殊なものに対する新規作成オブジェクトから構成される形が一般的である（図１参照）。

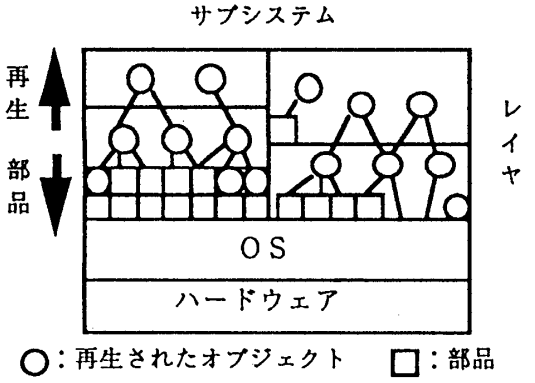

図１　構成モデルの関連

4　設計プロセス

上記のモデルを用いた設計プロセスを概説する。

（１）要求仕様をもとに、ソフトウェア・アーキテクチャの必要部分を選択する。

（２）選択した部分に対応するオブジェクト・フレームワークで規定されているオブジェクトをもとに、基本的なオブジェクトを選ぶ。

（３）オブジェクト・フレームワークで規定されているオブジェクト間の関係をもとに、各オブジェクトの役割を明かにする。

（４）要求仕様から、ステップ(2)で得られていない重要なオブジェクトがないか確認する。もし抜けがあるなら、そのオブジェクトの概要を決める。

（４）以上の結果を適当なＯＯＤ手法で表現する。

以後のオブジェクトの詳細を決定するプロセスはは既存のＯＯＤで行なえばよい。つまり、上記のステップはＯＯＤのための最初の原案を作ることである。ＯＯＤではこののステップが難しく、また、やり直しができない部分である。それゆえ、その決定には時間がかかるし、優秀なスタッフが行なわなければならない。適切なソフトウェア・アーキテクチャとオブジェクト・フレームワークの利用により基本設計の工程を大幅に短縮できる。

過去の対象領域の設計結果を再利用し、より生産性を向上させるために、オブジェクト・フレームワークをリファインすべきである。しかし、オブジェクト・フレームワークの開発には多くの努力が必要であるので、実際の適用では、オブジェクト・フレームワークの開発コストと、再生によって得られるコスト削減の適切なトレード・オフを考える必要がある。

5　考察

現在のモデル及びプロセスは、基本的な方針を示している段階でしかない。今後、以下のような検討が必要である。（以下の文章でのモデルとはソフトウェア・アーキテクチャとオブジェクト・フレームワークの両方を意味する）

・オブジェクト・フレームワークのより詳細な規定が必要である。

・モデルは構成的な観点だけからではなく、ＯＭＴ[2]のFunctional ModelやDynamic Modelのような観点からのモデルも必要である。

・モデルをどれだけ抽象してあればよいのか、適切な目安が必要である。

・プロセスには下位レイヤで部品を取り込むプロセスも考慮／検討しなければならない。

・モデルのリファインのためのメタなプロセスを考える必要がある。

6　おわりに

ＯＯＤの生産性は対象領域の設計情報や設計知識の部品利用／再生に大きく依存している。本論文では、ＯＯＤにおいて設計情報や知識の利用が必要であり、ソフトウェア・アーキテクチャとオブジェクト・フレームワークによって設計情報や設計知識の表現が可能であることを示した。また、それらの表現方法と利用方法の基本的な方針を示した。

参考文献

[1] P. Coad and E. Yordon: Object-Oriented Desing, Prentice Hall, 1991

[2] J. Rumbaugh, et al: Object-Oriented Modeing and Design, Prentice Hall, 1991

[3] ANSA: An Engineer's Introduction to the Achitecture, Architecture Projects Management Limited, 1989

[4] ISO DP10165: 管理情報構造