事務処理分野へのオブジェクト指向４ＧＬ適用の試み（２）
−Ｃ＋＋による再構築例との比較−

1T−8

陳 思悦* 松永 亨* 橋口 一生* 森 忠夫* 竹下 亨**
*㈱アドバンスド・システム研究所 **中部大学

１．はじめに

ＯＯ４ＧＬによる事務処理アプリケーションの書き換え［１］において設計した各モジュールの動作を検証することを目的として、既存のオブジェクト指向言語を用いてさらなる書き換えを行った。その際、事務処理アプリケーションにおけるオブジェクトのカプセル化の効果を検討するため、単純継承及び多重継承を用いて２通りの書き換えを行い、比較した。本稿では、それらの結果についてＯＯ４ＧＬによる書き換えとの比較をふまえて報告する。

事務処理分野においては、オブジェクト指向事務処理向け言語は普及していない。そこで、現在最も普及しているオブジェクト指向言語であり、多重継承機能をもつＣ＋＋［２］を用いて書き換えを行った。

２．Ｃ＋＋によるアプリケーションの再構築の試み

ここでは、Ｃ＋＋を用いて既存事務処理アプリケーションの書き換えを行う。クラスの継承方法は、単純継承と多重継承を用いた場合をそれぞれ試行した。

これらの試行は、IBM PS/5571（メモリとハードディスクの容量がそれぞれ 16 メガバイトと 300 メガバイトである）の上で行った。使用したＣ＋＋はＢｏｒｌａｎｄ　Ｃ＋＋　３．１である。このＢＣ＋＋は、ＷＩＮＤＯＷＳ　３．１の上で動作する。

なお、ＢＣ＋＋単独では、ＯＯ４ＧＬがもつようなＤＢＭＳの機能が含まれていない。データベースの処理関数を作成するのは、本研究の本質ではない。そこで、データベースに関する入出力を行うクラスの存在を仮定し、その実装は行わなかった。

２．１　単純継承の設計

ＯＯ４ＧＬによる書き換えとの比較を行うため、ここでは、同じクラス階層を採用する。それぞれのクラス階層は単純継承のみを利用している。

一方、このような機能ごとに分割されたモジュールにおいては、異なるクラス階層に属すオブジェクト間でのデータの参照が行われる。

その方法は以下の３つが考えられる。

（Ａ）オブジェクト内のデータを公開し、他のオブジェクトから直接参照する。

（Ｂ）メソッドを通してオブジェクトのデータを参照する。

（Ｃ）Ｃ＋＋特有の機能であるフレンドを用いて、関連のあるクラスのオブジェクトだけが直接オブジェクトのデータを参照する。

（Ａ）では、オブジェクトのデータを公開しているので、オブジェクトのデータが見えてしまう。これは、オブジェクト指向のカプセル化の思想に違反する。

（Ｂ）では、オブジェクトのデータが直接参照されていないので、オブジェクトの独立性は向上するが、オブジェクトのデータが増加するにともない、オブジェクトのデータを参照するメソッド数は増加する。例えば、１万個のデータが存在すれば、それらのデータを参照するためにそれぞれに対応するメソッドを作成しなければならない。よって、オブジェクトとのインタフェースが複雑になる。一般に、事務処理アプリケーションにおいては、大量のデータが存在するケースはしばしば見られる。したがって、これは、非現実的な方法だと考えられる。

（Ｃ）では、オブジェクトのデータは、関連のあるクラスに公開されているが、その以外のクラスには隠蔽されている。また、オブジェクトのデータを参照するインタフェースは簡単になる。

今回書き換えの対象とした事務処理アプリケーションにおいては、多くのデータが存在している。オブジェクトのインタフェースを簡単化にするためとオブジェクトのデータをなるべく隠蔽するため、ここでは、（Ｃ）を採用した。これにより、オブジェクトのデータは完全隠

A Trial of Object-Oriented 4GL for Business Applications (2)

Si-Yue CHEN*, Toru MATSUNAGA*, Issei HASHIGUCHI*, Tadao MORI* & Toru TAKESHITA**
*Advanced System Research Organization Ltd.
**Chubu University

蔽ではないが、オブジェクトのインタフェースは簡単になる。これは、われわれの本来のねらいである事務処理アプリケーションのモジュールの単純化に準拠している。

2．2　多重継承の設計

単純継承を用いた結果では、オブジェクトの一部のデータは他のオブジェクトに公開された。これにたいし、オブジェクトのカプセル化を目指すため、ここでは、C＋＋の多重継承を利用して、2．1に述べた機能ごとのクラス階層を統合して1つのクラス階層とする。すなわち、オブジェクト間の直接の参照を多重継承の参照に置き換える。これにより、オブジェクトのデータの参照はすべて継承を通してオブジェクト内で行われる。

3．結果

3．1　単純継承による結果の比較

C＋＋とOO4GLとの事務処理アプリケーション書き換え結果の比較を表1に示す。

表1．OO4GLによるアプリケーションとの比較

項目＼名称	C＋＋により	OO4GLにより
クラスの継承	単純継承	単純継承
クラスの定義の量	約1700行* **	約1200行***
クラス定義の理解度	比較的困難	比較的容易
クラスの再利用性	良い	良い
重複定義情報	ほぼない	ほぼない
カプセル化	一部公開	一部公開

* データベースに関する処理関数ソース量を含まず
** コメント行、空白行および大括弧だけある行を計算せず
*** コメント行及び空白行を計算せず

C＋＋による事務処理アプリケーションのクラス定義は、データベースに関する処理関数ソースを含まないにも関わらず、OO4GLによる同じクラス定義よりコードの量が多い。もしデータベースに関する処理関数を作成すれば、C＋＋によるクラスの定義が大量に増加することが予想される。また、クラス定義の理解度においては、C＋＋による記述の方が比較的困難である。これは、低水準言語と4GLとの差に起因している。

3．2　単純継承と多重継承による試行結果の比較

単純継承と多重継承との比較結果を表2に示す。

表2．C＋＋による単純継承と多重継承との比較

項目＼名称	単純継承	多重継承
クラス階層	単純	複雑
クラスの定義の量	約1700行	約1650行
カプセル化	一部公開	完全隠蔽

単純継承を用いたクラス定義では、オブジェクトの一部のデータは公開されている。これにたいし、多重継承を利用することにより、オブジェクトのデータがすべて隠蔽されており、かつオブジェクトのインタフェースが簡単になる。だが、ある帳票クラスは、他の帳票クラスを継承する以外に、ある業務クラスをも継承する。同じように、ある業務クラスは別の業務クラスと画面クラスの両方を継承している。

このようなクラスの継承関係が増えてくると、クラス階層の全体は複雑になる。そのため、クラスの定義の変更においてその影響がクラス階層の全体に及ぶおそれがある。したがって、アプリケーションの保守性と移植性は一般に低下する。

4．おわりに

今回の再構築プログラムの動作結果により、事務処理モジュールの設計の妥当性を確認した。また、事務処理向け言語である4GLの、3GLに対するメリットはオブジェクト指向を導入しても同様に発揮される。

多重継承を用いた場合は、オブジェクトのデータが完全隠蔽されるがクラスの階層が複雑になるため、アプリケーションの保守性・移植性が悪くなる。単純継承を用いた結果は、オブジェクトのデータを一部公開することになったが、プログラム生産性を向上させるために、これは現実的な方法であると考えられる。

参考文献

［1］松永（他），“事務処理分野へのオブジェクト指向4GL適用の試み（1）”，情報処理学会第45回全国大会，1T-07，1991

［2］BJARNE STROUSTRUP,"The C++ Programming Language", ADDISON-WESLEY PUBLISHING COMPANY,1987