オブジェクト指向フレームワーク再利用性の一実験

5M－7

荒野高志　青野博　藤崎智宏
NTT ソフトウェア研究所

１．はじめに

アプリケーションフレームワーク（あるいは単にフレームワーク）[1]がオブジェクト指向ソフトウェアの再利用性の鍵と言われてから久しい。フレームワークとは、ある特定のドメインのアプリケーション、あるいはサブシステムのスケルトンインプリメンテーションであり、互いに協調して動作するいくつかの抽象クラスと具象クラスからなるものである。しかし、現在のところ、フレームワークの成功したドメインは非常に限られており、グラフィックユーザインタフェース[2]など、数例が報告されているにすぎない。さまざまな分野においてフレームワークを構築し、利用してみる試みが重要となっている。

われわれは、フレームワークの再利用性を調べる実験として、オブジェクト指向で記述された既存のネットワーク管理システムのアプリケーションをフレームワークとして再利用し、交換機管理システムのプロトタイプを作成したので、その経験を報告する。

２．実験の概要

フレームワークとして扱ったアプリケーションは、もともとＡＴＭ交換機のネットワーク管理システム用のものであり、言語はObjective-Cで記述され、総規模約７００クラスであった。機能としては、交換機の架やボード単位の管理、コネクションの設定・解除や、アラーム処理、性能などの統計処理、ネットワーク全体の状態を一望にみるためのアプリケーションなどが含まれていた。

一方、実験で作成するプロトタイプは、交換機をソフトウェアシミュレータとして実現したもので、コネクションの設定・解除や、交換機シミュレータからのアラーム処理を行なうことができる。つまり、もとのフレームワークから、新しい管理対象に対しサブセットの機能を実現したものである。３人の開発者は、いずれもオブジェクト指向の概念、及びプログラミングについては熟知しており、またもとのアプリケーションに精通したアドバイザがついた。

実験の観点としては、どれくらいの規模のものがどのくらいの期間でできるか、どれだけのクラスが再利用でできるか、フレームワーク利用上の問題は何か、などを調べた。

３．実験結果及びその評価

(1) 開発規模及び期間について

規模は、再利用分を含めて、実際に動いているインスタンスが必要とする総クラス数が約１５０、総ステップ数４０Ｋ行のプログラムとなった。機能はかなり豊富であり、フレームワークに含まれていたアラームのフィルタリングやソーティングの機能などをそのまま受け継いで実現している。

これを開発するのに、正味２週間の既存アプリケーションについての教育受講と、正味１週間の設計／コーディングを必要とした。開発の中ではまずスタブでもよいから動くものを作り、徐々に機能を拡張していくいわゆるプロトタイピング型開発を行なった。

それだけの規模のプログラムがそれだけの期間でできた理由は再利用率の高さである。全クラス約１５０のうち、実際に新規にインプリメントしたクラス数が３、サブクラス化により作成したクラスが３、Objective-Cのカテゴリ（既存クラスへのメソッドの追加）を行なったクラス数が２であり、残りの約１４０のクラスはそのまま再利用できた。また、実際にコーディングした量は２Ｋ弱だった。

ただし、１週間という値自体には絶対的な意味があるわけではない。既存アプリケーションに精通したアドバイザの助言があればこそ１週間で開発可能だったわけだし、また、今回の開発は単なる実験であり、要求がラフであったことも追記しておく必要がある。実際の開発では、要求分析はより綿密に行う必要があ

An Experiment for Object-Oriented Framework Reusability
Takashi Arano, Hiroshi Aono, Tomohiro Fujisaki,
NTT Software Laboratories

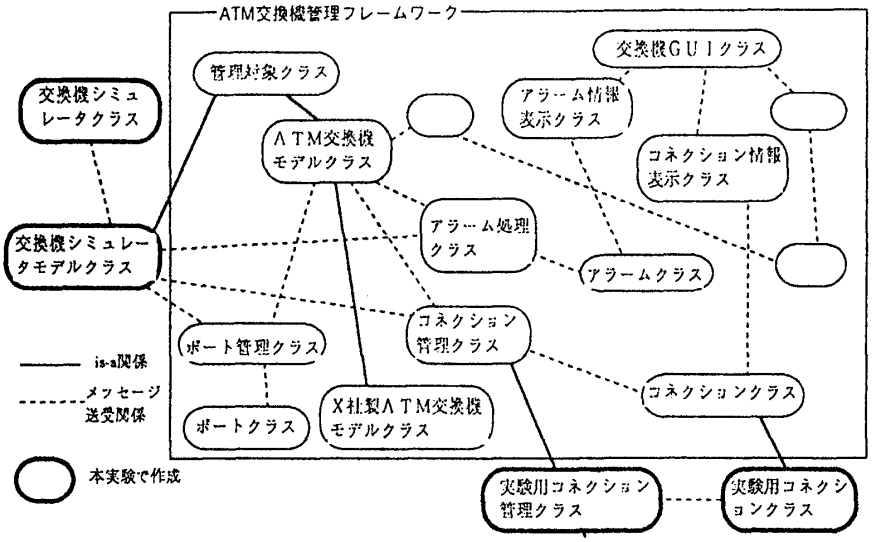

図　フレームワークと本実験プロトタイプのイメージ

る。　しかし、この結果は、ネットワーク管理システムドメインにおいては、モデル化がうまく行なえさえすれば、格段に再利用率があがるというひとつの実例となっており、さらに実開発においても、再利用がうまくいけば、設計・コーディング・テストの工程が大幅に短縮されることにより、従来１～２年かかっていた開発でも１週間ということはないにせよ２～３カ月程度で済むであろう。

(2) フレームワークの問題点

フレームワークの再利用には、まだ解決すべき問題をいくつか実感した。例えば、フレームワークの再利用にあたっては、そのフレームワークを概念レベルから詳細な実現レベルまで十二分に理解しておかねばならないという点がある。これは大変な仕事であり、結局教育やドキュメンテーションの問題に帰着する。実際、われわれの実験でも、たまたまフレームワークの細部を理解しないまま、フレームワーク中のあるコンポーネントとして、既存クラスがあるにもかかわらず、独自の（インタフェースが異なる）クラスを作成してしまい、フレームワークの他の部分と整合がとれずに、その部分をほぼ作りなおしたという経験をした。

また、よいフレームワークを作成するというのも大変である。ある抽象度の処理を適切な抽象度の箇所に記述するというのは、概念設計レベルではさほど難しい仕事ではないが、いざプログラミングレベルになると、他の性能面の要因などとからみ、なかなかうまくいかないものである。フレームワークは、いくつかのアプリケーションを積み重ねて、徐々に成熟させていくべきものであるという、Ralph Johnsonらの指摘[1]は当を得ている。

4．まとめ

この事例はこれからのソフトウェア開発の姿を予見させるものと言えるかも知れないと、われわれは考えている。すなわち、この仮説が正しければ、将来のソフト開発は、従来のように何年もかけてスクラッチから（スタックなど小さな部品を利用するのも含め）開発するということはない。フレームワーク／クラスライブラリの再利用性の恩恵を十分に受け、開発者は要求分析とその要求を反映するためのカスタマイズに専心できる。開発工期は従来の何分の１にもなり、その分、カスタマの要求をより早期にフィードバックをかけることにより、システムをより洗練させていくことができる。

ただし、すべての分野に対して、よいフレームワークが作りうるかという点については、疑問も残る。このドメインは、システムがモデル化している対象物や概念が、装置や回線のような具象物であり、時代によって、改良されていくことはあっても全く概念的に新しいものになることは滅多にないため、フレームワークが作りやすいということは言える。一方、事務処理のように、モデル化すべき対象が、抽象的かつ人為的な概念や規則であり、組織や文化により、あるいは時代により全くその構造を異にするような分野では、難しいかもしれない。今後、さまざまな分野でこのような経験の積み重ねが望まれる。

[1] Wirfs-Brock, R.J. and Johnson, R.E., 1990, A Survey of Current Research in Object-Oriented Design, CACM, Vol. 33, No. 9, pp. 104-124

[2] Schmucker, K.J., 1986, MacApp:An Application Framework, Byte, August 1986, pp. 189-193