分散オブジェクトプラットフォームの持つべき機能に関する一考察*

1K－2

陰山 克禎† 藤崎 智宏† 浜田 雅樹†

NTT ソフトウェア研究所‡

1 はじめに

次世代ネットワーク・コンピューティングの基礎技術として、分散オブジェクト技術が注目され普及しはじめている。現在、多くの分散オブジェクトプラットフォームが存在するが、オブジェクト指向の標準化団体である OMG の定めた CORBA(共通オブジェクト・リクエスト・ブローカ) が業界標準的な地位を築きつつある。

筆者らは、分散オブジェクト技術を利用したネットワーク管理システムを構築しており、分散オブジェクト・プラットフォームとして NEXTSTEP 上で動作する ROF(Remote Object Facility) を利用してきた [1]。これを CORBA に対応させるため、システムの CORBA 環境との統合実験を行っている。本稿では、実験を通じて 2 つの分散オブジェクトプラットフォームの違いを抽出し、両プラットフォームの特徴を述べ、分散オブジェクトプラットフォームが持つべき機能について考察する。

2 比較対象分散オブジェクトプラットフォーム

以下に比較した分散オブジェクトプラットフォームの概要を簡単に述べる。

Orbix2.0(CORBA2.0 準拠製品) IONA 社。最新バージョンである Orbix2.02 で CORBA2.0 完全準拠となっているが、筆者らが使用しているバージョン 2.0 では IIOP(Internet Inter-ORB Protocol) は搭載されていない。

ROF(CORBA 非準拠製品) InvenSys 社。NEXTSTEP 上で動作し、NEXTSTEP に標準で搭載されている DistributedObject(DO) と類似しているが、サブネットを越えたネーミングサービスが可能等が拡張されている。

対象言語は ObjectiveC のみであり、インタフェース定義に必要な言語は存在しない。

3 比較結果及び考察

3.1 開発スタイル

まず、CORBA の静的起動インタフェースを用いた一般的な開発スタイルに着目し比較する。

3.1.1 比較結果

CORBA ではインタフェースをプログラミング言語に依存しない形で定義するため、IDL(インタフェース定義言語) で記述する。このインタフェース定義と具体的なプログラミング言語 (Ver1.2 では C 言語のみ、Ver2.0 では C++、SmallTalk が追加) とのマッピング方法を規定する事により、ポータビリティを実現している [2]。

ROF と Orbix の開発スタイル [3][4] の違いを図 1 に示す。

3.1.2 比較結果からの考察

図 1 からわかるように、CORBA では ROF に比べインタフェースの変更に手間がかかる。

以下に具体例を示す。

インタフェースの増設 CORBA では (1)IDL を書き直しマッピングを再度行ってから実装を追加するか、(2) 別の IDL を新規に作って多重継承する事で可能となる。(2) の場合、増設毎にクラスが増え意味のないクラスが多数存在する事になる。ROF ではプロトコルファイルにメソッド宣言を追加し、サーバの実装ファイルにメソッド本体を追加するだけでよい。

内部メソッドから外部参照メソッドへの変更 CORBA では IDL を書き直し再度マッピングを行ってから実装部分をマージていく。ROF ではヘッダファイル内のメソッド宣言をプロトコルファイルに移動する。

オブジェクト指向による開発プロセスは様々なモデルが提唱されているが、インクリメンタル (incremental) とイテラティブ (iterative) の 2 つの共通とする特徴をもっている。そのイテラティブとは、1 つの開発のなかで分析や設計などの開発プロセスがウォータフォール型のように直線的ではなく繰り返して行なわれる形態である [5]。

CORBA はインタフェースの変更に手間がかかる事によって、このイテラティブな開発プロセスが困難になると考えられる。

3.1.3 解決案の提案

このような問題を解決するために、筆者らは具体的なプログラミング言語で書かれたヘッダファイルからスタブやスケルトンに逆マッピングする手法を考えている。

例えば、新規構築時には IDL 記述からマッピングし、変更時にはヘッダファイルを変更してから逆マッピングを行う。これにより、簡単なインタフェースの変更が可能であると考えられる。

3.2 機能

3.2.1 遠隔に存在するオブジェクトのコピーについて

クライアントが静的なデータを持つ遠隔オブジェクトを参照する場合、遠隔オブジェクトがリファレンスであると、同じデータの通信をサーバ / クライアント間で繰り返す事になる。

この様な無駄な通信を減少させるため、ROF(DO) には bycopy 機能がある。これは、CORBA などで用いているオブジェクトのリファレンスを渡す方法とは異なり、サーバのもつオブジェクトをクライアントにコピーする機能である (図 2 参照)。

ROF では、プロトコルファイルのメソッド宣言部に bycopy を記述する／しないで、コピー／リファレンスの戻り値を選択できる。しかし、CORBA では現在このような機能は定義されていない。

---

*A Study on Required Functions of Distributed Object Platform

†Katsuyoshi KAGEYAMA, Tomohiro FUJISAKI, Masaki HAMADA

‡NTT Software Laboratories

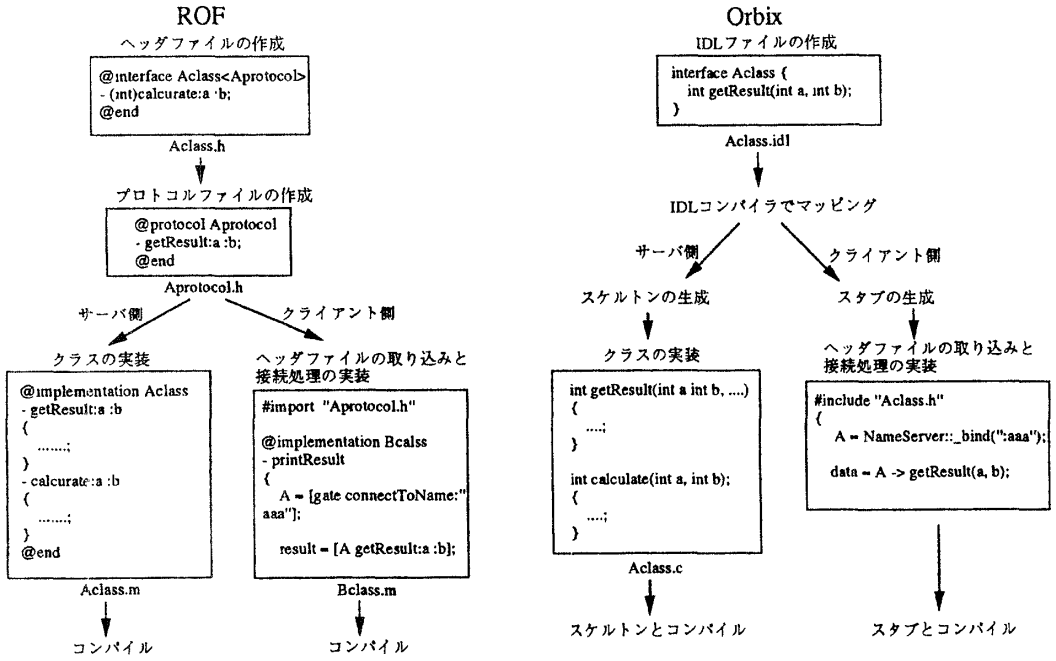

図 1: 開発スタイルの違い

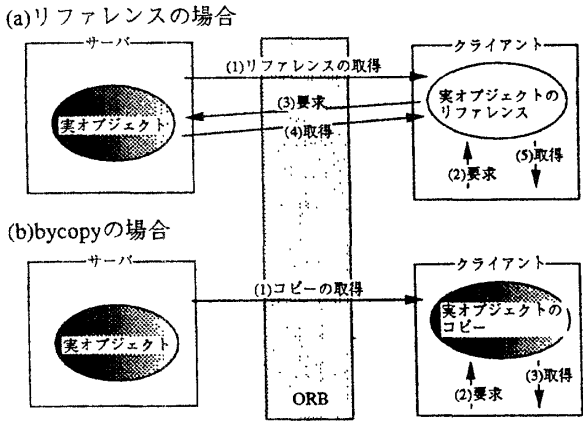

図 2: bycopy 機能

ネットワーク管理システムを分散オブジェクト指向で構築する場合など、ネットワーク機器の静的情報取得には、この bycopy 機能を利用する事で無駄な通信を減少させ処理能力を向上させる事ができる。

3.2.2 ネーミングサービスについて

ネーミングサービスとは、サーバがオブジェクトに名前を付けて登録することにより、名前でオブジェクト・リファレンスを得ることができるサービスである。

ROF、Orbix 共にネーミングサービスを実装しているが、ROF はネットワークで一元的に名前を管理する事により、動作中の全サーバ名を得る事ができる。

CORBA では、このようなネットワークワイドなネーミングサービスは規定されていない。これは、ORB の活性化処理により、クライアントはサーバが動作中であるか否か意識する必要はないためであると思われるが、アクセス可能なオブジェクトの名前一覧を取得したいような場合には対処できない。

4 結論

CORBA における開発手法は言語に依存されず、ポータビリティに優れている。しかし、インテラティブな開発プロセスをとるオブジェクト指向開発においては受け入れにくい。そこで、インタフェースの変更に柔軟に対応するには有効な解決案が必要であると考える。

また必要な機能として、2 つのサービスについて述べた。OMG でも現在新サービスの審議中であり、順次追加される予定であるが、本稿で述べたサービスも必要であると考える。

5 今後の課題

今後は、CORBA を用いた開発と ROF を用いた開発に関し、その開発容易性を定量化し、比較検討していく。また、筆者らが考案した CORBA を用いた開発手法について、その有効性を確認する。

参考文献

[1] Takashi Arano et al. "A Computer Network Management System Platform based on Distributed Objects". *IFIP/IEEE DSOM '95.*

[2] 小野沢 博文. *"分散オブジェクト指向技術 CORBA".* 株式会社 ソフト・リサーチ・センター, 1996.

[3] NeXT Computer Inc. *"NEXTSTEP OBJECT-ORIENTED PROGRAMMING AND THE OBJECTIVE C LANGUAGE".*

[4] IONA Technologies Ltd. *"programming guide".*

[5] 本位田 真一 他. "オブジェクト指向分析・設計". 共立出版株式会社, 1995.