3T－1

分散環境下における自律型オブジェクトによるアプリケーション構築モデル

岩尾　忠重　岡田　誠　竹林　知善
株式会社　富士通研究所

1. はじめに

コードの再利用によって、プログラマが記述しなければならないコード量の軽減を目的にオブジェクト指向プログラミングの導入がされ一般的になってきた。しかし、その一方で、近年のアプリケーションやシステムは複雑さを増し、オブジェクト間の関係や内部状態は煩雑さを増している。そのため、利用するオブジェクトのための記述や仕様変更などが頻繁に発生し、いまだプログラマは本質的でない周辺部分のコーディングに多くの時間を取られている。この問題はオブジェクトの関係を予め詳細に規定しなければならないモデルであるためである。

そこで、我々は、オブジェクトの関係を予め厳密に規定をせず、変更に対して柔軟に対応可能なオブジェクト分散環境を前提とした自律型オブジェクトと非自律型オブジェクトによるアプリケーション構築モデルの考察を行っている。本論文ではこのモデルについての報告を行う。

2. 背景

CORBA[1]や DCOM[2]などによりオブジェクト指向プログラミングは、ネットワークの場所に依存せず、また、プログラムコードモジュールとしての独立性が高くなってきた。しかし、実際のインプリメントを考えたとき、オブジェクトの利用には、利用されるオブジェクトの IDL のみならず、そのパラメータの意味などもあらかじめ熟知していなければならず、プログラム上のロジックとしての独立性はそれほど高いとはいえない。また、オブジェクト間の機能インタフェースが良く設計されていないと利用されるオブジェクトのバージョンアップや仕様変更に伴い、これを利用するオブジェクトの変更も余儀なくされる。しかし、最初からすべて設計できるわけでなく、何度も仕様変更やバージョンアップを行っているのが現状であり、開発効率が飛躍的によくなっているわけではない。

3. 柔軟な協調のための条件

バージョンアップや仕様変更の問題はオブジェクトの関係を強く意識する必要があることが原因であると我々は考えている。

そこで、我々は、 CORBA や DCOM を補完しながら、同時に、利用し利用される関係（サーバクライアント）でない柔軟なオブジェクトの連携について考察した。オブジェクトはそれぞれ独立して周りのオブジェクトを意識せず、「場」を用いてお互いに情報交換だけを行う。**(会話の場所)** 問い合わせや機能要求は答えを期待せずに、オブジェクトがその内容に応じてそれぞれ独立に反応する。**(会話の内容と動作)** そして、これら相手を意識しないオブジェクトの連携動作がアプリケーションとして動作**(適応と協調)**するようなモデルである。

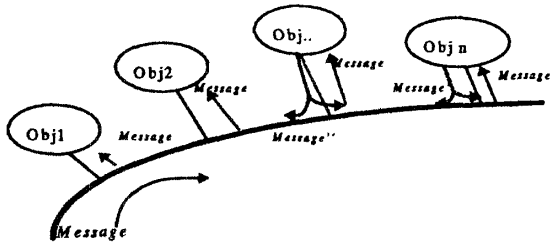

図１　オブジェクト会話モデル

(1) 会話の場所

本モデルでは、予め周りの特定のオブジェクトを意識しないことが重要である。オブジェクト間の会話を成立させるためには、共通に各々のオブジェクトが会話を認識できる場が必要となる。そこで、本モデルでは、各々のオブジェクトが会話を感知できる共通の会話の場所を設け、各々のオブジェクトは、この場所を流れる会話の内容やさまざまな情報を感知させるようにした。

(2) 会話の内容と動作

共通の会話の場所にさまざまな情報が流れるがこれらの情報に対して選択的に反応する必要がある。しかし、情報フォーマットがオブジェクトによって定義されるならば、結局、それぞれのオブジェクトが相手を意識する必要があり意味が無い。また、細かいフォーマットでは、汎用性がない。

そこで、本モデルでは会話の内容のフォーマットの大枠のみを定義する。そして、この会話の内容とメソッドのマッピングをテーブルによって管理し、ダイナミックに会話の内容とメソッドのマッピングが可能とさせた。

(3) 適応と協調

オブジェクト間の会話が成立し、内容によってメソッドが反応することができたとしても、そのオブジェクトの環境に適応するためには、その環境で話される会話の内容がテーブルにかかれなければならない。我々は環境に適応・協調するオブジェクトを自律オブジェクトと呼び、単にメッセージに反応するオブジェクトを非自律オブジェクトと呼ぶことにする。

Field Reactor Model for Distributed Objects
Tadashige Iwao, Makoto Okada, Tomoyoshi Takebayashi
Fujitsu Laboratories Ltd.
Internet: iwao@flab.fujitsu.co.jp

4. Field−Reactor モデル

上記を具体化したモデルを Field-Reactor モデルと我々は呼ぶことにする。Field-Reactor モデルは Field、Reactor、IRP の 3 つの部分より構成される。IRP は自律オブジェクトだけがもつ機能部位である。なおこの Field-Reactor モデルは CORBA や DCOM 上にも構築可能である。

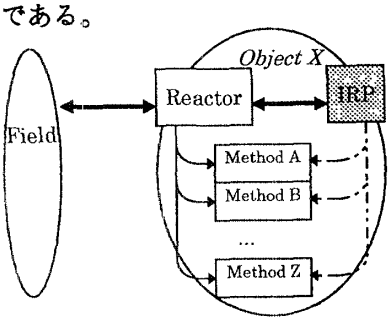

図 2　自律・非自律オブジェクトの内部構造

Field は会話の場所そのものであり、オブジェクトへの要求メッセージやさまざまな情報が流れる。Field には Common Field と Application Field がある。Common Field はすべてのオブジェクトが参加する共通の Field である。Application Field は Application が定義したそれぞれ固有の Field であり、その Application に関連するオブジェクトが参加する。Field は具体的には、UDP のブロードキャストであったり、IrDA であったり、IRC のチャンネルである。メッセージの内容は SVO の形をしており、すべてのメッセージは文字列である。ここでは柔軟性を考慮して、メッセージ内容にワイルドカードを含めた。

Reactor は、Field の会話内容とオブジェクトのメソッドを結び付けるテーブルである。テーブルの項目は、マッチするメッセージ内容と発火されるオブジェクトのメソッドのポインタからなる。登録する項目のメッセージの各部分文字列にはワイルドカードの使用も可能とした。

IRP(Inner Reconstituting Part)は協調・適応のための部位である。オブジェクト定義の環境の状態を重み付けし、記憶する。オブジェクトは現在の環境から次に起こすべきアクションを IRP から得えたり、IRP の記憶を元に Reactor のテーブルを環境に応じてダイナミックに書き換えることができる。

適応と協調の 1 つの解決方法として、すべての会話に反応し、あるメソッドが成功したとき、その会話内容とそのメソッドをダイナミックにテーブルに登録することで、その後、その会話内容に特定のメソッドが反応するようになり、そのオブジェクトがその環境に適応したかのように振る舞う。しかし、すべてのメソッドに対してこのような動作をしてよいわけでない。また、会話の内容とメソッドはマッピングすることは可能であるが、メソッドを組み合わせ動作させるには、メッセージの生成が必要となる。IRP の機能により、動的にメッセージ群を学習・生成することが可能となるが、直接必要でないメッセージ群を振り分けられないなどの課題が残る。

5. アプリケーション例

Field Reactor モデルのアプリケーションとしてネットワーク上の複数マシンでのマウスポインタ共有をとりあげる。もちろん CORBA や DCOM などのオブジェクト連携だけでもこのようなアプリケーションは作成可能であるが、レスポンス性などから実用は難しい。

このアプリケーションは、2 つの非自律オブジェクトから構成される。1 つはフィールドに対してマウスポインタに関するメッセージを投げる Mouse Obj オブジェクトであり、そのメッセージが受信されることを期待しない。もう 1 つのオブジェクトは、Field に流れてくるマウスポインタに関するメッセージに反応し、そのオブジェクトが動作する端末上のポインタ操作を行うものである。このオブジェクトもメッセージが必ずくることを期待しない。

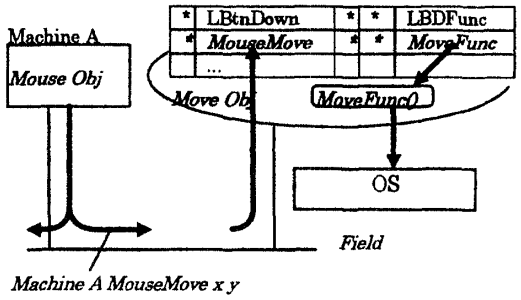

図 3　アプリ例（マウスポインタ共有）

しかし、それぞれのオブジェクトを動作させるとリモートにポインタ操作を行うことができ、あたかも協調動作しているかのように見える。それぞれのオブジェクトがお互いを強く意識せず、Reactor テーブルを柔軟かつダイナミックに変更できるため、1 台の端末から複数台の端末を、また、複数端末から 1 台の端末の操作が可能となる。また、別の Mouse Obj のメッセージを受信するオブジェクトとして、情報を発する端末別に最後に情報が発生した時間を表示することで、アウェアネス的なアプリケーションとすることも可能である。このとき、Mouse Obj の変更は全く必要ない。

6. まとめ

本稿では、オブジェクト間の関係をあまり意識しない Field - Reactor モデルとそのアプリケーション例についての報告を行った。

[1] OMG　http://www.omg.org/corba/
[2] DCOM - A Technical Overview
http://www.microsoft.com/msdn/sdk/techinfo/