マルチメディア通信と分散処理 67－11
（１９９４．１２．２）

分散オブジェクト開発支援用ツールキット

森 健　乙川 進一　中澤 修
沖電気工業株式会社 マルチメディア研究所

OMG/CORBA 準拠の分散オブジェクト環境上で動作する分散オブジェクトと、それを利用するアプリケーションプログラム(AP)の作成を支援するツールキットを開発した。
本ツールキットはオブジェクト指向言語 C++で記述したオブジェクトを、分散オブジェクトに変換するトランスレータ、オブジェクトやアプリケーションの記述を支援するオブジェクトライブラリ、オブジェクトの検索用ツール、登録用コマンド、プログラム実行時の AP からオブジェクトへの通信の可視化ツールから構成される。

A Toolkit for Distributed Objects Development

Takeshi Mori　Shinichi Otokawa　Osamu Nakazawa
Multimedia Laboratory,Oki Electric Industry Corporation

We developed a toolkit which supports development of distributed objects and distributed applications that works on the OMG/CORBA distributed object environment.
This toolkit includes a translator which translates C++ objects to the CORBA distributed objects, C++ class libraries for distributed application programming, a object retrieving and registrating tool, and a visualization tool that visualizes communications between an application program and distributed objetcts.

1 はじめに

近年、分散コンピューティングの新たなプラットフォームとして、分散オブジェクト環境が注目されている。分散オブジェクト環境は Object Management Group(OMG)による標準化が進行中で[1]、特にオブジェクト間の通信機構の標準仕様である Common Object Request Broker Architecture(CORBA)[2][3]の策定により、多くの分散オブジェクト環境の製品が発表／発売されつつある[4]。この CORBA 準拠の分散オブジェクト環境を利用すると、ネットワーク上でのデータの位置やプロセス間通信の管理が簡単になり、分散アプリケーション(AP)の作成が容易になると言われている。

我々は、実際に分散オブジェクト環境上で簡単な分散 AP の作成を行い、分散オブジェクト環境の使い勝手を評価した。その結果、分散オブジェクト環境の使い勝手にはまだ問題があり、分散 AP の作成が十分に簡単になったとは言い難いとの結論を得た。

この結果をふまえて、OMG/CORBA 準拠の分散オブジェクト環境上で動作する分散オブジェクト及び分散 AP の作成を支援するツールキット[5]を開発した。ベースとした分散オブジェクト環境は HD-DOMS Ver1.1[6]で、CORBA Ver1.1 に準拠したものであるが、将来的なプラットフォームの変更を考慮し、極力 HD-DOMS に依存しないかたちで設計している。AP 作成に使用するプログラミング言語は C++を基本としている。

2 ツールキットの目標

2.1 分散オブジェクト環境の問題点

分散オブジェクト環境の評価にも、前述のように HD-DOMS Ver1.1 を使用した。評価から得た分散オブジェクト環境の問題点を以下に示す。

- 分散オブジェクトの記述のために新たにインタフェース記述言語(Interface Definition Language:IDL)を使用する必要があり、特に単一言語環境に慣れたユーザにとって大きな負担となる
- AP からオブジェクトのオペレーションを呼び出す際にシステムが提供する API ルーチンを使用する必要があり、AP 作成者は数多くのルーチンを使いこなさなければならない
- オブジェクトの登録作業に手間がかかり、ユーザのオブジェクトの利用を妨げている

2.2 目標

上述したような問題点をふまえ、本ツールキットでは、分散オブジェクト及び分散 AP(分散オブジェクトを利用するクライアント)の両方の作成を支援する。具体的には以下の支援を行う。

2.3 分散オブジェクト／分散 AP 記述支援

CORBA 仕様に準じた分散オブジェクト環境では、分散オブジェクトの定義には二つの記述を必要とする。オブジェクトのインタフェースの記述(IDL)と、そのオブジェクトが持つオペレーションのインタフェースの記述(IDL)及びインプリメンテーションの記述(C 等による)である。これを C++による記述に一本化し、分散オブジェクトの記述を容易にする。また、クライアントから分散オブジェクトのオペレーション呼び出しを行う場合、分散オブジェクト環境が提供するルーチンを利用する必要があるが、これを一般の C++メンバ関数呼び出しの形で記述可能とし、クライアントの記述を簡易化する。これらを実現するツールを以下に示す。

2.3.1 トランスレータ

C++で定義したオブジェクトを分散オブジェクト化するためのトランスレータで、C++ によるオブジェクトの記述から、IDL のインタフェース記述(オブジェクト及びオペレーション)と、C++のインプリメンテーション記述(オペレーション内容)の両者を自動生成する。また、クライアントでは C++オブジェクトのメンバ関数呼び出しが定義されるが、これを分散オブジェクトのオペレーション呼び出しに変換するためのクラスライブラリも自動生成する。クラス自体はユーザが定義した C++クラスと同様だが、その C++オブジェクトのメンバ関数の中身を、そのオブジェクトに対応する分散オブジェクトの、そのメンバ関数に対応するオペレーション呼び出しに置き換える。この C++ オブジェクトを、分散オブジェクトの代理という意味合いから「代理オブジェクト」呼ぶ。図 1に代理オブジェクトを経由したオペレーション呼び出しの概念図を示す。

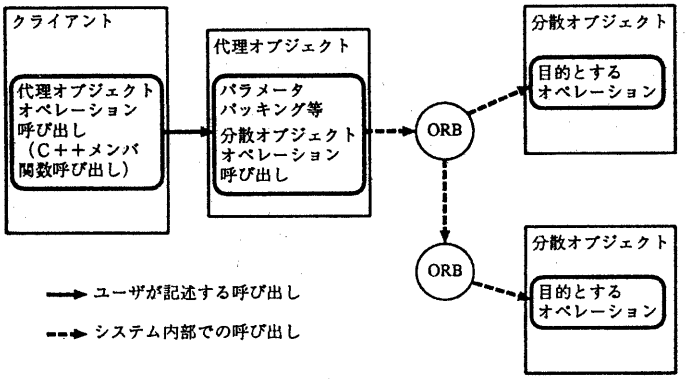

図 1: 代理オブジェクト経由のオペレーション呼出し

また、生成したオブジェクトを分散オブジェクト環境に登録するためのコマンドも同時に生成する。更に、これらの分散オブジェクトや代理オブジェクトの記述の生成時に、後述するロギングを行うメカニズムを、オプションによりコードの中に自動的に埋め込む。

またC++オブジェクトの解析時に、後述するオブジェクト検索ツールが使用する各クラスに関する情報(継承、属性、メンバ関数等)をファイルに書き出す。このように、トランスレータは他のツールが使用するデータ等も生成するため、本ツールキットの中心的存在となっている。

2.3.2 オブジェクトライブラリ

分散オブジェクト環境が提供している分散オブジェクトライブラリをC++で再記述したC++クラスライブラリで、ユーザの定義するC++オブジェクトをシステムが提供するオブジェクト群中に取り込むことができる。

2.4 分散オブジェクト検索・登録支援

オブジェクトの分散オブジェクト環境への登録作業は、オブジェクトの登録、オペレーションの登録、オブジェクトのインスタンスの登録という三つの作業からなる。これを容易にするための登録用コマンドを提供する。これは前述のトランスレータにおいて、オブジェクトの各記述を生成するのと同時に登録用コマンドを生成する。また、オブジェクトの継承関係や機能の検索を容易にするためのツールを提供し、オブジェクトやクライアントの開発を支援する。

2.5 デバッグ支援

作成されたオブジェクトとクライアントのデバッグを支援するために、クライアントとオブジェクトの間でORBを介したどのようなメッセージのやりとりがあったかをロギングし、それを可視化するツールを提供する。これによりユーザプログラムレベルでの、プログラムの構成／オブジェクト間通信等に関する問題点の洗い出しを容易にする。

以上述べた本環境の、分散オブジェクトとクライアントの作成支援の全体構成を図2に示す。

3 トランスレータ

前章で述べたように、本ツールキットの中心となるのはトランスレータである。このトランスレータをC++オブジェクトから分散ブジェクトを生成するということで、c++2idl と呼ぶ。本章ではこのトランスレータについて述べる。

3.1 構成

c++2idl はパーザと IDL 生成部で構成される。

パーザは C++で記述されたオブジェクトを解析し、内部形式に変換して IDL 生成部に渡す。この時、解析できる C++の構文にはかなりの制限を設けている。これは、C++でのオブジェクトの定義を、IDL での記述に近い形で行っているためである。このた

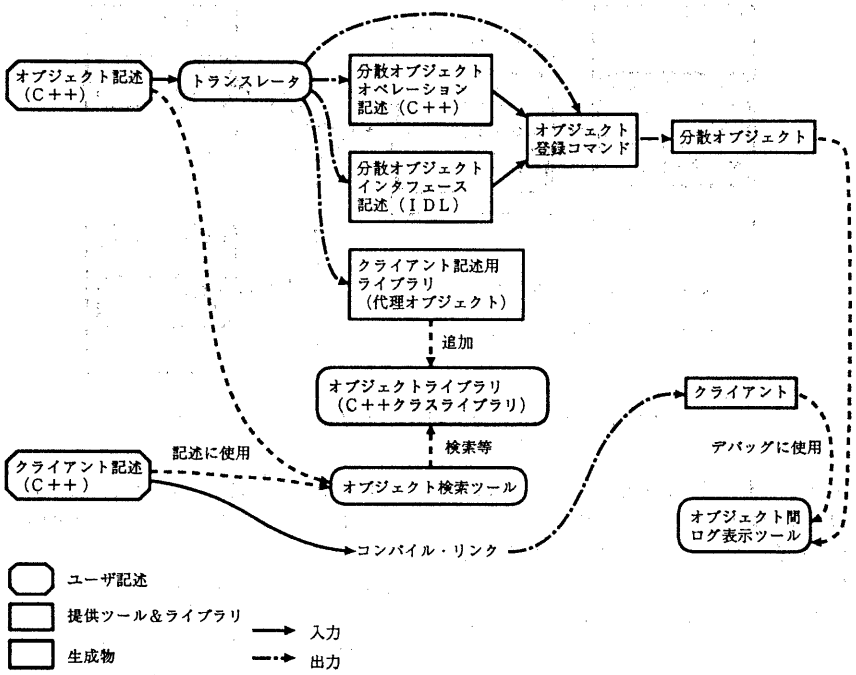

図 2: 分散 AP 作成環境の全体構成

め、例えば型定義の宣言を省いたり、メンバ関数の引数に input/output の指定を加えたりしている。

IDL 生成部では、使用する分散オブジェクト環境に依存する部分、例えばオブジェクト登録用コマンド名等を、設定データとして外部ファイルから読込む。このため、ファイルの内容を変更することで、様々な分散オブジェクト環境に対応できる。

IDL 生成部ではオブジェクト及びオペレーションを生成している。オブジェクトの生成は、C++のクラス定義の部分を変換することで行う。この時、生成したオブジェクトの登録用コマンドも生成する。正確には、コマンドはオブジェクトを登録するものと、登録したオブジェクトのインスタンスを生成するものの二種である。また、オペレーションの生成はC++のメンバ関数の定義を変換することで行う。この時、オペレーション登録用のコマンドも生成する。

オペレーションのインプリメンテーションには、二種類の方法が選べる。一つは動的呼び出しによるものであり、もう一つはプログラム(a.out 形式)をインタプリタ(シェル等の)からオペレーションとして呼び出すものである。以下ではこの両者について述べる。

3.1.1 動的呼び出し

現在、CORBA では IDL と C のマッピングは規定されているが、IDL と C++のマッピングは規定されていない。このため、我々が評価した分散オブジェクト環境も、動的呼び出しのためのインプリメンテーションは C で記述することが想定されていた。つまり、IDL で記述されたオペレーションのインタフェース(オペレーション名)に対し、同じ名前を持つ関数を結び付ける機構が用意されていた。ところが、C++のメンバ関数として記述したインプリメンテーションを C++のコンパイラにかけると、コンパイラがユーザの定義した関数名を別の関数名に置換えるため、オペレーションのインタフェースと結びつけられず、オペレーションのインプリメンテーションを呼び出せないという問題が発生した。そこで、以下の方法でインタフェースとインプリメンテーションを結びつけた。

1. c++2idl でインプリメンテーション/登録のためのIDL を作成
2. インプリメンテーションを C++でコンパイル(*.o ファイル生成)
3. UNIX の nm コマンドにより*.o ファイルからシンボル情報を取得
4. シンボル情報をオペレーション登録のためのIDL ファイルに反映
5. IDL を登録

つまり、C++コンパイラでコンパイルした結果から置き換えられたメンバ関数名を取り出し、それをインタフェース定義側に反映するのである。これにより、C++コンパイラにかけた C++メンバ関数をオペレーションのインプリメンテーションとすることができた。

3.1.2 a.out 形式のオペレーションの呼出し

c++2idl では C++のメンバ関数の記述に対し、それを C++でコンパイルして a.out 形式にできるような、main 関数を持った C++のソースを生成する。そして、そのソースを C++ でコンパイルする。つまり、メンバ関数毎に a.out 形式の実行ファイルが生成される。この形式のオペレーションは、呼び出し時のパラメータや戻り値に制限があるが、逆に既存のプログラムを分散オブジェクトの枠組に取り込み、オブジェクトのオペレーションとして動作させることができる。この形式のオペレーションを呼び出すと、以下のような処理が行われる。

1. a.out を実行するためのスクリプトの呼出し
2. a.out に渡す引数の処理
3. a.out を実行し、その結果をファイルへと書き出す
4. 結果を格納したファイルの操作と、結果の返却

3.2 トランスレータの問題点

上述してきたように、本トランスレータには幾つかの問題点がある。特に、オブジェクトを記述する C++の構文に制限が多い。しかし、それらの多くは IDL と C++のマッピング仕様が決定されれば解決可能である。また、a.out 形式のオペレーションについても値の受渡し等、制限事項が多いが、これもプラットフォームとなる分散オブジェクト環境による部分が多い。このため、これらの問題への本格的な対処は、IDL と C++ のマッピングが CORBA で仕様化された後に行う予定である。

4 その他のツール

4.1 分散オブジェクトライブラリ

前述のように、本ライブラリは分散オブジェクト環境で提供している分散オブジェクトを C++オブジェクト化したオブジェクト群を基本としている。さらに前述の代理オブジェクト群もここに加え、分散 AP で本ライブラリを読み込む事によって、AP の記述を簡易化することができる。現在は、HD-DOMS で提供している分散オブジェクトから基本的な以下のものを選択し、ライブラリとしている。

- Root：オブジェクトの最も基本的な枠組を提供する
- Item：オブジェクトの位置情報等の属性を提供する
- Bin：他のオブジェクトを保持するための最も基本的な枠組を提供する
- ChildBin：他のオブジェクトを子として保持するための枠組を提供する
- MemberBin：他のオブジェクトを子として、かつ順序づけて保持するための枠組を提供する
- Contained：他のオブジェクトに保持されるオブジェクトの枠組を提供する
- Container：他のオブジェクトを保持(子又はメンバとして)するオブジェクトの枠組を提供する

これらは皆インスタンス作成不能な仮想クラスであり、分散オブジェクトの持つ基本的な機能、属性を提供するものである。

4.2 オブジェクト検索ツール

C++クラスライブラリを検索するツールで、クラス継承木の表示、各クラスの詳細情報(データメンバ、メンバ関数、それらの詳細等)表示等の機能を提供する。本ツールが用いるオブジェクトの情報は、トランスレータで C++オブジェクトを解析する時に作成し、ファイルに書き出している。図 3にオブジェクト検索ツールのウインドウイメージを示す。

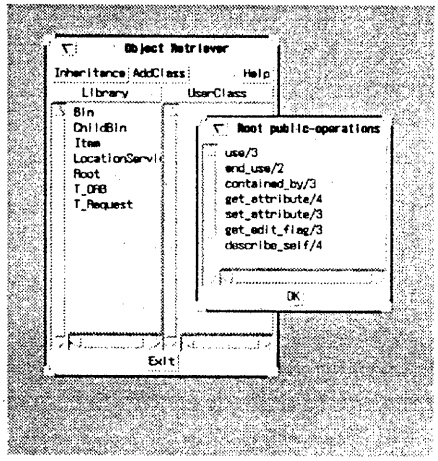

図 3: オブジェクト検索ツール

4.3 ログ収集&表示ツール

トランスレータでの C++代理オブジェクトの作成時に、代理オブジェクトのメンバ関数内に、分散オブジェクトへのオペレーションリクエストの送信をロギングするコードを埋め込む。またユーザが定義したオブジェクトに関しては、リクエストの受信時にロギングを行うコードを、同じくトランスレータでの分散オブジェクト生成時に埋め込む。送/受信側双方でログを取ることにより、両者の間の ORB の不具合と、オブジェクト側での不具合を見分けることができる。ログの内容としては時刻(呼び出し/呼び出され/応答(又はオペレーション終了)/呼び出し完了)、対象オブジェクト名、呼び出しオペレーション名、呼び出し及びオペレーションの種類(同期/非同期)を採取する。

一方、表示ツールは実行後にログ内容を視覚的に再現するツールであるが、現状では単にログの内容をウインドウ上に時間順に表示している。

5 おわりに

OMG/CORBA 準拠の分散オブジェクト環境上での分散オブジェクト及び分散 AP の開発を支援するツールキットについて述べた。本ツールキットはまだβ版であり、上述してきた内容からわかるように、未完成の部分も多い。今後は各ツールの完成度を上げると共に、各ツールの連係の強化や新たな機能の導入を考えている。特に、グラフィカルな AP の記述をサポートするエディタを提供し、前述のオブジェクト検索ツール、ログ収集/表示ツールとの連係を図る予定である。グラフィカルエディタの開発にあたっては、並行して研究を進めているプロセスツールキット(DOL)[7] での先行開発の成果を流用する予定である。

参考文献

[1] 大野、佐藤, "オブジェクト・マネージメント・グループとその活動", 情報処理学会誌, Vol.35, No.9, pp.845-852, 1994

[2] "The Common Object Request Broker : Architecture and Specification 1.2", OMG TC Doc 93-12-43

[3] 佐藤、大野, "共通オブジェクトリクエストブローカアーキテクチャCORBA1.1", 情報処理学会誌, Vol.35, No.9, pp.853-858,1994

[4] "SOMobjects Developer Toolkit : An Overview", IBM, June 1993

[5] 乙川、森、中沢, "分散ソフトウェア開発用ツールキット", 情報処理学会第 48 回全国大会予稿集 6D-8, 1994

[6] "Designing and Implementing Applications with the HyperDesk™ Distributed Object Management System™ : A Technical Overview", HyperDesk™ Corporation, 1992

[7] 鈴木、中沢、佐藤, "分散ソフトウェア開発用ツールキット—ライブラリ—", 情報処理学会第 48 回全国大会予稿集 6D-6, 1994