ソフトウェア工学 118－14
（1998．3．10）

# メタ階層モデル記述言語 Bramble の分散オブジェクト環境への対応

藤田 充典　小飼 敬　上田 賀一

茨城大学 工学部 情報工学科
〒316-8511 茨城県日立市中成沢町 4-12-1

ソフトウェア開発の有効な手段のひとつにプロトタイピング手法がある．本研究室では，これにドメイン指向の考えを取り入れたメタ階層アーキテクチャによるモデル開発環境を研究してきた．今回，モデル開発環境上のオブジェクトの分散化について検討を進め，その実現のために，本研究室で開発されているモデル記述言語 Bramble を分散オブジェクト技術である CORBA に対応させた．これにより Bramble によって記述されるグループ開発支援環境における開発者間のコミュニケーションや，オブジェクト間のメッセージ通信等の実現が容易になる．本報告では，この技術を利用することにより，成果物や開発プロセスの管理を行なうリポジトリシステムを構成するオブジェクト指向データベースの操作や，モデリング環境で作成された成果物の管理等を分散ネットワーク上で行なう応用例を示す．

# Adaptation of Meta Hierarchical Model Description Language: Bramble to Distributed Object Environment

Mitsunori Fujita　Kei Kogai　Yoshikazu Ueda

Ibaraki University
4-12-1 Naka-Narusawa, Hitachi, Ibaraki, 316-8511 Japan

A prototyping approach is one of the effective method of software development. Our laboratory has been researching a modeling environment based on a meta hierarchical architecture which has the concept of domain orientation. In this report, to make the distributed modeling environment, distributed object technology CORBA is introduced into model description language Bramble which has been developed in our laboratory. This way can make it easy to realize the communication among the developers and the message communication between objects on group development environment in the Bramble language. This report presents two application examples using this technique. One is the communication of remote object-oriented database to treat the products and development processes. Another is the management of products in distributed modeling environments.

# 1 はじめに

近年，次世代ネットワーク環境の基礎技術として，分散オブジェクト技術が注目されている．分散オブジェクト技術とは，従来のオブジェクト指向技術を分散環境上に当てはめ，ベンダやOSに依存しない分散オブジェクトのための共通の仕様を定めることで，分散環境上であることを意識せずに開発を行なうための技術である．分散オブジェクト技術を用いれば，開発者がネットワークのリモートシステムにあるオブジェクトを利用する際に，リモートオブジェクトであることを意識せずにオブジェクトが利用できる．最近では分散オブジェクト技術として，Horb[1]，DCOM[2]等が研究されているが，中でもOMGのCORBA[3][4]は，各方面で取り上げられその関心の高さを示している．

一方，本研究室で研究/開発が行なわれているメタ階層モデル記述言語Brambleは，前バージョンにおいて，低レベルのコミュニケーション用ライブラリが用意されていたが，簡便にリモートシステム上のオブジェクトを利用することができず，そのためにオブジェクト間の通信は主に単一マシン上で行われてきた．しかし，急速に普及するネットワーク環境において，分散開発を進めるためには通信機構の充実が必須である．また，開発において利用されるOODBとの通信手段も，柔軟に定義可能でなければならない．そこで，Brambleの将来性と汎用性のためにも，しっかりした基盤を持つ規格に対応させる必要があり，分散オブジェクト技術として有望なCORBAへの対応を目指すことになった．また，BrambleをCORBAに対応させることは，Brambleにより記述されるグループ開発支援環境[5]の構築のためにも非常に有益であると思われる．

# 2 基盤技術

## 2.1 CORBA

OMGが提唱する分散オブジェクト・モデルを説明したOMAの中で，オブジェクト間通信のメカニズムを提供するコンポーネントとして位置付けられているORBの仕様がCORBAである．ここでORBの役割とは，クライアントが発行するリクエストをサーバのオブジェクトに届け，その応答をクライアントに返却することである．これをアプリケーションから利用可能にするためにCORBAは，以下の仕様を定めている．

- インタフェース定義言語(IDL)
- 言語マッピング
- オブジェクトの生成やリクエストの配信をはじめとする各種コンポーネントの機能とインタフェース
- オブジェクト間通信のためのプロトコル

## 2.2 メタ階層アーキテクチャ

大規模なシステム開発において，ユーザと開発者との間で意味的な隔たりが大きくなることがある．これを解決するためには，システムの実行可能なモデル(プロトタイプ)を早期に作成し，動作確認をすることで要求を明確にしていくプロトタイピングの手法が有効になる．

そこで，本研究室ではこのプロトタイピングを容易にするため，モデルベースソフトウェア開発基盤となるメタ階層アーキテクチャに関する研究[6]を行ない，以下の項目について取り組んでいる．

- システムを構成するコンポーネントがドメイン毎に予め用意され，それらを使って視覚的にシステムを記述し，シミュレートすることで動作確認を行うことができるプロトタイピング環境を構築する．
- この環境上で広範囲のソフトウェアドメインをサポートしたプロトタイピングを可能とするために，コンポーネントの定義とそれから定義されるプロトタイピングツールを生成する．

メタ階層アーキテクチャは，3階層の構造から成り立っている．階層上に存在するモデルには，まずエンドユーザが作成するレベルのモデルがあり，これらのモデルを規定するためにモデリング対象世界のドメイン毎に定義されているメタレベルのモデルがあり，このメタレベルのモデルを規定するためのモデルとなるメタメタモデルがある．これらの内，メタメタモデルはモデリング環境によ

り最初から提供され，モデリング環境のユーザが生成･編集するのはメタレベルとベースレベルのモデルである．各階層間のモデルにはモデル-メタモデルという関係が存在する．つまり，ベースレベルモデルのメタモデルはメタレベルモデルであり，メタレベルモデルのメタモデルはメタメタモデルとなり，メタモデルがモデルを記述し解釈する．

そして，これらのモデルを編集するためのツールは，"モデリングのためのツール"というドメインに特化したメタレベルモデルから規定されるモデルとなるモデラとして存在する．モデラはそれぞれのドメインに適したメタモデルが持つ．従って，ユーザのメタモデルが持つモデラが必要である．

このように，メタレベルのユーザはメタメタレベルモデルからそれぞれのドメインに適した開発環境をベースレベルのユーザに提供することができ，ベースレベルのユーザは自分が十分に管理している専門のドメイン内でモデリングすることが可能である．

### 2.3 Bramble

Bramble とは本研究室で研究/開発されている言語であり，以下の様な特徴を持っている．

- 言語の全ての要素がオブジェクトである

  オブジェクトの集合は，その集合で一つのオブジェクトとして定義される．また，言語自身が振舞うために必要な要素もオブジェクトとして持つ．

- コピーベースのオブジェクト指向言語である

  これによってクラスやインスタンスの概念が無くなる．しかし，ユーザの扱い方にによって，オブジェクトをクラスやインスタンスの様に振舞わせることができる．

- オブジェクトは属性だけを含む

  オブジェクトは変数とメソッドを区別せず，属性としてオブジェクトを包含する．

- 型付けが無い

  全てをオブジェクトとして扱っているため，メソッド呼出し時等の型のチェックが無い．従って，ユーザは型付けの厳しい言語に比べて，より柔軟なプログラミングが可能になる．

- バイトコードによって仮想機械化を実現している

  Bramble 上で生成されたオブジェクト群は，バイトコード化されることによって，実行環境外でも存在することができる．実行環境ではその計算機で実行可能な Bramble の仮想機械を用いて，そのバイトコード化されたオブジェクト群を解釈し，評価・実行する．

Bramble のコアを形成するベースオブジェクトの種類と特徴を以下に示す．

| Object | 特徴 |
|---|---|
| Array | 名前と値を持つ要素の集合を扱う |
| Boolean | 真偽値を扱う |
| Bytecode | バイトコードを扱う |
| File | ファイルを扱う |
| Integer | 整数を扱う |
| List | 値だけを持つ要素の集合を扱う |
| Method | メソッドを扱う |
| Null | NULL を表す |
| String | 文字列を扱う |
| System | Bramble のシステムを扱う |
| Thread | スレッドを扱う |
| User | ユーザ定義情報を扱う |

これらのベースオブジェクトは，メッセージを受け取ることで様々な振舞いが可能となる．また Bramble 自身の拡張は，追加するオブジェクトの初期化と，ライブラリのリンクを行なうだけで実現可能であり，ベースオブジェクトに対するソースの変更は必要としない．これらは Bramble で開発を行なう上での大きな利点である．

## 3 Bramble/CORBA の設計

Bramble を CORBA に対応させることで，CORBA に準拠したリモートシステムにあるオブジェクトや，他言語で書かれたアプリケーションを扱える等，様々な利点を得ることができる．

Bramble の分散オブジェクト環境 CORBA への対応を目指すにあたって，開発する Bramble を以後 Bramble/CORBA と呼ぶ．

### 3.1 Bramble/CORBA の設計方針

主な設計方針について以下に述べる．

- IDL による表記に対応させる

CORBA を構成する重要な要素である，IDL によって表現できるインタフェースを Bramble 側でサポートすることは，CORBA 対応を目指す上で非常に重要となる．IDL で表現可能な全てのインタフェースに対応できなければ，完全な CORBA 対応とはならないからである．

- 通信の粒度を考慮に入れる

  通信を行なう際に，通信する対象の粒度が小さければ，それだけ多く通信を行なう必要がある．そこで，できるだけ一度にまとめて通信を行なえば，通信の頻度を下げることが可能となる．

- 動的起動をサポートする

  動的起動を用いることで，サーバ側のオブジェクトへの要求を動的に記述することができる．

## 3.2 Bramble/CORBA の基本設計

Bramble 側で用意するオブジェクトの機能として，次のものが必要となる．

- オブジェクトリファレンスの獲得
- オブジェクトリファレンスの生成
- メッセージ通信

ここで，オブジェクトリファレンスの獲得と生成は，同一オブジェクトの機能として設計を行なう．

## 3.3 Bramble/CORBA オブジェクトの設計

### 3.3.1 概要

実際に Bramble/CORBA オブジェクトで必要になるオブジェクトは，大きく分けるとオブジェクトリファレンスを生成するオブジェクトと，送られて来たメッセージをパラメタと共に参照先のオブジェクトに送信するオブジェクトの 2 つである．そこで以後，前者を ORB(Object Request Broker) オブジェクト，後者を OR(Object Reference) オブジェクトと呼ぶことにする．これらのオブジェクトを利用することで，図 1に示すように Bramble から CORBA を利用することができる．

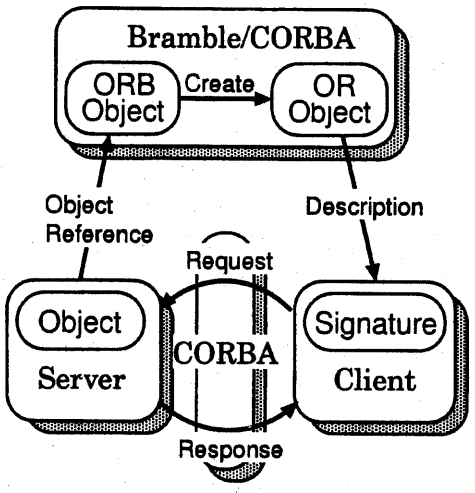


図 1: CORBA の利用

### 3.3.2 ORB オブジェクト

ORB オブジェクトは以下の様な機能を持つ．

- オブジェクトリファレンスから IOR を生成する
- IOR からオブジェクトリファレンスを生成する
- OR オブジェクトを生成する

### 3.3.3 OR オブジェクト

OR オブジェクトは以下の様な機能を持つ．

- 送られて来たメッセージをパラメタと共に参照先のオブジェクトに送信する
- 参照先に送信するパラメタとして IDL で記述可能な型を扱う
- 参照先のオブジェクトに送信するパラメタのためのシグニチャを持つ
- Bramble オブジェクトを参照先のオブジェクトが解釈できる型に置き換える
- 参照先のオブジェクトから送られて来た型を Bramble が解釈できる型に置き換える

# 4 Bramble/CORBA オブジェクトの実装

## 4.1 概要

実装に際して，まず ORB の選択が必要になる．採用の際に，いくつかの ORB について検討した結果，C および C++言語へのマッピングと DII(Dynamic Invocation Interface)，IIOP(Internet Inter-ORB Protocol) への対応は必須であり，マルチプラットフォームを目指す上で移植が容易であると考えられる OmniBroker[7] を採用した．以降の ORB の仕様は OmniBroker に対応させたものである．

## 4.2 ORB オブジェクト

OmniBroker の機能をそのまま利用して，ORB オブジェクトの実装を行なった．以下に ORB オブジェクトとして実装した機能と Bramble/CORBA 上での記述例を示す．なお，設計の段階で検討された OR オブジェクトの生成は，OR オブジェクトを返り値として指定する方法で行なう．

### 4.2.1 オブジェクトリファレンスを IOR に変換する

string_to_object という機能を実装した．この機能を使うことにより，サーバ側から得たオブジェクトリファレンスを，IOR に変換し，String オブジェクトとして扱うことが可能となる．

| 引数 | Object | 特徴 |
|---|---|---|
| 返り値 | String | String オブジェクトを指定する |
| OR | OR | OR オブジェクトを指定する |

例

```
str <- (orb object_to_string(or));
```

### 4.2.2 IOR をオブジェクトリファレンスに戻す

object_to_string という機能を実装した．この機能を使うことにより，IOR からオブジェクトリファレンスを生成できる．

| 引数 | Object | 特徴 |
|---|---|---|
| 返り値 | OR | OR オブジェクトを指定する |
| String | String | String オブジェクトを指定する |

例

```
or <- (orb string_to_object(str));
```

## 4.3 OR オブジェクト

CORBA の仕様に則って C++等で書かれたサーバとの通信を可能とするために，Bramble の OR オブジェクトには型の変換等の機能を実装した．以下に OR オブジェクトとして実装した機能と，Bramble/CORBA 上での記述例を示す．なお，ここでの IDL とは OmniBroker の仕様と，Bramble/CORBA の拡張の方針に則ったものである．

### 4.3.1 オブジェクトに送信するパラメタのためのシグニチャを持たせる

%set_method という機能を実装した．この機能を使うことで，IDL で記述されたインタフェースのオペレーションが Bramble 側で実行可能となる．

| 引数 | Object | 特徴 |
|---|---|---|
| name | String | メソッド名を指定する |
| result | String | 返り値の型を指定する |
| argument | Array | 引数について指定する |

引数についての若干の説明を加える．

- name
  IDL で記述されているインタフェースのオペレーションの名前を指定する．Bramble/CORBA では，まだ IDL からスタブを自動生成する機能が実装されていないため，IDL を直接参照して記述する必要がある．

- result
  IDL で記述されているインタフェースのオペレーションの返り値の型を指定する．IDL を直接参照して，指定されている返り値の型に合わせて，Bramble/CORBA が受け取るオブジェクトのタイプを指定する．現在では，以下のデータ型について実装してある．

| IDL 記述 | result の記述 |
|---|---|
| long | long |
| short | short |
| any | any |
| string | string |
| object | object |
| string sequence | stringsequence |
| any sequence | anysequence |

- argument
  IDLで記述されているインタフェースのオペレーションの引数の型を指定する．IDLの記述では，引数の型の他にin，inout，outという引数の特性を表すものがある．これを踏まえて，以下の様な記述をサポートした．

  ```
  argument:[ name1:"タイプ1:タイプ2"
             name2:"…" … ]
  ```

  ここで，'['と']'に囲まれた':'によって区切られる文字列と値の列挙は，Brambleに於けるArrayオブジェクトを意味する．IDLでの記述を実現するためにオブジェクトのメソッドの持つ引数の数にあわせて，引数の名前とタイプを指定する．nameにはそれぞれ任意の名前を指定する．タイプ1には，i，n，oのいずれかを指定する．これらはそれぞれ引数の特性であるin，inout，outを表わす．タイプ2には引数の型をresultの指定と同様の規則に従って指定する．

以下に，IDLの記述とそれに対応するBrambleの記述例を示す．IDLによる記述

```
//IDL
interface Sample
{
    string method(in string arg1,
                  inout short arg2);
}
```

Brambleによる記述

```
#Bramble
or %set_method name:"method"
    result:"string"
    argument:[ arg1:"i:string"
               arg2:"n:short" ];
```

%set_methodによって用意されたメソッドは，指定された名前と引数で，IDLにより記述されたインタフェースのオペレーションを実現できる．以下に，上記の%set_methodの例で用意されたメソッドを実行する場合の記述例を示す．この例の最後の行では，inout属性であるarg2の値を格納しているargから，arg2の値を取り出し出力している．

例

```
result <- ( or method
    ( arg <- [ arg1:"Hello,World!"
        arg2:1234 ]
    )
);
(arg get "arg2")println;
```

# 5 応用例

Bramble/CORBAの機能を確かめることと，応用例を示すために実際に簡易アプリケーションを作成する．

## 5.1 スケジューラ

実際にBramble/CORBAの機能を利用して「スケジューラ」を作成する．サーバはC++言語によって作成し，クライアントをBramble/CORBAを用いて記述する．

### 5.1.1 スケジューラの機能

スケジューラの機能として以下のものを考え，これらの機能を用意した．

- 今日の日付を表示する
- 現在の時間を表示する
- 今日/指定日の予定を表示する
- 指定時間の予定を表示する
- 今日/指定日の予定に新しい予定を追加する

ここでは，これらのサーバの機能は既に作成されているものとして，クライアントの作成を行なう．

### 5.1.2 Bramble/CORBAによるクライアントの作成

スケジューラのサーバに対して，メッセージを送信するクライアントを実際にBramble/CORBAによって記述する．

まず，オブジェクトリファレンスを得るために，ORB オブジェクトを生成する．ここでは，orb という名前で ORB オブジェクトを生成している．

```
orb <- (system template "ORB");
```

次に，サーバから獲得した IOR から ORB オブジェクトによって，オブジェクトのオブジェクトリファレンスを生成し，それから Bramble の OR オブジェクトを生成する．

```
or <- (orb string_to_object(Scheduler));
```

ここまでの記述でサーバからオブジェクトのオブジェクトリファレンスを得ることができる．ここからは,「指定日の予定を表示する」クライアントの記述例を示す．

指定日の予定を表示するためのメッセージ名等，サーバ側に送る情報からシグニチャを作成する．実際の記述は以下の様になる．

```
or %set_method name:"aday"
    result:"stringsequence"
    argument:[plan:"n:string"
        year:"i:short"
        month:"i:short"
        day:"i:short" ];
```

これを実行してその出力を表示するための記述は以下の様になる．

```
ret <- (or aday
    (res <- [ plan:"myplan"
        year:1998
        month:3
        day:10 ]
    )
);
ret foreach do:{var print};
```

以下に得られる出力例を示す．

```
% bramble aday.b4
 16:00 ソフトウェア工学研究会 (芝浦)
```

## 5.2 リポジトリシステム

Bramble/CORBA の機能を使って，ソフトウェアリポジトリで管理されているモデルをユーザが取り出し，自由に編集できるリポジトリシステム (図 2)[6] を構築する．

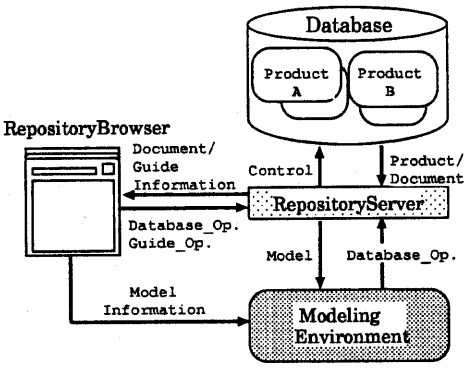


図 2: リポジトリシステム概観

### 5.2.1 リポジトリシステムの構成要素

リポジトリシステムの主な構成要素について，簡単に説明する．

- データベース
  データベースは，モデリング環境で構成されたモデルを格納するためのリポジトリとして利用される．データベースに格納されたプロダクトは永続性が保証され，リポジトリサーバにより管理される．データベースには，オブジェクト指向データベースである MATISSE [8] を利用する．

- リポジトリサーバ
  リポジトリサーバは，リポジトリに対するユーザの要求に応えるサーバである．リポジトリサーバがユーザに提供するサービスには，ソフトウェア開発におけるプロダクトとプロセスを対象としたサービスがある．ユーザは，リポジトリブラウザもしくはモデリング環境上で起動されているツールからリポジトリサーバに要求を送ることができる．リポジトリサーバは C++によって記述されている．また，リポジトリサーバの詳細については後述する．

- リポジトリブラウザ
  ユーザは，リポジトリブラウザを使ってリポジトリ内の成果物やツールを視覚的にブラウズすることができる．そしてブラウザからツールを起動したり成果物を取り出したりすることができる．リポジトリブラウザはBrambleによって記述される．

- モデリング環境
  リポジトリ内のツールの起動，モデルの編集は，モデリング環境上で行われる．モデリング環境からもリポジトリサーバに要求を出すことが可能なため，モデリング環境上で実行しているツールからもデータベースの制御を行うことができる．モデリング環境はBrambleによって記述される．

#### 5.2.2 リポジトリサーバ

リポジトリサーバは，RepositoryServer とDatabaseという2つのオブジェクトから構成されている．ユーザはRepositoryServerオブジェクトにデータベース接続の要求を出し，RepositoryServerオブジェクトがデータベースに接続された状態のDatabaseオブジェクトを生成する．以降ユーザはこのDatabaseオブジェクトを用いて，モデルリポジトリとなっているデータベースを操作する．そしてRepositoryServerオブジェクトにデータベース切断の要求を出すと，RepositoryServerオブジェクトは指定されたDatabaseオブジェクトをデータベースから切断し，Databaseオブジェクトを破棄する．

#### 5.2.3 CORBAによる通信の実現

リポジトリシステムを分散ネットワーク上で利用可能とするため，リポジトリサーバをCORBAのサーバ，BrambleをCORBAのクライアントとして実装を行なうことで簡単に実現できた．その手順を以下に示す．

1. ユーザは，まずBramble上でORBオブジェクトに対してリポジトリサーバ上に常に1つだけ存在しているRepositoryServerオブジェクトを参照しているORオブジェクトの生成を要求する．
2. 得たRepositoryServerオブジェクトに対するORオブジェクトにDatabase接続のリクエストを送る．
3. リクエストの返り値としてRepositoryServerオブジェクトが生成したDatabaseオブジェクトを参照したORオブジェクトを受け取る．

これでデータベースへの接続が完了する．以降のモデルの取り出し等のデータベースに対するリクエストは，Databaseオブジェクトに対するORオブジェクトに対して送ることで作業を進められる．

以上の様な仕組みによって，ユーザはリポジトリサーバ上のオブジェクトがあたかもBramble上に存在するようにメッセージ送信することができる．

## 6 おわりに

Bramble を CORBA に対応させることで，CORBAをサポートしたオブジェクトとの通信が可能になった．このことにより，グループ開発支援環境における開発者間でのコミュニケーションや，データベースの操作，オブジェクト間におけるメッセージ通信等の実現を容易に行なうことができる．

今後，Bramble の CORBA 対応については，Brambleオブジェクトを CORBA オブジェクトとして扱うCORBAサーバの実現が課題となる．

## 参考文献


[1] http://ring.etl.go.jp/openlab/horb/
[2] http://www.microsoft.com/cominfo/
[3] 小野沢博文: 分散オブジェクト指向技術CORBA, ソフト・リサーチ・センター, 1996
[4] 成田雅彦, 保西義孝, 勝亦幸善, 島村政義, 島村真己子, 杉能康明: CORBA と Java 分散オブジェクト技術, ソフト・リサーチ・センター, 1997
[5] 辻新太郎, 齊藤裕介, 上田賀一: ツール部品によるグループ開発支援環境構築基盤の構築, 情報処理学会, 研究報告 (SE), Vol.97, No.25, pp.25-32 (1997)
[6] 小飼敬, 中野喜之, 上田賀一: メタ階層アーキテクチャによるモデリングとリポジトリシステム, 情報処理学会, 研究報告 (SE), Vol.97, No.74, pp.9-16 (1997)
[7] http://www.ooc.com/ob
[8] MATISSE 2.3 TUTORIAL 1st Issue, A.D.B. S.A., 1995