分散仮想オフィスのための通信サービス群の検討

3R－3

満田成紀　中村達也　鰺坂恒夫　Kazimierz Subieta　上林彌彦

京都大学工学部

1　はじめに

分散システムにおいて、プロセスやコンポーネントなどの要素間における通信を行うサービスを異種プラットフォーム(ハードウェアアーキテクチャや言語環境が異なる)上で実現するには、それらの上で動くコンポーネント間での相互動作を保証するための共通の規格が必要である。実際にそのようなサービスとして設計・実装されているものの中から以下の4つのサービスについて 抽象サービスの記述能力の観点から比較・検討する。

i) CORBA (Common Object Request Broker Architecture):
Object Management Group 提唱
ii) SoftwareBus: *Eureka Software Factory* 提唱
iii) BMS (Broadcast Message Server):
HEWLETT PACKARD 社の CASE 環境である HP Softbench 上で運用
iv) OLE (Object Linking and Embedding):
Microsoft 社の *Windows* 環境で運用

2　ツール間通信

通信サービス群は、あるコンポーネント(クライアント)からのサービス要求を受けつけ、それを提供するコンポーネント(サーバ)と結合する(図1)。ここでいうサービスとはデータに対して作用する操作の集合である。この場合サーバとなるコンポーネントのインタフェースを何らかの形で記述し、定義しておく必要がある。

3　通信サービス群の概要

3.1　BMS

BMS は、HP Softbench 環境内の各種ツールにメッセージを振り分けるメッセージ・ディスパッチャである。HP Softbench ツールは BMS との接続を確立する際に、ツールがサポートしているツールプロトコルとサービスする操作を宣言する。ツールに対する操作は、操作のタイプである *Command-Class* (例 EDIT, DEBUG) と、Command-Class 内部での操作名称である *Command-Name* (例 SAVE-FILE, STEP, STOP など) の組み合わせにより記述する [3]。

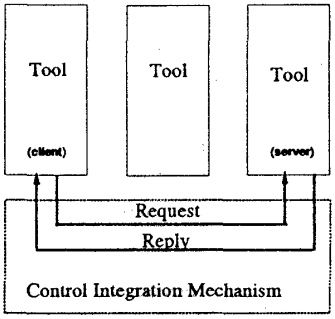

図 1: ツール間通信のモデル

3.2　OLE

OLE は *Windows* 上でのアプリケーション統合のための枠組である。サービスを行うコンポーネントはインタフェースを備えたコンポーネント・オブジェクト・モデル (COM) として定義される。他のサービスと比較した際に特徴的なのは、インタフェースを記述するための言語を持たないことである。インタフェースはオブジェクトに実装されている意味的に関連のある関数の集合である。COM はオブジェクトの実装についてバイナリの標準を記述する [2]。

3.3　SoftwareBus

SoftwareBus はコンポーネントの相互動作を保証するためのミドルウェアである。コンポーネントとサービスの機能の記述方法は別であり、抽象サービスの定義はデータ型とそれに対する操作を、言語 SADL(*Service Abstract Description Language*) により記述する。コンポーネントの定義は、どのサービスを要求 (クライアント) あるいは提供 (サーバ) するのかを記述する。コンポーネント間の結合は必要とするサービスにしたがって行なわれる [4]。

3.4　CORBA

CORBA は分散オブジェクト環境におけるオブジェクト間通信をサポートする標準化された規格として提唱されたものである [1]。インタフェース定義言語 (IDL:*Interface Definition Language*) で定義される機能を提供する実体が CORBA でいうオブジェクトである。

On Communication Services for Distributed Virtual Offices
Naruki MITSUDA, Tatsuya NAKAMURA, Tsuneo AJISAKA, Kazimierz SUBIETA and Yahiko KAMBAYASHI
Faculty of Engineering, Kyoto University.

4 CORBA と SoftwareBus の比較

サービス群の比較を行うにあたり、仮想オフィスを構築するツール間通信を題材として取り上げた。仮想オフィスとはネットワーク上で仮想的な組織を作ることにより、オフィスの仕事を実現するものである。以下データ型の扱い方に着目する。

BMS はメッセージの構造が比較的単純であるため、様々な形態のデータを取り扱うのは比較的難しい。また OLE は、関数のポインタを介しての実装レベルでの記述が要求されるため、抽象サービスのやりとりを行うコンポーネント間の通信の記述には向いていない。一方、SoftwareBus の SADL と CORBA の IDL は、比較的コンポーネント間のサービス記述の抽象度が近い。そこで SoftwareBus と CORBA について、同等の機能を持つインタフェースを用いるコンポーネントを記述して比較した。

ここでは親 (*parent*) と子 (*child*) の二つの階層からなるオブジェクトを生成・ストアできる簡単なモジュールの例を用いる。サービスとして以下のようなものを挙げる。

1. 親又は子オブジェクトの生成
2. 親又は子オブジェクトの名前を得る
3. 親オブジェクトに子オブジェクトを追加する
4. 子オブジェクトの一覧を得る

CORBA の IDL と、SADL を文法的に比較した場合、オペレーションの引数と返り値を、各々型名と識別子で指定する点は同じである。

以下に IDL で上記のサービスを実現するインタフェース (= オブジェクト) を示す。(図 2)

A. 子オブジェクト
B. 親オブジェクト
C. 子オブジェクトを生成し管理するオブジェクト
D. 親オブジェクトを生成し管理するオブジェクト

IDL ではこのようにオブジェクトの役割をモデル化することにより、オペレーションのカテゴリとそれを実装するインタフェースが以下のように決まる。

(オブジェクト管理インタフェースの実装)
- オブジェクトの生成
- オブジェクトへの名前付け

(オブジェクト自身の実装)
- オブジェクトの属性 (名前など) の参照
- オブジェクトの他のオブジェクトへのリンク参照

図 2ではこの定義に従って、1 ～ 4 のオペレーションと A ～ D のインタフェースを対応させている。

SADL ではオブジェクトは複合データ型として定義されており、オブジェクトを格納するストアが実装されている。又オブジェクトのメソッドとしてコンストラクタとオペレーションが使用できる。そこで親と子のオブジェクトを各々別のサービスとして宣言し、その中で各々に関係したオペレーションを別々に定義する。図 2において点線で囲まれている部分が、SADL におけるオブジェクトサービスの定義範囲を表している。

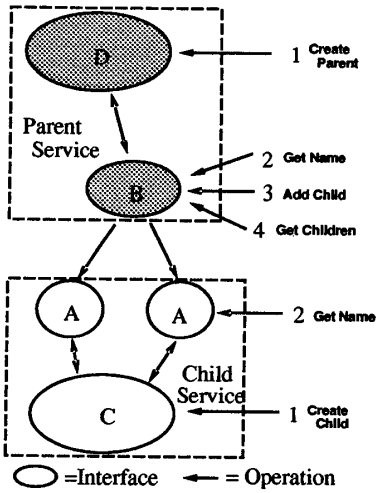

図 2: IDL でのインタフェース定義例

SADL でのサービスは IDL のようにオブジェクト自身の機能とそれを管理するための機能を明確には分けていない。これは他の複合データ型 (有向グラフ、ツリーなど) を扱う場合でも同様である。

IDL での実装の仕方では図 2の A と C の各インタフェースは互いに独立して機能を拡張することができる。(B と D も同様である。) IDL の継承の機能を用いて派生クラスを定義することで、特殊なオブジェクト型あるいは特殊なリンク参照機能などが容易に実現できる。

5 今後の予定

4 種のサービスをとりあげて、その中から記述できるサービスの抽象度が近い SADL と IDL を比較してみた。より柔軟なデータの取り扱いが可能な点で、CORBA の IDL の方が細かい記述が可能だと思われる。今後は仮想オフィスのツール間通信におけるメッセージの、より詳しい意味的カテゴライズを行えるように分析を進める予定である。

参考文献

[1] OMG, "The Common Object Request Broker: Architecture and Specification" Revision 1.1 Draft 10 Dec. 91
[2] Kraig Brockschmidt, "INSIDE OLE2"
[3] HP, "HP Encapsulator Programmer's Guide"
[4] Eureka Software Factory, "SADL Reference Manual"