4Q-3 * CORBA/Java 分散オブジェクト上のデザインパターン/プログラミングテクニックについて

金井 健一　疋田 輝雄
† 明治大学大学院理工学研究科

1 はじめに

多くのサービスがネットワーク上で提供されるようになってきている。サービスはサービスを利用するための情報の入力を利用者に求める。その入力はサービスによって様々である。家電におけるリモコンが一例である。様々な入力を１つの入力端末で行うために、入力端末からサービスの情報を切り離すことを提案し、実際に CORBA/Java で実装した環境 Neserde を紹介する。さらに、CORBA/Java におけるプログラミングテクニックについて論じる。本研究では CORBA3 準拠である VisiBroker for Java 4.1 を用いる。

2 入力端末統合環境 Neserde

Neserde はネットワーク上で提供されるサービスを簡単に開発することを目的に作られたネットワークサービス開発環境 (Network Service Development Environment) である。Neserde には２つの大きな特徴がある。

第一の特徴は、ネットワーク上の複数のサービスに対して、１種類の入力端末で対応可能なことである。あるサービスが変更・追加されても、入力端末は変更・追加する必要はない。入力端末はサービスの情報を持たず、サービスに対する入力用 GUI が必要なときは、各サービスによって動的に GUI が構築される。

第二の特徴は、サービス提供者は、雛型クラスを継承したクラスを作成することにより、簡単にサービスを作成可能なことである。雛型となるクラスは CORBA の初期化・終了処理、分散オブジェクトの生成・廃棄など、サービス提供とは直接関係がない部分を処理する。サービス提供者は自分のサービスに独自の部分だけを記述すればよい。

*Design Patterns/Programming Techniques for Distributed Objects on CORBA/Java
†Meiji University, Tama-ku, Kawasaki, 214-8571 Japan

図 1: Neserde による GUI

3 Neserde の仕組み

一般の GUI の構築と同様に、Neserde による GUI の構築には、GUI コンポーネントの生成、配置、イベント処理の３つの過程がある。

3.1 GUI コンポーネントの生成

サービスはリモートの入力端末上にボタンなどの GUI コンポーネントを生成しなければならない。サービスからの依頼に応じてコンポーネントを生成し、そのオブジェクトリファレンスを返す機能を持つ CORBA オブジェクトを入力端末側に生成しておく。入力端末側に生成されるコンポーネントをリモートコンポーネントと呼ぶ (図 2)。

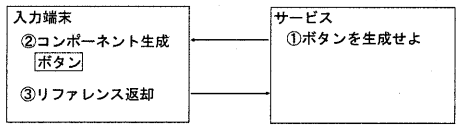

図 2: リモートコンポーネントの生成

3.2 リモートコンポーネントの配置

リモートコンポーネントの１つとして、コンポーネントのコンテナとして機能するリモートパネルを入力端末に用意する。入力端末にはトップパネルと呼ばれるすべてのリモートコンポーネントの最も外側のコンテナとして働くリモートパネルを用意しておく。サービスはトップパネルを取得し、生成したリモートコンポーネントを配置することができる。

3.3 イベント

リモートコンポーネントでイベントが発生したとき、サービス側でそのイベントを受け取る必要がある。そのためにサービス側にリスナオブジェクトを生成し、それをリモートコンポーネントに登録しておく。リモートコンポーネントでイベントが起こるとリスナのメソッドがコールバックされ、イベントが通知される (図 3)。

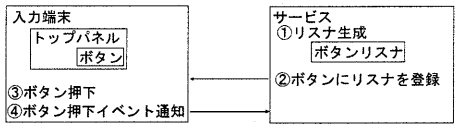

図 3: リスナ登録とイベント処理

入力端末とサービスのオブジェクトは異なるプロセス上にあるが、CORBA を利用することにより、オブジェクト同士はメソッド呼び出しで通信することが可能である。

4 CORBA/Java プログラミングテクニック

CORBA/Java プログラミングに特有のテクニックないし問題点についてまとめる。

4.1 異なるプロセス上での CORBA オブジェクトの生成と廃棄

Neserde ではサービスの要求により、入力端末上に CORBA オブジェクトを動的に生成している。一般にこのような際の問題点は 2 つある。

1) CORBA/Java にはリモートにオブジェクトを生成する標準的な機能がない
2) CORBA オブジェクトの廃棄責任を持つのはどちらのプロセスか

CORBA オブジェクトの生成は、実装側の Java の`new` オペレータによるサーバントの生成、POA の`activate_object()` による活性化の 2 段階で行われる。逆に廃棄は POA の`deactivate_object()` による非活性化と Java ガーベジコレクタにより行われる。CORBA オブジェクトが活性状態にあるときは、サーバントが POA から参照されており、ガーベジコレクタの対象にならない。

問題 1) については、オブジェクトを生成するプロセス上に、サーバントの生成・活性化を行う CORBA オブジェクトを静的に起動しておく方法がある。オペレーションが呼び出されると、CORBA オブジェクトが生成され、その参照が返されるようにしておく。

問題 2) は状況によって解決方法がいくつか考えられるがここでは 2 つの方法をの概略のみ述べる。オブジェクトを実際に生成するプロセスを F (Factory), 生成要求を出すプロセスを C (Client) として、

a) F が生成したオブジェクトを、廃棄依頼を受けたときに一括で廃棄する方法
b) C が F に対して 1 つずつオブジェクトの廃棄を依頼する方法

方法 a) では F が生成したオブジェクトのリストを作成しておき、C の廃棄要求に応じてまとめてオブジェクトを削除する。この方法には、C の処理が単純になるという利点があるが、F のリストが複雑になる可能性や、F にすでに不要になったオブジェクトが残っている期間があるなどの問題がある。

方法 b) では、C が生成したオブジェクトそれぞれに対して、C が必要ないと判断した時点で、F に廃棄要求を出す。F に対して C が複数ある場合でも複雑なリストが必要なく、不要なオブジェクトはすぐに廃棄されるので F 上のメモリを有効に使うことができる。一方、C のプログラマが廃棄するオブジェクトをすべて覚えておかなければならず、C のプログラマの負担は増える。

Neserde では、入力端末では方法 a) を、サービスでは方法 b) を用いている。

4.2 サーバントとオブジェクトリファレンスの区別

サーバントは Java のクラスなので、それを生成したプロセス内では、すべてのメソッドに対してアクセス制限に従ったアクセスが可能である。一方、サーバントから変換されたリファレンスは IDL で定義されたメソッドに対してのみアクセスできる。CORBA2 で曖昧であったこの二者の区別が、CORBA3 で厳密に区別されるようになったことを注意する必要がある。

5 まとめ

ネットワークサービス開発環境 Neserde を紹介し、CORBA/Java による実装の際に現れる考慮点について述べた。

参考文献

[1] E.Gamma, R.Helm, R.Johnson, J.Vlissides, オブジェクト指向における再利用のためのデザインパターン, ソフトバンク, 1995.

[2] 松野良蔵, Java+CORBA 分散オブジェクトシステム構築, 翔泳社, 1999.

[3] A.Vogel, K.Duddy, Java Programming with CORBA, 2nd ed., Wiley, 1998.