分散オブジェクト技術における スタブ・スケルトンの自動生成系の設計と実現

99B608 齋加 梨恵
指導教員　野呂 昌満

1　はじめに

我々の研究室では，分散オブジェクト技術のためのフレームワークをアスペクト指向で実現している．分散オブジェクト技術では位置透過性を満たした遠隔オブジェクト呼び出しができるのがのぞましいという考えをもとに，フレームワークは代理オブジェクトを生成することを前提としている．

本研究の目的は，スタブ・スケルトンの自動生成系を実現することである． 分散オブジェクト技術のためのフレームワークを利用する場合に，スタブスケルトンの自動生成系によって位置透過性を満たすことができるようにする．XML-RPC[2] のような位置透過性を満たすことを考えていない技術も位置透過性を満たした利用が可能になる．

スタブの自動生成系は遠隔オブジェクトのインタフェースを入力として受け取り，インタフェースにアドバイスを織り込み，スタブのコードを生成する．遠隔オブジェクトのインタフェースをコアコンサーン，スタブのコードを織り込まれたコードとみなす．自動生成系にはアドバイスの記述が書かれており，Aspect Weaver の役割も果たしてアドバイスを織り込んでスタブのコードを生成する．

2　アスペクト指向による設計

現在考えられているフレームワークではデータの整列化，非整列化，通信に関するスタブのコードが散在することが考えられるのでアスペクト指向で実現している．

2.1　クライアント側フレームワーク

我々の研究室では分散オブジェクト技術が位置透過性を満たすように，フレームワークを利用する場合には代理オブジェクトを生成することを考えている．現在考えられているフレームワークにおいて，アスペクトで実現されているクライアントのコンサーンを図 1にしめす．

コンサーン	アスペクト	コンサーンの説明
状態遷移機械	STM	オブジェクトの生成,メソッド呼び出し，消滅を状態遷移とする．オブジェクトは待機状態にいるときにメソッド名が呼ばれるとスクリプトを実行する
serverInfo	RemoteRef	遠隔オブジェクトのIDを管理する
marshal	Marshal	メソッドのシグネチャ，実引数の整列化
connection	Transport	データを送受信する
unmarshal	Marshal	メソッドの返り値を非整列化する
multicast	Runtime	複数のリモートオブジェクトに通信する
security	Runtime	通信先に対する認証や通信データの暗号化

図 1: クライアントのコンサーン

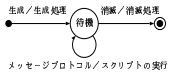

図 2: スタブの状態遷移機械

図 3に状態遷移機械をコアコンサーンとした，クライアントのアスペクト間の関係をしめす．図 3の網かけの部分である状態遷移機械と，アスペクト間のデータの受け渡し部分がスタブのコードに書かれる．

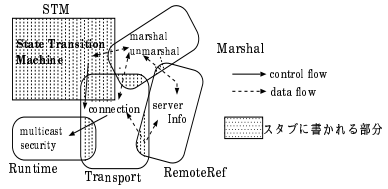

図 3: クライアントのアスペクトの関係

2.2　スタブの自動生成系

スタブの自動生成系では遠隔オブジェクトのインタフェースを状態遷移機械と考え，アスペクト間のデータの受け渡しを織り込み，スタブのコードを生成する．スタブの自動生成系にはアドバイスの記述が書かれている．一般のアスペクト指向では Aspect Weaver とアスペクトを実現するコードは分離するが，自動生成系ではアスペクトを実現するコードが同じなので両方の役割を自動生成系が果たしている．図 4, 図 5に遠隔オブジェクトのインタフェースの例と自動生成されたスタブのコードをしめす．

```
public interface Sample {
  int sum(int i, int j) throws OrbException;
}
```

図 4: 遠隔オブジェクトのインタフェースの例

```
public class SampleStub implements Sample {
 ServerInfo info;

 public SampleStub(String url, String objname) {
  info = new XmlrpcServerInfo(url, objname);
 }

 public int sum(int param_0, int param_1) throws OrbException {
  ClientMarshalStream ms = new XmlrpcClientMarshalStream();
  ms.open();
  //MarshalとRemoteRefの合流点
  ms.writeClassName(info.getObjectName());
  ms.writeMethodName("sum");
  ms.writeInt(param_0);
  ms.writeInt(param_1);
  ms.close();                                          ← STMとMarshal(marshal)の合流点

  //TransportとRemoteRefの合流点
  Connection con = new HttpConnection(info);
  con.connect();
  con.sendMessage(ms.getBytes());
  con.recieveMessage();
  con.disconnect();                                    ← STMとTransportの合流点

  //MarshalとTransportの合流点
  ClientUnmarshalStream ums = new
    XmlrpcClientUnmarshalStream(con.getResponse());
  ums.open();
  int result = ums.readInt();
  ums.close();                                         ← STMとMarshal(unmarshal)の合流点
  return result;
 }
}
```

図 5: 自動生成されるスタブのコード

3　スタブの自動生成系の設計・実現

スタブの自動生成系を次のような設計指針に基づいて設計する．

1. プログラミング言語および分散オブジェクト技術に依存する部分と独立する部分を分離する
2. プログラムの構造が整理しやすいオブジェクト指向で設計・実現し，オブジェクト指向において再利用がしやすい設計方法として知られているデザインパターンを用いる．

スタブの自動生成系は遠隔オブジェクトのインタフェースを受け取り，遠隔オブジェクトの情報を取得する．取得する方法としては字句解析・構文解析を用いる方法や，Java の Reflection を用いる方法が考えられる．今回は自動生成系もスタブのコードも Java で実現するので，既に Java で提供されている Reflection の機能を利用した．

設計指針 1 の言語や技術に依存する部分と独立する部分を分離することについて考える．独立した部分はスタブの構造，依存する部分はスタブのコードを生成する部分が考えられる．生成するスタブの構造は図 6 のようになる．

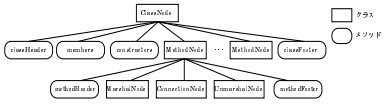

図 6: スタブの構造

スタブの自動生成系では図 6の構造のコンサーンにあたる木の節点をクラスで実現し，その他をメソッドで実現した．図 6の MethodNode は状態遷移機械コンサーンにあたる．図 6の木構造からスタブのコードを生成するような自動生成系を設計する．設計したスタブの自動生成系のクラス図を図 7にしめす．

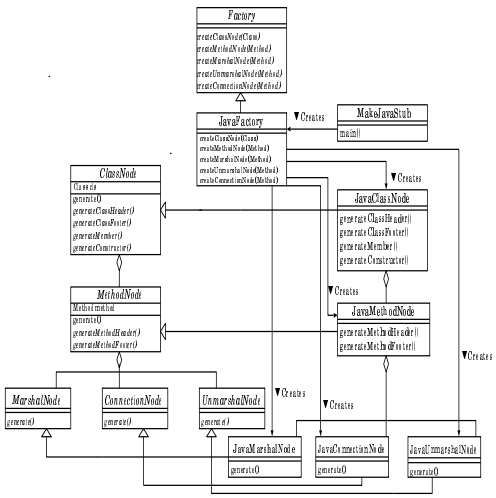

図 7: スタブの自動生成系のクラス図

図 7のクラス図の左側の部分が図 6の木構造をあらわしていて，それらのサブクラスの生成部分で Abstract Factory パターンを用いている．図の右側の部分がサブクラスにあたり，スタブのコードを生成する部分である．これらは言語・技術に依存する部分でありサブクラスの追加が必要になるので Abstract Factory パターンを利用した．

スタブのコードを生成する各クラス内の generate メソッドで Template Method パターンを用いている．コンサーンにあたる節点のオブジェクトに対して generate メソッドを呼び出し図 5のスタブのコードを生成する．例えば，ClassNode クラスでは図 5のクラス名，メンバ変数，コンストラクタ部分を生成し，JavaMarshalNode クラスでは STM と Marshal(marshal) の合流点部分を生成する．

4 考察

アスペクト指向を実現する方法は，Aspect Weaver を使っての生成方式と，ライブラリとして提供する方式が考えられる．分散オブジェクト技術が位置透過性を満たすためにはスタブ・スケルトンが必要であり，これらは一般に自動生成される．自動生成系も Aspect Weaver もコードを生成することから，自動生成系をアドバイスの生成と Aspect Weaver の両方の役割をするとみなし実現する方法を選んだ．

自動生成系の実現において，図 6のような木構造をたどる走査は Visitor パターンを使っても実現できる．Visitor パターンでは Visitor クラスのなかで木の節点に対する処理を記述するので再利用の単位が Visitor クラスになる．しかし，スタブの自動生成系では XML-RPC で使った HTTP のための Connection を SOAP の場合でも再利用したいのでコンサーンごとにクラスを作り再利用の単位としたい．このことから Abstract Factory パターンと Template Method パターンを使った実現を選んだ．

今回実現はしていないが Runtime 部分は，connection をコアコンサーン，Runtime をアスペクトとして考えることで実現できると考えている．　その際に Decorator パターンやアスペクトモデレータフレームワークを使って connection から Runtime を呼ぶことができると考えている．現状の XML-RPC では認証等については考えられていないが，これらの機能を追加することも可能だと考えている．

5 おわりに

アスペクト指向に基づいたフレームワークを実現するようなスタブの自動生成系を実現することができた．アプリケーションの開発者は位置透過性を満たした分散オブジェクト技術を利用してアプリケーションを開発することができると考えられる．他の言語や他の分散オブジェクト技術での実現が今後の課題である．

謝辞

本研究を進めるにあたり，一年半御指導していただきました野呂昌満教授，蜂巣吉成先生，有益なアドバイスをいただいた張漢明先生，大学院生の熊崎敦司さん，朝長武士さん，山本英貴さん，藤原泰昌さん，森貴彦さんに深く感謝します．

参 考 文 献

[1] 野呂 昌満，「プログラミング言語から見たアスペクト指向開発技術」，ソフトウェアシンポジウム，2002

[2] XML-RPC Home Page, http://www.xmlrpc.com/.

[3] E.Gamma 他，「Design Patterns」，ADDISON-WESLEY，1994