分散オブジェクト技術における クライアント側フレームワークの設計と実現

99B644 渡辺 智之

指導教員　野呂 昌満

1　はじめに

分散オブジェクト技術とは複数のコンピュータ上でオブジェクト同士がネットワークを介して，メッセージを送受信する環境を実現するための技術である．これらを実現する技術に，CORBA(Common Object Request Broker Architecture)[2]，SOAP(Simple Object Access Protocol)[3], XML-RPC[4] 等がある．分散オブジェクト環境の実現には，メッセージの交換形式の定義，メッセージを整列化する形式の定義をおこなう必要があり，位置透過性を満たしていることが望ましいとされている．

分散オブジェクト技術の構造は整理されていない．これにより実現が技術ごとに異なり，各ミドルウェアの開発者の経験に依存している．位置透過性が考慮されておらず，アプリケーションの開発者の負担が大きい技術も存在する．

本研究の目的は分散オブジェクト技術の構造を整理することである．整理した構造に基づいて，フレームワークの実現をおこなう．フレームワークによって，各分散オブジェクト技術の開発方法と，実現された分散オブジェクト技術の利用方法を統一する．代理オブジェクトの自動生成を前提とすることで，位置透過性を満たした利用を可能にする．本稿ではクライアント側フレームワークを実現する．

2　分散オブジェクト技術の構造の整理

分散オブジェクト技術の構造は階層化が考えられるが，データの受け渡しの際に構造が乱れるという問題がある．CORBA 等の仕様を調べると，図 1 のような階層構造が考えられる．この階層間のデータの受け渡しを考えると，意図しないデータの受け渡しが生じ，階層が乱れる．階層が乱れる例として，ユーザ層から遠隔オブジェクトに渡す引数を整列化するために，引数の情報を整列化層に渡す場合が考えられる．整列化層へのデータの受け渡しは，引数等の情報を必要としないリモート参照層を介しておこなう必要があり，リモート参照層での意図しないデータの受け渡しが生じる．

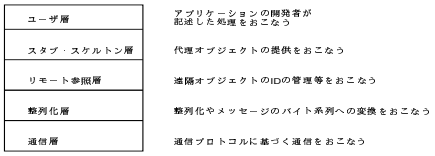

図 1: 構造を整理した階層

アスペクト指向技術 [1] に基づいて構造を整理することで，階層構造を考えた場合に生じていた構造の乱れを解決することができる．分散オブジェクト技術の各構成要素をアスペクトとして考えることで，ある構成要素に関して散在するコードや，アスペクト間のデータの受け渡しの処理等を整理できる．

階層の各層としていた構成要素をアスペクトとして考え，アスペクト指向技術に基づく整理をおこなう．クライアント側のコンサーンを図 2 のように考えた．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側のアスペクト間の関係をしめす．アスペクトの中の marshal 等はコンサーンである．

状態遷移機械	イベントとしてメソッド名を受け取り アクションとしてメソッドを実行する
serverInfo	遠隔オブジェクトのIDの管理をおこなう
marshal	メソッド名や引数の情報の整列化をおこなう
unmarshal	返り値の非整列化をおこなう
connection	HTTP等の通信プロトコルにしたがって通信をおこなう

図 2: クライアント側のコンサーン

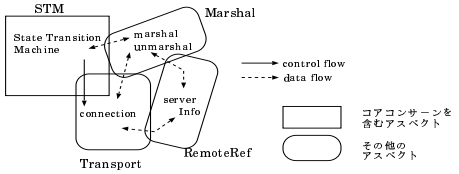

図 3: クライアント側のアスペクト間の関係

アスペクト間のデータの受け渡しは，アドバイスとして記述し，代理オブジェクトを介しておこなう．既存の分散オブジェクト技術には代理オブジェクトを生成する自動生成系を用いる技術が存在する．自動生成系を Aspect Weaver の役割を果たすものとして考え，データの受け渡しの処理を代理オブジェクトに織り込む．

3　フレームワークの設計と実現

整理した構造に基づくフレームワークの設計と実現をおこなう．

3.1　設計

分散オブジェクト技術に共通する部分を定義した抽象クラスをフローズンスポット，フレームワークの利用者が各技術に依存する部分を定義するサブクラスをホットスポットとする．例としてフレームワークの整列化におけるクラス図を図 4 にしめす．クライアント側とサーバ側での共通の処理があった場合，両側で共通して利用する抽象クラスを定義する．図 4 では整列化でのサーバ側とクライアント側の共通の処理を MarshalStream で定義した．クライアント側のみでの処理を加えた MarshalStream のサブクラスを，抽象クラス ClientMarshalStream として定義した．MarshalStream，ClientMarshalStream がクライアント側フレームワークのフローズンスポットとなる．フローズンスポットとして定義した抽象クラスの技術ごとのサブクラスがホットスポットとな

る．図 4 では MarshalStream のサブクラスである XmlrpcMarshalStream と，ClientMarshalStream のサブクラスである XmlrpcClientMarshalStream がクライアント側フレームワークのホットスポットとなる．

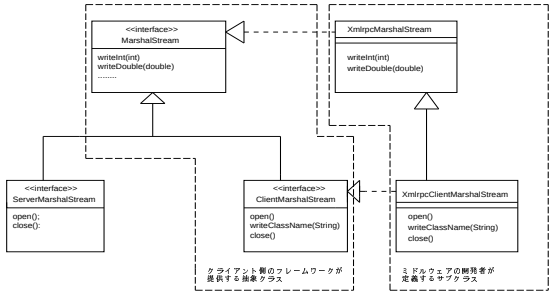

図 4: 整列化におけるクラス図

3.2 実現

実現は Java でおこない，ホットスポットとして XML-RPC のためのサブクラスを定義することで XML-RPC を実現した．XML-RPC は利用できる型が少なく単純な仕様なので，動作の確認に適している．

4 考察

フレームワークをミドルウェアの開発者とアプリケーションの開発者の視点から考察する．ミドルウェアの開発者は分散オブジェクト技術の規格に基づいて分散オブジェクト環境を実現し，アプリケーションの開発者は分散オブジェクト技術を利用したアプリケーションを実現する．

4.1 ミドルウェアの開発者の視点からの考察

開発方法が整理されていなかった分散オブジェクト技術の実現にフレームワークを利用することで，開発効率が向上する．例として整列化の部分の開発を考える．抽象クラス MarshalStream では，整数型の整列化をおこなうメソッドである writeInt 等が抽象メソッドとして定義されている． 開発する部分が明確化し，技術ごとのサブクラスで writeInt 等の実際の処理を記述することで，各技術の整列化部分を開発できる．

アスペクトを構成するオブジェクト間のデータの受け渡しを考えずに，コンサーンに関する処理のみを定義することで開発することができる． データの受け渡しは代理オブジェクトを介しておこなうので，データの受け渡しの処理は必要なくなる．

4.2 アプリケーションの開発者の視点からの考察

フレームワークを利用して実現した XML-RPC を利用した場合と， 既存の XML-RPC の実現である Apache XML-RPC[5] を利用した場合とを比較し，考察をおこなう．

フレームワークと自動生成系によって， Apache XML-RPC では考慮されていない位置透過性を満たした利用ができる．フレームワークは自動生成される代理オブジェクトの利用を前提としており，アプリケーションの開発者の負担を軽減できる．

フレームワークによって，遠隔オブジェクトの呼び出し方法を統一することができる．例としてオブジェクトを引数として扱う場合を考える． 図 5 にそれぞれの整列化をおこなうときの記述をしめす．XML-RPC の仕様ではオブジェクトを扱った呼び出しが規定されていない．Apache XML-RPC でオブジェクトを扱う場合には，アプリケーションの開発者自身がオブジェクトをシリアライズし，バイト配列として送る処理を記述する必要がある．フレームワークを利用した XML-RPC では，オブジェクトの利用が規定されている CORBA 等のように，シリアライズ等の処理をおこなわずにオブジェクトを扱うことができる．オブジェクトのシリアライズ等の処理はスタブが利用する整列化をおこなうオブジェクトでおこなう． これにより，XML-RPC の仕様で定義されていなかったオブジェクトを扱うことができる．他の技術でも同じ方法でオブジェクトを扱うことができるので，アプリケーションの開発者は統一されたインターフェイスを利用することができる．

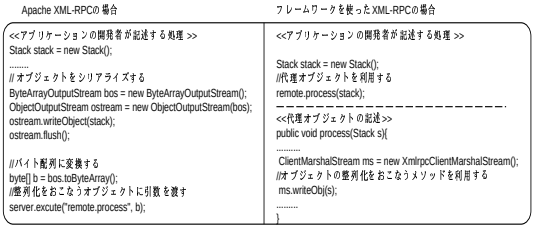

図 5: 整列化をおこなうコード

5 おわりに

本研究では各分散オブジェクト技術の構造の整理をおこなった．これに基づきクライアント側フレームワークを提案し，開発方法の統一を考えた．フレームワークを利用して XML-RPC を実現し，動作の確認をおこなった．

より複雑な仕様である SOAP 等をフレームワークを利用して実現し，フレームワークの妥当性を検証していくことが今後の課題である．

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] Common Object Request Broker Architecture: Core Specification, Object Management Group, 2002

[3] Simple Object Access Protocol (SOAP), http://www.w3.org/TR/SOAP/.

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

[5] XML-RPC - Apache XML-RPC, http://xml.apache.org/xmlrpc/.