リフレクションを利用した CORBA アプリケーション開発環境に関する研究

藤枝 和宏
北陸先端科学技術大学院大学

2000 年 3 月

論文の内容の要旨

CORBA は分散オブジェクトを実現するミドルウェアである ORB (Object Request Broker) の標準規格である。CORBA の特徴は、特定のプラットフォームやプログラミング言語に依存しないことにある。組み込み機器用の OS からメインフレームまでさまざまなプラットフォームを対象とした実装が存在し、分散オブジェクトとそのクライアントは、オブジェクト指向でない言語を含め、さまざまなプログラミング言語で記述できる。分散オブジェクトの実装に用いられた言語やプラットフォームの違いは ORB によって隠蔽されるので、分散オブジェクトの操作はこれらの違いを意識することなく行える。

CORBA のアプリケーションを開発する際には、IDL (Interface Description Language) で記述した分散オブジェクトの仕様から、IDL コンパイラを用いて生成したスタブとスケルトンと呼ばれるプログラムをアプリケーションに取り込む必要がある。そのため開発手順が煩雑であり、サーバとクライアントで用いる言語や開発プラットフォームが異なる場合には、分散オブジェクトの仕様を正確に共有できずにインタフェースの不一致を起こすこともある。本論文ではこれらの問題を、IDL で記述した仕様を格納するインタフェースリポジトリと、リフレクションをサポートするプログラミング言語を利用して解決する手法を示す。

インタフェースリポジトリは IDL で記述された定義を格納できる CORBA のサーバである。CORBA のプラットフォーム非依存性により、インタフェースリポジトリは異なる開発プラットフォームで容易に共有できる。インタフェースの不一致の問題は、インタフェースリポジトリを中心とする開発環境により解決される。この開発環境は、内容の更新日時を記録可能にしたインタフェースリポジトリ、その内容を編集するツール、および既存の IDL コンパイラを利用してインタフェースリポジトリに格納された仕様とスタブ/スケルトンの間の一貫性を保つツールからなる。この環境は特定のプログラミング言語や開発プラットフォームに依存せず、CORBA のアプリケーション開発が可能な、ほとんどすべてのプログラミング言語と開発プラットフォームの組み合わせで利用できる。

リフレクションをサポートするプログラミング言語をアプリケーション開発に利用する場合には、インタフェースリポジトリに格納された定義を元に、リフレクションを利用して必要なスタブ/スケルトンを実行時に生成して取り込むことで、開発手順の煩雑さを大幅に軽減できる。リフレクションによって得られる能力は、プログラム自身を参照/操作するプログラムを記述可能にする Linguistic リフレクションと、プログラムの実行環境 (処理系) を参照/操作するプログラムを記述可能にする Behavioral リフレクションに大別される。Linguistic リフレクションによってスタブ/

スケルトンの実行時生成が可能になり、Behavioral リフレクションによってアプリケーションに必要なスタブ/スケルトンを開発者による特別な指示なしで特定して取り込むことが可能になる。

Linguistic リフレクションを用いて、インタフェースリポジトリから直接スタブ/スケルトンを生成することで、IDL ファイルやプログラミング言語の構文上の制約から離れて、アプリケーションに必要な定義のみを取り込むことが可能になる。また、分散オブジェクトの仕様が更新されたときに、すでに取り込まれている定義の一部を書きかえることも可能になる。

アプリケーションに必要なスタブ/スケルトンの特定とアプリケーションへの組み込みは、Behavioral リフレクションを利用することで、特別なプログラミングや開発手順なしで実現できる。これを実現する方法は二つある。アプリケーションの実行時の振る舞いを利用する方法と、プログラムを実行する処理系の振る舞いを利用する方法である。

前者は、クライアントの取得した分散オブジェクトのオブジェクトリファレンスを利用することで実現できる。CORBA の実行時ライブラリにおいて、クライアントの取得したオブジェクトリファレンスを用いてインタフェースを特定する標準のオペレーションを実行することで、クライアントに必要なスタブを特定することができる。実行時ライブラリで生成したスタブは、Behavioral リフレクションによりアプリケーションの名前空間を操作することで、アプリケーションに組み込むことができる。

後者は、プログラミング言語の処理系によって検出された未定義名への参照を利用することで実現できる。Behavioral リフレクションにより未定義の名前を参照したときの処理系の振る舞いを変更し、その名前に対応するスタブやスケルトンを生成して参照の結果として返すことで、必要なスタブ/スケルトンをアプリケーションに提供することができる。特に未定義のメソッドを参照したときの振る舞いの変更により、CORBA Messaging などの API を利用するために必要な特別なスタブを、それが必要なときにだけ生成して提供できる。

リフレクションをサポートしており、対応する CORBA の実装が存在する Python と Java について、上記のアプローチの実装を行った。Python は比較的広範囲にわたってリフレクションを提供しており、上記のアプローチをすべて実装できる。アプリケーションに必要なスタブ/スケルトンを特定する方法としては、オブジェクトリファレンスを利用する方法と、未定義のモジュール名とメソッド名の参照時の振る舞いを変更する方法が利用できる。Java の提供するリフレクションはかなり範囲が狭く、スタブ/スケルトンの取り込みはクラス単位で行うことしかできず、一度取り込んだスタブクラスを変更することもできない。必要なスタブを特定する方法としては、未定義のクラス名を参照したときの振る舞いを変更する方法だけが利用できる。

どちらの言語においても、アプリケーションに必要なスタブ/スケルトンは実行時に自動的に生成されて組み込まれるので、開発手順は大幅に簡略化される。これらの言語は CORBA のクライアントの実装に利用されることが多い。分散オブジェクトがリフレクションの利用できないプログラミング言語で実装される場合でも、前述したインタフェースリポジトリ中心の開発環境を利用することで、インタフェースの不一致の発生を軽減できる。アプリケーションのプロトタイピングの段階では、分散オブジェクトの定義が変更されることが多いので、この環境を利用することで開発効率を大幅に向上させることができる。