オブジェクト指向分散環境 OZ++ のクラス管理方式

5B−9

吉屋 英二(富士ゼロックス情報システム*) 大西 雅夫(東洋情報システム*)
濱崎 陽一(電子技術総合研究所) 塚本 享治(電子技術総合研究所)
* 情報処理振興事業協会「開放型基盤ソフトウェア研究開発評価事業」研究員

1 はじめに

OZ++は、オブジェクトの交換と共有に基づく、オブジェクト指向分散処理環境である。基本的なオブジェクトモデル、および実行意味論は、電総研で開発された分散システム OZ+[1]を基本とし、これの実行性能を向上させ機能の洗練をして、PDS として実用に供する予定である[2]。

現在、OZ++はその基本設計を終えて、設計の詳細化を進めているところである。ここでは、オブジェクトの動作を規定するクラスを管理するための方式について述べる。

2 概要

クラスの管理情報としては、クラスの名称空間、クラスの継承関係、クラスの属性、インスタンス変数、インスタンスメソッド、などがある。また、クラスのバージョンの管理も名称空間の管理の一部として行なわれる。必要に応じてオブジェクトの実行環境がメソッドの実行コードのローディングを行なえるように、実行コードを管理し、これを実行環境に動的に提供する機能も提供する。

3 クラスのバージョン

クラスの利用者にとって、利用しているクラスが変更された場合、再コンパイルが必要かどうかは明確ではない。OZ++ではクラスの利用形態を、メソッドを利用する場合と継承する場合の二種類に分けて考え、それぞれについて再コンパイルの必要性を判別できるようにする。(ただし、OZ++では古いバージョンは削除しないので、新しいバージョンのクラスを利用したい時のみ再コンパイルをする必要がある。)

メソッドを利用する場合には、メソッドのシグネチャに変更があれば、また継承する場合には、メソッドのシグネチャ又はインスタンス変数に変更があれば再コンパイルする必要がある。これらの情報をそれぞれクラスの Definition 、Private Definition と呼ぶ。またクラスメソッドの実装に関する情報をクラスの Implementaion と呼ぶ。そしてこれら三つの情報にそれぞれ版を持たせる。

定義から明らかなように、Implementation は Private Definition に依存し、Private Definition は Definition に依存している。従ってクラスの版は図 1 に示すような木構造となる。

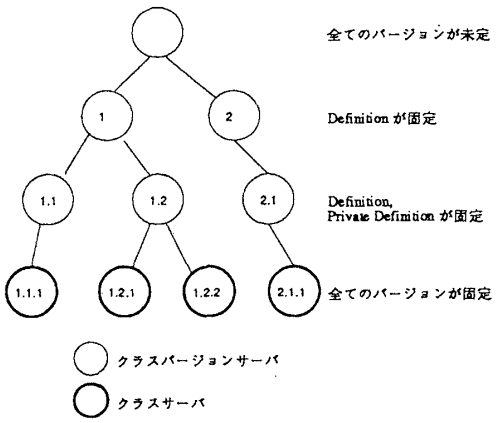

図 1: クラスバージョンの木

4 クラスサーバ

クラスサーバとは、コンパイル時にソースコードが参照しているクラスの情報を提供したり、実行時にクラスの実行コードを提供するオブジェクトである。図 1 において葉に対応するオブジェクトがクラスサーバである。節に対応するオブジェクトはクラスバージョンサーバといい、自分の子の中の一つを公開バージョンとして持っている。コンパイル時や実行時には、クラスバージョンサーバにクラスの情報を問い合わせるので、公

Class Management Mechanism in OZ++: Object-Oriented Distributed Systems Environment
Eiji Yoshiya (Fuji Xerox Information Systems, Co., Ltd.*), Masao Oonishi (Toyo Information Systems, Co., Ltd.*), Yoichi Hamazaki (Electrotechnical Laboratory), Michiharu Tsukamoto (Electrotechnical Laboratory)
*: Researcher of Research, Development and Evaluation of Fundamental Software Technology in Information-Technology Promotion Agency, Japan

開バージョンの変更を行うことにより、動的にクラスのバージョンを決定することができる。

5 スクールサーバ

小規模の環境と異なり、(特にマルチユーザの) 分散環境では、クラス名が容易に衝突してしまうため、これを大域的にユニークなものとして扱うことは難しい。OZ++では、この問題を解決するため、クラス名と実際のクラスとの対応表 (スクールと言う) を環境中に維持する。この役を行なうオブジェクトをスクールサーバという。OZ++では、クラス名の指定に際してスクールを指定することができ、出所の異なるライブラリを同時に利用してもクラス名の衝突が起きないようにすることができる。

6 スクールのモジュール化

前項で述べたスクールは、クラス名に対応してクラスの特定のバージョンを指定するのにも用いることができる。この機能を用いて、開発中のソフトウェアを構成するクラスのうち、安定している部分を選択的に利用することが可能である。

クラス名に対応するクラスがすべて特定のバージョンに固定されているようなスクールをモジュール化されたスクールと言う。ソフトウェアはモジュール化されたスクールを公開することでリリースできる。

クラスの実行可能コードは通常、利用するクラスのバージョンを動的に決定するようになっている。これに対してモジュール化されたスクールに含まれるバージョンは、他のクラスの特定の版を利用することが分かっているので、静的なバージョン決めによる最適化が可能である。

7 コンフィグレーション

これまで述べてきたように、クラスの名称は「スクール名」+「クラス名」+「バージョン」で一意に決定できる。

しかし、ソースコード上でクラスを一意に指定すると、新しくリリースされたスクールを使用したい場合にソースコードを変更しなければならない。そこで、ソースコード上ではスクール名を直接指定せずシンボリックな名称を与え、コンパイル時にこの名称と実際のスクールを対応させることにする。この対応をコンフィグレーションと呼ぶ。

クラスのバージョンの指定は、スクール中でクラス名とクラスサーバを対応させることで行なう。しかし、前に述べたようにバージョンは省略可能である。この場合はクラスサーバの代りにクラスバージョンサーバをクラス名に対応させる。バージョンが省略された場合は実行時に動的にバージョンが決定される。

8 まとめ

現在開発を進めているオブジェクト指向分散環境OZ++のクラス管理について、その名称空間の基本設計を中心に述べた。

OZ++では、クラスの名称を、スクール名、クラス名、バージョンで決定し、コンパイル時のコンフィグレーション、スクールによりこれらを指定する方法を用いる。これにより、柔軟な名前付けを行なうことができ、クラス名の衝突の回避や、クラスのバージョンの動的な指定や静的な指定を可能とすることができる。

本研究において熱心な討論を頂いた、藤野 晃延(富士ゼロックス情報システム)、千葉 滋(東京大学理学部)両氏に感謝する。

本研究は、情報処理振興事業協会「開放型基盤ソフトウェア研究開発評価事業」の一環として行われたものである。

参考文献

[1] 塚本他：「オブジェクト指向開放型分散システムOZ+の研究開発」、電総研彙報、vol. 56、No. 9、Sep. 1992

[2] 塚本他: 「OZ++ 開発計画」、情報処理学会第45回全国大会、Oct. 1992

[3] 西岡他: 「オブジェクト指向分散環境 OZ++ の言語の基本設計」、情報処理学会第46回全国大会、Mar. 1993