オブジェクト指向分散環境OZにおける分散クラス管理

3Ｔ－5　中川 祐(富士ゼロックス情報システム)　塚本 享治(電子技術総合研究所)

1　はじめに

オブジェクト指向分散環境OZは分散環境におけるソフトウェアの開発と利用を容易にすることを目的としたオブジェクトの共有と交換に基づくシステムである。このシステムは先に開発したオブジェクト指向分散環境OZ++[1]の次世代版である。

OZの実行コードの単位であるクラスはワールドワイドなIDを持ち、必要に応じてネットワークを介して配送される。クラスはネットワークを通じて世界中で共有されるため、分散したクラスを管理する機構が必要になる。本稿ではOZのクラス管理の設計を述べる。

2　OZのクラスの設計

2.1　クラスとパート

OZのクラスはOZ++のクラスの設計を元に再設計したものである。[2]

１つのクラスはインタフェース部、動作部と呼ばれる２つのパートから構成される。インタフェース部はクラスのインタフェースを定義する部分であり、公開メソッドのsignatureの部分がそれに該当する。動作部はインタフェース部以外の部分であり、クラスの動作を定義する。

インタフェース部は他の複数のインタフェース部を継承して定義することが可能である。１つの動作部は厳密に１つのインタフェース部の動作を定義するために提供される。動作部は他の１つの動作部を継承して定義することができる。

OZのプログラム上ではクラス相互の参照は通常はインタフェース部を参照する。動作部を参照するのは継承する動作部の指定と、オブジェクトを生成する場合のみである。

2.2　クラスID

OZのクラスはその性質上、ワールドワイドにユニークなIDを持たなければならない。これはOZの通信アドレス管理の仕組を利用する。OZの通信アドレスはまずDNSがドメインのIDとなる。ドメイン内でホストマシンにIDを付与する[3]。

クラスのIDはクラスがコンパイルされたホストマシンのIDにそのマシン内でユニークな名前を付与することで作成する。この名前はシステムが定めたあるディレクトリより相対のパス名に相当する。

2.3　スクール

OZのクラスはクラスIDで識別されるが、必然的に長い名前になりプログラム上に直接記述するには適さない。OZではOZ++に習ってスクールという対応表を用いる。プログラム上のクラス名はラベル名と呼ぶ。スクールはラベル名に対応するクラスIDを記すことができる。プログラマはコンパイルを開始する前にプログラムが参照するクラスのIDをスクールに記述しなければならない。CASE Toolはこれを支援する。

3　OZのクラス管理

OZのような分散システムのクラス管理はプログラムのバージョンと切り離して考えることはできない。OZはオブジェクトを永続させることができるシステムであるため、オブジェクトのライフサイクルは長い。また分散システムに限らずプログラムは進化し続けるものである。プログラムの変更を無制限に認めてしまうと、互換性のない古いオブジェクトは廃棄しなければならない。それでは保存する必要のあるデータはオブジェクトにできない。またプログラム間のインタフェースに互換性のない変更が加えられると、これも既存プログラムを廃棄することになる。分散システムはユーザがネットワークを介して世界中に分散していると考えるため、世界中のシステムを変更が起きる度に再コンパイルするわけにもいかない。

OZのクラスを設計する際、次の目標をたてた。

- １度生成されたオブジェクトの実行コードは生涯変化しない。これは旧バージョンのオブジェクトのデータが新バージョンに適合できるかシステムが判断する術がないためである。
- インタフェース部の変更に対しては互換性がない。インタフェース部を変更したクラスは全く別のクラスとして扱われる。
- 動作部の変更に対してはシステムは互換性を保証する。ただしそれはメソッドがプログラマの意図されたように呼び出されるということである。

Management of Distributed Classes in
An Object-Oriented Distributed Systems Environment
Yu Nakagawa(Fuji Xerox Information Systems, Co., Ltd.),
and Michiharu Tsukamoto(Electrotechnical Laboratory)

3.1 開発中クラスの管理

OZ++ ではすべてのクラスはコンパイル時にワールドワイドにユニークな ID を付与し、その再利用は認めない方針をとった。すなわちクラスは新規作成のみが許され変更は許されなかった。これは分散システム上のクラスはいかなる場合も共有が可能であり、変更を認めた場合そのクラスのすべての利用者を保護することが不可能であるという考え方に基づいている。しかしこの方法ではソフトウェア開発時に膨大なクラスを生成してしまうためハードウェアのリソースを浪費し、クラスの管理が繁雑になってしまった。

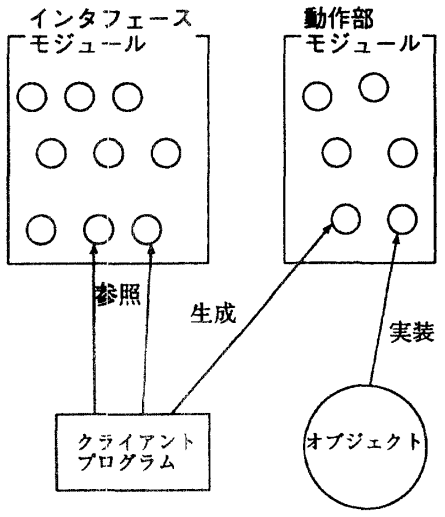
OZ ではクラスの管理をプログラムの開発時と運用時とに区別して設計することとした。これは開発時はプログラムの変更が頻繁であり、かつ開発中のものを何の手続きも経ず他のプログラマに提供する必要はないと考えたためである。

プログラム開発時のクラスはワールドワイドな ID を保持せず、またクラス ID の再利用が可能である。システムは開発中クラスのためのディレクトリを用意し、他のクラスと区別する。これらのクラスはクラス検索要求の対象にはされない。

3.2 クラスの公開

開発を終えたクラスを公開してネットワーク上で利用可能にするためにはもう 1 度コンパイルを行う。これは公開するクラスは別のディレクトリで行うためにクラス ID の変更が必要なことと、効率的なネットワーク転送を行うために複数のファイルを zip 形式に編集するためである。

クラスの公開はサブジェクトと呼ばれるモジュール単位で行うことを想定している。これは OZ++ の経験からあまり小さな単位で公開、更新、配送を行っても繁雑で転送の効率が悪く恩恵が少ないと判断したためである。



クラス ID はまずサブジェクトに付与され各々のクラスは相対的な名前を付加して ID とする。クラス ID は実際にはインタフェース部と動作部別々に付与されることになる。既存のオブジェクトは動作部を参照するため動作部のクラス ID は再利用することができない。しかし同一のインタフェース部を利用して新たなプログラムをバージョンアップしたいため 1 つのインタフェース部は複数の動作部に対応し、新たな動作部を追加できるようにしなければならない。これを可能とするにはインタフェース部、動作部とを別々に管理しなければならない。

3.3 オブジェクトのバージョンアップ

サブジェクトの提供者はインタフェース部を保存したまま動作部を変更することでサブジェクトのバージョンアップが可能である。しかしこれには問題がある。オブジェクト生成は動作部を指定して行われるため、顧客側のプログラムを再コンパイルしなければ新しい版のオブジェクトは生成されない。大抵のクラスライブラリは何かオブジェクトを生成するところから始まるためこれは不便である。

この問題を解決するために OZ++ ではシステムに特別な機構を用意した。OZ ではこれをプログラミングで解決する方針を採った。これはオブジェクトを顧客モジュールから直接生成せず、オブジェクト生成を行うコード自体を配送するという方法である。MVC を分散モデルに適用した場合、契約関係のあるクライアントのバージョンを自動的に更新するといったことも可能となる。

本研究は、情報処理振興事業協会 (IPA) の「創造的ソフトウェア育成事業」の一環として行われたものである。

このシステムは平成 9 年 7 月に α 版をリリースし、平成 10 年春に第 1 版のリリースを予定している。

参考文献

[1] 塚本：「オブジェクト指向分散環境 (OZ++) の研究開発」, 開放型基盤ソフトウェア研究開発評価事業報告書,IPA,'96

[2] 中川 他：「オブジェクト指向分散環境 OZ のクラスの設計」, 情報処理学会第 54 回全国大会,'97

[3] 杉野 他：「オブジェクト指向分散環境 OZ のアドレス解決機構」, 情報処理学会第 54 回全国大会,'97