7L－5

オブジェクト指向分散環境OZ++ のコンフィギュレーション管理の設計

西岡 利博*
三菱総合研究所

大西 雅夫*
東洋情報システム

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

*:開放型基盤ソフトウェアつくば研究室研究員

1　はじめに

ソフトウェアをネットワーク上で共有することによって、再利用が促進されるという期待が持たれている。これまでにも優れた品質のソフトウェアが数多くネットワークを通じて提供されている。しかし、ソフトウェアは常に進化するので、バージョンの不整合や、互いに関連のないパッチが発生しやすく、利用上の支障になることが多い。

OZ++は、分散オブジェクト指向プログラミング言語であるOZ++言語で記述されたプログラムをネットワーク上で実行できる環境である。OZ++は、ソフトウェアの共有を促進する枠組として設計されている。特に上述の問題は、コンパイル時に利用するクラスのインタフェースだけを定め、実行時にそれを実現している複数のバージョンから最もよいものが選ばれるようにすることで解決しようとしている[2]。メソッドのコードはネットワークを通じて自動的に供給される。これにより、常に最新のバージョンが利用でき、ソフトウェアの進化に柔軟に対応できるようになる。

しかし、ソフトウェアのデバッグ時や、動作の保証をするような場合に、バージョンが特定できないと不便なことがある。本稿では、この問題への対策について述べる。

2　クラスのバージョン管理

各クラスは、OZ++のクラス管理機構によって、バージョンを区別して管理されている。ひとつのクラスには、複数のインタフェースのバージョンがあり、ひとつのインタフェースには、複数の実装のバージョンがあるとされる。これらすべてにIDが与えられる。

インスタンス生成の場合、コンパイル時にはインタフェースのIDしか指定されないので、実行時に、そのクラスと、そのすべての祖先クラスにわたって、どの実装のバージョンを使うか決めなくてはならない。このため、使用する実装のバージョンのIDの組をあらかじめ作っておき、インタフェースのIDとペアにしてクラス管理機構で管理する。これをコンフィギュレーションと呼ぶ。ひとつのインタフェースに対して、複数のコンフィギュレーションを作成しておくことができる。

インスタンスは、実行時にコンフィギュレーションを指定して生成する。特に指定がないときは、そのような場合に使うことを推奨されているコンフィギュレーションが指定されているので、それを用いる。この機能により、バージョンの選択を実行時まで遅らせ、その時点で最もよいバージョンを選ぶことができると期待できる。

3　コンフィギュレーション管理

あるソフトウェアが使用すべきバージョンを指定するために、OZ++では、そのソフトウェアが生成するインスタンスに適用されるべきコンフィギュレーションをまとめたコンフィギュレーションセットを作成し、ソフトウェアの実行時に設定する。このとき、そのコンフィギュレーションセットを使用する範囲はどこまでか、および、それをどのように実装するかが問題である。

3.1　コンフィギュレーションセットの共有

共通のコンフィギュレーションセットを適用すべき範囲は、あるソフトウェアの実行の範囲である。

従来のOSでは、プロセスやジョブなどの単位で計算を区切り、そこに計算資源を帰属させるので、何がどのソフトウェアのものかは自明である。しかしOZ++は、ネットワーク上に分散したオブジェクトが互いにメッセージを送信し合うという計算モデルを持つので、どのメッセージがどのソフトウェ

The Design of the Configuration Management of OZ++: an Object-oriented Distributed Systems Environment
Toshihiro Nishioka* (Mitsubishi Research Institute, Inc.), Masao Onishi* (Toyo Information Systems, Co., Ltd.), Yu Nakagawa* (Fuji Xerox Information Systems, Co., Ltd.), and Michiharu Tsukamoto (Electrotechnical Laboratory);
*: Researcher, Tsukuba Laboratory, Open Fundamental Software Technology Project

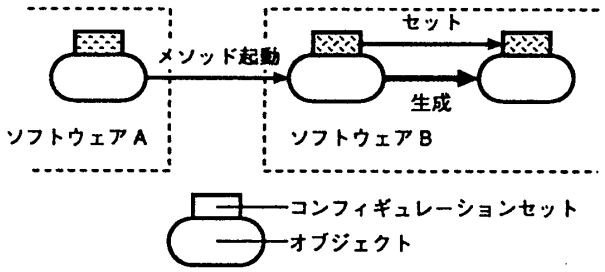

図 1: コンフィギュレーションセットの利用

アのものかは自明でない。このため、ソフトウェアの実行をメッセージ通信の連鎖として解釈するとうまくいかない。あるメソッドが生成するインスタンスのコンフィギュレーションは、そのメソッドの実装とは関係が深いが、そのメソッドを呼び出したメソッドとはあまり関係がない。そこで、ソフトウェアの実行の範囲を、次のように定めた。

- あるソフトウェアの実行の範囲とは、その最初に作られるオブジェクト (ランチャブルと呼ぶ) から始まって、そのメソッドの実行によって生成されたオブジェクトの再帰的な集合を指す。

すなわち、あるオブジェクトのあるメソッドがインスタンスを生成する際、そのオブジェクトに設定されているコンフィギュレーションセットがもしあればそれに従い、生成したインスタンスにもそのコンフィギュレーションセットを設定することとした(図1参照)。こうすることで、他のオブジェクトのメソッドを起動した場合は、起動された側のオブジェクトのコンフィギュレーションセットが使われることになり、直感と一致する。

以下では、実現の仕組みの概要を述べる。

3.2 コンフィギュレーションセットの生成

コンフィギュレーションセットは、OZ++の開発環境のひとつである、ブラウザ上の操作で作成される[1]。作られたコンフィギュレーションセットは、そのソフトウェアのランチャブルのクラスと組にして、カタログと呼ばれるデータベースサーバに登録される。ユーザは、ソフトウェアを起動するツールからカタログを参照して、ランチャブルのインスタンスを生成し、コンフィギュレーションセットを設定して実行する。

3.3 コンフィギュレーションセットの伝播

コンフィギュレーションセットは各オブジェクトが保持するのが自然である。しかし、そのメモリサイズは、100 個程度のクラスのインスタンスを生成するソフトウェアで1.6KB 以上と予想され、頻繁にはコピーできる大きさではない。

OZ++の実行モデルでは、オブジェクトはセルという単位に分かれて存在する[3]。セル内のオブジェクトはポインタで参照されるので、セル内にインスタンスを生成する場合は、単にコンフィギュレーションセットへのポインタのコピーで済む。コンフィギュレーションセットの本体のコピーが発生するのは、他のセルを作る場合と、他のセルへのメソッド起動の引数や返り値としてオブジェクトを送信する場合である。

他のセルを新たに作るのはそれほど多い処理ではないので、ここでは考えない。

セル間で頻繁にオブジェクトを通信する場合には影響がある。しかし、他のセルに送信されたオブジェクトが、そこでインスタンスを生成することは少なく、データやアルゴリズムを運ぶだけのことが多い。よって、そのようなオブジェクトはコンフィギュレーションセットを無効に(ポインタを NULL に) してから送信することで性能を改善できる。

なお、本稿で述べた、コンフィギュレーションセットの設定やコピーなどのためのコードは、コンパイラが自動的に出力するので、性能チューニングが必要な場合を除き、プログラマの負担とはならない。

4 まとめ

OZ++システムでは、ソフトウェアのコンフィギュレーションを実行時に決定するが、これが不便な場合に静的に決定する方法の設計と実現方針について述べた。

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

参考文献

[1] 音川他, “オブジェクト指向分散環境OZ++のプログラミングパラダイム”, SWoPP '95 プログラミング研究会にて発表予定, Aug. 1995.

[2] 新部他, “OZ++コンパイラによるクラスの版管理”, SWoPP '94 プログラミング研究会予稿集, IPSJ, Aug. 1994.

[3] 西岡他, “オブジェクト指向分散環境OZ++の実現”, SWoPP '95 プログラミング研究会にて発表予定, Aug. 1995.