3G−10

OZ+：オブジェクト指向開放型分散システム[1]
— OZ+ におけるバージョン機能 —

塚本 亨治（電総研）[2] 篠原 弘樹（松下電器）[3] 水谷 功（住友電工）[4] 梶浦 広行（シャープ）[5]

1 はじめに

OZ＋では、利用者に提供する基本的なタイプはタイプサーバという仮想計算機（以下VM）が保存する。タイプサーバは利用者の要求に応じてタイプの配布（要求元のVMへのタイプロード）を行なう。タイプサーバが保持するタイプの修正は多くの利用者に影響を与えるので、タイプサーバにはタイプのバージョンを管理する機能が必要である。今回、OZ＋のタイプに対するバージョン機能について検討したので報告する。

2 タイプのバージョン

バージョンとは一つの名前に対する複数の実体への対応のことをいう。複数の対応のうち、どの対応を使うかを決めるセレクタとなるのがバージョン番号である。タイプサーバの保持するタイプを利用するときはそのタイプのバージョン番号を指定する方法と、バージョン管理機能が用意する公開版を利用する方法の2つをOZ＋は用意する。

2.1 タイプのバージョン番号

タイプはタイプ名とバージョン番号で特定される。この指定は原則としてプログラム上で明示的に行なう。また、あるタイプのあるバージョンを作成するときもプログラム上で明示的に指定する。例としてタイプAのバージョン2の中でタイプBのバージョン3を使用するときの記述を以下に示す。

```
class A;
option version(A,2),version(B,3);
method xxx;
{    ....
}
end A;
```

このversion宣言はサブタイプに継承される。

2.2 タイプの公開版

上記のように利用するタイプのバージョンの全てをプログラム上で指定するのは利用者に大きな負担である。そこで、バージョン番号の指定のないタイプに関してはあらかじめ定められた公開版バージョン番号（公開版と呼ぶ）を割り当てる。この公開版のバージョン番号は変更可能である。また、タイプサーバ内の全てのタイプは公開版を持つ。公開版の管理はディレクトリが行なう。

3 OZ＋におけるタイプのバージョンの実現

タイプのバージョンの実現方法として次の2つが考えられる。

1. 1つのタイプ名オブジェクトに複数のタイプオブジェクトが対応する。
2. プログラムに記述されたタイプ名に複数のタイプ名オブジェクトが対応する。

OZ＋では名前は名前オブジェクトで表される。方法1では、タイプ名オブジェクトに複数のタイプオブジェクトが対応し、バージョン番号により、どのタイプオブジェクトにアクセスするかを決定する。このときタイプ名とタイプ名オブジェクトは1対1に対応する(図1)。しかしこの方法ではタイプ名オブジェクトが送信されるとき、そのタイプ名オブジェクトに対応するバージョン番号もいっしょに送らなければ、受信側は受け取ったタイプ名オブジェクトを用いてタイプオブジェクトにアクセスすることができない。また、バージョン番号をタイプ名オブジェクトといっしょに送った場合、受け取ったバージョン番号の管理は受信者側がおこなわなければならない。バージョンの指定は作成タイプごとに行なうので、実行系にかなりの負担がかかる。方法2では、アプリケーションで記述したタイプ名にOZ＋上の複数のタイプ名オブジェクトが対応し、バージョン番号により、タイプ名がどのタイプ名のオブジェクトに対応するかを決定する（図2）。この方法ではタイプ名オブジェクトとタイプオブジェクトが通常どおり1対1に対応しているので、実行系はタイプのバージョンを意識しない。また、タイプ名のオブジェクト自身にバージョンに関する情報を持たせることができるので、タイプのバージョンに関するいくつかのサービスはタイプ名オブジェクトのメソッドとして実現できる。以上のことからOZ＋では方法2を採用することにした。タイプ名と複数のタイプ名オブジェクトの対応付けを正しく行なうためには

- 名前オブジェクトの拡張
- ソースファイルのコンパイルからタイプオブジェクトをVMにロードするまでに行なうバージョンに関する処理の追加

が必要である。以下、これらについて述べる。

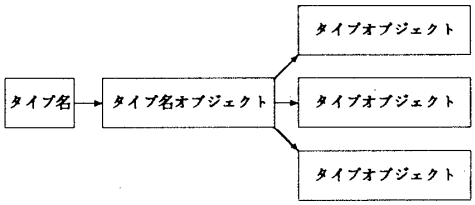

図1: 方法1

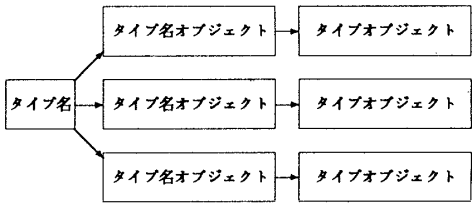

図2: 方法2

3.1 名前オブジェクトの拡張

1つのタイプの名前に対しては複数の名前オブジェクトが対応するが、タイプ名以外の名前（例えばメソッド名）は名前オブジェクトと1対1の関係にある。そこで、タイプ名オブジェクトと他の名前オブジェクトとを分け、前者のタイプをtypesymbol、後者のタイプをsymbolとする。タイプtypesymbolはタイプsymbolのサブタイプである。タイプtypesymbolのイン

[1] OZ+ : Object Oriented Open Distributed System — A Version Function in OZ+ —
[2] Michiharu TUKAMOTO (Electrotechnical Laboratory)
[3] Hiroki SHINOHARA (MATSUSHITA Electric Industrial Co., Ltd.)
[4] Isao MIZUTANI (SUMITOMO Electric Industries, Ltd.)
[5] Hiroyuki KAZIURA (SHARP Corporation)

スタンスを生成するときはプログラム上で

%タイプ名

と記述する。また、タイプ symbol のインスタンスを生成するときはプログラム上で

#名前

と記述する。

3.2 コンパイルからタイプロードまでの動き

コンパイラがソースファイルを読み込み、ローダがタイプオブジェクトを VM にロードするまでの動きを示す（図3）。コンパイラはソースファイルを読み込み、ソースを解析しコード化するが継承関係に関する処理は行なわない。リンカはコンパイラの出力を読み込んで、継承関係に関する処理を行ない、タイプオブジェクトファイルを生成する。タイプオブジェクトファイルはソースファイルで定義したタイプに対応する。ローダはタイプオブジェクトファイルを読み込み、VM 上にタイプオブジェクトを生成する。バージョンに関する処理は、このコンパイラ、リンカ、ローダによって行なう。コンパイラ、リンカはソースファイルに記述された全てのタイプのバージョン番号を決定する。ローダはバージョン番号をセレクタとし、タイプ名とタイプ名オブジェクトとの対応を決定する。タイプオブジェクトファイルはそのタイプの中で使用するすべてのタイプのバージョン情報を保持するテーブルを持つ。このテーブルを versiontable と呼ぶ。versiontable はタイプ名とバージョン番号の対応表である。この versiontable の作成はコンパイラ、リンカによって行なわれる。以下コンパイラ、リンカ、ローダが行なうバージョンに関する処理について述べる。

3.2.1 コンパイラにおけるバージョンの処理

以下の手順で処理を行なう。

1. 作成タイプのバージョンがオプションの version 宣言で記述されていないときは作成タイプのバージョン番号を 0 とし、これらを versiontable に登録する。version で宣言されているときはそのバージョン番号を versiontable に登録する。

2. 作成タイプのスーパータイプのバージョン番号が version で宣言されているときは、スーパータイプ名とそのバージョン番号を versiontable に登録する。version 宣言されていないときはスーパータイプ名だけを versiontable に登録する。対応するバージョン番号の欄は空けておく。

3. 利用者が version 宣言で指定したタイプ名とバージョン番号を versiontable に登録する。

4. 作成タイプのメソッド内に version 宣言されていないタイプのタイプ名オブジェクトの記述（%タイプ名）があるとき、そのタイプ名だけを versiontable に登録する。対応するバージョン番号の欄は空けておく。

3.2.2 リンカにおけるバージョンの処理

以下の手順で処理を行なう。

1. 作成タイプのスーパータイプのバージョン番号がコンパイラで決定できなかったとき、リンカはタイプサーバのディレクトリにスーパータイプの公開版のバージョン番号を要求する。ディレクトリから受け取ったバージョン番号を versiontable のスーパータイプ名に対応するバージョン番号の欄に登録する。

2. スーパータイプ名とそのバージョン番号を使って、タイプサーバにスーパータイプの versiontable を要求する。タイプサーバから受け取ったスーパータイプの versiontable と作成タイプの versiontable をマージする。

3. まだバージョン番号が確定しないタイプ名が versiontable にある場合、タイプサーバのディレクトリにそのタイプの公開版のバージョン番号を要求する。ディレクトリから受け取ったバージョン番号を versiontable に登録し versiontable を完成させる。

3.2.3 ローダにおけるバージョンの処理

OZ +ではオブジェクトは UID（Unique ID）によって識別される。名前に対して、該当する名前オブジェクトの UID を割り当てることにより名前と名前オブジェクトは対応付けられる。ローダは読み込んだタイプオブジェクトファイル内の名前に対し、この対応付けを行なう。通常の名前オブジェクトは 1 つの名前に対し、1 つの名前オブジェクトが対応する。しかし、タイプ名の場合、名前だけでは 1 つのタイプ名オブジェクトと対応を付けることができない。バージョン番号を用いて対応付けを決定するためにタイプ名オブジェクトはタイプ名とバージョン番号から構成されるシステム用の名前を持つ。タイプオブジェクトファイル内のタイプ名に対するタイプ名オブジェクトの UID の獲得は、次の手順で行なう。

1. タイプオブジェクトファイルが持つ versiontable からタイプ名に対応するバージョン番号を求める。

2. タイプ名と上で求めたバージョン番号を用いてシステム用の名前を生成し、それを用いてタイプ名オブジェクトの UID を獲得する。

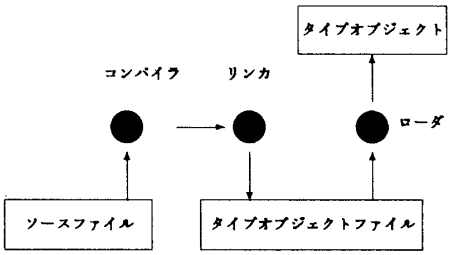

図 3: コンパイラ、リンカ、ローダの動き

4 おわりに

タイプのバージョンの実現方法について報告した。現在、実装中である。今後はこのバージョンの機能を用いて OZ +タイプサーバのタイプのバージョン管理システムを構築する予定である。なお、本研究は通産省大型プロジェクト「電子計算機相互運用データベースシステム」の一環として行なわれている。

参考文献

[1] M. TSUKAMOTO etal., The Architecture of Object-Oriented Open Distributed System : OZ, Interoperable Information Systems ISIIS '88, Ohmsha, PP153-166 (1988,11)

[2] 塚本他, OZ：オブジェクト指向開放型分散システムアーキテクチャ −オブジェクト指向型分散プログラミング言語とその実装 −, 情報処理学会 プログラミング言語研究会,21-4 (1989,6)

[3] 塚本他,OZ：オブジェクト指向開放型分散システムアーキテクチャ −オブジェクト指向型分散プログラミング言語の複数ユーザ環境への拡張 −, 情報処理学会 プログラミング言語研究会, 22-4 (1989,10)

[4] 塚本他、OZ：オブジェクト指向開放型分散システムアーキテクチャ −OZ +における名称の問題とタイプサーバのバージョン管理 −, 情報処理学会 第 4 1 回全国大会,6Q-2 (1990,9)

[5] 塚本他、OZ：オブジェクト指向開放型分散システムアーキテクチャ −OZ +システム管理の基本設計 −, 情報処理学会 第 4 1 回全国大会,6Q-3 (1990,9)

[6] 塚本他, OZ+：オブジェクト指向開放型分散システム −オブジェクトの分散管理 −, 情報処理学会 1990 年代の分散処理シンポジウム論文集, PP57-64 (1990,11)