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

6Q−2

－－－　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　－－－

塚本享治（電総研）　篠原弘樹（松下電器）　水谷功（住友電工）
近藤貴士（シャープ）梶浦広行（シャープ）笠井裕之（ＶＩＣ東海）澤田則和（ＶＩＣ東海）

１．はじめに

ＯＺとそれを拡張したＯＺ＋は異なるユーザ間でオブジェクトを共有し交換しながら処理を進めることを基本としている。このことはシステム上の名称、ｕｉｄ、及び実体との対応に矛盾があっては実現しない。

本稿ではこの問題を解決するために名称空間を導入する。また、タイプのバージョンも名称とｕｉｄの対応に関する問題であるので、名称空間を用いたタイプサーバのバージョン管理について述べる。

２．名称空間

2.1 名称空間の必要性

ひとつの名称にはひとつのｕｉｄが割当てられなければならない。そこで割当てを行なう管理機能が必要となる。しかし、すべての名称を一ケ所で管理することは難しい。そこでこの管理をいくつかの名称空間に分割する。

2.2 名称空間の定義

名称空間はｕｉｄを割当てる名称の範囲を定める。名称空間はディレクトリを持つ。ディレクトリは名称とｕｉｄの対応表を持ち、名称にｕｉｄを割当て、これを対応表に登録する。ユーザはディレクトリを用いて名称からｕｉｄを引いたり、ｕｉｄから名称を引いたりできる。

ある名称空間に属する名称は

空間名＄名称

と記述される。しかし、すべての名称に空間名をつけるとプログラムが煩雑になる。そこで、空間名を省略したインスタンス変数名やメソッド名の記述を許す。空間名を省略したインスタンス変数名やメソッド名は、それらを定義したタイプの名称と同じ名称空間に属する。

2.3 名称空間における継承の問題

親タイプと子タイプがそれぞれ異なる名称空間に属するときと同一の名称空間に属するときでself,here,superなどのメソッドサーチの動きが異なる場合がある。これは、名称空間が異なるときは、同じメソッド名に異なるｕｉｄが割当てられるからである。多くの異なるタイプで定義された同一な名称を持つメソッド群のことを「共通なメソッド」と言う。「共通なメソッド」に対し、上記のようなあいまいなことが起きないように、これらのメソッド名にｕｉｄを割当てる名称空間Ｐを用意する。

３．タイプサーバの名称空間

3.1 タイプサーバの名称空間の必要性

タイプサーバは異ユーザ間のオブジェクトの交換や共有に必要なタイプを記憶する。タイプサーバの位置付けはＯＺのライブラリであると考えられるので、一つのタイプサーバにあらゆるタイプを記憶させるより機能別にタイプサーバを作る方が、保守や管理がしやすい。そこで、ＯＺに複数のタイプサーバを持たせ、その機能ごとにタイプサーバの名称空間も複数作ることにする。

ＯＺのタイプ全体は単一継承の木構造をなす。タイプサーバが複数存在すると、継承の木構造は分割され、各々のタイプサーバで記憶されることになる。しかし、個々のタイプサーバの名称空間がそれぞれ、名称にｕｉｄを割当てると、継承木中に出てくる同じ名称に異なるｕｉｄが割当てられてしまう。

以下にこの問題の解決について述べる。

3.2 タイプサーバの名称空間

タイプサーバの名称空間はタイプサーバ内の全ての名称からなる。これらの名称空間は互いに通信しあい、名称のｕｉｄを一意に割当てる。（図１参照）ディレクトリは名称とｕｉｄの対応の他にタイプ名に関してはタイプサーバが割当てたタイプのｕｉｄも表に持つ。

3.3 タイプのバージョンと名称

あるタイプのバージョンとは、そのタイプと名称が等しくタイプのｕｉｄが異なるタイプのことをいう。これは今まで述べたディレクトリの機能だけでは管理できない。そこで、次にタイプサーバのバージョン管理について述べる。バージョン管理の問題は異なるタイプのｕｉｄを持ち、かつ同じ名称を持つタイプ群に対し名称のｕｉｄの割当てをどう行なうかということである。

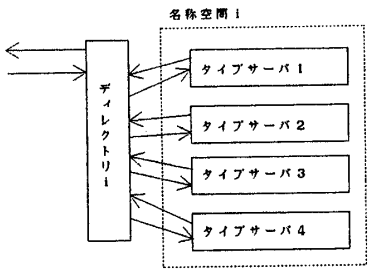

図１．タイプサーバの名称空間

OZ: Object Oriented Open Distributed System Architecture -- Study of Naming Probrems and a Version Management for OZ+
Michiharu TSUKAMOTO(1), Hiroki SHINOHARA(2), Isao MIZUTANI(3), Takashi KONDO(4), Hiroyuki KAZIURA(5),
Hiroyuki KASAI(6), Norikazu SAWADA(7)
(1)Electrotechnical Laboratory (2)MATSUSHITA Electric Industrial Co.,Ltd. (3)SUMITOMO Electric Industries,Ltd.
(4)SHARP Corporation (5)SHARP Corporation (6)VIC Tokai,Ltd. (7)VIC Tokai,Ltd.

4．タイプサーバにおけるタイプのバージョン管理

4.1 バージョン管理の必要性

タイプサーバの中のタイプを変更したときの影響はタイプサーバ内部だけでなく、外部にも影響する。しかも、その影響はそのタイプを利用しているオブジェクトやそのタイプのインスタンスだけでなく、そのタイプの子供のタイプ、及びそのインスタンス、その子供のタイプを利用しているオブジェクト等に及ぶ。そこでタイプを変更するかわりに、そのタイプの新しいバージョンを作りこれらに影響が及ばないようにすることが考えられる。

タイプのバージョン管理は次の事を実現する。

①あるタイプの新バージョンができる前に既にそのタイプを使用していたオブジェクトは新バージョンの影響を受けない。

②タイプのバージョンを意識しなくても、ユーザはそのタイプを利用することが可能である。

4.2 バージョン番号

タイプのバージョンを区別するためにバージョン番号を導入する。バージョン番号はプログラム上でユーザが設定する。ユーザはバージョン番号を０から始め、バージョンを作成するごとに、１、２、３、．．．と順番に増加させていく。あるタイプのバージョンを作るとき、プログラム上で次のように書く。

```
class typeName;
option version(typeName,versionNumber);
```

オプションのversionを省略したときはバージョン番号は０となる。バージョン番号をプログラム上で設定することによりソースの保守が簡単になる。

4.3 バージョン管理

ディレクトリの対応表にバージョン番号の欄を追加し、バージョン番号の管理もディレクトリが行なうようにする。ディレクトリを用いたバージョン管理を以下に示す。

①タイプサーバはタイプ名、そのタイプのｕｉｄ、そのタイプのバージョン番号を引数として、ディレクトリにタイプ名のｕｉｄ要求を行なう。

②ディレクトリはバージョン番号が正しいかチェックを行なう。正しくないときはタイプサーバにエラーを通知する。

③バージョン番号が正しいとき、ｕｉｄの割当て、及び対応表への登録を行なう。

これによりタイプ名が同じでもバージョンが異なれば異なるｕｉｄが割当てられる。

ユーザがタイプサーバの中のあるタイプのあるバージョンをロードするには、そのタイプの名前とバージョン番号を用いてディレクトリにｕｉｄを問合せなければならない。そこでプログラム上でオプションのバージョン指定（version）を用い、タイプ名とバージョンの番号の指定を行なう。例えばあるタイプのバージョンをスーパータイプに用いるには次のように書く。

```
class typeName;
option version(stypeName,versionNumber);
super stypeName;
```

また、あるタイプの最新バージョンがいくつかを知るにはディレクトリのオブジェクトaVersionに次のようなメッセージを送る。Ｔｓはタイプサーバの名称空間の空間名である。

```
aVersion:Ts$latest(typeName)
```

しかし、タイプのバージョンをそのまま使うのでは利用者は常にバージョン番号を意識しなければならない。これでは②の要求を満たすことができない。そこで、タイプの公開版という考え方を導入する。

4.4 タイプの公開版

タイプの公開版とはタイプサーバにバージョン番号を指定しないでタイプのロード要求を行なうとき（前述のオプションのバージョン指定を省略するとき）、ロードされるタイプのバージョンのことである。公開版の設定はディレクトリにあるオブジェクトaVersionにメッセージを送ることにより行なわれる。公開版を指定しないときはバージョン０がデフォルトで公開版となる。公開版の指定は次のように行なう。

```
aVersion:Ts$publish(typeName,versionNumber)
```

公開版のバージョン番号を知るには次のようなメッセージをaVersionに送る。

```
aVersion:Ts$publishNum
```

5．おわりに

ＯＺ＋による名称空間、タイプサーバにおけるタイプのバージョン管理の方法について報告した。今後、これらを実装し、評価を進めていく予定である。

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:ｵﾌﾞｼﾞｪｸﾄ指向開放型分散ｼｽﾃﾑｱｰｷﾃｸﾁｬｰ ｵﾌﾞｼﾞｪｸﾄ指向型分散ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ言語とその実装、情報処理学会 ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ言語研究会、21-4 (1989,6)

3) 塚本他：OZ:ｵﾌﾞｼﾞｪｸﾄ指向開放型分散ｼｽﾃﾑｱｰｷﾃｸﾁｬｰ LLCﾀｲﾌﾟ3を活用する通信ｱｰｷﾃｸﾁｬ、実装、およびその評価、情報処理学会 ﾏﾙﾁﾒﾃﾞｨｱ通信と分散処理研究会、43-4 (1989,9)

4) 塚本他：OZ:ｵﾌﾞｼﾞｪｸﾄ指向開放型分散ｼｽﾃﾑｱｰｷﾃｸﾁｬｰ ｵﾌﾞｼﾞｪｸﾄ指向型分散ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ言語の複数ﾕｰｻﾞ環境への拡張、情報処理学会 ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ言語研究会、22-4 (1989,10)