オブジェクト指向分散環境OZ++の言語における例外処理の記述

1H－1

音川 英之*（シャープ） 新部 裕*（三菱総合研究所）
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）
* 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++言語は、オブジェクト指向分散環境OZ++上においてネットワーク上のソフトウェアの記述を容易に行うためのオブジェクト指向プログラミング言語である[1]。ここでは、OZ++言語の例外処理に関する言語機能について述べる。

2 例外処理の概要

OZ++言語はオブジェクト指向プログラミング言語であり、オブジェクトに対するメッセージ送信によるメソッド呼び出しによって処理を記述する。1つの処理に複数のメソッド呼び出しが関わる場合にはメソッド呼び出しの連鎖を構成することになる。OZ++言語の例外処理では、このようなメソッド呼び出しの連鎖中における大域脱出の手段を提供する。つまり連鎖中のある部分で例外を発生させることによって、その連鎖の前に制御を移動しその例外に対する処理を行うことが可能である。またOZ++言語でのメソッド呼び出しはネットワーク上の任意のオブジェクト間で記述することが可能であるため、メソッド呼び出しの連鎖はネットワークをまたがったものになる可能性があるが、例外処理ではこのような連鎖中においても大域脱出が可能である。また発生させる例外と共に1つの任意の型の値を渡すことが可能である。

3 例外名の定義

OZ++言語の例外処理では例外名によって例外を識別し、例外名はシステムであらかじめ定義したもの以外にユーザによる定義が可能である。ところでOZ++言語ではメソッドやインスタンス変数などのユーザが定義するものはすべてクラスに定義するが、これはそれらがクラスあるいはクラスから生成されるインスタンスと密接な関係があるからである。これに対して例外はクラスやインスタンスではなくメソッド呼び出しの連鎖に関わるものであり、例外名はクラスあるいはインスタンスに格納し実行時に動的に利用される情報ではなく、プログラミング時にのみ利用されるものである。したがって例外名をクラスに定義するのではなく、例外名を定義するためにクラスとは異なる言語機能であるsharedを導入している。sharedはプログラミング時に複数のクラスで共有して利用されるものを定義するものであり、例外名の他には定数を定義することができる.

sharedは次のように定義する。

```
shared SharedString {
  OutOfRange (unsigned int);
        // 例外名の定義
  int DefaultInitialLength = 20;
        // 定数の定義
}
```

sharedに定義した例外名および定数にアクセスするには「shared名::例外名または定数」の形式で指定する。

4 例外処理の記述

OZ++言語の例外処理は、例外を発生させるraise文と発生した例外を受け取る例外文によって記述する。システム定義の例外とsharedで定義するユーザ定義の例外とでは、例外名の記述の際に前者が単に例外名だけであること以外には言語上の違いはない。つまりraise文ではシステム定義の例外も発生させることが可能であり、例外文ではシステム定義の例外に対する処理も記述することが可能である。

raise文は通常、

```
raise SharedString::OutOfRange (100)
```

のように例外名の指定が必要であるが、例外ハンドラの中でのみ例外名を省略した記述が可能である。この

A language specification of exception handling in OZ++: An Object-Oriented Distributed Systems Environment
Hideyuki Otokawa* (Sharp Corporation),
Yutaka gNiibe* (Mitsubishi Research Institute, Inc.),
Yu Nakagawa* (Fuji Xerox Information Systems, Co., Ltd.),
and Michiharu Tsukamoto (Electrotechnical Laboratory)
*: A Research Fellow of Open Fundamental Software Technology Project in Information-technology Promotion Agency, Japan

場合は、受け取った例外を現在のメソッド呼び出しの連鎖の1つ前に送る (例外のreraise) という意味になる。

例外文は次のように例外ブロックと例外ハンドラリストからなる。

```
String str;
        :
try {   // 例外ブロック
  str->At (100);
} except {      // 例外ハンドラリスト
  SharedString::OutOfRange (index)  {
        //例外ハンドラ
    str->At (str->Length () - 1);
  }
  default {
        :
    raise;
  }
}
```

例外ブロックには通常の処理内容を記述し、例外ハンドラリストには、0個以上の例外ハンドラを記述し、各例外ハンドラにおいて例外ブロックの実行時に発生する可能性のある例外に対する処理を記述する。例外ハンドラには例外名にdefaultを指定したものを記述することが可能である。この例外ハンドラは同じ例外ハンドラリストにある他の例外ハンドラで指定されていない例外を受け取るためのものであり、発生した例外に対する処理は特に行なわないが例外ブロックで獲得した資源の解放処理が必要であるという場合に利用するためのものである。

5 例外発生時のモニタロックの解放

OZ++言語ではモニタによる排他制御を行うメソッドを定義することが可能である。このメソッドは終了時 (return文、raise文の実行を含む)にモニタロックの解放処理を実行する。ここで

a() → b () → c (), b は排他制御付きメソッド

のようなメソッド呼び出しの連鎖内で例外が発生した場合、最終的には制御は連鎖の先頭aまで戻っていく[2]が、bの中で実行されるcのメソッド呼び出しが例外ブロック中になければ、連鎖を戻る際にスキップされてしまいモニタロックが解放されないことになる。このような状況を防ぐために、排他制御を行うメソッドについては、

```
b () : locked {          b () : locked {
        :                  try {
  c ();          →               :
        :                    c ();
}                                 :
                           } except {
                             default {
                               raise;
                             }
                           }
                         }
```

のようにメソッド全体が例外ブロック中に記述されているものとして扱うことによって、連鎖を戻る際にスキップされることなく例外をreraiseする際にモニタロックの解放が行われることを保証している。

6 まとめ

OZ++言語はメソッド呼び出しの連鎖内における大域脱出の手段として例外処理の機能を提供している。例外処理の対象となる例外にはシステム定義のものとユーザ定義のものがあり、例外は例外名によって識別する。ユーザ定義の例外を定義するための言語機能としてsharedを提供する。sharedには例外の他に定数を定義することが可能である。例外処理はraise文と例外文により記述する。

現在ここで述べた機能および必要なランタイムルーチンを処理系、実行系に実装を終え、テストを行っている。今後は例外処理を管理系やアプリケーションの実装に導入し評価を行う。

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] 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

[2] 濱崎他: 「オブジェクト指向分散環境OZ++のプロセスと例外処理の実装」, 情報処理学会第50回全国大会, Mar. 1995