分散ソフトウェア開発用ツールキット－環境－

6D－7

佐藤 泰典† 鈴木 寿郎‡ 中澤 修†

†沖電気工業（株） マルチメディア研究所 ‡（株）沖テクノシステムズラボラトリ

1 はじめに

近年, 計算機環境の分散化が進み, アプリケーションシステム自体もネットワーク結合された複数の計算機を意識する分散マルチプロセス型となりつつある. しかし, マルチプロセス型のシステムを開発するためには, プロセス間通信による複数プロセス間の同期/交信処理が必要となり, Socket/Stream あるいは RPC 等の低レベルな API を利用しなければならない. また, マルチプロセスプログラム開発のためのデバッグ環境も提案されているが, 分散環境下では分散プロセスの状態把握が容易ではなく, プロセス間のグローバルな時間管理などに問題があった[1].

そこで, 我々はオブジェクト指向の概念に基づき, 複雑な通信処理を隠蔽した分散アプリケーション開発用オブジェクト指向ライブラリ[2]と, そのライブラリを利用してプログラムを作成するための開発支援用ツールの開発を行なった. 本論文では, 現在試作中の開発支援用ツールの概要を述べる.

2 構造

開発支援用ツールは大きくわけてモニタ機構, 実行状態表示機構およびプログラム作成支援機構からなる. 以下にそれぞれの機構の概要を示す.

2.1 モニタ機構

モニタ機構は文献[2]に述べた主サーバPMと従サーバPOBを拡張することにより実現した.

図1に示すようにプロセス α とプロセス β が交信する場合, それぞれのプロセスを管理する POB α /POB β を介して行なう. POB で処理するイベントは, モニタ機構起動時はその情報を PM へ通知し, PM がモニタ出力を生成する. ただし, 伝達者が伝えるメッセージは必ず PM に伝えられるため, POB は PM へ報告する必要はなく, PM が各 POB へ配送する情報を用いてモニタ出力データを生成する.

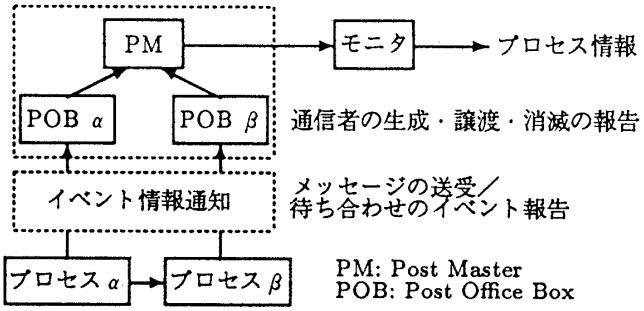

図1: プロセスモニタ機構

2.2 実行状態表示

モニタ機構からのプロセス情報をもとに, ウインドウへプロセス実行状態およびプロセス通信状態の表示を行なう.

表示形式としては, アニメーション的にプロセス状態/通信の遷移がわかるものと, 時間軸に沿ってプロセス状態/通信の遷移がわかるものの2種類をサポートしている. 図2に表示例を示す.

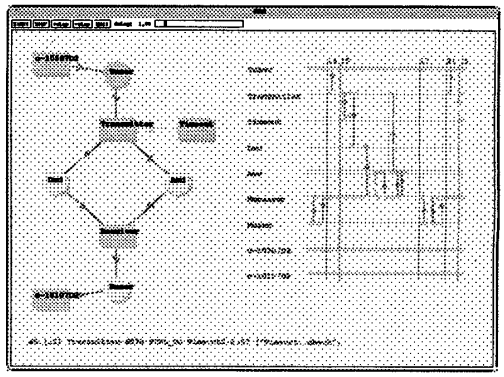

図2: 実行状態表示例

なお, モニタ機構からの出力がファイルであった場

The ToolKit for Distributed Software Development
- Environment -

Yasunori SATO [1], Hisao SUZUKI [2], Osamu Nakazawa [1]
1 Oki Electric Industry Co.,Ltd.
2 OTSL

合は, プロセス状態ログ表示開始点やログの時間軸スクロールが行なえるため, 詳細なプロセス実行状態の把握が可能である.

2.3 プログラム作成支援環境

通常, プロセス型のオブジェクトの処理は,

- 他プロセスオブジェクトから送信されるメッセージを待つ
- 受信メッセージを解釈し, 対応するメンバ関数を実行する
- 必要であれば, 実行結果メッセージを返信する

という無限ループ型の構造を持つ. そこで, AP 開発者には必要最低限の宣言/定義をプロセス宣言/関数定義ファイルとして記述させ, そのファイルから上記プロセス型処理構造に対応するヘッダファイル等を自動生成することで, プログラムの作成効率を向上させることができる. プロセス宣言部の構造は

プロセス宣言 = $\alpha\{\beta \mid \gamma \mid \delta\}\varepsilon$

であり, α, β, γ, δ, ε は, それぞれ表1のような物理行である.

表 1: プロセス型クラスのプロセス宣言部の構造

クラス名宣言行(α)	"process" 識別子 "{"
手続き名宣言行(β)	"procedure" 識別子 {"," 識別子 } ";"
関数名宣言行(γ)	"function" 識別子 {"," 識別子 } ";"
空行(δ)	
終止行(ε)	"}" ";"

α で記述された識別子を π とするとき, 自動生成するファイルは,

- π.ch （d_Process 派生クラス π 定義ヘッダ）
- πR.cc （遠隔プロセス専用実装プログラム）
- π.cc （実装プログラム）

の3つである. 以下に, プロセス宣言／関数定義ファイルの例と, その内容から生成するプログラムの例を示す.

1. プロセス宣言/関数定義の例

```
process Receiver | function count; | ;
Data Receiver::count(const Data &,const
                MessageID &)
  | static int count = 0; return count++; |
```

2. プロセス宣言/関数定義を基に自動生成されるプログラム例(抜粋)

```
/*  Receiver.ch  */
class Receiver: public Process |
 Receiver(Remote't, const char *NAME):
      Process(Remote, NAME) ||
 ~Receiver() || ;
 enum | Receiver'count | ;
 const Data & count(const Data & OPERAND =
   Data::Nil);
#ifndef REMOTE
 Receiver(const char *NAME);
 void applies(const Data &OPERATOR,
          const Data &OPERAND,
          const MessageID &);
 Data count(const Data &OPERAND,
       const MessageID &);
#endif
| ;
#endif              // Receiver.ch

/*  Receiver.cc  */
Receiver::Receiver(const char *NAME):
                  Process(NAME)
  | if (this ->forks() == 0) | Message m;
  for (;;) | if (this ->receives(m))
             this ->evaluates(m);|
 ||
Data Receiver::count(const Data &,
                const MessageID &)
 return count++; // Receiver.cc

/*  Receiver-R.cc  */
const Data & Receiver::count(const Data
                    &OPERAND)
  | static Message m;
  int i = invokes(Data("Receiver::count",
                Receiver'count), OPERAND);
  m = replyTo(i);
  return m.text();|   // Receiver-R.cc
```

3 おわりに

以上, 本稿では分散計算機環境における分散アプリケーション開発支援環境用ツールについて述べた. 本ツールの特長としては, デバッグ/実行状態把握の容易性, プロセスライブラリのカスタマイズの容易性などがあげられる.

今後は, プロセス間通信/関数定義の制御などを GUI 環境から行なうための統合的ビジュアルプログラミング環境の開発等をすすめていきたい.

参考文献

[1] Charles E.McDowell, David P.Helmbold: "Debugging Concurrent Programs", ACM Computing Serveys, Vol.21, No.4, pp.593-622, 1989

[2] 鈴木他:分散ソフトウェア開発用ツールキット − ライブラリ −", 情報処理学会第 48 回（平成 6 年前期）全国大会, 6D-6, 1994