2M−1

C++に基づく離散型シミュレーション言語CCDSの拡張機能

越田一郎　横川智巳
(東京工科大学　工学部)

1．はじめに

CCDS (C++ Classes for Discrete Simulation) はC++言語に基づいた離散型シミュレーション言語である．CCDSは離散型シミュレーションを行うために必要な機能をC++のクラスとして提供しており，ユーザはそれらを組み合わせることにシミュレーションプログラムを作成する．CCDSの基本的な機能および内部構造については既に報告した[1, 2]が，本稿では統計機能の拡張，ユーザによる既存ノードのカスタマイズについて述べる．

2．CCDSの概要

CCDSでは，シミュレーションの対象となるシステムをメッセージとノードという2種類のオブジェクトを用いてモデル化する．メッセージは，システムの中を流れる実体や情報を抽象化したもので，ノードはメッセージに対して処理を行うプロセスを抽象化したものである．CCDSのプログラムは相互にメッセージをやりとりする複数のノードによって構成される．また，シミュレーションの実行を管理するために，シミュレーションドライバと呼ぶ特別なオブジェクトが存在するるが，これはCCDSシステムにより自動的に生成される．

離散型シミュレーション言語ではイベントリストを用いてイベントの時刻管理を行うのが普通である．CCDSではイベントリストを，各ノードで分散管理することにより，イベント追加に要するコストを軽減している．

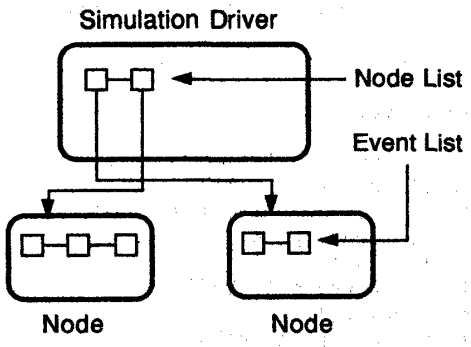

図1　CCDSシステムの構造

3．統計機能の拡張

複数ノードにまたがる統計量を収集するため，メッセージにパラメタを持たせ，時間および整数の値をセットできるようにした．さらにパラメタを扱うためのノードを3種類作成した

- AssignParameter：メッセージに整数データをセットするノード
- AssignTime：メッセージに時間データをセットするノード
- Tabulate：指定したパラメタのデータを集計し，統計量を計算するノード

4．その他の新ノード

統計機能の拡張の他，以下のノードを新しく定義した．

- Test：条件分岐のためのノード．分岐の条件はユーザが自由に設定できる．
- Priority：メッセージに優先度を指定するノード．

5．既存ノードのカスタマイズ

CCDSを使ったシミュレーションでは，CCDSが標準的に提供するノードを組み合わせてプログラムを作成するのが基本である．そのために前述のような様々なノードを提供している．

しかし，ユーザの要求は多様であり，それに対応しようとすると極めて多種類のノードを用意しなければならない．だが，そうしても，すべてのユーザを満足させることはできないだろう．

そこで，CCDSではシステムが提供しているノードをユーザがカスタマイズすることを可能にした．その実現方法は，X-toolkitのコールバック関数と同様な考え方に基づくものである．

Extended Features of C++ based Discrete Simulation Language CCDS
Ichiro KOSHIDA, Tomomi YOKOKAWA
Tokyo Engineering University

CCDSの各ノードは，ノード本来の機能以外の処理を実行するためのコールバック関数を持っている．メッセージの入出力がある標準的なノードでは，メッセージがそのノードに入ってくる時およびメッセージがノードから出て行く時に対応する，少なくとも2個のコールバック関数を持っている．ただし，CCDSが標準的に提供しているノードのクラスでは，コールバック関数は何も行わないものとして定義されている．

コールバック関数はC++の仮想関数として実現されている．ノードに新たな機能を付け加えたいと思うユーザは，コールバック関数を再定義する形でオリジナルのノードから派生クラスを作れば良い．

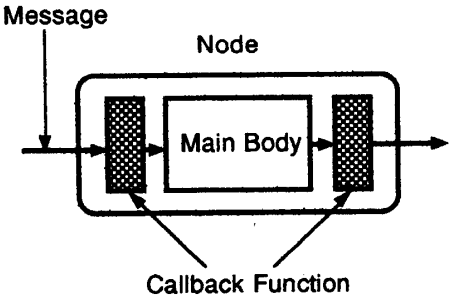

図2　ノードの拡張機能

6．プログラム例

図3に示すモデルのCCDSプログラムをリスト1に示す．このプログラムはSun4/1+ワークステーション上でSun C++ 2.0を用いてコンパイルおよび実行した．

7．おわりに

離散型シミュレーション言語CCDSの拡張機能，特に統計機能の拡張とユーザによるカスタマイズについて述べた．現在，さらに追加すべき機能を評価する目的でCCDSシステムを用いて種々のシミュレーションを行っている．また，シミュレーションプログラムの開発およびシミュレーション結果の解析を容易にするための，グラフィックユーザインタフェースも開発中である．

＜参考文献＞

[1] 越田：「C++に基づくシミュレーション言語の試作」，1989年電子情報通信学会春期全国大会D-345，1989

[2] 森川，三角，越田：「C++に基づく拡張可能な離散型シミュレーション言語」，情報処理学会第40回全国大会4J-3，1990

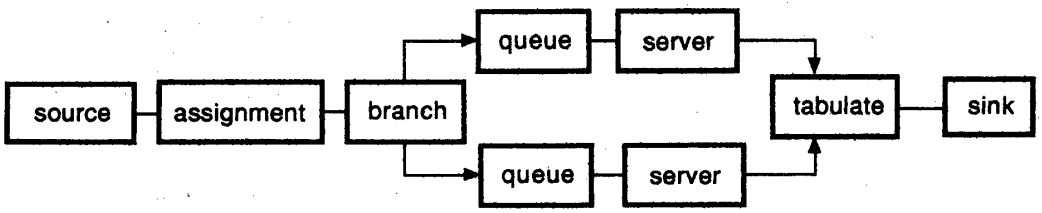

図3　プログラム例のモデル

```
main(){
  Source        source1;
  AssignTime    assigntime1;
  Branch        branch1;
  FIFO          queue1,queue2;
  Server        server1,server2;
  Tabulate      tabulate1;
  Sink          sink1;

  long          count;

  source1.set( &assigntime1 , SourceDist );
  assigntime1.set( &branch1 );
  branch1.set( &queue1 , &queue2 , BrSel );
  queue1.set( &server1 );
  server1.set( &tabulate1,ServerDist1,1 );
  queue2.set( &server2 );
  server2.set( &tabulate1,ServerDist1,1 );
  tabulate1.set( &sink1 , 10 , 20 , 20 , 0);

  cout << "SourceDist ServerDist1 MaxCount¥n";
  cin  >> lambda1 >> lambda2 >> count;

  sink1.setEndCount( (unsigned long)count );

  _sim_driver.run();

  printf("¥nSourceDist:%8.2f¥n",lambda1);
  printf("ServerDist1:%8.2f¥n",lambda2);
  printf("End count :%lu¥n¥n",count);
  printf("queue1 print¥n");
  queue1.print();
  printf("queue2 print¥n");
  queue2.print();
  printf("server1 print¥n");
  server1.print();
  printf("server2 print¥n");
  server2.print();
  tabulate1.print();
}
```

リスト1