Hybrid Simulator に必要な機能

6K−5

藤崎智宏 荒野高志 青野博

NTT ソフトウェア研究所 ソフトウェア基礎技術研究部

1 はじめに

ソフトウェアの生産過程において、設計を変更するのにかかるコストは生産過程の後の段階にいけばいくほど高くなる。そのため、初期の段階でプロトタイピングの手法を使い、要求者と生産者とのギャップを埋める、性能の見積りをする、といった手段が用いられて来た。

[Arano93a]において、プロトタイピングの新しい手法が提案されており、この手法をシミュレートするSimulator (以下 Hybird Simulatorと呼ぶ)の構成法が[Arano93b]において述べられている。 Hybrid Simulator は設計の途中の段階で仕様記述オブジェクトと実装オブジェクトを混在させてシミュレートする環境を与える。この環境を使うことで、仕様記述のみのシミュレートよりもより詳細な振舞いの観察や実装部分の性能実測が可能となる。すなわち、一般的な仕様検証で得られる動作の保証やデッドロック検出などだけでなく、性能から来るボトルネックなどを示すことが出来る([青野93a])。しかしながら、[Arano93b]で述べられている構成は、シングルプロセスからなるシステムの設計に主眼をおいたものであるため、設計に対して制限があり、また、複数プロセスからなるシステムの設計には使用することが出来ない。本稿では Hybrid Simulator において、[Arano93b] の構成法における設計者に対する制限を緩和し、複数のプロセスからなるシステムの設計にも適用できるように拡張する際に考慮しなければならない点について述べる。

2 Hybrid Simulator Versin 1

現在、[Arano93b]で提案された構成に基づく Hybrid Simulator が稼働している。以下、この版の Hybrid Simulator を HSV1 と表記する。HSV1 を用いて設計している途中の様子を図1に示す。

図1では、6個のオブジェクトからなるシステムを設計している。そのうちの一つのオブジェクト(長方形に描かれているオブジェクト: CardHandler) は既に実

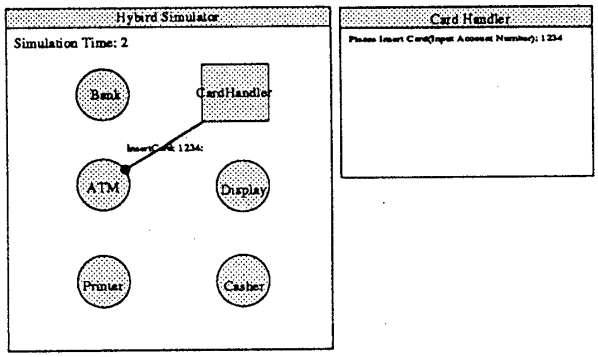

図 1: Hybird Simulator での設計

コードとして実装されているが、他の仕様記述言語で記述されたオブジェクト (円形のもの) とメッセージパッシングを行なうことが出来る。この機構によってシステム全体の動作チェック、性能実測などが可能になる。仕様記述言語で書かれたオブジェクト全てが実コードとして実装されると、システムが完成する。

HSV1 において、設計途中の状態、すなわち、仕様記述と実コードのオブジェクトが混在している状態でのシステムの構成を図2に示す。

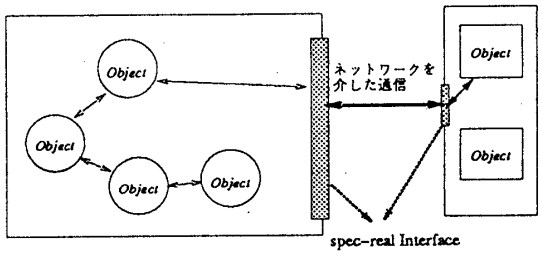

図 2: HSV1 のシステム構成

HSV1 は単一プロセスからなるシステムを対象にしているため、実コード化された部分は一つのプロセスとして実現されており、仕様記述 - 実コードの間のインタフェースは IPC を使って実装されている。また、現在は使用記述として状態遷移表記を、実装言語として

The function of New Hybrid Simulator.

Tomohiro FUJISAKI(fujisaki@nttslb.ntt.jp),
Takashi ARANO(arano@nttslb.ntt.jp),
Hiroshi AONO(aono@nttslb.ntt.jp),
NTT Software Laboratories.

C++ を対象としている。
現状では設計の際に以下のような制限がある。

- 仕様を記述する際、メッセージの送信、受信をペアで書かねばならない (図 3)
  これは、実装に使う言語として C++,Objective-C などの手続き型の非並列言語を使用し、1 プロセスからなるシステムを設計するために起こる。状態遷移表記では並列性を記述できるが、C++ などの言語では並列性を記述することが出来ない。すなわち、両言語間で対象とするモデルにギャップがあることが原因となっている。仕様記述者がこの制限の元にシステムを設計するのは面倒である。

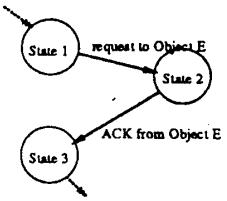

図 3: 仕様記述の例

- 他のオブジェクトへの呼び出しを含まないオブジェクトのみ実コード化出来る
  実コード化されたオブジェクト群は一つのプロセスに属するため、実コード化されたオブジェクトが他のオブジェクトを呼び出すことが出来ない (図 4)。これは、実コード化されたオブジェクトを扱うハンドラが 1 プロセスで実現されているため、制御を他のオブジェクトに移すことが不可能であるためである。

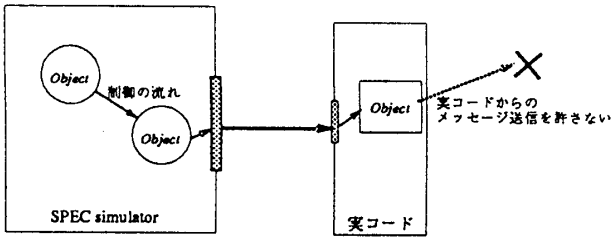

図 4: 実コードからの他オブジェクトの呼び出し

3 Hybrid Simulator の拡張

HSV1 の持つ制限を取り払うためには、実装部分に並列性を持たせること、すなわち、複数プロセスから構成されるシステムの設計に Hybrid Simulator を使用できるように拡張すればよい。複数プロセスに対応させることで制御の流れの多重化をすることができるため使用記述とのモデルギャップをなくすることが出来る。拡張の際、HSV1 と同様、1 プロセスからなるシステムの設計にも使えるようにする。そのための拡張 Hybrid Simulator の構成は 図 5の用になる。

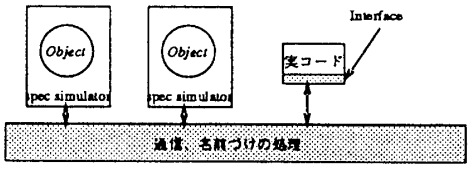

図 5: 拡張 Hybrid Simulator の構成

拡張 Hybird Simulator に必要で、HSV1 にはない機能は、以下のものである。

- 通信部分の自動生成
  サーバ、クライアント間の通信部分のコードを自動生成する。これにより、設計者がシステム自体には関係ない IPC について気にせずにいられる。問題点として、どのようなプロトコルを使った通信にするのがいいのか、ということを判定できるか、ユーザに指定させるのかといったことがある。

- オブジェクトのグループ化
  二つのオブジェクトを 1 プロセスにしたいと設計者が考えた場合、両オブジェクト間での通信は IPC でなく、関数呼びだしになる。これも (半) 自動的に切替えるような機構が欲しい。 HSV1 と同等の機能もこの機構で実現できる。

4 まとめ

HSV1 での制限を取り払い、複数プロセスからなるシステムを Hybird Simulator で設計可能にするために Simulator に必要な機能ついて考察した。今後、実装これらの機構を実装していく。

参考文献

[Arano93a] Arano,etc. : “An Object-Rriented Prototyping Approach to System Development”, COMPSAC ’93 to be published.

[Arano93b] Arano,etc. : “A New simulation Technique in Software Prototyping”, Summer Computer Simulation Conference ’93.

[青野 93a] 青野、荒野: “Hybrid Simulator における性能実測とその課題” 93 年情報処理学会秋季大会