3D-2

Ａｄａ*による分散型システムの開発について（Ⅲ）
― 試作システムの改良 ―

堀　正弘，　荒木　啓二郎，　牛島　和夫
（九州大学　工学部　情報工学科）

１．はじめに

我々は、従来よりＡｄａを用いたソフトウェア開発法について研究を行っている。そこでは、Ａｄａを単に実現のためのプログラミング言語として利用するのみならず、システムの仕様記述や設計のための言語としても用いる。即ち、まずＡｄａを用いてシステム全体を記述し、次にこれを動作させることによってシステムの仕様や設計の妥当性を確認し、その後で更にプログラムを改良する。このサイクルを繰り返すことによってプログラム自体の完成度を高めていくという方法である。その際Ａｄａのタスクを使う事によって、いわゆるオブジェクト指向のプログラミングを行うことができる。この結果自然で理解し易いプログラムを書くことが容易になる。

我々はこれまでにワークステーションとプリントサーバーとからなる通信システム[1]やＸ．２５パケットレベルプロトコル[2]の記述や実現を行い[3]、これらの開発実現に対してＡｄａが有効に利用できることを実証してきた。ところが、ワークステーションとプリントサーバーとからなる通信システム（以下、初版プログラムと呼ぶ）については、出来上がったＡｄａプログラムを見ると改良すべき点がいくつかあるのに気付いた。

本稿では、上記の通信システムを実現したＡｄａのプログラムの改良について報告し、システム開発過程の中でのその位置付けについて述べる。

初版プログラムに対する考察

今回改良の対象とした通信システムは、上位から順に、アプリケーションレイヤ、バルク転送レイヤ、論理リンク制御レイヤの三層から構成される。

初版プログラムの作成では，与えられた問題（プロトコル）に対する仕様記述、設計、動作の確認までを一貫してＡｄａを用いて行うことが可能であり、またそれが有用である事を示した。しかしこれは試作品というべきものであり、プログラム構成や、プログラミング技法に対しては十分な検討が行われているとはいえなかった。従って、一個のプログラムとしてこれを見た場合、いくつかの問題点を見つけることができる。

１）情報の隠蔽が不完全

初版プログラムは、Ａｄａの有効な特徴の一つである情報隠蔽の機能を十分に活用していない。例えば、状態遷移表やイベント等の情報は全域的なデータで記述しており、with節によって、任意のパッケージから参照できる。これらの情報は出来るだけ局所化し、外部からのアクセスを不可能にしておくことが、プログラムの信頼性や保守性の面からも望ましい。

２）各レイヤの不明瞭な境界

初版プログラムは、バルク転送レイヤを規定したプロトコルのみに基づいて作成した。このため結果的に、機能の点でも、またデータ構造の点から見ても初版プログラムにおける各レイヤの境界に不明瞭な部分があった。

３）低効率

初版プログラムは実行時の効率が非常に悪かった。これは主に、一部のタスクがポーリングを行い、ビジーウェイトをしてＣＰＵ時間を浪費していた為である。タスク呼び出しの方向を変え、ビジーウェイトを行っているタスクを無くしたところ、実行速度を格段に向上させることができた。

４）タイマの不備

初版プログラムでは単なるインターフェース用のタスクであるはずのチャネルタスクでタイマを代用している。このため、プロトコルで規定しているタイマ機能を十分に実現できていない。

３．プログラムの改良

前章の考察を基に、以下のような方針でプログラムの改良を試みた。

まず第一にプロトコルの再検討を行った。アプリケーションレイヤとバルク転送レイヤとの不明瞭な境界と機能を明確化し分化させる為に、転送データ構造の階層化を徹底し、更に各レイヤのプロトコルを新たに規定、または変更した。

＊Ａｄａは米国政府（Ada Joint Program Office）の登録商標である。

Development of Distributed Systems using Ada (Ⅲ)
- Improvement of a Prototype System -
Masahiro Hori, Keijiro Araki, Kazuo Ushijima
Kyushu University

第二に、プロトコルの階層構造に従ってモジュール化を行った。すなわち、各レイヤを一つのパッケージに割り当て、更に機能毎にタスクを用いて細分化した。プログラムは図１にのような構成をしており、各レイヤに対応するパッケージは図２に示すように基本的には同じ構造をしている。即ち、状態遷移を管理する STATE_MANAGERタスク、入力を司るINPUT_MANAGER タスク、出力を担当するOUTPUT_MANAGERタスク、そしてバッファタスク、タイマタスクとから構成される。各タスクは、情報を局所化し、見せる必要のない情報をできるだけ隠蔽するように設計した。また、各レイヤのパッケージの間には、インターフェースとして一個のチャネルタスクを設けている。

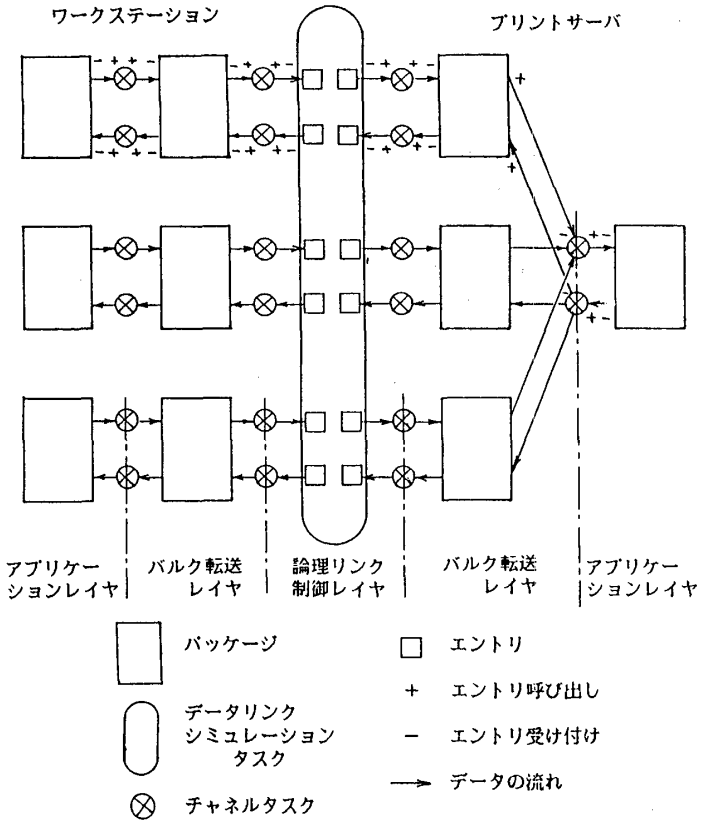

図１．システム構成

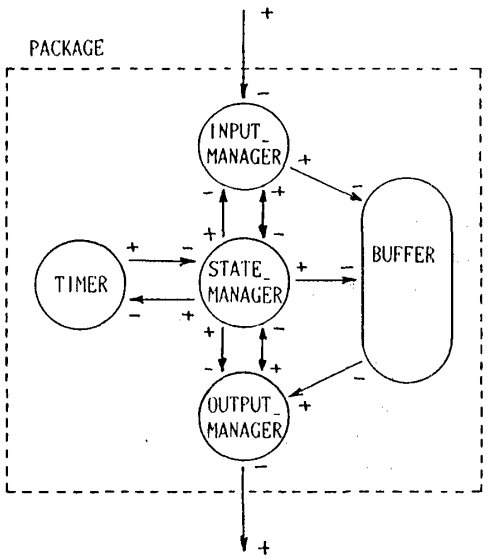

図２．各レイヤのパッケージの基本構造

第三に、複数のコネクションを一つのタスクが一括して管理するのではなく、各々のコネクションに対してそれを管理するタスクを一つづつ割り当てる方法をとった。この方法をタスク／コネクションと呼ぶ。タスク／コネクション構成は、一つのタスクが一つのコネクションにだけ注目すれば良いので、タスクの構造が簡単になり、記述が容易になる。また、プロトコルとの対応も素直になるという利点がある。

第四に、以上の様な改良を行った際、効率を低下させるような手法は使用しないように心がけた。

４．おわりに

ソフトウェアの開発は一般に次のような手順をたどるだろう。まず与えられた問題に対して頭の中でモデル化を行い、次にこれを具体化してシステム全体を設計する。この後、実際にプログラムを作成し、その動作を確認する、という手順である。

今回は、概念モデルの具体化からシステム設計、プログラム作成までを、一貫してＡｄａ言語を用いて行い、動作の確認にはそのままＡｄａ処理系を用いる方法を取った。これによって、ソフトウェアの設計・開発ツールとしてのＡｄａの有用性を確認できた。

またこの方法により、問題そのもののより深い理解が比較的早期に得られ、その結果、モデル化やシステム設計の改訂、改良に、多大な寄与が有り得る事も記しておかねばならない。

参考文献

１．荒木，牛島：通信システムのＡｄａによる統合的開発事例，情報処理，Vol.26,No.4,pp.299-309, 1985

２．CCITT: Recommendation X.25, 1984.

３．中山，荒木，牛島：Ｘ．２５パケットレベルプロトコルのＡｄａによる記述，本大会論文集，1986.

４．United States Department of Defence: Reference Manual for the Ada Programing Language, ANSI/MIL-STD-1815A, 1983.