ネットワークソフトウェアの設計法

高橋　薫、　白鳥則郎　　野口正一
（東北大学　電気通信研究所）

1．まえがき

ソフトウェアを含め、一般にシステムの開発（仕様化から保守まで）を複数人で行なう場合、その規模が大きくなるにつれて以下のような難点が生じてくる。(1)１人でシステムを開発、或いは同一組織内の少人数で開発する場合を除き、仕様書の記述に特定の形式的な言語を用いることは困難である。例えば、ネットワークソフトウェアの開発過程で協同開発者に特定の記述言語による表現を強要することは難しい。その為、自然言語で仕様を記述する場合が多い。(2)仕様が自然言語で表現されると、記述内容にあいまい性が生なう。(3)保守者にとって、設計者の意図（例えば、プログラムの内容）を理解することが困難になる。

前記の難点を解決する為、これまで種々の記述言語が考案されてきた。[(1)(2)(3)]

本論文では、仕様記述から論理設計、検証、インプリメンテーションまで一貫した開発過程の中で、前述の難点をとらえ、これを解決するための設計法と記述形式について考察する。特に、ネットワークソフトウェアの開発に効果的である設計法と記述形式を提案する。この設計法と記述形式の基本的な諸概念は、他のシステムを開発する場合にも有効である。

2．ネットワークソフトウェアの設計法

本章では、システム（ネットワークソフトウェア）のアーキテクチャの設計法とこれに対応した仕様記述形式の基本的な枠組みと諸概念を与える。次章でこの概念に基づく具体的な記述形式を設計する。

2.1 アーキテクチャの設計法

《アーキテクチャの設計手順》

<Step-1> ユーザ・インターフェース（図1）
(1)システムが提供する機能に応じた、ユーザとシステム間の入出力コマンドの規定
(2)入出力コマンドに対応したシステムの動作を後述する記述形式で表現

<Step-2> システムの論理構造（図2）
(1)情報処理部と通信制御部の分離
(2)分離したモジュール間にインターフェース部の設定

<Step-3> 階層化とモジュール化（図3）
(1)情報処理部と通信制御部の階層化とモジュール化
(2)階層に応じたモジュール化の機能表現
Protocol-i（モジュール）の記述形式：「有向グラフ」,「Common Primitive」,「Primitive-i」
ここで、「Primitive-i」はプロトコル階層iに応じたPrimitiveを示す。

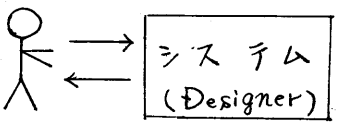

図１．ユーザ・インターフェース

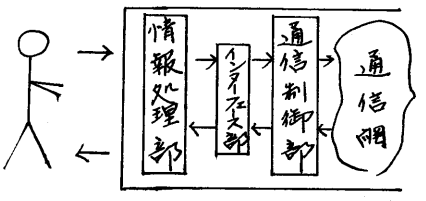

図２．システムの論理構造

2.2 記述形式の基本概念

設計及び保守を容易にするシステムの系統的な設計法[4]に基いた記述形式の主な特徴と記述形式間の相互関係を表1に示す。

本論文では特に、仕様記述に焦点を絞って議論を展開する。システム設計に伴なう前述の難点を解消するには、仕様記述形式として設計者及び保守者に分かりやすい表現法の導入が基本的で重要である。このような観点から『有向グラフ』と『Primitive』を基本とした仕様記述形式(NESDELと呼ぶ)を提案する。ここで『Primitive』は一種の言語であり、(1)「Common Primitive」と(2)「Module-dependent Primitive」から構成されている。「Common Primitive」は①自然言語と②どのモジュールにも共通な、動作を表すPrimitive-operation及びモジュールで用いる変数、入出力対象となる情報等を定義するPrimitive-definitionの集合，更に「Module-dependent Primitive」は①自然言語と②モジュールに依存したPrimitive-operation及びPrimitive-definitionの集合を用いた表現形式を表す。

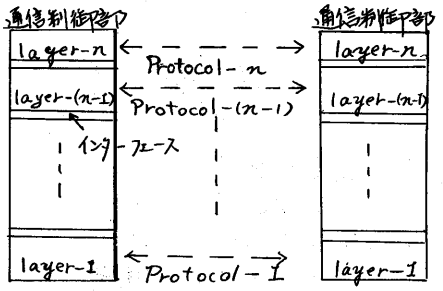

図3. 階層化とモジュール化

項目	仕様記述	論理設計	インプリメンテーション
適用範囲	・すべての設計者に共通	・設計者毎に異なって良い	・プログラマー毎に異なって良い
目的とする容易性	・仕様の厳密な規定 ・論理的な機能の理解	・記述 ・内容の理解と保守	・インプリメンテーション ・内容の理解と保守
	自然言語 検証 代数的仕様記述 NESDEL 記述形式(言語)による仕様記述 検証 EXPA[4]	COPDEL[4] 設計言語 検証 EXPA[4]	I-表現[4] インプリメント用表現

表1. 記述形式と特徴

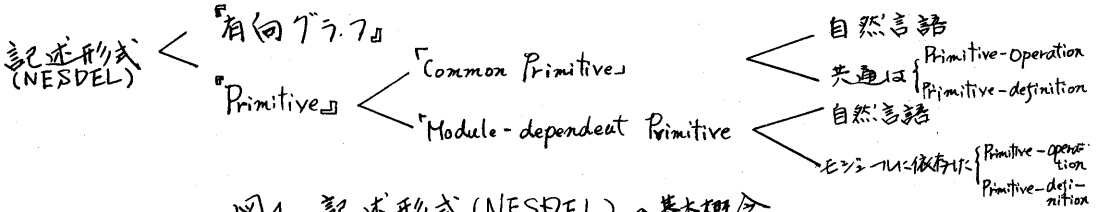

図4. 記述形式(NESDEL)の基本概念

NESDELの特徴は、(1)『有向グラフ』を基本とすることにより、システム全体の動作の理解を容易にし、(2)『Primitive』により細部を記述することである。表2に設計段階に応じた記述形式の相互関係を示す。尚、より厳密なネットワークソフトウェアの仕様表現法として代数的仕様記述法(表1)を想定しており、現在検討中であり、別途報告する予定である。

記述者	User	User Designer	Designer	Programmer
記述形式	自然言語	要求仕様記述言語	設計仕様記述言語	プログラミング言語
本論文での記述形式	—	「有向グラフ」と「Primitive」	—	—

表2. 設計段階に応じた記述形式

3. 仕様記述形式 NESDEL-α

本章では前述した基本概念に基いた各階層のモジュール（以下、PM と呼ぶ）の仕様記述形式NESDEL-α（αはNESDELのversionを示す）を与える。最初にNESDEL-αの重要な基礎となるPMの外部インターフェースのモデル化を行い、そして「有向グラフ」による記述形式と「Primitive」による記述形式を述べる。

3.1. PMのインターフェースのモデル化と「Common Primitives」

通信プロトコルの階層性から、各プロトコル階層のPMは、1)上位PMとの情報の入出力、2)下位PMとの情報の入出力、3)タイマーの起動とタイムアウトによってその外部actionを特徴化できる。ここで、上位PMとの間で交換する情報をlocal messages, 下位PM(通信チャネルと見なされる)との間で交換する情報をglobal messages と呼ぶことにする。

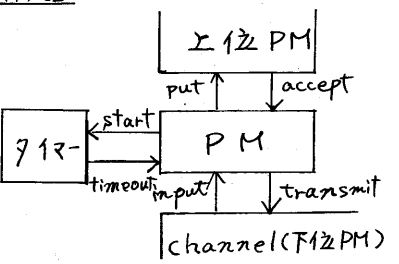

図5. PMのモデル化

これらのactionを記述する為に、初めに図4に示した基本概念に於ける「Common Primitive」に対応した各階層に共通な6つのprimitive-operationを導入する。

(i) transmit　(ii) input　(iii) put　(iv) accept　(v) start　(vi) timeout

(i)～(vi)はそれぞれ、global messageの送信、global messageの受信、local messageの出力、local messageの入力、タイマーの起動、タイムアウトを表す。

これらのprimitive operationはPMの記述に本質的であり、後述する「有向グラフ」による記述形式のベースとなる。図5にPMのモデル化と外部actionを記述するprimitive operationを示す。

3.2. 『有向グラフ』による記述形式

3.2.1 準備

有向グラフは、ノードの集合、アークの集合、アーク・ラベルの集合より成る。

ノードには次の3つの種類がある。

(1)事象待ちノード（以下、ew-nodeと呼ぶ）
PMが外部からの入力を待っている状態を表す。

(2)処理ノード（以下、p-nodeと呼ぶ）
PMへの個々の入力に対するPMの処理モジュールの対応を表す。

(3)初期/終結ノード（以下それぞれ、I-node, T-nodeと呼ぶ）
グラフ中で初期処理或いは終結処理を表す。

アークは上記のノード間の推移を表す。但し、各アークは図6の形をしており、aを親アーク、bを子アークと呼ぶ。

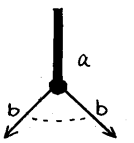

図6. アークの形式

アーク・ラベルはアークに添加され、親アークのラベルは推移に伴なうactionを表し、子アークのラベルはそのactionの対象内容を表す。

3.2.2. 記述形式

グラフの始点はI-nodeであり、そこからアークの連結で示されるようなノード間の推移が行なわれ、T-nodeへの到達でPM全体の動作が終結する。

ここではグラフの記述形式を最も典型的に表しているew-node, p-nodeから

述べ次にI-node, T-node を述べる。

(1) w-node (図7)

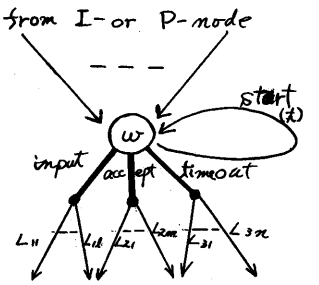

図7. w-node

w-node への入射アークは I- 或いは P- node からの出射アークであり、w-node からの出射アークは P- 或いは T-node への入射アークとなる。出射アークに許される action は、① input, ② accept, ③ timeout の3種類である。そしてこれらが親アークのラベルとなり、子アークのラベルはその action に伴う入力内容である。許される例外アークとして自身から自身へのアークがあり、それは start(1) のラベルを持つ。

ここで1はタイマー名のリストであり、このノードへの推移があった時に1で示されたタイマーを起動することを示す。

各action (start を除いて) は排他的に1つだけが生起するものとし、即ちグラフ中，只1つのパスが選択され次のノードへ推移する。input, accept, timeout, start のうち記述の上で必要のないものがあればそのアークを排除して良い。子アークのラベル，即ち，入力事象内容は「Primitive」による記述と対応がとられる。

(2) P-node (図8)

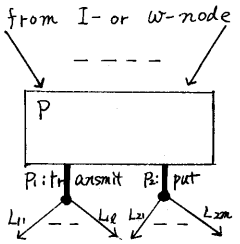

図8 P-node

PMの内部処理を表すノードである。P-node への入射アークは I- 或いは w-node からの出射アークであり、P-node からの出射アークは w- 或いは T-node への入射アークとなる。出射アークに許される action は、① transmit, ② put の2種類があり、次の形式のラベルを有する： <label> ii = <predicate> : <action>

ここで、<predicate> は「整変数と整変数の比較」，或いは「整変数と整定数との比較」を表す条件式及び論理変数を論理演算子によって組合せたものである。つまり、指定された predicate が真の時に対応する action を実行し、次のノードへ推移することを表す。各親アークは子アークを持ち、それらは次の形式のラベルを有する：

<label> ii = <predicate> : <メッセージ内容>

このラベルは、対応する predicate が真の時に、それに対応するメッセージ内容が親アークに付された action の具体的内容として与えられることを示す。全体的には、親アークに付された action の実行条件はそれに対応した predicate から決まり、更にその action の具体的内容の決定は子アークに付された predicate によって行なわれることを意味する。w-node と同様，各action は排他的に1つだけが生起するものとし、只1つのパスが選択され次のノードへ推移する。記述の上で必要のない親アーク、子アークがあれば排除して良い。又、常に真であるような predicate は省略して良い。

親アーク及び子アークに付された predicate に用いた変数及び子アーク・ラベルのメッセージ内容，そしてノード・ラベルは「Primitive」による記述と対応がとられる。

(3) I-node (図9)

グラフ中に只1つ決まり、初期処理を表す。アークは p- 或いは w-node への出射アークだけであり、アークの目的ノードが p-node の場合には，w-node

に於いて定めた出射アークとそのラベルの規則に、w-nodeの場合には、p-nodeに於いて定めた出射アークとそのラベルの規則に従う。但し、特別な場合として無条件推移を表すラベルなしのアークを許す。

ノードにはラベルを付し、「Primitive」による記述と対応をとる。

(4) T-node (図10)

P-或いはw-nodeからの入射アーク1だけから成り、終結処理を表す。T-nodeはグラフ中に複数個存在して良い。

ノードにはラベルを付し、「Primitive」による記述と対応をとる。

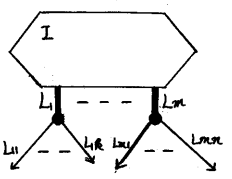

図9. I-node

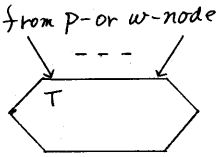

図10, T-node

3.3. 『Primitive』による記述形式

本節では、『有向グラフ』中の各要素の詳細な意味を定義する為に、図5中の「Common Primitive」に対応する、全てのPMに共通な定義Primitive (primitive-definition)と「Module-dependent primitive」を導入する。表3にこれらのprimitive definitionとその構文を与える。

同表に於いては次のa)～h)のprimitiveを定義している。

a) PM名, b) local messageの入出力対象となる上位PM名とその入出力メッセージの種類, c) global messageの入出力対象となる下位PM名とその入出力メッセージの種類, d) 使用するタイマー名とそのタイムアウト時間値(ミリ秒), e) 有向グラフ中に現れるpredicateで用いる変数とその型(int or bool), f) タイマー, 変数, メッセージのそれぞれについて、どういう役割や意味を持つかを自然言語で記述, g) PMに依存したprimitive: ここで定義したprimitiveはh)に於ける処理定義の中で用いることが許され、処理内容を自然言語の代わりに形式的に与えることを可能とする, h) 有向グラフ中に現れるP-, I-, T-nodeの処理内容を定義する。node labelはグラフ中のノードに付されるラベルとの対応である。bodyは、自然言語或いはg)で定義したPM依存primitiveを用いて処理内容を記

定義内容	Primitive definition
a) PM名	protocol machine (PM名)
b) local messages	local messages (上位PM名): INPUT={入力メッセージの集合} | OUTPUT={出力 〃 }
c) global messages	global messages (下位PM名): INPUT={入力メッセージの集合} | OUTPUT={出力 〃 }
d) タイマー	timers (タイマー名1=time1 | --- | タイマー名n=timen)
e) 変数	var (変数名1:型 | --- | 変数名n:型)
f) 意味	meaning タイマー名1 : 自然言語による意味記述 | S タイマー名n : 〃 | 変数名1 : 〃 | S 変数名m : 〃 | local messages : 〃 | global messages : 〃
g) PM依存 Primilive	Prim プリミティブ1 : 自然言語による意味記述 | S プリミティブk : 〃
h) 処理	Proc node label 1 : body | S node label l : body

表3. Primitive definition

述する部分である。　一般に、PMがプロトコル階層のどの位置にあるかにより処理内容を記述するprimitiveは異なる。従ってNESDEL-αでは、PM依存primitiveをPMの仕様記述者が与え、そのもとで処理内容を厳密に定義するような方針を採用した。

3.4. NESDEL-αによる仕様記述例

例題として、下のプロトコルを実現するPMの仕様記述例を図11に示す。

《メッセージの送信規約》

i) 上位階層(USER)からメッセージを入力し、相手PMへ送信する。　ii)応答を受信する。肯定応答であればメッセージが正しく送信完了したとして"成功"を上位へ通知し、否定応答或いは無応答(3秒)であれば、前のメッセージを再送する。再送を3回行なっても回復しない時には、メッセージ転送が失敗したとして"失敗"を上位へ通知する。　iii) i)～ii)をメッセージが発生する毎に繰り返す。

```
define
   protocol machine(PM);
   local messages(USER):INPUT={message}|
                        OUTPUT={success,
                                failure}  ;
   global messages(CHANNEL):INPUT={ack,nak}|
                            OUTPUT={message};
   timers(timer=3000);
   var   (r:integer);
   meaning
     timer : for message timeout detection |
     r     : retransmission counter        |
     local messages : message=data message,
                      success=successful
                              transmission,
                      failure=unsuccessful
                              transmission |
     global messages : message=same as
                               local,
                       ack=acknowledgement,
                       nak=negative ack ;
   prim a:=b+c  : ordinary assignment
                  statement|
        stop(t) : stop a timer.
                  t is the timer-name.   ;
   proc I  : initialization |
        P1 : r:=0           |
        P2 : stop(timer)    |
        P3 : stop(timer);r:=r+1
end define
```

①: PM名、メッセージ、タイマー、変数の記述

②: 意味記述

③: PM依存Primitiveの記述

④: プログラムの処理内容の記述

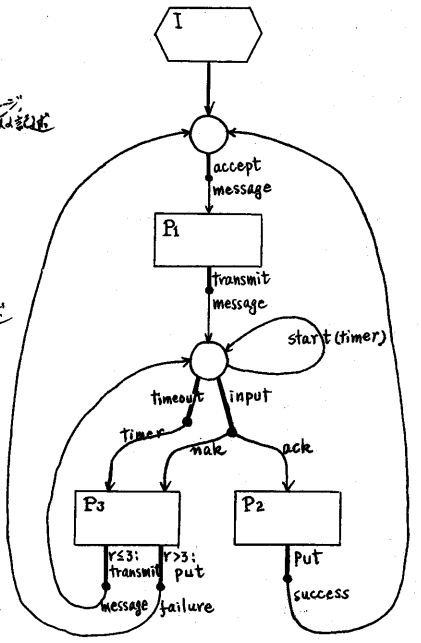

図11. NESDEL-αによる仕様記述例

4. むすび

ネットワークソフトウェアの設計法と記述形式を与えた。特に、ソフトウェアの設計者に対して共通な記述形式としてNESDEL-αを提案し、その有効性をデータリンク・コントロール・レベルのソフトウェアを例題として記述することによって示した。

《参考文献》

(1) A. Rockström and R. Saracco: "SDL-CCITT Specification and Description Language", IEEE Trans. Comm. (June 1982)　(2) -: "Current Trends in Programming Methodology Vol. I", Prentice-Hall (1977)　(3) P. M. Merlin: "A Methodology for the Design and Implementation of Communication Protocols", IEEE Trans. Commun., COM-24, 6, pp. 614-621 (June 1976)　(4) N. Shiratori, T. Gohora and S. Noguchi: "A New Design Language for Communication Protocols and a Systematic Design Method of Communication Systems", Proc. of ICCC, pp. 403-412 (1982)