ストラクチャード・プログラミング導入と効果

松下 温　山崎晴明　高橋恒治　吉田 勇　伊藤良雄
（沖電気工業株式会社）

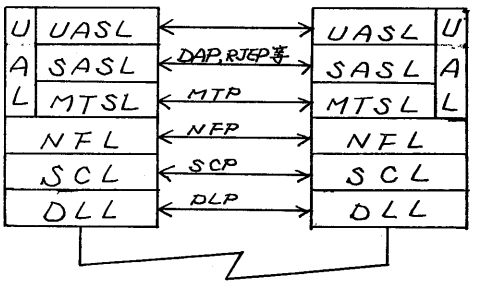
1. はじめに　ソフトウェアの開発，特にプログラミングに関して各種のソフトウェア工学的考察がなされてきている。各ソフトウェア工学的手法の適用結果が報告されているが，適用対象システムはOSの管理下で動作する各種ユーティリティや実時間性を必要としないバッチ処理等が主であった。そのため各種ソフトウェア工学的手法の主たる目的は，プログラム製造工数の削減，プログラム開発期間の短縮，メンテナンス作業の負荷軽減であったと思われる。一方，スループットについては多少犠牲にされてきたと言えよう。オンラインリアルタイム処理を実現するプログラムでは，スループットの低下は無視できない。従ってこの様なプログラムに各種ソフトウェア工学的手法を適用することが可能であるか否かを評価したものは少ない。

そこで本稿では，実時間的に厳しいデータ通信処理の標準化を目的とするNA（ネットワークアーキテクチャ）にソフトウェア工学的手法を適用してその効果を報告するものである。NAには，非分散解放型ネットワーク体系（DONA；Decentralized Open Network Architecture）[1][2][3][4]を採用した。またNA製品実現ハードウェアには，当社のミニコンピュータOKITAC－50シリーズを採用した。

次章以降で適用システムDONAの概略，採用したソフトウェア工学的アプローチ，DONAのインプリメント状況，ソフトウェア工学的手法適用の効果について述べる。

2. DONAの概略　ここではまずDONAの論理的なプロトコル階層について述べ，そのアーキテクチャに従って構築可能なシステムの構成例を述べる。

DONAのプロトコル階層は図.1に示すごとくDLL，SCL，NFL，UALの4階層からなっている。DLLは隣接ノード間でのデータ転送を実行するレイヤである。SCLはその上位層であるNFLに対してデータグラム型のパケット交換機能を提供するレイヤである。NFLはいわゆるホストーホストプロトコルを実行する階層である。UALはアプリケーションオリエンテッドな機能を遂行する階層であり，さらに3つのサブレイヤにより構成されている。MTSLはUALで実行される各プロトコルに共通に必要な機能，すなわちプロセス相互間のメッセージ通信を実行するサブレイヤである。SASLは汎用的な標準プロトコルを実行するサブレイヤである。UAS



DLL; Data Link Control Layer
SCL; Switching Control Layer
NFL; Network Facility Layer
UAL; User/Application Layer
MTSL; Message Application Sub-Layer
SASL; Standard Application Sub-Layer
UASL; User/Application Sub-Layer

図.1　DONAのプロトコル階層

レは各システム毎の業務処理を実行するサブレイヤである。

DONAは専用線を用いて構築することも，公衆パケット網を用いて構築することも両方可能となっている[3]。公衆パケット網を使用する場合には網とのインタフェースはNFLで扱っており，この場合SCLはスキップされる。またNFLではその上位レイヤであるUALに対してバーチャルサーキット（VC）サービスとトランザクション処理のように比較的短いデータ転送に適したレターグラム（LG）サービスとを提供しており，アプリケーションのトラヒック特性によって両サービスを任意に使い分けることにより，各アプリケーションに即した効率のよい通信を可能としている。このように専用線を用いて構築するSCLサブネットと公衆パケット網の両者をその通信サブシステムとして適用できること，およびアプリケーションに対してVC，LG両サービスを提供していることがDONAの大きな特長である。

DONAはミニコンによる分散処理システムを主として想定している。今回のインプリメントもミニコンOKITAC-System 50を用いて行った。前述のプロトコル階層とハードウェア構成との関係は独立であり，システムの規模や想定するスループットに応じて任意の構成をとることが可能である。即ち，SCLサブネットを物理的にも分離して構成することも可能であるし，SCL，NFLを1CPUに入れてFEP的に使用することも，又すべてのレイヤを1CPUで実現することも可能となっている。これらの柔軟性は特にミニコンネットワークの場合には必要であると考える。これら各ハードウェア構成のパターンを図示すると図.2のようになる。O-50は，OSとしてDOSとMOSを提供している。今回インプリメントした各プログラムは両者の下で動くことが可能である。

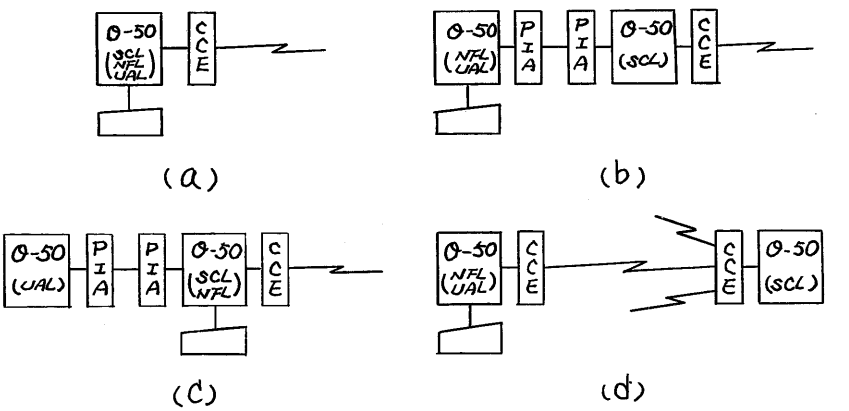


図.2　ハードウェア構成例

3. 適用したソフトウェア工学的アプローチ

3.1　記述言語の選択

DONAの各レイヤを記述するに際してプログラム記述言語の選定が議論された。プログラム製造やデバックに要するマンパワーの削減を目的とするため，高水準の言語である必要があり，また後述するストラクナヤードプログラミング（SP）が容易に行なえる言語であるという前提条件のもとに，以下のような観点から各種言語の比較，検討が行なわれた。

(1) ネットワークシステムの記述言語であるため，スループットの低下を無視するわけにはいかない。従ってある程度パフォーマンスに対する配慮を可能

とするような言語が必要となる。

(2) プログラミング，デバック等を効率よく行なうためにはプログラムを適切なサイズのモジュールに分割して，そのモジュール毎に部分的なコンパイルが可能となることが必須条件となる。

(3) ネットワークシステムに特有の処理として，ルーティングテーブルの索引，各種アドレスの変換，対応づけ等のテーブル操作，ヘッダ操作が頻繁に行なわれることが予想される。このようなデータ操作が柔軟に，かつ効率よく行える言語であることが条件となる。

(4) (1)と関連するが，パフォーマンスの良いモジュールを作成するために，絶対番地でのメモリ操作，直接にレジスタへのアクセス等が記述できる言語であることが望ましい。

記述言語の選定に当り，高水準言語を基本とする前提条件よりマシン語等の低レベル言語は対象外とした。次に，条件(1)，(3)からFORTRAN，COBOL等の比較的広汎に普及はしているが，パフォーマンス上で余り効率的でないと考えられる言語も対象からはずした。また条件(3)からALGOL 60等の言語は不適切と判断された[5]。従ってNAシステムの記述をするために結論づけられた方法は，PL/1 like の言語（SPL）とアセンブラ言語との併用であった。つまり，プログラムはSPL（System Programming Language）[6]をベースとして記述され，特に高パフォーマンスが必要な部分あるいはSPLでは記述が困難な部分はアセンブラ言語を用いて記述するという形式である。SPLは当社で開発され，現在各種のシステム記述用として用いられている言語である。

3.2 SPL概要

SPLはシステムプログラムの記述を目的に当社で設計，製造されたもので，昭和52年2月より実際の各種システム記述に用いられている。SPLはPL/1をベースとして設計されたもので，システム記述のための機能および科学計算用の機能が付加されている。図.3にSPLプログラムの構造を示す。尚SPLは，U-50の各シリーズ上でDOSのもとで動作する。

```
<SPLプログラム>::=<モジュールリスト>
<モジュール>::=<モジュール文><プロシジャリスト><モジュールEND文>
<モジュール文>::=<モジュール名>：MODULE：
<モジュールEND文>::=END[<モジュール名>]：
<プロシジャ>::=<プロシジャ文><文リスト><プロシジャEND文>
<プロシジャ文>::=<プロシジャ名>：{PROCEDURE|PROC}
                    [<オプション>][(<パラメータコンマリスト>)]：
<オプション>::=MAIN|TASK|RENT|COMMON|SUB
<プロシジャEND文>::=END[<プロシジャ名>]
<パラメータ>::=<識別名>
<プロシジャ名>::=<識別名>
```

図.3 SPLのプログラム構造

一方SPLの扱うデータは，それが置かれる記憶域のちがいにより次の3種に分類される。(1)スタティックデータ（記憶域がプログラムエリア内に確保される），(2)ベースドデータ（動的にプログラムによって確保される），(3)物理データ（汎用レジスタに名前付けしたもの） またSPLでは，種々の異なる属性を持つデータを扱うことができる。図.4に，SPLの扱うデータ属性の一覧を示す。さらにSPLでは，(1)単純データ（1つのデータで構成される），(2)配列データ（同一属性のデータを順序づけたものである），(3)構造データ（個々のデータ要素をまとめた複合体として定義される）の各データ構造を扱うことができる。図.5に，

これらのデータ構造のシンタックスを示す。

尚詳細は述べないが，このような構造化データを用いてルーティングテーブル，各種ポートの対応テーブル等の記述がなされた。次節ではこのSPLを用いてどのようにストラクチャードプログラミング技術が採用されたかを述べる。

INTEGER	算術データ	
REAL		
CHARACTER	ストリングデータ	
BIT		
POINTER	制御データ	
LABEL		
EXTERNAL ENTITY		

図.4 SPLの扱うデータ属性

3.3 ストラクチャードプログラミングの採用

プログラムの製造に当っては，設計，デバック等の容易さ，モジュール分割のし易さといった観点から，ストラクチャードプログラミング技法を採用することとなった[7][8][9]。

SPの特徴の1つに処理のout-side-inに基づく記述がある。このためSPLのプロシジャに階層構造という概念が入れられることになり，これをどのような手法で表現すればよいか，また通常用いているようなフローチャートは不要であると判断してよいのか等の問題を発生させることとなった。このような新手法の導入に際しては，その表現手段としてもまた思いきった新しい手法を採用するのが適切であると考えられる。しかしフローチャートには視覚に訴える力は大きく，現段階では多数のプログラマがアルゴリズムを考えるときに頭に思い浮べるものは従来のフローチャートによっているとの判断から，SPLのプロシジャ記述に当りその主たる表現手段としては，旧来のフローチャート記法を用いることとなった。従ってSPの特徴であるout-side-inという概念をフローチャート上でどのように表現するかということが大きな問題となった。そこでデータ構造の記述に関してレベルという概念が必要であったと同様に，プロシジャの各ブロックに対してレベルの概念を定義することとした。このレベルとは表記上のレベルでありまた処理の詳細度を示すレベルともなっている。以下にレベルの定義を示す。

<単純データ> ::= <変数名> [<データ属性>]
<配列データ> ::= <配列名> (<添字>) [<データ属性>]
<添字> ::= <正整数>
<構造データ> ::= 1 <構造名> [<データ属性>] <構造要素コンマリスト>
<構造要素> ::= <レベル> {<単純データ> | <配列データ> | {* SPACE <精度>}}
<レベル> ::= <正整数>

図.5 SPLの扱うデータ構造

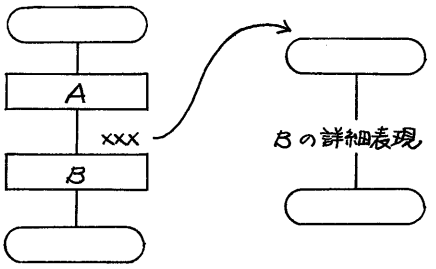
(1) Concaterationで記述された処理はすべて同一レベルとする。
(2) Selection文 "IF L then A else B" において，処理A，BはSelection文自身より1つだけ低いレベルとする。
(3) Repetition文 "while L do A" において，処理Aは，Repetition文自体より1つだけ低いレベルとする。
(4) 最低レベルの処理とは，SPLのステートメントのことである。

図.6にこのレベルの概念を例示する。

次に、処理のフローチャート記述に当ってこのレベルの概念が明確にしかも理解のし易い形式で表わされるように種々の規約を設けた。主な規約を以下に示す。

(1) プログラム基本構造を，Concatination，Selection（Case文を含む），Repetition文の3種に限定する。

(2) フローチャート上で，あるレベルの記述とそれを詳細化した下レベルの記述との対応を次のように定義する。即ちその記述が non-terminal 記述（SPL ステートメントでない）ことを示すためオープンサブルーチン記述をとり，その refinement 記述とは，座標によりリンク付けを行なう。



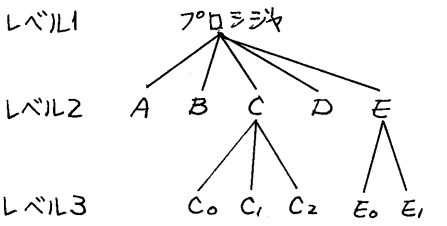
(3) Terminal Box を除き，各処理 Box への入出力は1つでなければならない。このことは，out-side-in refinement の当然の帰結でもある。

(4) Decision Box は SPL ステートメントに対応するものでなければいけない。これにより各 Box の refinement は独立に行ない得ることが保証される。

(5) 1つのフローチャート表記では，同じレベルの記述を行なうことを原則とする。但し，Repetition, Selection の場合には便宜上1レベル低い処理も記述することを可能とする。

(6) 処理ボックスの refinement は，SPL ステートメントに至り終了する。但し，アセンブラ記述のプロシジャはこの限りではない。

レベル表現によるプログラム構造



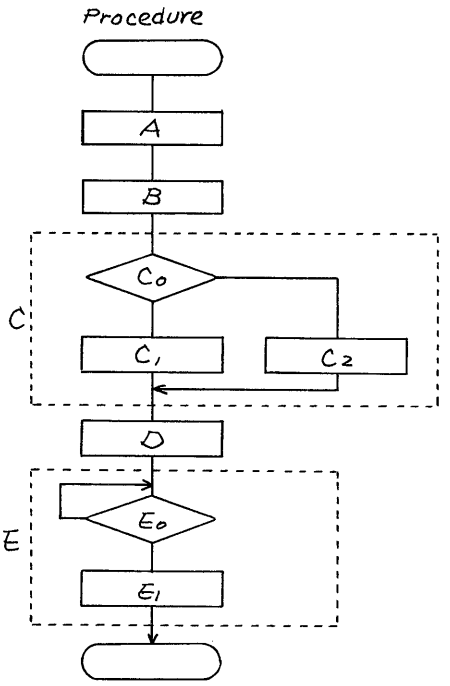


図.6 レベルの概念(例)

このような規約のもとに，refinement をくり返し適用しつつ SPL によるプログラム記述を行なった。次章では，実際にこの手法を用いてネットワークシステムを記述してゆく上でのモジュール構成について述べる。

4. DONA のインプリメント　2章で述べた DONA の各レイヤのうち，DLL，SCL，NFL および保守機能を実行する SASL の4つを今回インプリメントした。各レイヤの独立性をはかるために機能実現のプログラムもそれぞれ独立のモジュールとした。その各モジュールに対して SP 技法を適用した。各プログラムのモジュール構成と流れを図.7に示す。

次に各プログラムの主な機能を示す。

UAL のサブレイヤ SASL に位置する保守運転プログラムは，NA システムを円滑かつ迅速，有効に遂行するための補助機能を実行するもので，保守コマンド投入による保守機能の実行や内部要因に基づく自律メッセージの出力等を行

なう。

ＮＦＬの機能は大きく分けて、受信プログラムとＮＦＣＰ（Network Facility Control & Processing Program）の２つにより実行される。受信プログラムは他レイヤからの入力を受付け、定められたフォーマットに従ってＮＦＣＰに処理依頼する。他レイヤとの物理的なインタフェースとしては、回線、他ＣＰＵとのチャネル結合等が考えられ、これらは受信プログラムですべて吸収している。一方ＮＦＣＰでは、要求受付プログラムで受信プログラムから依頼された要求内容を分析し、その結果によって各実行プログラムを起動している。この実行プログラムは専用線を用いた場合のＶＣ，ＬＧ（Letter Gram）各機能を実行するもの，ＤＤＸパケット交換網を用いた場合の同様の機能を実行するもの，異種ネットワークを接続した場合の交換機能（ゲートウェイ機能）を実行するもの，保守機能を実行するもの，の４つから構成されている。これらの実行プログラム群はいつでもすべて実装させる必要はなく，実現しようとするノードの機能に応じて必要なモジュールのみ実装させればよい。

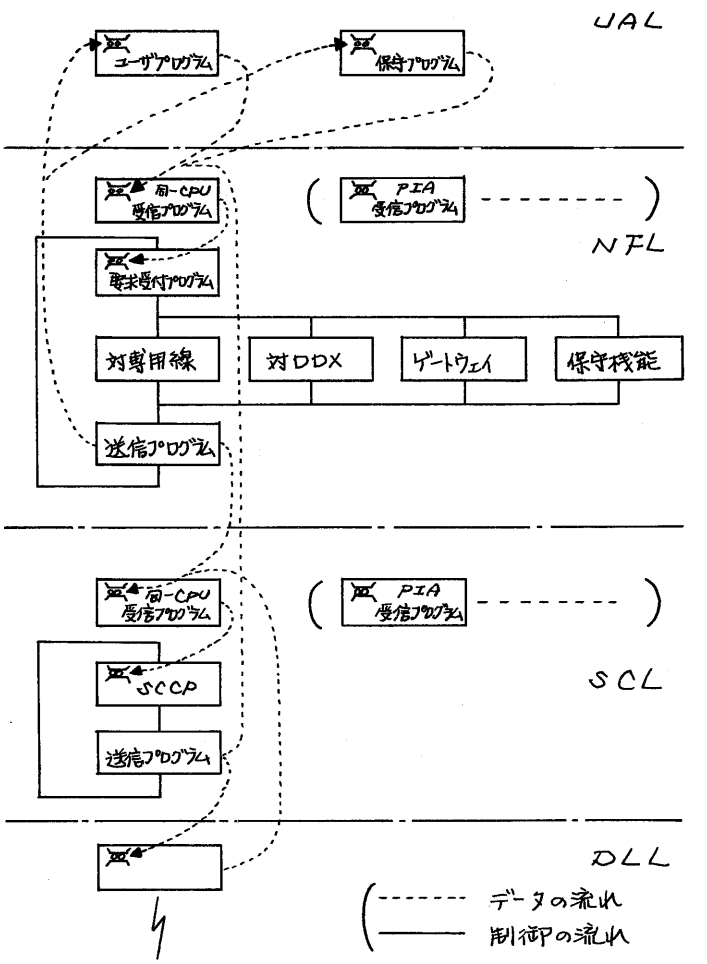


図.7　プログラム.モジュール構成図

ＳＣＬの機能は受信プログラム，ＳＣＣＰおよび送信プログラムの３つにより実現される。受信プログラムはＮＦＬのそれとほぼ同等の機能を実現する。ＳＣＣＰはＳＣＬのメインプログラムであり，ネットワークのルーティング処理，特にＤＤＮＡの分散型ネットワークを効率よく，信頼性の高いルーティング処理を実現する他，各種のテスト機能やトラヒック等の統計情報の収集の機能を実現している。

ＤＬＬは、データ伝送手順実現部で，ＨＤＬＣ手順をサポートしている。

今回インプリメントした部分は、対網側のアプリケーションインディペンデントな基本部で，ＮＡシステム構築時の網側の標準アクセスメソードとなる部分である。この基本部の上にＵＡＬとして従来の一般アプリケーションが構築されることになる。この場合，一般アプリケーションプログラムは，網側の煩しいネットワーク制御から一切解放され，本来の業務処理に専念することができる。

5. 高水準言語の適用とSP技法導入による効果　本章では高水準言語とSPといったソフトウェア工学的な手法をネットワークアーキテクチャという比較的システム記述に関係が深い分野に適用した場合の効果について述べる。比較に当っては、同システムとほぼ同規模（機能的に）と考えられるシステムを対象に選び、プログラム開発工数，保守性，スループット等を評価要因として，比較を行なった。比較対象として選んだシステムは，通常のstore-and-forward型のメッセージ交換システムで，O-50モデル40のCPU，LP，PTR，ET等の周辺装置，最大メッセージ長128バイト，総ステップ数24kですべてアセンブラ言語によって記述されている。勿論，比較する際にひとつはNAに基づくレイヤ構成を想定したシステムであるのに対し，他方はそのような標準化等に対する考慮を行なっていないシステムであること，また両方のシステム作成に関与したプログラマの資質の差等は考慮していないこと等の理由により単純な比較は困難と思われるが，全く同一のシステムを異なる2つの手法で製造することは事実上不可能であるため，あえてこのシステムを比較対象に選んだ。

また，プログラムの評価要因には種々様々なものがあり統一的な見解はないと思われるが，本稿では比較的一般的と考えられる下記の要因を選定し，両プログラムシステムの比較を行なった。

(1) 完全性チェック　プロシジャもしくはその細分であるプロシジャブロックに対して，仕様漏れあるいはその手続きでは対処できない入力パターンが存在するか否かのチェックは，プログラムのデバック，仕様変更等に際して重要な事となる。従って，このチェックの容易性を比較要因の一つに選んだ。

(2) プログラムの理解のし易さ　プログラムの保守あるいはプログラム製造の作業分担といった意味からもアルゴリズムの明解性，残されたドキュメントの理解し易さは重要である。新しいアプローチがこのような要因に対してどのような効果をもたらしたかを調査するのは充分意義がある。

(3) デバックの難易　プログラム製造に要する工数，期間の短縮という観点からはデバックの容易なアルゴリズムであることが必須となる。勿論これは(1)および(2)と独立には論じ得ないが，後述するようにバグの発生件数およびデバックに要した期間という形で定量的に論ずることができるため，この比較要因を付け加えた。

(4) プログラム変更の難易　バグの発生あるいは仕様変更等により，プログラム変更を行なう場合どのモジュールもしくはプロシジャに対し，どの様な修正を加えればよいかが，迅速かつ適確に理解できるものであることが望まれる。

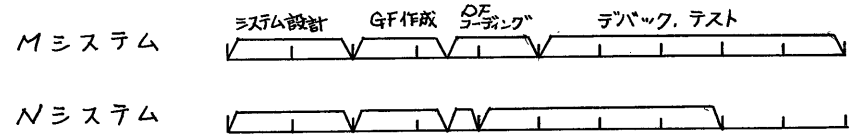
(5) プログラム実行時のスループット　両システムはともに実時間性が必要とされ，またユーザに対する応答性も問題となるため，実行時のパフォーマンスを無視する訳にはゆかない。新しく採用したソフトウェア工学的手法がどの位のパフォーマンスの低下をひき起したかを考察するため，この要因を付加した。

(6) プログラムの冗長度　高水準言語の使用およびSP技法の採用は，ともに冗長度をもった実行モジュールを作り出すと考えられる。いちがいに比べることはできないが，両システムのプログラム容量の差によって，新手法の持つデメリットを定量的に推測することを試みる。

(7) プログラムの作成効率　上記(1), (2), (3)の総合的な結果として，プログラム設計，製造，デバックに到るまでの開発期間，工数の評価を行う。

5.1 両システムにおける測定データ　高水準言語およびSP技法を適用したNAシステム（Nシステムと呼ぶ）と旧来の方式で作成したメッセージ交換システム（Mシステムと呼ぶ）の各種測定データを以下に示す。

a. 開発期間および作業スケジュール



b. バグ発生件数比　Nシステムを1とした場合のバグ発生件数比を表.1に示す。

c. メモリ量とスループット　OS部分およびバッファを除く両システムのメモリ量比，および128バイトのデータ（1メッセージまたは1パケット）の送信，受信に要するダイナミックステップ数比を表.2に示す。但し，送信および受信の比率はそれぞれ1/2づつとした。

表.1　バグ発生件数比

	Nシステム	Mシステム
論理ミス	1	2
インタフェースミス	1	5
修正ミス	1	3

表.2　メモリ量とスループット

	Nシステム	Mシステム
ダイナミックステップ数比	1	0.77
メモリ量比	1	0.80

5.2 両システムの比較　バグ発生件数比のうち，論理ミス，インタフェースミスは前述の評価要因(1), (2), (3)の総合的結果とみなすことができる。測定データから，明らかに新手法の効果があったとみなすことができる。従って高水準言語とSP技法の使用は，プログラムの完全性をチェックするために適しており，またアルゴリズムの明解性もあり，さらに5.1節a項よりデバックも行い易い手法であると結論づけることができよう。

プログラム変更の難易に対しても，表.1から新手法の効果があったと考えることができる。

パフォーマンスについては，表.2から高水準言語とSP技法を使用することにより，スループットおよびメモリ量とも約1.3倍程度大きくなり，新手法導入によるデメリットも見逃すことはできない。従って高度のパフォーマンスを要求されるようなシステムやメモリ容量がネックとなるシステム（NAシステムがそのようなシステムであるか否かの議論は別にして）では，高水準言語やSP技法の導入には細心の配慮が必要である。がしかし，一般的には，開発期間の短縮に対しては新手法はかなりの効果があったと考えられる。表.3に各比較要因をまとめて示す。

6. おわりに　DDNA仕様のインプリメントにおいて採用したSPLとSP技法は一応その目的を達したと思われる。高水準言語SPLは，製造工数の削減，開発期間の短縮に対して大きな力を発揮した。しかし，メモリ効率，スループ

ットの面においては，ある程度の低下は予想されたが期待した値よりいく分悪いものであった。しかしながら，全体としての総合評価は導入前のものより導入後の方が優位にあると考える。

表.3 各種比較要因による総合評価

	Mシステム	Nシステム
完全性チェック	×	○
プログラムの理解のし易さ	△	○
デバックの難易	△	○
プログラム変更の難易	△	○
実行時のスループット	○	△
プログラムの冗長度	○	△
作成効率	×	○

今後の課題としては，各レイヤの持つプロトコルをいかに効率よく記述するか，どのような言語がNA表現に向いているのかを検討する必要がある。

さらに今後のソフトウェア工学は，人的効率化つまり工数削減，期間短縮，メンテ費用削減は勿論のこと，その上に立っていかに能力的効率化をはかっていくかということがテーマとなるのではないかと考える。

参考文献

[1] 「沖分散解放型ネットワーク・アーキテクチャ：DONAの設計思想」松下，他 情報処理学会第18回全国大会 52年10月

[2] 「DGとVCの両パケット交換網に適するコンピュータ・ネットワーク・アーキテクチャ：DONA」 松下，他 情報処理学会コンピュータネットワーク研究会

[3] 「An Overall Network Architecture Suitable for Implementation with either Datagram or Virtual Circuits Facitities」 Y.Matsushita, et.al. Computer Communication Review Vol.8, No.3, July 1978

[4] 「An Effective Utilization Method of Public PSN in DONA」 N.Sugiura, Y.Matsushita, et.al. ICCC'78

[5] 「ソフトウェア開発技術に関する調査報告」 電子技術総合研究所 53年3月

[6] 「SPL概説書」 沖電気工業(株)

[7] 「ストラクチャードプログラミングの適用と考察」 松下，丹下 第56回情報科学研究会 51年1月 慶応大学 情報科学研究所

[8] 「Top-down structured programming techniques」 C.L.McGoman, John R, Kelly PETROCELLI/CHARTER

[9] 「Structured Programming」 Dahl, O-J, Dijkstra, E.W. Hoare, C.A.R. Academic Press London, 1972