解　説

3.3 通信ソフトウェアを対象としたCASEの現状と動向†

市　川　晴　久††

1. はじめに

通信ソフトウェアとは，通信端末，通信制御装置，交換機などいわゆる通信装置のソフトウェアを指すが，内容的な特質からすれば，並列処理に関わる広範囲なソフトウェアとみることができる．ただし，アクションやイベントと呼ばれる単位現象のシーケンス制御が設計の中心となる点で，計算を主体とする情報処理ソフトウェアとは区別される．

CASE の一般的現状はマーケティングレポートや関係者の発言など非公開資料に述べられることが多い．著者が見聞きする範囲では，「将来への期待は大きいが現時点での CASE ツールの適用についてはユーザはかなり懐疑的」というのが現状と言っていい．通信ソフトウェア向け CASE の場合には，対象を限定することにより高度な支援が可能となると想定されるが，現状での CASE ツール適用状況は特筆するほど進んでいるとは言い難い．

現状の通信ソフトウェア向け CASE を特徴づけているのは，形式仕様記述及びプロトコル設計の支援である．この10年ほどの間にこれらの技術はかなり成熟し，形式仕様記述言語の標準化(SDL, Estelle, LOTOS) やプロトコル設計支援システムの開発として結実してきている[BOC 87a, RUD 88, LIU 89]．一方，OSI プロトコルの標準化，ISDN 網の展開，コンピュータ技術の発展などがめざましい．通信ソフトウェア向け CASE 技術は，新しい環境条件から生まれるニーズを考慮し，実用的な支援技術に飛躍しつつある．あるいは，飛躍させるべき状況にある．

CASE に対するユーザのニーズを考える上で，Rudin の考察[RUD 88] が参考になる．プロトコル開発支援のうち，プロトコル製品検証に対するニーズが他に比して高い．Rudin はその理由を以下のように分析している．

● 心理的な要因：製品出荷前に生ずる，製品に対する不安．

● 完成度に対する測度：検査内容と結果の記録は，製品に対する自信とエラーに対する苦情への防御材料となる．

●「本物」の検査：製品検証では，完成・総合された設計結果であるインプリメンテーションを検査する．そこで得られる情報は，仕様検証などで得られる情報よりも，製品に関する直接的で確実な情報である．

● プロトコル製品検証では，製品を作る作業のやり方を変える必要はない．

一方，仕様からのインプリメンテーション生成支援は，製品を生成することでユーザの最終目的を達成する点では魅力的であるが，製品開発手順の変更を要する場合が多く，適用障壁が高い．仕様検証などの設計支援は，設計作業の結果が仕様であって最終製品ではないので，製品検証に比べるとユーザの関心を集め難い．以上の考察から，これからの CASE に求められる性質は，最終製品に関係すること，及び現状の開発作業の変更が少ないことと要約できよう．

最終製品に関わる支援という点から，次の支援が注目される．

（1） プロトコル製品検証
（2） プロトコルインプリメンテーション
（3） 通信システムの機能向上

最初の2項の支援はソフトウェアの生産性向上を目的とするが，最後の項目の支援は，支援とアプリケーションとが一体化し，効用の高いシステムづくりを目指すものである．

以下，2. においてプロトコル開発支援全般を概観する．プロトコル開発支援の中でプロトコル検証は最も

† A Survey of CASE for Communications Software by Haruhisa ICHIKAWA (NTT Software Laboratories).
†† NTT ソフトウェア研究所

研究が盛んな領域である．プロトコル検証の中で到達可能解析と呼ばれるプロトコル解析技術が基本的で，実用性も高い．3.では，到達可能解析の技術動向と実用的なCASE例を紹介する．4., 5.では，上記(1)と(2)の支援技術について現状と今後期待される技術を述べる．(2)については(3)への貢献の観点からの技術動向についても述べる．なお，本稿では技術動向の把握に重点を置くため，具体的なCASEシステムの紹介は限定したもののみとする．各種システムについては文献[BOC 87a, LIU 89, ITO 87]や「情報処理」1990年1月号が詳しいので参照されたい．

2. プロトコル開発支援の概要

プロトコル開発支援の内容は，支援対象とその技術で捉えることができよう．このうち，支援対象は次の二つに大別されるプロトコル開発工程を構成する作業である．

1. プロトコル仕様開発工程：サービスあるいは機能を実現するプロトコル（通信手順）の仕様を開発する工程である．この工程での作業は次の二つに大別される．

（a） 仕様の作成・改造

（b） 仕 様 検 証

2. プロトコルインプリメンテーション：適当なハードウェアとソフトウェアによりプロトコルを実現する工程である．この工程での作業は次の二つに大別される．

（a） プログラムコード作成

（b） テ ス ト

以上の関係を図-1に示す．

仕様の記述形式は，文章，図形，表，形式言語と形式性の低いものから高いものまで多様である．プロトコル設計支援を自動化するには，形式性の高い仕様を記述することが重要である．このため，各種の記述モデルとこれに基づく記述言語が考案され，適用が提案されている．

プロトコル記述においては状態概念がよく用いられる．また，並列処理を強く意識したモデルも多い．これらの視点から分類したモデルの一覧を表-1に示す．表-1において，抽象プログラムは，既存のプログラミング言語（あるいはそのサブセット）を用いて記述した仕様をいう．

以上のモデルのうち，実用的な意味で最もポピュラなのは有限オートマトンモデルである．CCS (Calculus of Communicating Systems)[MIL 89], CSP (Communicating Sequential Processes)[HOA 85], ACP (Algebra of Communicating Processes)[BER 85]はプロセスの動作を主たる記述対象とする代数的理論であるためプロセス代数(process algebraic theory of behavior)と総称される[BRI 89a]．プロセス代数は，システム等価性概念など，プロトコルの構成的な設計のための基礎概念を定式化しているため，最近注目されているモデルである．

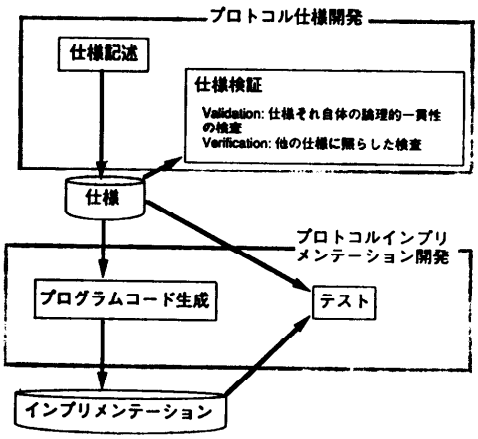

図-1 プロトコル開発の流れ

表-1 仕様記述モデル

状態記述を ＼ 並行処理を	特に指向しない	指向
指向		ペトリネット
	有限オートマトン	
特に指向しない	抽象プログラム 抽象データタイプ テンポラルロジック	CCS, CSP, ACP

また，上位階層プロトコルの記述では，メッセージデータの記述の重要性が認識されてきている．このため，メッセージデータ記述に抽象データタイプを適用し，上述のモデルと組み合わせてプロトコルを規定する手法の検討が盛んである．

これらのモデルを基に形式記述言語が開発されている．そのうち，有限オートマトンモデルをベースに，SDL (Specification Description Language) がCCITTにおいて，また，Estelle (Extended State Transition Language) がISOにおいて標準化されている．ISOではさらに，CCS及びCSPをベースとする言語LOTOS (Language of Temporal Ordering

Specification) が標準化されている．仕様記述言語には SDL のように図形表現を含むものも多く，構文エディタ，図形エディタなどが仕様作成段階の基本サポートとして実現されている．標準言語関連の支援については「情報処理」1990年1月号を参照されたい．

仕様検証には，仕様それ自体の論理的一貫性を検査するものと，他の仕様と比較して与えられた仕様の正しさを検査するものとがある．Rudin は前者を validation，後者を verification と呼んでいる[RUD 88]．論理的一貫性を犯す仕様誤りの例としては，デッドロック (dedlock) や予測されていない信号の受信 (unspecified reception) などがある．validation において検査する論理的一貫性は，仕様で規定すべき対象に依存せず要求される性質であるのに対し，verification では，対象依存の性質を別な仕様（要求仕様）として規定し，仕様（実現仕様）がそれを満たすことを調べる．

プログラムコード作成，すなわち，プロトコルを実現するプログラムを仕様から導出する作業は，さらに次の作業に細分できる．

（1） ターゲットシステムの条件を考慮して，与えられた仕様を詳細化する作業

（2） 詳細化された仕様に基づいてプログラムをコーディングする作業

プログラムコーディングは，仕様の記述形式を限定すれば自動化は比較的容易であるため，多くの自動化システムがこれをサポートしている．これに対し，仕様の詳細化の自動化支援は研究段階のものが多い．詳しくは 4. で述べる．

自動化支援には，必ず前提条件がある．実際のシステムはこの条件を満たさない場合があるため，テストは必ず必要な作業である．テストすべき内容は（1）性能 (performance)，（2）相互運用性 (interoperability)，（3）仕様充足性 (conformance) である．このうち，仕様充足性のテストは，プロトコル製品検証 (conformance testing) と呼ばれ，支援手法の研究が盛んに行われている[SAR 89]．プロトコル製品検証については 3. でもう少し詳しく紹介する．

1986年，Bochmann は世界の主要研究機関に対し，約70のプロトコル開発支援ツールのアンケート調査を行っている[BOC 87a]．その調査結果から，これまでに開発されているツールの現状は次のように要約できる．

（1） プロトコル設計及びインプリメンテーション生成とテスティングとは独立に（別なシステムで）支援されている．

（2） データの設計・コーディングの支援は手薄である．ただし，ASN. 1 (Abstract Syntax Notation One) サポートは進んでいる．（後述）

（3） 設計及びインプリメンテーションに対しては次の支援が代表的である．

- 仕様の動的な解析（網羅的あるいはシミュレーション）
- 仕様のシンタックスチェック
- プログラムコード生成

（4） テスティングに対しては次の支援が代表的である．

- PDU (Protocol Data Unit) 生成と表示のためのインタラクティブな支援
- テストの自動実行

3. プロトコル検証：到達可能解析

プロトコル検証に関する研究は，プロトコル設計支援の中で最も研究の盛んな領域である．検証内容には，シンタックスエラーのように記述の静的な解析で分かるものと，仕様に記述されたシステムの動作を動的に解析する必要のあるものとがある．動的解析では，システムが取り得る全域的な状態 (global state) の遷移状況を計算することになる．このような解析を到達可能解析 (reachabilty analysis) と呼んでいる．本章では，プロトコル検証における最も基本的で実用的な技術である到達可能解析について現状とプロトコル検証システム例を紹介する．なお，詳しくは文献 [ITO 88, LIN 87, SAJ 84, YUA 88, LIU 89] を参照されたい．

到達可能解析で計算すべき全域状態数はしばしば，きわめて膨大になる．この問題は状態爆発 (state space explosion) と呼ばれ，この問題解決が主要課題である．

状態爆発の問題に対処するために，これまでに提案されている手法は適用される段階によって次の3種類に分類できる[LIU 89]：（1）モデル化時，（2）モデル化した後で検証計算をする前，（3）検証計算時．第一の手法は，モデルに制限を加えるもので，変数の取り得る値，メッセージの種類，メッセージバッファサイズなどを制限することが提案されている[WES 82]．チャネルが FIFO (First In First Out) バッファの場合，全域状態の遷移関係を表すグラフの有界性判定は決定不能であり，このため，デッドロックの存在，チャネ

ルに蓄積されるメッセージの有界性などが決定不能である[BRA 83]．これを考慮し，解析したい性質が決定可能になるようなモデルの研究が進められている[VUO 82a, FIN88]．これも第一の手法に分類される．

第二の手法には，与えられたプロトコルを小さなプロトコルに分割して検証する手法[VUO 82b, CHO 83]，与えられたプロトコルを検証したい性質を表す小さなプロトコルに射影して検証する手法[LAM 84]などが提案されている．

第三に分類される手法には，縮退到達可能解析 (reduced reachablitiy analysis) がある．プロトコルシステムの動作はデッドロックの有無などの性質に関し同値な動作の集合に分類できる場合がある．この場合，同値類の代表元のみを計算し検証することにより，計算量を大幅に削減できる．この発想で各種の縮退到達可能解析法が開発されている[ITO 88, LIN 87, LIU 89, SAJ84, YUA 88]．

第三に分類される他の高速検証手法の中で，きわめて現実的なアプローチにより検証可能な仕様の規模を大幅に向上させている点で Holtzman の開発したプロトコル検証システム Trace[HOL 87] が注目される．Trace では，到達可能解析に要する計算量がきわめて大きいことを考慮し，プロトコル設計をデバッグ段階と検証段階に分けて扱う．デバッグ段階では，網羅的な解析よりも，設計誤り検出の応答性に重点を置く．ユーザは全域状態の探索範囲を制御 (scatter search と呼ばれる) することにより，最初に設計誤りを発見するまでの時間を短くできる．適用結果によると，scatter search により解析時間に対して指数関数的にエラーが見つかる．一例では 15 万状態を覚えるキャッシュを使い scatter search なしの場合には 9 時間かかって解析していたものが，2.5% のキャッシュを使って主要なエラーが 3 分以内に計算されている．また，最初にエラーを見つけるまでの時間も 20 分から 3 秒に短縮された．Trace の特徴は，上記の探索範囲制御だけでなく，到達可能解析アルゴリズムの効率的なインプリメンテーションにもある．インプリメンテーションの工夫により，通常の約 200 倍に解析速度を向上させている．

4. プロトコル製品検証

プロトコル製品検証とは先に述べたように，最終製品が満たすべき三つの性質のうち，仕様充足性をテストすることをいう．仕様どおりに動作することは標準化により通信可能領域の拡大を図ろうとする OSI の目的達成にとって重要な性質である．プロトコルテスティングの研究領域は以下の領域に大別される．

(1) テストメソッド：テストアーキテクチャとも呼ばれる．テストメソッドは，テストの対象となるプロトコルエンティティ (IUT (Implementation Under Test)) の出力観測方法，IUT への入力制御方法とで分類される．

(2) テストスート設計：プロトコル標準仕様の充足性をテストするための一群のテストシーケンスをテストスート (Test suite) という．テストスートは，対象となるプロトコルの各種パラメータを考慮した効率的なテストを行うことを目的に設計される．

(3) テスト実行：テストの実行とは，具体的には，IUT にテスト入力を与え，その動作を観測して仕様充足性を解析することである．

以上のうち，テストスート設計についての研究が多い．テストスート設計においても有限状態機械 (FSM (Finite State Machine)) は最も代表的なプロトコル記述モデルである．FSM で記述されたプロトコルのテストシーケンス生成法は次の二つに大別される[SAR 89, SID 89]．

(1) 全遷移の存在を検査するためのテストシーケンス生成法

(2) 遷移の存在だけでなく遷移後の状態も検査するテストシーケンス生成法

(1)は(2)に比べエラー検出能力に劣る一方，(2)は対象とする FSM に条件を必要とする．

(1)はT法と呼ばれ，全遷移を少なくとも 1 回は含む遷移アクションシーケンス (Transition Tour と呼ばれる) をテストシーケンスとして生成する．(2)では遷移後の状態をチェックするために状態を特定するためのシーケンスを用いる．このシーケンスの選び方により，U法，D法，W法と呼ばれる 3 種類が提案されている．U法は unique input/output (UIO) シーケンスを用いる．このシーケンスはある状態について他の状態とは異なるユニークな入出力動作を示すシーケンスである．D法は distinguishing sequence と呼ばれるシーケンスを用いる．このシーケンスは，プロトコルマシンの全ての状態に対してその状態からこのシーケンスで遷移させると異なる出力が現れるという性質をもつシーケンスである．W 法では，characterization set と呼ばれるシーケンス集合を用いる．この集合の各シーケンスで遷移させた後の最後の出力の組

がすべての状態について異なるのがこの集合の特徴である.

エラー検出能力に関して U, D, W 法は同じである. テストシーケンスの存在条件はU, W法が同じであり, D法はより厳しい条件が必要となる. 導出されるテストシーケンス長については, 一般的に論じることは困難であるが, 平均的にはT法が一番短く, U法とD法がほぼ同じでW法はかなり長くなる傾向がみられる.

FDT (Formal Description Technique) で記述されたプロトコル標準が増えているため, 上述の理論などを用いて FDT 記述からテストシーケンスを自動生成することが研究されている. 特に FSM をモデルとしている Estelle や SDL による記述からの導出は直接的である. 以下では, 典型的な支援例として Estelle に対するもの[SAR 87] を紹介する.

Estelle では FSM と異なり, データコンポーネントも記述できる. 文献 [SAR 87] の手法は次の手順によりデータも考慮したテストスートの設計を支援する.

1. 標準形式への変換：状態遷移のための内部変数の条件などを整理することにより, Estelle 記述を標準形に変換する.

2. 制御フロー, データフローの生成：標準形から制御フローとデータフローを表す2種類のグラフを生成する. 制御フローグラフは状態間の遷移関係を表す FSM で, これに上述した手法を適用して部分テストシーケンスを導出する. データフローグラフは入力, 内部変数, 出力の間のデータの流れとこれに必要なアクションとを表す.

3. テストシーケンスの生成：データフローグラフを内部変数対応のグラフ（ブロック）に分解する. 各ブロックについて, そこに現れるすべてのアクションをテストするシーケンスを求める. このテストシーケンスは上述の部分テストシーケンスを組み合わせて求める. テストシーケンスの実行に必要な各種パラメータの候補値は Estelle のデータ記述から導く.

本手法はファイル転送プロトコルやトランスポートプロトコル（クラス）に実験的に適用されている.

上記よりも実用性を重視したテストスート設計支援として, TTCN (Tree and Tabular Combined Notation) 支援がある. TTCN は ISO で標準化が進められているテストスート記述言語である. TTCN の特徴及び支援については文献 [KAT 90] を参照されたい.

5. プロトコルインプリメンテーションのための CASE

本章ではプロトコルインプリメンテーションのための CASE をその適用の前提条件に着目して紹介する.

5.1 インプリメンテーションの構造を前提にした自動化

未決定な要素を残した規定が仕様であるという立場に立つならば, 仕様からのインプリメンテーションの完全な自動生成はありえない. 多くの場合, 仕様からのインプリメンテーションの自動生成支援とは, 自動生成可能な部分とそうでない部分をインプリメンテーション上分離する技術とみることができる. これまでの報告によれば, 60% 程度は自動生成できると結論できる[LIU 89, RUD 88].

実用ソフトウェアへの適用で有名なのは, IBM の SNA (Systems Network Architecture) の開発に適用されたインプリメンテーション自動生成技術である[NAS 87]. この手法は10年近くの歴史をもつこと, 他の技術も類似したアイデアに基づいている点で基本的である. 入力は FAPL (Format And Protocol Language) と呼ばれる言語で記述される. インプリメンテーション生成は2ステップの処理を通じて行われる. まず, FAPL 記述は PL/S という中間言語記述に変換される. 一方, 自動生成できない部分はマニュアル記述された PL/S プログラムとして用意され, FAPL から生成された PL/S 記述と一緒にコンパイルされてインプリメンテーションコードが生成される. FAPL は拡張状態遷移モデルに基づく言語で, PL/I ライクなシンタックスをもち, 次の特徴を有する.

（1） プロトコルエンティティの動作を規定する状態遷移表を記述するための言語要素をもつ.

（2） Mappable/Nonmappable data を宣言できる.

(2)の特徴は, プロトコル記述のデータ依存性を陽に規定するためのものである. プロトコルエンティティ間の通信を保証するには, 伝送データ, いわゆる PDU (Protocol Data Unit) はビット表現まで規定されなければならない (Mappable でなければならない) が, そうでないデータはターゲットマシンに合わせて最適化の余地を残しておくのが望ましい. (2)の宣言はこの違いを宣言するためのものである.

拡張状態遷移モデルに基づく自動化は他にも例が多い. NIST (National Institute of Standards and

Technology) は ISO の標準言語 Estelle のサブセットからなる言語記述で OSI のファイル転送プロトコルを開発し，仕様からC言語インプリメンテーションの 40% の自動生成に成功している[MIL 84]. Serre らはトランスポートプロトコルを Estelle 記述から生成し約半分のプログラムコード生成，メモリスペースで 60%増の結果を出している[SER 86]. Blumer らも同様な結果を報告している[BLU 82]. CCITT の言語 SDL も状態遷移モデルに基づく言語として代表的である．SDL 関連の支援については文献 [WAK 90] を参照されたい．

上位プロトコル層では全体のプロトコル記述に占める，データ (PDU) 記述の割合が大きいため，PDU の組み立て/分解を行うプログラムの自動生成支援ツールが開発されている．インプリメンテーション自動生成の観点から重要なのは ASN. 1 支援技術である．ASN. 1 は，OSI 応用層エンティティが扱うデータの表現形式と，転送のためのデータ表現形式を区別して PDU を規定するための言語として開発された．各種 ASN. 1 支援システムについては文献 [TAK 90] を参照されたい．文献 [HAS 88] では，Estelle と ASN. 1 の組み合わせにより自動化率が向上することを報告している．状態遷移モデルに基づく仕様記述言語と ASN. 1 とを組み合わせることにより，実用性の高い支援が得られるものと期待される．

5.2 開発メソッドを前提にした自動化

前節では，インプリメンテーション構造を仮定することによる自動化を紹介した．前提条件を増やすことにより，さらに大きな自動化が期待される．このような前提条件として，本節では開発メソッドを取り上げる．

通信ソフトウェアの開発メソッドに関しての一般的見解については，ISDN サービスの特徴づけと定義のための CCITT 勧告 (I. 130) が参考になる．多くのプロトコル仕様記述言語にみられるように，通信ソフトウェアではメッセージフローの記述が重要な位置を占める．I. 130 では，メッセージフローの設計を次の段階に分けている．

ステップ 1 ユーザと通信システム(ネットワーク)の間のメッセージフロー記述．

ステップ 2 上記を実現する通信ネットワーク機能要素間のメッセージフロー記述．正常フローなど主要なもののみを規定する．

ステップ 3 メッセージフロー制御に関する通信ネットワーク機能要素の機能の記述．ステップ2では記述されない異常フローなども規定する．

SDE[ICH 87] は，ステップ1，2から3への過程に次の解釈を与えて構成された通信ソフトウェア開発環境である．

「メッセージフローを列挙して所要機能を把握した後，これらを統合してシステムの動作フローを記述する．」この解釈の利点は，メッセージフローのシステム動作フローへの統合を自動化支援すれば，列挙されたメッセージフローの選択，編集及び新フローの追加などにより，システムを効率的にカスタム化できることにある．

SDE では列挙すべきメッセージフローを SAL という言語で記述する．複数の SAL 記述からこれらを統合して実現するプロトコルエンティティプログラムが生成される．この過程で，各 SAL 記述のプロトコル的正当性，プロトコルエンティティへの統合可能性などが検証される．SDE が生成するプロトコルエンティティプログラムの実行方式が開発され，商用 PBX プログラムの生成実験が行われている[KAT 89]. PBX ソフトの 60% 以上がサービス制御プログラムであり，その 90% 以上が SAL 記述から生成できる．生成されるプログラムのメモリ量及び性能も実用的な値であることが確認されている．

SDE と同様な構想による支援システムに，EXPRESS[SHI 87] と SNAPSHOT[KOY 90] がある．SDE が状態遷移モデルをプログラムモデルとしているのに対し，EXPRESS はペトリネットモデルを用いている．SNAPSHOT システムでは，グラフィックインタフェースを介した設計支援，及び通信端末間の接続関係の変化シーケンスからのメッセージフロー生成に力点が置かれている．Newster[MIZ 89] では，サービスプロトタイピングを目的とし，サービスごとに列挙すべきメッセージフローの設計に的を絞った支援を実現している．端末の動作記述をあらかじめ用意しておくことにより，端末が出すべきメッセージをメニュー選択できる．また，基本メッセージ，手続きを実現する PBX システムと結合しており，動作確認できる．

開発メソッド自体の研究は記述言語と密接な関係にある．LOTOS や上述の SAL は，通常のプログラミング言語とは異なる作法を必要とするため，メソッドに関し研究発表がなされているが[VIS 88, SCO 87, BRI 89b, ICH 90, TAK 88]，発展途上の分野と言える．

6. おわりに

通信ソフトウェアを対象にした CASE にとって基本的なプロトコル設計支援技術の動向を概観した後，CASE に対するニーズに注目して技術動向を紹介した．ニーズとして特にプロトコル製品検証とインプリメンテーション自動化に注目した．

通信ソフトウェア CASE の研究はそろそろ実用性を問うべき段階に到達しているというのが筆者の基本認識である．また，実用的な CASE は，単なるソフトウェア生産性向上だけでなく，通信システムの機能向上に貢献しなければならないと考えている．

通信サービスの理想はいつでも，どこでも，だれとでも通信できることである．このため，通信システムは標準化されたインタフェースを提供する必要から，機能変更が容易でない「固いシステム」となる傾向が強い．一方通信網の用途は電話から大きく踏み出そうとしている．機能の多様化と便利さの両立を目指して開発・改良を繰り返す必要性がますます大きくなる．通信網や通信ノードを「柔らかくする」構想[HAA 87, KON 87, FRA 88]の検討が進んでいる．また，ISDN を考慮した通信サービスモデルの研究[BAU 88]なども報告されている．CASE 研究においても，5.2 で紹介したように，機能の多様化を支援するために強い前提条件を導入して強力な自動化を達成しようとする傾向がある．これらの多角的研究との総合化はまだまとまった動きにはいたっていないが，今後期待される分野である．

謝辞 本稿執筆に当たり，ご討論いただいた NTT ソフトウェア研究所関係各位に感謝します．

参考文献

[BAU 88] Baumgartner, T. J., Hwang, Y. H., Morgan, M. J. and Leung, W. H. : *The User Programmability of a Multimedia Workstation for Fast Packet Networks*, Globecom '88, 1. 1 (1988).

[BER 85] Bergstra, J. A. and Klop, J. W.: *Algebra of Communicating Processes with Abstraction*, TCS 37, pp. 77-121, North-Holland (1985).

[BLU 82] Blumer, T. P. and Tenney R.: *A Formal Specification Technique and Implementation Method for Protocols*, Computer Networks and ISDN Systems, Vol. 6, No. 2, pp. 201-217 (1982).

[BOC 87 a] Bochmann, G. v.: *Usage of Protocol Development Tools: The Results of a Survey*, Proc. of 7th IFIP Symposium on Protocol Specification, Testing, and Verification, pp. 139-161, North-Holland (1987).

[BOC 87 b] Bochmann, G. v., Gerber, G. W. and Serre, J. M.: *Semiautomatic Implementation of Communications Protocols*, IEEE Soft. Eng., Vol. SE-13, No. 9, pp. 989-1000 (1987).

[BRA 83] Brand, D. and Zafiropulo, P.: *On Communicating Finite-State Machines*, Journal of ACM, Vol. 30, No. 2, pp. 323-342 (1983).

[BRI 89 a] Brinksma, E.: *Specification Modules in LOTOS*, FORTE '89 (Dec. 1989).

[BRI 89 b] Brinksma, E.: *Constraint-Oriented Specification in a Constructive Formal Description Technique*, Stepwise Refinement of Distributed Systems, Proc. REX Workshop, W. P. de Roever (ed.), Lecture Notes in Comp. Science, Springer-Verlag (1989).

[CHO 83] Choi, T. Y. and Miller, R. E.: *A Decomposition Method for the Analysis and Design of Finite State Protocols*, Proc. of 8 th ACM/IEEE Data Comm. Symp., pp. 167-176 (1983).

[FIN 88] Finkel, A.: *A New Class of Analyzable CFSMs with Unbounded FIFO Channels*, Proc. of IFIP 8th Symposium on Protocol Specification, Testing, and Verification, North-Holland, pp. 283-294 (1988).

[FRA 88] Frank, H. G., Luke, D. C. and Cooper, B. A.: *The MACSTAR System: A User-Programmable Centrex CSR System*, 1. 3, Globecom '88 (1988).

[HAA 87] De Haas and Humes, R. W.,: *Intelligent Network/2: A Network Architecture Concept for the 1990s*, ISS '87, A 12. 1. 1 (1987).

[HAS 88] Hasegawa, T. et al.: *Automatic Ada Program Generation from Protocol Specifications based on Estelle and ASN. 1*, Proc. of ICCC '88, pp. 181-185 (1988).

[HOA 85] Hoare, C. A. R.: *Communicating Sequential Processes*, Prentice-Hall International (1985).

[HOL 87] Holtzmann, G. J.: *Automated Protocol Validation in Argos: Assertion Proving and Scatter Searching*, IEEE Trans. Soft. Eng., Vol. SE-13, No. 6, pp. 683-696 (1987).

[ICH 87] Ichikawa, H., Itoh, M. and Kato, J.: *Communications Software Management with Verification and Transformation of Protocol-Oriented Specifications*, Globecom '87, 17. 4, pp. 651-655 (1987).

[ICH 90] Ichikawa, H., Yamanaka, Y. and Kato, J.: *Incremental Specification in LOTOS*, submitted to IFIP 10 th Symposium on Protocol

Specification, Testing, and Verification (1990).

[ITO 87] 伊藤, 市川：通信分野における自動プログラミング, 情報処理, Vol. 28, No. 10, pp. 1405-1411 (1987).

[ITO 88] Itoh, M.: *Reduction Techniques for the Protocol Reachability Analysis*, Technical Report of Motreal Univ., Publication No. 656 (1988).

[KAK 86] Kakuda, Y. and Wakahara, Y.: *A New Algorithm for Fast Protocol Validation*, Proc. of COMPSAC '86, pp. 228-236 (1986).

[KAT 89] Kato, J. and Arakawa, N.: *Software Architecture for Automated Communications Software Development*, SETSS (1989).

[KAT 90] 勝山, 佐藤：試験仕様記述言語 TTCN の特質と処理系の現状と動向, 情報処理, Vol. 31, No. 1, pp. 65-74 (1990).

[KON 87] Kondo, T., Sakai, H., Yamaguchi, T. and Kamimura, E.: *Architecture of Open Application System*, Report of Technical Group, IECE J, SE 87-89 (1987).

[KOY 90] Koyamada, M. and Iwado, D.: *A Stepwise Approach to Behavior Design for Real-Time Software*, 1st International Conference on Systems Integration (Apr. 1990).

[LAM 84] Lam, S. S. and Shankar, A. U.: *Protocol Verification via Projections*, IEEE Trans. Software Engineering, SE-10, No. 4, pp. 325-342 (1984).

[LIN 87] Lin, F. J., Chu, M. and Liu, M. T.: *Protocol Verification using Reachability Analysis: The State Space Explosion Problem and Relief Strategies*, SGICOMM '87 Workshop, Frontiers in Computer Communications Technology, Stowe, Vermont (1987).

[LIU 89] Liu, M. T.: *Protocol Engineering*, Advances in Computers, Vol. 29, pp. 79-195, Academic Press (1989).

[MIL 84] Mills, K. L.: *Testing OSI Protocols: NBS Advances the State of the Art*, Data Communications, Vol. 13, pp. 277-285 (1984).

[MIL 89] Milner, R.: *Communication and Concurrency*, Prentice-Hall International (1989).

[MIZ 89] 水野, 新津：サービス記述手順に基づく階層形サービスソフトウェア生成法の検討, 信学技報 SSE 89-63 (July 1989).

[NAS 87] Nash, S. C.: *Format and Protocol Language (FAPL)*, Computer Networks and ISDN Systems, No. 14, pp. 61-77 (1987).

[RUD 88] Rudin, H.: *Protocol Engineering: A Critical Assessment*, Proc. of 8th IFIP Symposium on Protocol Specification, Testing, and Verification, pp. 3-16, North-Holland (1988).

[SAJ 84] Sajkowski, M.: *Protocol Verification Techniques: States quo and Perspectives*, Proc. of 4th IFIP Workshop on Protocol Specification, Testing, and Verification, pp. 697-720, North-Holland (1984).

[SAR 87] Sarikaya, B., Bochmann, G. v. and Cerny, E: *A Test Design Methodology for Protocol Testing*, IEEE Trans. Soft. Eng., Vol. 13, No. 5, pp. 518-526 (1987).

[SAR 89] Sarikaya, B.: *Conformance Testing: Architectures and Test Sequences*, Computer Networks and ISDN Systems, Vol. 17, North-Holland, pp. 111-126 (1989).

[SCO 87] Scollo, G. and Sinderen, M. v.: *On the Architectural Design of the Formal Specification of the Session Standards in LOTOS*, Proc. of IFIP 6th Workshop on Protocol Specification, Testing, and Verification, pp. 3-14, North-Holland (1987).

[SER 86] Serre, J. M., Cerny, E. and Bochmann, G. v.: *A Methodology for Implementing High-Level Communication Protocols*, Proc. 19th Hawaii In. Conf. on System Sciences, pp. 744-754 (1986).

[SHI 87] Shibata, K., Tanaka, W. and Hasegawa, H.: *Support System for Specification Phase of Communication Software*, Globecom '87, 17. 3, pp. 646-650 (1987).

[SID 89] Sidhu, D. P. and Leung, T. K.: *Formal Methods for Protocol Testing: A Detailed Study*, IEEE Trans. Soft. Eng., Vol. 15, No. 4, pp. 413-426 (1989).

[TAK 88] 田倉, 市川：サービス追加を考慮した通信ソフトウェア仕様記述スタイル, 63信学会秋季全国大会, B-205 (1988).

[TAK 90] 高橋, 中川路：構文定義用言語 ASN. 1 の特質と処理系の現状と動向, 情報処理 Vol. 31, No. 1, pp. 56-64 (1990).

[VIS 88] Vissers, C. A., Scollo, G. and Sinderen, M. v.: *Architecture and Specification Style in Formal Descriptions of Distributed Systems*, Proc. of IFIP 8th Symposium on Protocol Specification, Testing, and Verification, pp. 189-204, North-Holland (1988).

[VUO 82a] Vuong, S. T. and Cowan, D. D.: *Reachability Analysis of Protocols with Non-FIFO Channels*, Proc. COMPCON Fall 82, pp. 267-276 (1982).

[VUO 82b] Vuong, S. T. and Cowan, D. D.: *A Decomposition Method for the Validation of Structured Protocols*, IEEE INFOCOM, pp. 209-220 (1982).

[WAK 90] 岩原：SDL 言語の特質と処理系の現状と動向, 情報処理, Vol. 31, No. 1, pp. 23-34 (1990).

[WES 82] West, C. H.: *Applications and Limitations of Automated Protocol Validation*, Proc. of 2nd IFIP Workshop on Protocol Specification, Testing, and Verification, pp. 233-242, North-Holland (1982).

[YUA 88] Yung, M. C.: *Survey of Protocol Verification Techniques Based on Finite State Machine Models*, Proc. of IEEE Computer Networking Symposium, pp. 164-172 (1988).

(平成2年2月28日受付)