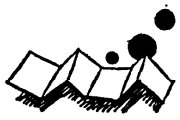
Vol. 28 No. 4　　情　報　処　理　　Apr. 1987

解　説

2. 基　礎　技　術



2.4 試　験　検　証　技　術†

白　鳥　則　郎††

1. まえがき

計算機システムは，集中処理型を中心として発達し，ジョブの質的向上および量的拡大，さらに半導体技術の進歩にともない，しだいに分散処理型へ移行した．分散処理システムの具体例として，密結合のマルチプロセッサシステムから，通信網を基本とした大型システムまで，多種多様の形態がある．

このような分散処理システムの構成問題は，空間的，時間的，及び論理的に分散しているプロセッサや情報資源などを統合し，ユーザ側とシステム提供側の要求である「仕様 (specification)」を満足しながら，質的及び量的な機能の向上をはかると同時にコストの低下とパフォーマンスの向上を目的関数とする情報処理システムの構成問題である．そのため，基本的に通信技術と計算機技術の融合が必須の条件となる．

前述の構成問題の中で，分散処理システムの試験検証問題は，実現されたシステムの機能が正しく動作し，かつ，与えられた「仕様」を満足しているかどうかを調べる問題である．このようなシステムの試験検証問題は，複雑・多岐にわたっている．そのため，全体の問題を部分問題に分解して検討が進められている．

以下では，初めに 2. で試験検証の観点から分散処理システムを概観する．3. では，分散処理システムの試験検証問題の基本的な考え方について述べる．続いて，4. 以降では，通信網における中心的な課題であるプロトコルの試験検証技術に焦点を絞る．5. と 6. でそれぞれプロトコル検証と適合性試験について詳述する．最後の 7. では，課題と将来動向を展望する．

2. 分散処理システムの特徴

2.1 目　的

分散処理システムは複数のプロセッサの集合であり (a)処理効率の向上，(b)情報資源の共用，(c)信頼性の向上などの目的をもった情報処理システムである．

(a)は，並列処理，負荷分散と機能分散である．(b)は，データベースなどの高価な情報資源を共有し，活用することである．(c)については，たとえば同一のプロセッサ，あるいは通信路を複数個用意し，一つのプロセッサ，あるいは通信路が故障してもシステム全体としては，その機能を保存することにより信頼性を高めることができる．

2.2 分　類

分散処理システムは，プロセッサ及び情報資源間の結合の度合により，(a)密結合形と(b)疎結合形に分類される．(a)は，プロセッサ及び情報資源間の結合が強いシステムで，バス結合でメモリを共用するマルチプロセッサシステムが代表例である．(b)は，結合の度合が弱く，広域ネットワークを基本としたシステムである．また，(a)と(b)の中間の形態がある．たとえば，ローカルエリアネットワークは，結合方式や規模に応じて(a)や(b)に近い形態となる．さらに，複数のネットワークがゲートウェイで結合され，統合化された大規模システムもある．

疎結合形のシステムにおける構成問題及び試験検証問題は，密結合形システムにおける対応する問題を含んでいる．一般的な傾向としては，疎結合形では，後述する通信処理や情報処理の機構が複雑であり，密結合形より解決すべき課題が多く，その構成問題及び試験検証問題はやっかいである．たとえば，広域ネットワークにおけるメッセージの経路選択法やプロトコルに対応するものは，メモリを共用するマルチプロセッサシステムでは，比較的簡単なものとなっている．

† Techniques of Verification and Conformance Testing by Norio SHIRATORI (Research Institute of Electrical Communication, Tohoku University).

†† 東北大学電気通信研究所

2.3 通信処理と情報処理

分散処理システムの機能を処理の観点から分類すると，通信処理と情報処理に分けることができる．ここで，通信処理とは，情報の内容を変えない処理，また，情報処理とは，情報の内容に変化をともなう処理と定義する．

2.3.1 通 信 処 理

通信処理は，システムが円滑に，かつ，効率的に動作するための機構が中心であり，図-1 に示すように，アルゴリズムとプロトコルに分類できる．アルゴリズムは手続き的に与えられ，この中には，制御アルゴリズムと変換アルゴリズムがある．

制御アルゴリズムは，分散処理システムが仕事を円滑に処理するための制御機構であり，空間的及び論理的な分散環境におけるプロセッサのスケジューリング，メモリ管理，データベースなどの資源管理などから構成されている．また，プロセス間通信の方法，分散データベースにおける同時実行 (concurrency) 及び一致性 (consistency) 制御，デッドロック問題，障害及び誤りの検出・回復の方法などもこの範疇に含まれる．

変換アルゴリズムには，メディア変換，プロトコル変換，表現形式の変換などがあげられる．

アルゴリズムが手続き的に規定されるのに対し，プロトコルはルールの集合として非手続き的に与えられる．プロトコルは，人間や機械が円滑に会話するための約束事であり，現在，機械と機械の会話においてだけでなく，他の場合でも広く使用されている．以下では，主として機械同士の会話におけるプロトコルを議論の対象とする．なお，プロトコルは，ルーチングやフロー制御などのようにアルゴリズムとして与えられる場合もある[30]．

分散処理システムでは，入力された仕事を処理するために，まず通信処理機能の制御アルゴリズムが動作し，プロトコルを用いて通信路を設定する．次に，この通信路を用いてプロセッサ間の通信を実現し，プロセッサの情報処理機能を活用しながら仕事を処理する．

2.3.2 情報処理機能

通信処理機能が分散処理システムの円滑な動作を実現するための制御を行うのに対し，情報処理機能は，実質的な計算アルゴリズムを実行する機能である．応用プログラムがその例としてあげられる．

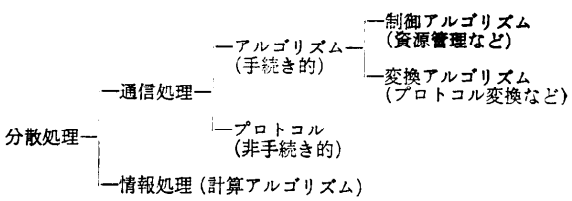


図-1 分散処理システムにおける処理の分類

3. 試験検証技術

3.1 システムの性質

システムの性質は，試験検証の観点から安全性 (safeness) と生存性 (liveness) に分類できる．安全性は，すべての時点でシステムが満足しなければならない条件であり，生存性は，システムが，いつかは期待する望ましい結果に到達することを表す．試験検証問題は，対象とするシステムについて，このような安全性と生存性を調べる問題である．

3.2 試験検証の基本概念

一般に，システムの構成問題は，図-2 に示すように，システムの仕様 S が与えられたとき，これらを満足する機能 F を構成することである．一方，試験検証問題は，構成されたシステム機能 F が正しく動作し，かつ，仕様 S を満足しているかどうかを調べることである．

分散処理システムの機能は，2. で述べたように，通信処理と情報処理に大別される．そのため，システムの仕様は，通信処理の仕様 S^C と情報処理の仕様 S^I に類別される．システムとして実現する機能 F も，これらに対応して，通信処理機能 F^C と情報処理機能 F^I に分類することができる．

以上のことから，分散処理システムの試験検証問題は，次の三つの部分問題に分割することができる．第1は，図-1 に示す通信処理について実現された機能 F^C が正しく動作し仕様 S^C を満足しているかどうかを調べる問題．第2は，同様にして，F^I が正しく動作し S^I を満たすかどうかの問題．第3は，第1と第2の結果を用いて，実現された機能 F が正しく動作し仕様 S を満たすかどうかを調べる問題である．

これらの問題は，いずれも複雑・多岐にわたっている．たとえば，通信処理に限定し，その中からプロセ

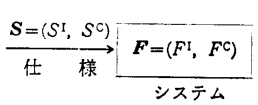


図-2 システムの構成と試験検証問題

ッサのスケジューリング，プロセス間通信の方法，デッドロック問題などを考えてみても，前述した試験検証問題は，やっかいな証明問題となっている．

システムが集中処理から分散処理へ移行するにつれ，通信処理の規模が大きくなり，同時に，その機構も複雑となる．そのため，分散処理システムの試験検証問題では，集中処理システムの問題と比較し，主として通信処理に関した問題があらたに生起し，これが試験検証問題をやっかいな証明問題としている主要因である．

3.3 仕様検証とシステム検証

分散処理システムが大規模化するにつれ，システムの仕様自体の規模も大きくなる．そのため，試験検証は，後述する仕様検証とシステム検証に分けて図-3の手順で行われることが多い

図-3 において，初めに(1)でメモ的な素案を作成する．次に(2)では，(1)を詳細化，厳密化した仕様記述を行う．ここで，仕様が複雑で大規模化すると，仕様を分かりやすくかつ厳密に記述することが必要となる．これが仕様記述法の問題である．次のステップ(3)が仕様検証である．これは，仕様自身に誤りがあるかどうか調べる問題である．検証が終わると，次の(4)において，その仕様を実現する機能の構成及びインプリメントが行われ，実際に稼動できる状態となる．このようにして実現されたシステムの機能が正しく動作し，かつ，もとの仕様を満足しているかどうかを調べるステップ(5)をシステム検証と呼ぶ．

これまで，前述の仕様検証やシステム検証について種々の接近法を用いた諸研究が行われてきた．しかしながら，簡単で小さなシステムを対象とした場合を除くと，完全な正当性を示すことは，非常に難しい問題となっているのが実情である．そのため，検証法については，完全な正当性を求めず，現実的な立場から，デッドロックなどの不適格な条件について，この条件が成立するかどうかを調べる手法が開発されている．次に述べるプロトコルの試験検証技術は，このような例の一つである．

4. プロトコルの試験検証技術

4.1 プロトコルの試験検証

以下では，分散処理の基本となる通信処理において，円滑な通信を保証する機構を与えるプロトコルを代表例として，その試験検証技術について述べる．制御アルゴリズムの問題については，本特集の対応する解説を参照されたい．

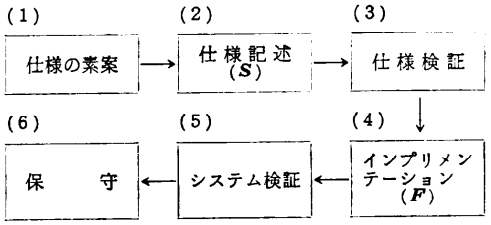


図-3 システム開発のライフサイクル

図-4 にプロトコルのライフサイクルを示す．この図は，図-3 においてシステムの例としてプロトコルを想定した場合の開発過程とその具体的な要素技術を示している．プロトコルの試験検証は，プロトコル検証と適合性試験 (conformance testing) からなっている．ここでプロトコル検証と適合性試験は，それぞれ，図-3 の仕様検証とシステム検証に対応している．プロトコル検証は 5. で，また適合性試験については，6. で詳述する．

4.2 プロトコルの安全性と生存性

3.1 で述べたように，試験検証の目的は，システムの性質，つまり安全性と生存性について調べることである．プロトコル検証についても同様である．

プロトコルは，システムを構成する要素の間で円滑な通信を行うための約束事 (rules) である．このようなプロトコルを規定する場合，状態の概念が重要となる．つまり，システム全体あるいは構成要素の状態として，何を，いかにして選ぶかが問題となる．従来，設計者は，経験的な知識と勘をもとに状態を決めてきた．もし，これらの状態が決まれば，プロトコルの設計は，これらの状態間の関係を規定することによって与えられる．

プロトコルを記述する場合，状態遷移を基本としたモデルだけでなく，代数や時間的論理に基づいたモデルにおいても，状態に関して陽に表現あるいは陰に表現しているかの違いはあるにせよ，いずれにしても，基本的には状態間の関係を記述することによってプロ

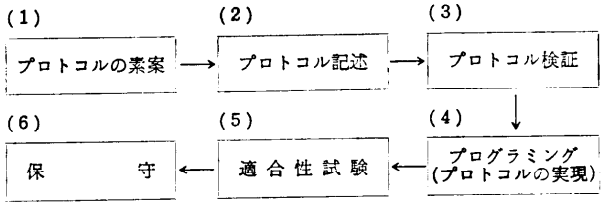


図-4 プロトコルのライフサイクル

トコルを規定している.

通常，システムの状態は，構成要素の状態の組によって表現される．このようなシステム状態に注目すると，プロトコルの安全性と生存性を次のように定義することが可能となる.

プロトコルの安全性は，システム状態の集合に関する性質である．安全性の定義は，この集合の任意の状態において常に成立している性質で，その中で特にシステムが具備すべき望ましい性質として与えられる．たとえば，デッドロックフリーや受信不可となるイベントの受信状態がない，などがある.

一方，プロトコルの生存性は状態のシーケンス，つまりパスに関する性質である．生存性は，状態のシーケンスからなるパスにおいて成立する望ましい性質として定義できる．たとえば，送信メッセージが相手に正しく受信される，また，正しく停止する，などがある.

4.3 プロトコルのライフサイクル

プロトコルのライフサイクルは，主として，図-4 に示す諸項目から構成されている．プロトコルの設計者は，初めに素案のイメージを抱き，メモを作成する．これが，図-4 のステップ(1)である．次に，ステップ(2)では，ステップ(1)をもとにし，形式的な記述法を用いてプロトコルを詳細に規定する.

ステップ(3)は，ステップ(2)で設計されたプロトコルの仕様検証である．ステップ(4)では，ステップ(2)で規定されたプロトコルを計算機システムに実装するためのプログラムを作成する．ステップ(4)で作成されたプログラムが，ステップ(2)で規定されたプロトコルを忠実に反映し，満足しているかどうかを調べる必要がある．これがステップ(5)の適合性試験(conformance testing) である．最後のステップ(6)は，保守のためのドキュメンテーションの作成である.

最近の傾向として，プロトコルと通信ソフトウェアの自動生成，及び適合性試験の検討が活発になっている．前者は，ソフトウェア工学を中心とした設計方法論の問題である．また，後者の適合性試験は，研究レベルだけでなく，設計・製造レベルにおいて緊急の課題となっており，6. で詳述する.

5. プロトコル検証法

5.1 状態遷移を基本とする方法[10)~17)]

これは，プロトコルを状態遷移図によって表現し，諸性質を検証する方法であり，他の手法と比較し，現実的に実際のプロトコル設計によく適用されている方法である．状態遷移図を用いる検証法では，プロトコルを通信するプロセス間におけるイベントの交換としてモデル化している場合が多い．また，時間に関係する条件も除いてモデルを簡単化している.

この検証法の長所は，デッドロックや特定の状態への到達可能性など，システム全体の動作可能性を調べるのが比較的容易なことである．また，計算機による自動化も容易である．一方，短所としては，プロトコルの規模の増大や複雑化により状態数が増加し，計算機処理が困難になってしまうことである.

このような手法には，デュアログマトリックス法 (dualogue matrix theory) と到達可能性解析 (reachability analysis) などがある.

5.2 プログラム言語を用いる方法[18)]

PASCAL などの高級プログラム言語を基本に，プロトコル表現に適した固有の命令セットを追加した言語仕様でプロトコルを記述する方法である．このような記述法を用いた仕様の検証法では，プロトコルの実行中に成立する状態変数間の不変論理関係式に関する検証となる.

この手法は，状態遷移図による方法では難しいシーケンス番号やタイマなどのパラメトリックな処理手順の検証に適している．一方，現在この検証法は，基本的にプログラムの正当性の証明と同様であり，現実的な観点から，あまり有効なものはないのが実情である.

5.3 時間的論理を用いる方法[19),20)]

時間的論理(temporal logic)による仕様記述法では，古典論理で用いる論理和，論理積，含意，否定などの演算子に加えて時間的演算子 (temporal operator) を導入している．ここで，時間演算子は，三つの単項演算子①□ (always または，henceforth)，②▽ (eventually または，sometimes)，③○ (next) 及び二項演算子 U (until) からなっている.

このような時間的論理を用いると，安全性はA⊃□B，生存性は A⊃□▽B として表現できる．たとえば，A⊃□Bは，Aが真ならば，現在から先Bが真となることを表している．現在，いろいろな角度から，プロトコルの生存性と安全性を示す試みが進められている．この方法は 5.1 で述べた状態遷移に基づく方法と比べて，生存性と安全性の議論を展開しやすく検証能力も高いという長所をもっている．一方，仕様の規

模が大きくなるにつれ，記述内容の理解性が低下し，検証の証明問題も困難となる．そのため自動検証へ向けた検討や検証支援システムの構成に関する考察が重要となろう．

5.4 代数を用いる方法[21)~23)]

代数的記述法では，オペレーションと変数の宣言部(定義域，値域)及び公理系によって仕様を規定する．ここで，変数はオペレーションの意味の定義に用いられる．また，公理系はオペレーションの意味，あるいは約束を規定する．

本記述法の長所は，仕様が公理系として与えられるので，検証が仕様記述と同一の枠組の中で議論できることである．ここで，前述の検証は，プロトコルの性質として検証すべき命題が，その公理系における定理として成立することを示すことである．また，仕様を目的に応じて適当な抽象レベルで表現することが可能である．

一方，短所は，仕様がステートメントのシーケンスとして与えられるので，プロトコルの規模が大きくなるにつれ，理解性が低下し保守などに難点をもたらす．また，この仕様をもとに規模の大きい実際的なプロトコル製品を作成することにより，実システムで稼動させる環境などについての考察も，今後，必要となろう．

6. 適合性試験

6.1 適合性試験とは

各種記述法によって表現されたプロトコルは，計算機プログラムを作成することによって実現される．図-4 の適合性試験は，実現されたプログラムがプロトコル仕様を満足しているかどうかを調べることであり，プログラムの正当性を示す問題に帰着する．このようにプログラムなどで実現されたシステムがもとのプロトコル仕様を満たすかどうかを調べることを適合性試験と呼ぶ．

一方，仕様としてのプロトコルは，比較的規模が大きく複雑である．そのため，形式的な正当性の証明はきわめて困難となっている．基本的には，プロトコルを満たす入力を実現されたシステムへ与えて出力を調べればよい．ところが，実現されたシステムの正しさを調べるために必要なすべての入力パターンを生成して，これを実現されたシステムへ入力し，その出力を調べることは，現実的に不可能な場合がほとんどである．

そのため，次善の策として，完全な正当性ではなく，現実的に重要となるプロトコルの機能に注目し，この機能が正しく動作することを調べるために必要な入力パターンを生成して実現されたシステムへ入力し，対応する出力を見てその機能の良否を判断する方法が採用されている．現実的には，このような方法で適合性試験が実施されている．

適合性試験では，同一プロトコルに基づいて製造された多種多様の製品を共通的，かつ，統一的に試験するための方法論を確立することが基本的な課題である．具体的には，試験の種類，テスタの位置，試験方法，試験の記述法，試験スイート (test suite) の生成法などの技術を確立する必要がある．

6.2 試験の種類と手順

適合性試験の種類には，机上で行う静的試験と実際に製品を動作させて実施する動的試験がある．前者は，プロトコルクラス，パラメータ値の範囲などの実装能力を中心に製品の製造者から提出されたドキュメントの検査を行う．この試験に合格してから後者の動的試験が実施される．

また，試験の目的に応じて，(a)基本適合性試験，(b)完全適合性試験と(c)適合性解決試験の3種類に分けられる．(a)は，基本的な機能を対象とする試験であり，(b)では，すべての機能を試験する．(c)は，試験手順の内部動作を確認することを目的としている．

6.3 試験の方法

試験の方法は，実施する場所と試験の対象となる層に応じて表-1 のように分類できる．前者の観点から類別すると，同一装置内で行うローカル試験(L)，分散した複数の装置を用いる分散試験(D)と遠隔地の製品を対象とする遠隔試験(R)がある．また，後者の対象となる層に注目すると，単一の層に対する試験(S)，複数の層に対する試験(M)と複数の層に埋めこまれた単一の層に対する試験(ME)に類別される．また，図-5，図-6 と図-7 に，それぞれローカル試験，分散試験，遠隔試験のシステム構成の概略を示す．

表-1 試験の方法

	単一の層 (S)	複数の層 (M)	埋めこみ層 (ME)
ローカル試験(L)	LS	LM	LME
分散試験 (D)	DS	DM	DME
遠隔試験 (R)	RS	RM	RME

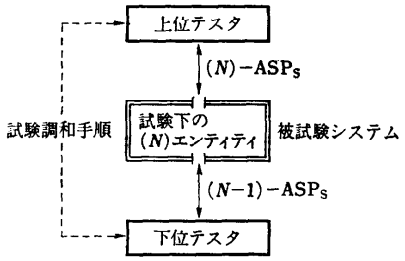


図-5 ローカル試験のシステム構成

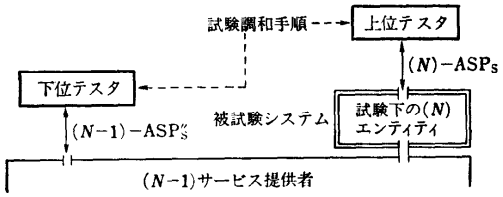


図-6 分散試験のシステム構成

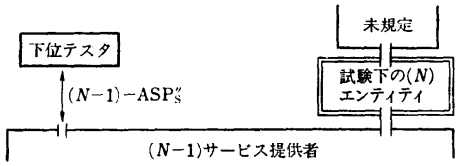


図-7 遠隔試験のシステム構成

現在，試験方法の標準化や定量的な評価，及び試験手順の最適化などについて種々の角度から検討がなされている．

6.4 試験の記述法

試験手順の記述法として，自然言語と形式言語がある．プロトコルの記述法と同じく，言語の具備条件として，厳密性や理解性が要求される．加えて，試験手順の特徴を適切に表現できる表現形式であることが望ましい．

これまで，試験手順の記述は，自然言語を基本とし，フローチャートやタイムチャートを併用して行われており，厳密性などに欠けていた．そのため，前述の条件を満たし，かつ，試験手順の表現に向いた記述法の開発が課題となっている．

現在，試験手順の記述法として，従来のプロトコルの記述言語を用いたり，新しく専用の言語を開発する試みがなされている[25),26)]．

6.5 試験スイートの生成法

実現された製品が仕様としてのプロトコルを満たしているかどうかを調べるための統一的に組織化された試験手順の集合を試験スイート (test suite) と呼ぶ．適合性試験では，プロトコルから試験スイートを構成し，被試験対象に入力し，対応する出力を検査する．

理想的には，プロトコルから自動的に試験スイートを生成することが望ましい．しかしながら，プロトコルと試験手順の記述法とも関係し，その実現は容易ではない．そのため，現実的には，プロトコルから試験スイートを構成するための支援システムの構築が，当面の問題となっている．

7. 課題と展望

分散処理システムの試験検証問題は，構成されたシステムが正しく動作し，かつ，仕様を満足しているかどうかを調べる問題である．この問題は，複雑多岐にわたっているため，部分問題に分割して検討が進められている．この中で，通信処理に関連した問題が中心課題である．部分問題は，大まかに各種の制御アルゴリズム，プロセス間通信，プロトコルなどに類別される．プロトコルは，プロセス間に論理的な通信路を設定し，これを用いてプロセス間通信が遂行される．さらに各種制御アルゴリズムが，対象となる仕事に応じて，プロセス間通信を用いて情報資源を活用しながら仕事を処理する．

以上のことから，試験検証問題では，制御アルゴリズム，プロセス間通信，プロトコルなどの試験検証技術の確立が基本となる．これらの技術を統合することによりシステム全体の試験検証が可能となる．したがって，まず，部分問題に対応する試験検証技術の確立が当面の課題である．ここで，制御アルゴリズムとプロセス間通信は，特に分散処理 OS，分散データベースなどと密接な関係をもっている．これらの問題については，本特集の対応する解説を参照されたい．以下では，プロトコルの問題に焦点を絞る．

プロトコルの試験検証は，プロトコル検証と適合性試験に分けられる．両者とも，検証法及び試験法に関して，(1)能力，(2)効率，(3)ヒューマンインタフェースの向上が課題である．(1)と(2)は，アルゴリズムの問題である．(3)は，プロトコルの保守あるいは，試験検証システムを構成する場合，プロトコル開発の生産性を向上させるための，ユーザフレンドリィなインタフェースの設計問題である．

従来，試験検証において，ヒューマンインタフェースの問題は，ほとんど議論されていない．プロトコルの誤りを修正するのは人間であり，生産性を向上させるには重要な課題である．これには，知識情報処理技

術を応用した知的インタフェースをもつ試験検証システムの構成が必要となろう．

プロトコル検証については，完全な正しさを調べる検証技法の確立を目指すと同時に，現実的な観点からの接近法も重要となる．前者は，時間的論理や代数を用いたものであり，後者は状態遷移を基本とした接近法である．両者には，長所と短所が混在している．そのため，両者の向上をはかるとともに，両者を止揚し，包括するような第3の接近法の研究・開発を期待したい．

現在，内外の標準化機関を中心にプロトコルの標準化が精力的に進められている．標準プロトコルに基づいた製品が，早晩，市場に多数出まわることになる．そのため，各種プロトコル製品が相互に矛盾なく通信できることを統一的に確認する適合性試験の技法を確立することが急務となっている．

適合性試験の課題は，主に次の3点であり，それぞれ相互に関連している．

（1） 試験手順の記述言語の開発
（2） 試験スイートの自動生成
（3） 試験システムの構成

（1）は，試験手順を厳密に，かつ，理解性に富む形式に表現する言語の開発である．（2）は，与えられたプロトコルから，試験スイートを構成し，（1）の言語で表現する問題である．（3）は，（2）で構成された試験スイートを用いて試験する具体的なシステムの構築である．

プロトコル検証および適合性試験は，プロトコルと対応するソフトウェアの生産性を向上させるための要素技術である．個々の要素技術を確立する接近法も大切だが，さらに，これらの要素技術の特徴を生かして統合する手法，つまり設計方法論を確立することが，より重要である．たとえば，プロトコル検証法は，プロトコルのライフサイクルにおける位置づけをふまえた上で，その特徴を生かせるような適切な設計法と融合，調和することにより，検証法の効果の飛躍的な向上が期待できるからである．

図-4 のプロトコルのライフサイクルにおいて，プロトコル検証及び適合性試験の最終的な目標は，プロトコルと対応する製品の円滑な開発とその生産性の向上に資することである．そのためには，前述した多くの課題を解決しなければならない．いずれにしても，仕様記述や適合性試験などの要素技術を確立するだけでなく，これらの融合性および全体性を志向しつつ，目的を明確にし一貫した思想のもとで研究・開発を進めることが，前述の目標を達成するための必要条件である．

参 考 文 献

1) Stankovic, J. A : A Perspective on Distributed Computer Systems, IEEE Trans. Comput., Vol. C-33, No. 12, pp. 1102-1115 (Dec. 1984).
2) Witt, B. I. : Communicating Modules : A Software Design Module for Concurrent Distributed Systems, IEEE Computer Magazine, Vol. 18, No. 1, pp. 67-77 (Jan. 1985).
3) Multiprocessing Technology, IEEE Computer Magazine, Vol. 18, No. 6 (June 1985).
4) 特集・分散処理，情報処理，Vol. 20, No. 4 (1979).
5) Chen, B. and Yeh, R. T. : Formal Specification and Verification of Distributed Systems, IEEE Trans. Software Eng., Vol. SE-9, No. 6, pp. 710-722 (Nov. 1983).
6) Blumer, T. P. and Sidhu, D. P. : Mechanical Verification and Automatic Implementation of Communication Protocols, IEEE Trans. Software Eng., Vol. SE-12, No. 8, pp. 827-843 (Aug. 1986).
7) Shiratori, N., Takahashi, K. and Noguchi, S. : IDESS/85 : Intelligent Support System for Protocols and Communication Software Development, Proc. of the Eighth International Conference on Computer Communication, pp. 543-548 (Sept. 1986).
8) Shiratori, N., Gohara, J. and Noguchi, S. : A New Design Language for Communication Protocols and a Systematic Design Method of Communication Systems, Proc. of the Sixth International Conference on Software Engineering, pp. 403-412 (Sept. 1982).
9) Zafiropulo, P. et al. : Towards Analyzing and Synthesizing Protocols, IEEE Trans. Commun., Vol. COM-28, No. 4, pp. 651-661 (1980).
10) Zafiropulo, P. : Protocol Validation by Duologue—Matrix Analysis, IEEE Trans. Commun., Vol. COM-26, No. 8, pp. 1187-1194 (1978).
11) West, C. H. : General Technique for Communications Protocol Validation, IBM J. Res. Dev., Vol. 22, No. 4, pp. 393-404 (1978).
12) Itoh, M. and Ichikawa, H. : Protocol Verification Algorithm Using Reduced Reachability Analysis, Trans. of IECE, Vol. E 66, No. 2, pp. 88-93 (1983).
13) 白鳥，郷原，野口：EXPA：パータベーション解析に基づく通信プロトコルの検証法，情報処理学会論文誌，Vol. 26, No. 3, pp. 446-453 (1985)

14) Rubin, J. and West, C. H.: An Improved Protocol Validation Technique, Computer Networks, Vol. 6, pp. 65-73 (1982).
15) Gouda, M. G. and Yu, Y. T.: Protocol Validation by Maximal Progress State Exploration, IEEE Trans. Commun., Vol. COM-32, No. 1, pp. 94-97 (Jan. 1984).
16) Sidhu, D. P. and Blumer, T. P.: Verification of NBS Class 4 Transport Protocol, IEEE Trans. Commun., Vol. COM-34, No. 8, pp. 781-789 (Aug. 1986).
17) West, C. H.: A Validation of the OSI Session Layer Protocol, Computer Networks and ISDN Systems, 11, pp. 173-182 (1986).
18) Stenning, N. V.: A Data Transfer Protocol, Computer Networks, Vol. 1, pp. 99-110 (1976).
19) Hailpern, B. T. and Owicki, S. S.: Modular Verification of Computer Communication Protocols, IEEE Trans. Commun., Vol. COM-31, No. 1, pp. 56-68 (Jan. 1983).
20) 越田, 斉藤, 猪瀬: 時間論理に基づいたプロトコル記述と検証, 信学論 (D), Vol. J 69-D, No. 3, pp. 346-354 (1986-03).
21) Sunshine, C. A. et al.: Specification and Verification of Communication Protocols in AFFIRM Using State Transition Models, IEEE Trans. Software Eng., Vol. SE-8, No. 5, pp. 460-489 (Sept. 1982).
22) Lam, S. S. and Shankar, A. U.: Protocol Verification via Projections, IEEE Trans. Software Eng., Vol. SE-10, No. 4, pp. 325-342 (July 1984).
23) Higashino, T. et al.: An Algebraic Specification of HDLC Procedures and Its Verification, IEEE Trans. Software Eng., Vol. SE-10, No. 6, pp. 825-836 (Nov. 1984).
24) 斉藤, 加藤, 榊, 猪瀬: オートマトンモデルによるHDLCプロトコル製品検証方式の実験的考察, 信学論 (D), Vol. J 65-D, No. 2, pp. 155-162 (1982-02).
25) Proc. of the Fifth IFIP International Workshop on "Protocol Specification, Testing and Verification" (June 1985).
26) Proc. of the Sixth IFIP International Workshop on "Protocol Specification, Testing and Verification" (June 1986).
27) Kato, T., Suzuki, K. and Urano, Y.: Conformance Testing for OSI Protocols in the Multiple Layer Environment Based on Automation Model, Proc. of the Eighth International Conference on Computer Communication, pp. 519-524 (Sept. 1986).
28) 適合性試験の国内及び国際動向について—認証調査委員会第1次成果報告書—, 日本規格協会, 情報処理標準化研究センター (1986-10).
29) 水野: プロトコルの形式記述とコンフォーマンス試験, 情報処理, Vol. 26, No. 4, pp. 420-427 (1985).
30) Gerla, M. and Kleinrock, L.: Flow Control: A Comparative Survey, IEEE Trans. Commun., Vol. COM-28, No. 4, pp. 553-574 (Apr. 1980).
31) 榎本 肇(編): ソフトウェア工学ハンドブック, オーム社 (1986-09).

(昭和61年12月15日受付)