OSインタフェース検定システム

竹中市郎† 小田英雄†† 森廣政治††

アプリケーションプログラム (AP) の移植性を確保するためには，オペレーティングシステム (OS) インタフェースの標準化とともに，製品として実現されたOSが標準仕様に準拠していることを検証することが重要である．既にPOSIX，シグマOSに対しOSインタフェースの検定 (Validation または Conformance Testing) が実施されている．これらのインタフェース規定においては，製品が標準仕様に含まれない機能を持つこと（機能過多）を許容している．このため検定もテストスイート（テストプログラムの集合）の実行により標準仕様との照合を行う方法（機能検定）が用いられており，機能過多の検出は行っていない．しかし，例えばCTRONのように機能過多を許容しないインタフェース規定の場合には，この方法のみでは十分な検定ができない．また，機能検定においては，どこまで厳密にテストを行うべきかという問題がある．本論文では，上記問題点に対しそれぞれ，機能過多を検出する方法としてドキュメント検定の採用，検定精度の定量的評価法として，システムコールの入力パラメータ値の組合せに基づく16のレベルを定義し，レベルに応じたテスト項目選定法を提案している．これらを適用したCTRON検定システムの構成と実現方法および検定実施結果を分析し，本提案が当初狙いとしたAPの移植性確保に効果があることを示す．

OS Interface Validation System

Ichiro Takenaka,† Hideo Oda†† and Masaharu Morihiro††

Validation (or Conformance testing) of Operating System (OS) interface specifications is a key to assure portability of application programs (AP). Validation services for OS such as POSIX and Sigma OS employ function test method in which conformance to standard specifications is checked by executing a set of test programs called test suite on the OS under test. However, this method can not detect functions excessive to the standard specifications. Therefore, the method is not enough for validation of an OS interface specification which does not allow the existence of excessive functions in an implementation. Another issue to be discussed in the function test method is what the pratically sufficient level of testing accuracy is. In this paper, the document test method which enables the detection of excessive functions is employed. Also introduced here is a definition of testing accuracy levels for test suite together with a method to uniquely design test cases for each of 16 levels defined here. The proposed methods are applied in the CTRON validation system. The data show that the methods are effective to assure AP portability.

1. はじめに

アプリケーションプログラム (AP) の移植性，相互運用性等の確保を目的として，APの開発と実行のための環境の標準化が行われている[1)~3)]．OS (Operating System) は，これら環境の中でも最も基本的な部分であり，標準化にむけて精力的や取り組みが行われている．代表的な例として，POSIX[4)]，TRON (The Realtime Operating system Nucleus の略)[5)]，シグマOS[6)] 等がある．APの移植性を保証するには，インタフェースの標準化を行うとともにOS製品が標準仕様に準拠していることを検証する必要がある．これがいわゆる検定 (Validation または Conformance Testing) であり，上記のOSについて，一部既に検定サービスが実施されている．POSIXについては米国のNIST[7)]，TRONについてはトロン協会[8)]，シグマOSについては(株)シグマシステム[9)]がそれぞれの認証機関としてテストスイートの提供，検定の実施，合格製品リストの発表等の業務を実施している．

従来のOS検定においては，検定対象であるOS上でテストスイートを起動，順次システムコールを実行し，標準仕様どおりのレスポンスがあるか否かを確認

† 日本電信電話(株) NTT交換システム研究所 伝達ソフトウェア研究部
Transport Switching Software Laboratory, NTT Communication Switching Laboratories, Nippon Telegraph and Telephone Corporation
†† 日本電信電話(株) NTT 情報通信網研究所 網オペレーション研究部
Network Operations Laboratory, NTT Network Information Systems Laboratories, Nippon Telegraph and Telephone Corporation

する方法（以下，機能検定と呼ぶ）を採用している．しかしながら，この方法では原理的に標準仕様以外の機能の存在（機能過多）を検出することはできない．POSIX，シグマ OS の場合は機能過多を許容しており，この方法で問題ないが，例えば CTRON（Cは“Communication”，“Central”を表す）[10),11)] のように機能過多を許容しないインタフェース規定の場合は，これだけでは不十分である．

プログラムのテスト技法は，その目的によって大きく内部仕様テストと外部仕様テストに分けることができる[12)]．本稿で対象としているインタフェース検定は外部仕様テストとして位置づけることができる．

外部仕様テストの良否を決めるのはテスト項目の選定方法である．仕様が有限状態マシン（FSM）の形で厳密に定義されている場合のテスト項目の選定方法については，すでに多くの研究がなされており，文献13) に詳しい．また形式仕様記述言語，例えば ISO の Estelle と LOTOS および CCITT の SDL，によって記述されている場合には，テスト項目も機械的に設定することが可能と考えられ，これらの言語を使った仕様記述の試みがなされている[14),15),27)]．しかしながら，形式仕様記述言語は厳密ではあるが抽象的で，わかりにくく，必要性は認識されていながら，現場ではあまり使われていない．現状では外部仕様の記述は自然言語によるのが一般的である．

このような背景から，テストスイート仕様の記述方法として表明（アサーション）による方法が採用されている[16)]が，表明自身が自然言語のため曖昧さを完全に除去することは難しい[17)]．このためテストの精度がテスト開発者の解釈に依存し，システム全体にわたった均質なテストが難しいという問題がある．また外部仕様テストにおいては，原理的にはすべての入力条件の組合せをテスト項目とすれば，完全なテストが可能であるが，組合せ数が膨大となり現実的ではない．実用上どこまで簡略化できるかが問題となる．

これらの問題に対処するため，仕様をある程度モデル化して図表等で表現することにより客観的で精度の高いテスト項目を作成する方法が知られている．代表的な方法としては，原因結果グラフ[18),19)]，AGENT 技法[20)]，実験計画法による技法[21)]がある．原因結果グラフ（CEG）技法と AGENT 技法では文章表現の仕様を論理的に表現可能なグラフや図式（決定表・状態表）に変換し，一定のアルゴリズムでテスト要因の組合せを最小化しようとするものである．実験計画法によるものは，テスト要因を実験計画法の直交表に対応させることにより，少ない組合せで均質なテスト項目を設定しようとするものである．実験計画法の組合せ表を適用するために，因子（ここではパラメータに相当）の取り得る値の数を素数のべき乗に揃える必要がある．これらの方法はいずれも，規模の大きいプログラムのテストには適用が難しい．

これに対し，本論文で提案するテスト項目選定方法は，OS のシステムが入力パラメータと実行結果によって規定されていること，および同一システムコールのパラメータ相互間の独立性が高い[8)]という性質を利用して，さらに簡略化を図ったものと位置づけることができる．

本論文では，(1)機能過多による AP 流通性の阻害を防止する方法として，ドキュメント検定を機能検定と併用する．(2)検定精度の定量的評価尺度の一案として，システムコールの入力パラメータ値の組合せに基づく 16 のレベルを定義し，レベルに応じたテスト項目選定法を提案している．ついで CTRON 検定に本提案を適用した場合の具体的検定方法および検定システムの構成について述べる．最後に実際の CTRON 検定で得られたデータの分析を行い，提案した手法が所期の目的である AP の移植性確保のみでなく，仕様自身の不具合の早期発見にも有効であることを示す．

2. OS インタフェース検定の位置づけ

2.1 OS インタフェースの標準化

ソフトウェア危機の叫ばれている今日，ソフトウェアの生産性を上げる最も効果的な方法として，システム間でのソフトウェア資産の共用すなわちソフトウェアの移植性の確保が重要な課題となっている．既にいくつか OS について，OS-AP 間のインタフェースの標準化が行われている．代表的な例として，IEEE および ISO の POSIX，トロン協会のトロン仕様 OS，(株)シグマシステムのシグマ OS 等が挙げられる．特に最近では，OS のみでなく AP の開発・実行に関係する言語，データベース，通信制御等すべてのインタフェースを含む API (Application Program Interface) として標準化を行う方向であり，OS と AP とのインタフェースはその中心的な要素として位置づけられている（**図 1**）．

2.2 OS インタフェース検定の定義

図 2 に OS インタフェースの標準仕様と実現された OS の仕様との関係を示す．α は標準仕様に規定さ

れているが，当該 OS で実現されていない機能，γ はオーバスペックの機能である．OS インタフェース検定とは，与えられた OS が特定の標準仕様の要求条件を満たしていることを検証することであり，以下の2つに分類される．

（1） 厳密準拠 (strictly conforming)：$\alpha=\phi$ かつ $\gamma-\phi$.

（2） 準拠 (conforming)：$\alpha=\phi$.

3. OS インタフェース検定における技術的課題

3.1 γ 機能に起因する流通性阻害の防止

従来の検定では，AP と同レベルのテストスイートを実際に OS 上で実行させ，システムコールを順次発行し，OS からのレスポンスおよび状態変化が仕様書に記述されたとおりかどうかを自動的に確認する方法（機能検定）が用いられている．

機能検定においては原理的には図2における α，β に対してそれぞれ NG，OK という形で検証することができるが，γ の有無を判定することはできない．γ 機能が AP に混入するのを防ぐ方法として，以下が考えられる．

【直接法】 OS のプログラムの内容を調べ，γ 機能をチェックする方法．

a） OS ソースプログラム解読方式：検定対象 OS のソースプログラムを解読し，γ 機能の無いことを確認する．

b） OS 走行ルート動的追跡方式：テストスイート実行時に，OS プログラムの実行番地を記録し，全スイート終了時の未実行番地に相当するソースプログラムをチェックし，γ 機能の無いことを確認する．

【間接法】 OS 自身には γ 機能が存在しても，AP のプログラマが使用しなければ，AP の流通性の低下が防げるという考え方に基づく方法．

c） 個別マニュアルチェック方式：OS のユーザマニュアルをチェックし，γ 機能に関する記述が無いことを確認する．マニュアルに記述がなければ使われない．

d） 標準マニュアル方式：OS 個々のマニュアルを作成せず，標準仕様そのものをマニュアルとする．

それぞれに長所，短所があるが，a）については効率，コスト面で，b）については OS の走行するハード機種ごとにトレーサが必要となりコスト面で実現は困難である．現状では間接法の c），d）いずれかの方法が現実的な解といえる．

3.2 テスト項目の選定

OS インタフェース仕様書では，システムコール対応に入力パラメータと実行結果とが規定されている．すべてのパラメータの組合せを網羅的にテストするのが理想的であるが，それには膨大な時間とリソースが必要であり，実用性の観点から適当なレベルを設定する必要がある．

そこで，テストレベルとして，入力パラメータのすべての組合せについてチェックするレベル (L_4)，注目する一つのパラメータ以外のパラメータの値を固定して，注目パラメータの取り得るすべての値についてチェックした後，順次注目パラメータを変えながらすべてのパラメータをチェックするレベル (L_3)，各パラメータの値を最低一度だけチェックする (L_2)，システムコールの特長的機能を確認するため一回だけチェックを行うレベル (L_1) を定義する．

一般にパラメータの取り得る値の数は，極めて大きいものもあるが（例えば，‘数値で −32767〜+32767），同値分割法の考え方[22]すなわち，仕様から判断して，プログラム中で同一の手順で処理されると思われるデータについては，同値クラスに組み入れ，その中の任意の値で代表させる方法，を採用することにより大

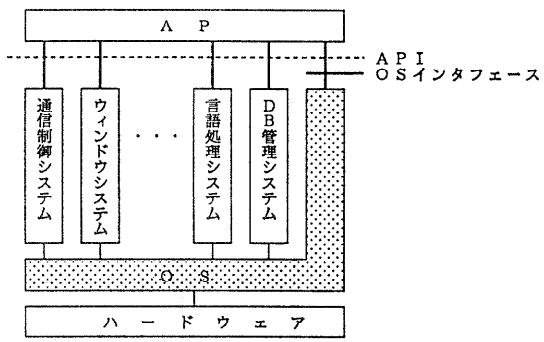


図 1 API の標準化
Fig. 1 Standardization of API.

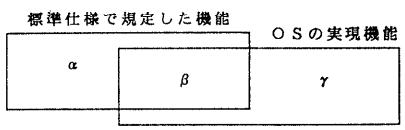


図 2 標準仕様と実現仕様の関係
Fig. 2 Standard and implemented specifications.

幅に削減することができる．また，限界値分析法[22]の考え方により処理の特異点のチェックを行う．

上記の例では，同値クラスとして，$x<-32767$, $-32767<x<+32767$, $+32767<x$ の3つの区間の任意の値 x，限界値として，-32767, $+32767$ の2つの値の計5個がチェック対象となる．

いま，k 個のパラメータ P_1, $P_2 \cdots P_k$ を持つシステムコールを考える．各パラメータ P_i の取り得る値の数を p_i とすると，レベル m におけるシステムコール当たりのテスト項目数 $N(L_m : m=1\sim4)$ は以下の式で表すことができる．

$$N(L_1)=1.$$

$$N(L_2)=\max\{p_1, \cdots, p_k\}.$$

$$N(L_3)=\sum_{i=1}^{k}\{p_i\}-(k-1).$$

$$N(L_4)=\prod_{i=1}^{k}\{p_i\}.$$

図3に示す $k=3$ の例で説明する．$N(L_1)$, $N(L_4)$ についてらは自明のため省略する．レベル2の場合，すべてのパラメータの値を最低1回とる組合せの例としては，$\{p_{11}, p_{21}, p_{31}\}$, $\{p_{12}, p_{22}, p_{31}\}$, $\{p_{13}, p_{23}, p_{32}\}$, $\{p_{13}, p_{24}, p_{32}\}$ の4個が求められる．レベル2の条件を満足する組合せはほかにも考えられるが，組合せの数は常に $\max(p_1, p_2, p_3)$ に等しくなる．レベル3の場合は，まず P_2, P_3 を標準的な値（例えばデフォールト値，p_{i1} で表す）p_{21}, p_{31} に固定したまま，p_1 を変化させることにより，$\{p_{11}, p_{21}, p_{31}\}$, $\{p_{12}, p_{21}, p_{31}\}$, $\{p_{13}, p_{21}, p_{31}\}$ の3個の組合せを得る．次に，P_1, P_3 を標準的な値 p_{11}, p_{31} に固定して P_2 を変化させることにより，$\{p_{11}, p_{21}, p_{31}\}$, $\{p_{11}, p_{22}, p_{31}\}$, $\{p_{11}, p_{23}, p_{31}\}$, $\{p_{11}, p_{24}, p_{31}\}$ の4個の組合せを得る．同様に P_3 について $\{p_{11}, p_{21}, p_{31}\}$ $\{p_{11}, p_{21}, p_{32}\}$ の2個の組合せを得る．こうして得られた9個の組合せから $\{p_{11}, p_{21}, p_{31}\}$ の重複分2を差し引くことで，独立な組合せの数7を得る．

一般に，システムコールのパラメータの取り得る値には，正常値と異常値がある．それぞれに対し上記のレベルを対応させ，組み合わせることによって L_{ij} が定義できる（ただし i は正常値に対するレベル，j は異常値に対するレベル）．すなわち，L_{11} から L_{44} まで16のレベルを定義することができる．例えば，L_{11} は正常，異常値を含め1つのシステムコールに対し1回だけチェックするレベル，L_{22} は各入力パラメータの取り得る値（正常，異常値とも）に対し一度は必ずチェックするが，最小限のチェック回数ですませるレベル，L_{33} は1つずつパラメータを振りながらすべてのパラメータのとり得る値をチェックするレベル，L_{44} は正常値，異常値を含めてすべての組合せをチェックするレベルである．

L_{ij} において，$i>j$ の場合は正常値に重点をおいたチェック，逆に $i<j$ の場合は異常値に重点をおいたチェックと言うことができる．各レベルにおけるシステムコール当たりのテスト項目数算出式を付録1に示す．

従来から知られているテスト項目選定技法である

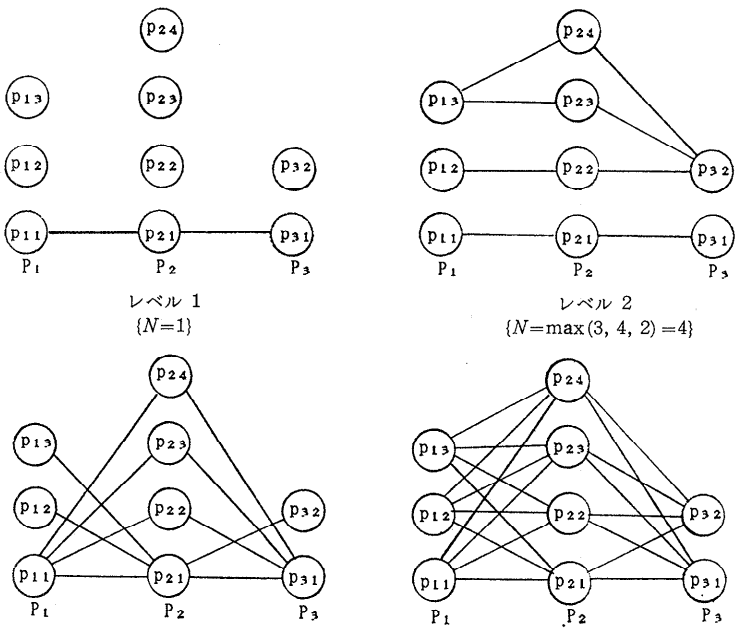


図3 テストレベルとテスト項目数（例）
Fig. 3 Test lebels and number of test cases.

CEG 法，AGENT 法，実験計画法による方法等は，いずれも入力パラメータ相互間の組合せを含めたテスト項目を選定するものであり，L_{33} と L_{44} の間に位置づけることができる．

POSIX 適合性試験においては，Exhaustive testing：L_{44} に相当，Thorough testing：L_{33} に相当，Identification testing)：L_{11} に相当，の3レベルを定性的に規定しており，テストスイートが満たすべきテストレベルとして L_{33} に相当する Thorough Testing を目標としている[16]．

4. CTRON インタフェース検定への適用[23),24)]

4.1 基本的考え方

POSIX をはじめ多くの OS では，検定合格製品に γ 機能（図2参照）の存在を容認している．しかし，この場合，あるベンダの CTRON 製品の γ 機能を使った AP は他ベンダの CTRON 厳密準拠の OS 上で動作しないことになり，いわゆるマルチベンダ環境下での AP の流通性が阻害される．このため CTRON 検定では γ 機能は認めないという方針が立てられた．厳密準拠の検定を実現するため，ドキュメント検定と機能検定の両方で検定を行い，ともに満足することを合格の条件とした．

4.2 ドキュメント検定システム

CTRON においては，マルチベンダ環境を想定しており，ベンダ間の競争を通じた製品のレベルアップをめざしている．マニュアルも OS 製品の一部として位置づけ，理解性，正確性等の品質面での競争を重視している．このような観点から，γ 機能の検出方法として，前述の間接法 c）個別マニュアルチェック方式を採用した．以下「ドキュメント検定」と呼ぶこととする．

CTRON インタフェース仕様書と被検定 OS のユーザ向けマニュアルの記述の対応関係をチェックするために，ドキュメント検定チェックシート（**図 4**）とその記入要領を作成した．受検者は，記入要領に従いチェックシートに対応関係を記入するとともに，チェックシートにより抽出されたマニュアル上の γ の部分について，理由書を作成する（**図 5**）．検定者はマニュアルの記述すべてについてインタフェース仕様書との対応関係がつけられているか，また γ の部分に対する理由書がすべて出されているかをチェックする．チェックシートと理由書の両方について審査を行い，NG 項目のすべてについて最終的に以下の項目に分類する．

（1） 検定対象であるマニュアルの修正が必要な項目．

ｻﾌﾞｾｯﾄ	共通部	分類	CREATE_TASK	ｺｰﾄﾞ	KC-CA01	ﾍﾟｰｼﾞ	1/4

区分	項番	検定項目	P	V/P	合否	備考
名称	0001	CREATE_TASK [(task-name)],(program_name) ・ ・ ・	15	▒		
機能	0002	タスクを生成する	15	▒		
	0003	タスクの固有空間を生成・初期化し，実行環境を整え，ドーマント状態にする．	15	▒		
	・ ・ ・	・ ・ ・		▒		

▒ ：受検者が記入

P ：インタフェース仕様書の記述ページ

V/P ：被検定ＯＳのユーザマニュアルでの記述位置(V:Volume,P:Page)

合否 ：検定者が判定結果を記入

図 4 ドキュメント検定チェックシートの構成
Fig. 4 Document validation checksheet.

(2) CTRON 仕様書自身の修正が必要な項目.

このうち，問題となるのは(1)であり，これがすべて修正されたことを確認した段階で，ドキュメント検定合格と判定する．(2)にたいしては CTRON 専門委員会に仕様変更の手続きをとる．

4.3 機能検定システム

4.3.1 検定精度とテスト項目の選定

$N(L_{ij})$ 算出式を実際の CTRON のシステムコール(基本 OS カーネル共通部：84 システムコール) に適用した場合の，各レベルにおけるテスト項目数を以下に示す．ただし正常値，異常値とも均等にチェックする $i=j$ の場合のみを示す．

$N(L_{44})=10,729$

$N(L_{33})=$ 489

$N(L_{22})=$ 378

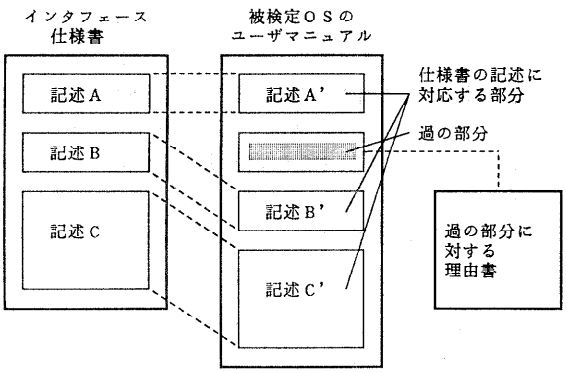


図 5 ユーザマニュアルのチェック
Fig. 5 Validation of OS user manual.

$N(L_{11})=$ 84

CTRON においては，システムコールの仕様設計に際して，パラメータ相互が独立となるよう考慮されていること[8]，すべてのパラメータの値を均等にカバーし，かつプログラム規模も極端に大きくならないという点を考慮して，L_{33} レベルを機能検定のレベルとして採用した．L_{33} レベルに対応するテスト項目の抽出手順を付録 2 に示す．

4.3.2 入出力のないシステムへの対応

CTRON の適用範囲である組込み型のシステムの場合，入出力機能が備わっていないものが多い．CTRON 検定システムは，このようなシステムにおいても何らかの形で検定スイートをロードしたり，検定結果を出力しなければならない．

この条件を満たすため，検定システムは CTRON カーネルのシステムコールを使用して検定結果をメモリ上に書き出すのみとし，それをフロッピディスク等の記憶媒体に出力する機能としては，被検定 OS のデバグ環境として通常用意されているメモリダンプツール等を用いる方式とした．

4.3.3 機能検定システムの構成

機能検定システムは，被検定 OS のシステムコールが CTRON の規定どおりに動作することを確認するテストスイートとそれらの実行制御を行う管理プログラムとからなる (**図 6**)．また機能検定システムは多様なプロセッサ上で走行する必要があるため，ソースプログラム形式で提供し，それぞれのマシン環境ごとにコンパイル，リンクし，実行形式のロードモジュー

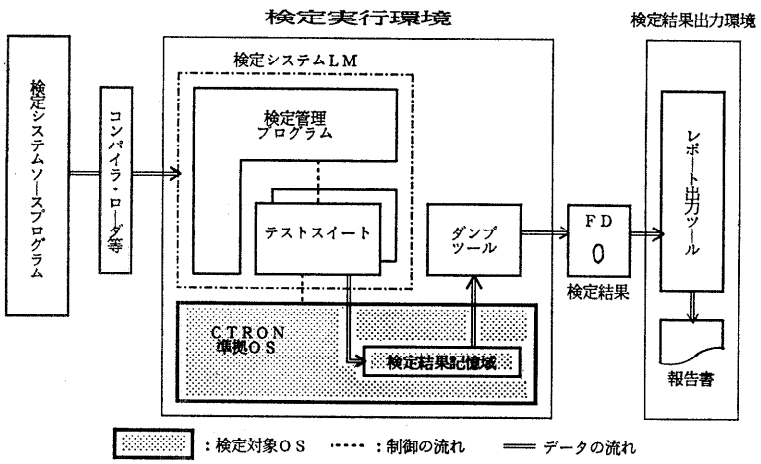


図 6 機能検定システムの構成
Fig. 6 Configuration of function validation system.

ル (LM) を作成する．コンパイル時オプションの指定によって，使用可能なメモリ容量に応じて検定システムの LM サイズを変更可能としている．

4.3.4 機能検定の流れ

上記機能検定システムを用いた機能検定の流れを以下に示す．

(1) 検定システムの LM を作成し，被検定 OS の AP 走行領域上にロードする．
(2) 検定管理プログラムを起動する．
(3) 同プログラムが検定結果記憶域を確保する．
(4) 同プログラムが指定されたテストスイートを選択実行する．
(5) テストスイートが検定結果 (OK, NG, NG の場合の理由等) をメモリ上の検定結果記憶域に書き込む．
(6) 検定システムの実行が終了する．
(7) ダンプツール等を用いて，検定結果をメモリから FD (フロッピディスク) に転送し，検定者に提出する．
(8) 検定者はレポート出力ツールを用いて検定結果レポートを作成する．

4.3.5 機能検定の合否判定

ドキュメント検定と同様に，検出された NG 項目を分析し，以下の項目に分類する．

(1) 検定対象 OS のプログラム修正の必要な項目
(2) CTRON 仕様書の修正が必要な項目

このうち，問題となるのは(1)であり，これがすべて修正されたことを確認した段階で，機能検定合格と判定する．(2)については仕様変更の手続きをとる．

5. 検定システムの評価

検定システムの効果を評価する場合，標準仕様に対する機能の過不足を検出することによる AP 移植性の確保という直接的効果と，検定を実施する過程で標準仕様の曖昧さ，不完全さを検出し，仕様自身の品質向上に貢献するという間接的効果の両面を考慮する必要がある．

筆者らが開発した機能検定システムおよびドキュメント検定システムは 1989 年より社団法人トロン協会において，CTRON の公式検定用システムとして採用され，1992 年 3 月末現在で C 言語バインディング基本 OS 12 件の検定を実施した．データの要約を**表 1**，**図 7** に示す．これらの分析により以下のような検定システムの効果が明らかとなった．

5.1 機 能 検 定

(1) 機能検定では，受検者が事前に検定システムを入手し，自社内で確認ののち正式な検定を受けるため，検定本番ではほとんど問題はでない．

(2) 機能検定による CTRON 仕様修正は全体で 3 件である．すべてがエラー処理関連のものであり，内訳は仕様書の記述が不明確：2 件，仕様誤り：1 件である．

5.2 ドキュメント検定

(1) ドキュメント検定により非準拠が検出され，マニュアル修正となったのは全体で 167 件 (検定当たり約 14 件) である．このうち完全な機能過多に起因するものは 1 件である．150 件 (約 90%) が，CTRON 仕様ではインプリメント依存となっている規定を，マニュアル上であたかも CTRON 規定項目のように記述しているためである．これを放置しておくと AP プログラマはこれらの機能を CTRON の標準仕様と思って使ってしまうことになり，その結果できあがった AP の移植性は保証されないことになる．すなわち，ドキュメント検定の目的である AP の流通性阻害の防止に効果があることが確認された．残り 16 件 (約 9%) は，字句の修正，表現の改良に類するものである．こ

表 1 CTRON 検定により指摘された問題件数
Table 1 Number of problems detected by CTRON validation.

項　　目	ドキュメント検定	機 能 検 定
検定実施回数	12	12
製品非準拠件数**	167(13.9)*	0(0.0)
CTRON 仕様修正件数	26(2.2)	3(0.3)

*：( )内は検定当たりの件数
**：ドキュメント検定の場合はマニュアルの修正，機能検定の場合は OS プログラムの修正

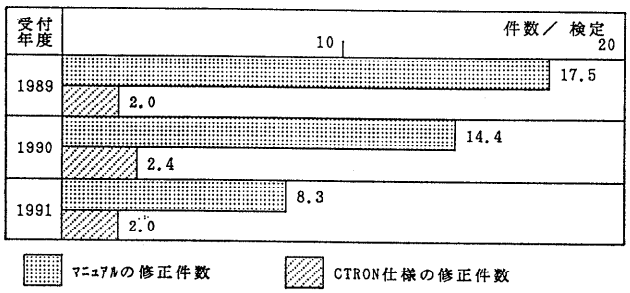


図 7 ドキュメント検定による修正件数の年度推移
Fig. 7 Change in the number of problems per validation.

れらの数値は，検定がメーカによる社内検査の後で行われたという事実と考え合わせると，検定の効果の大きいことを示している．

（2） ドキュメント検定における仕様書修正は26件である．CTRON仕様の変更はCTRON専門委員会で承認されたものが，随時チェンジズリストに登録され公開される．91年度までの3年間のチェンジズリストによる総変更件数は117件である．従って，ドキュメント検定の寄与率は0.22であり，仕様書自身の不具合の早期発見にも効果があった．

5.3 年度推移

（1） ドキュメント検定によるマニュアル修正件数は，年と共に減少している．これは仕様書自身の曖昧さが年と共に減ってきていることを反映している．すなわち，チェンジズリストにより仕様を継続的に改良し，これらを公開したことがマニュアル修正件数の減少に寄与しているためと思われる．

（2） 仕様修正はほぼ一定の割合で発生している．これらのほとんどは受検者がOSの開発段階で仕様の曖昧な部分に関して，トロン協会に確認することなく独自の判断でインプリメントし，検定時に仕様上の問題として検出されたものである．

5.4 移植性評価実験

検定の有効性を評価するには，実際に検定に合格したOS相互間でのAPの移植性を評価する必要がある．このため，メーカの異なる複数の検定済みCTRON基本OS上で同一の拡張OSプログラム（基本OSから見ればAPに相当）を実際に走行させて移植性を確認する実際がトロン協会によって実施された．その結果，極めて高い移植性が確認された[25),26)]．実験を通じて，検定のチェック漏れに起因する移植性阻害の報告はない．機能検定，ドキュメント検定とも比較的単純な方法ではあるが，その組合せにより，十分な精度の検定が実現できたということができる．

6. おわりに

OS検定に関する技術上の課題として，機能過多の問題，検定精度の定量的尺度の検討を行い，本論文で提案した方法を適用したCTRON基本OS検定システムの実現方法とその効果について述べた．本システムは1989年よりトロン協会におけるCTRON検定に使用されており，1992年3月末時点で12件の検定を完了している．本システムを用いることにより，検定対象OSおよびマニュアルの非準拠部分の早期検出によるAPの移植性確保という直接的効果に加え，CTRON仕様自身の曖昧さを指摘することにより，仕様自身の不具合の早期発見に貢献することができた．またトロン協会によって実施された移植性評価実験でも結果は良好であり，検定の当初の目的を十分達成した．

本論文で提案したテストレベル L_2, L_3, L_4 の概念は，テストカバレッジ技法における C_0, C_1, C_2 に類似の概念であり，OSに限らずパラメータの組合せによりインタフェースが規定される場合のテストの精度を，定量的に評価する際の尺度として利用することが可能である．

CTRON検定においては，これまでのところ L_{33} レベルでチェック漏れ等の問題は生じていないが，CTRONがさらに普及した段階で，より精度の高いテストレベルが求められる可能性もある．今後の課題として，L_3 と L_4 の間をさらに細かくレベル分けし，対応するテスト項目選定アルゴリズムを明らかにすることが残されている．γ 機能の検出については，間接的な方法としてドキュメント検定を試み，有効性を確認したが，数千項目にもおよぶチェックを人手で行うため，多大の工数を要するという問題があり，より効果的な γ 機能の検出方法の研究が待たれる．

その他の残された課題としては，（1）テストレベルの最適化，（2）既存テスト技法を用いた場合のテストレベルの定量的評価，（3）インタフェース仕様の形式的記述とテスト項目選定アルゴリズム等がある．

最後に，情報処理学会に1991年1月より，システムインタフェース検証研究グループが新設され，OS，言語処理系，プロトコル等のシステムインタフェースについて，テスト技術のみならず，制度上の問題も含めた新しい研究分野の模索が始まっている．本論文の提案するテストレベルの概念，γ 機能の検出法等がこの分野での検討のきっかけとなれば幸いである．

謝辞 CTRONインタフェース検定システムの開発および本論文のまとめに際し終始励ましと貴重なご意見を頂いた，NTT情報通信網研究所石野福弥博士，トロン協会CTRON専門委員会OS検定作業部会および検定委員会の方々，NTT研究開発技術本部河西宏之博士に深く感謝の意を表します．検定システムの評価に必要となった検定実施データを提供頂いたトロン協会のご好意に対し深く感謝いたします．論文推敲にあたり多くの貴重なコメントを頂いた査読委員の方々に深く感謝いたします．

参考文献

1) 毛内 健：ソフトウェア認証制度に関する国際動向，トロン季報，Vol. 1, No. 4, pp. 17-26，トロン協会 (1989).
2) 石田晴久，村井 純，小島富彦，井原 実：UNIXの将来性と課題，情報処理，Vol. 33, No. 1, pp. 76-92 (1992).
3) 米田 透：共通アプリケーション環境を実現する X/Open 仕様，NIKKEI COMPUTER 別冊 (1990. 7. 11).
4) IEEE Standard 1003.1 POSIX System Interface (1988).
5) 坂村 健：TRON の思想と今後，情報処理，Vol. 30, No. 5, pp. 522-531 (1989).
6) 通商産業省機械情報産業局監修：情報サービス産業白書，コンピュータ・エージ社 (1990).
7) NVLAP: Computer Applications Testing—POSIX Conformance Testing, NVLAP Program Handbook, NIST (1990).
8) 坂村 健：CTRON 概説，トロン協会編 (1988).
9) ΣOS の適合検査とは，Σ NEWS，シグマ協会，No. 10 (1988).
10) CTRON 仕様書カーネルインタフェース編，Ver. 1. 00. 00. 00，トロン協会 (1987).
11) CTRON 仕様書入出力制御インタフェース編，Ver. 1. 00. 00. 00，トロン協会 (1987).
12) 保田勝通：ソフトウェアの品質保証技術，電子情報通信学会誌，Vol. 73, No. 5, pp. 467-474 (1990).
13) Fujiwara, S., Bochmann, G. V., Khendek, F., Amalou, M. and Ghedamsi, A.: Test Selection Based on Finite State Models, *IEEE Trans. on SE*, Vol. 17, No. 6, pp. 59-603 (1991).
14) 大蒔和仁，二木厚吉：LOTOS に基づくプロトコルの形式記述，電子技術総合研究所彙報，Vol. 56, No. 9, pp. 51-63 (1992).
15) 藤原 洋，鈴木忠道，加藤英樹：知識ベース利用の SDL 支援システム，情報処理学会論文誌，Vol. 29, No. 5, pp. 497-505 (1988).
16) IEEE Standard for Information Technology-Test Methods for Measuring Conformance to POSIX, Std 1003. 3 (1991).
17) 林 健志，五十嵐滋：プログラムの検証理論，情報処理，Vol. 17, No. 5, pp. 437-447 (1976).
18) Myers, G. J.: *The Art of Software Testing*, Wiley (1976).
19) Yokoi, S. and Ohba, M.: TCG: CEG-Based Tool and Its Experiment, ソフトウェア信頼性シンポジウム（奈良）論文集，pp. 42-49 (1992).
20) 古川善吾，野木兼六，徳永健司：AGENT：機能テストのためのテスト項目作成の一手法，情報処理学会論文誌，Vol. 25, No. 5, pp. 736-744 (1984).
21) 石井康雄（編）：ソフトウェアの検査と品質保証，日科技連出版社 (1989).
22) 玉井哲雄，三嶋良武，松田茂広：ソフトウェアのテスト技法，共立出版 (1988).
23) 竹中市郎，小田英雄：CTRON インタフェース検定システム，NTT R&D, Vol. 38, No. 12, pp. 1539-1546 (1989).
24) Takenaka, I. and Oda, H.: CTRON Interface Validation System, *The 6th TRON Project Symposium* (*International*), pp. 171-181 (1989).
25) Ohta, T., Terazaki, T., Wasano, T. and Hanazawa, M.: Software Portability in CTRON, *The 8th TRON Project Symposium* (*International*), pp. 86-92, IEEE Computer Society Press (1991).
26) Nishimura, C., Iwata, N., Tanaka, T. and Nakayama, K.: Portability Experiment for CTRON General Program Management, *The 9th TRON Project Symposium* (*International*), pp. 137-145, IEEE Computer Society Press (1992).
27) 白鳥則郎，野口正一：OSI：開放型システム間相互接続（Ⅳ・完）—OSI 実装と試験・検証—，電子情報通信学会誌，Vol. 72, No. 2, pp. 209-216 (1989).

付録 1

テストレベル L_{ij} のテスト項目数算出式を以下に示す.

$$L_{11}=1.$$

$$L_{12}=1+\max\{e_i\}.$$

$$L_{13}=1+\sum_{i=1}^{k}\{e_i\}-(k-1).$$

$$L_{14}=1+\prod_{i=1}^{k}\{e_i\}.$$

$$L_{21}=\max\{n_i\}+1.$$

$$L_{22}=\max\{(n_i+e_i)\}.$$

$$L_{23}=\max\{n_i\}+\sum_{i=1}^{k}\{e_i\}-(k-1).$$

$$L_{24}=\max\{n_i\}+\prod_{i=1}^{k}\{e_i\}.$$

$$L_{31}=\sum_{i=1}^{k}\{n_i\}-(k-1)+1.$$

$$L_{32}=\sum_{i=1}^{k}\{n_i\}-(k-1)+\max\{e_i\}.$$

$$L_{33}=\sum_{i=1}^{k}\{(n_i+e_i)\}-(k-1).$$

$$L_{34}=\sum_{i=1}^{k}\{n_i\}-(k-1)+\prod_{i=1}^{k}\{e_i\}.$$

$$L_{41}=\prod_{i=1}^{k}\{n_i\}+1.$$

$$L_{42}=\prod_{i=1}^{k}\{n_i\}+\max\{e_i\}.$$

$$L_{43}=\prod_{i=1}^{k}\{n_i\}+\sum_{i=1}^{k}\{e_i\}-(k-1).$$

$$L_{44}=\prod_{i=1}^{k}\{(n_i+e_i)\}.$$

ここで

k：パラメータ数

n_i ($i=1\sim k$)：i 番目のパラメータの正常値（同値クラス）数

e_i ($i=1\sim k$)：i 番目のパラメータの異常値（同値クラス）数

付 録 2

CTRON 検定で採用した L_{33} レベルの場合のテスト項目選定アルゴリズムを以下に示す．

個々のシステムコールに対し，**表 2** に示すマトリクスを作成し，以下の手順でテスト項目を設定する．

①パラメータの取り得る正常値，異常値をすべて抽出して，表の縦軸を決める．

②特定のパラメータの特定の値に着目し●をつける．それ以外のパラメータはすべて正常値あるいは（省略可能パラメータの場合は）省略値を選び○をつけることにより，1列のパラメータの組合せを得る．これを1つのテスト項目として定義する．

表 2 L_{33} のチェックマトリクス
Table 2 Check matrix for L_{33}.

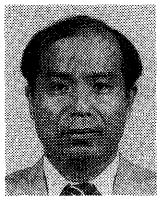
チェック条件				テスト項目ID				
内容			ID	1	2	3	4	
入力条件	ﾊﾟﾗﾒｰﾀA	0	01	●			○	
		1	02		●			
		異常値	03			●		
	ﾊﾟﾗﾒｰﾀB	"A"	04	◎	○	○		
		異常値	05				●	
	ﾊﾟﾗﾒｰﾀC	10	06					
		省略	07	◎	○	○	○	

●：着目点
◎：既にチェック済みの着目点
○：着目点に付随するパラメータ

③以下，着目点を順次下にずらせながら同様の手順を繰り返すことによりテスト項目を定義する．

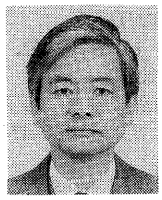
④着目点として選んだ行の左側に既に○がある場合には，既に着目済みと等価であり，スキップする．このことを表すために左側の○を◎に変更する．これにより同一のテスト項目の重複を除外する．

（平成4年4月16日受付）
（平成5年3月11日採録）



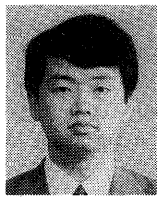
竹中 市郎（正会員）

昭和18年生．昭和41年九州大学工学部電子工学科卒業．昭和43年同大学院修士課程修了．同年日本電信電話公社武蔵野電気通信研究所入所．以来，オペレーティングシステムの開発，交換ソフトウェア開発支援システム，ソフトウェア品質保証等の研究に従事．現在，NTT 交換システム研究所主幹研究員．電子情報通信学会会員．



小田 英雄（正会員）

1948年生．1971年横浜国立大学工学部電気工学科卒業．1973年同大学院電気工学専攻修士課程修了．同年 NTT 入社，現在，情報通信網研究所主幹研究員．入社以来，主に PL/I 等の言語処理プロセッサの知究・実用化に従事．ソフトウェア品質管理に興味を持つ．電子情報通信学会会員．



森廣 政治

昭和40年生．昭和63年信州大学工学部情報工学科卒業．同年，日本電信電話(株)入社．以来，ソフトウェアの品質の研究，CTRON 基本 OS 検定システム，ソフトウェア開発管理システムの開発を経験．現在，IN サービスオペレーションシステムの研究開発に従事．電子情報通信学会会員．