マルチメディア通信と分散処理　62－17
（1993. 9. 21）

相互接続試験システムＡＩＣＴＳにおける試験スイートの検討

後藤邦弘　高橋健一　似内　聡　石幡吉則　高橋　薫

高度通信システム研究所（ＡＩＣ）

〒989-32 仙台市青葉区南吉成６丁目６番地の３

あらまし

開放型システム間相互接続技術の普及、発展と共に、その通信システムを構成する異機種端末間の相互接続試験の必要性が増している。そこで我々は、試験の自動化と汎用性を目指したアーキテクチャをもつ相互接続試験システムＡＩＣＴＳの構想をまとめ、現在その開発を進めている。本稿では、ＡＩＣＴＳにおける抽象試験スイートと実行型試験スイートについて述べる。ＡＩＣＴＳでは、抽象試験スイートはＴＴＣＮ．ＧＲ記述を用いて作成し、実行型試験スイートは状態遷移表により作成している。

和文キーワード　ＯＳＩ、相互接続試験、試験スイート、試験ケース、ＴＴＣＮ

Test Suite in the Interconnectability Testing System AICTS

Kunihiro GOTOH, Ken'ichi TAKAHASHI, Satoshi NITANAI,
Yoshinori ISHIHATA, Kaoru TAKAHASHI

Advanced Intelligent Communication System Laboratories (AIC)

6 - 6 - 3, Minami Yoshinari, Aoba-ku, Sendai, 989-32 Japan

Abstract

Interconnectability testing for a variety of terminal equipments developed in conformity with the same OSI (Open Systems Interconnection) standard is very important for the diffusion of OSI. We have been developing the interconnectability testing system AICTS (AIC's Inter-Connectability Testing System). This paper describes the abstract and executable test suites in AICTS. The abstract test suite in AICTS uses TTCN.GR and the executable test suite is realized by state transition table .

英文 key words　OSI, Interconnectability Test, Test Suite, Test Case, TTCN

1．はじめに

開放型システム間相互接続（ＯＳＩ：Open Systems Interconnection）の実現に向けてＩＳＯおよびＣＣＩＴＴは、プロトコル仕様及びサービス定義に関する国際標準化を行っている。しかし、これらはあるレベルで抽象化されたものであり、必ずしも実装に必要十分な細部仕様までを定義してはいない。そのため同一の標準に従って開発された情報通信端末であっても機種ごとに細部の仕様、設計は異なっている。このことは適合性試験 [1], [2] に合格していても相互接続が必ずしもできない理由の一つと考えられる。そこで異機種端末間の相互接続試験に対する認識が高まってきており、研究例もいくつか報告されているが、標準的・統一的なアーキテクチャは未だ確立されていない [3], [4], [5], [6]。

そこで我々は、相互接続試験アーキテクチャの確立を目指し、相互接続試験システムＡＩＣＴＳ（AIC's Inter-Connectability Testing System）の開発を進めている[7]。これまで我々はＡＩＣＴＳの開発において、その基本構成である適合性試験システムＡＣＴＳ（AIC's Conformance Testing System）の開発を行った [8], [9]。ＡＩＣＴＳは、ＡＣＴＳの開発資産の流用・機能拡張により開発を進めている。

本稿では、現在開発を進めているＯＳＩトランスポートプロトコルクラス２を試験対象とする相互接続試験システムＡＩＣＴＳにおける試験スイートについて報告する。

以下では、まず２章でＡＩＣＴＳの構成について述べ、次に３章で抽象試験スイートの構成・記述について述べる。次にこの抽象試験スイートから実行型試験スイートに変換する手順と実行型試験スイートの構成について４章で述べる。５章では、実行型試験スイートを用いた試験実行手順について述べる。

2．相互接続試験システムＡＩＣＴＳの構成

相互接続試験システムＡＩＣＴＳの構成を図１に示す。

ＡＩＣＴＳは、Ｔｅｓｔ Ｓｙｓｔｅｍ、ＳＵＴ (System Under Test)＃Ａ、ＳＵＴ＃Ｂに構成・分割される。Ｔｅｓｔ Ｓｙｓｔｅｍは、システム全体の制御・管理を行うＴＭ (Test Manager)、ＳＵＴ＃Ａとの中継機能を持つＲｅｐｅａｔｅｒ＃Ａ、ＳＵＴ＃Ｂとの中継機能を持つＲｅｐｅａｔｅｒ＃Ｂ、ＳＵＴ＃ＡおよびＳＵＴ＃Ｂから送られてきたデータを監視するＭｏｎｉｔｏｒにより構成され、ＳＵＴ＃ＡおよびＳＵＴ＃Ｂはそれぞれ試験対象実装（ＩＵＴ：Implementation Under Test）＃ｎ（ｎ：Ａ or Ｂ）、ＩＵＴの上位テスタ（ＵＴ：Upper Tester）＃ｎ－１およびＵＴ＃ｎ－２、２つのＵＴの制御・管理を行うＵＴＭ (Upper Tester Manager)＃ｎにより構成される。ここでＵＴは、ＩＵＴの提供する複数のトランスポートコネクションを試験するために２つ設けている。また、ＵＴを自発的に動作させることにより、Ｔｅｓｔ Ｓｙｓｔｅｍにかかる負荷の軽減と制御・観測点の明確化をはかっている。

制御・観測点（ＰＣＯ：Point of Control and Observation)は、ＩＵＴの上位にそれぞれ２つ設けている。

２つのＩＵＴは、下位層（Lower Layer）を介して直接試験チャネル（Channel Under Test）により接続される。これにより実運用に近い構成での相互接続試験が可能である。ＡＩＣＴＳでは試験チャネルとは別に、試験管理用のＴＭＣ (Test Management Channel)＃ＡとＴＭＣ＃Ｂを独立して設けている。これにより試験の信頼性の向上が期待できる。

試験は、試験管理プロトコル（ＴＭＰ：Test Management Protocol）により規定された試験管理メッセージ（ＴＭＭ：Test Management Message）をＴＭＣを通して各ＵＴとＴＭが送受信することによりＴｅｓｔ ＳｙｓｔｅｍとＳＵＴの協調を取りながら実行される。

3．抽象試験スイート

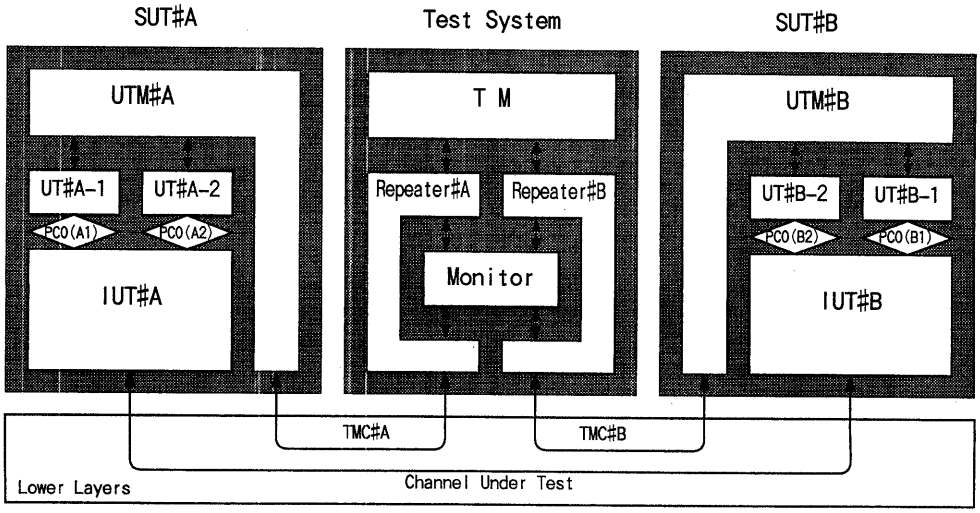


図１　ＡＩＣＴＳの構成

ＴＴＣＮ (Tree and Tabular Combined Notation) [10] は、適合性試験の方法と枠組みに従って、抽象試験スイート（ＡＴＳ：Abstract Test Suite）を記述するための言語として国際標準化されたものであり、試験に必要な試験目的、試験の条件といったドキュメント的な性質のものから試験データの定義、試験シーケンスの定義といったものを併せて記述できるようになっている。また、ＴＴＣＮにはデータの定義やシーケンスの定義に表形式を用いるＴＴＣＮ．ＧＲ（GRaphical form）とテキスト表現を用いるＴＴＣＮ．ＭＰ（Machine Processable form）の２つがあり、両者は同じ意味を表現することができる。

我々は、相互接続試験システムＡＩＣＴＳ用ＡＴＳを適合性試験システムＡＣＴＳ[9] の場合と同様にＴＴＣＮ．ＧＲの表記法を用いて開発を行っている。

相互接続試験は、適合性試験とは目的が異なっており、適合性試験が１つの試験対象に対してのプロトコル中心の試験であるのに対し、２つの試験対象が提供するサービス中心の試験である。このため、試験スイートもプロトコルの振る舞いを観測するのではなく、正しくサービスが提供されることを確認するものが要求される。

３．１　試験スイートの構成

ＡＩＣＴＳの試験スイートの構成を図２に示す。試験スイートは、試験グループ、試験ケースによる階層構造をしている。ＡＩＣＴＳでは、トランスポートプロトコルクラス２を試験対象としており、試験スイートをまずＩＵＴが提供するトランスポートコネクションにより、単一コネクションと複数コネクションの試験グループに分類し、次に通信フェーズによりコネクションの確立と解放、データ転送の試験グループに分類、最後にデータ転送を普通データ転送と優先データ転送の試験グループに分類している。

３．２　試験スイート記述

試験ケースには、ＰＣＯに対する制御・観測内容を記述し、その結果として予測されるＩＵＴの動作に対して標準に従った判定を記述する。判定は、通過（pass）、失敗（fail）のいずれかである。ＡＩＣＴＳでは、ＰＣＯをＩＵＴの上位にのみ設けている。このため、試験に失敗した場合、失敗箇所の特定ができない（ＩＵＴ＃Ａ or ＩＵＴ＃Ｂ or 下位レイヤ）。ＡＩＣＴＳでは、ＩＵＴは適合性試験に合格していること、下位レイヤは完全であることを前提とし、不確定（inconclusive）という判定は設けていない。

表１にＰＣＯの存在位置を定義したＰＣＯ宣言、表２にタイマの種類を定義したタイマ宣言の記述例を示す。この例では、ＵＴ＃Ａ－１でＰＣＯ（Ａ１）を制御・観測し、イベント受信用のタイマとしてタイマ（Ａ）を使用している。ＵＴ＃Ｂ－１においても同様にＰＣＯ（Ｂ１）とタイマ（Ｂ）を使用している。

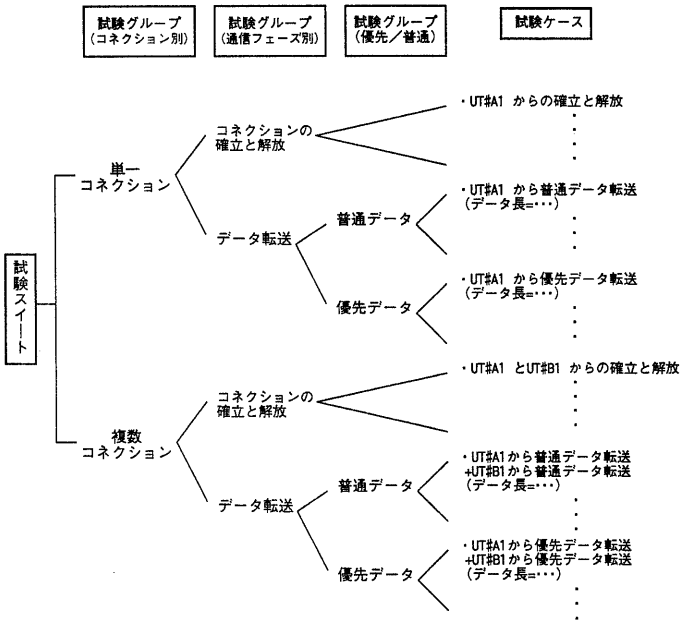


図２　試験スイートの構成

また、図３にＵＴ＃Ａ－１からＵＴ＃Ｂ－１へのトランスポート（Ｔ：Transport）コネクションの確立を試験するための抽象試験ケース（ATC ：Abstract Test Case）の例を示す。記述内容としては、上部に試験ケース名、試験グループ名、試験目的、デフォルト動作の参照等が記述され、下部に試験の動作が記述される。ここで、試験グループ名

表１　ＰＣＯ宣言記述例

PCO Declaration			
PCO Name	PCO Type	Role	Comments
Ａ１	TSAP	UT#A-1	ＵＴ＃Ａ－１のＴＳＡＰ
Ｂ１	TSAP	UT#B-1	ＵＴ＃Ｂ－１のＴＳＡＰ

TSAP : Transport Service Access Point

表２　タイマ宣言記述例

Timer Declaration		
Timer Name	Unit	Comments
Ａ	sec	ＵＴ＃Ａ－１用受信タイマ
Ｂ	sec	ＵＴ＃Ｂ－１用受信タイマ

Test Case Name : TEST001					
Group : AICTS/TCON1/CONNECT/					
Purpose : ＵＴ＃Ａ－１からＵＴ＃Ｂ－１へトランスポート コネクションの確立ができることを確認する。					
Default : Def1					
Comments :					

Nr	Label	Behaviour Description	Refer	Verdict	Comments
01		Start(B)			(1)
02		A1!T_CON_REQ	CONreq1		(2)
03		Start(A)			(3)
04		B1?T_CON_IND	CONind1		(4)
05		Cancel(B)			(5)
06		B1!T_CON_RES	CONres1		(6)
07		Start(B)			(7)
08		A1?T_CON_CNF	CONcnf1	PASS	(8)
09		Cancel(A)			(9)
10		+Pos1/Postamble			(10)
11		?Timeout(A)		FAIL	(11)
12		+Pos1/Postamble			(10)
13		?Timeout(B)		FAIL	(12)

図３　抽象試験ケースの例

は、図２の試験スイートの構成に従って付けられる。なお、動作記述部の基本的な記述法を以下に示す。

＜ＰＣＯ＞！ＡＳＰ名（ASP : Abstract Service Primitive）

＜ＰＣＯ＞？ＡＳＰ名

‘！’はＵＴからの送信を表し、‘？’はＵＴでの受信を表す。以下に内容の概要をコメントの番号毎に示す。

（１）ＵＴ＃Ｂ－１の受信タイマー（Ｂ）を起動する。

（２）ＵＴ＃Ａ－１からＩＵＴ＃Ａにトランスポートコネクション確立要求（T_CON_REC）サービスプリミティブを送信する。

（３）ＵＴ＃Ａ－１の受信タイマー（Ａ）を起動する。

（４）ＵＴ＃Ｂ－１でＩＵＴ＃Ｂからのトランスポートコネクション確立指示（T_CON_IND）サービスプリミティブを受信する。

（５）ＵＴ＃Ｂ－１の受信タイマー（Ｂ）をリセットする。

（６）ＵＴ＃Ｂ－１からＩＵＴ＃Ｂにトランスポートコネクション確立応答（T_CON_RES）サービスプリミティブを送信する。

（７）ＵＴ＃Ｂ－１の受信タイマー（Ｂ）を起動する。

（８）ＵＴ＃Ａ－１でＩＵＴ＃Ａからのトランスポートコネクション確立確認（T_CON_CNF）サービスプリミティブを受信すれば、判定結果は［ＰＡＳＳ］となる。

（９）ＵＴ＃Ａ－１の受信タイマー（Ａ）をリセットする。

（１０）後処理部であり、トランスポートコネクションの解放を行う。

（１１）ＵＴ＃Ａ－１の受信タイマー（Ａ）がタイムアウ

Test Case Name : Pos1/Postamble					
Group :					
Purpose : ＵＴ１からトランスポートコネクションの解放を行う。					
Defaults : Def1					
Comments :					

Nr	Label	Behaviour Description	Refer	Verdict	Comments
01		A1!T_DIS_REQ	DISreq1		
02		Start(A)			
03		B1?T_DIS_IND			
04		Cancel(B)			
05		?Timeout(A)			
06		?Timeout(B)		FAIL	
07		B1!T_DIS_REQ	DISreq1		
08		Start(B)			
09		A1?T_DIS_IND			
10		Cancel(A)			
11		?Timeout(B)			
12		?Timeout(A)			
13		?Timeout(B)			

図４　後処理動作(Pos1/Postamble)記述例

トになれば、判定結果は［ＦＡＩＬ］となる。

（１２）ＵＴ＃Ｂ－１の受信タイマー（Ｂ）がタイムアウトになれば、判定結果は［ＦＡＩＬ］となる。

また、図４に後処理動作（Pos1 / Postamble）の記述例を示す。この例では、ＵＴ＃Ａ－１からトランスポートコネクションの解放を行う。正常に解放されれば終了し、解放されない場合は、ＵＴ＃Ａ－２からあらためてトランスポートコネクションの解放を行う。

また、他の試験ケースについても同様に記述される。

４．実行型試験スイート

抽象試験スイートは、試験システムによる試験実行用ではないので、ＡＴＳの抽象的な部分や省略されている部分を具現化し、試験システムが処理できる実行型試験スイート（ＥＴＳ：Executable Test Suite）を作成する必要がある。そのため、ＥＴＳ開発者はＡＴＳおよびプロトコル仕様を参照し、エディタ等によりＥＴＳを作成してファイルとして保存する。試験ではテストスイートローダがそのファイルをテストドライバのメモリ上に展開し、それをテストスイートインタプリタが操作しながら各部の制御を行い、試験を実行する。（この試験実行方式を試験スイートインタプリタ方式と呼ぶ。）

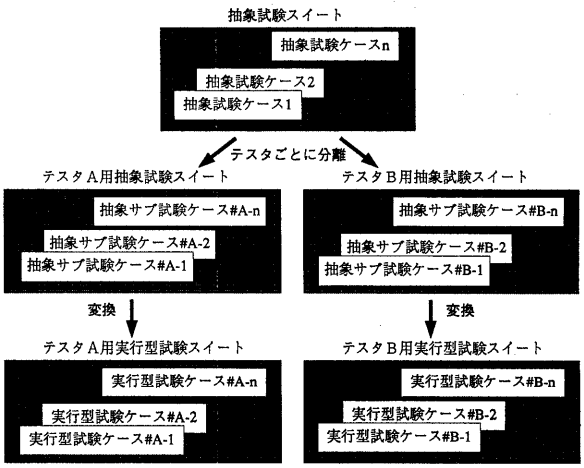


図５　実行型試験スイートの導出

４．１　実行型試験スイート導出手順

ＡＩＣＴＳでは、テスタをＩＵＴの上位に分散配置し、独立に動作させることにより試験を実行する。このため実行型試験スイートもＵＴごとに用意する必要がある。抽象試験スイートは、図５に示す手順により実行型試験スイートに変換される。

Test Case Name	:TEST001/A1		
group	:AICTS/TCON1/CONNECT/		
Purpose	:ＵＴ＃Ａ－１用抽象サブ試験ケース		
Default	:DefA1		
Nr	Behaviour Description	Refer	Verdict
01	A1!T_CON_REQ	CONreq1	
02	Start(A)		
03	A1?T_CON_CNF	CONcnf1	PASS
04	Cancel(A)		
05	+PosA1/Postamble		
06	?Timeout(A)		FAIL
07	+PosA1/Postamble		

Test Case Name	:TEST001/B1		
group	:AICTS/TCON1/CONNECT/		
Purpose	:ＵＴ＃Ｂ－１用抽象サブ試験ケース		
Default	:DefB1		
Nr	Behaviour Description	Refer	Verdict
01	Start(B)		
02	B1?T_CON_IND	CONind1	PASS
03	Cancel(B)		↑
04	B1!T_CON_RES	CONres1	付加
05	Start(B)		
06	+PosB1/Postamble		
07	?Timeout(B)		FAIL

Test Case Name	:PosA1/Postamble		
group	:		
Purpose	:ＵＴ＃Ａ－１用後処理		
Default	:DefA1		
Nr	Behaviour Description	Refer	Verdict
01	A1!T_DIS_REQ	DISreq1	
02	Start(A)		
03	?Timeout(A)		
04	A1?T_DIS_IND　削除		
05	Cancel(A)　↓		
~~06~~	~~?Timeout(A)~~		

Test Case Name	:PosB1/Postamble		
group	:		
Purpose	:ＵＴ＃Ｂ－１用後処理		
Default	:DefB1		
Nr	Behaviour Description	Refer	Verdict
01	B1?T_DIS_IND		
02	Cancel(B)		
03	?Timeout(B)		FAIL
04	B1!T_DIS_REQ	DISreq1	
05	Start(B)　削除		
06	?Timeout(B)　↓		
~~07~~	~~?Timeout(B)~~		

図６　抽象試験ケースの分離例

まずＴＴＣＮ．ＧＲ記述により試験系全体の動作を記述した抽象試験スイートをテスタごとの抽象試験スイートに分離する。これは、抽象試験ケースごとにテスタＡ用の動作記述とテスタＢ用の動作記述に分離することにより作成する（抽象サブ試験ケース#A-Xと抽象サブ試験ケース#B-X）。次に分離した抽象サブ試験スイートをそれぞれ試験ケースごとに実行型試験ケースに変換し、実行型試験スイートを作成する。

４．２　抽象試験ケースの分離

図６に図３および図４のＡＴＣをテスタごとに分離した例を示す。図３および図４のＡＴＣからＰＣＯ［Ａ１］またはタイマ［Ａ］の記述が含まれるイベントを抽出し、インデントを付け替えてＵＴ＃Ａ－１用抽象サブ試験ケース、ＰＣＯ［Ｂ１］またはタイマ［Ｂ］の記述が含まれるものを抽出し、インデントを付け替えてＵＴ＃Ｂ－１用抽象サブ試験ケースとする。また、「Ｐｏｓｔａｍｂｌｅ」についてもＵＴ＃Ａ－１用後処理(PosA1/Postamble)とＵＴ＃Ｂ－１用後処理（PosB1/Postamble）に分離する。ここで問題となるのは、元になるＡＴＣには、試験の総合判定が記述されるため判定結果［ＰＡＳＳ］が１つしかないことである。複数のＥＴＣにより試験を実行し、その試験の判定を行うためには、ＥＴＣごとの判定が必要となる。このため、分離した抽象サブ試験ケースにも、テスタごとの判定が必要となる。図３のＡＴＣのシーケンスを示すと図７のようになる。

（１）ＵＴ＃Ａ－１からＩＵＴ＃Ａに[T_CON_REQ]送信。
（２）ＵＴ＃Ｂ－１でＩＵＴ＃Ｂからの[T_CON_IND]受信。
（３）ＵＴ＃Ｂ－１からＩＵＴ＃Ｂに[T_CON_RES]送信。
（４）ＵＴ＃Ａ－１でＩＵＴ＃Ａからの[T_CON_CNF]受信。判定結果は［ＰＡＳＳ］。

ここで、ＵＴ＃Ｂ－１において（４）のイベントの直前の受信イベントは、（２）のイベントであるからこれをＵＴ＃Ｂ－１における［ＰＡＳＳ］判定のイベントとし、これを図６のＵＴ＃Ｂ－１用抽象サブ試験ケースに付加する。

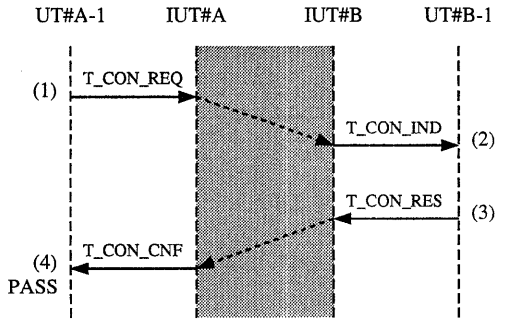


図７　試験シーケンス例

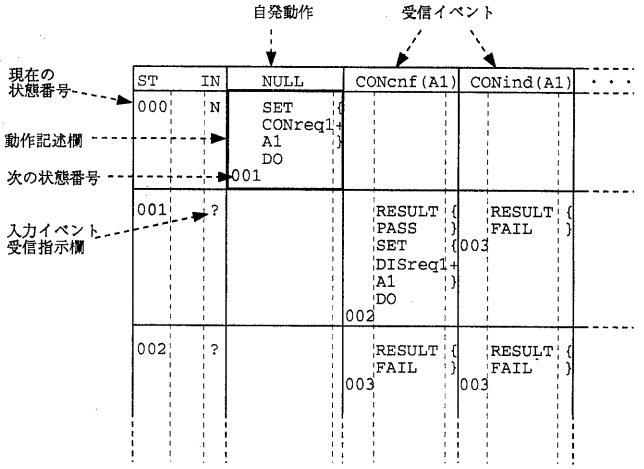


図８　実行型試験ケース記述例

また、ＴＴＣＮ記述を分離した場合には、余分な記述が現われることがある。ＴＴＣＮでは、同一時間に発生するイベントが複数考えられる場合、同じインデントの位置に複数のイベントが記述される。ＡＩＣＴＳのＡＴＣは、ＵＴ＃Ａ－１のイベントとＵＴ＃Ｂ－１のイベントが交互に記述されている。ここで、他方のイベントにかかわらず同じイベントが記述された場合には、そのイベントがＡＴＣを分離したときに複数現われることになる。具体的には、図４のＮｒ０５と１２、Ｎｒ１１と１３がこれに相当し、図６のＵＴ＃Ａ－１用後処理の試験ケースでは、同じインデントの位置にＴｉｍｅｏｕｔが２つ現われ、ＵＴ＃Ｂ－１用後処理の試験ケースでは、Ｔｉｍｅｏｕｔが２回繰り返されている。この余分な記述を削除し、ＡＴＣの分離は終了する。

４．３　実行型試験スイートの構造

ＡＩＣＴＳのＥＴＳとしては、我々が既に開発を完了した適合性試験システムＡＣＴＳ[9]と同様の状態遷移表である実行型試験スイートを使用する。図８に実行型試験ケース（ＥＴＣ：Executable Test Case）の例を示す。

状態遷移表の横軸は外部からの入力イベントまたは自発動作であり、縦軸は状態番号を表し、表の各欄にはその状態で実行するアクションを記述する。

ＡＩＣＴＳでは、ＩＵＴの上位にＰＣＯを設けているためＥＴＳにはトランスポートサービスプリミティブの交換手順が記述される。

表３、表４に動作記述欄と入力イベント受信指示欄に記述されるコマンドの種類を示す。なお、動作記述欄内の下位機能が実行するアクション（記号 'Conreq1'，'A1' など）は引数として扱われ、{ } 内で記述される。引数は、{ で始まり、+ で連結され、} で終了する。

表3　動作記述用コマンドの種類

コマンド	機能
SET	行うべきアクションの命令をメモリにセット
DO	アクション実行指示
END	試験終了
RESULT	結果判定

表4　入力イベント受信指示欄用コマンドの種類

コマンド	機能
?	監視タイマを起動し入力イベントを待つ
N	自発的動作

例えば、図8の状態遷移表の場合、テストスイートインタプリタは、初期状態［０００］、入力イベント［ＮＵＬＬ（自発動作を示す。）］において次の動作を自発的に行う。

（１）ＰＣＯ（Ａ１）に対して、コネクション確立要求のＴ－サービスプリミティブ（記号 'CONreq1'）の発行を下位機能に指示する（SET { CONreq1 + A1 } DO）。これは、ＡＴＣの［A1!T_CON_REQ］という記述に相当する。

（２）次の状態［００１］に遷移し、監視タイマ（この場合はタイマ（Ａ））を起動しイベントの入力待ちの状態（記号 '?'：表4参照）となる。これは、ＡＴＣの［Start (A)］［A1？･･･］という記述に相当する。このようにいくつかの動作を１つの記号で表すことによりＥＴＳのサイズの縮小化を図っている。

5　試験ケースによる試験実行手順

ＡＩＣＴＳにおける試験ケースと試験の実行との関係を図9に示す。

試験は、以下の手順で実行される。

（１）試験系全体の試験手順を記述したＡＴＳを作成する。

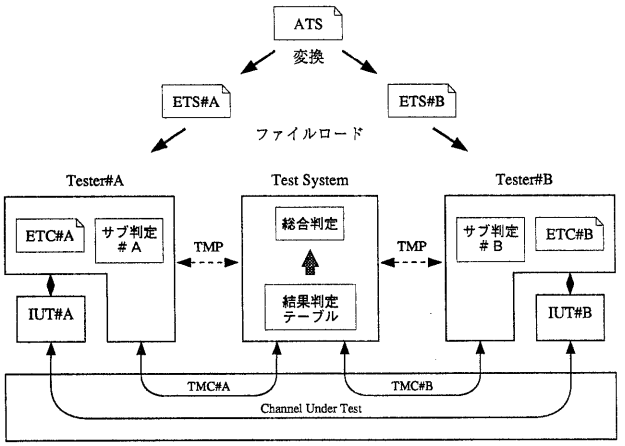


図9　試験ケースと試験実行

表5　試験結果判定テーブル

		テスタ#A判定		
	総合判定	Pass	Fail	Abort
テスタ#B判定	Pass	Pass	Fail	Abort
	Fail	Fail	Fail	Abort
	Abort	Abort	Abort	Abort

（２）抽象試験スイートをテスタごとのＥＴＳに分離・変換する。

（３）Ｔｅｓｔ Ｓｙｓｔｅｍは、実行する試験ケースの順序を定義した試験スケジュールに従って、各テスタに試験項目ごとのＥＴＣファイルをロードする。

（４）各テスタでは、Ｔｅｓｔ Ｓｙｓｔｅｍからの指示によりロードしたＥＴＣに従い同時に試験を開始する。

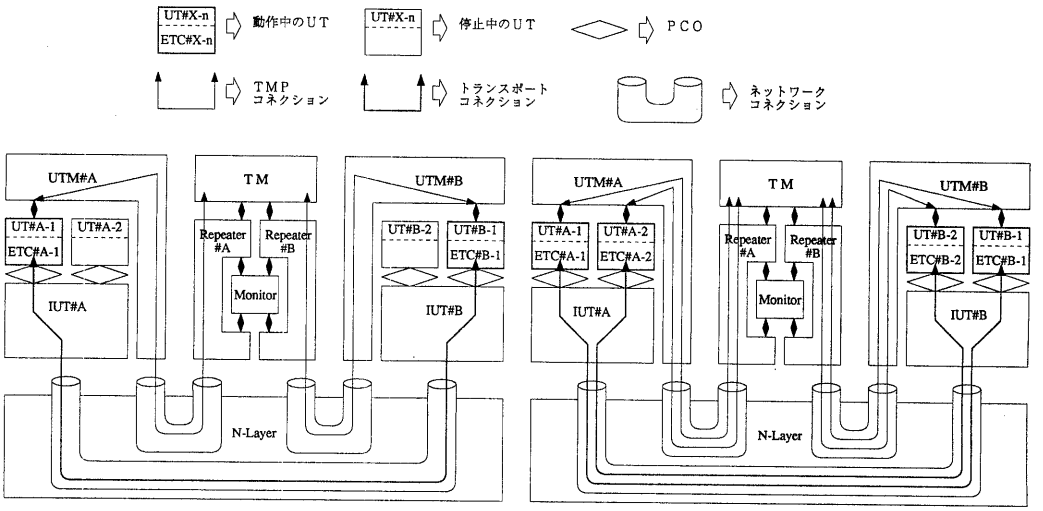
（５）試験終了後、各テスタでは、ＥＴＣの実行結果であるサブ判定を得る。判定結果は、通過（Ｐａｓｓ）、失敗（Ｆａｉｌ）、そしてＴｅｓｔ Ｓｙｓｔｅｍの異常等による試験の中断を示す（Ａｂｏｒｔ）がある。

（６）Ｔｅｓｔ Ｓｙｓｔｅｍでは、テスタごとのサブ判定と結果判定テーブル（表5）により総合判定を行う。判定ロジックは、両方のテスタの判定がＰａｓｓの場合のみ総合判定はＰａｓｓとなる。ＰａｓｓとＦａｉｌでは片方でも試験の失敗であるＦａｉｌであれば総合判定はＦａｉｌとなる。また、Ａｂｏｒｔはシステムの異常であり、正確な判定ができないので最優先される。

（７）試験スケジュールにある全ての試験項目に対して（３）から（６）の処理を繰り返し行う。

また、ＡＩＣＴＳにおける実行型試験ケースの配置を図１０に示す。実行型試験ケースは、ＩＵＴの上位テスタであるＵＴに配置される。

ＡＩＣＴＳでは、トランスポートプロトコルクラス２エンティティを実装するＩＵＴを試験対象とし、そのＩＵＴが提供する複数のコネクションに対しての試験が可能である。ＡＩＣＴＳでは、ＵＴをＩＵＴの上位に２つ設け、それぞれのＵＴに独立にＥＴＣを配置している。これにより、２つのＵＴが独立にトランスポートコネクションを確立し、それぞれのＰＣＯに対し制御・観測を行う。このように複数のコネクションに対しての試験は、２つのトランスポートコネクションで異なる試験を並行に行う。１つのトランスポートコネクションの両端のＥＴＣが一組となり試験を行う。単一コネクションで試験する場合（図１０（ａ））は、このＥＴＣを１本のトランスポートコネクション上でのみ実行し、複数のコネクションに対して試験する場合（図１０（ｂ））は、２本のトランスポートコネクション上で独立にこのＥＴＣを実行する。ＡＩＣＴＳでは、トランスポートコネクションごとにＥＴＣを用意しているため、



（a）単一コネクションで試験する場合　　（b）複数コネクションで試験する場合

図１０　ＥＴＣの配置とコネクションごとの試験

ＥＴＣの組み合わせを変えることにより多様な複数のコネクションに対しての試験を容易に実現できる。

６．まとめと今後の予定

本稿では、現在我々が開発を進めている相互接続試験システムＡＩＣＴＳに関し、試験スイートを中心に述べた。

ＡＩＣＴＳにおいて抽象試験スイートは、適合性試験システムＡＣＴＳと同様にＴＴＣＮ．ＧＲ記述により作成する。

実行型試験スイートは、ＡＩＣＴＳのシステム構成に合わせて抽象試験スイートをテスタごとの記述に分離し、それぞれ状態遷移表に変換し実行型試験スイートを作成する。試験は、試験スイートインタプリタ方式により実行される。

今後は、以下の点について研究を進める予定である。

（１）試験の網羅性を考慮しながら必要な試験項目を選定し、抽象試験スイートと実行型試験スイートを作成する。

（２）作成した試験スイートをＡＩＣＴＳに実装し、動作確認を行う。

（３）試験対象の下位ＰＣＯを制御・観測するためのシステム及び試験スイートの検討を行い、ＡＩＣＴＳの機能拡張を行う。

謝辞

本研究を進めるにあたって、有益なご助言を頂いた日本大学の野口正一教授、東北大学の白鳥則郎教授に深謝いたします。また、本研究の機会を与えて頂いたＡＩＣの緒方秀夫常務、小林主幹研究員に深謝いたします。

参考文献

[1] ISO/IEC : "Information technology - Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 1 : General concepts", ISO/IEC IS 9646-1 (1991).

[2] 若杉："ＯＳＩ適合性検証試験の現状", 情報処理学会マルチメディア通信と分散処理研究会 52-21, (1991.9).

[3] G.Bochman, R.Dssuli, J.R.Zhao : "Trace Analysis for Conformance and Arbitration Testing", IEEE Transactions on Software Engineering, Vol.15, NO.11, pp. 1347-1356 (1989.11).

[4] O.Rafig and R.Chastanet : "From conformance testing to interoperability testing", Proceedings of IFIP TC6 Third International Workshop on Protocol Test Systems, pp. 371-385 (1990).

[5] L.Lenzini , F.Zocconi : "Interoperability tests on OSI products in the framework of the OSIRIDE-Intertest initiative", Computer Networks and ISDN Systems 24 (1992).

[6] 中井他："相互接続試験アーキテクチャの構成法", 電子情報通信学会技術研究報告, SSE90-58 (1990.7).

[7] 高橋他："相互接続試験システムＡＩＣＴＳの機能検討", 情報処理学会マルチメディア通信と分散処理研究会 60-5, (1993.5).

[8] 高橋他："適合性試験システムＡＩＣＴＳの開発とその評価", 情報処理学会マルチメディア通信と分散処理研究会 59-2, (1993.1).

[9] 似内他："適合性試験システムＡＩＣＴＳにおける試験スイートの開発", 情報処理学会マルチメディア通信と分散処理研究会 59-3, (1993.1) .

[10] ISO/IEC : "Information technology - Open Systems Interconnection Conformance Testing Methodology and Framework - Part3 : The Tree and Tabular combined Notation", ISO/IEC 9646-3 (1991).