マルチメディア通信と分散処理　50－8
（1991. 5. 17）

マルチパーティテスタの開発

岡村耕二　佐藤文明　勝山光太郎　水野忠則

三菱電機 (株) 情報電子研究所

概要

近年、OSI-TP (Transaction Processing : トランザクション処理) のようなマルチパーティと呼ばれる複数ノード間で相互に通信を行なう OSI プロトコルが標準化されてきた。しかしながら、従来の適合性試験手法は 1 対 1 の通信 (シングルパーティ) をベースにしているため、マルチパーティに適用するテスタを開発するためには、従来の適合性試験手法を拡張する必要があった。その拡張には、テスタの構成に関するものと、試験シーケンスに関するものがあった。我々は、テスタの構成への拡張に対してはオブジェクト指向を適用して、新たな機能を付加させ、試験シーケンスに対しては、シーケンスユニットを定義し、シーケンスユニットに基づいて効率のよいマルチパーティ試験を行なうことができるようにした。

Design and Implementation of Multi-Party Tester

Koji OKAMURA Fumiaki SATO Kotaro KATSUYAMA and Tadanori MIZUNO

Computer & Information Systems Laboratory, MITSUBISHI Electric Corporation
Ofuna 5-1-1, Kamakura, Kanagawa 247 Japan

Abstract

Recently multi-party protocol of OSI such as distributed transaction processing was standardized. An existing test method is not applicable for the multi-party testing, because the tester is designed for the single party protocol. So , to test the multi-party protocol, the tester must be extended for the multi-party testing. We extended the tester for single party to the tester for multi-party. The extensions were about components of tester and treatment of the test sequence. We added the new components by concepts of object-oriented and defined sequence unit to test multi-party effectively.

1. はじめに

ISO (International Organization for Standardization: 国際標準化機構) および CCITT (International Telegraph and Telephone Consultative Committee: 国際電信電話諮問委員会) では、異種システム間の相互接続を目的とし、OSI (Open Systems Interconnection : 開放型システム間相互接続) の各種プロトコルの標準化と、開発されたソフトウエアがプロトコル規約に正しく適合しているかを試験する適合性試験に関する標準化を行なっている。我々は、OSI 通信ソフトウエアの効率的開発を目的として、特に試験の効率化を目指し、適合性試験手法の標準にのっとったテスタの開発を行なってきた[1]。

近年、OSI-TP (Transaction Processing : トランザクション処理) のようなマルチパーティと呼ばれる複数ノード間で相互に通信を行なう OSI プロトコルが標準化されてきた。ところが、従来の適合性試験手法は 1 対 1 の通信 (シングルパーティ) をベースにしているため、マルチパーティに適用するテスタを開発するためには、従来の適合性試験手法を拡張する必要があった。本稿では、シングルパーティテスタを拡張したマルチパーティテスタの概要と、マルチパーティテスタ用の試験シーケンスについて述べる。

2. 適合性試験

2.1 適合性試験の概要

適合性試験 (Conformance Testing) とは、OSI 製品が OSI 規格に正しく準拠しているか否かを統一的に試験することである。近年、この適合性試験技術は、システム間の相互接続性を高める上で重要となってきている。ISO および CCITT では、OSI プロトコルに関する適合性試験の標準化を行なうために、適合性試験の方法と枠組を定め、それに基づき、各プロトコルごとに抽象試験仕様を標準化している[2]。

適合性試験手法にのっとって開発された試験システムをここでは単にテスタと呼ぶ。テスタは被試験対象である IUT (Implementation Under Test) の振舞いを抽象サービスプリミティブ (ASP : Abstract Service Primitive) により制御し観測する。このような試験手法では被試験対象の内部構造には触れないため、いわゆる“ブラックボックステスト”ということができる。

2.2 試験方法

現在、標準化されている試験方法では、OSI 基本参照モデルに基づき、抽象的試験方法が用いられている。抽象的試験では、被試験対象にどのような入力が制御できるのか、またその入力を被試験対象に与えた時、その入力に対して、被試験対象からどのような出力が観測できるか、という観点で試験を行なう。

抽象的試験方法の試験アーキテクチャは、ブラックボックス的試験アーキテクチャである。図-1 に試験アーキテクチャを示す。

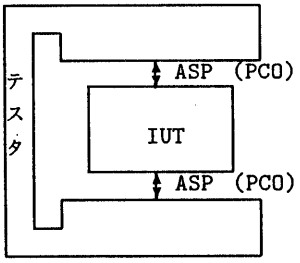

図-1 : ブラックボックス的試験アーキテクチャ

このブラックボックス的試験アーキテクチャにおいて、テスタから被試験対象への制御観測点 (PCO: Point of Control and Observation) は次の 2 つの待ち行列としてモデル化できる。

- 被試験対象に向けて送られる試験事象の制御のための 1 つの出力待ち行列。
- 被試験対象から受け取った試験事象の観測のための 1 つの入力待ち行列。

試験は、試験シーケンスに従って、試験データをこのキューに送信することと、それに対する応答をこのキューから受信し、期待される結果と比較することによって行なわれる。

2.3 テスタの構成要素

テスタは大きく 2 つの機能部分に分けることができる。1 つは被試験対象の上位インタフェースと対応するもので上位テスタ (UT : Upper Tester) と呼ばれ、被試験対象の振舞いを ASP により、制御し観測する。もう 1 つは被試験対象の下位インタフェースと対応するもので下位テスタ (LT:Lower Tester) と呼ばれ、被試験対象の振舞いを PDU により、制御し観測する。試験はこれらの下位テスタと上位テスタが協調動作を行なうことにより、実行される。試験は、下位テスタと上位テスタが協調して行なう必要があり、この協調手順を試験調和手順 (TCP: Test Coordination Procedure) と呼ぶ。

2.4 テスタの実現方法

被試験対象、下位テスタ、上位テスタからなる概念モデルをもとに、実際の適合性試験の方式を、被試験対象、下位テスタ、上位テスタとの位置関係や被試験対象の構成からいくつかの方式に分類することができる[3]。

我々は、図-2のような被試験対象、下位テスタ、上位テスタの位置関係でテスタを開発してきた。

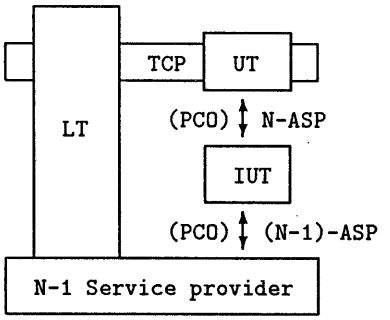

図-2：IUT、UTおよびLTの位置関係

図-2で示されるように、上位テスタは、被試験対象とはN-サービスプリミティブのインタフェースに制御観測点があり、下位テスタは、(N-1)-サービスプリミティブ(N-PDU)のインタフェースに制御観測点がある。

上位テスタと下位テスタは試験管理プロトコルにより協調動作を行なう。我々は、上位テスタと下位テスタの協調手順は、試験協調手順の簡素化を図るためにフェリー法を採用している。フェリー法は、上位テスタと下位テスタの間でフェリー制御チャネルを被試験対象とは、独立に設計し、上位テスタは被試験対象の上位テスタ側で送受信されるイベントをフェリー制御プロトコルで下位テスタに送受信する。

フェリー法は以下のような特徴を持つ[6]。

1. 被試験対象装置に搭載する上位テスタのモジュールが単純なため、容易に開発でき、また、可搬性が増す。
2. 上位テスタと下位テスタの同期がとりやすく、試験結果が容易に得られる。

フェリー法では基本的に下位テスタが能動的な動作を行ない、上位テスタは受動的な動作を行なう。イベントの判定は下位テスタ側で行なう。フェリー法で用いるフェリープロトコルは、上位テスタ及び下位テスタのプログラム作成量を削減するために、極力簡単なプロトコルにした。フェリープロトコルは、下位テスタが上位テスタに対して、活性化/非活性化命令および、データの送信/受信命令、を行なうプロトコルのみを定義している。

3. マルチパーティ試験とその実現方式

3.1 マルチパーティ試験法

マルチパーティ試験とは、従来の適合性試験手法が1対1のOSIプロトコルを実装した通信システムにのみ適用可能であるのに対して、1対nの通信を同時に行なう通信システムに対しても適用可能とするように従来の試験手法を拡張したものである。1対nの通信を同時に行なう形態としては、複数のアソシエーションを同時に使用する形態と、複数の利用者が存在する形態がある。これらの1対nの通信を同時に行なうOSIプロトコルを実装した被試験対象に、従来の上位テスタや下位テスタを用いる試験手法に基づいたテスタを適用させるためには、複数のアソシエーションを使用する被試験対象に対しては、複数の下位テスタを用意する必要があり、複数の利用者が存在する被試験対象に対しては、複数の上位テスタを用意する必要がある。

これらの複数のテスタの機能を、並列テスタと呼び、それらの並列テスタを管理・協調させる機能をマスタテスタと呼ぶ。上位テスタにおけるマスタテスタは、マスタ上位テスタ(MUT,Master Upper Tester)、並列テスタは、並列上位テスタ(PUT,Parallel Upper Tester)と呼び、下位テスタではそれぞれ、マスタ下位テスタ(MLT,Master Lower Tester)、並列下位テスタ(PLT,Parallel Lower Tester)と呼ぶ。図-3にマルチパーティテスタの例を示す。

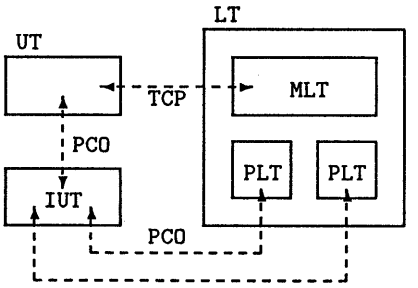

図-3：マルチパーティテスタの例

従来の適合性試験手法を拡張して、マルチパーティ試験を行なうためには、上位テスタ、下位テ

スタに、これらのマスタテスタ、並列テスタの機能を新たに付加する必要がある[4]。我々は、マルチパーティテスタにおけるマスタテスタ、並列テスタの拡張を、管理の面ではテスタの構成要素に着目して行ない、協調の面に関しては、試験シーケンスに着目して行なった。

3.2 構成要素の実現方針

マスタテスタが並列テスタを管理できるために、テスタの構成要素を拡張する必要がある。ただし、テスタの基本的な構成要素はシングルパーティテスタと同様に、上位テスタと下位テスタから構成される枠組のままである。

新たなマルチパーティテスタの構成要素として、並列テスタ、マスタテスタの機能を拡張する必要がある。並列テスタは、被試験対象の対となる機能を持ち、マスタテスタは並列テスタの管理、協調を行なう機能を持たなければならない。マスタテスタが並列テスタに対し行なう管理、協調とは、試験シーケンスに従って、それぞれの並列テスタの試験データや試験結果の送受信の順序を決定し、実際の送受信を行なわせることである。

複数の並列テスタに対し、被試験対象は1つしか存在しないので、一度に、被試験対象に試験データを送信または受信できる並列テスタの数は必ず1つだけである。ゆえに、並列テスタの機能のためにテスタ自体を物理的に並列(マルチプロセス)にする必要はない。マスタテスタが並列テスタの振舞いを管理することにより、並列テスタを論理的に並列に動作させればよい。

このため、並列テスタは、並列テスタが被試験対象の対として機能するために必要な、ネットワークの接続に必要な設定値と、送受信用のデータ領域があれば、いいことになる。本マルチパーティテスタでは、1つ1つの並列テスタはそれぞれオブジェクト指向でいうインスタンスに対応させた。先のネットワークの接続に必要な設定値や、送受信用のデータ領域は、そのインスタンス変数に対応させた。

マスタテスタもまた、1つのインスタンスで表現し、マスタテスタが並列テスタを管理、協調させるために行なうデータのやりとりは、オブジェクト指向でいうメッセージパッシングを適用することができる。

なお、マルチパーティテスタの開発には、我々が通信ソフトを効率的に開発するの目的で開発してきたオブジェクト指向言語 superC [7][8] を用いる。

3.3 試験シーケンスの拡張方針

マスタテスタが並列テスタの協調を行なうためには、テスタは試験シーケンスに従って動作するので、試験シーケンスに関する拡張も行なう必要がある。

従来のシングルパーティ用の試験シーケンスは、キューが1つであったが、マルチパーティでは、複数のキューが存在し、それぞれのキューは異なった試験シーケンスで、異なった試験データを流す必要がある。ただし、それらのキューは被試験対象においてシリアライズされるため、試験データの与え方には注意が必要である。

試験シーケンスの拡張方針は以下の2点である。

テスタと試験データの組合せ

> どのテスタからどの試験データを被試験対象に送信するかが表記できなければならない。また、複数の並列テスタがそれぞれ異なる試験データを送信し得る場合、例えば、n個並列テスタがあれば、組合せの数は n! 通りの場合がある。

試験データ送信の順列

> テスタと試験データの組がどの順序で発行されるかが表記されなければならない。このテスタと試験データの組の順序は、例えば、並列テスタが、n個あった場合、組合せの数は n! 通りの場合がある。

このように、マルチパーティ試験では、複数の並列テスタの組み合わせに注意を払う必要がある。我々は、ここで、1つの試験シーケンスにより、複数の並列テスタの組合せを記述する方法を提案し、採用することにした。これにより、本マルチパーティテスタでは、1つの試験シーケンス中の各試験データと各並列テスタの組合せから考えられる複数の試験シーケンスを生成し、試験を行なうことにした。

4. マルチパーティテスタの設計

4.1 モジュール構成

本マルチパーティテスタは、上位テスタが1つで、並列下位テスタが2つの構成で OSI-TP の試験を行なう予定であるが、テスタの設計は OSI-TP へのアクセス方法や、OSI-TP が利用するサービスのアクセス方法および、テスタ間の同期機構に拘束されない汎用的なテストシステムを目指し、OSI-TP だけではなく他の分散システムにも適用できるように設計した[5]。

そのため、上位テスタと下位テスタともに、機能をモジュール化することにより、試験対象のプロトコルや、上位、下位テスタ間のプロトコルに依存する部分と、依存しない部分を独立に開発し、多くの通信システムの試験に対して汎用的に使用できるようにした。

モジュール分割としては、まず、テスタを、試験を行なうモジュールであるテスタモジュールと、通信を行なうモジュールである CM (Communication Manager) モジュールに分けた。テスタモジュールは、テストシーケンスに基づく、テスタの振舞いの決定を行ない、CM モジュールは、テスタと被試験対象間や、下位テスタと上位テスタ間の通信を行なう。テスタモジュールと、CM モジュールは、それぞれ、試験対象とやりとりを行なうモジュールと、相手テスタとやりとりを行なうモジュールに分割されている。マスタテスタの機能は、テスタモジュールに含まれ、並列テスタの機能は、CM モジュールに含まれている。

4.2 マルチパーティテスタの試験シーケンス

シングルパーティの試験では、被試験対象に試験データを出力するテスタはただ 1 つのみであったため、試験データを与えるテスタの組合せや、順列を考慮する必要がなかった。しかし、マルチパーティの試験では、試験データを出力するテスタが複数存在するために、テスタの組合せや、順列を考慮する必要性が生じてきた。そのため、マルチパーティ試験においては、シングルパーティ試験用の試験シーケンスに対して拡張する必要がある。

試験シーケンスは、全体のシーケンスを、入力待ち (静止状態) の被試験対象にテスタからの出力した後から、被試験対象から入力を得て、再び、被試験対象が入力待ちになるまでの期間の連続と見ることができる。この 1 つの期間をシーケンスユニットと呼ぶことにする。そのユニットは、3 つのフェーズからなる。

1. テスタが、被試験対象に試験データを出力する。
2. 被試験対象がその試験データに基づいて動作する。
3. テスタが、被試験対象から試験データを入力する。

ただし、シーケンスユニットは、テスタから被試験対象への出力の全ての組合せに関して、被試験対象 からの入力が必ず同じものでなければならない。例えば、テスタの選択において、同様な試験データを 1 つのテスタから 被試験対象 へ出力する時、上位テスタを選択した場合と下位テスタを選択した場合では、被試験対象からテスタへの入力は異なる場合がある。このようなシーケンスユニットは、別なシーケンスユニットとして扱わなければならない。

マルチパーティテスタがマルチパーティテスタの試験シーケンスを扱うためには、結局このシーケンスユニットを扱うことができればよいことになる。そのためには、上であげた 3 つのフェーズのうちの 1. と 3. に注意すればよい。マルチパーティテスタがシーケンスユニットを扱うために必要な機能を挙げる。

被試験対象への出力

シーケンスユニットでの、被試験対象への出力において問題となるのは、複数の試験データと複数の並列テスタとの組合せ方と、その並列テスタの出力の順番である。全ての、試験シーケンスを記述するよりは、シーケンスユニットにおいて、取りうる組合せをテスタが自動的に生成すれば、効率の良い試験を行なうことができる。

マルチパーティテスタが、シーケンスユニットにおいて、被試験対象への出力のフェーズで行なう試験シーケンスに関する処理は以下のものがある。

1. 試験データと並列テスタの取りうる組合せを生成する。
2. 試験データを出力する並列テスタの考えられる順列を生成する。

被試験対象 からの入力

被試験対象 からの入力の順番は、テスタが決定することはできないので、バッファリングの機能が必要である。これは、上位テスタ、下位テスタが同時に被試験対象 からデータを入力する時は、上位テスタ、下位テスタの順番すらわからないので、上位テスタ、下位テスタそれぞれで、バッファリングの機能が必要である。

このようにして、テスタがシーケンスユニット内で考えられる組合せを生成し、全てのシーケンスユニットを組合せて生成される試験シーケンスに従った試験を行なえば、全ての試験シーケンスを記述する必要がなく、少ない試験シーケンスで効率の良い試験を行なうことができる。

4.3 シーケンスユニットの表記

本マルチパーティテスタでは、シーケンスユニットの表記を以下のように行なう。下位テスタはL、上位テスタはUで表し、テスタから被試験対象への出力は!、被試験対象からの入力は?で表す。試験データと並列テスタとの組合せはC[m(i)]で表わす。mは組合せに関する識別番号であり、mの数字が同じ出力は1つのグループで、その出力の間で並列テスタの排他的な選択を行なわせることを示している。ただし、並列テスタから同じ種類のデータを送信するような場合は、並列テスタの選択を行なう必要はなく、この時は、m=0で表し、iに割り当てるべき並列テスタの識別番号を入れる。mが0でない時は、iは空にする。

被試験対象への出力の順列はP[j]で表す。これは、そのシーケンスユニットで、データを出力する順序を替えてよい出力がj個あることを表している。そのため、順序を替えてよい出力は必ずj行表記する必要がある。第1行目のみP[j]のように表記し、2行目以降は、P[.]で表記し、同じグループであることを示す。

並列テスタがn個で、そのシーケンスユニット内に並び替え可能なグループがp個あった場合、その1つのシーケンスユニットで、$((n!)^{MAXofm_1} \times j_1!) \times ((n!)^{MAXofm_2} \times j_2!) \times ... \times ((n!)^{MAXofm_p} \times j_p!)$通りの試験シーケンスを表すことができる。

被試験対象からテスタが試験結果を入力する時は、入力元の並列テスタの識別子番号を添えて表示する。

このように、並列テスタから被試験対象への出力の組合せによる複数の試験シーケンスの生成は、マルチパーティテスタ内で行なうので、全ての組合せの試験シーケンスを生成する必要がなくなる。

以下にシーケンスユニットの表記例を示す。この例では、並列上位テスタの数は1つで、並列下位テスタの数は2個とする。

例 1

2つの並列下位テスタから同種類のrequest-データを送信し、上位テスタでindication-データを受信する場合は、

```
L  C[0(1)]P[2] ! req-データ
L  C[0(2)]P[.] ! req-データ
U1             ? ind-データ
```

と、表記する。この1つのシーケンスユニットは2通りのシーケンスを表している。このシーケンスユニットをマルチパーティテスタが解釈して、2通りの試験シーケンスを生成する手順を述べる。

ステップ 1

1行目を読み込み、C[0(1)]の表記により、選択を行なう必要がないことを解釈する。()内の数字が1なのでより、この出力は必ず、並列下位テスタL_1に割り当てられる。

P[2]により、並び替え可能な出力が2つあることを解釈し、もう1行読み込む。

ステップ 1-1-1

1行目の出力は、L_1に割り当てられた状態で処理は進む。

2行目の、C[0(2)]の解釈により、この出力は必ず、並列下位テスタL_2に割り当てる。

ステップ 1-1-2

1行目で割り当てられた、L_1と2行目で割り当てられたL_2の可能な順列は2通りあり、それらのシーケンスを生成する。

ステップ 2

結局、このシーケンスユニットからは以下に示す2通りの試験シーケンスが生成される。

```
L1 ! req-データ        L2 ! req-データ
L2 ! req-データ        L1 ! req-データ
U1 ? ind-データ        U1 ? ind-データ
```

例 2

2つの並列テスタ中、1つの並列テスタがrequest-データAを送信し、他方がrequest-データBを送信し、上位テスタでindication-データを受信する場合は、

```
L  C[1()]P[2] ! req-データA
L  C[1()]P[.] ! req-データB
U1            ? ind-データ
```

と、表記する。この1つのシーケンスユニットは4通りのシーケンスを表している。このシーケンスユニットをマルチパーティテスタが解釈して、4通りの試験シーケンスを生成する手順を述べる。

ステップ 1

1 行目を読み込み、C[1()] の表記により、並列テスタの選択を行なう必要があることを解釈する。今、並列テスタは 2 つあるのでこの出力は、L_1 に割り当てられる場合と、L_2 に割り当てられる場合がある。

次に、P[2] により、並び替え可能な出力が 2 つあることを解釈し、もう 1 行読み込む。

ステップ 1-1-1

1 行目のデータを、L_1 に割り当てられた状態で処理は進む。

2 行目の、C[1()] の表記により、この選択は 1 行目の選択と同じグループ内の排他的な選択と解釈され、すでに L_1 には出力が割り当てられているのでこのデータは必ず並列テスタ L_2 に割り当てられる。

ステップ 1-1-2

1 行目で割り当てられた、L_1 と 2 行目で割り当てられた L_2 の可能な順列は 2 通りあり、それらのシーケンスを生成する。

ステップ 1-2-1

次に、1 行目のデータを、L_2 に割り当てられた状態で処理は進む。

2 行目の、C[1()] の表記により、この選択は 1 行目の選択と同じグループ内の排他的な選択と解釈され、すでに L_2 には出力が割り当てられているのでこのデータは必ず並列テスタ L_1 に割り当てられる。

ステップ 1-2-2

1 行目で割り当てられた、L_2 と 2 行目で割り当てられた L_1 の可能な順列は 2 通りあり、それらのシーケンスを生成する。

ステップ 2

結局、以下に示す 4 通りの試験シーケンスを生成する。

```
L1 ! req-データA      L2 ! req-データB
L2 ! req-データB      L1 ! req-データA
U1 ? ind-データ       U1 ? ind-データ

L2 ! req-データA      L1 ! req-データB
L1 ! req-データB      L2 ! req-データA
U1 ? ind-データ       U1 ? ind-データ
```

なお、本稿の表記法では、2 番目の並列テスタの選択を排他的に行なわないシーケンスを自動生成するシーケンスユニットの表記はできないため、その場合は、別々のシーケンスユニットとして表記する必要がある。しかし、本マルチパーティテスタの並列テスタの数は 2 つであり、このような場合を別の試験シーケンスとして表記しても、試験シーケンスの数の増加は少なく、問題はない。

5. マルチパーティテスタの適用例

OSI-TP に本マルチパーティテスタを適用した時の、いくつかのシーケンスユニットの例を示す。

例 1

```
U  C[0(1)]p[2] !  BEGIN_DIALOGUE-req
U  C[0(1)]P[.] !  BEGIN_DIALOGUE-req
L1             ?  BEGIN_DIALOGUE-ind
L2             ?  BEGIN_DIALOGUE-ind
```

この例は、上位テスタから各並列テスタを指定して被試験対象にデータを送信するパターンで、先に出す並列下位テスタの選び方によって、2 通りの試験シーケンスが生成される。

例 2

```
L  C[0(1)]P[2] !  END_DIALOGUE-req
L  C[0(2)]P[.] !  END_DIALOGUE-req
U1             ?  END_DIALOGUE-ind
U1             ?  END_DIALOGUE-ind
```

この例は、各並列テスタから被試験対象にデータを送信するパターンで、どちらが先に出すかによって 2 通りの試験シーケンスの試験が生成される。

例 3

```
L  C[0(1)]P[3] !  DONE-req
L  C[0(2)]P[.] !  DONE-req
U  C[0(1)]P[.] !  DONE-req
```

```
L1              ?  ROLLBACK_COMPLETE-ind
L2              ?  ROLLBACK_COMPLETE-ind
U1              ?  ROLLBACK_COMPLETE-ind
```

この例は、各並列下位テスタと、上位テスタから被試験対象にデータを送信するパターンで、どの順番でデータを出すかによって6通りの試験シーケンスが生成される。

例 4

```
L  C[1()]P[2] !  U_ABORT-req
L  C[1()]P[.] !  DONE-req
U1            ?  TP_U_ABORT-ind
L1            ?  ROLLBACK_COMPLETE-ind
L2            ?  ROLLBACK-ind
```

この例は、どの並列テスタが、U_ABORTを出すかという並列下位テスタの選択、および、どの順番に並列テスタがデータを送信するかという組合せにより4通りの試験シーケンスが生成される。

6. まとめと今後の課題

本稿では、シングルパーティテスタを拡張したマルチパーティテスタの概要を示した。マスタテスタと並列テスタの実現にはオブジェクト指向の概念を適用し、テスタを複雑にすることなしにそれらの機能を付加した。また、試験シーケンスにおいては、試験シーケンスをユニットに分け、それらのユニットをシーケンスユニットと定義し、シーケンスユニット毎に、マルチパーティに対応したテスタの組合せからなる複数の試験シーケンスを生成するようにした。そのため、マルチパーティテスタがこの試験シーケンスから試験の組合せを生成することにより、全ての試験シーケンスを記述する必要がなくなった。

本稿によるシーケンスユニットを用いた試験シーケンス生成法では、本マルチパーティテスタがターゲットとしている、ルートノードが1つで、リーフノードが2つである小規模なマルチパーティ試験では問題はない。しかし、今後はマルチパーティのノードが増えた時に試験シーケンスの組合せの数が爆発的に増加する大規模なマルチパーティの効率の良い試験を行なうことができる試験シーケンス生成のアルゴリズムや、試験の方法を開発する必要がある。

参考文献

[1] 勝山、佐藤、中川路、水野:“国際標準形式技法に基づく体系的試験支援環境 FOREST の提案と実現”, 情報処理学会論文誌 Vol.31 No.7 pp. 1123 - 1133 (1990).

[2] ISO DIS 9646, Information Processing Systems-Open Systems Interconnection - OSI Conformance Testing Methodology and Framework.

[3] 田中、清水、清水:“OSI の実現とその課題 (V) 適合性試験”, 情報処理学会解説 Vol. 31 No. 1 pp. 89 - 97 (1990).

[4] ISO/IEC JTC1/SC21 N5076, “Working draft for multi-party test methods”, Aug. (1990).

[5] 岡村、佐藤、勝山、水野:“マルチパーティテスタの設計”, 情報処理学会第 42 回 (平成 3 年前期) 全国大会 (1991).

[6] v.Bochmann,G and He, C.S.:“Ferry Approaches to Protocol Testing and Service Interfaces”,*Proceedings of the Second International Symposium on Interoperable Information Systems* pp. 303-309 (1988).

[7] 勝山、佐藤、中川路、水野:“通信ソフトウェア向けオブジェクト指向言語 superC”, 情報処理学会論文誌 Vol.30 No.2 pp. 234-241 (1989).

[8] 岡村、佐藤、中川路、勝山、水野:“オブジェクト指向言語 superC の評価”, 電子情報通信学会春季全国大会 (1991).