

今井 功, 寺島 美昭, 木野 茂徳, 下間 良樹
三菱電機(株) 情報技術総合研究所

1. はじめに

分散オブジェクト技術CORBA(Common Object Request Broker Architecture)は, 移動体通信など次世代ネットワークを実現するソフトウェア基盤記述として注目されている. 一方, ソフトウェアの迅速な開発を実現するため, 効率的なソフトウェア試験を実施する必要性が高まっている. 我々は, CORBAサーバを実装した通信機器の機能試験を効果的に行うための試験手法の研究に取り組んでいる[1]. 本稿では, 効率的なソフトウェア試験を実現する手段として, メッセージトレース機能の実現方式について報告する.

2. メッセージトレース機能の設計

2.1. 目的

メッセージトレースは, 分散オブジェクト指向システムの効率的なデバッグ手法として注目されている[2]. 我々は, 通信機器を対象に, CORBAインタフェースを介した機能試験におけるメッセージトレース機能を検討した. 本メッセージトレース機能は, クライアント(試験スイート)とサーバ(CORBAオブジェクト)間のメッセージ通信の情報収集及び解析を通じて, CORBAオブジェクトの機能確認及び通信機器の動作確認を行う事を目的とする.

```
/* 試験スイート記述例 */
function main()
{
    AmpFun_init( ); /* 初期化関数 */

    /* 変数宣言 */
    var index = 1;
    var status = IntParam();

    /* メソッド呼び出し */
    var ret1= AmpFun_getStatus( index, status );
    if ( ret == 1 ) {
        var level = 5;
        var ret2 = AmpFun_setPowerlevel( level );
    }
}
```

図1 試験スイート例

図1に, CORBA試験手法で定義した試験スイートの記述例を示す. メッセージトレース機能の実現には, 上記の試験スイートの入力に対し, 通信機器側で発生す

る動作を十分に評価するための必要な情報収集方式が課題となる. 以下に, 本課題を解決するため, 適切な観測ポイントと収集するメッセージトレース情報について定義する.

2.2. 情報収集のための観測ポイント

初めに, 観測ポイントについて述べる. 本試験手法では, 図2に示す通り, クライアントアプリケーションのコード下に, メッセージ通信の観測ポイントを置く. これにより, クライアントやサーバのアプリケーションコードに直接手を加えることなく, 機能確認を行うことが可能となる

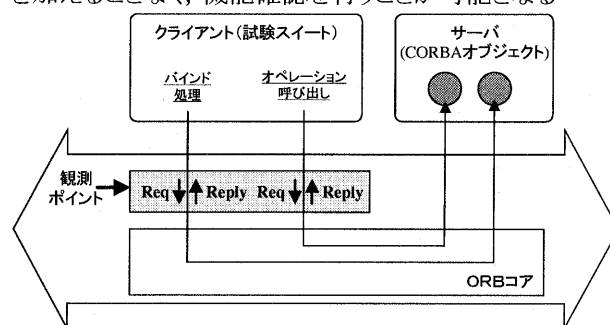


図2 メッセージトレースの観測ポイント

本観測ポイントは, ORB間の通信プロトコルを規定しているIIOP(Internet Inter-ORB Protocol)と呼ばれるポイントであり, 全ての通信処理動作を把握することが出来る重要な情報を収集することが出来る. 本観測ポイントから, 以下の情報を収集する.

- (1) バインド処理の要求情報及び応答情報
- (2) CORBA IDL(Interface Definition Language)で定義されるメソッド呼び出しの要求情報及び応答情報

2.3. メッセージトレース情報

続いて, 上記観測ポイントにより収集できるトレース情報について述べる. CORBAでは, アプリケーション間でのメッセージ通信をトレースする手段として, インターセプタと呼ばれるサービス[3]が規定されている. 本メッセージトレース機能では, このインターセプタの概念を利用する事により, 表1に示す情報を収集する. 1~3及び8は, 要求情報(Req)を表し, 5~7は, 応答情報(Reply)を表す. 尚, 4は, 要求情報及び応答情報の両方を表す.

3. メッセージトレース機能の実装

3.1. システム構成

通信機器のネットワーク管理機能評価を目的とした試験実行環境としてCORBAテストの実装実験を行い, 前章で述べた設計内容を元に, メッセージトレース機能の実装を行った. 図3に, システム構成を示す. ORBに

表1 メッセージトレース収集情報一覧

	項目	内 容
1	リポジトリ ID	インタフェース定義情報でオブジェクトを一意に識別する識別子
2	メソッド名	試験スイートのメソッド名
3	リクエスト ID	ORB 内部で採番される要求番号
4	動作(処理)状態	ORB 内部の処理状態
5	戻り値	メソッドの戻り値
6	処理結果	ORB 内部の処理結果
7	例外理由	例外が発生した場合の障害理由
8	通信種別	双方向通信や一方方向通信等、通信形態を示す

は、Inprise社のVisibroker for Java Version3.4(以下、Visibrokerと称する)を利用した。また、メッセージ通信の情報収集手段として、Visibrokerが提供するインタセプタ機能を利用した。更に、Visibroker上にメッセージトレース機能を実装した。試験スイートを実行すると、インタセプタ上で通信情報を取得し、これをメッセージトレース機能に送信する。メッセージトレース機能では、この通信情報を解析し、その結果を出力することによりCORBAインタフェースの機能確認を行う。

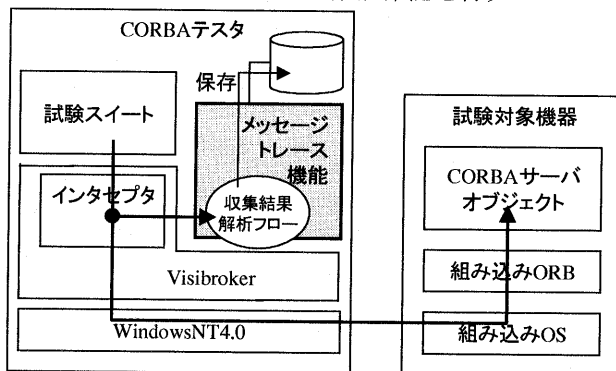


図3 システム構成

3.2. 出力フォーマットと動作説明

表1に示すトレース情報を元に、メッセージトレース機能が表示するトレース出力フォーマットを表2に示す。図3のCORBAテストにおいて、図1の試験スイートを実行すると、その結果として表2に示すトレース出力結果を得ることが出来た。これは、getStatus()及びsetPowerlevel()とメソッド呼び出しが順次行われている事を示している。また項目4の出力内容を見ると、例えばgetStatus()では、send_request_succeededにより試験対象である通信機器への通知が成功した事や、receive_replyにより応答情報を正しく受信できた事等が分かる。以上の通り、本出力フォーマットにより、試験スイートで実行されたメソッド呼び出しに関するORB内部での動作手順を把握する事が出来、通信機器の動的な動作確認が確認できた。

4. 考 察

実際のメッセージトレースの開発に試験スイートを適

表2 メッセージトレース出力例

1.	2.	3.	4.	5.	6.	7	8.
IDL:Amp Fun:1.0			bind				
IDL:Amp Fun:1.0			bind_succeeded				
	getStatus	1	prepare_request				req/ reply
	getStatus	1	send_request				
	getStatus	1	send_request_ succeeded				
	getStatus	1	receive_reply	0	NO_EXC EPTION		
	setPower level	2	prepare_request				req/ reply
	setPower level	2	send_request				
	setPower level	2	send_request_ succeeded				
	setPower level	2	receive_reply	0	NO_EXC EPTION		

用した結果、以下の評価を得た。初めに、提案した観測ポイントとトレースにより、試験スイートで期待した動きを把握する事が出来た。但し、この出力結果を試験スイート実行中の実際の動作状況と関係づけるのは難しい。即ち、試験対象である通信機器のデバッグ作業では、試験スイートの実行結果をリアルタイムに把握することにより、試験の中断や試験実行者による次の試験動作の選択が必要となる場合が多い。本メッセージトレース機能は、CORBAインタフェースを中心とした処理状態を把握することは可能であるが、通信機器依存データ情報の出力機能を持たないため、リアルタイムな情報取得というニーズには答えられない。この解決には、メッセージトレース機能とは別に、通信機器からの応答情報を解析しリアルタイム表示する機能が必要である。

また、本メッセージトレース機能を複数の通信機器を対象とした分散システムの試験に適用した場合、大量のトレース情報が出力される事が予想される。しかし、現状の機能では、効果的に評価する方法が欠如しており、この解決方法を検討する必要がある。

5. おわりに

本稿では、通信機器に対するCORBA機能試験において、メッセージトレース機能に関する設計方針と、これに基づいた実装結果について報告した。今後の課題としては、分散システムを対象とした試験結果の効率的なメッセージトレース方式について検討する予定である。

参考文献

- [1] 寺島他:「通信機器におけるCORBAインタフェース試験の検討」, DICOMO2000シンポジウム論文集, 情報処理学会, pp.49-54(2000).
- [2] 久保田:「分散オブジェクト指向におけるメッセージのトレース」, 情報処理学会論文誌, Vol.39, No.1, pp.91-101(1998).
- [3] OMG:「The Common Object Request Broker: Architecture and Specification, Technical Report, Revision2.3.」, June 1999.