# メッセージ制御による分散システムのテスト・デバッグ支援

上野知樹† 太田剛‡

静岡大学大学院情報学研究科† 静岡大学情報学部‡

## 1. 研究の背景

コンピュータネットワークの普及により，クライアント・サーバに代表される分散システムが広く利用されている．分散システムを実現するためのプログラムを分散プログラムという．分散プログラムでは逐次プログラムとは異なり，入力される値だけでなく，クライアントやサーバのメッセージ送受信タイミングによっても動作が異なるという特徴がある．このような分散プログラムに対するテスト・デバッグの難しさは，ある操作に対して一度テストを行い不具合が発生しなかったからといって，その操作の正当性が保証されないという点にある．

この分散プログラムの開発を容易にするため，実行履歴を取得しこれを次回の実行時に忠実に再現するシステムの開発や，デバッグ時に分散プログラムの位置透過性を提供するデバッガの開発[1] といった研究が行われている．しかし，これらは不具合の再現は出来るが，送受信タイミングを指定し不具合を発見するといったような，ユーザが望む状況にプログラムを導くのが難しいという問題がある．

本研究では，分散システムのための業界標準である CORBA 環境での，特定のプログラミング言語にとらわれないテスト・デバッグ支援システムの開発を目的とする．このシステムの特徴は，対象プログラムの送受信メッセージを捕らえユーザ側で送受信タイミングを自在にコントロールできる点である．この機能により，再現が難しかった不具合も容易に再現することが可能である．

## 2. 提案手法

CORBA アプリケーションのメッセージ送受信を制御するには，まず CORBA 呼出しの実行履歴を取得し，その履歴情報をユーザに提示してタイミングの違うシーケンスを指定してもらう．次に指定されたシーケンスになるよう制御しながら実行するといった流れである．本研究では，実行履歴の取得はインターセプタによるトレース[2]技法に注目した．まず CORBA のアーキテクチャとインターセプタに基づく実行履歴の取得について簡単に述べ，次にシステムの概要を述べる．

### 2.1 CORBA

CORBA は環境や言語に依存しないオープンな分散システムの仕様であり，複数の環境や言語の混在した分散アプリケーションを構築することができる．主要な構成要素の１つに ORB(Object Request Broker) がある，ORB はクライアントオブジェクトとサーバオブジェクトとの間のメッセージ通信を仲介するオブジェクトバスであり，オブジェクトの位置透過性を提供する．

### 2.2 インターセプタによる実行履歴の取得

ORB は通信のフック(遮断点) を提供する．インターセプタはこのフックを使って ORB の通常の実行の流れを遮断することができる．

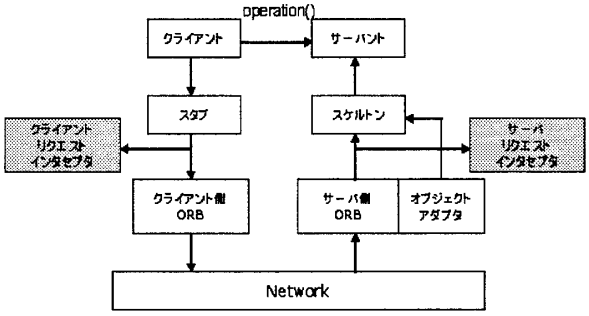


図１: CORBA におけるインターセプタの配置

インターセプタは，CORBA 仕様で定義されており，CORBA の実装系に依存しない．このインターセプタを使って，CORBA 呼出しの履歴を取得する．

図 1 に示すように，本研究ではクライアントインターセプタとサーバインターセプタの２種類を扱う．これらのインターセプタは，CORBA

A testing and debugging support for distributed systems by controlling messages

† Tomoki Ueno ・ Graduate School of Informatics, Shizuoka University

‡ Tsuyoshi Ohta・Faculty of Informatics,Shizuoka University

呼び出しの特定のポイントで ORB によって呼び出される４つのインターセプションポイントを定める．

・send_request, receive_reply

・receive_request, send_reply

これら４つのインターセプションポイントを使うことで，最も重要なイベントを取得することを可能にする．

### 2.3 システムの概要

ユーザはまず ORBInitializer を用いて各コンポーネントにインターセプタを登録する．この際アプリケーションの再コンパイルは必要ない．本システムの構成を図２に示す．

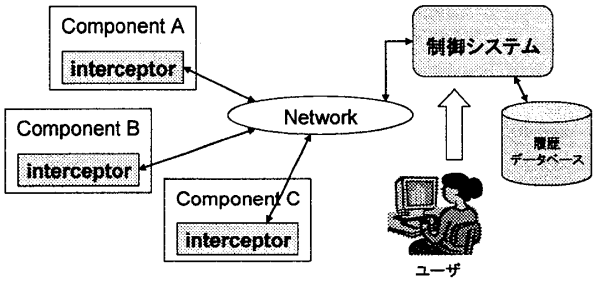


図２： テスト・デバッグ支援システム

ユーザは分散プログラムに対して，次の３つのモードを切り替えて実行する．

**履歴取得モード**

実行の履歴を取るモードである．このモードでプログラムを実行すると，メッセージ送受信のたびにインターセプタ内で履歴をとる．また，メッセージ制御モード時にメッセージを区別するため各メッセージに ID を付ける．

**再現実行モード**

取得した履歴の再現実行を行うモードである．送信プロセスがメッセージ送信メソッドを呼び出すと，インターセプタがこれを捕まえて，制御システムへ履歴通りか確認する．受信側の状況が履歴通りではない場合は，ある時間待ってまた確認を行う．履歴通りであるなら受信側へ送信メッセージを送る．

このように，このモードではプログラムは実行履歴の通りに何度でも同じ挙動をする．デバッガを用いるときには，このモードを使うことによりプログラムの実行を追う事ができる．

**メッセージ制御モード**

ユーザが指定した通りにメッセージを制御するモードである．まずユーザがあるシーケンスを指定するための条件式を書く．書くことができる条件はメッセージ(mx) 受信の順序関係を表す'$m1 < m2$'（$m2$ より $m1$ の方が先に受信）や'(条件 1)&&(条件 2)'である．その条件式を制御システムで解析しユーザが指定した条件を保存する．

送信プロセスがメッセージ送信メソッドを呼び出すと，インターセプタがこれを捕捉して，ユーザが指定したメッセージか確認する．指定されている場合はその通りになるよう制御する．指定されていないメッセージの場合，再現モードと同じように履歴通りに制御する．ただし，ユーザが指定した制御が一度でも行われた後は，履歴通り実行することに意味がなくなるため何もしない．

このモードではユーザの指定通りにプログラムを挙動させることができる．テストを実施する時には，このモードを使うことで容易に送受信タイミングを含めたテストが可能になる．

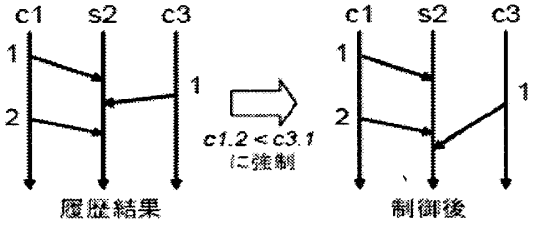


図３： メッセージ制御の例

## 3. 実装と評価

本システム自体も CORBA アプリケーションとして実装し，動作確認を行った．また，有名な同期問題である「生産者・消費者問題」，「食事する哲学者の問題」について本システムを適用し，送受信タイミングを制御することによるテスト・デバッグ支援の有用性を確認した．

## 4. まとめと今後の課題

メッセージ制御機能により，ユーザが望む状況にプログラムの実行を導くことができ，分散システムのテスト・デバッグ作業が軽減されると考えられる．

現在はメッセージ制御を行う際，基本的な条件式のみであるので，今後はテスト・デバッグを行う者がどう制御したいのかをさらによく検討し，条件式を追加実装していく必要がある．

**参考文献**

[1] 黒石光雄，中里秀則:分散透過デバッガの開発，電子情報通信学会 2001 年総合大会講演論文集 D-3-5, p. 33, 2001.

[2]Z. A. Mann, K. Kondorosi: Tracing system-level communication in distributed systems, Softw. Pract. Exper, pp. 727-755, 2004.