通信ソフトウェア自動試験に関する一考察
－試験自動化へのアプローチ－

2F－7

黒田憲一　白石智　立元慎也
NTT交換システム研究所

1. はじめに

通信ソフトウェアの生産性向上を狙いとして、プロセスに着目した開発方法論、及び、この方法論に基づくソフトウェアプロセスを効果的にサポートする開発環境の確立を目指している。[1][2] 本稿では、ソフトウェア開発工程のうち、定型作業が多く、自動化の効果が期待できる試験工程を取り上げ、試験自動化に至るアプローチ法について述べる。

2. 試験の概観

通信ソフトウェアの試験は、一般のソフトウェアの試験と同様に、モジュール単体・結合・複数複合・安定化の工程から構成されるが、要求仕様が膨大で複雑なことから、全工程に対して試験工程の比率が高くなっている。各工程では、設計と逆方向に、処理の断片・論理上一つの意味を持つ単位／ハードウェアを含めた動作・競合動作・異常環境下を含む総合動作の順に正常性を確認する。

3. 試験自動化の実現性と期待される効果

試験の各工程における作業（試験プロセス）は、試験項目の抽出・手順書作成・試験実行・結果解析等のサブプロセスに分解できる。サブプロセスは、更に詳細なプロセス、例えば試験実行の場合、条件設定・実行・結果収集と分解できる。このような手順を繰り返し、プロセスの構造的な詳細化を図る。この過程で、問題解決部分を人間に、形式変換部分をツールに分担させることで、自動試験へ向けてのプロセスが導出できる。形式変換部の最終的な出力は、試験項目毎に、試験者の活動を記述したもの（試験シナリオ）となる（図1）。
シナリオによる自動試験では、試験実行の作業効率が向上するだけでなく、以下のような付帯的効果も期待できる。

(1) 再試験の効率化：デグレを防ぐための試験、流用部分に対する確認試験に、シナリオが再利用でき、効率を上げられる可能性がある。

(2) 作業レベルの確保：標準化により最低レベルは確保でき、また、問題発生時には、残した記録が再試験指示等の有力な判断材料となる。

(3) 教育・ノウハウの継承：徒弟制度的、口承的であった試験活動のノウハウが、明示的書物として引き継がれる。

(4) プロセスの改良：記録を計算機で操作できる形で残すため、従来、再現ができない、どこが悪かったのか調べられない、あるいは別途実験するには金と時間がかかりすぎる等で困難であった、客観的な評価・改善活動を行える可能性がある。

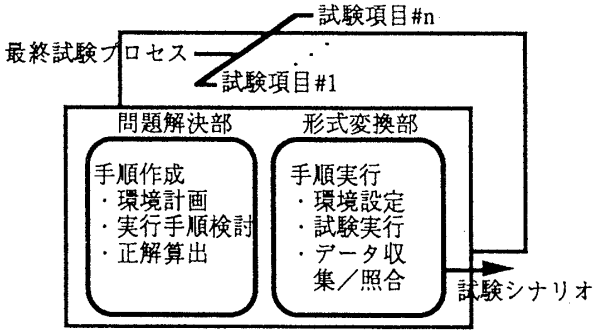

図1　最終試験プロセスと試験シナリオ

4. 自動試験システム構成法

4.1 システム構築上の要求条件

以下に示す進化する仕組みを内蔵させておくことが重要である。

(1) 独自性の囲込み：通信ソフトウェアには、超多重・リアルタイム性、機能追加の容易性、高信頼性が要求されるため、システム毎にCPU、OS、言語、ソフトウェア構造に工夫がこらされている。自動試験システムは、これらの選択に対して柔軟に対処できるよう構成されている必要がある。

A Study of Automatic Testing System for Communication Switching Software
-- An Approach to Automatic Test --
Kenichi Kuroda, Satosi Shiraishi, Shinya Tachimoto
NTT Communication Switching Laboratories
9-11, Midori-Cho 3-Chome, Musashino-Shi,
Tokyo, 180 Japan

(2) 継承性の確保: 通信システムは、ハードウェアを含めた繊細・緻密な機能分担の上に成立っており、最終的には、ハードウェアを含めた試験が必須である。この試験用のツール類が発達しており、これらを有効に活用できることが要求される。

(3) 汎用性の取込み : WSと分散環境、Unix、CASEツール類の発展は目ざましいものがあり、今後もこの傾向が続くと思われる。これらを基盤とし、試験の各局面で独立に発展してきた・発展していくツールを有機的に結合する機構が必要となる。

4.2 システム構成

自動試験システムは、Unix + ネットワーク環境の上に、" 試験の進行を制御する部分 " と " 試験内容（個々のツールが行う動作）を制御する部分 " の２階層から構成する（図２）。前者をIntegration Platformと呼び、試験者が記述した試験シナリオを解釈し、逐次実行する。この時、バックエンドにて配下の様々なツールの環境の違いを吸収し、連動するツール相互間のタイミングを吸収する等の協調動作を可能としている。また、各ツールの実行状態に基づき、次動作への移行やツール初期設定等のエラー処理を行う。

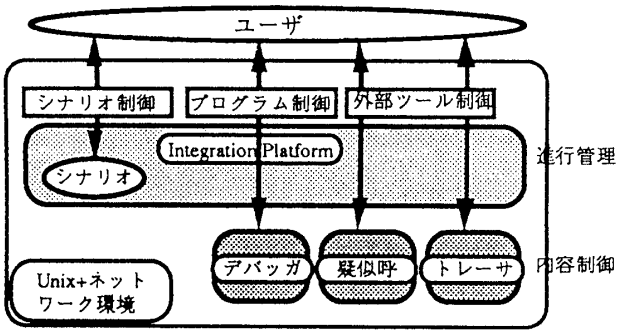

図２　自動試験システム構成

4.3シナリオ記述言語

個々のツールには、試験対象プログラムへ直接操作を行うツールと、それ以外の外部ツールとがある。前者の操作は、ソフトウェア試験という観点からは共通のものであり、使用頻度も高いことから、直接試験者が記述できるレベルのコマンドで行う。後者の操作は、既存ツールの活用・新規ツールの独立な発展を狙い、各ツール共通のコマンドで試験進行レベルを、独自のコマンドパラメータ、制御ファイルの内部データで詳細な指示を行う。この結果、シナリオを記述するための言語は、試験の進行を制御するもの（シナリオ制御）、試験対象を操作するもの（プログラム制御）、外部ツールを制御するもの（外部ツール制御）の３種類で構成される（表１）。

5. おわりに

本稿では、通信ソフトウェアの試験自動化へ向けての一つのアプローチ法について述べた。既に本アプローチに基づいた自動試験システムを、現在開発中のATM交換システムの試験に適用している。今後は、シナリオ作成作業を含む試験時間の短縮、試行錯誤で行う人間の作業に適合するメカニズム等とともに、使用後のフィードバック事項を盛込み、使い易いシステムの実現を目指したい。

【参考文献】

(1)立元, 白石, 黒田, 清野 : "交換ソフトウェアプロセスに関する一考察", SSE92-6 pp.1-6

(2) 清野, 白石, 大久保, 黒田 : "ATMソフトウェア技術" NTT R&D Vol.42 N0.3 1993 pp307-318

表１　シナリオ記述言語（例）

I.F.種別	コマンド	概要
シナリオ制御	サブシナリオコマンド (submit)	サブルーチンシナリオを実行するためのコマンド。このコマンドによってシナリオの部品化、階層化が可能となる。
	試験制御コマンド (scenario)	試験項目毎のテスト条件（試験項目、ログ、エラー手順、タイムアウト手順、結果判定）を設定するためのコマンド。
	変数設定コマンド (setdata)	試験シナリオ中で用いる変数を設定するコマンド。このコマンドによってシナリオ中でのデータ内容の流用や実行回数カウントが可能となる。
	条件文 (if, switch)	試験シナリオ中で条件分岐を行なうためのコマンド。
プログラム制御	基本コマンド (load, run, stop etc.)	プログラムを制御するための基本コマンド　（一般のデバッガ機能に相当する。）
	データ設定／収集コマンド(get, set)	データを設定、収集するときにシンボリックに指定する。従って、プログラムが変更になっても、データ設定、収集指示は影響受けず、シナリオの再利用が可能である。
	ブレイク／チェックポイントコマンド (bpi)	要求ポイントでプログラムの実行を停止させるコマンド。物理アドレスやソース行を指定せず、ソースコード中の論理ラベル（シンボル）により示される。従ってプログラムが変更されてもシナリオは再利用可能である。
外部ツール制御	ツール起動コマンド (exec, fork)	UNIXベースのツール、子プロセスとしてのツールを起動するコマンド。本コマンドのパラメータとしてツールのコマンド列が与えられる。