Appendix B

シンプルなテスト・スクリプトで信頼性と生産性を上げる

SESSAMEの仙人

テスト・プログラム（テスト・スクリプト）は，ソフトウェアの不ぐあいを検出するために作成する「ソフトウェア」である．つまり，テスト・プログラムそのものに不ぐあいが含まれていないという確証はない．ここでは「テスト・スクリプトの構造をシンプルにする」という方針のもと，スクリプトの汎用化・共有化を図った事例を紹介する．（編集部）

以前，筆者がマイコン向けのクロス開発ツール（実際に搭載されるCPUやOSとは異なる環境でプログラムを開発するためのソフトウェア開発ツール）を開発していたとき，アセンブリ言語の記述に関する不ぐあいの報告を受けました．開発チームのまとめ役を引き受けたばかりのころで，チーム内の実情がよくわかりませんでした．報告書を見る限りでは，テストしていないとは思えないような基本的なパターンです．開発者にたずねると，「おかしいなぁ，テストしているはずだけど」と首をかしげています．

テスト仕様書を見ても，その記述のチェックは確かにテスト・パターンに含まれています．テスト報告書にも，テストを実施して問題がなかったことが記述されています．しかし，実際に不ぐあいは出ています．期待値がまちがっているのでしょうか？

開発者といっしょに原因を追及すると，テストを実行するためのシェル・スクリプト[注1]にまちがいがあったことがわかりました．テストの一部がごっそり実行されていなかったのです．スクリプトが非常に長いうえ，マイコンの機種によって実行するテストが異なるため，いつの間にか必要なテストが実行されない状況に陥っていたようです．

これを，単なる担当者の失敗として片づけてよいものでしょうか？ 単純なテスト実行用のスクリプトといっても，一種のプログラムです．その正当性をどう保証するかを考えなければなりません．単にテストを実行しているかどうかだけではなく，想定している実行モジュールを使っているか，必要なパラメータを使用しているかなどを確認する必要があります．スクリプトもプログラムなので，それをテストするためのプログラムを作るべきでしょうか？ それとも，テスト・プログラムやテスト実行用のスクリプトのコード・レビューを実施するべきでしょうか？[注2]

このときに考えた方針は，「テスト・スクリプトの構造をシンプルにする」です．スクリプトが長くなれば検証が必要になるという教訓から，スクリプトを短くし，確実にテストを実施することを考えました．まず，テスト用のテーブルを用意し，1回のテストに必要な情報（変数値の設定，動作モードなど）を1行に書きます（実行するテストの数だけ行があるテスト・テーブルを作成する）．そして，テスト・スクリプトがこのテーブルを1行ずつ読みながら，テストを実施するようにしました（テーブル・ドリブン型）．これにより，スクリプトは決まったパターンを毎回実行するだけになるので，テスト・スクリプトの信頼性はぐっと上がります（**図1**）．

ここで，テーブルをどう設計するかがポイントです．「テスト・パターンが多ければテーブル・データが多くなり，その検証が必要だから，もとの長いスクリプトと同じじゃないか」と思う人もいるでしょう．しかし，スクリプトに散りばめられた情報をチェックするより，情報だけが記述

注1：当時はUNIX環境で開発しており，Cシェル・スクリプトを使用していた．
注2：このような「テストのテスト」を実施しているところは少ないのではないだろうか．テストに潜む不ぐあいは，開発の盲点になりがちである．

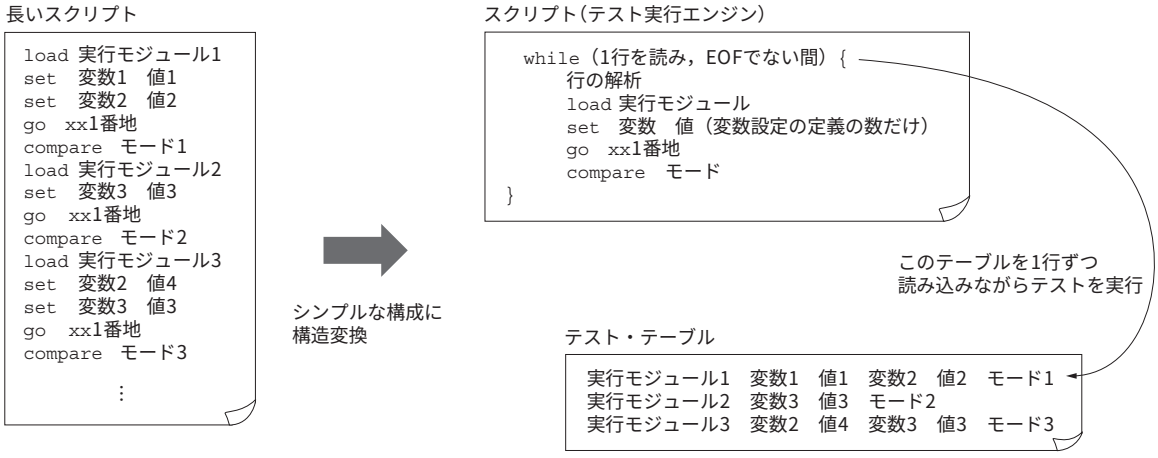

図1 テーブル・ドリブン型のテスト実行

長くなっていたテスト・スクリプトを，実行処理部分と個別のテスト情報部分に分けて，シンプルな構成にした．実行処理を記述したテスト・スクリプトがテスト・テーブルから1行ずつ個別の情報を読み込みながら，テストを実行する．

されているテーブルのほうがずっと見やすく，まちがいも見つけやすいのではないでしょうか．

● テスト実行用のスクリプトを共有する

プログラム開発の各段階に応じて，それぞれについて最適な開発環境（UNIX，Linux，Windows，実機，ICEなど）を使うわけですが，テストに使うスクリプトはできれば同じもの（少なくとも，少々変更した程度のもの）を使いたいと考えます．これは効率化や再利用の観点から，ごくあたりまえのことです．

テスト・パターンの作成をパソコンで行う場合，通常，そのテスト・パターンの検証もパソコンで実施します．また，先ほど述べたようにテーブル・ドリブン型でテストを実施する場合も，テーブルの検証をパソコンで実施します．そこで，パソコンで使ったテスト実行用のスクリプトをそのままデバッガで使用して，信頼性や生産性を上げたいと考えました．また，LinuxやUNIX上でも同じテスト・スクリプトを使いたいとも考えました．

どの環境でも動作するスクリプトにするにはどうすればよいのでしょうか？ マルチプラットホーム対応のスクリプト言語はいろいろありますが，その当時，それらのスクリプトをそのまま使えるデバッガはありませんでした．独自のコマンド・スクリプトを持っているデバッガもありましたが，それほど強力なものではなく，スクリプト間のコンバータを作るのもたいへんです．そこで，同じスクリプトを使えるデバッガを開発することにしました．マルチプラットホームで，デバッガ開発にも簡単に利用でき，GUIも作れるスクリプトを検討した結果，Tcl/Tk[注3]を用いることにしました．テスト実行用のスクリプトをTclで書き，デバッガ用に拡張したTclコマンドを，パソコン側でその動作を模擬する関数として定義することで，意図したしくみをかなり実現できました．

デバッガを作るというのは特殊なケースですが，テスト環境面の信頼性と生産性を上げるために「スクリプトを共有しよう」という考えかたは重要です．できるだけ同じものを使う，つまり情報の共有化により，人間によるエラーの混入を少なくして，全体の信頼性を高めるわけです．

なお，先ほどテストに用いるテーブルを共通で使うと書きましたが，実際には，開発フェーズや環境によってテストの実行に必要な情報量にはバラツキがあり，スマートにはいきません．それでも共通化が重要なので，例えば，大本のテーブルから各フェーズごとに必要となるテーブルをプログラムで生成する，などの手を使います．そのような場合，makeコマンドなどを利用し，ファイルの依存関係を定義して，つねに最新のテーブルを自動で生成するようにしておくことが重要です．ただし，あまり凝りすぎると「テスト・スクリプトの構造をシンプルにする」という方針から外れていくので，注意が必要です．

注3：Tcl/Tkは，John Ousterhout氏によって作成・開発されたスクリプト言語である．