新人技術者のための

ロジカル・シンキング入門 第7回

「ひたすら流すだけ」にさようなら

冴木 元

システムの記事　ビギナーズ

「テスト・ケースはテストを実施する前に作成しておくもの」という当たり前のことを，実際の開発現場では実行できていないと感じている設計者は少なくないだろう．今回は，テスト・ケースを漏れなく，かつ効率良く作成するための考え方を説明する．開発するシステム（モジュール）に応じて，入力パラメータや入力データを考えて場合分けすることで，テスト・ケースを充実させていく．　（編集部）

Gさんは，今月から新しい開発チームに所属することになった，入社2年目の若手のシステム・エンジニアです．

既存のプログラムのテスト工程から参加するようになったGさんに，リーダのPさんは「仕様書を見ながら担当するモジュールのテスト項目を考えてほしい」と伝えました．ところが，この当たり前の要求にGさんは思わず面食らってしまいました．「ん，なんだなんだ？ テスト項目って本当に考えるものなのか…．そんなのは新人研修用の建前だとばかり思っていたぞ…」．

● 納期前日に項目を作成，なんてことをしていませんか？

それもそのはず，Gさんがこの間までいた開発チームでは，テスト項目を用意して動作を確認することはほとんどなかったからです．

そのチームでは，開発スケジュールがあまりにも短期間であったため，コーディングとデバッグに多くの時間を費やしていました．結局，納期を大幅にずれこんで，やっとのことで顧客にシステムを提供したため，テスト項目を用意してバグを見つける暇などなかったのです．実運用開始までの間，本番用のデータをひたすら流し込み，システムがおかしな動作をすればその都度直していたのでした．そのため仕様が明確でない部分も多くあり，目の前で起こった事象がバグなのか仕様なのかをめぐって，深夜にもめごとになったことも何度かありました．

結局，「テスト項目」として納品したドキュメントは，納品日の前日に全員で急きょまとめ上げ，テスト日を工程上問題なさそうな日付に記したものだったのです．

GさんはおずおずとPさんに尋ねました．「あの〜，参考にしたいので，このチームでこれまで使っていたようなドキュメントを見せていただけませんか」．すると，リーダのPさんは**表1**を見せ，手始めにテスト項目をツリー上に分類してみるように伝えました．「項目分けを表にして整理することで，過不足なくテスト項目が整理できるから，みんなそうしている」ということです．「なるほど．これな

表1
ツリー構造によるテスト・ケースの分類
ツリーを使ってテスト・ケースを分けるのはオーソドックスなやり方．分類を大きい方から小さい方へと分けていき，テストの漏れと重複がないようにしていく．問題はテスト・ケースの分け方である．

			テスト・データ	期待する結果
1 動画機能ブロック・テスト	1.1 通常処理	1.1.1 Aモード	Aモード入力データ	正常に描画すること
		1.1.2 Bモード	Bモード入力データ	正常に描画すること
	1.2 エラー処理	1.2.1 CRCエラー時	エラー・データ	エラー保護すること

KeyWord　テスト，モジュール，テスト・ケース，全数テスト，組み合わせ回路，順序回路，ステート・マシン，入力パターン

ら取りあえず何か思い付くかもしれない」．ほっとしたGさんは，与えられた時間，じっくりとテスト項目を考えてみることにしました．

仮に，読者のみなさんがリーダのPさんだとしたら，Gさんが提出してくるドキュメントにどのようなことを期待するでしょうか．

● ロジカル・ツリーの分け方

Pさんが指示したようなテスト・ケースの分類はオーソドックスな方法であり，それ自体は至極まっとうなものです．とはいえ，実際にテスト・ケースを分類するとなると，経験の少ない人であればいろいろと悩むこともあるでしょう．たとえベテランと呼ばれているような人でも，初めて接するタイプのシステムであれば，なにかと工夫しなければならないことが生じることに気づくと思います．

ここでは，組み込み開発で必要となりそうなテスト・ケースを，次の二つの点に焦点を当て，なるべく一般論（アプリケーションに依存しない）で解説します．

- モジュール単体のテスト・ケースの考え方
- 単体テストと結合テストのすみ分け

● モジュール単体のテスト・ケースを考える

モジュール単体のテスト・ケースを考える場合，誰でも思い付きそうなことは，「モジュールに与えられたパラメータの組み合わせを考え，それらを一通りテストする」というものでしょう．**表1**のようなツリーを使った分類でも，パラメータの種類と個別のパラメータを分けてテスト・ケースを充実させていくのが普通のアプローチの仕方だと思います．実際，一般に普及しているソフトウェア工学ですと，このような考え方にたって「複合条件網羅」といったテスト・ケースの設計条件が整理されているようです．読者の中には情報処理試験の受験のためにそうした知識を覚えた方もいるでしょう．

しかしこの考え方には，見落とされている点があると筆者は考えます．それは，そのモジュールに「ステート」があるかないかでテスト・ケースに対する考え方が変わってくるということです．

ステートの有無というのは，要するにテスト対象のモジュールの中にスタティック・データが存在するかどうかです．スタティック・データとは，モジュール処理が終わっても次の処理まで値を保持するデータのことです．C言語であれば，外部変数を使うとか，`malloc`を使って実装するデータです（連載第2回，本誌2006年6月号，pp.60-66を参照）．このようなステートが存在するモジュールの場合，単純に入力パラメータの組み合わせを網羅してもテスト・ケースに漏れが生じます．

● ステートがなく，自動化できれば全数テストがお勧め

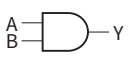
まず，ステートがないモジュールから考えることにしましょう．ここでは，このようなモジュールをハードウェア回路に倣って，「組み合わせ回路」と呼ぶことにします．

図1のようなAND回路を例にとって考えてみましょう．AND回路とは，**図1(b)**の真理値表のような動作を行う回路のことです（ハードウェア技術者の方にとっては当たり前過ぎて今更解説の必要などないかもしれないが…）．**図1(b)**を見ると，4通りの入力の組み合わせに対して，Yの期待値が与えられていることが分かります．入力信号の状態としては‘0’または‘1’しかないため，AとBの2種類で$2^2 = 4$通りの入力パターンが考えられるというわけです．

このように，組み合わせ回路については，起こり得るすべての入力パターンを拾い上げれば全数テストになります．回路がステートを持たないため，入力に対して出力が一意に決まるということがその理由です．

この考え方によると，どんなに複雑な回路（あるいはモ

図1
AND回路の例
組み合わせ回路の場合，入力パターンをすべて掛け算すると全数テストになる．組み合わせ回路はステートを持たないからである．



（**a**）回路記号

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

（**b**）真理値表

図2
全数テストとその限界
全数テストを実施するのが確実だが，テスト・ケースが膨大になることもある．やみくもにテスト・ケースを増やせばよいというものでもないが，テストの実施を自動化して効率化を図れるのであれば頑張って全数テストを行うことも大事．