B-33

状態遷移図とユーザータスク・フレームワーク併用のテストケース設計

Test Case Design with State Chart Diagram and User Task Framework

稲田 貴史†
Takashi Inada †

1. はじめに

テストケースのカバレージアップと作業の効率化を計るためテストケースを作成する前にグラフや表を作成する。(文献 1)
しかし従来の手法はプログラムのみを対象とするため、クロスプラットフォームのプログラムテストには不十分である。ここでは有限状態テスト法（文献２）に独自のユーザータスク・フレームワークを考案しテストケース設計プロセスを統合化した。

2. 概略

2.1. 有限状態テスト法の概略と問題点

もともとハードウェア論理のテストに適用された有限状態モデルは、メニュー駆動型ソフトウェアをテストするのに優れたモデルと言われる。また、オブジェクト指向設計にも使用されるため、オブジェクト指向のソフトウェアをテストする上でも重要なテスト技法とされる。（文献２）

基本的には Myler （文献１）が示している原因ー結果グラフの作成手順と同じである。すなわち、１） 仕様を ”扱いやすい”断片に分割し、2) 原因 （入力条件の同値クラス、入力値の集合）、と結果 （出力条件、またはシステムの変化） を示す。プログラムの遷移プロセスをこの手順でグラフ化していけば、ある程度高いカバレージのテストケースを設計することができた。

以下は、状態グラフ、または状態遷移表作成後のチェック項目である。(文献２)

入力のコード化を検証する。
出力のコード化を検証する。
初期状態から始まるか検証する。
全ての終了状態に到達するかを検証する
状態を確認する。
余分な状態が存在するか検証する。
全ての遷移を確認する。

表１．状態遷移グラフ検証項目

これを見る限り、検証項目がユーザー環境に関してはあまり考慮されていない。このため多様な既存ライブラリを使用することが多い現代のソフトウェアではカバレージが不十分であり、抜けが起こる可能性がある。

†（社）日本アイ・ビー・エム株式会社
ソフトウェア開発研究所

2.2. ユーザータスク・フレームワークの提案

テスト漏れを避けるため、ユーザータスクをフレームワーク化し、ソフトウェアを分析、テストケース設計をプロセス化することを提案した。

１．トランザクション関連 （ユーザーシナリオ）
２．データ関連 （入力、出力、外部データ）
３．ユーザー環境関連
a. オペレーティングシステム
b. 協業する他のプログラム
c. テスト対象ソフトウェアのオプション
d. 割り込み・共有
e. その他

表２．ユーザータスク・フレームワーク

表２がユーザータスクフレームワークである。全て入力と出力を左右する要素であり、項目１はプログラムの状態遷移を元にユーザーシナリオの分析、項目２はユーザーがどんなデータを使用する可能性があるかを分析する。項目３はユーザー環境を分析する。

3. 実例 （実際のプロセス）

3.1. 有限状態テスト法の状態図の作成

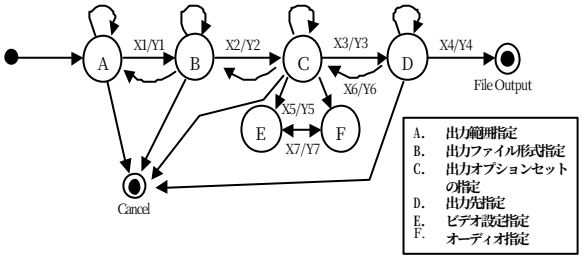

図１．ビデオファイル出力ウィザード

図１はある Windows ソフトウェアのビデオファイル出力ウィザードの状態図である。各入力 Xn により状態が遷移し、それによってそれぞれの出力 Yn が出力されることを示している。仕様書によると出力ファイルフォーマットは AVI, Windows Media Format, QuickTime の３つとある。後は各 Xn, Yn の要素をリストし、文字列なら構文テスト、数値ならドメインテスト（文献２）を適用し、テストケースの作成が行える。

3.2. ユーザータスク･フレームワークを使用したテスト項目の分析

3.2.1. トランザクション関連

作成した状態図を参考に、ユーザーが行うであろうシナリオを全て洗い出す。たとえば、File Output をする物だけでも、

ABCD, ABC (EF) CD, ABABCD, ・・・

など多数ある。Cancel の場合も列挙する。(EF に関しては入れ子構造を考え、ひとつの状態とすると簡略化される(文献２)

3.2.2. データ関連

作成した状態図を参考に各状態でのユーザー入力のデータタイプを調べる。表３のようになる。それぞれの状態で入力されたデータは次の状態に遷移する際、出力と考えることができる。

状態	データ名	データタイプ
A	出力範囲	Enum
B	出力ファイル形式	Enum (AVI, WMV, QuickTime)
C	出力設定	Enum
D	出力先	Binary

表３．各状態のユーザー入力データ

次に初期入力を分析する。表４になる。

データ名		データタイプ
音声	WAV	Binary
	AIFF	Binary
	MP3	Binary
ビデオ	AVI	Binary
	MPEG1	Binary
	MOV	Binary
ビデオ効果		Enum (ただしサブルーチンが走る)
トランジッション		Enum (同上)
テキスト		Binary
テキスト効果		Binary (同上)

表４．初期入力データのタイプ

3.2.3. ユーザー環境関連

今回の場合のユーザー環境のパラメーターを以下に列挙した。

項目	パラメーター
OS	Windows95, 95OSR2, 98, 98SE, Me, NT, NT SP3, NT SP6a, 2000)
他のプログラム	QuickTime (有無、バージョン)
オプション (環境設定)	ビデオ設定プリセット、
割り込み･共有	QuickTime Player, Media Player など。状態 D から File Output の時にストレスがかかる。
その他	なし

表５．ユーザー環境パラメーター

3.3. テスト項目の統合

基本的にデータがシナリオの入出力となり、環境がそれの入出力という考え方ですすめる。具体的には、

1. 各シナリオに対してのデータバリエーションをふやし、
2. 各（シナリオ＋データバリエーション）に環境のバリエーションをふやす。

という段階を踏み、テストケースを作成していく。データやユーザー環境のバリエーション、組み合わせ選択はドメインテスト法などを用いた。

４. 結果

以下のように、３個のソフトウェア製品の一部機能に、このテストケース設計法を適用した。問題報告数は全て 0 個であるが、各製品ともテスト期間が 1 ヶ月程度と短かった(通常 3 ヶ月)ことから、効果があったことがわかる。（各機能テストについてはテストケース作成も含め 1 週間程度）

	問題報告数 (個数)	プロット時（出荷後）
Video (事例)	０	１２ ヶ月後
Product A	０	５ ヶ月後
Product B	０	１．５ ヶ月後

表６． テスト対象機能の出荷後の問題報告数

５. 検証と考察

この方法を用いると、テストケース設計をある程度機械的に処理できるため、高いカバレージのテストケースをより短時間に作成できる。そのため効率よくテストケース作成ができた。しかしながら予測不可能な欠陥の発見ができない可能性や状態図が元来持っている弱点も依然残っている。(文献２)

これからはこれをより発展させ、予測不可能な欠陥の発見の研究やテストケース作成の自動化などについても研究したい。

６． まとめ

ユーザータスク・フレームワークを導入することで、クロスプラットフォームのソフトウェアのテストケース設計に実用的なプロセスを提案した。

謝辞

コメントを下さったテスターの方々に感謝します。

参考文献

1) Glenford J. Myers: ソフトウェア・テストの技法 (The Art of Software Testing), 近代科学社, 日本 (1980)

2) Boris Beizer: 実践的プログラムテスト入門 ソフトウェアのブラックボックステスト (BLACK-BOX TESTING), 日経 BP 社, 日本 (1997)