

## 詳説 FV 表

2009/7/12 (日)

秋山浩一

### 1. FV 表の位置づけ

ソフトウェアのテストを実施する時には、テストオブジェクトを識別し、それを様々な「視点」から扱いやすい粒度になるまで分解する必要がある。

分解の仕方には、鈴木三紀夫および池田暁によるマインドマップを使う方法や、西康晴による NGT を用いる方法が有名であるが、いずれにしてもツリー状の図は、スペースをとるのが難点である。ほんの数枚の仕様書であっても、マインドマップを書き出すとすぐに A3 では足りなくなり、模造紙が欲しくなるのだ。

したがって、ツリー状のグラフではなく、表で表現しようという考えも存在する。たとえば、Rick Craig による「インベントリ」リストや、湯本剛による「ゆもつよメソッド」が、知られている。これらは、主に機能を表の縦軸に割り付け、横軸には、品質特性等の「視点」を割り付け、機能と視点の関連性を明らかにし、関連するセルにおいて、テストオブジェクトを識別する方法である。

表を用いてテスト分析をまとめるやり方は、多くの企業で採用されている標準的な方法であるが、「視点」の選び方（抽象度）が適切でなく細かすぎる場合には、巨大な表となり扱いにくいという問題点がある。そこで、筆者は、テストオブジェクトの認識と分解のために、列数が多くならない FV 表を考案・改良しそれを使用している。

FV 表は、HAYST 法の中で、テスト分析を行うためのメソッドであるが、テスト技法を用いたテスト設計については、テスト分析の次のフェーズであるテストアーキテクチャ設計以降で実施するため、直交表を用いたテスト以外であっても FV 表を使用することができる。

## 2. FV 表の形式

FV 表とは、「Function Verification Table」の略であり、基本のフォーマットは、

No.	目的機能(F)	検証内容(V)	テスト技法(T)

表 1 シンプルな FV 表

という単純なものである。それぞれの列の意味については、次項で解説する。

バリエーションとしては、「市場リスク（プロダクトリスク）」、「技術リスク（プロジェクトリスク）」、「FLFP（Factor Level Function Point＝因子水準数を用いた複雑度でありテスト工数見積りに使用する）」の 3 列を追加した、以下の形式のものもある。

No.	目的機能(F)	検証内容(V)	テスト技法(T)	市場リスク	技術リスク	FLFP

表 2 リスクと見積りが付与された FV 表

しかし、複雑な表が必ずしもよいものというわけではない。私は、主にシンプルな表 1 を用いている。

FV 表は、このように単なる 2 元表であり、階層構造（や入れ子構造）は持たない。

しかし、目的機能には、階層があるものがほとんどであるため、「No.」列を「1./1.1/1.1.1」といったように階層付けることが多い。また、「目的機能」を書くときにインデントをすることで視認性を向上し、目的機能間の関係を明らかにすると良い。

FV 表は、EXCEL などの表計算ソフトでまとめることをお勧めする。使い慣れたソフトウェアを用いることで、多くの方が FV 表に目を通し、テスト分析に対するレビューすることが重要だからである。

### 3. FV 表の使い方

本項では、FV 表の使い方について、解説する。

#### 3.1 目的機能

まず、テスト対象のソフトウェアの機能仕様書を開き、そこから、「機能」を取り出す。次に、仕様書から取り出した「機能」を「目的機能」の欄にコピー&ペーストしたくなる気持ちになるが、それはしてはならない。機能仕様書を書き写したくなる気持ちをぐっと堪え、その機能が持っている目的について考え、それを記入する。

ほんの 1 分でよいので、本書を閉じて「ソフトウェアの世界に、目的を持っていない機能があるかどうか？」について考えてみて欲しい。目的を果たしていないソフトウェアは数限りなく存在するが、目的そのものがないソフトウェアは存在しない。そもそも、ソフトウェアは自然界にある神が創ったモノではなく、人間が作ったモノであるから、そこにはなんらかの製作意図、すなわち、果たすことが期待される目的があるのである。それを FV 表では「目的機能」と呼んでいる。

さて、「機能」ではなく、「目的機能」を記入する意味であるが、それは、開発とテストの違いに由来する。開発は、ソフトウェア製品と言うモノを作るのが仕事である。したがって、ソフトウェア製品の設計図といえる機能仕様書には、どのような動きをするか、どの位のレスポンスで動作するように作る必要があるか等々の「機能仕様」が書かれている。

ところが、ソフトウェアテストにおいては、機能が仕様書通りに「正しく動作」することの確認はもちろんのこと、ユーザが求める「正しい動作」をすることを確認する必要がある。「正しい動作」とは、その機能が、ユーザの目的（＝したいコト）を達成することができるということである。

まとめると、「機能仕様書には、ソフトウェアというモノを開発するための情報が書かれているが、それがどのようなコンテキストで使われるかといったコトについては、記載されていない。したがって、FV 表の目的機能欄にはモノの情報だけではなく、コトの情報を記載することで正しい動作が行われていることのテストを実施する」である。

次に、機能から目的機能を発見する方法について述べる。基本的には、テストベースから機能仕様書を作成する時に使用した前工程のドキュメント（要求仕様書等）を入手し、そこに記述されている機能に対応する要求を書き写せばよい。

テストベースが DOORS のような要件管理ソフトウェアによって管理され、要求と仕様のトレーサビリティが確保されているようなソフトウェア開発環境では、この作業は一瞬で完了する。しかしながら、多くの場合、要件管理ソフトウェアはもとより、ドキュメントの構成管理も怪しい場合が多い。したがって、現実的には、機能から目的機能を想像する必要が生じる。

世の中の製品を見ると、機能がてんこ盛りである。ほとんど使われていないという実態が時々ニュースになるが、実は、何に使ったらよいか分からない機能も多い。「何が起こるかはわかるが、何に使うのかわからない」のだ。つまり、そのような機能は目的を見失っている。FV 表の目的機能を埋める際には、機能からその目的を掘り起こす必要があるのだ。その方法は、「Why 展開」と呼ばれている。機能があったら、「なぜ?(Why?)」と問いかけてみるのである。

たとえば、「下線を引く」という機能があったら、Why?と問いかけることで「文章の重要な部分を目立たせるために文字の下に線をひく」という目的を含んだ機能説明を目的機能の欄に記入する。

### 3.2 検証内容

次の欄は、機能仕様書から発見した目的機能に対する検証内容を記述する。

検証内容に書くものは、「どのようなテストをしたらユーザの目的に叶う機能が作りこまれたことが確認できるか」である。

「ユーザの目的に叶うか?」の確認方法を抜け漏れなく記述するためには、5W2H（目的機能の Why?を含めると 6W2H）が入っていることを確認する必要がある。すなわち、

What? : 機能仕様（正常、異常）

When? : いつ使われるか

Where? : どこで使われるか

Who? : 誰が使うのか

Whom? : 誰のために使うのか

How? : どのように使うのか

How much? : 価格はどれくらいのソフトウェアなのか

を分析した上で、記載を行う。たいていの目的機能の場合は、頭の中で 5W2H の検討ができるが、経験が浅いうちは、マインドマップを使用すると良い。マインドマップの中心に目的機能の名前を記述し、目的機能の Why?の枝を含めて、8本の主枝を伸ばし様々なユー

ザシーン（＝コンテキスト）を想像しながら、機能の使い方（＝コト）を記述していくのである。

### 3.3 テスト技法

最後は、テスト技法の欄である。ここには、その目的機能をテストするために使用するテスト技法を記述する。組合せを考慮する必要がない機能はほとんどないため、HAYST 法と書く場合が多いが、論理関係が複雑な場合は、原因結果グラフ法や、CFD 法と書くこともある。また、状態遷移が絡む場合は、2-switch テスト法と記述するかもしれない。

FV 表の 1 行は、目的機能で、テストのための Why、検証内容で What を、そしてテスト技法で How を抑えるという関係にある。つまり、ある機能の本来の目的を把握し、それが実現できたかの確認内容を記述し、最後に確認方法を記述する。これは、ソフトウェア開発における、課題（Why）、仕様（What）、設計（How）と同じ構造である。

### 3.4 市場リスク、技術リスク、FLFP

オプション的な列として、「市場リスク（プロダクトリスク）」、「技術リスク（プロジェクトリスク）」、「FLFP（Factor Level Function Point＝因子水準数を用いた複雑度でありテスト工数見積りに使用する）」の 3 列を記述することがある。

市場リスクとは、その目的機能にバグがあった場合の市場インパクトのことである。「大・中・小」で記述することが多いが、5 段階で記述しても良い。機能の重要度と、使用頻度との足し算や掛け算でリスクを表現したがる人がいるが私は賛成できない。きちんと、目的機能が理解できていれば計算する必要はなくリスクを判断できるはずであるし、その方が他の目的機能とのリスク比較がしやすい。実際に、足し算や掛け算派の人で、リスクを計算で出した後に微調整と言って機能の重要度と、使用頻度の数値を調整している人がいるが、本末転倒である。

技術リスクとは、簡単にいうと、開発の困難度である。こちらも、「高・中・低」の 3 段階で記述することをお勧めしている。もちろん、5 段階でできるのならそれでもよいが、いずれにしても、市場リスクのレベル数と合わせておいた方がリスクマトリクスを作りやすいし判断しやすい。各々 3 段階であってもリスクマトリクスでは、「市場リスク×技術リスク＝3×3」のマトリクスになるからである。

FLFP とは、COSMIC-FFP に似たファンクションポイント計算法で HAYST 法独自のものである。こちらは、因子・水準を明らかにした後に計算した結果を記述する。

## 4. FV 表の狙いと効果

### 4.1 FV 表の狙い

FV 表の狙いは次の 2 点である。

- テスト対象のカバレッジの確保
- テストオブジェクトの非機能要件を含めたカプセル化

#### 4.1.1 テスト対象のカバレッジの確保

テストに抜け漏れは厳禁である。市場に出た後に、「そこはテストしていませんでした」という言い訳は最もしたくないことの一つである。

テストカバレッジを確保する方法は一つしかない。それは、当該ソフトウェアの開発において最も多くの人が参照する開発ドキュメントを元にして、それに紐付けながらテストを作っていくのである。

FV 表が機能仕様書から機能を抜き出している理由はそこにある。本来は、要求仕様書の方がテスト分析には有効であるが、要求分析の後工程である基本設計あたりにならないと、ソフトウェアの仕様が固まらないのが常である。これは、躰の問題もあるが、機能仕様書を書いている最中に新しいアイデアが浮かび、それを盛り込むという面もあるため一概に悪いことと切り捨てるわけにはいかない。

したがって、FV 表では、機能仕様書と 1 対 1 対応することで、テストカバレッジを確保している。

#### 4.1.2 テストオブジェクトの非機能要件を含めたカプセル化

目的機能という「視点」で、テストオブジェクトを分解することの利点は、テストオブジェクトをセキュリティ要件や、非機能要件も含めてカプセル化できる点にある。

ユーザは何かの目的を実現するためにソフトウェアを使用する。通常、一つの目的を実現する途中でセキュリティ要件が変わることはありえない。また、目的には、それを実現する時の振る舞いである性能や操作性といった非機能要件がついてまわる。

たとえば、ある目的を実現するために我慢できないくらいの時間がかかるのであったらそ

の目的機能は使用されないのである。

つまり、目的機能単位に非機能要件の確認をすると非常に効率が良い。そうすれば、機能を確認する際に、非機能要件の「計測」をすればよいだけだからである。

## 4.2 FV 表の効果

FV 表は非常にシンプルな表ではあるが、使いこなすためにはかなり高いテストスキルを必要とする。しかし、考えてみれば、テスト分析からテストアーキテクチャ設計までの工程が最もテストで難しいところであり、その先のテスト設計はテスト技法を知っているか否かの差しかないし、テスト実装・実施にいたっては自動化ができる/できないといったテストとは少々離れた技術分野の話となる。

FV 表を正しく使用すると、必ずユーザの使用目的や市場条件を考えることになる。初めて使用する人はこんなに検討しなければならないことがあるのかと驚くが、検討しなければいけないのである。それを不十分なまま経験や勘や過去のトレンドでテストをするから問題が取り切らないまま製品が市場導入されてしまうのである。

本稿がテストによるソフトウェア品質の向上に少しでも役立てば幸いである。

以上