要求定義工程での品質保証とシステムテストのテストケース自動生成新技法

Feature Oriented Testing : New Method for Quality Management and Test Case Modeling to generate Test Cases Automatically

テクマトリックス（株）　システムエンジニアリング事業部
TechMatrix Corporation
○西田　啓一 [1)]　　中島　裕生 [2)]
○Keiichi Nishida[1)]　　Yusei Nakashima[2)]

Abstract　Software testing consists of two phases. One is test case designing phase where developers/testers create test cases which describe the content of testing to be performed, and the other is testing phase where the created test cases are executed and validated. If there are unnecessary or missing tests during test case designing phase, testing efforts become meaningless. These days, software become more and more high-functional and rapidly expanded. This increases the number of test case input parameters and makes constraints among parameters become more complex. This situation demands more effort for developers and testers who are responsible for designing and creating combination tests. Some testing tools, such as PICT, support combination tests which need to take account of complicated parameters, however, these tools have not become standard in many organizations since they require cumbersome tasks, for example, entering complicated inputs in order to generate test cases. This paper describes Feature Oriented Testing technique developed to solve these problems involving combination test, and reports the result of validation tests on effectiveness of this technique performed on Web applications.

謝辞　本論文の作成に当たり、ご協力を頂いた産業技術総合研究所の北村崇師氏に感謝します。

1. はじめに

ソフトウェアテストにおいて、Pair-wise 法とは組み合わせ技術を用いたテストケース生成技法の一種である。組み合わせテストでは、テスト対象システムのテスト関心事について、その構成要素の「因子（もしくは、パラメータ）」とそれが取りうる「水準（もしくは、値）」、さらに「それらの間の相互依存関係を論理式で表現される制約式 1 」をテストモデルとして抽出し、その制約式を満たすように因子・水準を組み合わせることでテストケースの集合（テストスイーツ：Test-suite）を作成する。Pair-wise 法とは、そのように作成するテストスイーツが全ての可能な水準のペアを少なくとも一度は含むという網羅基準（Pair-wise 網羅基準）のこと、もしくは、その網羅基準を満足するテストスイーツ、それを用いたテスト実施方法を言う。Pair-wise 法の妥当性は、不具合の多くが少ない数の因子間の相互作用によって惹起されるという経験や実験に基づき示される [1]。Pair-wise 法の必要性は、テストケースの数を根拠を持って減らすことができる点にある。基準を設けず全網羅的にテストケースを作成すると、テストケースの数が指数的に増えてしまう。Pair-wise 法は上記の網羅基準の妥当性に基づき、テストケースの数を大幅に削減できる。近年では、 PICT [6] や ACTS [4]、CIT-BACH [5, 8, 9]、AETG [2] など、Pair-wise 法を支援するツールも整備されつつある。これに伴い、産業界においても Pair-wise 法が使用されるケースも増えてきている。

Pair-wise 法は有効なテスト手法であるが、実世界のソフトウェア開発現場へのより広い普及に向けていくつかの課題を抱えている。その主な課題の一つが、「正確なテストモデルの作成」である。Pair-wise 法を実際に使用する際に、開発者に課される作業の一つにテストモデルの作成がある。上述の通り、「テストモデル」とは、テスト対象システムのテスト関心事の構成要素である「因子」と「水準」、さらに「それらの間の依存関係を命題論理式で表す制約式」のことである。

テクマトリックス（株）　システムエンジニアリング事業部
TechMatrix Corporation System Engineering Division

東京都港区高輪４－１０－８ TEL03-5792-8624 nisida@techmatrix.co.jp 10-8,4-CYOME,TAKANAWA,MINATO-KU,TOKYO JAPAN
1) Engineer
2) Engineer

こうしたテストモデルを正確に高品質に作成することは、Pair-wise 法の有効性を保つための必要条件である。Pair-wise 法では、「因子」と「水準」、「制約式」のいずれも正しく抽出しなければ、テストすべき水準ペアに漏れが生じ、ひいてはテスト漏れを惹起する。

PICT や ACTS は与えられたテストモデルについて漏れのないテストスイーツを生成するが、そもそもテストモデルに漏れがあるとその有効性を失う。漏れのあるテストモデルから、漏れのないテストスイーツを作ることに意味が無いからである。一方で、こうしたテストモデルを正確に作成することは容易な作業ではない。一般に、テストモデルの作成は開発者が要求仕様書等を分析しつつ行うが、複雑なシステム等を対象にする際には、分析の際に考慮漏れや重複等が発生しがちで、つまり正確性が低下しがちである。また、近年のソフトウェアの大規模化・複雑化に伴い、テストモデルの品質確保はより困難な状況になっている。このように、Pair-wise 法の実世界でのより広い普及に向けては、各適用現場において、いかに正確なテストモデルを作成できるかが課題になっている。同様の課題は、最近 Kuhn ら [3] によっても指摘されている。

本論文では、テスト技法 Feature Oriented Testing [7]（以後、FOT と呼ぶ）を紹介する。FOT はロジックツリー（And-Or 木）の拡張を用いたモデル（ダイアグラム）ベースのテスト分析・設計技法を提供する。拡張ロジックツリーを用いたシステマティックな分析技法を用いて、漏れ抜けを低減したより正確なテストモデルの設計を支援する。さらに、そしてそうした拡張ロジックツリーによるテスト設計から Pair-wise 法（もしくはその自然な拡張である N-wise テスト法、但し N = 1, 2, 3,...）のテスト網羅基準を満たすテストスイーツを機械的に生成できる。さらに本論文では、FOT の有効性を検証するために、実際の Web アプリケーションのテストケース作成に適用した結果を報告する。

2. FOT について

FOT は、拡張ロジックツリーを用いた系統的なテストモデルの作成（テスト設計）の支援、及び、そうしたテストモデルから Pair-wise 網羅基準を満たすテストケースの機械生成の 2 つの側面を持つ。本節では、それぞれの側面についてその概要を説明する。

2.1 FOT におけるテストモデルの作成

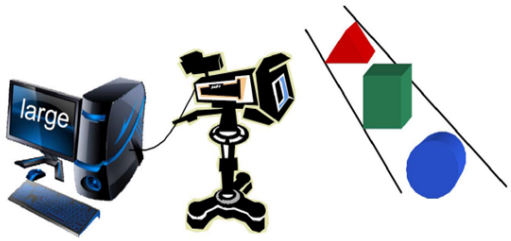

図 1

図 1 に示す 、ベルトコンベア上を流れる様々なブロックのサイズを識別するコンピュータビジョンシステムを例題として、FOT におけるテストモデルの作成（テストケース設計）の概要を示す。上述の通り、FOT では要求仕様を基にテストモデルを設計するテスト技法であり、ブラックボックステストに分類されるテスト技法である。ここでは、まず図 1 に示すコンピュータビジョンシステムの要求仕様を与え、その後、それを基に FOT によるテストモデルの作成例を説明する。

コンピュータビジョンシステムの要求仕様は下記の通りである。

要求１　ブロックは１個ずつ流れる
要求２　ブロックはサイズ、色、形の３つの属性をもつ
要求３　ブロックのサイズは、大、小のいずれか
要求４　ブロックの色は、赤、緑、青の何れか
要求５　ブロックの形は、四角、三角、丸のいずれか
要求６　三角形には、二等辺三角形、正三角形、不等辺三角形の３種がある
要求７　赤色で三角形のブロックはない。（つまり、赤色のブロックは必ず四角または丸のいずれかである）

要求８　　サイズ小のブロックは緑色のみである

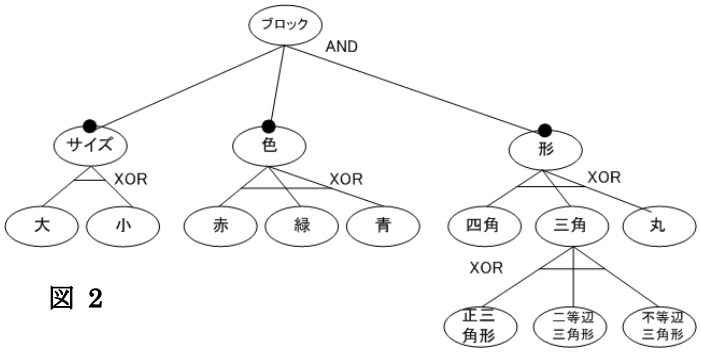

図 2

この要求仕様に基づき、FOT の拡張ロジックツリーを用いて作成したテストモデルの設計例を図 2 に示す。

FOT では、拡張ロジックツリーの木構造を用いて、木構造の根ノード（Root node）を頂点とするトップダウンな系統的な分析を行いつつ､テストモデルを作成する。木構造の根ノードは、テスト対象システムのテスト関心事である。このコンピュータビジョンシステムの例では、テスト対象システムの関心事は、コンピュータビジョンシステムの入力ドメインである「ブロック」である。よってこの例では「ブロック」を木の根ノードとする。

FOT の基本的な分析は､「根ノードを頂点とした分解（division）の繰り返しによる分析」である。テスト関心事を表す根ノードを頂点として、トップダウンに、テスト関心事の様々な構成要素（｢テスト観点」とも呼ぶ）で分解を繰り返すことで分析を行い、テストモデルの作成を行う。図 2 に示すテストモデルでは、根ノードの「ブロック」を頂点に、まず､「大きさ」と「色」､「形」の 3 つのテスト観点で分解している。同様に､「サイズ」は「大」と「小」の 2 つ､「色」は「赤」と「緑」､「青」の 3 つ､「形」は「四角形」と「三角形」､「丸」の 3 つのテスト観点よって、それぞれ分解される。「三角形」に関しては、さらに「正三角形」と「二等辺三角形」､「不等辺三角形」の 3 つの下位概念により分解される。さらに、こうした分解のそれぞれは AND 分解、もしくは、OR 分解 2 によって区別される。分解が直行的なテスト観点によって分解される際には AND 分解、選択的な概念で分解をする際には OR 分解を適用する。例えば、「ブロック」の「大きさ」「色」「形」の 3 つによる分解は、それぞれの概念が互いに直行するため、AND 分解を適用する。一方で､「サイズ」の「大」と「小」の分解は､「大」と「小」が互いに選択的な概念であるため、OR 分解を適用する。「色」と「形」､「三角形」の分解も同様に、OR 分解が適用される。このように FOT のテストモデルは、基本的な木構造の頂点を開始とするトップダウンな繰り返しの分解による分析と AND/OR 分解の区別により、ロジックツリーを成す。

さらに、FOT のテストモデルでは、こうした木構造を成す分解による分析によって表現される分解関係に加え、木横断的制約（Cross-Tree Constraints）と呼ぶ、4 種類の木構造を横断する関係を記述する制約演算子（mutex、requires、attaches、removes）で拡張されている。それぞれは二項関係の制約演算子であり、大まかな意味は以下の通りである。

- a mutex b: 「a と b は両立しない。」
- a requires b: 「a の時は必ず b である。」
- a attaches b: 「a の時のみ、b を考慮する。」
- a removes b: 「a の時は、b を考慮しない。」

要求 7, 8 の制約は、先述の分解関係によって表現するのは難しく、一方で、この木横断的制約を用いることで容易に記述可能である。図 2 のテストモデルでは、要求 7 と 要求 8 がそれぞれ､「赤 mutex 三角形」と「小 requires 緑」として表現されている。

この拡張ロジックツリーによる表現されたテストモデルはテストケースの集まり（つまり、テストスイーツ）を表現している。ロジックツリーにおける OR ノードを「因子」､その子を「水準」とみなすことにより、テストケースを得ることができる。但し、AND 分解及び OR 分解されたノードを、それぞれ AND ノードと OR ノードと呼ぶ。表 2.1 に、ロジックツリーにおける OR ノードを「因子」その子を「水準」とみなし、組み合わせテスト技術による、全網羅的なテストケースを作成した結果を示す。

表 1

	サイズ	色	形	三角形
1	小	赤	丸	
2	小	青	丸	
3	大	緑	丸	
4	小	緑	丸	
5	小	青	四角形	
6	小	赤	四角形	
7	小	緑	四角形	
8	大	緑	四角形	
9	大	緑	三角形	正三角形
10	小	青	三角形	正三角形
11	小	緑	三角形	正三角形
12	大	緑	三角形	二等辺三角形
13	小	緑	三角形	二等辺三角形
14	小	青	三角形	二等辺三角形
15	小	青	三角形	不等辺三角形
16	大	緑	三角形	不等辺三角形
17	小	緑	三角形	不等辺三角形

2.2 Pair-wise テストケースの生成

FOT では、拡張ロジックツリーを用いて作成されたテストモデルから、Pair-wise 網羅基準を満たすテストスイーツを作成できる。ロジックツリーにより表現されたテストモデルの OR ノードを「因子」、その子を「水準」とみなすことにより、全ての可能な水準ペアが出現するようにテストスイーツを作成するのである。FOT では、その際に、PICT や ACTS などの既存の Pair-wise 法支援ツールを利用して、Pair-wise 網羅基準を満たすテストスイーツを生成する。具体的には、FOT では、拡張ロジックツリーで表現されたテストモデルを、PICT や ACTS 等の Pair-wise テストケース生成ツールの入力形式に変換し、それらのツールをテストケース生成エンジンとして用いることによりテストスイーツを作成する。

```
# 因子とその水準の列挙
サイズ: 大, 小
色: 赤, 青, 緑
形: 丸, 三角形, 四角形
三角形: 正三角形, 二等辺三角形, 不等辺三角形, dummy
# [要求8] サイズ小のブロックは緑色のみである。
IF([サイズ] = "小") THEN [色] = "緑";
# [要求7] 赤色で三角形のブロックはない。
NOT ([形] = "三角形" AND [色] = "赤");
# 階層構造を表現するための制約式
IF([三角形] = "正三角形") THEN [形] = "三角形";
IF([三角形] = "二等辺三角形") THEN [形] = "三角形";
IF([三角形] = "不等辺三角形") THEN [形] = "三角形";
NOT([形] = "三角形" AND [三角形] = "dummy");
```

図 3

図 2 のテストモデルを、PICT 表現に変換した例を図 3 に示す。PICT 表現では、「因子とその水準の列挙」部と命題論理を使って記述される「制約式」部の 2 つの記述部に分けることができる。図 3 に示す PICT のテストモデルの因子とその水準リスト部には、図 2 のテストモデルの因子（ OR ノード)と水準(OR ノードの子)が列挙される。但し、「三角形」の因子には、その階層構造を表現するために、その水準に加えてダミーの水準「dummy」が追加される。図 3 の制約式として記述される 7 行目 から 11 行目は、要求 7 と要求 8 を表現している。さらに、同じく制約式として記述される 12 行目から 16 行目では、図 2 のテストモデルの「三角形」の箇所に見られる階層構造を表現するための制約記述である。このように FOT のテスト

モデルに特徴的な性質である木を構成する階層表現は、PICT 表現ではダミーの水準と制約式を工夫することで表現できる。

このように、FOT では、Pair-wise 網羅基準を満たすテストスイーツの作成を、既存の Pair-wise テストケース生成ツールを介して行う。このテストケース作成技法の利点は、様々な Pair-wise テストケース生成アルゴリズムやツールを選択・切り替えて利用できることである。PICT や ACS、 CIT-BACH、AETG などのツールは、生成速度が高速なものや生成されるテストケースの数など、それぞれテストケース生成に使われているアルゴリズムが異なる。その他にも、ツールに備わる機能や知的財産権等の使用環境もそれぞれのツールにより異なる。FOT では、こうした異なるアルゴリズムの性質や使用環境を持つツールを、目的に応じて選択・切り替えて使用できる。

3. FOT ツール

産業技術総合研究所（以後、産総研）は、FOT を支援するツール（FOT ツール）[7] を開発している。FOT ツールは、開発者のテストモデル作成から、テストモデルの変換、テストケースの自動生成までを一貫して支援する。図 4 にそのツールの GUI 概観を示す。FOT ツールの GUI は、主に左右の 2 つのパネルにから成る。左のパネルは、開発者が拡張ロジックツリーを用いてテストモデルを作成する場所である。ツールは、開発者が文法的に正しい拡張ロジックツリーをテストモデルとして入力するのを、対話方式で支援する。また、FOT ツールは入力されたテストモデルを受け取り、テストケースを自動で行ない、その結果を右側のパネルに表示する。

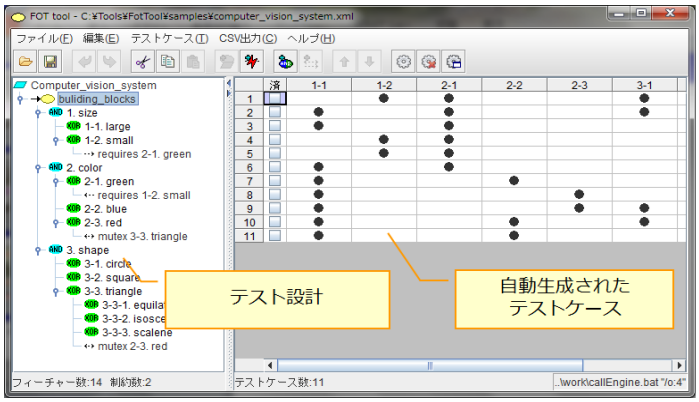

図 4

ツールが拡張ロジックツリーとして表現されるテストモデルからテストケースを自動で行うために、指定したテストケース生成ツール（PICT や ACTS など）の入力形式へのテストモデルの自動変換、テストケース生成ツールを介して Pair-wise 網羅基準のテストスイーツの生成、生成されたテストスイーツの右側のパネルに表示という一連の作業をツールの内部で行なっている。

3. Web アプリケーションへの適用例

2、3 章で、FOT の詳細を解説した。本章では、FOT の有効性を検証するために行った Web アプリケーションシステムへの適用事例結果について報告する。

4.1 対象としたシステムと範囲

適用対象は、既に開発が終了、リリースも完了している中規模の BtoC を目的とした Web アプリケーションのサブシステムである。対象のサブシステムは、チケット予約購入システムで、このサブシステムの要求概要は以下の通りである。

事前条件 1　予約対象となるイベント情報は、イベント管理サブシステムで登録、管理されている。

事前条件 2　チケット予約購入システムにログインするための顧客管理情報は顧客管理システムで登録、管理されている。

シナリオ概要　　チケット予約の大まかな流れは、以下の通りである。

1. 予約を希望するユーザは、チケット予約購入システムのログイン画面からログインする。ログイン可能なユーザは、顧客管理システムを通して既にユーザ登録が完了していること。
2. 予約を希望するユーザは、イベント検索画面で、イベント種類、イベント名、開催場所、開催日時を指定して希望するイベントを検索する。
3. 検索は、イベント種類、イベント名、開催場所、開催日時を適宜変えながら何回でも検索可能。
4. 希望するイベントが見つかり、かつ、空きがあれば予約をする。

本システムは、システム全体を UML で分析を行い設計された。本システムの作成時、特に重視した課題は、上流での品質保証である。これを実現するために、ユースケーステストによるシステムテストを実施した。

ユースケーステストは､ユースケースモデルで記述されている基本機能を網羅的にテスト可能なことから、UML による設計が広く利用されている現在、多くの開発現場で使用されている。ユースケーステストの手順は概ね、以下の通りである。

1. 機能を洗い出し、ユースケース図を作成
2. 各ユースケース図に対して、操作と属性の選択肢を分析、シナリオを作成
3. シナリオ作成で抽出した操作、属性からテスト条件を表の形式で作成
4. テスト条件の合否を決める
5. テスト実施手順を作成して、テスト計画書を作成する

システムの開発時に、上記の手順に従って、上流設計の段階でシステムテスト計画書が作成された（イベント管理サブシステムを含むすべてのシステムが対象)。

3.2　4.2 有効性の確認方法

(1)　開発時の問題点

本システムは、上流での品質保証を実現するため、ユースケーステストを実施したが、下記のような問題点が発生したため、十分な成果が上がらなかった。

- シナリオ作成時、操作と属性の選択肢は上げられたものの、これらの組合せをマニュアルで計算したため、適切な組合せのシナリオを作成できなかった。
- ユースケーステストのテストケースを作成する際、各シナリオ別に作成したため、重複が多く、無駄なテスト作業が多く発生した。
- 画面単体のテストは、画面上の入力項目、表示項目が多く、多数の属性を考慮する必要があるため、マニュアルでのテストケースを作成する際、抜け、漏れ、重複が発生し、効果的なテストが実施できなかった。特に、入力項目に設定されている制約条件、例えば、項目 A、B、C があった場合、A が入力された場合、B は不要で、C は必須。あるいは、B が入力された場合には、A、C は不要など、同じ画面でも、入力される項目に複雑な制約条件が設定されているケースが多かった。これらの組合せをマニュアルで行ったため、テストケースの作成が非常に複雑になり間違いが多発した。

(2)　確認内容

今回の実システムへの FOT 適用の目的は、以下の点を検証することを目的とした。

- FOT で生成されるテストケースは、過不足が無いか。特に、FOT が持つ制約条件の表現能力は十分かどうかの確認にも重点をおいた
- 開発時に発生した問題点は解決可能か。即ち、

- ✓ 必要十分なシナリオの組合せを求めることは可能か。
- ✓ シナリオ個別に作成していたテストケースを、シナリオ全体で使用可能なテストケースとして生成可能か。
- ✓ FOT を使用して生成された画面単体のテストケースは、開発時に作成されたテストケースの抜け、漏れ、重複をカバー出来たか。

以上の観点で、FOT でテストケースを自動生成して結果を評価した。

(3) 実施内容

有効性の評価は、４．１で述べた通り、本システムは既に開発が完了しているので、FOT で作成されたテストケースを実際に実行して結果を評価した。シナリオの妥当な作成必要数は、FOT で作成対象となるべきシナリオの組合せを生成し、レビューすることで評価した。

テストケース作成の際には､上流工程で実施されていたドメイン分析の成果であるユースケースモデル（実際の開発でもユースケーステストで使用された）が、FOT を使用したテストモデル設計に非常に有益であった。

4.3 結果

(1) 妥当なシナリオ数の検証

チケット予約購入システムのシナリオ数は､4.1 で示したシナリオ概要にある通り､イベント名、開催日、開催場所をそれぞれ変更しながら検索を進めるシナリオを作成する必要があり、この部分だけで、２７通りのシナリオを作成する必要がある。これ以外の属性として、チケットを購入するかしないか、購入したいチケットがあるかないか、などを考慮する必要があり、その組合せは、３４通りに達する。この組合せをマニュアルで計算し、シナリオを作成した結果 4.2.1 で示した問題が発生した。

FOT での適用は、シナリオ作成の操作、属性は開発時のものを使用し、これを図 5 に示すロジックツリーに整理した。

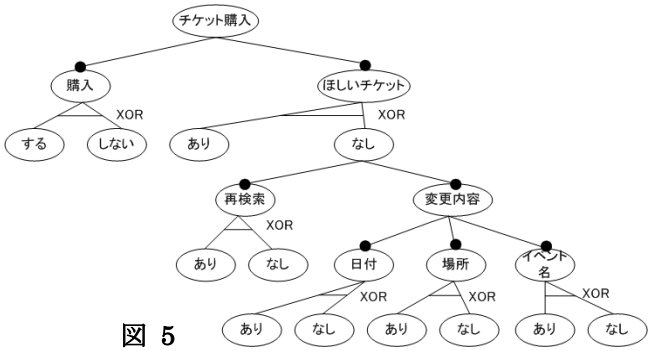

図 5

この内容を､FOT に入力し､Pair-wise 法適用し生成。結果、８個のシナリオで十分であることが､FOT 生成結果のレビューで確認できた。

(2) 生成されたテストケースの過不

過不足を検証する際には、既存の設計書、テスト計画書を参考にした。加えて、開発時のメンバーから、テストケース作成時に考慮した制約条件を聴取し、FOT のロジックツリーに再度整理し直し、FOT に入力した。その結果

1. 本システムのテストケースを作成する際に必要となる制約条件は FOT で全て記述可能であった。特に、画面に設定されていた複雑な制約条件は、2.1 で述べた、木横断的な制約条件を活用することで、テストモデル作成が容易に進められただけでなく、抜け、漏れの検証にも有益であった。　また、入力データに関しては、同値分解を使用したテストケースの作成が必須だが、FOT を使用することで、同値分割の分割基準を統一することができた。
2. 全体の組合せは数十万通りに達したが、Pair-wise 法を適用することで、この組合せは 80%程度まで削減可能で、かつ、開発時のテストケースより、約９０%程度まで削減可能であった。
3. 生成されたテストケースの質についても、FOT で生成されたテストケースを再度実行した結果、初期テストの不具合を全て検出することができた。新たな不具合も検出された。

さらに副次的な効果として、FOT の入力データを使用してレビューを進めることにより、テ

ストケース作成の意図が理解しやすくなり、レビューの質も向上することができた。

4. 考察と今後の課題

この手法の特徴を整理すれば以下の通りとなる。

- テスト対象システム（の入力領域）を木構造の頂点に設定することで、テスト設計時の観点が明確になる。ロジックツリー構造を用いてトップダウンに段階的にテストモデルを詳細化することで、テスト設計の考慮漏れを低減できる。ロジックツリーは物事を論理的に分析・検討するときに、その論理展開を樹形図に表現して考えていく思考技法であるが、その概念・事象間の論理的なつながりをツリー状に図示することで、相互関係が明確に把握できるなり、網羅・重複がない設計を支援する。FOT ではこのロジックツリーの性質を応用して、漏れ抜けのないテストモデルの作成を支援する。
- ロジックツリー全体をテストケース設計の結果として残すことで、設計の意図や意味を確認しやすくすることが出来る。即ち、レビュー等の作業が容易になる。
- 複雑な制約条件を含んだテストケース設計結果に対しても、最適な組合せテストケースの生成が可能。
- 導出されたテストケースは、ツリーで指定されている制約を満足した、論理的に整合性をたもった値である。

5.1 制約条件の抽出

制約条件は、テストケースの削減には必須である。しかし、自然言語で記述された仕様書等からこれらを抽出することは、非常に難しく、また、ドメイン固有の知識（暗黙知の類）が要求される場合も多い。今後の課題は、この作業をエンジニアリングレベルに落すこと、かつ、漏れ、間違い、矛盾を FOT の中で、探し出す手法を確立したい。

6. 参考文献

[1] D. Richard Kuhn, R. Dolores, Albert M. Gallo "Software Fault Interactions and Implications for Software Testing", IEEE Transaction On Software Engineering, Volume. 30, NO. 6, 2004

[2] D. M. Cohen, S. R. Dalal, M. L. Fredman, and G. C. Patton, The AETG System: An Approach to Testing Based on Combinatorial Design, Software Engineering, IEEE Transactions on Software Engineering, Volume 23 Issue 7, Page 437-444, July 1997

[3] M. Borazjany, L. Yu, Y. Lei, R. Kacker, R. Kuhn: Combinatorial Testing of ACTS: A Case Study. In Proc. of ICST 2012, pp. 591-600, 2012

[4] Available from http://csrc.nist.gov/groups/SNS/acts/

[5] Available from http://www-ise4.ist.osaka-u.ac.jp/ t-tutiya/CIT/

[6] J. Czerwonka,“ Pairwise testing in real world. practical extensions to test case generators, "in Proc.of PNSQC 2006, pp. 419 - 430, 2006

[7] T. Kitamura, N. T. B. Do, H. Ohsaki, F. Ling, S. Yatabe "Test-Case Design by Feature Trees", inProc. of ISoLA 2012, LNCS 7609, page. 458-473, Springer, October 2012

[8] Toru Nanba, Tatsuhiro Tsuchiya, Tohru Kikuno: Using Satisfiability Solving for Pairwise Testing in the Presence of Constraints. IEICE Transactions 95-A(9): 1501-1505 (2012)

[9] Toru Nanba, Tatsuhiro Tsuchiya, Tohru Kikuno: Constructing Test Sets for Pairwise Testing: A SAT-Based Approach. in Proc. of ICNC 2011: 271-274