B-006

アクティビティグラフからのテストケース生成技術

Test Generation based on the Activity Graph

片山 朝子　上原 忠弘　大橋 恭子　山本里枝子
Asako Katayama　Tadahiro Uehara　Kyoko Ohashi　Rieko Yamamoto

(株) 富士通研究所　FUJITSU LABORATORIES LTD.

1. まえがき

ビジネスアプリケーション開発において、品質保証のため顧客からのテストの品質強化要求が高まっている。しかし今までテスト品質はテストケースを作成する開発者の経験、スキルの差異によるばらつきが多く見られた。そこで本稿では設計フェーズで作成される UML アクティビティダイアグラムからテストケースの自動生成を行う手法を提案する。これによって SE が経験的に取得するテスト観点を取り込んだ一定品質のテストを作成できるようになるだけでなく、テストフェーズに関わる作業の効率化を実現することができる。

本稿では近年増加している Web アプリケーションを特に対象として取り上げる。

2. 対象テストと本研究のアプローチ

Web アプリケーションのテストにおいて単体テストについての研究は多くなされており、画面単体テストツールの Abbot や Cactus を使ったサーブレットの単体テストツールなどが存在する。しかし業務上のユースケースについて Web フロントからバックエンドまでを対象とする統合(IT)テストはカバー範囲が広く設計ドキュメントも多岐に渡り、テストケースを作成が難しいため本研究では IT を対象とする。

Webアプリケーション開発の最近の動向として設計モデルを元にコード生成し、実行、デバッグおよびテストを行いながらアプリケーションを完成させていくMDA技術による開発方法がある。本研究ではMDA技術を利用するアプローチを取ります。Webアプリ 開発を行うMDAツール[1]上で作成する２つの設計モデル、Webの画面遷移モデルのUMLアクティビティ図(画面遷移図)、クラスモデルのUMLクラス図(画面項目定義)を利用し、設計モデルからテストモデルへの変換を行い、バックエンドからのデータのやりとりも考慮に入れたユースケースに渡るITテストケース生成を考える。

3. テストケース生成の概要

本テストケース生成の元になる画面遷移図は１ユースケースにおける画面のとりうる遷移がアクティビティグラフで表されている。画面はノードとして表され、各画面で使われるデータはオブジェクトノード状態で表現され画面と関連を持ち、オブジェクトノード状態は画面項目クラスに存在するクラスへの関連を持っている。

テストケース作成手順を以下に示す。

(1) 遷移モデルを解析しテストパス部品を抽出する

(2) テストパス部品を元にテストパスを生成する

(3) 画面の入出力クラスを分析し、テストパスを分類しながらテスト観点を適用する

次にこのテストケース生成の概要図を示す。

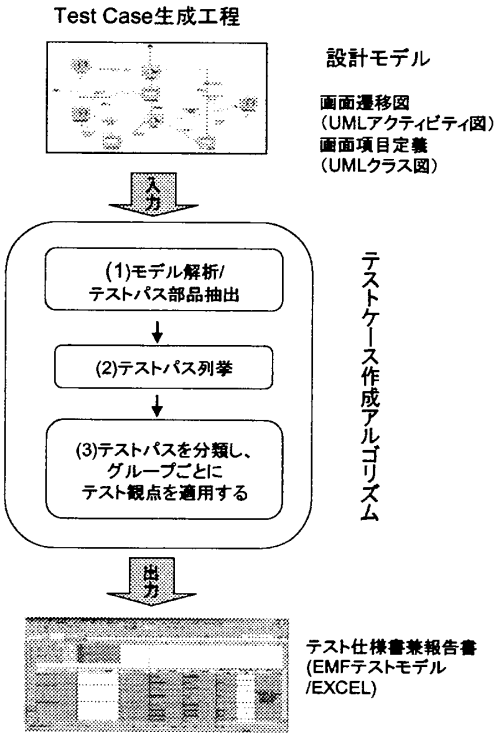

図１テスト生成工程

4. テストケース作成要領

4.1 モデル解析/テストパス部品抽出

設計ドキュメントである画面遷移モデルを解析し、TestCaseFragment と呼ばれる単位で抽出する。TestCaseFragment[i,j]は遷移元画面または初期状態 N[i]、遷移先画面または最終状態 N[j]、N[i]から N[j]への遷移を表す T[i,j],の３つ組からなり、N[i]画面からユーザの操作で次の画面 N[j]がブラウザ上に出力されるまでの１画面遷移を表す。

4.2 テストパス列挙

TestCaseFragment を組み合わせてテストパスを生成していく。テストパスは TestCaseFragment の列から構成され、TestCaseFragment[i,j]TestCaseFragment[j,l]...TestCaseFragment [m,f]であって、N[i]は初期状態、N[f]は最終状態。ただしループによるパス数発散を考慮し、2 回以上のループは含めない。

[1] Interstage Apmodeler

次に遷移図例とそこから抽出される TestCaseFragment とテストパスの例を示す。

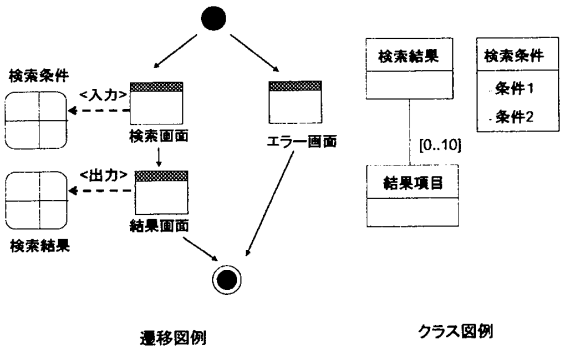

図２モデル例

TestCaseFragment:
①[初期状態] -> [検索画面]
② [検索画面]->結果画面]
③[結果画面] ->[最終状態]
④[初期状態] -> [エラー画面],
⑤[エラー画面] -> [最終状態]
テストパス:
1.[初期状態] -> [検索画面] -> [結果画面] -> [最終状態]
2.[初期状態] -> [エラー画面] -> [最終状態]

図３ モデル(図２)からの生成例

4.3 テスト観点と適用方法

4.2 で挙げたテストパスは単に遷移しうるパスを挙げたものにすぎない。実際の現場における IT テストでは、画面で使われるデータ構造を考慮に入れたテストバリエーションが重要である。例えば次のような観点である。

(a) 入力データが存在する画面での正常/異常入力

(b) 明細表の出力行数が０行から N 行まで取りうる場合に０件と N 件の表示させる

本アルゴリズムでは上記２つのテスト観点をテストパスに適用しテストケースを生成する。

しかし網羅的にするとテストケース数は非現実的かつ、人間ならばやる必要がないと判断する冗長なテストケースを多数生成する。本アルゴリズムは実際に業務アプリ開発で使われることを考慮し、テスト実行数として現実的かつ Web アプリとして有効なテストケースを挙げるためにテストパスを分析する。例えばある画面の異常入力テストについて、遷移が違うからといって毎回同じ画面で行う必然はない。そこで同じ遷移区間(TestCaseFragment)に対してテスト観点の重複適用を防ぐためにテストパスの短い順に次のようにパスを分類する。

(I) テストパスの集合で最初に出現する TestCaseFragment を含むテストパス

(II) (I)以外のテストパス

上記のグループに基き、テストパスの短い順に画面のデータ構造の分析をクラス図と遷移図を用いてテスト観点の適用を次のように行う。

まず(I)のグループのテストパスをたどり、テストパス中で最初に現れる TestCaseFragment の各画面に対しその画面で使われるデータを探す。入力である場合は(a)を適用する。出力であればそのデータのクラス構造を見る。クラス構成の関連に [*]や[0..20]などの多重度を表すものがあれば(b)を適用する。図２のモデル例の検索画面には入力データがあり(a)を適用する。結果画面には検索結果クラスの出力データがあり、その構造には[0..20]という多重度があるので(b)を適用する。(II)のテストパスについては TestCaseFragment が以前適用したものと重複するので観点は適用せず、テストパスをテストケースとしてそのまま挙げておき、観点を適用するかどうかはユーザの判断に委ねることにする。

4.4 生成されたテストケース

観点をテストパスに適用して作成されたテストケースは EMF によるテストモデルに変換、保存される。図３の 1.のテストパスからは次の３つのテストケースが生成される。

[初期状態] -> [検索画面] (観点: 正常/異常値入力)-> [結果画面] -> [最終状態]
[初期状態] -> [検索画面] -> [結果画面](観点: 0行表示) -> [最終状態]
[初期状態] -> [検索画面] -> [結果画面](観点: 20件表示) -> [最終状態]

図４ テストパス例１に観点を適用したテストケース

２のテストパスについては観点がないので観点のないテストパスをテストケースとしてあげる。

またテストモデルから、適当なフォーマットを用いて Excel のテスト仕様書として出力することができる。

5. 評価

製品開発で実施されたテストとの比較実験を行った。設計ドキュメントから人手で作成したテストケースと本アルゴリズムが作成したテストケースを比較し、両者のテスト遷移パス本数およびテスト観点もほぼ同程度であった。しかし業務の意味を考慮したテストパスになっていないため人手による順序変更が必要であった。

6.まとめ

本アルゴリズムにより設計モデルを元にデータを考慮したテスト観点を取り込み一定品質の Web アプリ IT テスト作成を実現した。

Struts 等 Web フレームワークの設定ファイルから画面遷移を作成しテストパスを作成する研究が[1]でされているが、ここでは画面データ構造やパス数削減については言及されていない。またここでは静的モデルからテストパスを生成したが、動的リソースからモデルを作成しテストパスを作成している研究は[2]でなされている。しかしここでも画面データ構造への言及はされていない。

今後はループ観点を含むテスト観点の更なる充実および業務上有効なテストパスの生成を目標にしていく予定である。

7.参考文献

[1] S.Yuen, et al. A Testing Framework for Web Applications based on the MVC model with Behavioral Descriptions, ICITA 2004,pp238-243

[2] Filippo Ricca, Paolo Tonella, Analysis and Testing of Web Applications, ICSE 2001, pp25-34