B-002

Web アプリケーションを対象とした高網羅率の単体テスト自動生成について
Automatic generation of high coverage unit test for Web-applications

鎌田 高如† 樋口 昌宏†
Takayuki Kamada Masahiro Higuchi

1. はじめに

近年では，業務処理システムの多くが Web アプリケーションとして開発されている．また，ソフトウエアの品質を保証する上で，単体テストの重要性が増してきている．単体テストのテストスイートに求められる要件として，試験対象の機能を確認すると共に再利用性，保守性の観点からプログラムコードの網羅率の高さが挙げられる．V 字モデルでは，詳細設計より導出した単体テスト仕様書からテストメソッドを抽出し，それらをまとめたテストスイートを作成する．更に網羅率を高めるためにプログラムコードを分析し，手作業によりテストメソッドを追加する．そのような手作業が煩雑であるため，単体テスト自体が軽視されがちなのが現状である．このため，単体テストのテストスイート生成の自動化に関する研究が行われている[1][2]．本研究では，Web アプリケーションにおける，データベースアクセス部分に特化した高網羅率の単体テスト自動化について検討する．

2. Web アプリケーションの単体テスト

2.1 web アプリケーションについて

図 1 に典型的な Web アプリケーションの構成図を示す．これは，MVC(Model，View，Controller)モデルと呼ばれる．まず，クライアントからの http Request を Controller 層が受け取る．Controller 層は，http Request に基づいて Model 層に処理内容を伝える．Model 層は，必要に応じてデータベースアクセスを行い，処理終了を Controller 層に伝える．その後，Controller 層は View 層を呼び出す．View 層は，Model 層の処理結果を参照し，クライアントに http Respons を返す．

2.2 データベースアクセス部分について

図 2 に Model 層の構成図を示す．Model 層は複数の Service クラスと DAO クラス，VO クラスから構成される．Service クラスは，Model 層の主体となるクラスで，DAO クラスとやり取りをしながら処理を進める．DAO クラスは，データベースに接続し，データベースへの追加，変更，参照，削除を行う． データベースアクセスは，データベースのテーブルの属性に対応した Getter/Setter を持つ VO クラスを DAO クラスが利用し行う．

2.3 単体テストと網羅率

単体テストは，ソフトウエアの部品単位の，個々の動作を確認するテストである．機能確認テストに加え，網羅率が重要となる．高網羅率を実現することが，再利用性，保守性の向上に繋がる．テストで実行した文の割合を文網羅率，実行した分岐の割合を分岐網羅率，条件判定に用いられる個々の論理式における真と偽の出現の割合を条件網羅率と言う．

†近畿大学 KinkiUniversity

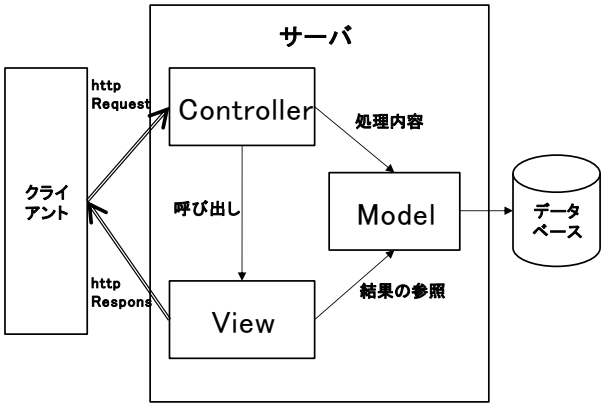

図 1 Web アプリケーション構成図

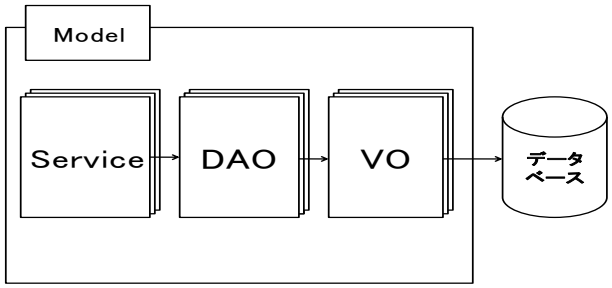

図 2 Model 層構成図

2.4 テストファースト

従来の開発手法では，実装後にテストスイートを作成し，単体テストを行っている．これに対し，テストファースト開発では，詳細設計後，実装前にテストスイートを作成し，テストを実行しながら実装を進める．単体テストに合格するまで繰り返しながら実装の見直しを行っていく開発手法である．

2.5 dbUnit

本研究は，プログラミング言語は Java，データベースは MySQL，統合開発環境の eclipse に dbUnit のような開発ツールをプラグインして使用することを前提とする．

データベースアクセスのテストで，テスト結果の確認を目視で行うには，データ項目数やテーブル数が多数になると限界があり，確認ミスの可能性が高くなる．また，テストを複数人で行う場合，テストデータを更新してしまう可能性があり，更新してしまうと正しいテストができない．通常 dbUnit のようなテスト支援ツールを利用する．dbUnit は，データベースの各テーブルに初期値として用意する属性値を.xml ファイル形式によりプログラム内で設定できる．テスト実行結果の確認は，期待する出力値

を.xml ファイル形式で準備し，テスト実行後のテーブルの属性値と比較する[3].

3. 単体テスト自動生成システム

本システムは，テスト対象クラスの機能を確認するためのテストスイートが与えられた時，テストメソッドを自動的に追加することによって，高網羅率のテストスイートに拡張するものである．システムに対する入力は，テスト対象メソッド，機能確認のためのテストスイート，.xml ファイルである．

3.1 システムの概要

図 3 にシステムの構成図を示す．テスト対象メソッドとテストスイートをインタプリタでの実行により，実行レポートを出力し，網羅率を求める．実行レポートより未網羅パスを抽出し，未網羅パスと.xml ファイルをテストメソッド自動生成器に通し，新たなテストメソッドをテストスイートに追加する．以上のサイクルを，十分な網羅率が達成されるまで繰り返す.

3.2 システム構成

3.2.1 Java ソース構文解析器とメソッドグラフ

テスト対象メソッドを構文解析し，メソッドグラフを作成する．メソッドグラフとは，1 つのテスト対象メソッドを，式単位に分割し，式の実行順序を表したものである．

3.2.2 テストメソッド構文解析器とオブジェクト状態定義表

テストスイートをこの構文解析器に通すと，オブジェクト状態定義表が作成される．オブジェクト状態定義表とは，テスト対象メソッドを実行させるため，必要なフィールド変数とローカル変数の変数情報（型，変数名，値）を定義したものである．

3.2.3 インタプリタと実行レポート

メソッドグラフとオブジェクト状態定義表を使用し，テストを実行させ，実行レポートを出力する．実行レポートとは，単体テストで実行されたプログラムコードと変数情報について記録したものである．

3.2.4 未網羅部分抽出器と未網羅パス

実行レポートに記録されている実行パスとメソッドグラフの式の実行順序を基に，未網羅部分抽出器により単体テストにおいて，実行しなかった未網羅部分を通すための未網羅パスを抽出する．

3.2.5 テストメソッド自動生成器

動作確認されていないプログラムコードを通すための未網羅パス，実行レポートに記録されている変数情報とテスト実行後の期待する出力値の.xml ファイルから dbUnit を利用できるテンプレートを考え，テストメソッドを自動生成する．

4. 実験

酒屋問題を例題として，参考文献[4]の手順に従って Web アプリケーションを開発し，実験を行った．

4.1 システム開発状況

図 3 において，現状ではテストメソッド自動生成器は，未実装であるため，今回の実験では，手作業でテストメソッドを追加した．

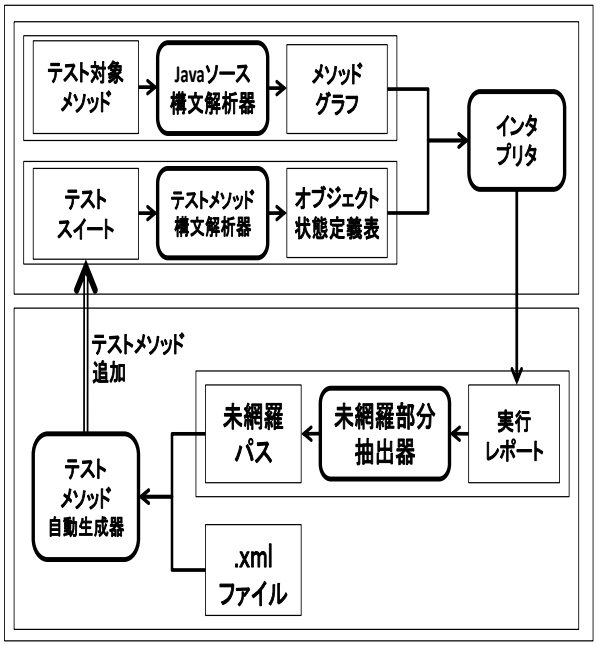

図 3 システム構成図

4.2 実験結果

今回作成した Web アプリケーションは，Control 層の Servlet 数は 6，View 層の jsp 数は 9，Model 層の Service クラス数が 5，DAO クラス数が 6，VO クラス数が 6，である．Service クラスは平均 50 行，DAO クラスは平均 200 行程度のプログラムである．

上記の中の，DAO クラスのメソッド 1 つに対して，テストメソッドを自動生成しテストスイートに追加する実験を行った．テスト対象メソッドは 38 行であり，初期的なテストスイートとして詳細設計より導出した単体テスト仕様書から手作業で作成した 3 つのテストメソッドからなるものを用いて単体テストを行った．このテストスイートの分岐網羅率は 75%であった．図 3 の構成に従い，2 つのテストメソッドを追加することにより，分岐網羅率は 91.6%に向上した．以上の結果より，本研究で提案するシステムの有用性が確認できた．

5. 今後の課題

テストメソッド自動生成器を開発し，酒屋問題の Web アプリケーション全てのクラスに関する実験を行うことである．

参考文献

[1]Patrice Godefroid, et al., “Automating Software Testing Using Program Analysis”, IEEE Software, Vol.25, No.5, pp.30-37 (2008).
[2]式見遼，小形真平，松浦佐江子，“UML 要求仕様からのカバレッジに基づく機能テストのテストケース生成”，情報処理学会第 73 回全国大会，(2011).
[3]サイバービーンズ(株)，“JUnit によるテストファースト開発入門”，(2004).
[4](株)NTT データソフトウェア工学促進センタ，“実例で学ぶソフトウェア開発”，(2008).