2008－EVA－24 (7)
2008／3／19

拡張ユースケースシナリオからテストプログラムの自動生成を支援する開発環境の提案と評価について

古田 裕久、石原 鑑、安井 照昌、石井 俊直
三菱電機株式会社

テストプログラムを用いて対象プログラムを自動テストするツールが普及しつつあるが、テストプログラムの開発に手間がかかっている。何故なら、開発者は文書で記述されたシステムの仕様を分析し、その情報を基にプログラムならびにテストプログラムを開発しているからである。また、開発したテストプログラム自身がシステムの仕様に則しておらず、テストプログラムの再開発といった手間が発生している。そこで、本研究では、システムの仕様を拡張ユースケースシナリオで設計し、拡張ユースケースシナリオからテストプログラムを自動生成する開発環境を提案し、開発したプログラムの信頼性の評価を行う。

Automation of a Test Program by using the Extended Usecase Scenario

Hirohisa Furuta, Akira Ishihara, Terumasa Yasui, Toshinao Ishii
Mitsubishi Electric Corporation

There are many unit test tools that support running test programs. However, there aren't any effective tools that support developing test programs which covers all system requirements specifications. In this paper, we propose a test development environment which automates test programs by analyzing system requirements specifications in format of the Extended Usecase Scenario. We will explain model of the Extended Usecase Scenario, and show how it converts test programs.

1. はじめに

ＩＴ技術革新に伴い、今までは各業務単独でシステム化を行っていたが、それらの業務システムを統合した統合業務システムの開発が進んでいる。多くの統合業務システム開発のプロジェクトでは、システム要件仕様が確定できないまま、システム仕様の設計、システム開発、システム試験が進められ、その結果、システム要件仕様不足による手戻りが発生するなど、工期、予算、品質を満足しない事例が増加している。この背景には、多くのプロジェクトで、システム要件仕様が頻繁に変更されているにもかかわらず、後戻りが許されない従来型のウォータフォール開発プロセスで開発が進められていることが挙げられる。

このような課題に対して、システム要件仕様の変更は当然あるものとし、大まかな仕様でテストプログラムと対象プログラムを実装し、システム要件仕様の妥当性を検証するテスト駆動開発（Test Driven Development）プロセスを実践するプロジェクトが増加している[1]。しかし、テスト駆動開発プロセスは、幾つかの成功事例のエッセンスを抽出し、纏めた方法であり、実際のプロジェクトに組み込み実践するためには、プロジェクト独自にテスト駆動開発プロセスを拡張していく必要がある。特に、プログラム開発者が開発したテストプログラムが、システム要件仕様を十分に満たしているかを検証する手段がない。

この課題を解決するために、本研究では、システム要件仕様からのテストプログラムの自動生成を支援する開発環境を提案する。本研究では、アリスター コーバーンによって提唱されたユースケースシナリオを拡張した拡張ユースケースシナリオを用いて、システム要件仕様の記述ならびに、テストデータの設計を行う。本開発環境は、これらの情報を用いて、テストプログラムを自動生成する。

また、統合業務システムの例題として、電力、自治体が保有している設備の維持管理業務を支援する統合業務システムを取り上げる。そのシステム開発プロジェクトにおいて、本研究の開発環境を用いてテスト駆動開発プロセスを実践し、開発した対象プログラムの信頼性の評価を行う。

以下２章では、テスト駆動開発プロセスを説明し、３章では、拡張ユースケースシナリオならびにそれを用いた開発環境を説明し、４章では、そして開発された対象プログラムの信頼性の評価方法ならびに、維持管理業務を支援する統合業務システムへの適用実験ならびにその評価を行い、５章で結論と今後の課題を述べる。

2. テスト駆動開発プロセスの課題

本章では、ウォータフォール開発プロセスやテスト駆動開発プロセスの特徴やその課題について纏める。

2.1. ウォータフォール開発プロセス

図 1 は、ウォータフォール開発プロセスの概要を示した図である。ウォータフォール開発プロセスは、左上の設計工程（要求分析、システム要件定義、システム要件仕様、基本設計、詳細設計）から始まり、開発工程（実装）、そしてテスト工程（単体テスト、結合テスト、総合テスト、受け入れテスト）という工程順に実施していく。各テスト工程におけるテストの設計仕様は、各設計工程の設計仕様から設計され、その結果を用いて、開発された対象プログラムが仕様どおり動作するかを評価する。

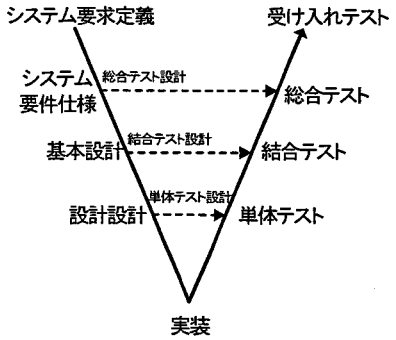

図 1：ウォータフォール開発プロセスの概要

ウォータフォール開発プロセスは、図 1 の設計工程で設計した仕様の変更が起きないことを前提としている。つまり、ある程度工程が進んだ段階で仕様の変更が発生した場合、仕様を変更し、テスト設計を見直し、テストプログラムならびに対象プログラムを変更し、動作評価するという行為を、変更が発生するたびに実施しなくてはならない。

2.2. テスト駆動開発プロセス

テスト駆動開発プロセスは、システム要件仕様などの各設計工程で設計する仕様は、変更されるということを前提としている。テスト駆動開発プロセスには、テストプログラムを予め開発し、蓄積することで、回帰テストやリファクタリングが容易に出来るという利点もある反面、幾つかの課題もある。図 2 は、テスト駆動開発プロセスにおける課題を整理した図である。

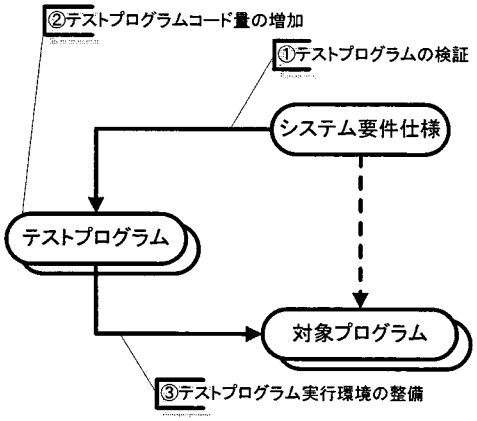

図 2：テスト駆動開発プロセスの課題

(1)テストプログラムの検証：プログラム開発者は、システム要件仕様を分析し、その分析結果に基づきテストプログラムを開発していく。しかし、そのテストプログラムがシステム要件仕様を十分に満たしているか検証する手段が無い。

(2) テストプログラムコード量の増加：効果的なテストをするためには、テストプログラムは対象プログラムのコード量と同程度またはそれ以上になる。つまり、開発者にとっては、テストプログラムの開発は大きな負担となる。

(3)テストプログラム実行環境の整備：テスト駆動開発プロセスを進めていくに当たっては、開発したテストプログラムを実行できる環境を整備する必要がある。

このような課題がある背景には、テスト駆動開発プロセスは、幾つかの成功事例のエッセンスを抽出し、纏めた方法であり、実際のプロジェクトに組み込み実践するためには、プロジェクト独自の拡張が必要であるという事実がある。

3. テストプログラムの生成を支援する開発環境の提案

本研究では、テスト駆動開発プロセスが抱えている３つの課題を解決するため、システム要件仕様からテストプログラムの自動生成を支援する開発環境を提案する。

本研究では、対象とするテストプログラムをWeb アプリケーションのテストを支援することとする。図３は、テストプログラムの自動生成を支援する開発環境の概略図である。Eclipse 統合開発環境上に、システム要件仕様の仕様化の支援、テストプログラムへの変換、そしてテストプログラムの実行を支援する機能を提供している[3]。以下ではそれぞれについて説明する。

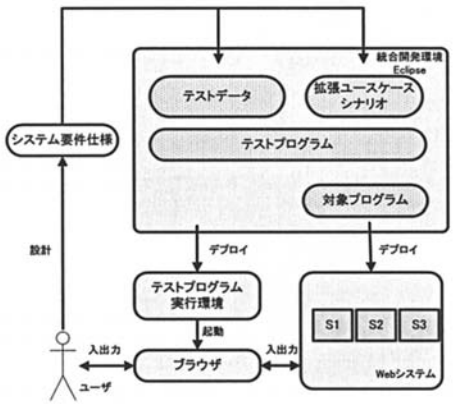

図 3：開発環境の概要図

3.1. 拡張ユースケースシナリオ

課題１と２を解決するために、システム要件仕様からテストプログラムを自動生成する。システム要件仕様の記述方法としてユースケースシナリオに着目する。

・ユースケースシナリオとは

ユースケースシナリオとは、システムがどのように振舞うのかを、その振舞いに関わるアクター（システムを利用するユーザ）を交えて文章で記述したものである[2]。

表 1 はユーケースシナリオの一例である。ユースケースシナリオは、

アクター：システムの振舞いを実行させるユーザ

事前条件：システムが振舞いを開始するために、事前に満たされていなくてはならない条件

事後条件：システムが振舞いを終了したときに成立している条件

基本系列：アクターを用いて詳細な振舞いを「○が、△を□する」というフォーマットの文章で記述する

代替系列：ある条件下での詳細な振舞いを文章で記述する

例外系列：例外となった場合の詳細な振舞いを文章で記述する

の６つの項目で記述する。

ユースケース名	設備分類を選択し、リストさせる
アクター	設備管理者
事前条件	アクターがシステムにログインしている
事後条件	選択した設備分類の設備がリストされている
基本系列１	アクターは、設備分類リストボックスから、設備分類を選択する。
基本系列２	システムは、選択された設備分類の設備を検索する。
基本系列３	システムは、選択された設備分類の設備の一覧を表示する
代替系列１	
例外系列１	

表 1：ユースケースシナリオ例

・ 拡張ユースケースシナリオとは

ユースケースシナリオからテストプログラムを自動生成するために、ユースケースシナリオとテストプログラムの関係に着目し、ユースケースシナリオの各系列に記述されたユーザとシステム又は、システムとシステム間で受け渡しされるデータを具体的な値で記述した文章であることを利用し、ユースケースシナリオに具体的なテストデータを記述したテストのシナリオを設計するという方針でユースケースシナリオを拡張する。それを拡張ユースケースシナリオと定義する。

図 4 は、ユースケースシナリオとテストプログラムとの関係を示したクラス図であり、点線で囲まれた範囲が拡張ユースケースシナリオである。拡張ユースケースシナリオは、UML（Unified Modeling Language）のユースケース、ユースケースシナリオを、テストシナリオとそのデータ

で記述することを特徴としている。表 2 は、その一例である。以下では、拡張ユースケースシナリオの詳細を説明する。

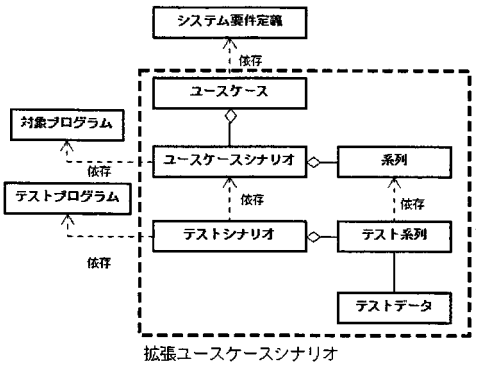

図 4：ユースケースシナリオと
テストプログラムとの関係を示したクラス図

拡張ユースケースシナリオでは、アクター、事前条件、そして系列の 3 つのユースケースシナリオ項目を用いて記述する。系列とは、ユースケースシナリオの基本系列、代替系列や例外系列を区別せず呼ぶ場合の総称である。

ユースケース名	設備分類を選択し、リストさせる
ユースケース ID	UC51-08
ACT	設備管理者
PRE	UC51-01
N	設備管理者は、設備分類リストボックスから、[設備分類]を選択する
N:LIST	システムは、選択された設備分類の設備の一覧を表示する

表 2：拡張ユースケースシナリオの一例

ACT（アクター）：システムの振舞いを実行させるユーザ

PRE（事前条件）：システムが振舞いを開始するために、事前に満たされていなくてはならない条件

N（系列）：アクターを用いて詳細な振舞いを記述する。ただし、アクターとシステム又はシステムとシステム間で受け渡しされるデータはすべて[<データ名>]という形式でパラメータ定義する。

・ テストデータとは

テストデータとは、拡張ユースケースシナリオで定義したパラメータに、どのような値が設定されているか記述されている。表 3 はその一例である。

PRE（事前条件）：システムが振舞いを開始するために、事前に満たされていなくてはならない条件

P（パラメータデータ）：パラメータ定義（<データ名>）とその具体的なデータ

R（事後条件）：システムが振舞いを終了したときに成立している条件

項目	ラベル	値
テストデータ名	設備分類を選択し、リストさせる	
テストデータ ID	TD51-08	
PRE	TD51-01	
P	設備分類	道路
P	属性	道路名
P	属性一覧	国道１号線 国道２号線
R:LIST	[属性]に[設備一覧]が表示されていること	

表 3：テストデータの一例

3.2. 開発環境の実装

システム要件仕様に対して、拡張ユースケースシナリオとテストデータを記述し、テストシナリオを生成することが可能となった。そこで、それらの設計仕様を用いてどのようにテストプログラムを自動生成するか説明する。

・ テスティングフレームワーク

テスト駆動開発プロセスを支援するため沢山のテスティングフレームワークが開発されている。Java 開発言語ミドルウェアの場合、JUnit、TestNG といった代表的なテスティングフレームワークから、JUnit を拡張し、Web システムのテスティングを支援する Selenium という用途に特化したテスティングフレームワークが開発されている[4]。

・ テストプログラムの自動生成

開発環境は、次の手順で Selenium テスティングフレームワーク向けのテストプログラムを、拡

張ユースケースシナリオならびにテストデータから自動生成する。その概要を示したのが図 5a である。

1. テストデータから、パラメータの値をテストプログラムが読み込むためのテストデータテンプレートの自動生成
2. 拡張ユースケースシナリオから Selenium が実行可能なテストプログラムテンプレートの自動生成（図 5b）
3. テストプログラムテンプレートに、テストデータの値を実装した、テストプログラムの自動生成（図 5c）

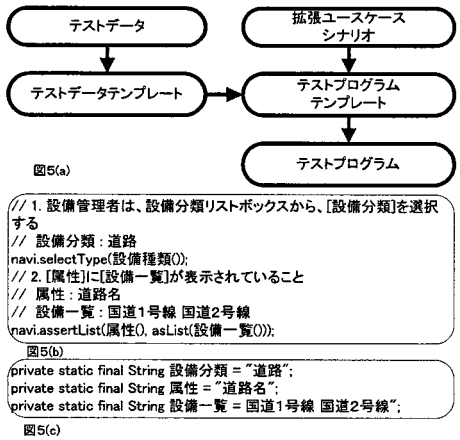

図 5：テストプログラム生成の概略図

これにより、システム要件仕様を反映させたテストプログラムを自動生成することが出来る。ただし、拡張ユースケースシナリオからテストプログラムテンプレートをすべて自動生成することができない。よって、テストプログラムテンプレート上記に記述されたコメントに従い、開発者がテストプログラムを拡張開発していく。

4. 統合業務システムへの適用実験と評価

本章では、本研究で提案した開発環境を、維持管理業務を支援する統合業務システム開発への適用ならびに開発された対象プログラムの信頼性を評価した。

4.1. 統合業務システムの概要

本節では、適用実験の対象とした設備維持管理を支援する統合業務システムについて述べる。

・ 設備維持管理統合業務システム

設備維持管理統合業務システムとは、複数の部門をまたがる設備に対する新設の計画、設計、工事、維持管理という一連のライフサイクルを、部門間で情報を共有させ、効率的に作業を遂行することを支援するシステムである。対象とする設備分野は、土木、建築、水道であり、情報を共有するデータは、台帳、文書（設計図面、設計書、写真など）、地図情報などがある。

本研究の実験では、設備維持管理統合業務システムの一機能であるサブシステム S1 に対して、本研究で提案した開発環境を適用し、開発された対象プログラムの信頼性を評価した。また、テスト駆動開発プロセスで開発プロセスを実践したサブシステム S2 と、従来型のウォータフォール開発プロセスを実践したサブシステム S3 に対しても同様の信頼性評価を実施し、開発環境を適用した場合と適用しなかった場合とでの信頼性を比較する。

4.2. 信頼性の評価方法について

本節では、適用実験の対象とした設備維持管理で開発された対象プログラムの信頼性の評価方法について述べる。

開発した対象プログラムの信頼性の評価方法としては、信頼性モデルの一つである非斉次ポアソン過程モデルを用いて発生した不具合や仕様変更の収束状況を監視し、収束した段階で開発した対象プログラムがシステム要件仕様を満たしていると判断する[5]。その数式が(1)である。

$$\mu(t) = V_0\{1-\exp(-\frac{\lambda_0}{V_0}t)\} \quad \ldots(1)$$

$\mu(t)$：ある時間における発見された累積不具合や仕様変更数

V：総累積不具合や仕様変更数

t：時刻

λ：パラメータ

4.3. 評価

・ 対象プログラムの信頼性評価

図 6 は、各サブシステムの不具合や仕様変更数の累積を図にしたグラフである。サブシステム S 1 は、開発の早期のフェーズから不具合や仕様変更が発生していることがわかる。しかし、不具合や仕様変更をテストプログラムに反映させながら開発した結果、計画した期間内に不具合や仕様変更を収束させることが出来たことがわかる。

その一方、サブシステムＳ２やＳ３は、システム要件仕様が確定しないまま開発が進めた結果、何回かのレビューで大幅な仕様変更等が発生し、その結果、開発期間内に不具合や仕様変更を収束させることが出来なかった。

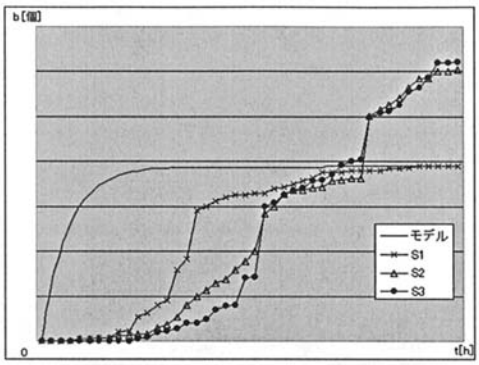

図 6：不具合や仕様変更数の累積

・ テストプログラムの評価

表 4 は、Ｓ１サブシステムに着目し、対象プログラム、自動生成されたテストプログラムテンプレート、テストデータテンプレート、テストプログラムならびに、拡張開発されたテストプログラムテンプレートの総ライン数とデータ数を纏めたものである。

ソース	ライン数 [KL]	データ数
対象プログラム※1	38.2	131
テストプログラムテンプレート（自動生成）	2.1	73
テストプログラムテンプレート（拡張開発）	0.8	73
テストデータテンプレート（自動生成）	0.3	84
テストプログラム(自動生成)	3.1	342

表 4：Ｓ１サブシステムのライン数

※1：Web ブラウザ上から処理されるプログラムのみを対象としている

表 4 が示すとおり、拡張開発したテストプログラムすべてテストプログラムテンプレートであることがわかる。つまり、不具合や仕様変更が発生した場合でも、開発者は変更されたプログラムテンプレートのみを拡張開発することで、不具合や仕様変更に対応するテストプログラムを開発することが出来る。

5. まとめ

本研究では、テスト駆動開発プロセスを実際のプロジェクトで実践するため、テストプログラムの検証、テストプログラムコード量の増加、テストプログラム実行環境の整備といった三つの課題を解決する、システム要件仕様からテストプログラムの自動生成を支援する開発環境を提案した。そして、その開発環境を用いて、テスト駆動開発プロセスを実践した結果、少ないテストプログラムコード開発量で効率的に統合業務システムのサブシステムＳ１の開発を行うことが出来た。

今後の課題としては、不具合や仕様変更に対応する為、拡張ユースケースシナリオの系列項目に対応するテストプログラムテンプレートに埋め込むコードの種類を増加させ、テストプログラムテンプレートの自動生成率を高めていく。また、システムに対する性能面などの非機能要件を拡張ユースケースシナリオに組み入れ、開発したプログラムの信頼性と性能を同時に向上させていく。

6. 参考文献

[1] ケント・ベック、テスト駆動開発入門. ピアソン・エデュケーション、 2003.

[2] アリスター・コーバーン、ユースケース実践ガイド. 翔泳社、2001.

[3] Eclipse. http://www.eclipse.org/

[4] OpenQA: Selenium. http://www.openqa.org/iselenium/

[5] 玉井　哲雄、ソフトウェア工学の基礎. 岩波書店、2004.