ソフトウェア開発演習におけるテスト駆動開発支援手法の提案

Proposal of a Support Method for Test-Driven Development in Software Development Practice

大橋 旭雄[*1], 神長 裕明[*2], 中村 勝一[*2], 森本 康彦[*1], 宮寺 庸造[*1]
Akio OHASHI[*1], Hiroaki KAMINAGA[*2], Shoichi NAKAMURA[*2], Yasuhiko MORIMOTO[*1], Youzou MIYADERA[*1]
[*1] 東京学芸大学 [*2] 福島大学
[*1]Tokyo Gakugei University [*2]Fukushima University
Email: m133302s@st.u-gakugei.ac.jp, miyadera@u-gakugei.ac.jp

あらまし：近年のソフトウェア開発の現場ではテスト駆動開発（TDD）の実施が推奨されている．TDD での開発には様々な利点があるため, ソフトウェア開発初心者の TDD スキルを養うことが有効と考えた．そこで本稿では，ソフトウェア開発初心者の TDD スキル育成を目的とし，TDD の開発サイクルやテスタブルコードの書き方といった TDD スキルを, ソフトウェア開発演習の中で育成する支援手法を開発する．
キーワード：テスト駆動開発（TDD），ソフトウェア開発，ソフトウェア開発演習，ソフトウェアテスト

1. はじめに

近年のソフトウェア開発の現場では，アジャイルソトウェア開発手法の採用が増加しており, その中ではテスト駆動開発[(1)]（TDD）の実施が推奨されている．TDD は開発者自身がプログラムとして動作するテストを作成しながら開発を進める手法である．TDD での開発には開発促進や品質向上等様々な利点があり, ソフトウェア開発を学び始めた初心者の内から TDD を実践し，そのスキルを身につけることが有効と考えた．しかし，初心者が独学で TDD スキルを身につけるのは困難である．

そこで本稿では，ソフトウェア開発初心者の TDD スキル育成を目的とし，ソフトウェア開発演習の中で TDD スキルを育成する支援手法を開発する．

2. 研究背景

2.1 テスト駆動開発の概要

TDD ではまず，作成するシステムを段階的に詳細化し，必要な機能・モジュールを列挙する．次に，それらが行うべき振る舞い（満たすべき仕様）を ToDo リストに記述する. そして記述した項目中で実装が容易そうなものから，以下の手順で開発する．

①：テストを作成し実行，失敗することを確認
②：目的のコード（プロダクトコード）を記述しテストを実行，成功することを確認
③：テスト成功を維持したままリファクタリング

以上 3 手順を 1 サイクルとし, これを短時間で繰り返すことで機能・モジュールを 1 つずつ実装し，1 つのシステムを組み上げる. テストはメソッドの引数と期待する戻り値の組み合わせによるテストデータと, 実際の戻り値と比較するテストコードによって作成され, テスティングフレームワーク上で実行される. テスト失敗後はできる限り小さな修正で成功させるよう努め, その後にリファクタリングを行う．実装完了後当該項目を ToDo リストから削除し，次の項目を実装する．テストすべき項目が新たに発生した際は，それもリストに追加する．

テストを実行するとプログラムの成否という形でフィードバックを得られるが, 開発者はこれを記述したプロダクトコードの正しさの判断材料, あるいは次に取るべき行動の指針として使うことができ, 開発促進の効果をもたらす．また，自動テストが可能（テスタブル）なテストコード及びプロダクトコード（ソースコード）を必要とするため, 開発者はテスタブルな設計を意識するようになる．さらに，自動テストの存在により開発中の予期せぬバグの作り込みが減少し，品質向上に繋がる．

2.2 ソフトウェア開発初心者への導入

TDD は現場の開発者のみならず，ソフトウェア開発初心者にも有効な手法と考えられる．そこで本稿では，大学におけるソフトウェア開発初心者に TDD を学ばせたいと考えた．しかし，初心者が TDD を行うには幾つかの困難がある．TDD を用いた開発プロセスは特殊であり，初心者が自力で遂行するのは困難と考えられる（問題点①）．また TDD に必要なテスタブルコードの作成には, 設計や実装のスキルが不可欠である（問題点②）．また，テストの作成は初心者にとっては難しく，かえって開発を停滞させかねない（問題点③）. さらに，大学の講義等で専門的に TDD を学ばせるような時間は取れない（問題点④）.

そこで本稿では, 以上の問題点を解決し，ソフトウェア開発初心者に TDD を実践させることで TDD スキルを育成することを目的とし，支援手法を開発する．

2.3 先行研究・関連研究

吉田ら[(2)]は，プログラミング学習に TDD を取り入れることでの様々なスキルの育成を目指し, 支援システムを開発した．しかし，プログラミング学習段階での支援であるため，ソフトウェア開発の現場に則した TDD スキルを育成するには十分ではない．上河内ら[(3)]は，テスト方法がわからない学習者に対し単体テストフェーズでの支援を行っている．TDD を行うにはテスト方法の理解が重要と述べており, 本稿でもその考えを踏襲する．

3. テスト駆動開発支援手法

3.1 支援手法の要件

ソフトウェア開発初心者の TDD スキル育成を目的と

する支援手法の開発にあたり，以下の要件を定める．

要件①：ソフトウェア開発プロセスの中で TDD スキルの育成が行えること．

要件②：テスタブルコード作成支援が行えること．

要件③：テスト作成の負担を減らすこと．

要件①は問題点①と④を，要件②と③はそれぞれ問題点②と③を解決する．以上 3 つの要件を満たすため，本稿では開発環境上で状況をリアルタイムに監視し，収集した履歴を分析して学習者に様々な支援を行う．

3.2 ソフトウェア開発演習への導入支援

TDD では開発サイクルをモジュール 1 つ 1 つに対して短時間で繰り返すことが重要であるが，この開発サイクルは一般的なソフトウェア開発プロセスとは大きく異なっており，初心者が初めから自身で行うのは難しい．そこで本稿では，初心者育成のために大学等で広く行われているソフトウェア開発演習の中で，TDD の開発サイクルを円滑に実施できるよう支援を行う．具体的には，ソフトウェア開発プロセスの中で今現在自身がどんな作業をしているか，TDD の開発サイクルのどの段階にいるのかといった開発状況を把握し，それに基づいてソフトウェア開発プロセスと TDD の開発サイクルに則った開発ができるよう誘導や提示を行う．これにより，要件①を達成する．

3.3 テスタブルコード作成支援

TDD を行うにはテスタブルコードが重要となるため，その作成支援が必要である．本稿では，あるソースコードとテストデータの組が与えられた時，テスタブルコードは以下の性質を持つと定義する．

・テスタブルなテストコード

独立性：あるテストは他のテストに依存せず，常に同じ結果を示す

再現性：テストを異なる条件下で実行しても，常に同じ結果を示す

・テスタブルなプロダクトコード

同一性：同じ入力に対し常に同じ出力を返却する

単独性：1 メソッド内のロジックは 1 つである

本稿では，以上 4 つの特徴を観点として開発者のソースコードを分析し，テスタブルでないと判断された箇所と原因を開発者に提示することで，テスタブルコード作成の支援を行う．これにより，要件②を達成する．

3.4 テスト作成支援

テストの作成は通常の実装工程では行われない作業であり，現場の開発者であっても負担となると言われている[(4)(5)]ため，その負担を軽減する支援が必要である．本稿では，テストを全て手動で作成するのではなく，テスト項目に対してテストデータを与えることでテストコードを自動的に生成する手法により支援を実現する．これにより，要件③を達成する．

4. システム設計

3 章で述べた支援手法を実装したシステム開発に向け，システム要件や機能の洗い出しを行う．システム要件とそれに対応する機能は以下の通りである．

①：開発者の開発状況をモニタリングし，開発履歴を収集すること（開発履歴収集機能）

②：モニタリングに基づき現在の開発状況を把握すること（開発状況把握機能）

③：ソフトウェア開発プロセスと TDD の開発サイクルに則った開発ができるようナビゲートすること（開発ナビゲート機能）

④：テスタブルコードかどうかを判定しアドバイスを提示できること（テスタブルコード判定機能）

⑤：テストデータからテストコードを自動生成できること（テストコード自動生成機能）

開発履歴収集機能は，他の機能全ての基礎となる機能である．開発者の開発状況を常にモニタリングし，ソースコードのスナップショットや，どのようなナビゲートを行ったか等他の機能が発生させる履歴等を収集する．

開発状況把握機能では，収集した履歴により開発者の現在の開発状況をリアルタイムに把握する．

開発ナビゲート機能では，把握した状況を用いて，開発者が現在どの段階にあり次何をするべきかを提示したり，テストを作成しない等 TDD のサイクルから外れた時にその旨を警告したりする．さらに，予め定義した形式に則った機能設計書を読み込ませることで ToDo リストを自動生成し，リスト中のテスト項目とテストコードを対応付けることで，両者の関係を視覚化する．

テスタブルコード判定機能では，収集したソースコードのスナップショットを解析してソフトウェアメトリクスを計測し，メソッドのシグネチャや規模，コード内の重複等からテスタブルでない箇所を推定し，原因と共に学習者に提示する．

テストコード自動生成機能では，システムにテストデータの組み合わせを与えることで，テストコードを自動的に生成する．また，Web アプリケーション開発のことを考慮し，フォームからの入力やデータベース部をテストするためのテストコードも自動的に生成する．

5. おわりに

本稿では，ソフトウェア開発初心者の TDD スキル育成を目的とし，その支援手法の開発を行った．TDD を初心者に行わせることの問題点と要件を挙げ，支援手法の具体化と開発を行い，システムとして実装するための要件と機能を検討した．

今後は，検討した要件に基づいてシステムを実装し，実際のソフトウェア開発演習の中で用いて評価を行う．

参考文献

(1) K. Beck: “Test-Driven Development: By Example”, Addison-Wesley Professional（2003）

(2) 吉田英輔，角川裕次:“テスト駆動開発に基づくプログラミング学習支援システム: 初心者開発者のためのセルフトレーニングアーキテクチャ”，電子情報通信学会技術研究報告．SS，Vol.105，No.331，pp.27-32（2005）

(3) 上河内頌之，松浦佐江子:“Java プログラミング初学者に対するテスト方法学習支援ツール”，電子情報通信学会技術研究報告．ET，Vol.106，No.364，pp.37-42（2006）

(4) B. George and L. Williams: “A structured experiment of test-driven development”, Journal of Information and Software Technology，Vol.46，No.5，pp.337-342 （2004）

(5) N. Nagappan，M. E. Maximilien，T. Bhat，et al.: “Realizing quality improvement through test driven development: results and experiences of four industrial teams”, Journal of Empirical Software Engineering，Vol.13，pp.289-302 （2008）