Vol.2010-CE-103 No.21
2010/3/7

テスト駆動型開発手法のJavaプログラミング教育応用におけるテストコード提出機能

福山裕輝[†1]　舩曳信生[†1]
中西透[†1]　天野憲樹[†1]

本グループでは，これまで Java プログラミング教育における学生の学習支援，教員の負担軽減を目的として，テスト駆動型開発手法による，その支援システムを提案している．本システムでは，教員は，まず課題に対する模範解答コードとテストコードを登録する．次に学生は，テストコードを仕様書として解答コードを作成・提出する．その上でシステムがその自動検証を行う．本稿では，学生のソフトウェアテストに関する知識を深め，正しいコード仕様作成のための教育支援を目的として，学生によるテストコードの作成・提出を可能とするシステムの拡張を行う．併せて，本システムの実用性を高めるため，複数の講義科目での利用，セキュアなプログラムテスト環境の構築も行う．後者は，学生の不完全なコードが及ぼすシステムへの悪影響の抑制を狙いとしている．学生 25 名による評価実験により，本システムの有効性を示す．

Test Code Assignment Function in Java Programming Education Support System Using Test-Driven Development Method

Yuki Fukuyama ,[†1] Nobuo Funabiki,[†1]
Toru Nakanishi[†1] and Noriki Amano[†1]

Based on the test-driven development (TDD) method, we have developed a Web-based Java programming education support system to help learning activities of Java programming by students and to reduce loads by teachers. Using a software tool for the TDD method called *JUnit*, this system can verify the source codes made by students automatically after the teacher register the correct source code and the test code for the assignment. However, this system has three drawbacks. The first drawback is that it can be used at only one class. The second one is that it may not work properly if students submit codes containing errors and incorrect procedures. The third one is that it does not allow students to make test codes, although the programming of test codes can help the understanding of software tests and software designs. In this study, we expand this Java programming education support system so that it can solve the three drawbacks. Particularly, for the last expansion, it asks teachers to register the code specification in the assignment in addition to the correct code and the test code. Through experiments to 25 students, we verify the effectiveness of our proposal.

1. まえがき

本グループでは，学生の Java プログラミング教育の支援，教員の負担軽減を目的として，テスト駆動型開発手法[1)] による Java プログラミング教育支援システムを提案している．本システムは，教員が登録した課題およびテストコードを仕様書として，学生が課題に対する解答プログラム（本稿では解答コードと呼ぶ）を作成し，オンラインで検証する Web システムである．

ここで，現状のシステムでは，学生は教員が登録したテストコードを仕様書として解答コードを作成するのみである．学生自身がテストコードを作成し，その検証を行うことはできない．そのため，学生のテストコードに関する知識を深めることが出来ないといった問題点がある．テストコードの作成は同時に，エラー処理を含む，詳細なプログラム仕様を検討することにも繋がり，必要なすべての機能を網羅的に有する解答コード作成のための教育手段としても，有効であると考えられる．

そこで本稿では，学生によるテストコードの作成・提出と，その検証を可能とする拡張を行なう．その際，同時に，本システムの実用性を高めるために，複数の講義科目への対応，セキュアなプログラム実行環境の導入も行う．本システムは，これまで同時には一つの講義科目でのみ，利用可能となる実装がなされている．これに対して本稿では，データベースを用いて，登録された複数の講義科目の中から受講者毎にアクセス可能な科目を選択し，その科目の課題のみを表示するように拡張する．また，学生のテストコード，解答コードの解析により，ファイルアクセスや外部コマンドの使用など，本システムに悪影響を及ぼしかねない処理の実行を未然に防ぐこととする．同時に，学生により提出されたプログラムの実行中

†1 岡山大学
Okayama University

に，その実行時間を監視することによって，無限ループが含まれる処理の実行による本システムの誤動作を防止する．

以下，本論文の構成を示す．2. で本研究の前提となるテスト駆動型開発手法を紹介する．3. で従来システムとその問題点について述べる．4. で提案するシステムの拡張について述べる．5. で評価について述べる．6. で関連研究を紹介する．最後に 7. でまとめと今後の課題を示す．

2. テスト駆動型開発手法

本章では，本研究の前提となるテスト駆動型開発手法，および，本研究で採用した同手法のツールの 1 つである JUnit[2] について述べる．

2.1 テスト駆動型開発手法

テスト駆動型開発 (test-driven development; TDD) とは，プログラム開発手法の一種で，プログラム本体よりも先にテストケースを作成するスタイルを取る手法である．ここでテストケースとは，プログラムの仕様テストを行なうためのプログラムを指す．このスタイルはテストファーストとも呼ばれ，多くのアジャイルソフトウェア開発手法[3]，例えばエクストリームプログラミング[4] などにおいて，強く推奨されている．以下にテスト駆動型開発の開発サイクルとその利点を示す．

TDD での基本となる開発サイクルは以下となる．

(1) テストコードを作成する
(2) テストがパスするようにプログラム本体を作成する
(3) 全てのテストがパスするまで，プログラムの修正とテスト (2) と (3) を繰り返す

TDD での利点を 3 つ，以下に示す．

(1) テストコードがドキュメントとしての役割も持つこと
テストコードには，テストすべきプログラムの仕様（機能）が全て網羅されていなければならない．すなわち，テストコードを見ればそのプログラムがどのような動きをするのかが分かる．
(2) 効率の良いテストが行えること
詳細な仕様単位でのテストが行えるため，効率の良いテストを行うことができる．
(3) 改良 (リファクタリング) がしやすくなること
プログラムを改良した場合にも，それが仕様を満たしているかどうかを機械的に判断させることが可能となり，リファクタリングにも役立つ．

2.2 JUnit

本研究のシステムでは，学生から提出されたプログラム（解答コード）が，教員の定めた機能（動作仕様）を満たしているかどうかを，ツールを用いて自動検証する．この検証ツールには，Java プログラムのテストツールである，*JUnit* を利用する．JUnit は，Java プログラムにおいて，ソフトウェアを部品単位で独立してテストを行なう，ユニットテスト（単体テスト）自動化のためのツールである．ここで，Java ではクラス (class) がユニットに相当する．JUnit では，テストコード作成支援からテスト実行までを行うことができる．JUnit は，Java での開発に慣れたユーザには自然に受け入れられるように設計されている．テストコードを記述するために多くの知識を習得する必要はなく，数個の規則を学ぶだけで良い．そのため，保守性の高いテストコードを手軽に作成できるという利点もある．

本システムでは，教員がプログラミング課題を提示する際に，同時にテストコードも作成する．このテストコードには JUnit のライブラリを使用する．以下で，JUnit におけるテストコードの作成方法を説明する．

2.3 JUnit におけるテストコード

ここでは簡単な Math クラスを例に，JUnit によるテストコードの記述方法を説明する．

- テスト対象コード

以下に，テスト対象コードを示す．

```
1: public class Math{
2:     /*
3:     ここでは 2 つの引数を足した結果を返す plus メソッドを実装する
4:     */
5:     public int plus(int a, int b){
6:         return( a + b );
7:     }
8: }
```

上記 Math クラスには，足し算結果を返す plus メソッドを実装している．このクラスをテストするためのテストコード（テストクラス）は，以下の様に記述する．

- テストコード

```
1: import junit.framework.*;
2:
3: public class MathTest extends TestCase{
```

```
4:      /*
5:      testPlus メソッドは Math クラスの plus メソッドをテストするメソッド
6:      */
7:      public void testPlus(){
8:          Math ma = new Math();
9:          int result = ma.plus( 1, 1 );
10:         asserEquals(2, result); //期待値と実行値を比較
11:     }
12:}
```

あるクラスのテストコードでは，junit.framework.TestCase を拡張したテストクラスを作成する．この例では Math クラスに対応する MathTest クラスを作成している．テストコードの 1 行目で，TestCase などを含む junit.framework パッケージをインポートしている．3 行目で TestCase を拡張し，テストクラスとして MathTest クラスを宣言している．テストクラスの中に，対象クラスのメソッドのテストを行うプログラムを記述する．この例では足し算の結果を取得する plus メソッドのテストメソッドを記述する．

テストメソッドの名前は,「test」で始める必要がある．ここでは Math クラスの plus メソッドのテストメソッドのため「testPlus」といった名前にしている (7 行目)．なお，テストメソッドは 1 つのテストクラスに複数設定しても構わない．テストメソッドでは，文字通り,「テストを行う」プログラムを記述する．plus() のテストをする場合，以下の流れでテストを行う．

(1)　Math クラスのインスタンスを作成する
(2)　作成したインスタンスの plus メソッドに適当な引数を入れて呼び出す
(3)　2 で取得した値が正しい値かどうかを assert メソッドで比較する

testPlus メソッドは，上記の手順を記述したプログラムとなる．JUnit では，テスト結果が期待通りかどうかの判定は，assert メソッドを使用する．今回はその中の assertEquals メソッドでテスト結果を判定している．assertEquals メソッドでは 2 つの引数が等しいかどうかを判断する．1 つ目の引数は期待される結果 (期待値)，2 つ目の引数は実際の結果 (実行値) である．

JUnit ではこのようにテストコードを作成することができる．また，assert メソッドの使い方次第で，様々なテストを行うことが可能である．

3. 従来システムとその問題点

本章では，本グループによる従来の Java プログラミング教育支援システムの概要とその問題点を述べる．その上で，本研究の目的を述べる．

3.1 従来システムの概要

本グループでは，Java プログラミング教育において，学生の学習支援，教員の負荷軽減を目的として，Web ベースのテスト駆動型開発手法によるその支援システムを提案している．本システムでは，テスト駆動型開発手法のためのツールである JUnit を用いて，学生が提出したプログラム（解答コード）を，教員があらかじめ登録しておくテストコードによって自動検証する．

図 1 に，本システムの構成を示す．教員は，Web ブラウザを用いてプログラミング課題を登録する際，課題文，模範解答コード，テストコードを登録する．学生には，課題文，テストコードが提示され，テストコードをプログラム仕様書として，そのテスト項目が充足されるように解答コードを作成，提出する．提出された解答コードの検証は Web サーバでオンラインで行われ，検証結果は即座に学生にフィードバックされる．そのため，教員に負担を与えずに，学生はテストをパスする正しい解答コードの作成を行うことが可能となる．

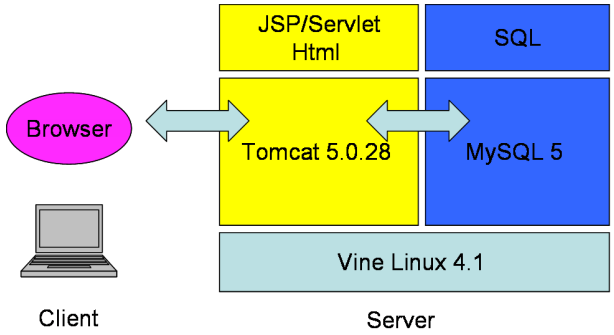

図 1　システム構成

3.2 従来システムの問題点

本節では，従来システムの問題点を挙げる．まず，学生によるテストコードの作成が出来ない点が挙げられる．テストコードは，プログラムレベルの仕様書と等価であり，その正し

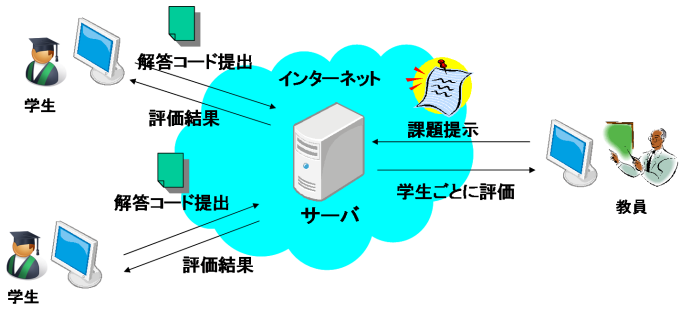

図 2 オンラインでのプログラム課題評価

い作成は，プログラミング教育において非常に重要である．また，学生時代から，社会で重視されているソフトウェアテストについて学ぶことは非常に有用であると言える．

次に，従来システムは複数の講義科目に対応していない点が挙げられる． Java プログラミング教育に関連する講義科目は複数存在する場合，その全ての科目の全課題が同一データベース上に登録される．それを避けるためには，科目毎に独立して本システムの起動が必要となるが，これは，システム管理上，望ましいことではない．

更には，セキュアなプログラム実行環境を提供できていない点が挙げられる．繰り返し処理を含む課題が提示された場合，提出された解答コードの不備により，無限ループが発生する恐れがある．その場合にも，従来システムではそれを実行してしまい，停止できない．また，解答コードにシステムファイルの書き換えといった悪意のある処理が記述されていたとしても，従来システムはそのコードを実行してしまう．

そこで本稿では，本システムの実用性向上を目的に，学生によるテストコードの提出，複数の講義科目での利用，セキュアな実行環境の構築の 3 つの拡張を行なう．

4. 提案する 3 つのシステム拡張

本章では，本稿で提案する，本システムの 3 つの拡張について述べる．

4.1 学生によるテストコードの提出

4.1.1 テストコード検証方法

学生によるテストコードの作成・提出を行う場合，学生の解答コードの正当性検証に加え，学生のテストコードの正当性検証も行わなければならない．これは，学生のテストコードによる，教員の模範解答コードの検証により実現する．そのためには，学生のテストコードで使用するクラス，メソッドの構成を，模範解答コードのものと一致するように，予め学生に指示しておく必要がある．そこで，本システムでは，教員は必要な情報（**名称リスト**と呼ぶこととする）も提示することとしている．

図 3 学生の作成したテストコードの検証

4.1.2 教員による課題登録

本システムでの課題登録時，教員は，学生のテストコード検証用として，模範解答コードと名称リスト，学生の解答コード検証用として，模範テストコードを登録する．名称リストには，学生がテストコードを作成する際に必要となる解答コードのクラス名，テスト対象のメソッド名，メソッドの引数の構成と戻り値の型が記述される．ここで，本システムでは，教員が登録した模範解答コードからクラス名，メソッド名のリストを出力する機能を実装することで，名称リスト作成における教員負荷を軽減している．図 4 に，2 値の加算を行なうクラスの模範解答コードと，出力される名称リストの雛形を示す．

模範解答コード(MyPlus.java)

```
1 public class MyPlus{
2         int plus(int a, int b){
3                 return a + b;
4         }
5 }
6
```

名称リスト

```
クラス: MyPlus
メソッド:
int plus(int,int)
処理:
```

図 4 教員が登録する模範解答コードと名称リストの雛形

4.1.3 テストコード提出手順

本節では，学生によるテストコードの作成・提出と，その検証の手順を説明する．まず，

本システムアクセス後，学生には，教員によるプログラミング課題が講義科目別に見えている．対象科目を選択すると，その課題一覧が表示される．ここでは，課題毎に，課題番号，内容（課題文），提出状況が表示され,「テスト作成」ボタンによりテストコードの作成・提出,「解答」ボタンにより解答コードの作成・提出を行うことができる．

次に，図 5 に示す，テストコードの作成・提出のための画面では，課題文，名称リスト，解答コード入力欄 (コードエディタ)，解答ファイルのアップロード欄が用意されている．解答コードの提出方法には，コードエディタによるものと，予めローカル環境で作成した解答コードファイルのアップロードの 2 つの方法がある．コードエディタを使用する場合，学生は一連の作業を Web ブラウザのみで行うことができる．但し，コードエディタの機能は，eclipse などの統合開発環境に比べて機能が低いため，ローカル環境で解答コードを作成してから，そのファイルのアップロードを推奨している．

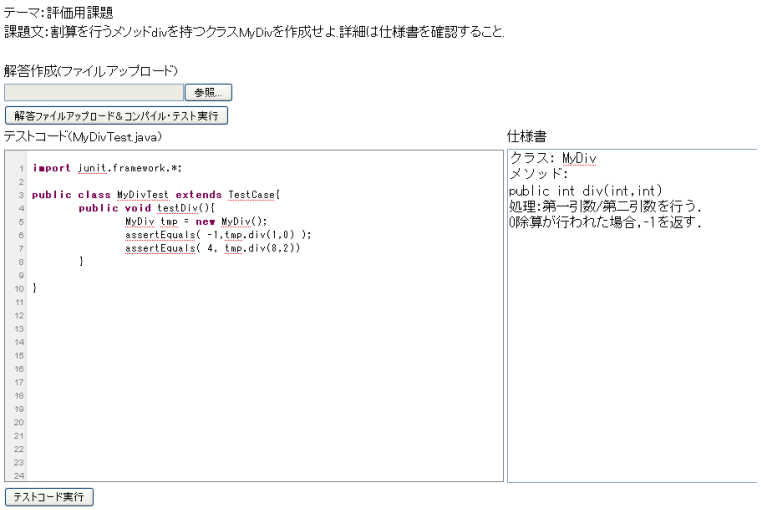

図 5 テストコード作成・提出画面

テストコードの作成・提出後,「テストコード実行」ボタンによりテストコードの検証が行なわれる．図 6 に示すように，検証結果として，テストコードのコンパイル結果，テスト結果，コーディングルールチェック結果が表示される．検証結果が正しい（正しくなくとも，諦めた）場合,「この内容で提出」ボタンを押すことでテストコード登録が完了する．この登録後も，再度編集して登録しなおすことが可能である．その後，解答コードの作成・提出を行なう．

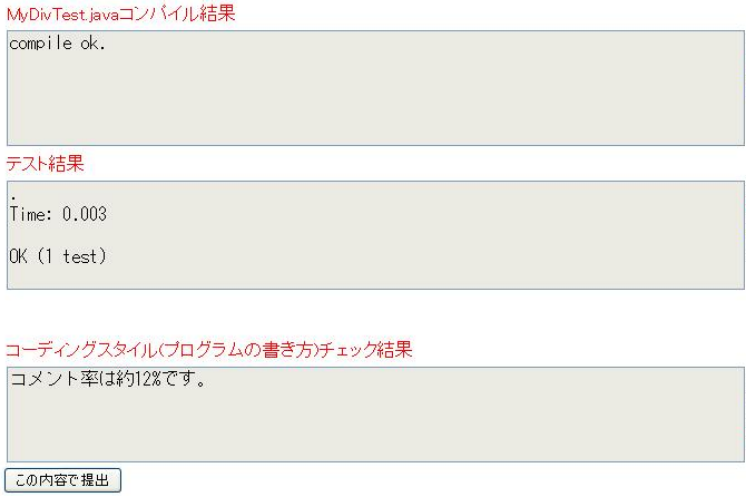

図 6 テストコード検証画面

4.1.4 提出コードの検証

まず，本システムに提出された学生の解答コードは，教員による模範テストコードで検証することで，その正しさを検証する． 検証結果が正しい場合，解答コードは正しいと言える．

次に，学生のテストコードは，それにより教員の模範解答コードを検証することで，その正しさを検証する．ここでは，テストがパスした場合でも，課題で要求されているすべての機能や仕様の検証がテストコードに含まれているとは言えないため，現時点での学生テストコードの検証方法は不十分である．

最後に，学生の解答コードをそのテストコードで検証することで，学生がテストコードと解答コードの組合せを正しく作成していることを検証する．ここで，模範コードにより正しく検証された解答コードを，学生のテストコードが正しく検証した場合には，テストコードも正しい可能性が非常に高いと言える．

4.2 複数の講義科目での利用

複数の講義科目での利用のために，本稿ではデータベースによる科目管理機能，履修学生管理機能を実装した．教員は，新規に本システムを利用する講義科目を担当する際，教員の情報，科目名，履修学生のリストをデータベースに登録する．それにより，教員は担当科目のみで，課題作成や提出結果の閲覧を行うことができる．また，学生は履修している科目のみで，課題の閲覧や提出を可能としている．

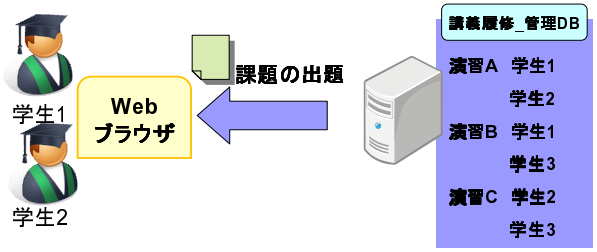

図 7 複数講義科目のためのデータベース管理

図 7 の例では，学生 1 は演習 A と演習 B を履修しているため，その 2 つの科目のみ表示される．また，学生 2 は演習 A と演習 C を履修しているため，その 2 つの科目のみ表示される．この拡張により，各学生には履修していない科目の課題が表示されなくなり，システム利用上の混乱を避けることができる．

4.3 セキュアなプログラム実行環境

学生による不完全な解答コードの実行による，本システムへの影響を防ぐ方法として，以下の 2 点を採用する．以下にその詳細を述べる．

(1) ソースコード解析によるプログラム実行制限
(2) スレッドによる実行時間監視

4.3.1 ソースコード解析によるプログラム実行制限

ここでは，本システムでの検証のために学生からの解答コードを実行する前に，ファイル書き込み，外部コマンド実行など，実行時に本システムに悪影響を及ぼす恐れのある処理が含まれる場合に備え，解答コードのソースコード解析を行う．具体的には，解答コードに以下のクラスが含まれる場合，その検証（実行）を行なわずに，検証失敗を通知する．

(1) File クラス，BufferedWriter,PrintWriter など，ファイル入出力を扱うクラス
(2) Process クラス，Runtime クラスなど，外部コマンドを扱えるクラス

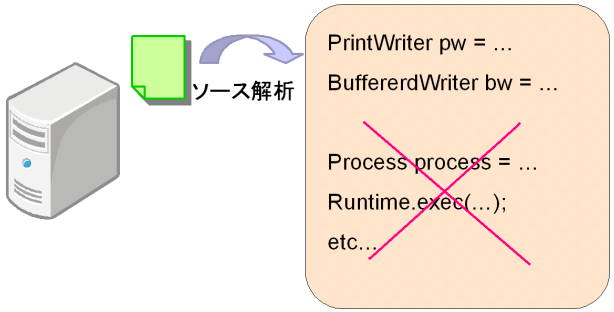

図 8 ソースコード解析

4.3.2 スレッドによる実行時間監視

ここでは，学生からのテストコードや解答コードに無限ループが含まれる場合に備え，スレッドを用いた解答コードの実行を行うことで，その実行時間を監視する．スレッドによる実行時間監視の流れを図 9 に示す．

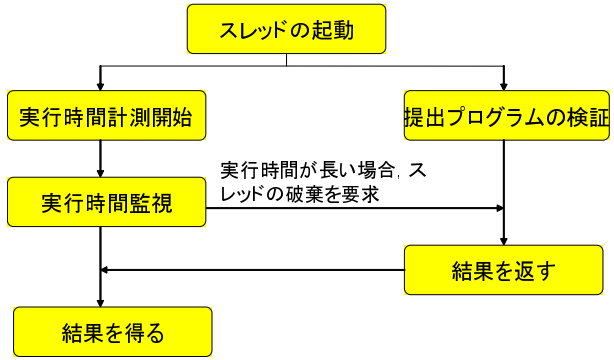

図 9 実行時間の監視

学生からテストコード，解答コードが提出された際，本システムのサーバでは，スレッドを起動し，スレッド上で各プログラムの実行を行なう．次に，スレッドを起動したメイン処

表 1 各課題での検証結果

項目	課題 1	課題 2
テストコード作成成功者数	25	25
課題正解者数	24	25

理において，プログラムの実行時間を測定する．そして，一定時間が経過してもプログラムの処理が終了しない場合，プログラムが無限ループを含んでいるものと判断し，スレッドを破棄すると共に検証失敗とする．

5. 評　　価

提案システムの評価のために，Java プログラミングレベルの異なる学生 25 名（本学学生 8 名，専門学校生 17 名）を対象に，本システムの適用後，アンケートによる 5 段階評価を行った．

5.1 プログラミング課題と検証結果

Java プログラミング課題として，(課題 1)2 値の除算結果を出力するクラス，(課題 2) 引数の値によって異なるメッセージを出力するクラス，の 2 つを与えた．これらは，初心者レベルの学生にも容易であり，同時にテストコードにおいて複数項目のテストを行なう必要のある課題として，選択した．

表 1 に各課題でのテストコード，解答コードの検証結果を示す．ここで，学生が提出したテストコードの中には，本システムで正しいと判定されたにも拘わらず，テスト項目に抜けのあるものも存在した．これは，学生のテストコードの検証を，教員の模範解答コードを検証することで実施した際に，テスト項目に抜けがある場合にも，テスト成功と判定するためである．その対策として，テストコードが検証（実行）する模範解答コードのパスを，コードカバレッジツール[5] を用いて調査することが挙げられる．すなわち，模範コードに，テストコードが検証しないパスが存在するようであれば，テストコードにテスト項目漏れがあると判定する．

5.2 アンケート評価

表 2 にアンケート項目，表 3 にその回答結果を示す．

表 3 より，設問 1 では半数以上の学生が 2 以下を選んでいることから，本システムのユーザインターフェースが必ずしも使い易いものではないと言える．これは，本システムで採用したエディタの機能が FireFox 以外のブラウザでは十分に機能しないこと，検証結果のエラーメッセージがわかりづらいことが，原因として挙げられる．今後は，これらの改善が必要である．

表 2 アンケート項目

項目	内容
Q1	ユーザインターフェースは使い易かったか
Q2	テストコードの作成は難しかったか
Q3	プログラムに必要とされる機能をテストコードの作成で実装し易くなったか
Q4	「テストコード作成→プログラム作成」の流れが理解できたか
Q5	本システムはプログラムに必要とされる機能の検証に役立つか
Q6	本システムを利用したことでプログラムテストに対する理解が深まったか

表 3 アンケート結果 (5 段階評価)

項目		1	2	3	4	5	
Q1	使い難い	10	3	6	5	1	使い易い
Q2	難しかった	4	5	11	2	3	易しかった
Q3	作り易くならなかった	7	2	7	2	7	作り易くなった
Q4	理解できなかった	2	2	4	8	9	理解できた
Q5	役に立たない	5	2	5	6	7	役に立つ
Q6	理解できない	3	1	6	7	8	理解が深まった

設問 4,5,6 では，半数以上の学生が 4 以上を選んでいることから，多くの学生はプログラムテストに関する知識を深め，その 1 つであるテスト駆動型開発の流れについて理解できていると言える．すなわち，本システムは学生のテストに関する学習に役立っている．

設問 2,3 では，2 以下の評価，4 以上の評価の割合が同程度であるが，これは今回の評価実験の対象が様々な Java プログラミングレベルの学生であったことが原因として考えられる．プログラミング経験の浅い学生にとっては，解答コードに必要とされる機能（＝テスト項目）の把握が困難であり，テストコードの作成も難しかったものと考えられる．今後，経験の浅い学生にもテストコードの作成を容易に行なえるように，システムの改善が必要である．それには，標準的なテスト項目のタイプと，タイプ毎のテストコードの書き方の一覧の提示などが挙げられる．

6. 関 連 研 究

文献[6] では，「失敗から新たな知識を学ぶ」失敗学の概念をプログラミング教育に適用することを狙いとした支援システムを提案し，学部 2，3 年生 21 名に対する評価実験を通じて効果を示している．ここでは，プログラミング教育における事象・概念を失敗学における失敗知識に対応させている．具体的には，エラー（コンパイルエラー，実行エラー，論理エ

ラー）を「事象」，演習課題を「背景」，エラー発生時のソースプログラムを「経過」に対応させ,「原因」,「対処」,「総括」は学習者に記述させることで失敗情報の知識化を行い，内省を支援する．実際には本システムでは，エラー発生時のソース該当箇所の提示と，その原因や対処を記入するメモ欄を実装しているのみである．そのため，エラー発生時に，学習者がその原因や対処方法を文章として整理し，システムに記憶させる（メモを残す）ことに，本システムの独自性があると言えるが，それだけのものである．ノートに記述することも可能であり，Web システムとしての特徴，長所は活かせていない．各学習者に，それらをノートに記載させるだけでも，同様の効果が得られるものと思われる．また，論理エラーは学習者による手入力のため，精度，手間の点で問題がある．実際，多くの学生は，演習課題に対して，コンパイルエラー，実行エラーは 0 とすることができるが，論理エラーを 0 とすることに難があるのが現状である．評価実験の事後テストにおいて，適用群と非適用群間に，正答率の平均値に有意差がなく，標準偏差に有意差があると言っているのみである．すなわち，評価も十分とは言えない．

文献[7] では，プログラミング教育において，個々の学習者の理解度と意欲に応じた演習課題を出題する手法を提案し，立命館大学での開講科目での適用結果を示している．学習者の理解度の評価は，演習課題の達成度に対する協調フィルタリングを用いる．協調フィルタリングは，Web サイト検索などで実用化されている，ユーザの傾向や嗜好を過去の行動で記録し，それを類似の行動を取るユーザの情報を元に，そのユーザの今後の傾向や嗜好を推測する手法である．達成度は，課題毎に設定された 10〜20 個程度の評価観点に対して，重要度を 1〜10 で設定し，それぞれに対して,「理解できている」,「どちらとも言えない」,「理解できていない」の 3 段階で教員が主観的に評価を行った結果を，重みでの加重平均を取ることで算出する．意欲は，教員が課題毎に 3 段階で主観的に行い，提出された課題での最低提出必要課題数に関する平均値で評価する．本手法の問題点は，まず，教員の負担が非常に大きいこと，公平性に問題があることである．多数の学生からの多くの課題のそれぞれに対し，多数の評価観点からの評価は困難である．評価の自動化，アルゴリズム化が不可欠である．また，多数の学生が課題の未提出となっており，推薦課題を提出したのは少数の元々意欲の高い学生と考えられることから，評価結果の分析は正確ではない（提案手法の有効性は言えない）と思われる．但し，本研究に対しては，学習者のプログラミング能力に関するプロファイルの作成と，それに基づいての，適切な課題提示，個別指導が行える機能の実装が挙げられる．

文献[8] では，学習者の状態，プログラミング教育，支援システムの分析結果をモデル化し，定義したそれぞれの情報に基づく，支援システム推薦アルゴリズムを提案し，支援システム選出のための枠組みを示している．プログラミング教育において，学習者を支援するシステムは数多く開発され，運用されている．しかし，それらの支援システムを効率的に使用するためには，学習者の状態に対応した支援システムを推薦する必要がある．本研究に対しては，学生個人のプログラミング能力を定義し，情報をデータベースに登録することで，レベルに応じた課題の提示機能の実装が挙げられる．

7. ま と め

本稿では，本グループが従来より提案している，テスト駆動型開発手法による Java プログラミング教育支援システムにおいて，学生によるテストコードの作成・提出，複数の講義科目への対応，セキュアなプログラム実行環境の構築の 3 つの拡張を行なった．様々な Java プログラミングレベルの学生 25 名に対する適用実験とアンケート評価により，その有効性の検証と問題点の摘出を行った．その中から，今後の課題として，ユーザインターフェースの改善，学生へのテストコード作成支援，テストコード検証方法の改善などが挙げられる．

参 考 文 献

1) Kent Beck：テスト駆動型開発入門 (2003)
2) JUnit.org, http://www.junit.org/
3) アジャイルソフトウェア開発：http://www.atmarkit.co.jp/aig/04biz/asd.html
4) エクストリームプログラミング：http://www.atmarkit.co.jp/aig/04biz/xp.html
5) コードカバレッジツール “Cobertura”：http://cobertura.sourceforge.net/
6) 知見邦彦, 櫨山淳雄, 宮寺庸造：“失敗知識を利用したプログラミング学習環境の構築, ” 信学論 D-I, vol. J88-D-I, no. 2, pp. 66-75(2005)
7) 田口浩, 糸賀裕弥, 毛利公一, 山本哲男, 島川博光：“個々の学習者の理解状況と学習意欲に合わせたプログラミング教育支援,” 情処論, Vol.48 No.2, pp. 958-96(2007)
8) 山本耕大，中村勝一，森本康彦，横山節雄，宮寺庸造：“プログラミング教育における学習者に適応的な支援システムの推薦手法” 信学技報，ET2009-12，pp.175-180(2009)