テスティング（応用）

平成２７年度シラバス

２０１５年1月９日

国立情報学研究所

トップエスイープロジェクト

代表者　本位田　真一

1. 科目名

テスティング（応用）

2. 担当者

加瀬　直樹

3. 本科目の目的

本科目では、産業ソフトウェアのテスティング問題を扱う。近年ますます大規模化・複雑化するソフトウェアの設計において、その動作の正しさを保証することがより困難になっており、プログラムの動作の自動的、または半自動的なテスティングの支援を行う各種ツールを使用し、実際のシステム開発に適用する方法を習得すること必要がある。しかし、これらのテスティングツールを使いこなすには多くの難しさがあり、普及への障害となっている。テスティング(基礎)で学んだテスト設計の技術は、実際に動作させ合否結果を判定できてこそ価値を見出す。そこで本科目では、現実的なプログラムのテスティングを通して、ソフトウェア・テスティングの難しさを体感した上で、テスティングツールの適用における難しさを解決するために必要なノウハウの体得により、組込みシステムからネットワークまで対応可能な、ソフトウェア・テスティングの実装技術を習得する。

4. 本科目のオリジナリティ

テスティングツールを使用したソフトウェア・テスティングについては、従来多くの同様な講座・講義が開講されている。しかしそれらの講座・講義にはさまざまな問題点があったため、受講生が習得した内容を速やかに開発現場で適用するための障害となっていた。本科目では、それらの問題点を解消し、本科目受講後テスティングツールを速やかに開発現場で適用できるように配慮している。**表 1** に、既存の講座・講義の問題点と、本科目における解を示す。

表 1 既存の講座・講義の問題点と、本科目における解

既存の講座・講義の問題点	本科目における解
テストの概念と基本的な技法の習得を目的とした簡単な例を扱うのみで、開発現場での適用が難しい	基本的な例題から、複雑で現実的な問題に順に取り組むことによって、テスティングの基礎的素養を養うため、現実的な開発プロセスへのシームレスな組入れが可能

5. 本科目で扱う難しさ

近年、ソフトウェアは大規模化、複雑化が進んでいる。特に、複数の機器がネットワークに接続され、相互に連携して動作することでさまざまな機能を実現する、スマートコミュニティネットワークが急速に立ち上がりつつある。従来の機器の制御ソフトウェアは単体で動作するように設計されていたのに対して、スマートコミュニティネットワークでは、他の機器との連携動作のための柔軟なプロトコルなどが必要になるため、制御ソフトウェアが非常に複雑になる。さらに、スマートコミュニティネットワークでは、動的に変更される接続相手の識別や、不安定なネットワーク環境などを考慮する必要がある。その一方で、大規模・複雑化、短納期化、頻繁なる仕様変更対応など、開発体制の環境も厳しさを増している。このような状況においては、品質を確保するための網羅性と効率性を両立したテスティング手法を用い、かつ効果的であることが求められている。また、既存開発や外注委託等の調達モジュールなど、外部モジュールとの組合せで製品開発を行い、品質を確保しなければならない状況にある。

6. 本科目で習得する技術

将来のネットワークでは、接続する機器や機器間で交換する情報の数・種類が従来型の機器よりも大幅に増加するため、その振る舞いの設計が大規模化、複雑化し、従来のアドホックなテスティングでは、プログラムの正しさを効果的にテストすることが困難になる。そこで近年では、テストケース作成やテスト計画における体系化されたプロセスや、単体テストツールやテスト管理ツールなどテスティング作業を自動的、または半自動的にサポートする各種ツールの利用が進みつつある。さらには、テスティング作業を中心にソフトウェア開発プロセス全体を再構築した、テスト駆動開発プロセスも広く知られてきている。

しかし、大規模ソフトウェア開発に、これらのプロセスやツールに代表される、ソフトウェア・テスティング技術を利用する場合には、各種技術の背景理論を理解した上で、様々なノウハウを駆使しなければ、効率的かつ効果的に活用することができない。たとえば、テストケース作成プロセスにおいては、テストケースを構成するデータ構造を熟知する必要があり、またテスト駆動開発プロセスにおいては、従来の開発プロセスの常識から離れる必要すらある。

本科目では、ソフトウェア・テスティングの基本知識を前提として、テストケースをプログラムとして実装するための技術を学ぶ。テストコードとデータの分離、ＧＵＩを介したテストの構成、テストを実施する際の代用コードを動的に作成するモック、テストと仕様とのつなぎであるビヘイビア駆動開発の技法を習得する。

7. 前提知識

本科目の受講生は、以下の項目を習得済みであることが望ましい。

・テスティング（基礎）

・Java プログラミング

8. 講義計画

第 1 回：テスティング概論
第 2 回：テスティングフレームワークとテスト駆動開発 TDD（JUnit）
第 3 回：テスティングフレームワークとテスト駆動開発 TDD（JUnit）
第 4 回：モック(JMock)
第 5 回：モック(JMock)
第 6 回：モック(JMock)
第 7 回：データ駆動開発 DDT（XLSBeans）
第 8 回：GUI を介したテスト(FEST-Swing)
第 9 回：GUI を介したテスト(FEST-Swing)
第 10 回：GUI を介したテスト(FEST-Swing)
第 11 回：GUI を介したテスト(FEST-Swing)
第 12 回：ビヘイビア駆動開発 BDD（JBehave）
第 13 回：ビヘイビア駆動開発 BDD（JBehave）
第 14 回：ビヘイビア駆動開発 BDD（JBehave）
第 15 回：まとめ

9. 教育効果

本科目を受講することにより、実際のシステム開発、あるいは組込みソフトウェアの開発に適用可能な、ソフトウェア・テスティングの構築技術を習得できる。その結果、開発現場において、テスティングツールを活用することにより、信頼性の高いシステムを、効率的に開発することができるようになる。

10. 使用ツール

JUnit：テスティングフレームワーク

- 使用する上での難しさ
  - テスト用コードの記述が難しい
- 使用上必要なノウハウ
  - テスト用コード記述ノウハウ
    - テストケースの表現
    - アサーションの表現
- 選択理由、実用性
  - Java プログラムの単体テストツールとして広く使用されている
  - 特にテスト駆動開発には必須

JMock：モッキングフレームワーク

- 使用する上での難しさ
  - モック用コードの記述が難しい
- 使用上必要なノウハウ
  - モック用コード記述ノウハウ
    - モックの動的作成
- 選択理由、実用性
  - JUnit のモッキングフレームワークとして広く使用されている

JBehave：BDD フレームワーク

- 使用する上での難しさ
  - 仕様とテストコードの対応付けが難しい
- 使用上必要なノウハウ
  - BDD 開発ノウハウ
    - 仕様解釈コード記述
    - 仕様とテストのマッピング記述
- 選択理由、実用性
  - BDD のフレームワークとして広く使用されている

FEST-Swing：Swing GUI のテスト自動化

- 使用する上での難しさ
  - GUI が絡むテストの自動化が難しい
- 使用上必要なノウハウ
  - GUI を介した自動テストの開発ノウハウ
- 選択理由、実用性
  - 扱いが簡単で、概念を理解しやすい

11. 実験及び演習

講義内でソフトウェア・テスティングのプログラミングを行い、難しさを実際に体感しながらソフトウェア・テスティングの実装方法を体得する。また、利点、欠点についての議論を行い、どのように実問題に導入していったら良いかを常に意識する。

12. 評価

課題レポート、講義・演習の理解度、出席日数等を総合して評価する。

13. 教科書/参考書

- Steve Freeman, Nat Pryce,「実践テスト駆動開発 ― テストに導かれてオブジェクト指向ソフトウェアを育てる」, 翔泳社, 2012.
  JUnit, JMock を使ってテストケースを実体化し、開発の上流からテスト駆動開発を実施するためのノウハウが記述されており、この科目に最適である。
- Janet Gregory, Lisa Crispin,「実践アジャイルテスト ― テスターとアジャイルチームのための実践ガイド」, 翔泳社, 2009.
  設計とテストとの関係性を考慮したテスティングの実践を扱っており、テストの考え方を整理する上で、この科目に最適である。

14. 受講に関しての注意

本科目は、講義の場での演習実施やレポートの考察を通して講義内容を理解する形であり、講義参加者には正規の「履修」を求める。相当の理由があるときにのみ「聴講」を許可するので、希望者は履修登録の際に理由を申告すること。業務多忙を理由とした「聴講」申請は認めない。