

State-Based Testing

NEIL EDELMAN, McGill University (110121860)

CASIMIR DÉSARMEAUX, McGill University

Assignment 1 report, part 1.

Categories and Subject Descriptors: D.1.3 [Software]: Concurrent Programming

General Terms: Software

Additional Key Words and Phrases: Subject

Reference Format:

N. Edelman, C. Désarmeaux. *State-Based Testing* (Fall 2015.)

304-429 (Section 001) A1

Instructor:	Prof. G. Mussbacher
Course:	ECSE 429 – Introduction to Software Quality Assurance
Date:	2015-10-05

Contents

1	Source Code	1
1.1	Source code	1
1.2	Generated code	1
2	Complete Test Class	1

1. SOURCE CODE

1.1. Source code

The source code that generates the test class for the CCoinBox example given the state machine definition and implementation of the state machine.

The code is in the package `ca.mcgill.ecse429.conformancetest.nplus`.

1.2. Generated code

The result of the test class generation for the CCoinBox example without any manual changes after generation. This class must be called `GeneratedTestCCoinBox.java` and saved in the same package as the implementation of the state machine.

The generated code is in `test`, following the same directory structure.

2. COMPLETE TEST CLASS

The complete test class for the CCoinBox example with additional code added manually as needed to fully test the CCoinBox state machine based on the N+ Test Strategy (conformance tests only). This class must be called `TestCCoinBox.java` and saved in the same package as the implementation of the state machine. Any manual changes have to be clearly identified in the complete test class. Any complete test class that cannot be executed as a JUnit test will result in a mark of 0 for this part.

The generated code is in `test`, following the same directory structure.

3. REPORT

- List.
- List.

Received 2015-10-05; revised 2015-10-05; accepted 2015-10-05