Test–Driven Learning: Intrinsic Integration of Software Testing into the CS/SE Curriculum

SIGCSE’06

David Janzen (djanzen@ku.edu)
Bethel College
University of Kansas

Hossein Saiedian (saiedian@eecs.ku.edu)
University of Kansas
Outline

- SIGCSE Dilemma
- Test-Driven Learning
- TDL Throughout the Curriculum
- TDL Benefits
- Evaluation
- Summary
The SIGCSE Dilemma

- So many good ideas, so little time
- Software Engineering perspective
  \[ CS \neq Programming, \text{ but} \]
  \[ CS \supseteq Programming \]
- Many objectives
  - Programming concepts
  - Programming language mastery
  - Good software design
  - Thorough software testing
- Many approaches, sometimes conflicting
  - Objects–, Procedures–, Events–, Everything–first
  - Topical (design course, testing course) or integrated courses

Janzen/Saiedian – SIGCSE 2006 • 3
Context

• How do we teach programming?
  – Concepts
    • Algorithms, encapsulation, semantics, …
  – Examples
    • Software presented in a particular language
  – Practice
    • Lab and project assignments
Teach By Example

- Traditional approach in lecture

```cpp
// This function sums the integers from min to max inclusive.
// Pre: min < max
// Post: return-value = min + (min+1) + ... + (max-1) + max
int sum(int min, int max) {
    int sum = 0;
    for(int i=min;i<=max;i++)
        sum += i;
    return sum;
}

int main() {
    cout << sum(3,7) << endl; //should print 25
    cout << sum(-2,2) << endl; //should print 0
    cout << sum(-4,-2) << endl; //should print -9
}
```
Teach By Example

- Traditional approach in texts or resources

```java
void printClassName(Object obj) {
    System.out.println("The class of " + obj + " is " +
                        obj.getClass().getName());
}
```

From Java 1.5 API

```cpp
#include <iostream>
using namespace std;

int main()
{
    int age;
    cout << "What is your age in years?" << endl;
    cin >> age;
    cout << "You are at least " << age * 12
         << " months old!" << endl;
}
```

Janzen/Saiedian – SIGCSE 2006 • 6
Test–Driven Learning

• Simple Idea
  – Introduce and explore new concepts through automated unit tests

• Inspired by
  – Testing Patterns In Beck’s “TDD: by Example”
    • Explanation Test
      – ask for and provide explanations in terms of tests
    • Learning Test
      – if you want to use a new method, class, or API, first write tests to learn how it works
Teach By Example

• TDL Approach

```cpp
#include <cassert>

// This function sums the integers from min to max inclusive.
// Pre: min < max
// Post: return-value = min + (min+1) + ... + (max-1) + max
int sum(int min, int max) {
    int sum = 0;
    for(int i=min;i<=max;i++)
        sum += i;
    return sum;
}

int main() {
    assert(sum(3,7)==25);
    assert(sum(-2,2)==0);
    assert(sum(-4,-2)==-9);
}
```
Teach By Example

- Traditional Approach

```java
void printClassName(Object obj) {
    System.out.println("The class of " + obj + " is " +
                      obj.getClass().getName());
}
```

- TDL Approach

```java
void testClassName1() {
    Integer i = new Integer(5);
    assert i.toString().equals("5");
    assert i.getClass().getName().equals("java.lang.Integer");
}

void testClassName2() {
    ArrayList al = new ArrayList();
    assert al.toString().equals("[]");
    assert al.getClass().getName().equals("java.util.ArrayList");
}
```
TDL Throughout Curriculum

- Any time you present an example with code, you can consider applying TDL

```cpp
#include <cassert>
class Exams { . . . };
int main()
{
    run_tests();
}
void run_tests()
{
    { //test 1 Minimum of empty list is 0
        Exams exam1;
        assert(exam1.getMin() == 0);
    } //test 1
    { //test 2
        Exams exam1;
        exam1.addExam(90);
        assert(exam1.getMin() == 90);
    } //test 2
}
```

Early examples might use simple asserts isolated in a test procedure
TDL in Data Structures

```java
import java.util Enumeration;
import javax.swing.tree.DefaultMutableTreeNode;
import junit.framework.TestCase;

public class TreeExploreTest extends TestCase {
    public void testNodeCreation() {
        DefaultMutableTreeNode node1 = new DefaultMutableTreeNode("Node1");
        DefaultMutableTreeNode node2 = new DefaultMutableTreeNode("Node2");
        DefaultMutableTreeNode node3 = new DefaultMutableTreeNode("Node3");
        DefaultMutableTreeNode node4 = new DefaultMutableTreeNode("Node4");
        node1.add(node2);
        node2.add(node3);
        node1.add(node4);
        Enumeration e = node1.breadthFirstEnumeration();
        assertEquals(e.nextElement(), node1);
        assertEquals(e.nextElement(), node2);
        assertEquals(e.nextElement(), node4);
        assertEquals(e.nextElement(), node3);
    }
}

Later examples might use JUnit or xUnit
```

---

Janzen/Saiedian – SIGCSE 2006 • 11
TDL in Static Analysis

```java
public void testFlowCombinations() {
    LiveVariableAnalyzer lva = new LiveVariableAnalyzer();
    Set flow = new HashSet();
    assureTrue(flow.isEmpty());
    lva.flow("[x:=3]1; while [x<9]2 do [y:=x]3", flow);
    assureEquals(flow.size(), 3);
    assureTrue(flow.contains(new Pair(1,2)));
    assureTrue(flow.contains(new Pair(2,3)));
    assureTrue(flow.contains(new Pair(3,2)));
}

public void testOneLV() {
    LiveVariableAnalyzer lva = new LiveVariableAnalyzer();
    lva.setProgram("[x:=5]1;[y:=x]2");
    lva.analyze();
    assureTrue(lva.getEntry(1).isEmpty());
    assureTrue(lva.getExit(1).contains(new Character('x')));
    assureTrue(lva.getEntry(2).contains(new Character('x')));
    assureTrue(lva.getExit(2).isEmpty());
}
```

TDL has been applied in graduate and corporate training courses.
Potential Benefits of TDL

- Teach testing for free
  - Replace current approach instead of add to it
- Improve student design skills
  - Focus on use (interface and behavior) before implementation

```python
assert(sum(3,7)==25);
```

- Improve student testing skills
  - Students emulate professors
- Improve student success on projects
  - More and earlier testing reduces defects and improves likelihood of delivering solution
Evaluation

- Short controlled experiment conducted
- CS1 Spring 2005 at University of Kansas

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Students</th>
<th>Exam 1 100 total</th>
<th>Quiz 10 total</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDL</td>
<td>13</td>
<td>86.15</td>
<td>7.84</td>
</tr>
<tr>
<td>Non-TDL</td>
<td>14</td>
<td>86.71</td>
<td>7.14</td>
</tr>
</tbody>
</table>

- Beware not to draw conclusions from such a small study!
Summary

• Test–Driven Learning
  – is a simple approach of presenting examples with automated tests
  – takes no extra time
  – can be applied at all levels of the curriculum
  – requires minimal tool support or configuration yet scales well
  – may encourage students to write more tests
  – may improve student testing skills
  – may improve student design skills
Resources

• Sample labs and project assignments available at http://www.simexusa.com/tdl/
• Questions or contributions: djanzen@ku.edu

Acknowledgements

• SIGCSE Special Projects Award