CS314 Software Engineering
Test Driven Development

Dave Matthews

Model View Presenter

Model

- updates model
- change events

Presenter

- updates view
- user events

View
MVP Hello World!

```java
public class Model {
    private String message;
    public Model() {
        message = "Hello World!";
    }
    public String getMessage() {
        return message;
    }
}

public class View {
    public View() {
    }
    public void display(String text) {
        System.out.println(text);
    }
}

public class Presenter {
    private View view;
    private Model model;
    public Presenter(View view, Model model) {
        this.view = view;
        this.model = model;
    }
    public void start() {
        String result = model.getMessage();
        view.display(result);
    }
}

public class Main {
    public static void main(String[] args) {
        Model model = new Model();
        View view = new View();
        Presenter presenter = new Presenter(view, model);
        presenter.start();
    }
}
```
Sprint 1 Test Data Observations

- Some still have issues with GitHub
  - master, notifications
- Data quality problems
  - outside Colorado
  - missing fields (altitude)
  - geographic coordinate formats not from Google Earth
  - characters (extra, commas, and quotes)
- Prevention? Correction?

Sprint 1 Test Data Correction

- Fix/add your line (and only your line)
- Two people must accept your pull request before you can merge/confirm
- Don’t allow others to merge/confirm bad data
- no duplicates, funny characters, extra commas, etc.

enName,brewery,city,latitude,longitude,altitude
davematt,Steamworks,Durango,37°16′27.47″ N,107°52′47.83″ W,6566
### CMMI for Development Model

<table>
<thead>
<tr>
<th>Maturity</th>
<th>Process</th>
<th>Project</th>
<th>Engineering</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>• Organizational Performance Management</td>
<td>• Quantitative Project Management</td>
<td>• Causal Analysis and Resolution</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>• Organizational Process Performance</td>
<td>• Integrated Project Management • Risk Management</td>
<td>• Requirements Development • Technical Solution • Product Integration • Verification • Validation</td>
<td>• Decision Analysis and Resolution</td>
</tr>
<tr>
<td>3</td>
<td>• Organizational Process Definition • Organizational Process Focus • Organizational Training</td>
<td></td>
<td>• Configuration Management • Measurement and Analysis • Process and Product Quality Assurance</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>• Requirements Management • Project Planning • Project Monitoring and Control • Supplier Agreement Management</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

### Verification and Validation

- **Goal is to establish confidence that the software is sufficient for its intended use**
- **Verification**
  - Are we building the right product?
  - Process of checking that the software meets its stated functional and non-functional requirements.
- **Validation**
  - Are we building the product right?
  - Ensure the software meets the customer’s expectations.
Scrum

Test Driven Development

https://www.scrum.org/
Shigeo Shingo

• Two kinds of tests
  – tests that find defects after they occur (a waste of time)
  – tests to prevent defects (the only kind to create)

Agile Testing Principles

• All code is tested code! Teams get no credit for delivering functionality that has been coded but not tested
• Tests are written before, or concurrently with, the code itself.
• Testing is a team effort. Testers and developers all write tests.
• Automation the the rule, not the exception.
Agile Testing Strategy

<table>
<thead>
<tr>
<th>Unit Test</th>
<th>Developers write unit tests for every class and method.</th>
<th>Each unit test returns a “pass” or “fail” against the developers build.</th>
<th>All unit tests must pass before code can be checked in.</th>
<th>Automated unit tests are run frequently (or continuously) against an integrated build of the system.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceptance Test</td>
<td>Testers / product owners write functional or acceptance tests for each new user story.</td>
<td>Acceptance tests are elaborated and written during the iteration planning and execution.</td>
<td>Acceptance tests are run during the iteration and serve as acceptance checkpoints for the iteration’s stories.</td>
<td>Acceptance tests are automated whenever possible and are added to the regression test suite at each iteration.</td>
</tr>
</tbody>
</table>

Test Driven Development

- Write a test - RED
  - fails because the code doesn’t yet exist
- Make it run - GREEN
  - quickly do the clean, simple, obvious solution
  - something that works
- Make it right - GREEN
  - refactor to something you can live with
  - clean code
Test Driven Development - Unit Tests

```java
public void testFibonacci() {
    assertEquals(0, fib(0));
}

int fib(int n) {
    return 0;
}
```

Test Driven Development - Example

```java
public void testFibonacci() {
    assertEquals(0, fib(0));
    assertEquals(1, fib(1));
}

int fib(int n) {
    if (n == 0) return 0;
    return 1;
}
```
public void testFibonacci() {
    int cases[][] = {{0,0},{1,1}};
    for (int i=0; i<cases.length; i++)
        assertEquals(cases[i][1], fib(cases[i][0]));
}

int fib(int n) {
    if (n == 0) return 0;
    return 1;
}
Test Driven Development - Example

public void testFibonacci() {
    int cases[][] = {{0,0},{1,1},{2,1},{3,2};
    for (int i=0; i<cases.length; i++)
        assertEquals(cases[i][1], fib(cases[i][0]));
}

int fib(int n) {
    if (n == 0) return 0;
    if (n <= 2) return 1
    return 1+1;
}
public void testFibonacci() {
    int cases[][] = {{0,0},{1,1},{2,1},{3,2};
    for (int i=0; i<cases.length; i++)
        assertEquals(cases[i][1], fib(cases[i][0]));
}

int fib(int n) {
    if (n == 0) return 0;
    if (n <= 2) return 1
    return fib(n-1)+fib(n-2);
}
Test Driven Development - Example

```java
public void testFibonacci() {
    int cases[][] = {{0,0},{1,1},{2,1},{3,2};
    for (int i=0; i<cases.length; i++)
        assertEquals(cases[i][1], fib(cases[i][0]));
}

int fib(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1
    return fib(n-1)+fib(n-2);
}
```

TripCO Architecture - MVP
CM baseline (DTR-99.jar)

DTR-99/src/TripCo.java
DTR-99/src/TestTripCo.java

DTR-99/src/Model/Model.java
DTR-99/src/Model/TestModel.java
DTR-99/src/Model/....java
DTR-99/src/Model/Test....java

DTR-99/src/View/View.java
DTR-99/src/View/TestView.java
DTR-99/src/View/....java
DTR-99/src/View/Test....java

DTR-99/src/Presenter/Presenter.java
DTR-99/src/Presenter/TestPresenter.java
DTR-99/src/Presenter/....java
DTR-99/src/Presenter/Test....java

JUnit example

package DTR-99.Model;
import static org.junit.Assert.*;
import org.junit.Before;
import org.junit.Test;

public class TestModel {
    @Before
    public void initialize() {

    }

    @Test
    public void testConditional() {
        assertEquals(true, conditional);
    }

    @Test
    public void testFunction() {
        assertEquals(false, functionResult);
    }
}
JUnit

- www.junit.org
- https://www.tutorialspoint.com/junit/index.htm