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

Daily Scrum

- Optimizes the probability that the team will meet the goal.
- Development Team is responsible for conducting daily 15 minute meeting held at same time and place.
- Inspect the work done since the last meeting, forecast the work that could be done by the next meeting.
- Scrum Master ensures you have meeting and facilitates.
Daily Scrum Questions

- **What did I do yesterday** that helped the team meet the Sprint Goal?
- **What will I do today** to help the team meet the Sprint Goal?
- **Do I see any impediment** that prevents me or the team from meeting the Sprint Goal?
- **What is your confidence**, on a scale of one to ten, that the team will accomplish the Sprint Goal?

Daily Scrum Report

- Done #23 #34 #11 #16 #15
- Doing #45 #18 #9 #17
- Impediments
  - csv parsing
  - nearest neighbor tour
  - unit tests
- Confidence 5
Class Scrum – Class Confidence

• 10%
• 50%
• 90%

Class Scrum – Class Impediments

• Problem statement, user stories, technical details
• Verification / TDD, JUnit
• Configuration Management / GitHub
• Project Planning, Monitoring & Control / Scrum, Waffle
• Tools (Eclipse, Slack, …)
• Teamwork
Model - nearest neighbor tour

- Pick a starting location. Mark it visited and make it current.
- While there are unvisited locations, choose the unvisited locations closest to the current location. Mark it visited and make it current.
- The tour is completed by returning from the last location to the first location.


Model – Geographic Coordinates

<table>
<thead>
<tr>
<th>Format</th>
<th>Example</th>
<th>DD (decimal degrees)</th>
<th>DD Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMS</td>
<td>179°59'59.99&quot;W</td>
<td>-179.9999972</td>
<td>[\pm D + \frac{M}{60} + \frac{S}{3600}]</td>
</tr>
<tr>
<td>DM</td>
<td>179°59.99'W</td>
<td>-179.9998333</td>
<td>[\pm D + \frac{M}{60}]</td>
</tr>
<tr>
<td>D</td>
<td>179.99°W</td>
<td>-179.99</td>
<td>[\pm D]</td>
</tr>
</tbody>
</table>
View - Map Coordinates

- View translates from geographic coordinates (DD) to SVG coordinates

Great Circle Distance

- Haversine
- Spherical Law of Cosines

http://www.movable-type.co.uk/scripts/latlong.html
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>
</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>
</tr>
</tbody>
</table>

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.
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 - REFACTOR
  - refactor to something you can live with
  - clean code

TripCO Architecture - MVP
TDD for TripCo

```java
package Model;
import org.junit.*;
public class TestModel {
    Model testModel;
    int locations;

    @Before
    public void testPlanTrip() {
        // create a stream with a few cities here
        testModel = new Model();
        locations = testModel.planVisit(testStream);
    }

    @Test
    public void testGetLocationId() {
        // assert for each value is at the expected index;
    }

    // similarly for each location name, latitude, longitude (decimal degrees for last two)
    @Test
    public void testGetLegDistance() {
        // assert each expected distance value exists, no particular order; and total distance
    }

    // similarly for each start location and end location (each # exists once, not on same leg)
    // test the route connections S1-F1/S2-F2/S3-F3/...-FN and FN=S1
}
```