CS314 Software Engineering
Sprint 3

Dave Matthews

Sprint 3 Summary

• Use Level 2 and 3 software engineering processes/tools
  – Clean Code, Coverage, White Box Testing, Code Climate
• Learn some additional technologies
  – SQL (MariaDB)
  – Traveling Salesman Problem
• Add features
  – Produce shorter trips
  – Build trips from existing information
Build process maturity to level 3

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

Internship Plan

<table>
<thead>
<tr>
<th>Sprint</th>
<th>Processes</th>
<th>Tools</th>
<th>Technology</th>
<th>TripCo Epics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>• Configuration Management</td>
<td>GitHub, ZenHub, CodePen, Unix</td>
<td>• Bootstrap 4, HTML, JavaScript</td>
<td>• Make a mobile resume</td>
</tr>
<tr>
<td></td>
<td>• Project Management</td>
<td></td>
<td>• ReactJS</td>
<td>• Calculate geographic distances</td>
</tr>
<tr>
<td></td>
<td>• Scrum, Planning Poker</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>• Continuous Integration</td>
<td>Maven, Travis-CI, Webpack, Node.js, JUnit, IntelliJ</td>
<td>• Java Spark, REST API/HTTP, JSON, SVG</td>
<td>• Accept destination file</td>
</tr>
<tr>
<td></td>
<td>• Test Driven Development</td>
<td></td>
<td></td>
<td>• Show map and itinerary</td>
</tr>
<tr>
<td></td>
<td>• Black Box Testing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>• Clean Code</td>
<td>Code Climate, Emma, Jacoco,...</td>
<td>• SQL, MariaDB</td>
<td>• Plan shorter trips</td>
</tr>
<tr>
<td></td>
<td>• Code Coverage</td>
<td></td>
<td></td>
<td>• Modify destination list</td>
</tr>
<tr>
<td></td>
<td>• White Box Testing</td>
<td></td>
<td></td>
<td>• Show useful information</td>
</tr>
<tr>
<td>4</td>
<td>• Code Smells</td>
<td>KML</td>
<td></td>
<td>• Plan shorter trips</td>
</tr>
<tr>
<td></td>
<td>• Refactoring</td>
<td></td>
<td></td>
<td>• Add more information</td>
</tr>
<tr>
<td>5</td>
<td>• Peer Reviews</td>
<td></td>
<td></td>
<td>• Map operations</td>
</tr>
<tr>
<td></td>
<td>• Inspections</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Metrics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Plan shorter trips</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Plan trips faster</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Finalize your resume</td>
</tr>
</tbody>
</table>
```
# connect to the database from a shell using your eID
mysql -u eID -D cs314 -h faure -p
# show a list of tables
show tables;
# show the columns in a table
show columns from airports;
# count the number of records in the table
select count(*) from airports;
# show the first 5 entries in the airports table
select * from airports limit 5;
# show selected columns
select id,name,municipality from airports limit 20;
# show types
select distinct(type) from airports;
# show municipalities sorted
select distinct(municipality) from airports order by municipality;
```
SQL

select name from airports where type = 'heliport';
# show all of the airports (large, medium, small)
select name from airports where type like '%airport%';

# show all records that refer to denver sorted by name
select id,name,municipality,type from airports where name like '%denver%' or municipality like '%denver%' order by name;

# select airports by ids
select id,name,municipality,type from airports where id in ('19CO', '26CO', '77CO', 'C023', 'C024', 'K00V', 'KFNL', 'KDEN');

// db configuration information
private final static String myDriver = "com.mysql.jdbc.Driver";
private final static String myUrl = "jdbc:mysql://faure.cs.colostate.edu/cs314";
// SQL queries to count the number of records and to retrieve the data
private final static String count = "";
private final static String search = "";
// Arguments contain the username and password for the database
public static void main(String[] args){
    try {
        Class.forName(myDriver);
        // connect to the database and query
        try (Connection conn = DriverManager.getConnection(myUrl, args[0], args[1])){
            Statement stCount = conn.createStatement();
            Statement stQuery = conn.createStatement();
            ResultSet rsCount = stCount.executeQuery(count);
            ResultSet rsQuery = stQuery.executeQuery(search);
            printJSON(rsCount, rsQuery);
        } catch (Exception e){
            System.err.println("Exception: "+e.getMessage());
        }
    }
}
private static void printJSON(ResultSet count, ResultSet query) throws SQLException {

    System.out.printf("\n\n");
    System.out.printf("\"type\": \"find\", \n");
    System.out.printf("\"title\": \"%s\", \n", search);
    System.out.printf("\"places\": [\n");
    // determine the number of results that match the query
    count.next();
    int results = count.getInt(1);
    // iterate through query results and print out the airport codes
    while (query.next()) {
        System.out.printf(" \"%s\"", query.getString("code"));
        if (results == 0)
            System.out.printf("\n");
        else
            System.out.printf(",\n");
    }
    System.out.printf(" ]\n\n");
}

Traveling Salesman Problem

- Find the shortest hamiltonian cycle in a graph.
  - O(n!)
  - heuristic algorithms gain speed at cost of tour quality
  - construction + improvement
- Construction
  - Nearest Neighbor
- Improvement
  - 2 opt
  - 3 opt

Nearest Neighbor

- Does the answer change if I select a different starting city?

```plaintext
nearestNeighbor(cities) {
    1. Select a random city.
    2. Find the nearest unvisited city and go there.
    3. Are there any unvisited cities left? If yes, repeat step 2.
    4. Return to the first city.
}
```

2-opt (from Wikipedia) - very slow

```plaintext
repeat until no improvement is made {
    start_again:
    best_distance = calculateTotalDistance(existing_route)
    for (i = 0; i < number of nodes eligible to be swapped - 1; i++) {
        for (k = i + 1; k < number of nodes eligible to be swapped; k++) {
            new_route = 2optSwap(existing_route, i, k)
            new_distance = calculateTotalDistance(new_route)
            if (new_distance < best_distance) {
                existing_route = new_route
                goto start_again
            }
        }
    }
}
```

```plaintext
2optSwap(route, i, k) {
    1. take route[1] to route[i-1] and add them in order to new_route
    2. take route[i] to route[k] and add them in reverse order to new_route
    3. take route[k+1] to end and add them in order to new_route
    return new_route;
}
```
"ill-advised data structure use"

ArrayList and LinkedList

The ArrayList class and the LinkedList class are concrete implementations of the List interface. Which of the two classes you use depends on your specific needs. If you need to support random access through an index without inserting or removing elements from any place other than the end, ArrayList offers the most efficient collection. If, however, your application requires the insertion or deletion of elements from any place in the list, you should choose LinkedList. A list can grow or shrink dynamically. An array is fixed once it is created. If your application does not require insertion or deletion of elements, the most efficient data structure is the array.

2-opt (improved)

```
2optReverse(route, i1, k) { // reverse in place
    while(i1 < k) {
        temp = route[i1]
        route[i1] = route[k]
        route[k] = temp
        i1++; k--
    }
}
```

```
improvement = true
while improvement {
    improvement = false
    for (i = 0; i <= n-3; i++) { // assert n>4
        for (k = i + 2; k <= n-1; k++) {
            delta = -dis(route,i,i+1)-dis(route,k,k+1)+dis(route,i,k)+dis(route,i+1,k+1)
            if (delta < 0) { //improvement?
                2optReverse(route, i+1, k)
                improvement = true
            }
        }
    }
}
```
2-opt inversions

| route | 23 | 15 | 3 | 9 | 7 | … | 21 | 11 | 5 | 4 | 23 |
| index | 0  | 1  | 2 | 3 | … | n-3 | n-2 | n-1 | n |

```
i  i+1  k  k+1
```

```
i  i+1  -  -  -  -  -  -  -  k  k+1
```

```
i  i+1  -  -  -  -  -  -  -  -  k  k+1
```

```
i  i+1  -  -  -  -  -  -  -  -  -  k  k+1
```

0 <= i < i+1 < k < k+1 <= n

Branding

- logos in tripco repo under sprint3
- color is #1E4D2B
TFFI - version 2

- Versions
- Optimization
- Distance Units
- Find Places
- Errors

TFFI - Version

- "version":0
- "version":"0"

- A number or a string?
TFFI - Optimization

- "none","short","shorter","shortest"
- "0","1","2","3"

- Or, should we support an arbitrary number of optimization levels, determined by the server. The client must query to server to determine this (and possibly other parameters). The client can use a slider or other method to select a value.

TFFI - Distance Units in Options

- "miles"
- "kilometers"
- "nautical miles"
- {"myunits":"12345"}
TFFI - Find Places

type="find",
query="string", // a single word
id=['id1','id2',...], // list of strings
results=[] // list of places

• May specify a query string or a list of ids.
• Results are the same as places.
• Should there be other, optional elements?

TFFI - Error reporting

• HTTP or TFFI?
• Reading
  – IETF RFC7807
  – Choosing an HTTP Status Code
  – REST API Error Codes 101