CS314 Software Engineering Sprint 4 - Worldwide Trips!

Dave Matthews

Colorado State University

Sprint 4 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

Maturity	Organization	Project	Engineering	Support
5	• Organizational Performance Management			• Causal Analysis and Resolution
4	• Organizational Process Performance	• Quantitative Project Management		
3	• *Organizational Process Definition* • *Organizational Process Focus* • *Organizational Training*	• *Integrated Project Management* • *Risk Management*	• **Requirements Development** • **Technical Solution** • **Product Integration** • **Verification** • **Validation**	• *Decision Analysis and Resolution*
2	**Scrum, Zenhub**	• **Requirements Management** • **Project Planning** • **Project Monitoring and Control** • *Supplier Agreement Management*	**GitHub, Maven, Travis, WebPack** **Scrum**	• **Configuration Management** • **Process and Product Quality Assurance** • *Measurement and Analysis*

http://cmmiinstitute.com/sites/default/files/documents/CMMI-DEV_Quick_Ref-2014.pdf

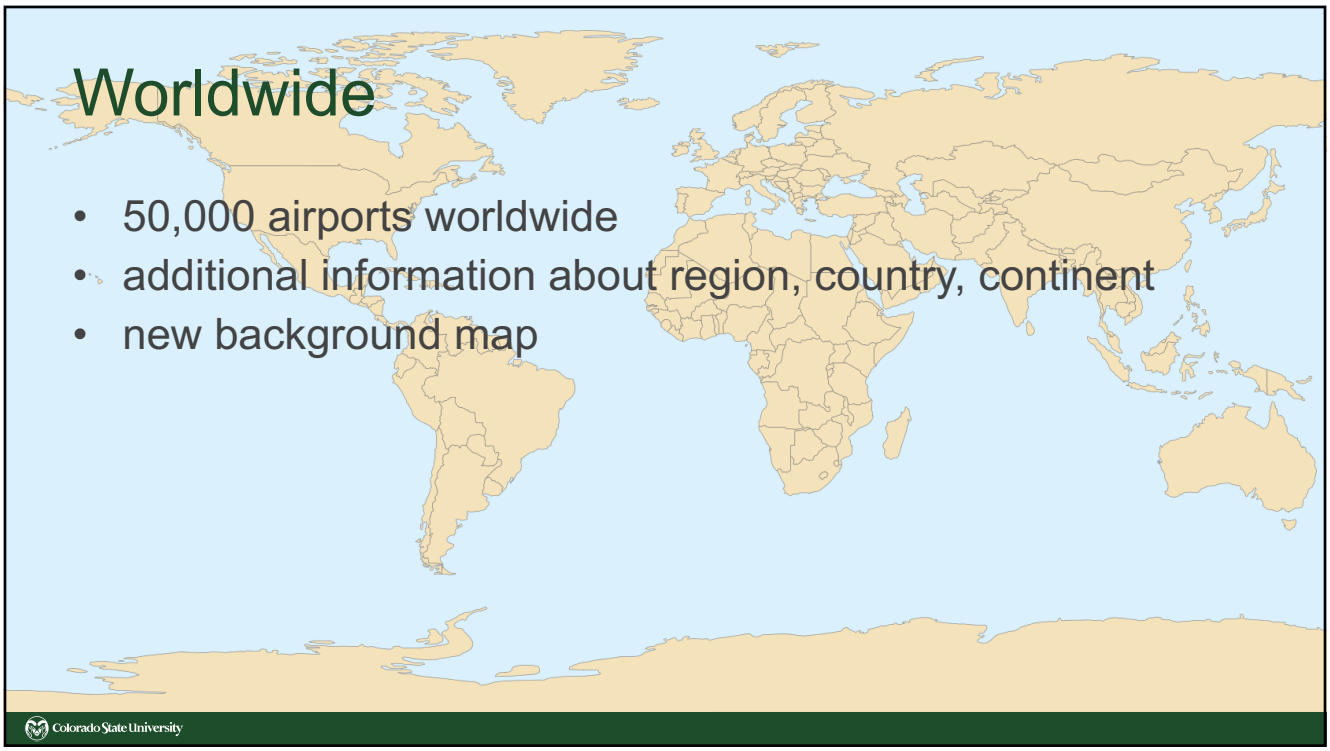
Internship Plan

Sprint	Processes	Tools	Technology	TripCo Epics
1	• **Configuration Management** • **Project Management** • **Scrum, Planning Poker**	• **GitHub, ZenHub** • **CodePen** • **Unix**	• **Bootstrap 4** • **HTML** • **JavaScript** • **ReactJS**	• **Make a mobile resume** • **Calculate geographic distances**
2	• **Continuous Integration** • **Test Driven Development** • **Black Box Testing**	• **Maven, Travis-CI** • **Webpack, Node.js** • **JUnit, IntelliJ**	• **Java Spark** • **REST API/HTTP** • **JSON, SVG**	• **Accept destination file** • **Show map and itinerary**
3	• **Clean Code** • **Code Coverage** • **White Box Testing**	• **Code Climate** • **Emma, Jacoco, …**	• **nearest neighbor** • **SQL Select**	• **Plan shorter trips** • **Modify destination list** • **Show useful information**
4	• **System Testing**	• **Jest, …**	• **2 opt** • **SQL joins** • **KML**	• **Plan shorter trips** • **Worldwide** • **Interactive map**
5	• Peer Reviews • Inspections • Metrics		• Concurrency	• Plan shorter trips • Plan trips faster • Finalize your resume

Sprint 4 Epics

- Distance units and configuration (TFFI)
- Optimization (none, NN, 2-opt)
- Worldwide (database, more tables, map)
- Filtered searches / configuration (TFFI)
- Zoom/pan map (interactive map vs image) (TFFI)
- Branding and improved user experience

Worldwide



- 50,000 airports worldwide
- additional information about region, country, continent
- new background map

SQL (https://mariadb.org/learn/)

```
# connect to the database from a shell using your eID
mysql -u eID -D cs314 -h faure -p
# once in SQL…
# see a list of tables
show tables;
# see the columns in a table (world replaces airports)
show columns from airports;
show columns from regions;
show columns from countries;
show columns from continents;
# notice the column similarities between tables
```


SQL

```
SET @phrase="%denver%";
SET @phrase="%london%";
SET @phrase="%alf%";
# search query, more columns for  plan query
SELECT airports.name, airports.municipality, regions.name, countries.name,
continents.name
FROM continents
INNER JOIN countries ON continents.code = countries.continent
INNER JOIN regions ON countries.code = regions.iso_country
INNER JOIN airports ON regions.code = airports.iso_region
WHERE countries.name LIKE @phrase
OR regions.name LIKE @phrase
OR airports.name LIKE @phrase
OR airports.municipality LIKE @phrase
ORDER BY continents.name, countries.name, regions.name, airports.municipality,
airports.name ASC
LIMIT 20;
```

TFFI

- supported units
- optimization labels/description
- supported map format
- places attributes
- query filter columns/values
- error response

TFFI supported units

- config
  - "distances": ["miles", "kilometers", "nautical miles", "user defined"]
- no caps so you can use it in a sentence.

TFFI optimization labels/descriptions

- config response
  - "optimization":3,
    "optimizations": [
      {"label":"nn", "description":"Nearest Neighbor…"} ,
      {"label":"2opt", "description":"2-opt…"} ,
      {"label":"3opt", "description":"3-opt…"}
    ]
- optimizations is optional

TFFI map format

- config reponse
  - "maps":["svg","kml"]
- trip request (in options)
  - "map":"svg"
  - "map":"kml"
- config only return supported formats
- the map attribute is optional, svg is assumed

TFFI query filter columns/values

- config response
  - return a list of columns and values that can be filtered
  - "filters":[…]
- query request
  - specify a list of columns and values to filter
  - "filters":[…]

TFFI query filter object

```
{
  "attribute" : "type",
  "values"    : [ "heliport", "balloonport", …]
}
```

- A null or [] for the values implies there is no filter, which is equivalent to matching all values.

TFFI error response

```
{
  "type" : "error",
  "version" : "version",
  "code" : "",
  "message" : "",
  "debug" : ""
}
```


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, cheapest link, …
- Improvement
  - 2-opt, 3-opt, k-opt, LK, tabu, SA, GA, …

https://web.tuke.sk/fei-cit/butka/hop/htsp.pdf

"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.

18

Colorado State University

306,617 km with nearest neighbor

Colorado State University

Tour Construction

- Nearest Neighbor, cheapest link, others?
- Does the answer change if I select a different starting city?

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


https://web.tuke.sk/fei-cit/butka/hop/htsp.pdf

271,618 km with 2-opt

2-opt (from Wikipedia) - very slow

```
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
                  }
          }
      }
 }

 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;
 }
```


https://en.wikipedia.org/wiki/2-opt

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

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

264,376 km with 3-opt

3-opt inversions and swaps

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

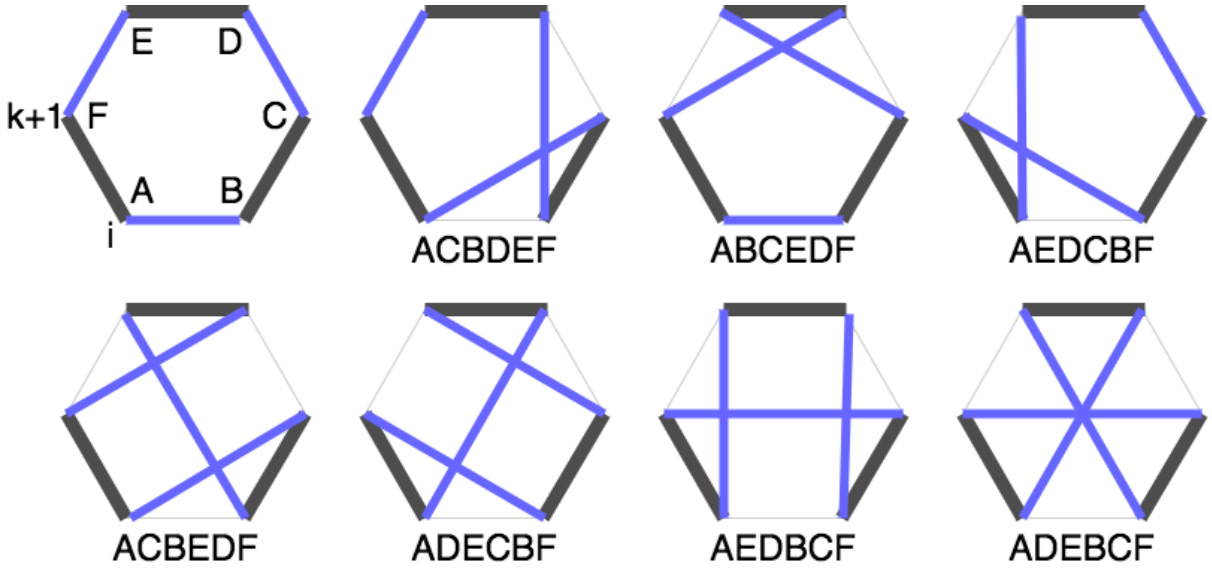
i [i+1 j] [j+1 k] k+1

i [i+1 j] [j+1 k] k+1

i [i+1 … j] [j+1 … k] k+1

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

3-opt cases (including 2 opt)



3-opt inversions, swaps, first/best

original	**i**	**i+1**	**-->**	**j**	**j+1**	**-->**	**k**	**k+1**
2-opt	**i**	**j**	**<--**	**i+1**	**j+1**	**-->**	**k**	**k+1**
2-opt	**i**	**i+1**	**-->**	**j**	**k**	**<--**	**j+1**	**k+1**
2-opt	**i**	**k**	**<--**	**j+1**	**j**	**<--**	**i+1**	**k+1**
3-opt	**i**	**j**	**<--**	**i+1**	**k**	**<--**	**j+1**	**k+1**
3-opt	**i**	**k**	**<--**	**j+1**	**i+1**	**-->**	**j**	**k+1**
3-opt	**i**	**j+1**	**-->**	**k**	**j**	**<--**	**i+1**	**k+1**
3-opt	**i**	**j+1**	**-->**	**k**	**i+1**	**-->**	**j**	**k+1**
distances	**(i,**	**?)**		**(?,**	**?)**		**(?,**	**k+1)**