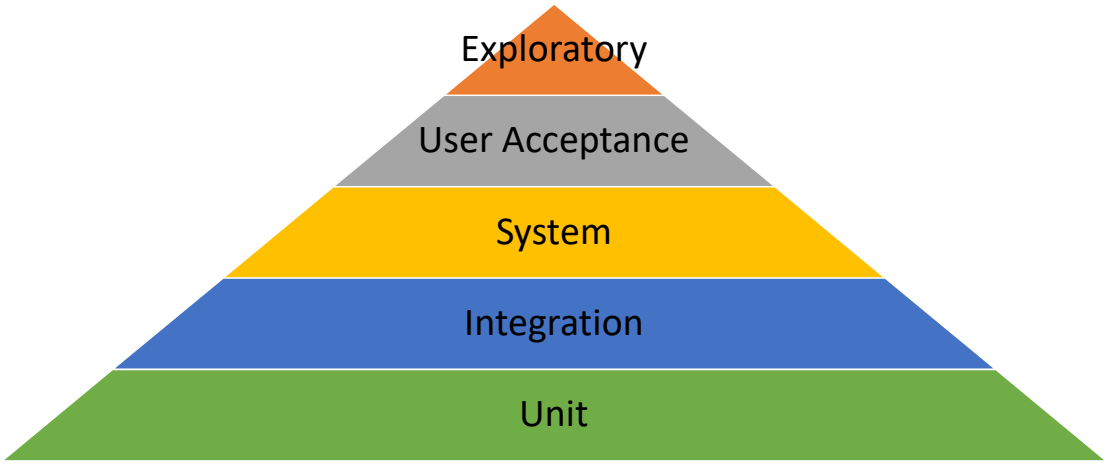
Software Testing
Unit Testing

CSCI 3300/5300

Software Testing Pyramid



Unit Tests

- Simplest form of test
- Focuses on each 'unit' (class, file, function) individually
- Tests the behavior of each unit as per that unit's specification
  - Need to know expected output of each function
- What if a class is complicated and is tightly coupled with other classes?

Example: how would you unit test this?

```
class InputThread
{
   public InputThread(Timer timer, Buffer buffer){//initialize variables}
   public void placeMessages()
   {
       while (numIntervals < SOME_VALUE)
       {
          timer.wait(interval);
          int numMessages = dataInterface.getRate();
          buffer.placeMessages(numMessages, timer.timeNow());
       }
   }
}
```

Possible Strategies

Advantages of Unit Testing

- Forces a simpler design
- Finds errors at the source
- Finding errors closer to the time when they were made saves time in the long term

Constructing Unit Tests: Equivalence Class Partitioning

- Inputs to the system are divided into groups that are expected to exhibit similar behavior
- Select one input from each group to test the behavior
- Consider valid and invalid classes
- Example: https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html#2
- What are the equivalence classes for the triangle testing challenge from last time?

Constructing Unit Tests: Boundary Value Analysis

- Testing between extreme ends or boundaries between partitions of the input values:
  - Start-End
  - Lower-Upper
  - Minimum-Maximum
- For each equivalence class, select a value at
  - Minimum
  - Just above the minimum
  - A nominal value
  - Just below the maximum
  - Maximum
- What are the boundary values for the triangle testing challenge?

Run unit tests with each build

- Frameworks can help automate unit tests:
  - gtest – Google test framework for C++
  - unittes – python unit test framework
  - JUnit – Java unit test framework
- Example from my repo

Isolating issues and Adding Unit tests

- Bugs may be discovered at any time (not just during unit testing)
- Try to isolate the issue in a unit test environment
- Build a test that replicates the issue
  - Will help you debug the problem
  - Will add to regression testing
- Fix the problem and rerun unit test
- Example: clock test