Jan. 11

- **Static techniques**
    - Find faults
    - Examples (compilers/linters):
        - Type checking
        - Dead code analysis
    - Code inspection – for functionality & style
    - Program verification
    - Trade-off: exhaustive, but subject to false positives
- **Dynamic techniques**
    - Observe failures
    - Must generate inputs & know the expected outputs
    - Easy to run the program
- Big questions:
    - "When should I start testing?"
        - As soon as possible
    - "When should I stop testing?"
        - When I run out of time
            - For automatic input generation, or open-ended exploratory testing
        - When I'm close enough to being exhaustive
        - Aka. explored enough (all) of:
            - Behaviours/use cases
            - Program states
            - Inputs
            - Statements/branches
- **Coverage**: find a reduced input space & cover it with test cases
- **Test requirement**: an element of an artefact that some test case must satisfy
    - Examples:
    - Have a TR for every method in a class

Jan. 13

- **Infeasible test requirements**
    - E.g. unreachable code
- **Coverage level**
    - Given a set of test requirements TR & a test set T, the coverage level is the ratio of (the # of TRs satisfied by T)/(the size of TR)
- **Exploratory testing**

    - Usually carried out by testers
    - Generally unscripted
    - Simultaneously learning, test design, and test execution
    - Good for:
        - Simulating actual use cases (realism)
            - i.e. diversifying testing beyond scripted cases
        - Finding single most important bug in the shortest time
        - Being less siloed
        - Evaluating a particular risk – see if scripted tests are needed
- Exploratory testing process
    - Start with a goal/charter
        - E.g. explore the product elements
    - Decide which area of the software to test
    - Design a test (informally)
    - Execute test & log bugs
    - Repeat as needed (back to step 2)
    - Don't produce exhaustive notes
    - Output should be:
        - Set of bug reports
        - Test notes
        - Artefacts (inputs, outputs)
- Example with WaterlooWorks:
    - Charter: evaluate the functionality of WW
    - First task: what functionalities to test?
        - Search for/view job rankings
        - Interview slots are single-booked
        - View scheduled interviews
        - Rank potential offers
        - Permissions (e.g. students can't access employer section)
        - Mobile view