Legacy Code and Refactoring

Characterization Tests

- Establish ground truth about how the app works today as a basis for coverage
  - Makes known behaviors Repeatable
  - Increase confidence that you're not breaking anything
- When you don't have tests and don't understand the code
- Do not try to make improvements at this stage

Integration-Level Characterization Tests

- Create the missing scenarios
- Watch / interview users while they use app to reproduce their workflows
- Do imperative/verbose scenarios now, improve on them later

Unit / Module Characterization Tests

- Cheat: write tests to learn as you go
- Use the test to poke at the code and see what happens
  - Should initially return an error of some type
- Update expectation to reflect the return value

Comments and Commits

- Comments tell what's not obvious about the code rather than repeating what's obvious
  - Example: code invariants, subtle problems that required unusual implementation, bug workarounds, strange corner cases
- If the developer needs to know this info while working on the code, put it in a comment

Code Smells

- SOFA captures symptoms that often indicate code smells in a single method
  - Short
  - One thing (method only does one thing)
  - Few arguments
  - Abstraction is consistent (either what is to be done or how to do it)
- Single Level of Abstraction
  - Complex tasks $\Longrightarrow$ should be doing divide and conquer
  - For methods, top level method outlines high-level steps and delegates detail to helper methods
- Why lots of args are bad
  - Hard to get good testing
  - Hard to mock / stub while testing
  - Boolean arguments should be a yellow flag (usually should be two diff functions)
  - Similar groups of args can be extracted into a class

Quantitative Code Complexity

- Look for hotspots $\triangleq$ multiple metrics throw red flags
- `metric_fu` gem gives metrics
- Take metrics with a grain of salt

ABC Complexity

- Counts assignments (A), branches (B), and conditions (C)
- Score = $\sqrt{A^2 + B^2 + C^2}$

- Ideally ≤ 20 per method

Cyclomatic Complexity

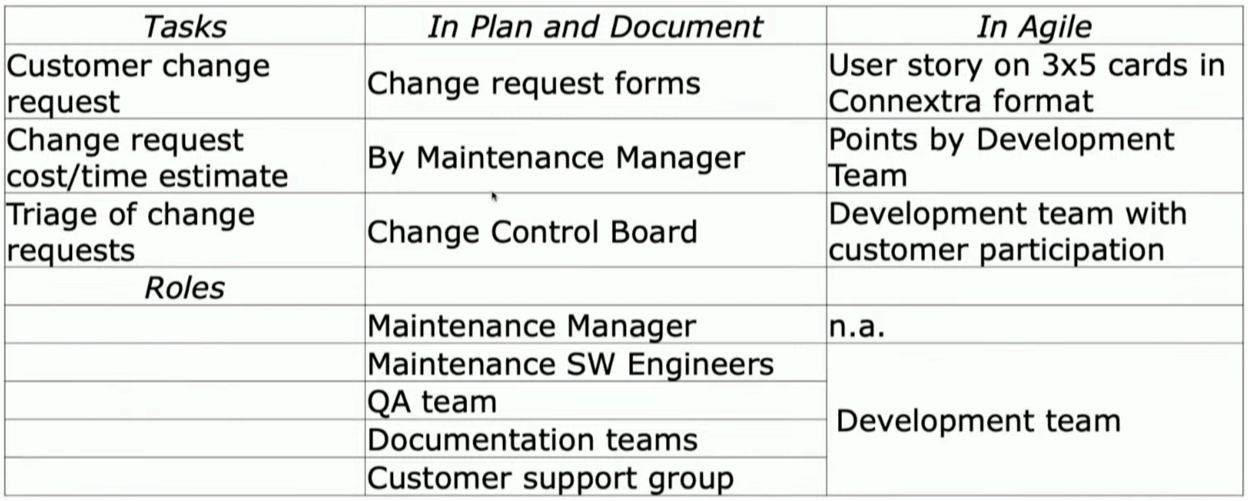
- How many paths there are through a block of code
- $E - N + 2P$
  - E := edges
  - N := nodes
  - P := connected components

Method-Level Refactoring

- Take small steps
- Transform code to get rid of smells in steps
- Protect each step with tests
- Can increase lines of code but still reduce complexity
- Goals: get rid of code smells, Improve testability, reduce complexity
- Side effects: Eliminate bugs
- Should cause existing tests to fail
- Should result in necessary updates to test suite

P&D Perspective on Software Maintenance

- Customers may pay software maintenance fee
- Development team is likely not the same as the original engineers
- Can have a separate maintenance manager
  - Like development manager
  - Estimate costs, maintain schedule, evaluate risks and overcome them
  - Recruits team
  - Document project maintenance plan
- Process
  1. Working in SW field $\implies$ new releases can't break features
  2. Customer collaboration $\triangleq$ work with customer to improve in next release vs meet contract spec
  3. Responding to change $\triangleq$ customer sends change requests; manager evaluates them
     - Change request forms have ticket tracking
- Change Control Board
  - Estimate cost/time per CR
  - QA team gives cost of testing for change request, including regression testing + new tests
  - Documentation teams gives cost of updating docs
  - Customer support group decides if urgent or workaround
  - Urgent $\implies$ code needs to be fixed ASAP without other policies (customer-facing bug, security, competitor's features)
- Emergencies can cascade and not have time to catch up
- Refactor time not usually built in so may not happen
  - Can sometimes run into issues where re-engineering is necessary
  - Builds up a lot of technical debt
-



Tasks	*In Plan and Document*	*In Agile*
Customer change request	Change request forms	User story on 3x5 cards in Connextra format
Change request cost/time estimate	By Maintenance Manager	Points by Development Team
Triage of change requests	Change Control Board	Development team with customer participation
Roles		
	Maintenance Manager	n.a.
	Maintenance SW Engineers	Development team
	QA team	
	Documentation teams	
	Customer support group	

Figure 1: Screenshot_2023-11-27_at_10.29.15_PM.png