SWE 265P
Reverse Engineering and Modeling

Lecture 9

Reality

“Reverse engineering test cases – there is a lot there.” – Leyna Cotran [Staff systems engineer, Lyft]

Reality

"When you are ready to dig into more low-level details, check existing unit/integration tests. Run them to verify your own understanding of each part of the system. If there are no tests, try to write them yourself, you will do a public good and it will increase your understanding of how differently pieces work independently and together. By all means, make sure there is enough test coverage before you modify any of the code! Otherwise there may be unintended consequences." – Alegria Baquero [Senior software engineer, Zocdoc]

Reality

“ […] add asserts()s to a codebase that doesn’t have them. If you can do that successfully (and run through all the code tests with asserts on) then you’ve demonstrated you’ve learned something about the codebase.” – Eric Dashofy [Deputy CIO and general manager, The Aerospace Corporation]

Reality

"Often I use a debugger to understand information flow and a diagramming tool to help me build a mental model of the system." – Lee Martie [Research staff member, MIT-IBM Watson AI Lab]

Today

- Last week’s material
- Key expert practices
- Leveraging tests
- In-class practice
- Eric Dashofy & Michael Gorlick (The Aerospace Corporation)

Last week's material

- Key expert practices
  - invest now to save effort later
  - socially embed and reinforce good practice
  - use analogy
- Software design patterns
- Any questions?

Last week’s homework – design patterns

- How difficult was it to find five different patterns in your system?
- What approach(es) did you use to do so?
- What did you learn about your system?
- Any questions?

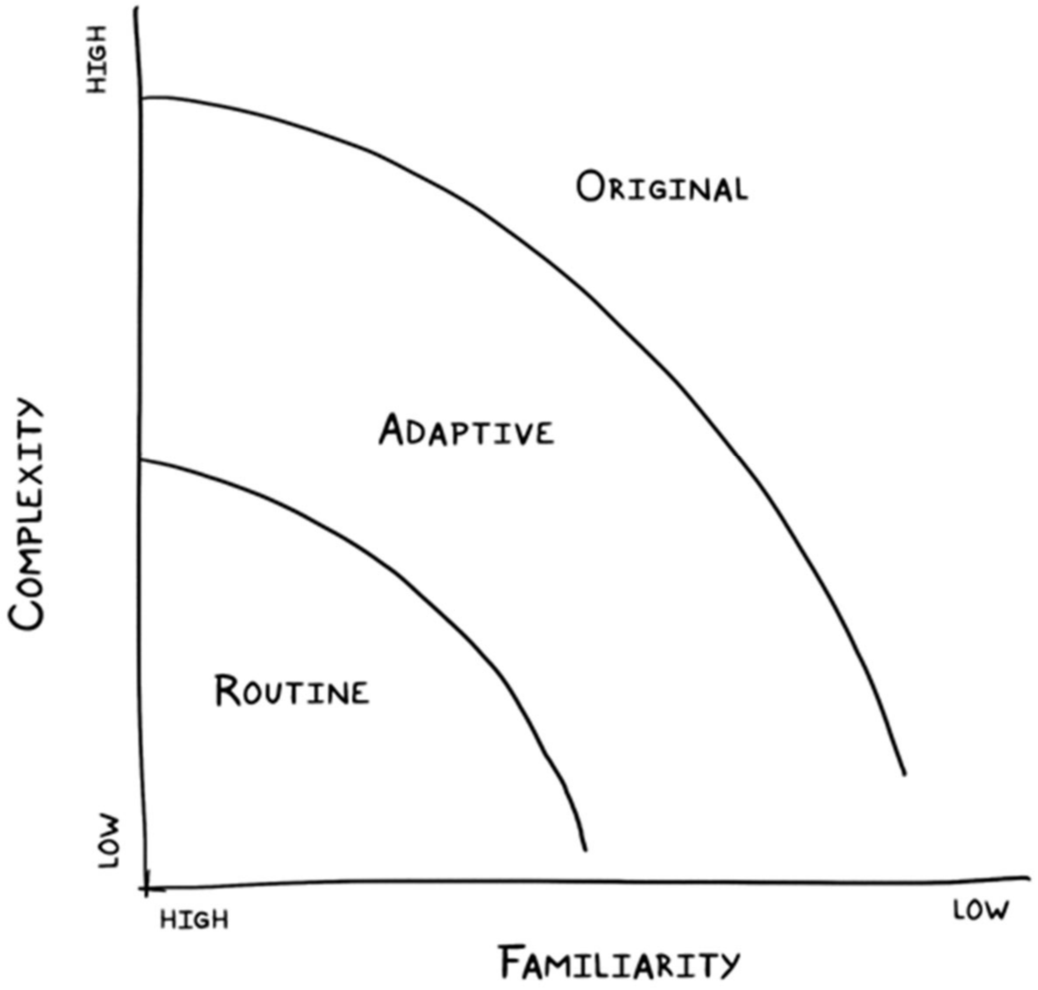
Last week's homework – pull request

- What change did your pull request contain?
- How difficult was it to submit the pull request?
- Did you get a response from the maintainers?
- What did you learn about your system?
- Any questions?

KEP #16: play the fool

KEP #17: are alert to evidence that challenges their theory

KEP #18: adjust to the degree of uncertainty present



Testing

- Unit testing
- Integration testing
- End-to-end testing

Testing as a mechanism for code understanding

- Reading test cases
- Running test cases
- Writing test cases

Complementary tactics

- Print statements
- Assertions
- Debugger

Let’s practice

- Open JPacMan4
- Study the test cases in the board folder
- Write down what you are learning on the whiteboard
- Write down open questions that remain or arise on the whiteboard

Let’s practice

- Now study the test case LevelTest in the level folder
- Write down what you are learning on the whiteboard
- Write down open questions that remain or arise on the whiteboard

Let’s practice

- Now study the test case NavigationTest in the npc.ghost folder
- Write down what you are learning on the whiteboard
- Write down open questions that remain or arise on the whiteboard

Let’s practice

- Now study the test case StartUpSystemTest in the integration folder
- Write down what you are learning on the whiteboard
- Write down open questions that remain or arise on the whiteboard

Let’s practice

- Now study the test case StartUpTest in the e2e.framework.start folder
- Write down what you are learning on the whiteboard
- Write down open questions that remain or arise on the whiteboard

Reflection

- How does this compare to reading the code base first?
- We haven’t run a single test yet
- We haven’t looked at asserts in the code yet
- Where would you like to add tests? Why?

Further reading

- https://en.wikipedia.org/wiki/Exploratory_testing
- https://www.scalyr.com/blog/the-10-commandments-of-logging/
- https://coralogix.com/log-analytics-blog/java-logging-right/
- https://logz.io/blog/logging-best-practices/
- https://nickjanetakis.com/blog/using-print-statements-are-a-handy-way-to-debug-and-explore-code

Further reading

- https://www.drdobbs.com/architecture-and-design/embedded-print-statements-debugging/240148855

- https://www.jotform.com/blog/how-to-become-a-good-debugger/

Homework (team)

- With your team, code up your second issue and submit a pull request
- Write up a brief report about the experience, and particularly what new understandings you gained about your system
- (Due date: 17th at noon)

Homework (team)

- With your team, study at least three existing, interesting test cases for your system
- Write up a brief report about the experience, and particularly what new understandings you gained about your system
- (Due date: 17th at noon)

Homework (team)

- With your team, develop at least three new test cases for your system
  - may cover your new functionality, in which case submit the test cases as part of your pull request with the code changes
  - may cover existing functionality, in which case submit the test cases as part of a new pull request
- Write up a brief report about the experience, and particularly what new understandings you gained about your system
- (Due date: 17th at noon)

Homework (individual)

- Make sure to regularly update your personal diary, including an entry for today’s lecture

Break

And now…

- …welcome Eric and Michael!