Intro to Methods of Software Engineering Fall 2018

Lecture 6 — October 11, 2018

Patrick Lam *version 1*

Feature Flags

Also known as *feature toggles*. Lots of information at `https://martinfowler.com/articles/feature-toggles.html`.

Objectives for this lecture:

- be able to apply feature flags to code that you've written.
- understand need for feature flags in trunk-based development (the way that you'll use Git).

Situation

Recall the Ideas Clinic exercise last week where you coded a spaceship simulation. One of the subsystems was the *navigation* system.

- First iteration: hard-code which warp gate to take.
- Second iteration: implement Dijkstra's shortest path algorithm.

Let's think about progressing from the first iteration to the second iteration. You had code for hard-coding decisions, which works in some universes. But now you want to start working on Dijkstra's algorithm.

For Hero Programmers. Well, what if the navigation squad works only on their own computer? What did you find to be the advantages and disadvantages of doing it that way?

Beyond Heroes. What you really wanted was a way to have both versions available for other squads. They should be able to use the hard-coded version to see whether their own code works in the first universe; and they should be able to try out Dijkstra's algorithm as you develop it.

Three Options

Let's say that the navigation squad wants to commit their work to version control. Here are three things they could do:

1. comment out one solution at a time.
2. use revision control to switch back and forth between versions.
3. use a boolean flag and conditional branch on the flag's value.

Are these options good engineering? Why or why not—what are the advantages and disadvantages? Think about whether they are fit for purpose and fit for future.

Implementing Feature Flags

Solution (3) above didn't talk about where you read the boolean flag from. Here are three options. Let's say that we started with this base code:

```
1  int navigate() {
2    // hard-coded navigation implementation
3    return 1;
4  }
```

Global constant/variable. This option is going to be enough for you until you start thinking about writing more sophisticated programs, for instance in CS 247 in 2B. It looks like this:

```
1  // feature flag
2  bool use_dijkstra = 1;
3
4  // move the old implementation here
5  int navigate_hardcoded() {
6    return 17;
7  }
8
9  // new implementation goes here
10 int navigate_dijkstra() {
11   // ...
12 }
13
14 int navigate() {
15   // switch on the feature toggle
16   if (use_dijkstra) {
17     return navigate_dijkstra();
18   } else {
19     return navigate_hardcoded();
20   }
21 }
```

Global flags array. The next step up is to use an array. This is more flexible.

```
#define NAV_DIJKSTRA 0
#define LAST_FLAG 1
bool feature_flags[LAST_FLAG];

// don't forget to call this early from main()!
void initialize_feature_flags() {
  feature_flags[NAV_DIJKSTRA] = 0;
}

// same navigate_hardcoded() and navigate_dijkstra() implementations

int navigate() {
  if (feature_flags[NAV_DIJKSTRA]) {
    return navigate_dijkstra();
  } else {
    return navigate_hardcoded();
  }
}
```

More indirection. You can replace the statement

```
  if (feature_flags[NAV_DIJKSTRA]) {
```

with a function call:

```
  if (is_feature_flag_set(NAV_DIJKSTRA)) {
```

and then swap out the implementation of `is_feature_flag_set` with something arbitrarily complicated, e.g. a read from a configuration file or a database.

More advanced topics

I'll briefly allude to two more advanced topics here.

We're going to talk about unit testing in the very near future. You can write test cases that exercise both values of the feature flags: the old test cases check the hard-coded navigation algorithm, while the new test cases check the Dijkstra's algorithm.

There is a thing called A/B testing. Websites use it all the time. The idea is to collect empirical data about, say, which "Buy" button users are more likely to click on. Feature flags are one way to implement A/B testing: you can have a flag which determines whether the "Buy" button is red or green. [It'll be red, but it's good to run the experiment.]