Joshua Boe

March 5, 2020

The Juicerz

Cyclomatic Complexity

Metric Data:

I used the tools CodeMetrics, an extension for Visual Studio Code, and Radon, a Python tool, to gather the below data from our javascript files and Lambda functions. The prefix “React” refers to React Native functions, “Lambda” refers to functions written in AWS Lambda, and “BackendServer” refers to functions used for logging charge data from NeoCharge devices.

BackendServer:

```
primary_listener.py
    M 52:4 PrimaryListener.poll_fds - A
    C 3:0 PrimaryListener - A
    M 23:4 PrimaryListener.on_connect - A
    M 29:4 PrimaryListener.on_message - A
    M 40:4 PrimaryListener.spawn_new_child - A
    M 62:4 PrimaryListener.subscribe - A
    M 5:4 PrimaryListener.__init__ - A
    M 16:4 PrimaryListener.get_uuid_from_log - A
    M 20:4 PrimaryListener.on_log - A
    M 37:4 PrimaryListener.on_disconnect - A
Publisher.py
    F 7:0 on_connect - A
    F 13:0 publishFromFile - A
    F 4:0 on_log - A
s3_logger.py
    F 16:0 upload_admin_log - A
    F 8:0 fileDoesExist - A
    F 43:0 send_push_notification - A
temp.py
    F 13:0 upload_log - A
    F 6:0 fileDoesExist - A
test_log.py
    C 3:0 TestServerMethods - A
    M 5:4 TestServerMethods.test_admin_single_log - A
test_log_pool.py
    F 4:0 test_admin_log - A

21 blocks (classes, functions, methods) analyzed.
Average complexity: A (2.0)
```

CC score	Rank	Risk
1 - 5	A	low - simple block
6 - 10	B	low - well structured and stable block
11 - 20	C	moderate - slightly complex block
21 - 30	D	more than moderate - more complex block
31 - 40	E	high - complex block, alarming
41+	F	very high - error-prone, unstable block

React and Lambda:

Method	CCN
React.screens.AuthLoadingScreen._bootstrapAsync()	10
React.screens.SignUpScreen.SignUp()	8
React.screens.SignUpScreen.checkValidInput()	8
React.screens.SignUpScreen.handleErrors()	9
React.screens.SettingsScreen.SignOut()	6
React.screens.SettingsScreen.render()	40
React.screens.SchedulingHomeScreen.render()	21
React.components.Dashboard.render()	20
React.screens.VerificationScreen.render()	18
React.screens.SignUpScreen.render()	15
React.screens.HomeScreen.render()	15
React.screens.SignInScreen.render()	14
React.screens.AuthLoadingScreen.render()	6
Lambda.sendPushNotification.lambda_handler()	3
Lambda.sendPushNotification.send_push_message()	3
Lambda.addusers-test.handler()	2
Lambda.registerPushNotification.lambda_handler()	1

Analysis:

We are in a similar situation regarding cyclomatic complexity that we were in during the first metrics posting but with some improvements. In the sprints since Metrics1, all team members were made aware of cyclomatic complexity concerns, and I believe that has had an influence in reducing the complexity of the new code we have written since then. Also, a few of the React functions that were overly complex in the first posting have been simplified through refactorings.

As before, many of our old React render() functions are still very complex. The team has discussed this concern, but we are currently very busy with new features. It is difficult to find the time to decompose those components into sub-components, but we hope this is something we can improve upon in the future. Perhaps we will have time in CSC 406 to improve our code before handing the finished product off to our customers. However, with our current limited time and large backlog of features, we must prioritize our time appropriately, and we do not believe reducing the complexity of our render() functions is of high-priority.