CIS 4150 Assignment 3 Part 1

By Ian McKechnie

**Q1) Explain function point analysis and design a situational example scenario in which performing function point analysis can be helpful.**

Function point analysis (FPA) is a standardized methodology for measuring various functions of software from the user’s perspective. FPA has been standardized by the *International Function Point Users Group.* It is based on the idea that the size of the software should be measured according to the functionalities specified by the user. The size of an application is measured in function points.

The process to calculate function points are as follows:

1. Determine the type of the project that the function point count is to be calculated.
2. Identify the counting scope and the application boundary
3. Identify data functions (internal functions and external interface files) and their complexity’
4. Identify transactional functions (External inputs, outputs, and queries) and their complexity
5. Determine the unadjusted function point found
6. Calculate ethe adjusted function point count.

An example of a situation where this would be helpful

The software in a self-checkout kiosk being developed would be excellent. This interacts with many different external and internal components. It has functions to handle adding items to cart (step 3) and transactional function from step 4(billing to visa and Mastercard, accessing external DB of all products). This would allow you calculate function points from every category.

**Q2) Explain the benefits of designing drivers and stubs, and can you create an example in python code to design test drivers and stubs for any example of your choice?**

Drivers and stubs are great for testing functions that require other components to be tested. So, if I have a function I want to test that calls other functions that are not yet developed. I would create stubs that go in their place to return hard coded data so the function being tested could run correctly. The benefit of using drivers is you can bottom-up test code. You can develop the lowest level of functions and use drivers to call all of them and with hard coded arguments and make sure what they return is correct and what is expected. The benefit is you can bottom-up code and still effectively test everything.

Stub and Driver Example:

# ------ Stub Example ------

# <Function being tested>

def function(a, b):

c = complexMathPerformed(a)

d = moreComplexMathPerformed(b)

return c + d

# </Function being tested>

# <stubs>

def complexMathPerformed(a):

return 0

def moreComplexMathPerformed(b):

return 0

# </stubs>

# <test>

ans = function(1, 2)

assert ans == 0

print("Test passed")

# </test>

# ------ Driver Example ------

def lowLevelFunction(a):

return a + 1

def lowLevelFunction2(b):

return b + 2

def lowLevelFunction3(c):

return c + 3

# <Driver>

def driverFunction(a, b, c):

d = lowLevelFunction(a)

e = lowLevelFunction2(b)

f = lowLevelFunction3(c)

return d + e + f

# </Driver>

# <test>

ans = driverFunction(1, 2, 3)

assert ans == 12

print("Test passed")