CSE464: Software Quality Assurance and Testing
Homework 1: Due 9/8/2019
Jacob Peters

1) Software Quality
    a)

Functional Suitability:

How much so the product meets the characteristics needed and to what degree of how well the product performs the tasks completely.

Performance Efficiency:

How well the product performs when put against the amount of resources used: processing time, amount of resources used and type, and the max limit of the product.

Compatibility:

To what degree the product can interact and exchange information with other products.

Usability:

How well a new user can learn to use and continuously use a product after learning

Reliability:

The ability of the product to perform the actions specified without fail for the allotted amount of time under different situations of aging or errors.

Security:

The ability to keep data separate from users based on the clearance level they have for the account they are on.

Maintainability:

How well a product can be edited for improvement or fixed.

Portability:

How well the product can be used on other software and hardware systems without the need of the user going through extra steps each time.

    b)
        i)

Quality Factor: Corresponding Categories

Time Behavior: Performance Efficiency
Adaptability: Portability, Reliability
Recoverability: Reliability, Usability
Interoperability: Compatibility
Data Integrity: Security
Learnability: Usability
Useful help Features: Usability
Availability: Reliability:
Modifiability: Maintainability
Confidentiality:Security

        ii)

Time Behavior:

Test the response time from the server for a single book purchase from an empty queue, then the time required for the server to process purchases from much larger queues across different users and situations.

Confidentiality:

Along the pathway the users use to purchase books from the making an account to the payment and shipping address determine at which points any information is being called from other scripts and if there's a point where the data is not protected by the site.

Recoverability:

Best way to test would probably be to force errors along the pathway from account creation to purchase/confirmation and see if any of the cases result in a return of an undesirable state or state other than where the user left off.

2) Perspective on Software Testing
    1. The overlap of all three. Want to maximize this area to cover as many circumstances as possible.
    2. Unknown since not tested. Possibly resulting in bugs
    3. Results from extra features not specified. Extra functionality to the program
    4. The expected and tested outcome is not working, the functionality is missing
    5. Essentially what you want to be present in the code, no way of knowing though whether the functionality is there or not.
    6. Just the observed outcomes. Not knowing the desired product results in a good chance that some are bugs.
    7. Tested unspecified behaviors not resulting in an error so this area is good.
    8. All other defects or possible outcomes that result from the code.

    a. Kind of:

        Yes in that the principle of the Specification black box testing allows the tester to determine the way in which the code is working and whether that be correctly or incorrectly and then the code-based structural testing allows for the outputs of the black box to be plotted and helps to determine just how well the software system is performing compared to where the desired outcome is.

        No in that there can be underlying issues in the code that will not appear in the output of the program, but might be eating away at memory or something else the whole time.

3)

a)

Exploratory Testing is based around designing new tests as you go and learning as you go to find out what else you need to test.

Exploratory testing focuses around the idea that you need to keep coming up with ideas through learning, designing , execution, and interpretation whereas the counter to that, Scripted Testing, which is just a set of instructions on how to test the program and the expected results from the program. Scripted testing is a very

machine-like process and can for the most part be automated. The benefit behind the use of Scripted testing is that in many cases the well made scripts that are used for a task can be used for another similar task.
In exploratory testing the idea of making a test falls on the tester who just thinks it up and runs it, but in scripted the tests come out as a machine that inputs then reads a specific of the output hoping it's correct. In this situation of scripted testing the important parts along the way such as time needed, memory usage or leakage, and any unshown undesired variables are not accounted for and simply neglected. The script will always miss the same things every time if it is not made to check it from the start.
Using the Exploratory Testing method you can help identify weaknesses that arise from the programmers and how those weaknesses might show up in the code to help better both the code and the coder.

b)

1. Printing File
    a. Give options for number of pages and printing style
    b. All accessible printers are shown
    c. Print in color or black and white
2. Save File
    a. If new file, save as instead
    b. Overwrite the document instead of making a new one
    c. Save to the location the file was opened from
3. New Document
    a. Do you want to save the previous document if unsaved?
    b. Document type and page style.
    c. Name of new document
4. Bullet Points/Numbered List
    a. Remember list type after a shift+enter
    b. When you exit list and a marker from the list is use continue the list
    c. Add emphasis to list characters if first word on the line is emphasized
5. Undo/Redo
    a. Test how far back is remembered in the queue
    b. When do actions leave the queue?
    c. What counts as an action in the queue versus multiple actions?
6. Insert Image
    a. Does the image compress or lose any data on insert
    b. Does the image compress or lose any data on Paste
    c. Does Word recognize the image and offer the image interface?
7. Adding Text
    a. Does the mouse insert to the correct spot on click
    b. Is there a delay for typing after pasting a ton of text?

    c. Does the screen focus the line when typing off screen?
8. Change Text Style/Size
    a. Does highlighting text and changing the style work
    b. If you change in the middle of an already typed line does it work properly
    c. If you have two characters in different styles and click between which style are you on?
9. Emphasize Text
    a. Does highlighting text and adding emphasis types work?
    b. If you change in the middle of an already typed line does the emphasis translate properly?
    c. Does emphasis stay enabled on a blank line?
    d. Can you stack and remove emphasis types properly while highlighted
10. Grammar Check
    a. Is the word dictionary up to date?
    b. Does it check for custom words users have added to their dictionary?
    c. Will it pick up typos for the words they are meant to be?

So this last part I kind of put the test for the functions in the contributing factors. There's not really much way to describe the contributing factors in a way that doesn't make it sound like a question to test for it.

9: Switch between highlighting text segments and switch between bolding, italicizing, and underlining them.

Have some overlap and some far apart.

Go back in and click in some emphasized areas and disable an emphasis and type and see how it flows to the new style.

Check for cases like **o**o**o** becoming **ooo** when you bold and underline them when all highlighted instead of a toggle to o**o**o.