CS7314 Homework #2

Name: Bingying Liang
ID: 48999397

Feb 12 2024

1. Compare the similarities and differences between the SQE process in Chapter 5 and the testing process in Chapter 6. Also, discuss the relative importance of planning/preparation activities in these two processes.

   Solution:
   Similarities:

   (a) Objective-Oriented: Both SQE and the testing process aim to enhance the quality of the software product. They achieve this by identifying and addressing defects and issues within the software development process.

   (b) Process-Driven: Both processes rely on well-defined steps or stages to systematically identify and rectify issues. This includes planning, execution, monitoring, and adjustment phases.

   (c) Continuous Improvement: SQE and the testing process both emphasize improving software quality through continuous evaluation and refinement.

   Differences:

   (a) Scope and Focus:
   - The SQE process is broader, encompassing all aspects of ensuring software quality, such as requirement management, design review, configuration management, risk management, and testing.
   - The testing process primarily focuses on executing test cases to identify defects in the software.

   (b) Activities and Techniques:
   - SQE utilizes a wide range of quality assurance activities and techniques, not limited to testing. It may include formal reviews and audits, static code analysis, and process improvement methodologies.
   - The software testing process mainly concentrates on dynamic testing activities, such as unit testing, integration testing, system testing, and acceptance testing.

   Relative Importance of Planning/Preparation Activities: In both SQE and the testing process, planning and preparation activities are crucial, but their focus and purpose differ. In the SQE process, planning activities involve the quality assurance strategy across the entire software development lifecycle, emphasizing defining a comprehensive quality management plan and ensuring all quality assurance activities align with project objectives. This includes determining the quality requirements, standards, and metrics for the project and how these requirements and standards will be measured and evaluated.

   In the testing process, planning activities focus on creating a detailed test plan, including defining test objectives, determining the scope of testing, allocating resources, scheduling, risk assessment, and preparing test cases and test data. Test planning also involves selecting appropriate testing methods and tools and defining criteria for evaluating test outcomes.

   In summary, while there are differences in objectives, scope, and activities executed between SQE and the testing process, planning and preparation activities are key to achieving effective and efficient quality assurance in both processes.

2. Choose one of the bottom-up or top-down model construction procedures for PT (partition test) to perform one of the following:

(a) Starting with a checklist, construct a corresponding set of partitions for PT.

Solution: Bottom-Up Approach: Starting with a Checklist. In the bottom-up model construction procedure, we start with a detailed checklist of specific conditions or test scenarios we want to cover. This checklist is derived from the functional requirements, user stories, or known use cases of the software. From this checklist, we abstract upwards to form broader partitions that can be used for partition testing.

i. Step 1: Develop a Checklist
Suppose I am testing a simple calculator application. My checklist might include specific operations and edge cases such as:
- Addition of two positive numbers
- Addition of a positive and a negative number
- Subtraction where the result is negative
- Multiplication that results in an overflow
- Division by zero

ii. Step 2: Construct Corresponding Partitions
From this checklist, I can derive the following partitions for my partition testing:
A. Addition Partition: This can be further divided into:
   - Addition of two positive numbers
   - Addition of a positive and a negative number
B. Subtraction Partition: Including cases where the result is positive or negative.
C. Multiplication Partition: Specifically looking at cases that could result in overflow.
D. Division Partition: Divided into:
   - Normal division cases
   - Division by zero as a special case

Each of these partitions represents a set of test cases that share common attributes or expected outcomes, allowing the tester to efficiently explore the behavior of the application under various conditions.

(b) Starting with an overall input domain definition, partition the input domain for PT.

Solution: Top-Down Approach: Starting with an Overall Input Domain Definition. In the top-down model construction procedure, I start with a broad definition of the input domain of the software and then divide it into smaller, manageable partitions based on different criteria such as input type, expected behavior, or boundary conditions.

i. Step 1: Define the Overall Input Domain
For the same calculator application, the overall input domain might be defined as all possible inputs the calculator can process, including all numbers (integers, decimals), operations (addition, subtraction, multiplication, division), and special characters (e.g., division by zero).

ii. Step 2: Partition the Input Domain
From this broad definition, I can partition the input domain as follows:
A. Numeric Input Partition: Further divided into:
   - Integer inputs
   - Decimal inputs
B. Operation Partition: Divided into:
   - Addition operations
   - Subtraction operations
   - Multiplication operations
   - Division operations
C. Special Cases Partition: Including:
   - Division by zero
   - Overflow scenarios

By partitioning the input domain from the top down, I ensure that all aspects of the software's functionality are considered, and I can systematically test the software against a wide range of inputs.

3. Enhance your testing model for the previous problem to build a Musa OP to support UBST. In particular, pay attention to and describe the information sources, participants and additional data collection needed, granularity and practicality of the OP constructed.

Solution:

To enhance the testing model for a simple calculator application, will construct a Musa Operational Profile (Musa OP) to support Usage-Based Statistical Testing (UBST). This construction pays particular attention to information sources, participants, additional data collection needed, granularity, and practicality of the constructed OP.

Customer Types and Weights

Customer Type	**Weight**
Business	0.45
Education	0.10
Government	0.40
Other	0.05

Software User Types and Profiles

Customer Type	**User Type Distribution by Customer Type**				**Overall User Profile**
	Individual	**Educational**	**Business**	**Other**	
Individual User	0.70	0.50	0.30	0.90	0.60
Students	0.20	0.40	0.05	0.00	0.15
Professionals	0.10	0.10	0.65 0.10	0.10	0.25

System Modes and Weights

System Modes	**Weight**
Normal Use	0.95
Configuration	0.04
Maintenance	0.01

Functions and Occurrence Rates Functions: Addition Operation, Subtraction Operation, Multiplication Operation, Division Operation, Error Handling (e.g., division by zero)

Function Usage Rates:

Function	**Usage**
Addition Operation	0.40
Subtraction Operation	0.30
Multiplication Operation	0.20
Division Operation	0.09
Error Handling	0.01

Data Collection Methods

- Log File Analysis: Collect data on function usage rates from software usage log files.
- User Surveys: Determine user preferences and usage patterns for different calculator functions through surveys for more accurate function usage determination.

Conclusions and Testing Focus

- Addition and subtraction operations are the most frequently used functions and should be the focus of rigorous testing to ensure accuracy and efficiency.
- Division operations and error handling (such as division by zero) may have lower usage rates but have a significant impact on user experience and should ensure robustness.
- Given the user profile overview, individual users are the primary user group, so testing should focus on meeting the needs of individual users, including usability, accuracy, and response time.
- To ensure comprehensive and accurate testing, data collection should combine log file analysis and user surveys to identify and prioritize the most critical test cases.

By constructing the Musa OP in this manner, testing resources can be effectively allocated to the most critical and frequently used functions within the simple calculator application, thereby enhancing the software's reliability and user satisfaction.