CMPE 287 – Software Quality Assurance and Testing

Homework #1 Assignment

First Name: Harish

Last Name: Marepalli

SJSU ID: 016707314

Professor: Dr. Jerry Gao

Question #1: Basic Concepts

a. What is the component testing process for component vendors? Please use a diagram to show the vendor-oriented component test process.

Answer:

Component Testing Process for Component Vendors:

Component testing is a critical phase in software development for component vendors, ensuring the reliability and functionality of individual software components. The step-by-step process is as follows:

1. Requirement Analysis:

- Understanding the component's specifications and requirements is the initial step.
- Identify its functionality, inputs, outputs, and expected behavior.

2. Test Planning:

- Create a comprehensive test plan that outlines the entire testing process.
- Define objectives, test cases, test data, resources, and schedules.

3. Test Case Design:

- Develop detailed test cases based on the component's specifications.
- Cover various scenarios, including normal behavior, boundary conditions, error handling, and edge cases.

4. Test Environment Setup:

- Establish a controlled testing environment that closely resembles the production environment.
- Include the necessary hardware, software, and configurations.

5. Test Data Preparation:

- Prepare a diverse set of test data, including valid and invalid inputs.
- Ensure that test data covers a wide range of scenarios to validate the component thoroughly.

6. Test Execution:

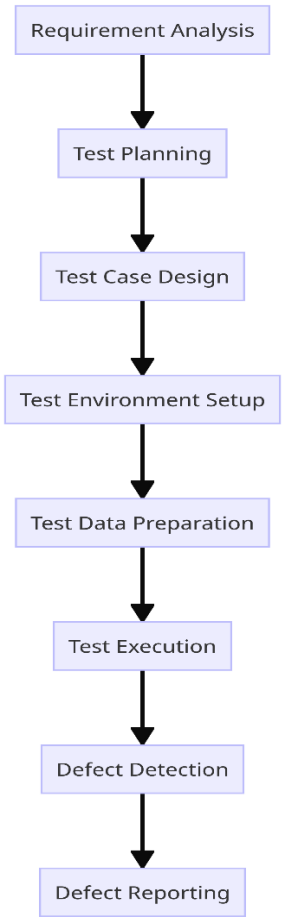
- Execute the test cases in the isolated testing environment.
- Use automated testing tools when applicable to streamline the testing process.

7. Defect Detection:

- During test execution, identify discrepancies between expected and actual outcomes.
- Document each detected defect with detailed information for debugging and resolution.

8. Defect Reporting:

- Report identified defects to the development team.
- Include information about the defect's severity, steps to reproduce, and relevant logs or error messages.



Component Testing Process

b. Why software testing is so hard? (Hint: three points. Please read the given paper #1)

Answer:

There are several reasons as to why software testing is so hard. They are given below.

1. **Platform and Environment Variability:** Software tested on one platform may not function correctly on another due to differences in environments and file formats.
2. **Test Coverage:** Ensuring comprehensive test coverage is difficult, as there are countless potential test scenarios, making testing time-consuming.
3. **Coordination with Developers:** Effective collaboration with developers is crucial for identifying and resolving issues during testing.

To summarize, software testing is demanding because it requires testers to possess extensive knowledge, write substantial code, and invest significant time. They must also have strong development skills and an understanding of algorithms, graph theory, and formal languages.

c. Identify the three major differences between white-box testing and black-box testing.

Answer:

White box testing	Black box testing
The main aim of this type of testing is to test the infrastructure of the application. It is performed by Software Developers.	The main aim of this type of testing is to test the functionality of the application. It is performed by professional Software Testers.
Access to the source code and internal working knowledge of the application is required to perform this testing. Therefore, it is time consuming and exhaustive.	Access to the source code is not required and there is no need to know work the internal code of the application works. Therefore, it is less time consuming and less exhaustive.
The test data is limited to a specific part of functionality at a given point of time so, all the possible inputs can be tested. Therefore, the test coverage is higher in this type of testing.	There is a wide range of possibilities of the test data so, all the possible inputs cannot be covered during testing. Therefore, the test coverage will be less in this type of testing.

d. List three major issues in testing AI software? (Please read the given Paper #2)

Answer:

The three major issues in testing AI software are as follows:

1. Insufficient AI Validation Models: One of the significant challenges in testing AI software is the inadequacy of traditional validation models like black box and white box testing. These models fall short when it comes to ensuring the quality and reliability of AI systems, which exhibit dynamic behaviors and features. Consequently, there is a pressing need to develop robust validation models tailored to AI testing requirements.

2. Lack of Industry-Focused QA Standards: Current software quality assurance (QA) standards and principles primarily cater to conventional feature-based software applications. These standards often do not align with the unique testing demands posed by AI software. As a result, there is a deficiency of industry-specific QA standards that can effectively guide the testing of AI-driven software solutions.

3. Shortage of Suitable Automation Tools: Automation has been a driving force in streamlining various aspects of work, including software testing. However, many of the automation tools developed over the past few decades are not well-suited for the dynamic nature of AI testing. While they have proven effective for testing traditional software, they often struggle to scale and adapt to the complexities and evolving requirements of AI testing. This shortage of appropriate automation tools poses a significant hurdle in ensuring efficient and effective AI software testing.

Question #2: Equivalence Partitioning Testing Questions. (20%)

Assume you have been asked to use Equivalence Partitioning method to check the ATM machine login function regarding the PIN number validation. Here are some detailed requirements:

Users must use valid ATM card and enter a correct PIN number to access their accounts. Otherwise, a rejection message will be displayed to them.

Here are some specification descriptions relating to a valid PIN number.

- Each valid PIN number must be formed with letters (‘a’ to ‘z’) and digitals.
- Each PIN must include at least one up-case letter and one lower-case letter.
- Each PIN must include at least 4 digital numbers from 1 to 10.
- Each PIN is acceptable if it includes one special chars from the following list:
    - !, #, *, %, ?
- The length of each valid PIN must be in the range from 8 to 12.

Whenever an invalid PIN is entered, one of following messages will be displayed:

“Incorrect Password” - This suggests that a PIN (entered by a user) is valid but not correct.

“Invalidated Password”- This suggests that a PIN (entered by a user) is invalid.

a) Please list your identified equivalence classes when you are testing this function and present them using a table (or a diagram). (10%)

b) Based on these equivalence classes identify and present test cases in a table. (10%)

Answer:

a) Below is the table that identifies all the equivalence partition classes.

Condition	Equivalence Class	Specification
Pin length = 0	EC1	Null pin
Pin length < 8	EC2	Any combination of alpha-numeric and characters
Pin length > 12	EC3	Any combination of alpha-numeric and symbols
Pin length >=8 and <=12	EC4	All digits
Pin length >=8 and <=12	EC5	All lower-case characters
Pin length >=8 and <=12	EC6	All symbols
Pin length >=8 and <=12	EC7	All upper-case characters
Pin length >=8 and <=12	EC8	Only alphanumeric. No symbols
Pin length >=8 and <=12	EC9	Only alphabets and symbols. No digits.
Pin length >=8 and <=12	EC10	Only digits and symbols. No alphabets.
Pin length >=8 and <=12	EC11	Matches all the specifications but does not contain at least one upper-case character.
Pin length >=8 and <=12	EC12	Matches all the specifications but does not contain at least one lower-case character.

Pin length >=8 and <=12	EC13	Matches all the specifications but contains only 3 digits.
Pin length >=8 and <=12	EC14	Matches all the specifications but contains symbol other than !, #, *, %, ?.
Pin length >=8 and <=12	EC15	Matches all the given requirements but not correct i.e., valid but not correct
Pin length >=8 and <=12	EC16	Matches all the given requirements and matches the real pin

b) The test cases based on the equivalence partition are as follows.

Answer:

Test Case: **Abcde#1234**

Test Case No.	**Equivalence Classes**	**Input**	**Output**
1	EC1	Null	Invalidated Password
2	EC2	A2#	Invalidated Password
3	EC3	SDfgty234!#ui	Invalidated Password
4	EC4	453681259	Invalidated Password
5	EC5	hajdiftvs	Invalidated Password
6	EC6	%!#%!#!%	Invalidated Password
7	EC7	EITOVNDVC	Invalidated Password

8	EC8	Dyi452nH	Invalidated Password
9	EC9	Doa#%!gh	Invalidated Password
10	EC10	%84957!#	Invalidated Password
11	EC11	dh7634%fd	Invalidated Password
12	EC12	SHAY#5678	Invalidated Password
13	EC13	Shay#567	Invalidated Password
14	EC14	Dh7634@fd	Invalidated Password
15	EC15	Abcde!1234	Incorrect Password
16	EC16	Abcde#1234	Correct Password

Question #3: Boundary Value Testing (20%)

Please use the boundary value analysis method to derive a set of black-box test cases for the following function Z (X, Y) by applying the boundary value analysis criteria. Please make sure to consider all input data and expected output data in a two-dimensional space.

$$
Z(X, Y) = \begin{cases} 3X - 2Y + 5 & \text{when } 10 < X \leq 20 \text{ and } 25 \leq Y < 35 \\ \text{Undefined} & \text{when } X = 4 \text{ and } Y = 0 \\ 4X - 2Y - 3 & \text{when } 0 < X < 8 \text{ and } 5 \leq Y < 15 \\ -40 & \text{Otherwise} \end{cases}
$$

a) Define all boundaries for Z (X, Y) (5%)

Answer:

Boundary 1 | $3X-2Y+5$ when $10 < X \leq 20$ and $25 \leq Y < 35$

Boundary 2 | Undefined when $X = 4$ and $Y = 0$

Boundary 3 | $4X - 2Y - 3$ when $0 < X < 8$ and $5 \leq Y < 15$

b) Define the boundary values for each boundary (7%)

Answer:

B1 | $3X-2Y+5$ when $10 < X \leq 20$ and $25 \leq Y < 35$

$X = 10, 20$

$Y = 25, 35$

B2 | Undefined when $X = 4$ and $Y = 0$

$X = 4$

$Y = 0$

B3 | $4X - 2Y - 3$ when $0 < X < 8$ and $5 \leq Y < 15$

$X = 0, 8$

$Y = 5, 15$

c) Define the test cases for each boundary. (8%)

Answer:

Test Case for B1:

Test Case	X	Y	Output
1	9	24	-40
2	9	25	-40
3	9	30	-40
4	9	35	-40
5	9	36	-40

6	10	24	-40
7	10	25	-40
8	10	30	-40
9	10	35	-40
10	10	36	-40
11	15	24	-40
12	15	25	0
13	15	30	-10
14	15	35	-40
15	15	36	-40
16	20	24	-40
17	20	25	15
18	20	30	5
19	20	35	-40
20	20	36	-40
21	21	24	-40
22	21	25	-40
23	21	30	-40
24	21	35	-40
25	21	36	-40

Test Case for B2:

Test Case	**X**	**Y**	**Output**
1	3	-1	-40
2	3	0	-40
3	3	1	-40
4	4	-1	-40
5	4	0	Undefined

6	4	1	-40
7	5	-1	-40
8	5	0	-40
9	5	1	-40

Test Case for B3:

Test Case	**X**	**Y**	**Output**
1	-1	4	-40
2	-1	5	-40
3	-1	10	-40
4	-1	15	-40
5	-1	16	-40
6	0	4	-40
7	0	5	-40
8	0	10	-40
9	0	15	-40
10	0	16	-40
11	4	4	-40
12	4	5	0
13	4	10	-10
14	4	15	-40
15	4	16	-40
16	8	4	-40
17	8	5	15
18	8	10	5
19	8	15	-40
20	8	16	-40
21	9	4	-40

22	9	5	-40
23	9	10	-40
24	9	15	-40
25	9	16	-40

Question #4: Decision table testing question (20%)

Please use decision table test method to design your tests for AVL Tree functions. Please answer your questions below.

For the AVL Tree program https://www.geeksforgeeks.org/avl-trees-containing-a-parent-node-pointer/

a) Please use the decision table testing method to generate your decision table for AVL Tree deletion Function. (5%)

Answer:

The decision table for AVL Tree deletion function is as follows.

Condition	R1	R2	R3	R4	R5	R6	R7
Not Binary Tree	T	F	F	F	F	F	F
Root = None	-	T	F	F	F	F	F
Node Balanced (Balance factor = 0)	-	-	T	F	F	F	F
If BF (node) = +2 and BF (node -> left-child) = +1	-	-	-	T	F	F	F
If BF (node) = -2 and BF (node -> right -child) = 1	-	-	-	-	T	F	F
If BF (node) = -2 and BF (node -> right -child) = +1	-	-	-	-	-	T	F

If BF (node) = +2 and BF (node -> left -child) = - 1	-	-	-	-	-	-	T
Actions							
Left Rotation	-	-	-	T	-	-	-
Right Rotation	-	-	-	-	T	-	-
Left rotation -> Right rotation	-	-	-	-	-	-	T
Right rotation -> Left rotation	-	-	-	-	-	T	-

b) Please use the decision table testing method to generate your decision table for search function for a given AV tree. (5%)

Answer:

The decision table for search function for a given AVL tree is as follows.

Condition	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**
Not Binary Tree	T	F	F	F	F	F	F	F	F
Root = None	-	T	F	F	F	F	F	F	F
Node Balanced (Balance factor = 0)	-	-	T	F	F	F	F	F	F
If BF (node) = +2 and BF (node -> left-child) = +1	-	-	-	T	F	F	F	F	F
If BF (node) = +2 and BF (node -> left -child) = - 1	-	-	-	-	T	F	F	F	F
If BF (node) = +2 and BF (node -> left -child) = 0	-	-	-	-	-	T	F	F	F

If BF (node) = -2 and BF (node -> right -child) = - 1	-	-	-	-	-	-	T	F	F
If BF (node) = -2 and BF (node -> right-child) = +1	-	-	-	-	-	-	-	T	F
If BF (node) = -2 and BF (node -> right-child) = 0	-	-	-	-	-	-	-	-	T
Actions									
Left Rotation	-	-	-	T	-	T	-	-	-
Right Rotation	-	-	-	-	-	-	T	-	T
Left rotation -> Right rotation	-	-	-	-	T	-	-	-	-
Right rotation -> Left rotation	-	-	-	-	-	-	-	T	-

c) Provide your decision table test case set for (a) (7%)

Answer:

The decision table test case set for (a) is as follows.

Testcases	BF_before	BF_after	Type of Node	Actions	Expected
1	Left (L)	Balanced (B)	Leaf (L)	Left-Left Rotation (LLR)	Balanced (B)
2	Equal Height (E)	Balanced (B)	Leaf (L)	No Rotation (NR)	Balanced (B)
3	Right (R)	Balanced (B)	Leaf (L)	Left-Right Rotation (LRR)	Balanced (B)
4	Left (L)	Equal Height (E)	Leaf (L)	No Rotation (NR)	Balanced (B)

5	Equal Height (E)	Equal Height (E)	Leaf (L)	No Rotation (NR)	Balanced (B)
6	Right (R)	Equal Height (E)	Leaf (L)	No Rotation (NR)	Balanced (B)
7	Left (L)	Rebalanced (RB)	Leaf (L)	Left-Right Rotation (LRR)	Rebalanced (RB)
8	Equal Height (E)	Rebalanced (RB)	Leaf (L)	No Rotation (NR)	Rebalanced (RB)
9	Right (R)	Rebalanced (RB)	Leaf (L)	Left-Left Rotation (LLR)	Rebalanced (RB)
10	Left (L)	Balanced (B)	One Child (O)	No Rotation (NR)	Balanced (B)

d) What is the test complexity when you apply the decision table testing in (a)? (3%)

Answer:

In decision table testing, the test complexity is determined by calculating 2 raised to the power of the number of conditions involved in the testing i.e., 2 ^ no. of conditions.

For the AVL Insertion node function, which has 7 conditions, the test complexity is 2^7.

For the AVL Deletion node function, which involves 9 conditions, the test complexity is 2^9.

Question #5: Online Search questions (20%)

Please search online to find the answers to the following questions:

Search online to find out at least 10 published papers or technical reports addressing the following topics:

1 - (8%) Identify and generate a reference list which displays publications and reports addressing the testing standards and quality assurance for intelligent systems and/or software (8%) (please use IEEE paper format)

Answer:

1. J. Gao, C. L. Xie and C. Q. Tao, "Quality assurance for big data-issuss challenges and needs", Proc. IEEE 10th Int. Symp. Service Oriented Syst. Eng., pp. 433-441, Apr. 2016.
2. B. Wang, "Automated Support for Classifying Software Failure Reports", Proc. of the 25th International Conference on Software Engineering (ICSE), pp. 465-475, 2003.
3. J. F. Bowring, J. M. Rehg and M. J. Harrold, "Active Learning for Automatic Classification of Software Behavior", Proc. of the ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA), pp. 195-205, 2004.
4. R. Gove and J. Faytong, "Identifying infeasible gui test cases using support vector machines and induced grammars", IEEE Fourth International Conference on Software Testing Verification and Validation Workshops, pp. 202-211, 2011.
5. C. Murphy, G. E. Kaiser and M. Arias, "An approach to software testing of machine learning applications", Nineteenth International Conference on Software Engineering & Knowledge Engineering, 2008.
6. Tian, Y., Pei, K., Jana, S., and Ray, B, "DeepTest: automated testing of deep-neural-network-driven autonomous cars," Proceedings of the 40th International Conference on Software Engineering," 303–314, 2018.
7. Wang, S., Heinrich, S., Wang, M. and Rojas, R, "Shader-based sensor simulation for autonomous car testing," 2012 15th International IEEE Conference on Intelligent Transportation Systems," 224–229, 2012.
8. Utesch, F., Brandies, A., Pekezou Fouopi, P., and Schießl, C, "Towards behaviour based testing to understand the black box of autonomous cars," European Transport Research Review, 2020.
9. Abdessalem, R. B., Panichella, A., Nejati, S., Briand, L. C., and Stifter, T, "Testing autonomous cars for feature interaction failures using many-objective search," Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, 143–154, 2018.
10. T. M. King, J. Arbon, D. Santiago, D. Adamo, W. Chin and R. Shanmugam, "AI for Testing Today and Tomorrow: Industry Perspectives," 2019 IEEE International Conference On Artificial Intelligence Testing (AITest), Newark, CA, USA, 2019, pp. 81-88, doi: 10.1109/AITest.2019.000-3.

2 - Generate a comparison table which compares and summarize your findings. Your table must be presented in terms of the followings:

- **Targeted intelligent systems (such as computer vision, e-learning, smart chatbot, etc.)**
- **Quality assurance parameters**
- **Quality assurance processes and standards**
- **Evaluation metrics and coverage complexity**
- **Test models**

Answer:

The comparison table is as follows.

Topic	Targeted Intelligent Systems	Quality Assurance Parameters	Quality Assurance Processes and Standards	Evaluation Metrics and Coverage Complexity	Test Models
Software Testing	E-learning platforms	Functional and Usability Parameters	ISO 29119, IEEE 829	Code coverage, Usability testing	Agile, Waterfall
AI Model Verification	Computer Vision	Accuracy, Robustness	V&V processes for AI	Precision-Recall, F1-score	CNN, LSTM
Autonomous Systems	Self-driving cars	Safety, Reliability	ISO 26262, DO-178C	Failure Mode Analysis	Simulators
Security and Privacy	Smart Chatbots	Data Security, Privacy	OWASP, NIST Cybersecurity	Vulnerability Scanning	Threat Models
Agile Testing	Agile Software	Adaptability, Speed	Agile Testing Principles	Test Automation, Continuous Testing	Scrum, Kanban