Software and Computational Systems Lab
Prof. Dr. Marie-Christine Jakobs
Max Barth

Exercise 01

1 Basic Testing Terms

a) Define the terms Failure, Error and Fault. Give examples for each term and explain how fault, error, and failure relate to each other.

- Testing speak (e.g., lecture):
  - Failure: Observable deviation from the specification.
  - Fault: “Defect” or “Flaw” of a system: programming or design flaw whereby the system does not conform to its specification. Also, called “Bug”.
  - Error: Human mistake that leads to a fault in the system
- Fault-tolerance speak (see A. Avizienis, J.-C. Laprie, B. Randell, C. E. Landwehr: Basic Concepts and Taxonomy of Dependable and Secure Computing, 2004.)
  - **Failure.** Observable deviation from the specification.
  - **Error.** Part of the system state that may lead to a failure.
  - **Fault.** “Defect” or “Flaw” of a system: programming or design flaw whereby the system does not conform to its specification.

b) Consider an accounting scenario in which a withdrawal is allowed only if the account balance is above 500. The given code in Listing 1 is an attempt to implement this specification. Identify the fault, error, and failure.

```c
int attempt_withdrawal(unsigned char amount) {
  int balance = get_balance();
  if (balance < 500) {
      printf("Cannot proceed: Balance too low\n");
      return -1;
    }
  allow_withdrawal(amount);
  return 0;
}
```

Listing 1: Accounting Example

When the balance is 500, the check for the balance being only for <500 instead of ≤ 500 will allow the withdrawal, leading to the failure of the system as its behavior violates the specification.

- Fault: incorrect condition
- Error (FT-speak): balance $= 500$ leading to activation of fault
  Error(Testing-speak): cannot be unambiguously determined. Could be a typo or a misunderstanding of the spec.

- Failure: allowing the withdrawal

c) Consider the following test description for the method in Listing 1 and identify the different components of the test in the description.

> At the beginning of the test the balance is greater than 500. The test should execute the method `attempt_withdrawal` with input 200. The expected return value of the call is 0 and the value of the balanance is reduced by 200.

- Precondition: balance is greater 500
- Input: 200
- Expected result: 0
- Postcondition: balance is reduced by 200
- Instructions: execute method `attempt_withdrawal` with specified input (200)

d) Name different test levels and explain which kind of testing activity you have explained on each level.

- Component test
- Integration test
- System test

2 Motivation for Testing

a) Although many techniques for ensuring software quality have been developed, software faults still exist. What are the main reasons?

- Software is complex.
- Too many features, too little time, diminishing return for fixing rare bugs.
- Bugs are often triggered by bad specifications, design and bad inputs from users, system (firmware) environment, aging.
- Humans make mistakes. Even in software developed by a professional programmer, typically 100-150 "bugs" are seen per 1000 lines of code.

b) Why is it important to detect failures and fix faults?

- Failures can cause a loss in reputation
- Failures can cause high costs
- Failures may be harmful up to endangering human lives

3 Testing versus Debugging

a) Using the introduced terms failure, fault, and error, state the goal of software testing and how that goal is conceptually achieved.

- Goal: Fault detection
- Method: Failure detection & fault existence inference by backtracking causal chain

b) Using the introduced terms failure, fault, and error, state the goal of debugging and how that goal is conceptually achieved.

- Goal: Fault removal
- Method: Fault localization via error (in the fault-tolerant-computing sense) analysis & correction

4 Risk-Oriented Testing

a) Explain why testing is typically incomplete and what is the impact of its incompleteness.

- Testing can only underapproximate the program behavior.
- Only a subset of the test inputs or the program behavior can be tested due to the large or infinite number of input combination in addition with the large or infinite number of execution paths.
- There is only limited time and budget available for testing.
- Testing can reveal faults (bugs) but cannot ensure their absence
- At some point testing is no longer economic.

b) What is a risk in software testing and how can it be determined?

- Risks is combination, e.g., product, of likelihood of failure and the costs of the failure
- Typically the total risk is composed of the risk of several individual failures
- The likelihood of a failure could be determined by looking at different aspects like the expected usage of a piece of code, the expectation of the failure probability, the probability that the failure has not been revealed by current testing. Often, these values can be estimated based on past experience of system behavior and detected faults, team performance, etc.
- The costs of the failure depends on various factors like fines, penalties, costs of fixing the fault and providing the fix to the users, impact on reputation, market value, etc.

c) Explain the idea behind risk oriented testing.

- Takes the individual risks of software pieces into account
- Goal is to test riskier software more intensively an prioritize its tests

General notes:

- The given solution is just a proposal. We do not guarantee correctness.