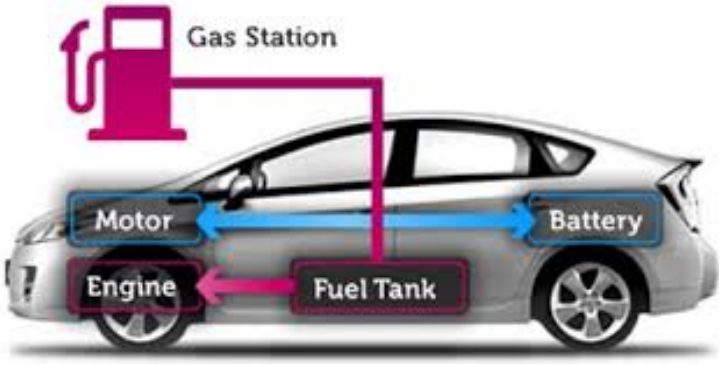
BACS2103 SOFTWARE QUALITY ASSURANCE AND TESTING

Chapter 1 – Software Testing Concepts

- **Objectives**
  - Understand the basic concept of software testing.
  - Identify the importance of software testing.
  - Explain the principle of software testing.

Software, No Problem?
Are you sure?

Motivation Case I



- May 2005, newspaper article reported that a major hybrid car manufacturer had to install a software fix on 20,000 vehicles due to problems with invalid engine warning lights and occasional stalling.
- In the article, an automotive software specialist indicated that the automobile industry spends $2 billion~$3 billion per year fixing software problems.

Source: https://www.123helpme.com/major-computer-system-failures-caused-by-software-bugs-view.asp?id=159485

Motivation Case II

- June 2007, news reports claimed that software flaws in a popular online stock-picking contest could be used to gain an unfair advantage in pursuit of the game's large cash prizes.
- Outside investigators were called in and in July the contest winner was announced. Reportedly the winner had previously been in 6th place, indicating that the top 5 contestants may have been disqualified.

Source: http://www.voidcn.com/article/p-nuvaeelc-dz.html

Motivation Case III

- June, 2011, Mt. Gox (a Japanese bitcoin exchange) had lost more than 850,000 bitcoins (equivalent to half a billion US dollars) due to a weakness in the system.

Source : https://raygun.com/blog/costly-software-errors-history/

Motivation Case IV

- In April 2015, Starbucks' point-of-sale systems encountered failure during a daily system refresh.
- 60 percent of the systems at Starbucks' 13,500 locations in the U.S. and Canada had shut down.
- Not able to accept payments or make change.
- Estimated lost is $3 million

Source :https://www.qasymphony.com/blog/4-worst-software-bugs-2015/

Lessons of the motivation case?

- Low-level **impacts of software bugs** may cause **inconvenience to the end users** while the serious-level impacts may cause **large financial cost / life-death matters**.

Overview of Software Testing

- Construction of industry product base on **set of requirements**.
- If problems exists, **correction must be done**.
- Checking on partial software product and final software product is difficult due to its **dynamic behavior**.
- To test the software, it must be **executed (run)** on a computer and **compare with the requirements**.

Purposes of Software Testing

- Analyzing a program or its documentation to prevent failures.
- To check if the system meets the requirements and be executed successfully in the intended environment.
- To check if the system is “Fit for purpose”.
- To check if the system does what it is expected to do.
- To identify and correct defects.
- To check whether the users need are satisfied.
- To avoid user detecting.
- Also to provide quality product.

Who perform the test?

- **Software Test Engineer/Tester**
  - Professional who is in charge of technical test activities; designing test inputs, producing test case values, running test scripts, analyzing results, and reporting results to developers and managers.
- **Programmers/Developers**
- **End-Users**

Terminology (1/6)

Software Error

- An **incorrect internal state** that is the manifestation (symptom/sign) of some fault.
- Cause of fault.
- Error or **mistake by a person**.
- Example: Bad programming or incorrect use of program statements.

Terminology (2/6)

Software Fault (DEFECT)

- Also known as **bug, defect or internal error**.
- Present from the time the software was developed but only when the software is executed, becoming visible as a failure.
- Example: incorrect/forgotten statements in the program.
- **Fault can cause none, one, or many failures for any number of users.**
- A particularly dangerous example is some small corruption of stored data, which may be found a long time after it first occurred.

Terminology (3/6)

Defect Masking

- It is possible that a **fault is hidden by one or more other faults** in other parts of the program.
- Failure happened after the masking defects have been corrected.
- This demonstrates that corrections can have side effects.

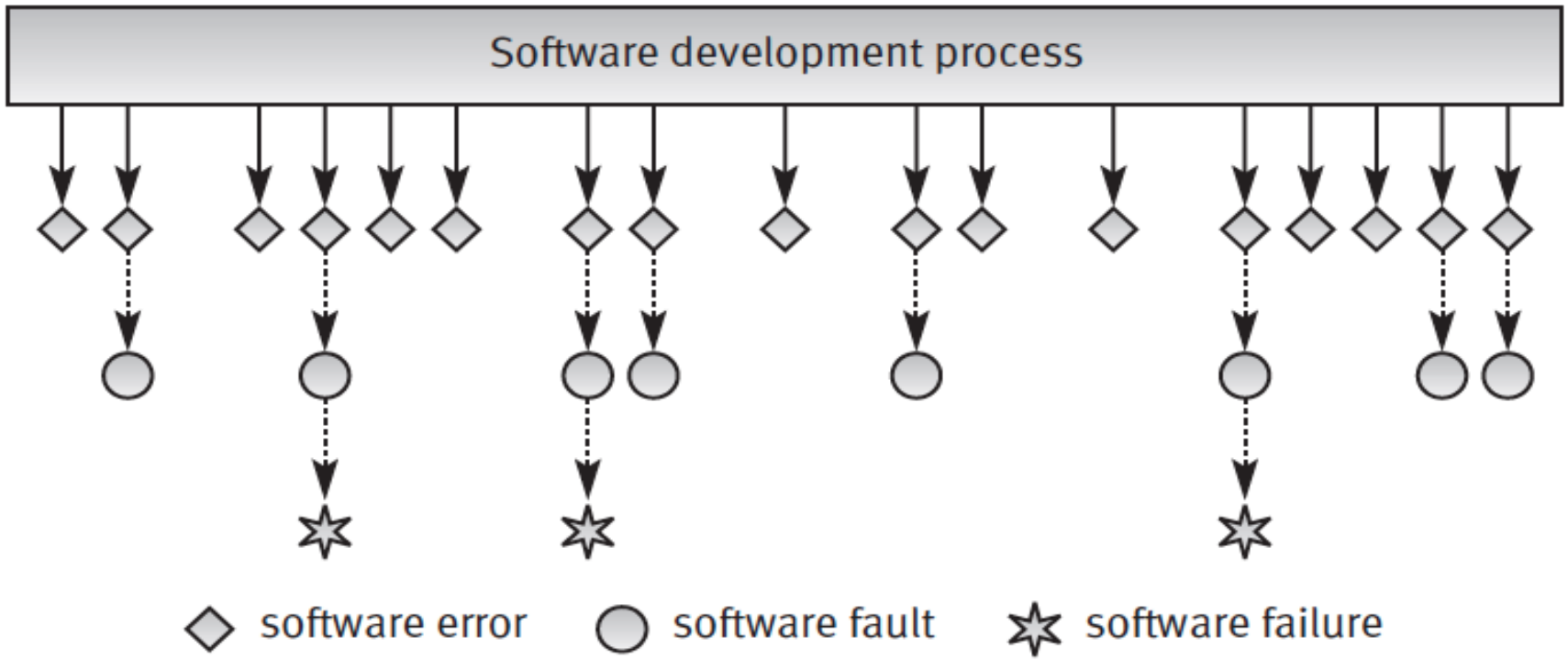
Terminology (4/6)

Software Failure

- Means that a given **requirement is not fulfilled.**
- Discrepancy between the actual result/behavior and the expected result/behavior.
- Present if a legitimate (user) expectation is not adequately met.
- Example: A software is too difficult to use or too slow but still fulfills the functional requirements.
- They may caused by **faults**.

Terminology (4/6)

Connections between software errors, faults and failures



Not all software errors will turn into software fault.
And, not all software faults will turn into software failure.

Terminology (5/6)

Validation

- The process of **evaluating software at the end of software development** to ensure its compliance with the intended usage.

  Validation: Are we building the right system?

Verification

- The process of **determining whether the products of a given phase of the software development process fulfill the requirements** established during the previous phase.

  Verification: Are we building the system right?

Terminology (6/6)

Software Testing

- **Process of exercising or evaluating a system** or system component by manual or automated means to verify that it satisfies specified requirement (IEEE 829-2008).
- Process of **executing a program with the intent of finding errors**. Extremely creative and intellectually challenging tasks. - Myers
- Process: Sequence of step performed for a given purpose.(IEEE)

Principle of Software Testing (1/7)

Principle 1:

Testing shows the presence of defects, not their absence.

Testing can show that the product fails, i.e., that there are defects. Testing cannot prove that a program is defect free. Adequate testing reduces the probability that hidden defects are present in the test object. Even if no failures are found during testing, this is no proof that there are no defects.

Principle of Software Testing (2/7)

- **Principle 2:**

  Exhaustive testing is impossible.

It's impossible to run an exhaustive test that includes all possible values for all inputs and their combinations combined with all different preconditions. Software, in normal practice, would require an "astronomically" high number of test cases. Every test is just a sample. The test effort must therefore be controlled, taking into account risk and priorities.

Principle of Software Testing (3/7)

- **Principle 3:**

  Testing activities should start as early as possible.

Testing activities should start as early as possible in the software life cycle and focus on defined goals. This contributes to finding defects early.

Principle of Software Testing (4/7)

- **Principle 4:**

Defect clustering.

Defects are not evenly distributed; they cluster together. Most defects are found in a few parts of the test object. Thus if many defects are detected in one place, there are normally more defects nearby. During testing, one must react flexibly to this principle.

Principle of Software Testing (5/7)

- **Principle 5:**

The pesticide paradox.

Insects and bacteria become resistant to pesticides. Similarly, if the same tests are repeated over and over, they tend to lose their effectiveness:

they don't discover new defects.

New and modified test cases should be developed and added to the test. Parts of the software not yet tested, or previously unused input combinations will then become involved and more defects may be found.

Principle of Software Testing (6/7

- **Principle 6:**

Testing is context dependent.

Testing must be adapted to the risks inherent in the use and environment of the application. No two systems should be tested in the exactly same way.

The intensity of testing, test exit criteria should be decided respectively for every software system, depending on its usage environment.

E.g., safety-critical systems require different tests than e-commerce applications.

Principle of Software Testing (7/7)

- **Principle 7:**

No failures means the system is useful, is a fallacy.

Finding failures and repairing defects does not guarantee that the system meets user expectations and needs. Early involvement of the users in the development process and the use of prototypes are preventive measures intended to avoid this problem.

Debugging vs Testing

- Debugging **is not** testing.
- Debugging is the process of **analyzing and locating bugs**, when the software does not behave as expected.
- Debugging supports testing

Test Process

- Test Planning and Control
- Test Analysis and Design
- Test Implementation and Execution
- Evaluating Exit Criteria and Report
- Test Closure Activities

Psychology of Testing

- Programmer test their own software ?
  - Possible or not?
  - What is the problem?

- Different independence level to carry out the test
  - Programmer himself
  - Other people in same project
  - Other departments.
  - Other organizations.

The End