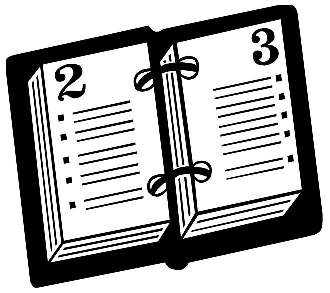
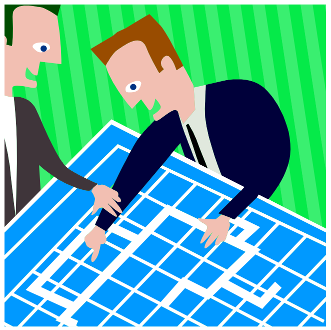
INFO5990 Professional Practice in IT

Lecture 08A





Testing software 1

Program testing

Component testing

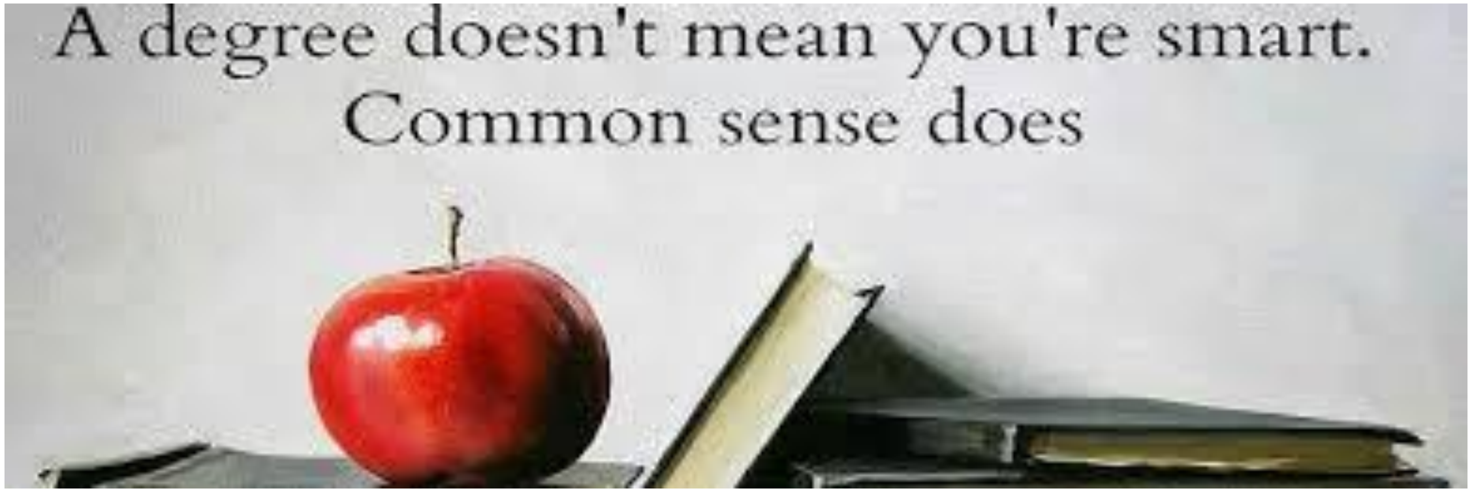
Software Tester's get paid

Between $350-$1200 per day

Depending on experience and testing field

Anyone interested?

Common Sense !!



By the end of this lecture you will be able to:

- Appreciate some of the difficulties of software testing
- Understand the terms error and failure
- Explain the concept of equivalence classes
- Appreciate the difference between ‘test data’ and ‘test cases’ and scenarios
- Construct a set of test cases to be used for program testing
- Go ahead with practice exercises

Testing for bugs

What makes software systems so different from engineering ones?

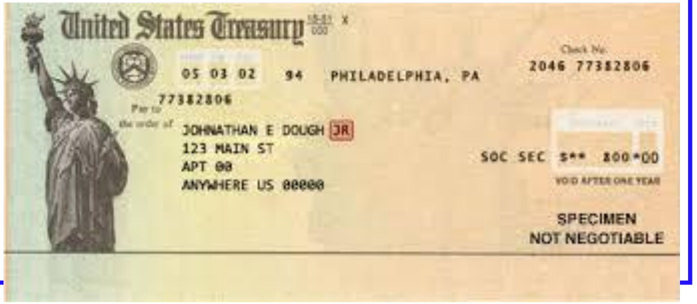
In 2010 MS Office had over 60 million lines of code!
By now … ?

- “They are expected to satisfy human institutions of almost capricious complexity”
- “Software comes under pressure because users devise **new ways to use it**”

The Sydney harbour bridge or the Opera House are both clearly visible, but, how can we visualise MS Office?

How much testing is enough? (1)

- In February 2003 the U.S. Treasury Department mailed 50,000 Social Security cheques without a beneficiary name.
  - A spokesperson said that the missing names were due to a software program maintenance error.
- In July 2001 a “serious flaw” was found in off-the-shelf software that had long been used in systems for tracking U.S. nuclear materials.
  - The software had recently been donated to another country and scientists in that country discovered the problem and told U.S. officials about it.



United States Treasury

Check No. 2046 77382806

05 03 02 94 PHILADELPHIA, PA

77382806

Pay to the order of

JOHNATHAN E DOUGH JR
123 MAIN ST
APT 00
ANYWHERE US 00000

SOC SEC $** 800*00

VOID AFTER ONE YEAR

SPECIMEN
NOT NEGOTIABLE

How much testing is enough? (2)

- In June 1996 the first flight of the European Space Agency's Ariane 5 rocket failed shortly after launching, resulting in an uninsured loss of $500 million
  - The disaster was traced to the **lack of exception handling for a floating-point error when a 64-bit integer** was converted to a 16-bit signed integer.
- In October 1999 the $125 million NASA Mars Climate Orbiter (an interplanetary weather satellite) was lost in space due to a **data conversion error**.
  - Investigators discovered that software on the spacecraft performed certain calculations in English units (yards) when it should have used metric units (metres).

Case Study

iPhone 7

What are you going to test ?
Functionality ?
Useability ?
What else ?

Questions about software that only testing can answer

1. Does it really work as expected?
2. Does it meet the users' requirements?
3. Is it what the users expected?
4. Do the users like it?
5. Is it compatible with users' other systems?
6. Is its performance acceptable?
7. How does it scale as more users are added?
8. Is it ready for release?
9. Which areas need to be improved?

What makes a program ‘good’?

Eight important attributes of software

1. Correct
2. Reliable
3. Robust

Essential attributes

4. Useful
5. Usable
6. Maintainable
7. Efficient
8. Scalable

Quality attributes

Three essential software attributes

1. **Correct**
   - behaves according to the functional requirements
   - produces the right result for any given set of inputs
2. **Reliable**
   - behaves as expected on every occasion
   - over any period of time
3. **Robust**
   - behaves in a predictable and controllable fashion even if the input not valid.
   - a program may be correct but not robust
     - e.g. division by zero or non-numeric input.

Two ‘quality’ attributes

4. **Useful**
   - accomplishes something the user needs
   - output can be input to another program
5. **Usable**
   - ‘intuitive’ , i.e. can use without having to be shown how
   - minimum input by the user
   - usability depends on the interface
   - compare terms
     ‘user friendly’ and ‘ease of use’

Three further ‘quality’ attributes

6. Efficient
   - requires minimum time and resource
7. Scalable
   - will continue to behave in an acceptable manner as size or volume of input is increased
   - scalability can depend on efficiency
8. Maintainable
   - easily modified for new requirements

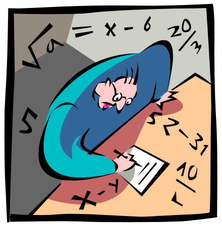
The software testing process

"Software testing is finding defects, especially those that are more likely to result in faults, and especially those with high economic exposure"

Beizer, B. (1990) Software Testing Techniques (2nd edition)

- The program is executed with 'test data' as input
  - A successful test is one that finds an error!
- Testing can never show that there are no errors in a program ...
- ... because it can never prove that the program will perform all its intended functions correctly under all circumstances

Why is software testing so difficult?



- Testing tends to be largely an informal task
- Testing can never guarantee to find all errors ...
- ... because there can never be a ‘enough’ test cases for testing to be said to be ‘complete’
- **Testing is expensive, takes a lot of time and is very tedious**
- **Programmers often fail to find errors in programs that they have written**

Q1. If you test a computer program with a particular value of input and the result is different from what you expected, then that probably means

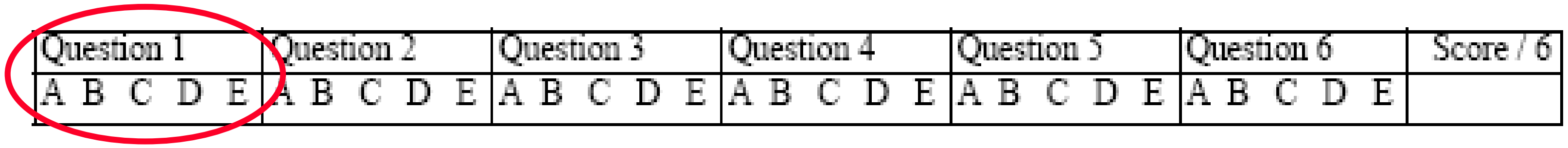
(A) there is a defect in the program

(B) you made a mistake determining the expected result

(C) the computer has made a mistake

(D) the requirements are not correct

[✓ (E) EITHER (A) or (B)]



Question 1	Question 2	Question 3	Question 4	Question 5	Question 6	Score / 6
A B C D E	A B C D E	A B C D E	A B C D E	A B C D E	A B C D E	

Steps in testing a programme

For each functional specification (use case)

1. Specify a set of inputs
2. **Determine the expected output or result**
3. Execute the software
4. Compare the result with expected
5. Determine Pass/Fail

What terms should we use?

- **Failure,**
- **Mistake,**
- **Error,**
- **Fault,**
- **Defect,**
- **Bug?**

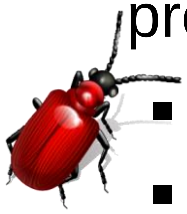
Terms used in connection with software testing

- A *defect*
  - any error or mistake, which may cause the program to not perform according to specifications.
  - may be the result of programmer error, or of incorrect or incomplete specifications
  - may not always cause the program to fail.
- A *failure*
  - occurs when the software does not behave as expected.

Classification of errors

- **Syntax error:** An error in the ‘grammar’ of the computing language – reported by the compiler during compilation, so not usually a problem

- **Runtime error**: A fault condition that causes execution of the program to halt prematurely (“bomb out”):
  - e.g. division by zero, square root of a negative number, non-numeric character entered where a number is expected, data file missing, printer turned off, memory overflow
  - A premature halt is bad, because it may leave the user stranded and it may also damage database
- **Logic error**: Produces incorrect output even though the program runs to completion.
  - May occur only for certain input values.
  - Can only be found by testing. Often called a ‘bug’.



The software defect cycle (1)

Programmer makes a <u>error</u> (mistake)

- Apart from carelessness, this may be caused by:
  - incomplete user specification
  - confusion about the proper way to calculate a value
  - incorrect database instruction
  - misunderstanding as to the internal state of the software.
- So, the programmer introduces a <u>defec</u>t into the code (often called a bug).
- Defects can sometimes be discovered through code review …
- … but, there may or may not be any physical manifestation of the error

The software defect cycle (2)

Code is sent to the tester

- **Tester executes the software that contains the defect:**
  - software behaves differently from what the tester expects (an anomaly). This is called a failure.
  - the defect is only manifest when a failure occurs.
- **Tester investigates the anomaly to determine, if possible, the nature and cause of the failure:**
  - failure may go beyond the obvious, immediately observable misbehaviour associated with the anomaly,
  - e.g. data might have been corrupted, another process improperly terminated, etc.
- **Tester prepares an incident report**
  - also called defect report, problem report, bug report, or issue

The software defect cycle (3)

The incident report is sent back to the programmer.

- **Programmer ‘debugs’ the program in order to repair the underlying defect**
- The bug fix comes back to the tester for confirmation
- Regression testing used to see that no new errors

A defect may remain undiscovered during testing but result in failure when the software is

- installed at the user site and in operation
- ported to a different hardware platform,
- enhanced or modified.

‘False positives’ in software testing

- Tester may observe an anomaly, which is not due to a defect.
  - For example caused by bad test data, improperly configured test environment, or a misunderstanding on the tester’s part.
- The programmer will investigate the anomaly to find what caused the apparent failure.
- There may be defects introduced other than by coding
  - For example, a business analyst incorrectly specifying requirements.
  - Ideally such defects should be detected by reviews.

Component or unit testing

Making sure that each component or unit works as expected and according to the specification

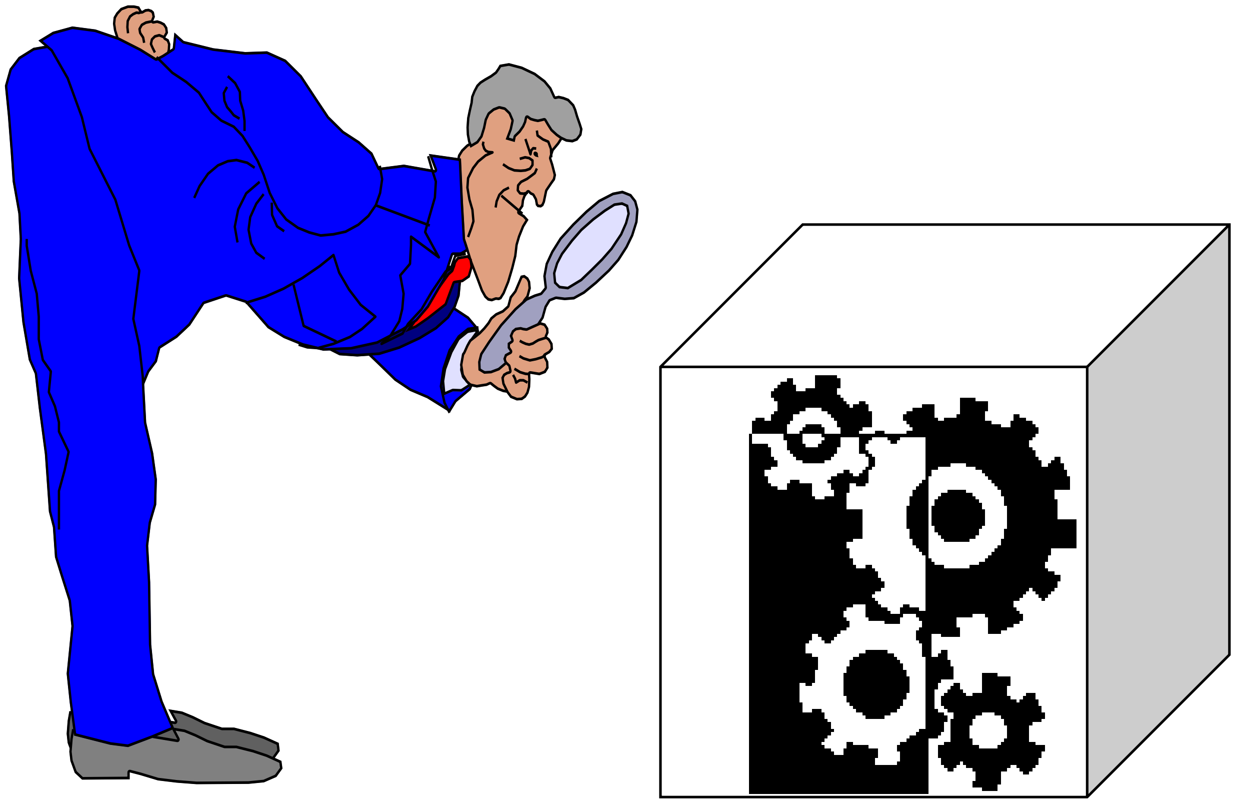
The secret of software testing is to use good test data

It is not sufficient to use a ‘scatter gun approach’

- Test data must designed with care.
- Random input data generated automatically has low probability of finding defects

Two types - White box testing

If we can “see” what’s going on inside ...



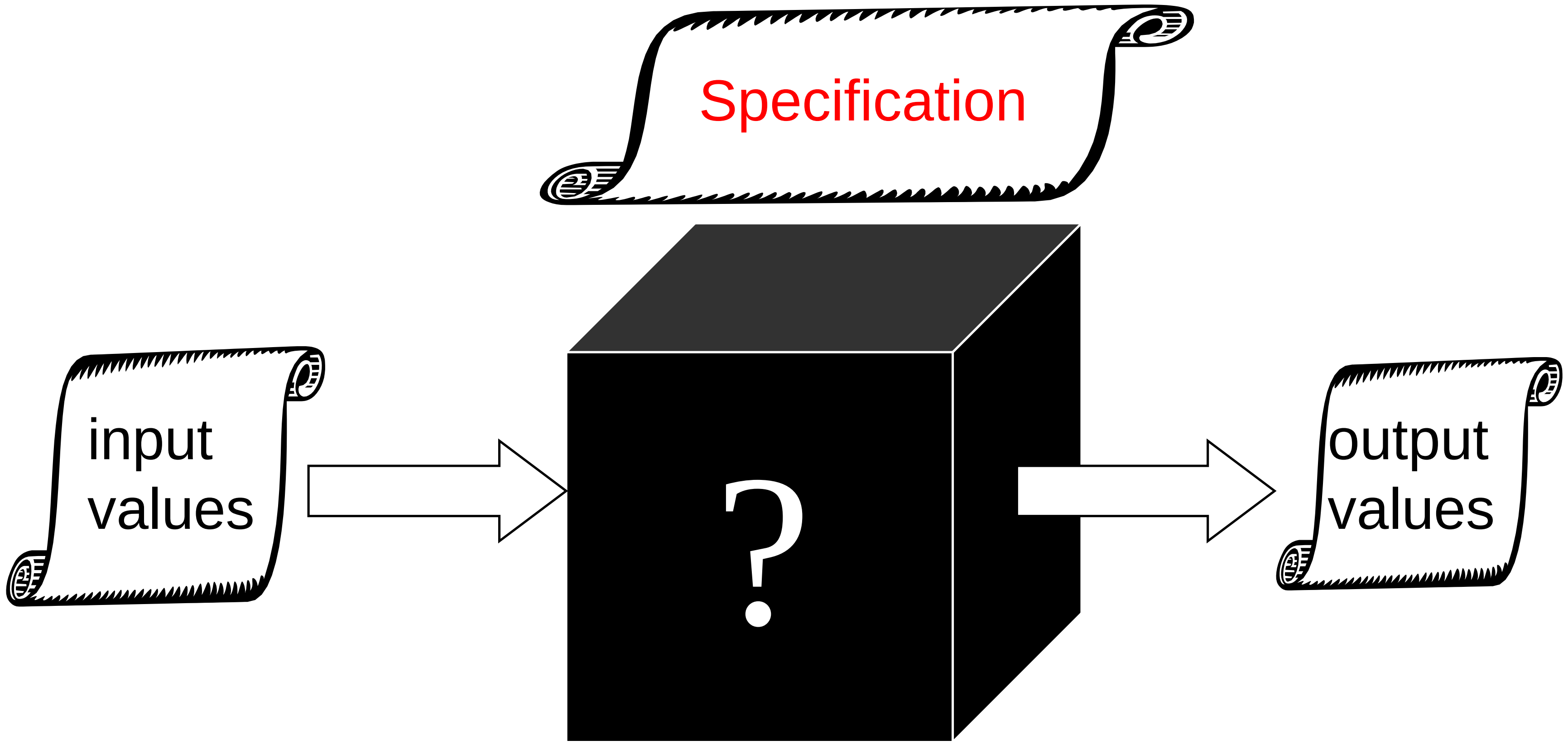
... then we know how best to test it!

“White-Box” testing

- Applicable only to individual programs
- Used by programmers when they are writing software
- Can be more insightful because the programmer knows about the internal structure of a program
  - devises tests that have the highest probability of causing the program to fail
  - tries to exercise all possible logic paths through the program

Black box testing

We don't know what's going on inside ...



... but, we do know what output we should get according to the specification

“Black-Box” (input-output) testing

- Assumes nothing about the inner structure of the program.
- Input data values should be ‘sensible’ according to the application domain and the client specifications.
- Expected outputs for each test input must be worked out in advance, according to specifications and ‘business rules’.
- ‘Outrageous’ input values can be used to test robustness of program

Test cases

- To define a 'test case' you must say
  1. What function is being tested [Function]
  2. The value(s) of the test input [Input]
  3. The result or output expected [Result]
     - output value
     - error message
     - further process
- One or more test cases are required for each function or rule specified by the client

A Test Scenario

- When testing a software system several test cases can be taken together in an appropriate <u>context</u> so that the users can imagine the software in action
- This is referred to as a ‘scenario’
- One or more scenarios will be used to test each functional area of the system.

Case Study

- ***Scenario 6.4***: Existing student enrolling in units of study
- ***Context***
- A student arrives at the counter to enrol. The student's name is Jane Harris, (Id = 7). She wishes to enrol in INFO5001 and INFO5990. She has a filled in enrolment form as supplied.
- ***Function***
- Enrolling in an existing unit of study.
- ***Input***
- Retrieve student record using either Id or Surname. Select combo-box. Available units are displayed including INFO5001and INFO5990 are displayed. Select INFO5001 from the list, press enter.
- ***Result***
- Subject description is displayed. Select combo-box for next line. INFO5001 no longer displayed

Q 2. Which of the following best describes a ‘test case’

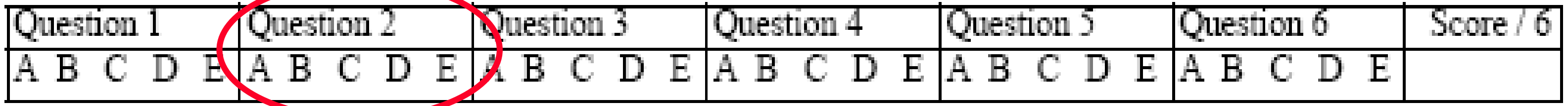
(A) A value specified by the client during requirements elicitation

(B) A set of data which can be input to test a program

(C) A complete story relating to the way users will employ a program

[✓ (D) A set of inputs together with corresponding outputs used to test a specific function]

(E) A test environment used for program testing



Question 1	[✓ Question 2]	Question 3	Question 4	Question 5	Question 6	Score / 6
A B C D E	A B C D E	A B C D E	A B C D E	A B C D E	A B C D E	

1 minute Stretch