GOVERNMENT OF WESTERN AUSTRALIA

TAFE WA South Metropolitan

Software Development Life Cycle and Software Testing

(Session 17)

Session 17

- Software Development Life Cycle
  - Planning
  - Requirements analysis
  - System design
  - Build
  - Test
  - Implementation
  - Maintenance
- Other System Development methods/tools
- White Box vs Black Box testing

Systems Development Life Cycle

- The Systems Development Life Cycle (SDLC) is a process used for mapping out the defined tasks required to develop IT-based projects.
- There are numerous SDLC methods, though they largely follow the same set of tasks.
- The tasks include:
  - Planning
  - Requirements analysis
  - System design
  - Build
  - Test
  - Implementation
  - Maintenance

Systems Development Life Cycle

- **Planning**
  - The main goal of the planning stage is to understand a business problem and what can be done about it.
  - This includes the type of response (develop a new system, enhance pre-existing systems, or do nothing) and feasibility studies to judge the solutions based upon the type of feasibility
  - For example, Economic feasibility: whether it is a sound investment

Systems Development Life Cycle

- **Requirement analysis**
  - Requirements analysis seeks to investigate the business problem the system will address.
  - This task aims at gathering requirements for the existing/to-be-created system.
  - These requirements can be broken down into two groups:
    - Functional requirements: what the system needs to be able to do.
    - Non-functional requirements: how the system needs to behave.

Systems Development Life Cycle

- **System Design**
  - System design seeks to address how the business problem will be resolved by the system.
  - This includes:
    - Technical specifications related to required hardware, software, procedures, and personnel required.
    - Desired system inputs, outputs, and user interfaces
    - Component integration planning
  - Overall, system design creates the planned blueprint we will follow to build our system.

Systems Development Life Cycle

- **Build**
  - The build task is as it sounds, we build the system based upon our previous steps.
  - The planned specifications are translated into code.

    (The fun part of your assessment)
- **Test**
  - Testing involves checking our code against the expected/desired results.
  - This step is also used to detect and rectify errors in our code.

Systems Development Life Cycle

- **Implementation**
  - Implementation involves introducing the system into the planned environment.
  - Where there are already pre-existing systems, this includes multiple potential methods of conversion between them.
- **Maintenance**
  - The implemented system is maintained for its period of relevance/support.
  - Types of maintenance include:
    - Updating: Changing the system to meet business-condition-based changes.
    - Adding new functionality: Expanding upon the existing functions of the system to meet new requirements/demands.
    - Debugging: Constant testing and fixing of bugs in the system.

Systems Development Life Cycle Contd.

- **Other System Development methods/tools**
  - Joint Application Design (JAD)
    - A group-based approach for requirements collection and system design creation.
  - Agile Development
    - A software development approach that utilised rapid iterations and constant communication, development, and testing.
    - These are generally better used for **smaller scale projects** in comparison to the SDLC approach.
  - Prototyping
    - Creates a scaled down version of the final product based upon user requirements and then seeks to refine it through further iterations and feedback.
  - Rapid Application Development (RAD)
    - A combination of JAD, prototyping, and computer-assisted software engineering to facilitate assisting each stage, especially the testing portion.

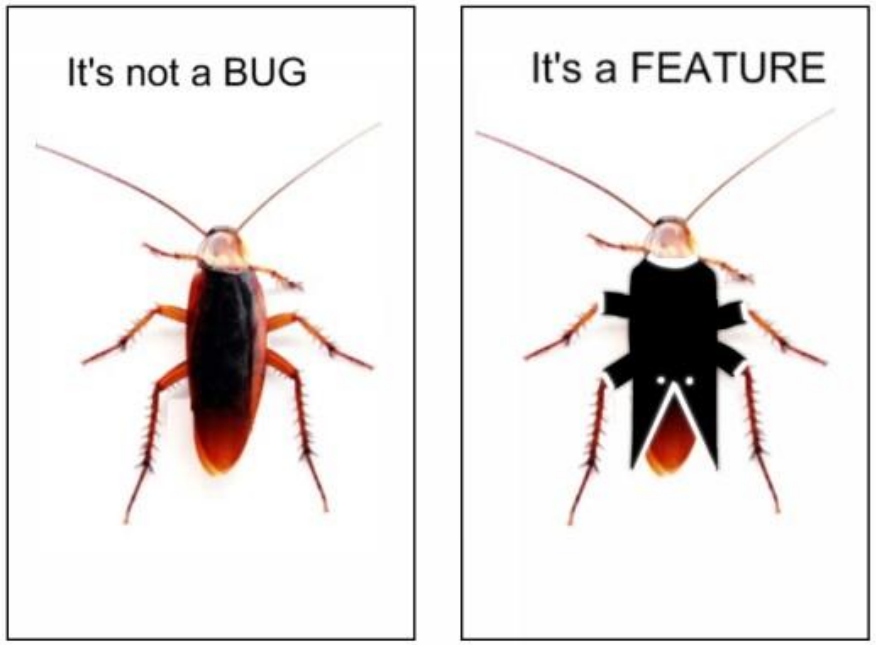
Testing

- Testing is the process of executing a program with the intention of finding errors; It is checking whether the code does what it should in a certain circumstance (we call these circumstances test cases).
- Can show the presence of bugs (the code acting in ways it shouldn’t).
- A good test case is one that has a high probability of detecting an undiscovered defect, not one that is likely to show that the program works correctly
- It is near impossible to test your own program in an unbiased manner
- A necessary part of every test case is a description of how the test case can be performed and the expected result.

What is a Bug?

- Defect
- Fault
- Problem
- Error
- Incident
- Anomaly
- Variance

- Failure
- Inconsistency
- Product Anomaly
- Product Incidence
- Feature!



Defective Software

- We often develop programs that contain defects
  - It happens all the time – don’t worry too much
  - What's important is that you know how many and what kind of defect are occurring
- As we continue to develop software and learn, our programs will be significantly better than the last

White Box vs Black Box Testing

- White-box testing:
  - written with knowledge of the implementation of the code being tested.
  - focuses on trying to cover all code paths/statements
  - examines the program structure and considers the internal working of the software (code, logic, structure)
  - carried out by software developers, for instance, unit testing and logic testing
- Blackbox testing:
  - form of testing where you test the input/output to code/parts of code, without knowledge of the internal code structure.
  - considers the external behaviour of the system (that the system as a whole is working as expected, for instance acceptance testing and functional testing)
  - not consider for algorithm testing

Let's try it – White Box Testing

- Our real code:

```
aNum=float(input("Enter a number: "))
if aNum >=10:
    print("Your number is bigger than 10")
elif aNum < 0:
    print("Your number is less than 0")
```

- Line coverage:
  - 2 test cases
    - User enters in number >=10
    - User enters in number <0

Questions?