Software Testing

Introduction

Introduction

- Software is one of the most complex things built by humans.
- To get it right you need to consider:
  - will the software produce the correct results?
  - will the software work with edge cases like no data?
  - will the software have the required efficiency?
  - will the software fail under high loads?
  - will the software do something sensible when it receives bad data?
  - has the software been integrated with the rest of the system so that there are no interations which cause errors?
  - if there is an unanticipated error, can the software recover and return itself to a stable state?

Software Testing

- Software testing is the process of
  - testing your software to see if it works and
  - meets all of its performance
  - and load requirements.

What Does Testing Prove?

- Testing proves that the tests you ran work correctly.
- It does not prove that your software is correct.
- Your software could pass all tests with flying colours yet still have undiscovered bugs in it.
- The amount of bugs discovered is proportional to the amount of time spent debugging.

Debugging

- Testing to determine if there are problems is only part of the process.
- After you determine that the software is incorrect, you need to find and fix the problem.
- This can be a highly complex and involved process that can take a large amount of time.

The Debugging Process

- The process of debugging can be broken down into smaller steps:
  - locate the source of the bug,
  - determine how to fix the bug,
  - implement the solution.

Testing Versus Debugging

- Testing and debugging are not the same thing.
- Testing runs the software and determines that the results are correct.
- Debugging is the process of trying to locate and fix a problem in code that you know has a bug in it.

The Need for Testing

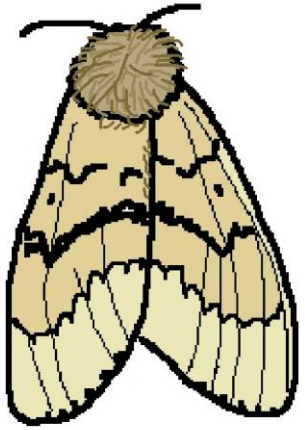
- Untested software usually fails at some point.
- Only testing can give us confidence that the software might be usable.
- Testing does not prove the software is correct.

Famous Software Bugs

- The history of software is littered with some of its greatest failures.
- By looking at these, we will see how even professionals can fail

The First Bug

- The first bug was discovered some 70 years ago by Grace Hopper, who was the developer of the COBOL language.
- She was using the Harvard mark II computer and discovered that a calculation was not being performed correctly and the problem was tracked down to a moth which was stuck between relays inside the computer.
- As a result of this, all problems associated with software have been described as bugs.



Therac-25

- Therac-25 was a machine to treat cancer with radiation
- It was built by Atomic Energy of Canada Ltd.
- The original version used hardware stops to prevent errors.
- The revised version used software to prevent errors.
- There was a bug in parallel programming that caused several deaths.
- At first, they thought it was an electrical problem as software cannot fail.
- The programmer did not understand how to do parallel programming.

The Ariane 5 Disaster

- In June of 1996 the very first Ariane 5 rocket was launched.
- 37 seconds after launch, the rocket rotated 90 degrees, in the wrong direction.
- less than two seconds later started to break up due to aerodynamic forces.
- This triggered the Rockets self-destruction mechanism causing it to explode in midair.
- This disaster cost approximately $370 million.
- A 64 bit floating point value was used to track a guidance variable.
- at some point in the calculation the value was assigned to a 16 bit integer.
- This worked for a few minutes until the value got too large.
- It then sent incorrect values to the guidance system.

Types of Testing

- Unit testing
- Black box
- White box
- Integration
- Functional
- End-to-end
- Load
- Security
- Acceptance
- Regression