Lecture 1 Introduction Software Testing

Agenda

– Software Engineering

– Software Testing

– Unit Testing

Testing in Software Engineering

Why Software Engineering?

Need to build high quality software systems under resource constraints.

• Social

• Satisfy user needs (e.g., functional, reliable, trustworthy)

• Impact on people’s lives (e.g., software failure, data protection)

• Economical

• Reduce cost; open up new opportunities

• Average cost of IT development ~$2.3m, ~$1.3m and ~$434k for large, medium and small companies respectively

• Time to market

• Deliver software on-time

– Software Quality Assurance

– Ensuring software under development have high quality and creating processes and standards in organization that lead to high quality software

– Software quality is often determined through Testing

Why software testing?

– Software development and maintenance costs

– Need to develop functional, robust and reliable software systems

– Human/social factor - society dependency on software in every aspect of their lives

• Critical software systems - medical devices, flight control, traffic control

– Meet user needs and solve their problems

– Small software errors could lead to disasters

What is software testing?

– Software process to

– Demonstrate that software meets its requirements (validation testing)

– Find incorrect or undesired behavior caused by defects/bugs (defect testing)

– Part of software verification and validation (V&V) process

Testing (Levels)

Unit / Functional Testing: The process of verifying functionality of software components (functional units, subprograms) independently from the whole system

Integration Testing: The process of verifying interactions/communications among software components. Incremental integration testing vs. “Big Bang” testing

System Testing: The process of verifying the functionality and behaviour of the entire software system including security, performance, reliability, and external interfaces to other applications

Acceptance Testing: The process of verifying desired acceptance criteria are met in the system (functional and non-functional) from the user point of view

Software Verification and Validation

– Software testing is part of software Verification and Validation (V&V)

– The goal of V&V is to establish confidence that the software is “fit for purpose”

– Software Validation

– Are we building the right product?

– Ensures that the software meets customer expectations

– Software Verification

– Are we building the product right?

– Ensures that the software meets its stated functional and non-functional requirements

Unit Testing

Why Unit Testing?

– Change and maintain code at smaller scale

– Discover defects early and fix it at cheaper costs

– Ease debugging

– Code reusability

– Reduce integration testing

How?

1. Design test cases

2. Prepare test data

3. Run test cases using test data

4. Compare results to test cases

5. Prepare test reports

Designing Test Cases

– Effective test cases show:

– The unit under test does what it spoused to do

– Reveal defects in the unit, if there is any

– Design two types of test cases

– Test normal operation of the unit

– Test abnormality (common problems/defects)

Choosing Test Cases – Techniques

– Partition testing (equivalence partitioning)

– Identify groups of inputs that have common characteristics

– From within each of these groups, choose tests

– Use program specifications, documentation and/or experience

– Guideline-based testing

– Use testing guidelines based on previous experience of the kinds of errors often made

– Test Fixture

– The context for testing

• Usually shared set of testing data

• Methods for setup those data

• E.g., a fixed string (test fixture), which is used as input for a method