Contents

1 Safety and releaiblity

Computing system errors stem from human errors. Errors are typically **traceable**, but moral and legal responsibility is **not always easy to establish**

2 Causes of Failure in Computing Systems

- Hardware errors
- Software errors
- Solving the wrong problem
- Misuse of the system
- Human communication failure
- Human malice

3 Dependability

Dependability of a computing system is the ability to deliver service that can justifiably be trusted. Dependability attributes are:

- Availability: Readiness of correct service
- Reliability: Continuity of correct service
- Safety: Absence of catastrophic consequences on the user(s) and the environment
- Confidentiality: Absence of unauthorized disclosure of information
- Integrity: Absence of improper system state alterations
- Maintainability: Ability to undergo repairs and modifications.

4 Software Engineering

Discipline focussed on the production of software and development of tools methodologies and theories supporting software production.

- **Specification**: Defining the functions to be performed by the software
- **Development**: Producing the software that meets the specifications
- **Validation**: Testing the software
- **Evolution**: Modifying the software to meet the changing needs of the customer

4.1 Specification

4.1.1 Process

- Determine the requirements of the system and constraints under which the software must operate
- Assess feasibility of the software development within the budget and schedule requirements

4.1.2 Result

- High level statement of requirements
- Mock-ups of the user interface
- Low-level requirements statement.

4.2 Development

First design phase based on high-level, abstract view of the system

- Reveals ambiguities, omissions, and errors in the specifications
- Mistakes are corrected; that is less expensive at a higher, abstract level

Subsequent design phases add levels of details

- Components of the system become clear
- Interface between each components are spelled out
- Algorithms are selected and data structure defined.

4.3 Validation

Software testing

- Process to assess the correctness, completeness and quality of developed computer software. Validate that software satisfies the specification and meets user needs.

Testing can reveal bugs but cannot prove that the program **will work correctly under all circumstance**.

- Formally proving that the software meets specifications may be unfeasible and is typically expensive
- Even if we prove that the program is correct and meets specifications that does not mean that specifications are correct.

4.3.1 Principles

- Testing shows presence of defects
- Exhaustive testing is impossible, aim for optimal amount of testing based on the risk assessment.
- Defect clustering: The risk is determined based on experience and critical thinking. In practice, a small number of modules contain most of the defects detected.
- Pesticide paradox: If the same tests are repeated they can no longer find new bugs. Tests need to be reviewed, revised, and changed.
- Testing is context dependent
- Absence of errors is fallacy
- Early testing is critical: Testing for bugs is useless if the system does not meet the users requirements. Testing should start as early as possible to ensure that it meets user needs.

4.4 Evolution

- Successful software systems evolve over time to meet the changing needs of the users
- Development of new software versions
  - New needs are compared with the strengths and weaknesses of the system
  - Decision is made about the changes to the system
  - Often the same CASE tools are used to support the development of the new version
  - Previous test suites may be reused and expanded to test the new version of the system.

5 Simulations

5.1 Computer simulations

Simulate physical experiments which may not be possible to run in real setting. Design of nuclear weapons, oil drills, fuel-efficient cars, etc. Uses of simulations

- of past events: astrophysicists derive theories about the evolution of the universe
- to explore and understand world around us: drilling oilsimulations made the process much more predictable
- to predict the future: weather predictions

5.2 Testing simulations

Simulations may be run **continuously, alongside** the use of the production system. The objective is to identify possible problems **before** they are detected in the real situations and prevent them. Simulations may produce erroneous results due to error in the **implementation** (software bug) or a **flawed model** upon which the simulation program is based.

6 Criteria for moral responsibility

- The action of the person **must have caused the harm** or been a significant causal factor.
- The person must have **intended or willed** the harm, or it must be a result of his or her **negligence**, carelessness or recklessness.
- The person must have been able to have known, or must **know of the consequences of the action**, or must have **deliberately remained ignorant** of them.

Reference section

placeholder