Umaru Musa Yar'adua University Katsina

Faculty of Natural & Applied Science

Department of Computer Science

CSC4311 SOFTWARE ENGINEERING

LECTURE FOUR

LECTURER: DR. AMINU AMINU MU'AZU

SOFTWARE TESTING

Overview of Testing

Software is everywhere

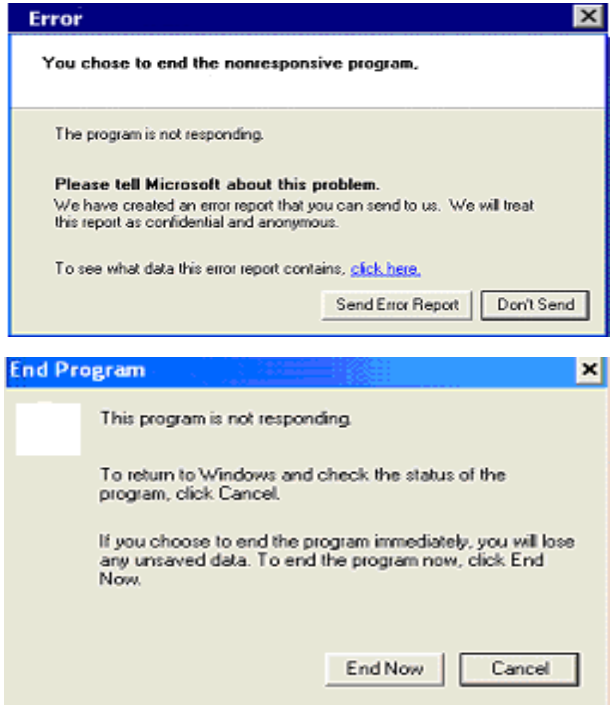
- Software is part of our lives.
- We use software everywhere i.e. our mobile phone, washing machine, air bag controller etc.
- Why do we choose for software rather than hardware whenever possible?
    - Easy to modify
    - Does not wear out
- Our dependencies on software raise issues on quality and reliability.

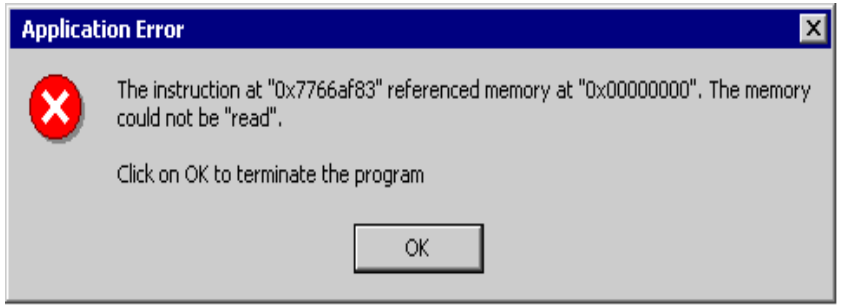
In God We Trust, The Rest We Test !!!

The need for software testing…

- Software failures can lead to terrible consequences

- Loss of data …..





- Loss of fortune …..

- Loss of lives …..

Why do faults occur in software?

- software is written by human beings
  - who know something, but not everything
  - who have skills, but aren't perfect
  - who do make mistakes (errors)
- under increasing pressure to deliver to strict deadlines
  - no time to check but assumptions may be wrong
  - systems may be incomplete
- if you have ever written software ...

why is testing necessary?

- because software is likely to have faults ✓
- to learn about the reliability of the software ✓
- to fill the time between delivery of the software and the release date ✗
- to prove that the software has no faults ✗
- because testing is included in the project plan ✓
- because failures can be very expensive ✓
- to avoid being sued by customers ✓
- to stay in business ✓

Testing Terminology

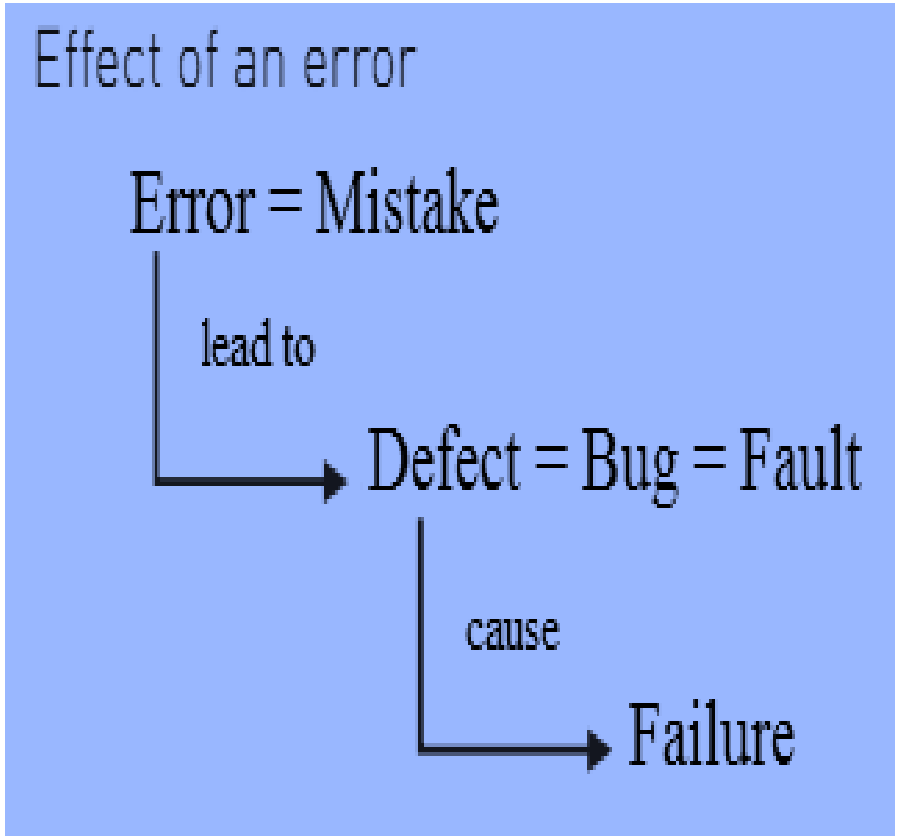
Software Testing - definition

- Is a sequence of processes, that was designed to make sure the software does what it was designed to do and that it does not do anything unplanned.

- A process of executing a software program with the goal of finding errors.

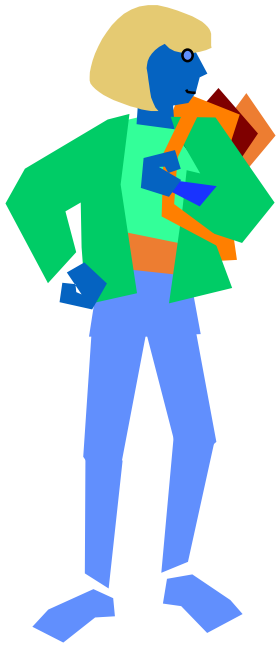
What is “error”, “bug”?

- Error: A human action that produces an incorrect result.
- Defect: An appearance of an error in software.
  - Also known as a defect or bug.
  - If executed, a fault may cause a failure.
- Failure: Deviation of the software from its expected delivery or service.

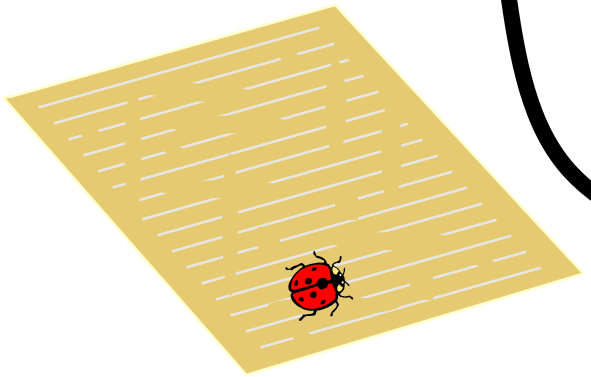


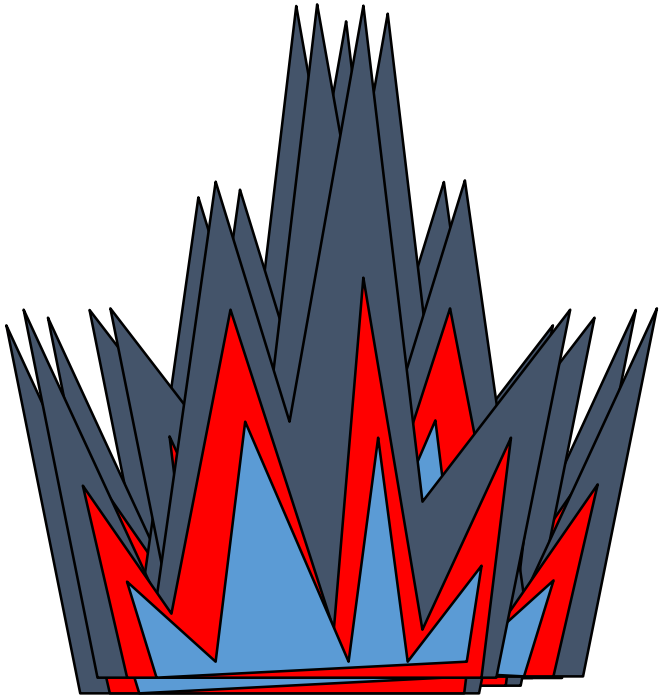
Error - Defect - Failure

A person makes an error ...



... that creates a defect in the software ...



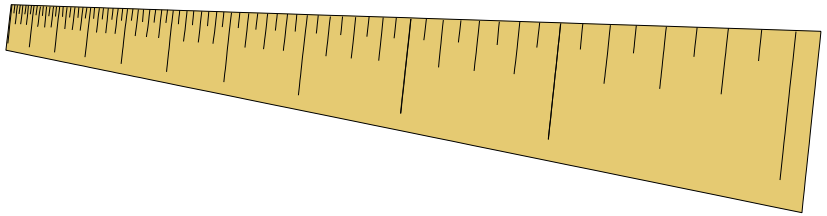


... that can cause a failure in operation

Reliability versus Faults

- **Reliability**: The probability that software will not cause the failure of the system for a specified time under specified conditions.
  - Can a system be fault-free? (zero faults, right first time) ✗
  - Can a software system be reliable but still have faults? ✓

Testing and quality



- Testing measures software quality
- Testing can find faults; when they are removed, software quality (and possibly reliability) is improved
- What does testing test?
  - System function, correctness of operation
  - Non-functional qualities: reliability, usability, maintainability, reusability, testability, etc.

- Quality software is **reasonably bug-free**, **delivered on time** and within budget, **meets requirements and/or expectations**, and is **maintainable**

Quality Products

Psychology of Testing

Behaviors of a Good Testers

- A Tester needs:
  - Good communication skills
  - Good observation skills
  - People treatment skills
  - Patience
  - Reliability
  - Creativity in terms of identifying likely faults
  - Experience

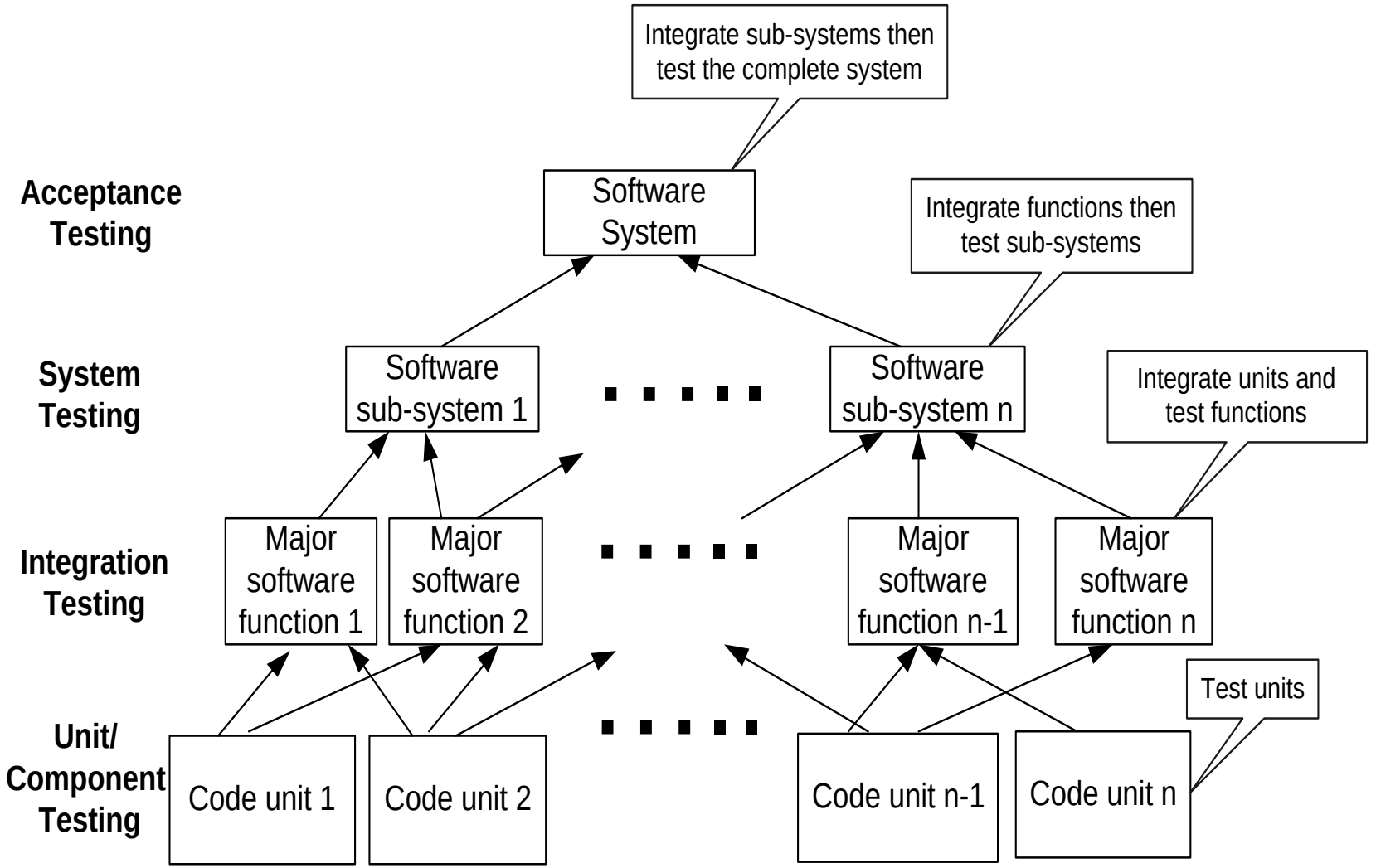
Developer vs Tester Relationship

- The relationship between a Developer and a Tester is not normally an easy one because:-
  - testers point out problems with software
  - developers like to think their software is perfect
  - testers are perceived as delaying the project by finding faults in the system.

- It is important that they work together
- It is also important that they have mutual respect for each other.
- Collaboration is the right approach – we work to a common goal!

Levels of Software Testing

- In developing large software system, testing usually involves several levels consisting of unit testing, integration testing, system testing and acceptance testing



Unit/Component Testing

- individual units of a software are tested.
- The purpose is to validate that each unit of the software performs as designed.

Integration Testing

- individual units are combined and tested as a group.
- The purpose of this level of testing is to expose faults in the interaction between integrated units.

System Testing

- Here a complete, integrated system is tested.
- The purpose of this test is to evaluate the system's compliance with the specified requirements.

Acceptance Testing

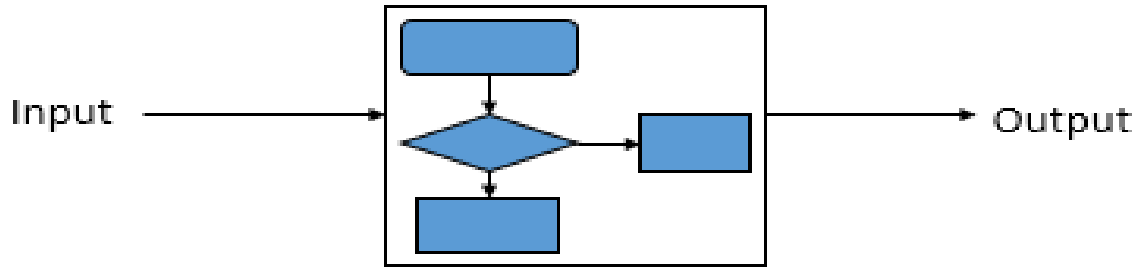
- Here a system is tested for acceptability.
- The purpose of this test is to evaluate the system's compliance with the business requirements and judge whether it is acceptable for delivery.

Types of testing

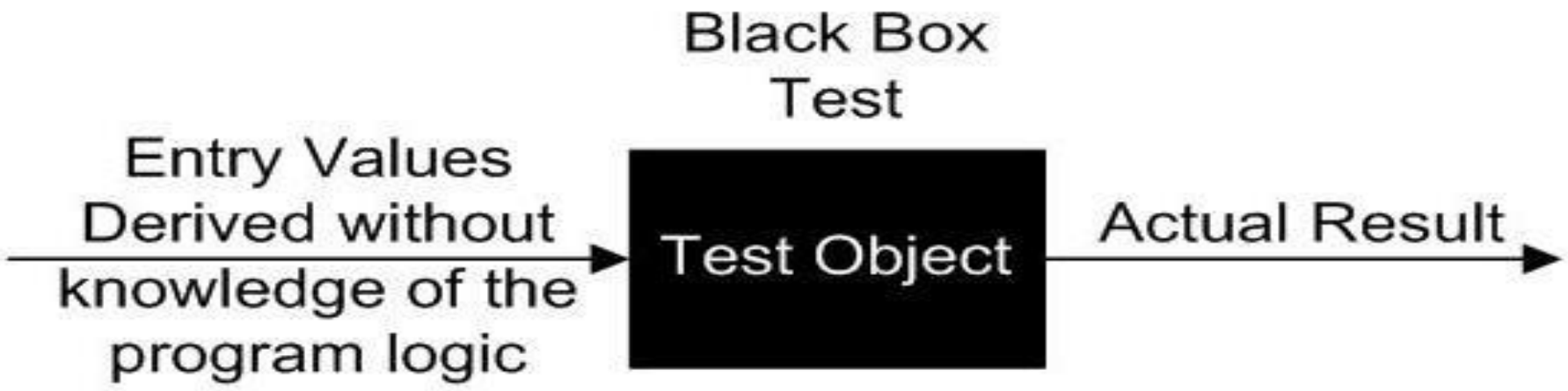
- Black Box Testing
  - testing without knowing the internal workings of the code
  - WHAT a system does, rather than HOW it does it

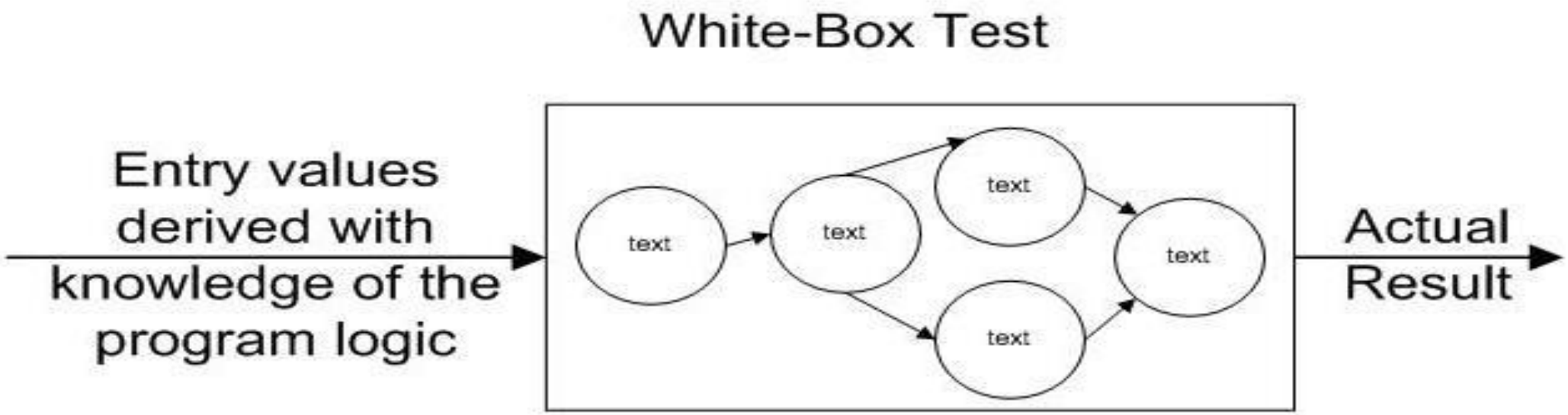
If Output = Expected result then pass

- White Box Testing
  - testing based upon the structure of the code
  - typically undertaken at Component and Component Integration Test phases by development teams



Black-Box vs White-Box



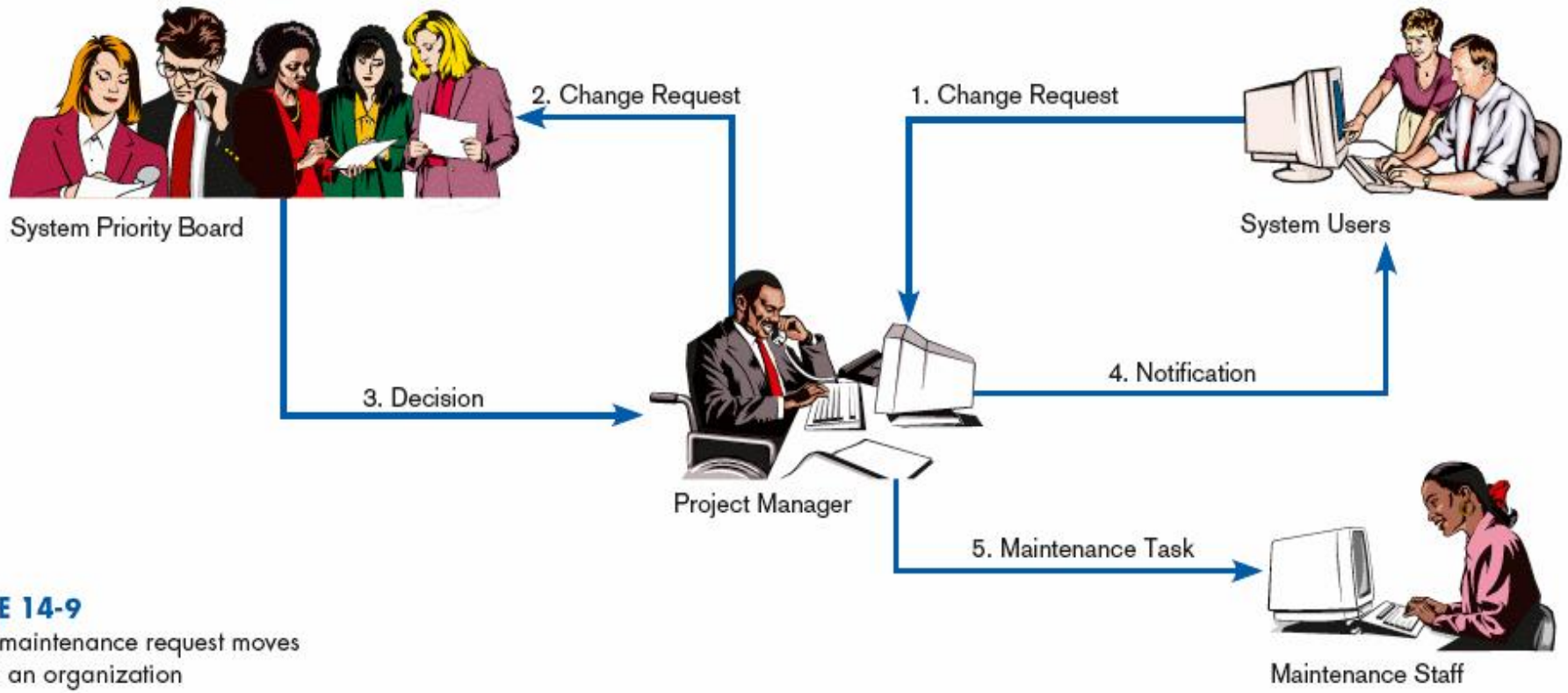


SOFTWARE CONFIGURATION MANAGEMENT

Software change

- **Software Configuration is about managing the change that happen during the software development process.**

Why change happens?



URE 14-9
a maintenance request moves
ugh an organization

- **Processing failure**
- **Processing inefficiency**
- **New requirements**
- **Performance enhancement**

An Aircraft Example

- Passenger aircraft are one of the most complex engineering feats to be attempted by a private corporation.
- They must be safe and reliable, and they must be economical. Unlike trains or cars, the aircraft systems cannot simply be shut down in the event of a failure.
- Airlines need to post a profit and pay dividends to their stockholders.
- These requirements result in very complex designs that include many components. A Boeing 747, for example, is composed of more than 6 million parts.

An Aircraft Example

- To develop and maintain such complex systems, while remaining economically competitive, aircraft manufacturers extend the life of a model by incrementally improving on the same design.
- The Boeing 747, for example, was first released in 1967 and is still in production It has gone through four major revisions, the latest, the 747-400
- Another approach to dealing with the high complexity of aircraft is the reuse of as many parts and subsystems across aircraft models as possible.

Software Configuration

- Although not quite as complex as a passenger aircraft, software system development nevertheless suffers from similar problems.

- Maintenance changes to existing and running software systems must be controlled and evaluated carefully to ensure a certain level of reliability.

Running Example

- Shortly after the A320 was certified in February 1988, a revised version, the A320-200, was recertified.
- The new version included a few modifications one of them was the addition of winglets at the end of the wings that significantly reduced drag and thus reduced fuel consumption. Fuel consumption is a major cost for the airline operator.
- During the A320 design, a change request was issued, describing the winglet change, its performance evaluation, and its estimated cost. The change was approved, implemented, and the A320-200 passed certification in November 1988.

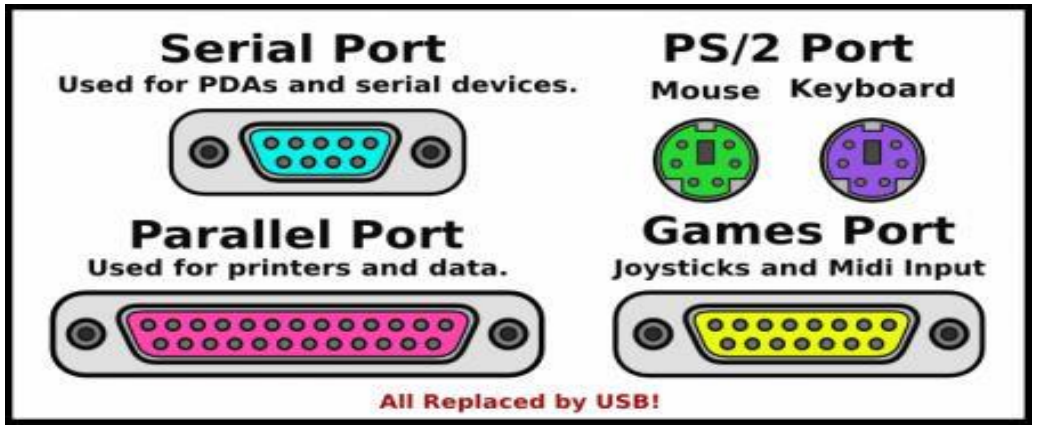
Definitions

- **CM is the discipline used in the (SDLC) to identify and control components associated with the system.**

- **The primary objective of CM is to ensure that changes to these components which may include requirements, design, code, documentation or hardware are controlled**

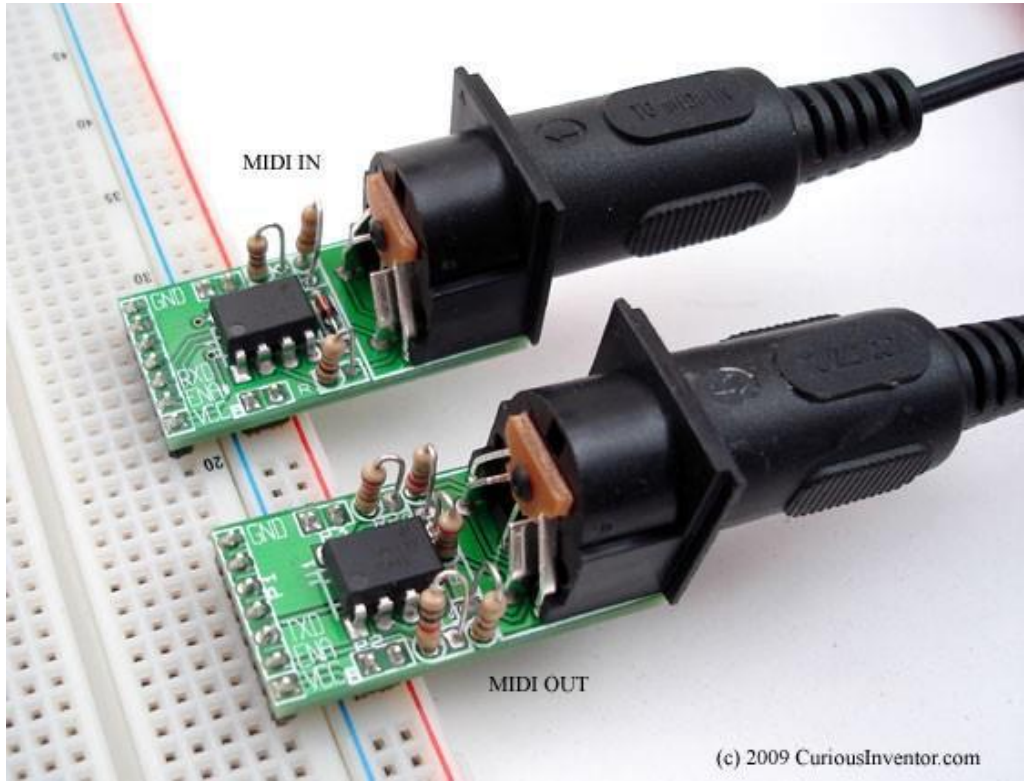
Computer hardware as example of SCM

- SCM is the organization of the components of a software system so that they fit together in a working order.



- When the hardware is manufactured, the interfaces that are essential for the operation of the final system are easily seen. Therefore, they are well known and are carefully examined whenever changes are made to the hardware design.

Computer hardware as example of SCM



Car models example:

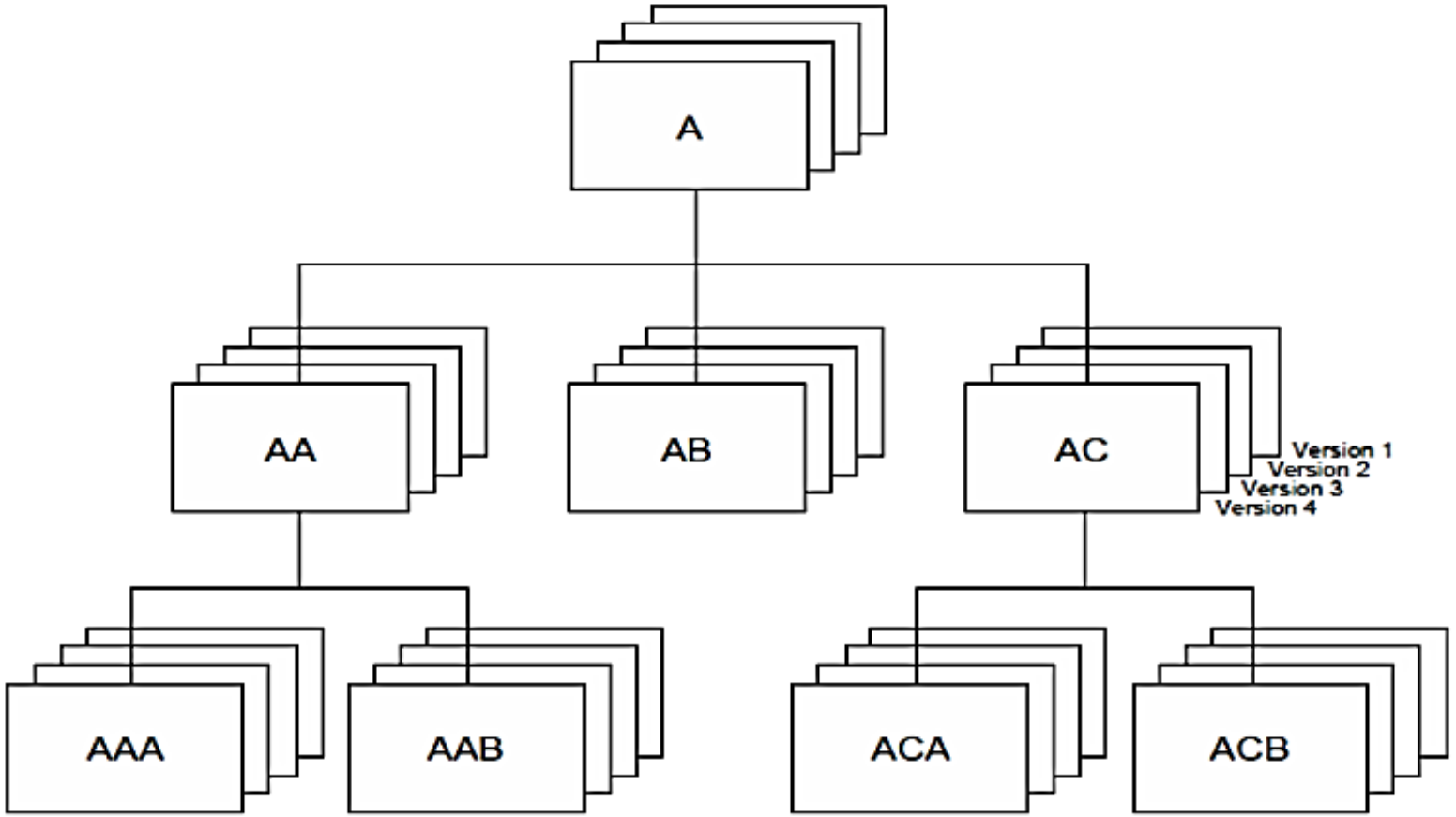
- The model year for a car is a version number of that car.
  - It is a Mitsubishi Pajero Wagon.

2003

1993

1990

history of changes



The Purpose of SCM

- *To establish and maintain the integrity of the products of the software project throughout the project's software life cycle.*

- *SCM involves **identifying configuration items** for the software project, **controlling** these items and **changes** to them, and **recording** and **reporting** status and change activity for these CI's.*

Integrity Means

- Having all the right parts fit together correctly to form the correct product or system.

What is a *configuration Item*?

- The configuration items (CI) are those “things” that represent internal and external project deliverables.

- Example of CI’s are:
  - Source code
  - Test plan
  - Test data
  - Support software (part of the product)
  - User manual
  - Requirements Analysis Document(RAD)
  - System Design Document (SDD), etc.

END OF SOFTWARE ENGINEERING

REFERENCES

- Douglas Bell. 2005. “Software Engineering for Students A Programming Approach”. Fourth Edition.
- Sommerville Ian. 2000. “Software Engineering”. 6th Edition.
- Mall Rajib, Fundamentals of Software Engineering, PHI.
- Pressman, Software Engineering Practitioner’s Approach, TMH.