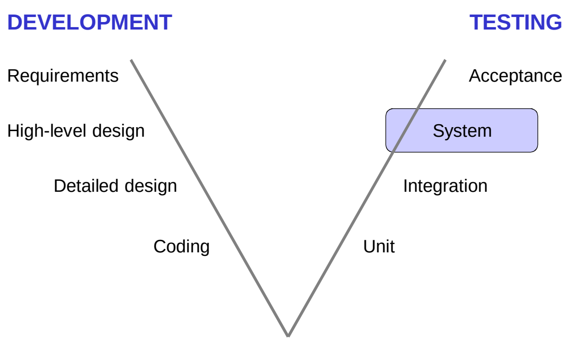
14. System testing

Scope and definitions
Requirements based testing
Functional and non-functional aspects
Management issues

Life cycle model for testing



Integration testing (R)

- Process of examining how the pieces of a system (Modules) work together, especially at the interfaces
- Tests to ensure that the various components of a system
  - Interact correctly
  - Pass data correctly
  - Function cohesively

System testing - scope

- System testing activities assume that module and integration testing have been done
  - i.e. single program functionality has been comprehensively tested
- System-level testing focuses on the testing of
  - Multiprogram functionality
  - External interfaces
  - Security
  - Recovery
  - Performance
  - Operator and user procedures

System testing - definitions

The process of testing of an integrated hardware and software system to verify that the system meets its specified requirements.

Verification: Confirmation by examination and provisions of objective evidence that specified requirements have been fulfilled.

IEEE definitions

System testing – definitions (cont)

- Testing to confirm that all code modules work as specified and that the system as a whole performs adequately on the platform on which it will be deployed.
- Testing conducted on a complete, integrated system to evaluate compliance with specified requirements.

System Testing vs Integration Testing

Integration testing	System testing
From interface specification	From requirements specification
Visibility of the integration structure	No visibility of code
Some scaffolding required	No drivers / stubs required
Attention to interfaces between the modules	Attention to system functionality

Key features of the Software Requirements Specification (IEEE)

- **Functionality**
  - What is the software supposed to do?

Functional

- **External interfaces**
  - How does the software interact with people, the system's hardware, other hardware, other software?
- **Performance**
  - What is the speed, availability, response time, recovery time, etc. of various software functions?
- **Attributes**
  - What are the portability, correctness, maintainability, security, etc. considerations?
- **Design constraints imposed on implementation**
  - Are there any required standards in effect, implementation language, policies for DB integrity, resource limits, operating environment etc?

Non-functional

System testing

- **Non-functional aspects**
  - Load
  - Stress
  - Performance
  - Volume
  - Security
  - Usability
  - Storage
  - Installability
  - Documentation
  - Recovery

Load testing

- Subjecting a system to a statistically representative (usual) load

  The load – an incoming transaction stream
- Aimed at determining various characteristics of the application working under a massive workload
- Consists of simulating real-life workload conditions for the application under test
- Test is normally performed several times with a different number of users working concurrently to find out how various parts of the application react to the increasing load

Stress testing

- Subjecting a system to an **unreasonable** load while denying it the resources (e.g., RAM, disc, mips, interrupts, etc.) needed to process that load
- Aimed at determining the load (for example, the number of user requests) that causes a crash or an unacceptable performance of the application under test

Stress testing (cont)

- A system is stressed to the breaking point in order to find bugs that will make that break potentially harmful
- The system is not expected to process the overload without adequate resources, but to behave (e.g., fail) in a decent manner (e.g., not corrupting or losing data)
- Bugs and failure modes discovered under stress testing may or may not be repaired depending on the application, the failure mode, consequences, etc.

Performance testing

- Concerned with checking that the system meets its **performance requirements**, e.g.
  - Number of transactions processed per second
  - Response time to user interaction
  - Time to complete specified operations
- Carried out on a working system under realistic conditions to identify performance problems
- Generally requires some logging software to be associated with the system to measure its performance
- May be carried out in conjunction with stress testing using simulators developed for stress testing

Volume testing

- Performed to find weaknesses of the system with respect to its handling large amounts of data over time
- Aims at verifying that the application performs correctly and is usable with **production volumes** of data
- Volume tests subject the system application with heavy volumes of data to ensure it can handle the volume of data
- If the performance does not measure up to the requirements / expectations, profiling tools are used to find the performance bottlenecks in the system.

Security testing

- Concerned with checking that the system and its data are protected from accidental or malicious damage
- Aimed at breaking the system's security
- Unlike other types of testing, difficult to test by planning system tests - the system must be secure against unanticipated as well as anticipated attacks
- Security testing may be carried out by inviting people to try to penetrate the system through security loopholes

Usability testing

- The process by which the human-computer interaction characteristics of a system are measured, and weaknesses are identified for correction
- Aimed at ensuring that the intended users of a system can carry out the intended tasks efficiently, effectively and satisfactorily
- Uses scenarios, questionnaires, user activity logs, inspections, etc.

Other system tests

- **Storage testing**
  Storage testing is for whether the system meets its specified storage objectives
- **Installability testing**
  Tests whether the software be installed correctly and all potential errors are handled tidily
- **Documentation testing**
  Tests work (of the technical authors) that is sent out with software. This covers installation guides and manuals.
- **Recovery**
  Tests whether the system can be recovered from failure. All levels of failures need to be counted (e.g. windows freezing to complete loss of the system and a complete failure of the environment).

System testing

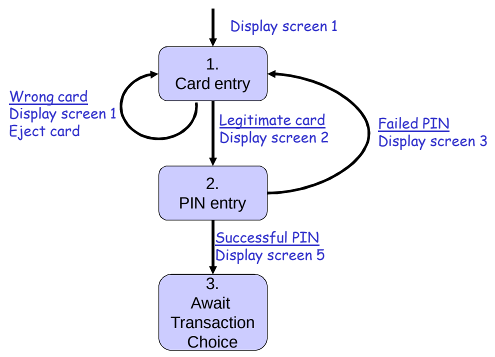
- Functional aspects
  - Use the system requirements specification and the user requirements specification to derive test cases
  - Based on user profiles (scenarios, use cases)

Test cases for functional testing

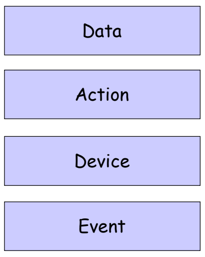
- Based on requirements / user specifications
- Functional aspects can be generated from a system model part of the specification
- Examples of system models
  - Context diagram
  - Data flow diagram
  - Entity / relationship model
  - Finite state machine model
  - …

Example

Top level Finite State Machine for an ATM terminal, PIN entry function



System models and testing strategies



System models and testing strategies

- **Data**
  - Focus on information created and used by the system and its flow
  - Entity – relation models
  - Data structure diagrams
  - ATM example
    - Customer accounts
    - Account balance
    - PIN

System models and testing strategies

- **Actions**
  - Focus on actions, invoked by either a user or a system
  - Input – output scenarios
  - Suited to imperative style models
  - System diagrams
  - Finite state machine models
  - ATM example
    - Display screen 1
    - Enter the card
    - Check whether the card is legitimate
    - If Yes then display screen 2 (PIN entry)
    - If Not then display screen 3 (Wrong Card) and eject the card

System models and testing strategies

- **Devices**
  - Focus on the behaviour of the input and output devices
  - Devices can be physical (keyboard, screen) or logical (network ports)
  - ATM example
    - All the inputs generated by button B1
    - All the actions and events centred around display screen 3

System models and testing strategies

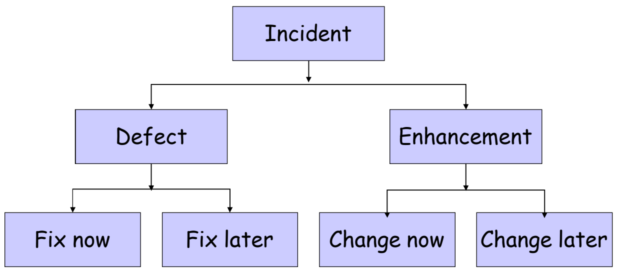
- **Events**
  - Focus on context-sensitive physical-to-logical “translations”
  - Involve data and action
  - Discrete (key strokes)
  - Continuous (temperature, altitude)
  - ATM example
    - Pressing button B1 in screen 5: Balance display
    - Pressing button B1 in screen 10: YES

Scenarios

Event-based scenario

Input events	Output events
Legitimate card	Display screen 2
Wrong card	Display screen 1
Correct PIN	Display screen 5
Incorrect PIN (first 3 times)	Display screen 3
Incorrect PIN (fourth time)	Display screen 4

Testing process management



Next lecture

Managing the testing process

Further reading

- Craig and Jaskiel “Systematic Software Testing, relevant sections of Chapter 4
- Other books on testing have good coverage of System Testing
- Study the IEEE Standard 830-1998 on Recommended Practice for Software Requirements Specification (this is important because System Testing cases are derived from SRS). The standard is available from the Web resources page
- Study a Beavers Primer on Requirements Engineering

Homework

- In the lecture you have seen an example of the Event-based scenario derived from the Finite State Machine (FSM) diagram for the PIN entry function. Develop testing scenarios based on
  - Data
  - Action
  - Device

  derived from the same FSM. Use any suitable data item, action and device for your example. Make up any items that are missing from the ATM system description (in the handout).