System Testing

- Performed after the software has been assembled.
  - But must be planned before

- Test of entire system, as customer would see it.
System Testing

- Check if system satisfies requirements for:
  - Functionality
  - Reliability
  - Recovery
  - Multitasking
  - Security
  - Compatibility
  - Stress
  - Performance
  - Ease/Correctness of installation

System Testing

- Acceptance Tests
  - System tests carried out by customers or under customers’ supervision
  - Verifies if the system works according to the customers’ expectations

- Common Types of Acceptance Tests
  - Alpha testing: end user testing performed on a system that may have incomplete features, within the development environment
    - Performed by an in-house testing panel including end-users.
  - Beta testing: an end user testing performed within the user environment.
Functional Testing

- Ensure that the system supports its functional requirements.

- Test cases derived from statement of requirements.
  - traditional form
  - use cases

Deriving Test Cases from Requirements

- Involve clarification and restatement of the requirements to put them into a testable form.
  - Enumerate single requirements
  - Group related requirements

  For each requirement do the following (at a minimum):
  - Create a test case that demonstrates the requirement.
  - Create a test case that attempts to falsify the requirement.
    - For example: try something forbidden.
  - Test boundaries and constraints when possible
Example: Requirements for a video rental system

The system shall allow rental and return of films

1. If a film is available for rental then it may be lent to a customer.
   1.1 A film is available for rental until all copies have been simultaneously borrowed.

2. If a film was unavailable for rental, then returning the film makes it available.

3. The return date is established when the film is lent and must be shown when the film is returned.

4. It must be possible for an inquiry on a rented film to reveal the current borrower.

5. An inquiry on a member will reveal any films they currently have on rental.

Test situations for requirement 1

- Attempt to borrow an available film.
- Attempt to borrow an unavailable film.

Test situations for requirement 1.1

- Attempt to borrow a film for which there are multiple copies, all of which have been rented.
- Attempt to borrow a film for which all copies but one have been rented.
Deriving Test Cases from Requirements

- Test situations for requirement 2
  - Borrow an unavailable film.
  - Return a film and borrow it again.

- Test situations for requirement 3.
  - Borrow a film, and check date on a non-returned film.

- Test situations for requirement 4
  - Inquiry on rented film.
  - Inquiry on returned film.
  - Inquiry on a film which has been just returned.

- Test situations for requirement 5
  - Inquiry on member with no films.
  - Inquiry on member with 1 film.
  - Inquiry on member with multiple films.
## Deriving Test Cases from Use Cases

- For all use cases:
  1. Develop a graph of scenarios
  2. Determine all possible scenarios
  3. Analyze and rank scenarios
  4. Generate test cases from scenarios to meet a coverage goal
  5. Execute test cases

## Scenario Graph

- Generated from a use case
- Nodes correspond to point where system waits for an event
  - \textit{environment event, system reaction}
- There is a single \textit{starting node}
- End of use case is \textit{finish node}
- Edges correspond to event occurrences
  - May include conditions and looping edges
- Scenario:
  - \textit{Path} from starting node to a finish node
Use Case Scenario Graph (1)

Title: User login
Actors: User
Precondition: System is ON
1. User inserts a card
2. System asks for personal identification number (PIN)
3. User types PIN
4. System validates user identification
5. System displays a welcome message to user
6. System ejects card
Postcondition: User is logged in

Use Case Scenario Graph (2)

Alternatives:
1a: Card is not valid
1a.1: System emits alarm
1a.2: System ejects card
4a: User identification is invalid
   AND number of attempts < 4
   4a.1 Ask for PIN again and go back
4b: User identification is invalid
   AND number of attempts = 4
   4b.1: System emits alarm
   4b.2: System ejects card
Scenarios

- Paths from start to finish
- The number of times loops are taken needs to be restricted to keep the number of events in a scenario finite.

<table>
<thead>
<tr>
<th>ID</th>
<th>Events</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-2-3-4-5-6</td>
<td>User login with regular card. Correct PIN on first try. Normal scenario.</td>
</tr>
<tr>
<td>2</td>
<td>1-1a.1-1a.2</td>
<td>User login with invalid card</td>
</tr>
<tr>
<td>3</td>
<td>1-2-3-4-3-4-5-6</td>
<td>User login with regular card. Wrong PIN on first try. Correct PIN on second try.</td>
</tr>
<tr>
<td>4</td>
<td>1-2-3-4-(2-3-4)3-4b.1-4b.2</td>
<td>User login with regular card. Wrong PIN on all four tries.</td>
</tr>
</tbody>
</table>

Scenario Ranking

- If there are too many scenarios to test:
  - Ranking may be based on criticality and frequency
  - Can use operational profile, if available
    - “Operational profile”: statistical measurement of typical user activity of the system.
    - Example: what percentage of users would typically be using any particular feature at any time.
- Always include main scenario
  - Should be tested first
Test Case generation

- Satisfy a coverage goal. For example:
  - All branches in graph of scenarios (minimal coverage goal)
  - All scenarios
  - n most critical scenarios

Example of Test Case

**Test Case: TC1**

**Goal:** Test the main course of events for the ATM system.

**Scenario Reference:** 1

**Setup/Precondition:** Create a Card #2411 with PIN #5555 as valid user identification

  System is ON

**Course of test case**

<table>
<thead>
<tr>
<th>#</th>
<th>External event</th>
<th>Reaction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User inserts card #2411</td>
<td>System asks for Personal Identification Number (PIN)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>User types PIN #5555</td>
<td>System validates user identification. System displays a welcome message to the user.</td>
<td></td>
</tr>
</tbody>
</table>

**Pass criteria:** User is logged in
Forced-Error Test (FET) - 1

- **Objective:** to force system into all error conditions
  - Basis: set of error messages for system.

- **Checks**
  - Error-handling design and communication methods consistency
  - Detection and handling of common error conditions
  - System recovery from each error condition
  - Correction of unstable states caused by errors

Forced-Error Test (FET) - 2

- **Verification of error messages to ensure:**
  - Message matches type of error detected.
  - Description of error is clear and concise.
  - Message does not contain spelling or grammatical errors.
  - User is offered reasonable options for getting around or recovering from error condition.
Forced-Error Test (FET) - 3

- How to obtain a list of error conditions?
  - Obtain list of error messages from the developers
    - Interviewing the developers
  - Information from specifications
  - Analyzing every possible event with an eye to error cases
  - Using own experience
  - Using a standard valid/invalid input test matrix

Forced-Error Test (FET) - 4

- For each error condition:
  1. Force the error condition.
  2. Check the error detection logic
  3. Check the handling logic
     - Does the application offer adequate forgiveness and allow the user to recover from the mistakes gracefully?
     - Does the application itself handle the error condition gracefully?
     - Does the system recover gracefully?
     - When the system is restarted, it is possible that not all services will restart successfully?
Forced-Error Test (FET) - 5

4. Check the error communication
   ■ Determine whether an error message appears
   ■ Analyze the accuracy of the error message
     ■ Note that the communication can be in another medium such as an audio cue or visual cue

5. Look for further problems

---

Usability Testing

- Checks ability to learn, use system to perform required task
  - Usability requirements usually not explicitly specified

- Factors influencing ease of use of system
  - Accessibility: Can users enter, navigate, and exit the system with relative ease?
  - Responsiveness: Can users do what they want, when they want, in an intuitive/convenient way?
  - Efficiency: Can users carry out tasks in an optimal fashion with respect to time, number of steps, etc.?
  - Comprehensibility: Can users quickly grasp how to use the system, its help functions, and associated documentation?
Usability Testing – cont.

- Typical activities for usability testing
  - Controlled experiments in simulated working environments using novice and expert end-users.
  - Post-experiment protocol analysis by human factors experts, psychologists, etc

- Main objective: collect data to improve usability of software