3.1 System Development Cycle

Information Systems

- System: set of components that interact to achieve a common goal
- Types of systems:
  - Transaction Processing Systems (Tier 1):
    - Operational level systems
    - Used by lowest level workers
    - Handles basic data information
    - E.g. cashier inputs
  - Management Information Systems (Tier 2):
    - Managerial control over the ongoing functions of a business
    - Built on data from TPS
    - Used by middle-management level
    - E.g. analyse if workers meet their criteria to be promoted
  - Decision Support Systems (Tier 3):
    - Knowledge based system
    - Support strategic planning: manipulate and build upon data from TPS/MIS to generate insights and new info
    - Used by senior managers
  - Strategic Information Systems (Tier 4):
    - Organises and presents data and info from both external data and internal MIS/TPS to support executives
    - Help executives and senior managers analyse current trends to give a competitive advantage to the organisation
  - Real-time Systems:
    - Controls an environment by receiving data, processing, returning results quickly to affect environment at that time
  - Expert Systems:
    - Guides performance of ill-structured tasks which require specialisation in experience and knowledge
- Components of Information Systems: input, output, feedback, control, adjustments

System Development Process

- Life cycle of a system:
  1. Business needs: requirement and need for new systems
  2. System development: process to analyse the needs, design and implement system
  3. System installation: system goes to live production
  4. System operation: period of active use
  5. System obsolescence: business environment changes and system is no longer suitable
- **System development goals:**
  1. System quality
     - Functionality
       - Reliability (completeness, accuracy, robustness, integrity)
       - Clarity (consistency, predictability)
       - Efficiency (fast turnaround, low resource requirement)
     - Ease of maintenance
       - Understandability (clear documentation, cohesiveness, consistency)
       - Modifiability (modular structure, modular independence)
       - Testability (clear documentation, modular structure)
     - Flexibility
       - Portability (site / device / language independence)
       - Adaptability (procedural flexibility, programme-data independence)
  2. Project management
     - Timeliness
     - Cost
     - User commitment
  3. Organisational relevance
     - Operational control
     - Management control
     - Strategic planning

System Development Life Cycle (SDLC)

1. Investigation
   - Objective: evaluate a request for system development as to its scope and feasibility
   - Process:
     - Initial investigation

        - Feasibility study: economical, technical, operational
    - End product: feasibility report
2. Analysis and General Design
    - Objectives:
        - Determine requirements for new system
        - Develop general design
        - Establish user acceptance of and concurrence in design
        - Obtain commitment from Computer Info System that design of new system can be implemented within established time and money limits
        - Develop project work
        - Present sufficient info for steering committee to determine if to continue/revise the scope/approach/abort project
    - Process:
        - Existing system review
        - New system requirements
        - New system design
        - Implementation and installation planning
    - End product:
        - User specifications
        - New system design specifications
        - Prelim detailed design and implementation plan
        - Prelim system test plan
        - User training outline
        - Prelim installation plan
3. Detailed Design and Implementation
    - Objective: produce a completely documented and fully tested new system
        - Encompasses computer processing, manual procedures, and interfaces between the two
    - Processes:
        - Technical design
        - Test specification and planing
        - Programming and testing
        - User training
        - Acceptance test
4. Installation
    - Objectives:
        - Replace existing system with new, tested, documented system
        - Maximise potential benefits of new system

    - Processes:
        - File conversion
        - System installation
    - End product:
        - No major new end product
5. Review
    - Objectives:
        - Review system development results: effectiveness of life cycle, management techniques required
        - Review new system to determine if projected benefits have been realised
        - Review new system to determine if modifications/ enhancements
    - Processes:
        - Development recap
        - Post-implementation review
    - End product:
        - System development recap report
        - Post-implementation review report

Prototyping

- Alternative to SDLC
- Allows users to quickly build a working model as initially envisioned by user
- Can help when user is unable to visualise how the system will work
- Objectives: enhance system development process
- Process:
    - Iterative process
    - User works with prototype, requirements are revised and refined
    - Prototype is revised based on changed requirements
- Goal: To enhance system development process
- Suitability:
    - More suitable for online interactive systems when the potential capabilities of system are beyond users' experience
    - Less suitable for batch-oriented subsystems / highly complex math processes

Information Gathering

- Importance:
  - Build understanding of business problem and nature / content of business operations
  - Critical for understanding and developing systems
- Sources of information:
  - Existing documentation
  - System users and managers
  - External sources (e.g. other companies, vendors, business publications)
- Methods of information gathering:
  - Interviews
  - Questionnaires
  - Observation (on-site)
- Categories of information:
  - Info about organisation (e.g. goals & objectives, org structure, policies)
  - Info about people (e.g. authority & responsibility relationships, job duties, interpersonal relationships)
  - Info about work (e.g. tasks & work flow, schedules, methods and procedures)
  - Info about work environment (e.g. physical environment, resources available)

Requirements Determination / Definition

- Set of activities performed to understand the problem
- Structuring definition of current system
- Producing clear & systematic definition of required system
- Identifying objectives
- Support the 3 goals of SD
- Follow similar strategies as info gathering

Software Testing

- Nature of testing:
  - Process of executing programmes to discover errors
  - Goals:
    - Force programme to work incorrectly
    - Discover cause of errors
    - Revise code to eliminate errors

    - Testing is a destructive process
- Levels of testing:
    - Unit / module testing:
        - Applied to individual programme units
        - Done by using modules to process test data and examining outputs to determine if expected results are obtained
    - Integration testing:
        - Applied to interface between modules
        - Done in parallel with unit testing
        - Models executed in combinations to determine if interface between them are working
        - Examines transfer of data among modules
        - Two approaches:
            - Incremental testing — modules added to one another for testing in an individual basis
                1. Top-down:
                    - Early verification of major modules and overall control logic
                    - Possible to demo complete programme functions
                    - Usable portions of system can be completed and tested while other pieces are being worked on
                    - Difficult to provide test cases
                    - Necessary to develop stubs which will be replaced with complete modules
                2. Bottom-up:
                    - Testing detects early identification of any detailed processing flaws
                    - Users view these components early — good PR tool
                    - Puts off ability to form overall skeletal programme until all modules are tested and put into place
            - Non-incremental testing — all modules developed and tested together as an entity
                - Extremely difficult and not recommended
    - Function testing:
        - Seeks to identify any variance between results of programme processing and specs for programmes

        - Concentrates upon results of complete programmes
        - Criteria:
            - Input & output formats
            - File organisation
            - File access
            - Human-machine interfaces
    - System testing:
        - Deals with the integration of a system
        - Extends beyond computer system to encompass all related procedures and processing
        - Purpose: try using the system as an operational and functional entity
        - Check if training / reference manuals are adequate to cover any problems that may arise
        - Simulate actual operations before implementation
    - Acceptance testing:
        - Performed by user before proper operation
        - Determine whether to accept / reject system according to requirements
- Testing strategies:
    - White-box testing:
        - Internal module testing approach
        - Examines logic of modules as if the processes were clear and transparent
        - Includes:
            - Examination of procedure details
            - Tests covering the execution of all statements
            - Tracing of decision paths
        - Only carried out at the module level
        - Test data coverage:
            - Execute every statement
            - Tests all combinations of decisions & conditions
        - Great amount of testing done at this level — exhaustive due to vast number of combinations
    - Black-box testing:
        - Concerned with potential module/programme IO
        - Used to review module / programme functions without knowing the actual code
        - Based on external design specs:
            - Monitor inputs & outputs
            - Judgements are reached based on results

        - Determine if inputs are accepted as planned / outputs meet expectations
      - Test data coverage:
        - Accepted values
        - Boundary values
        - Erroneous values
      - No attempt to uncover all errors
      - Combine with white-box testing
    - Error guessing:
      - Implies a list of potential troublesome areas are predicted
      - Test cases are derived based on list (trial & error)
      - Success lies largely on experience and intuition of specific test cases
      - Test data coverage:
        - IO errors — makes sure all records are transmitted & received as expected
        - Data structure errors — discover errors in handling & building of data elements
        - Arithmetic errors — check for calculation errors
        - Comparison errors — ensures programme doesn't permit comparison between different data types
        - Control logic errors (mainly done in white-box testing)
    - Alpha testing:
      - Purpose:
        - Find all the problems with the software
        - Modify software according to errors
      - Consists of extensive in-house software usage for operational stability and numerical accuracy
      - Test procedures, checklists, benchmarks are used to test every aspect of the software
      - Alpha testing continues until design team is satisfied
    - Beta testing:
      - Purpose: find any problems that remain in the software that previous tests fail to uncover
      - Involves a much larger group of testers:
        - Wide variety of test users under different conditions (OS & networks)
        - Public beta
- Typical approach to design of software testing:
  - Apply white-box testing selectively
  - Apply black-box testing to evaluate programme and module

functions
- Examine list of troublesome areas generated by error guessing