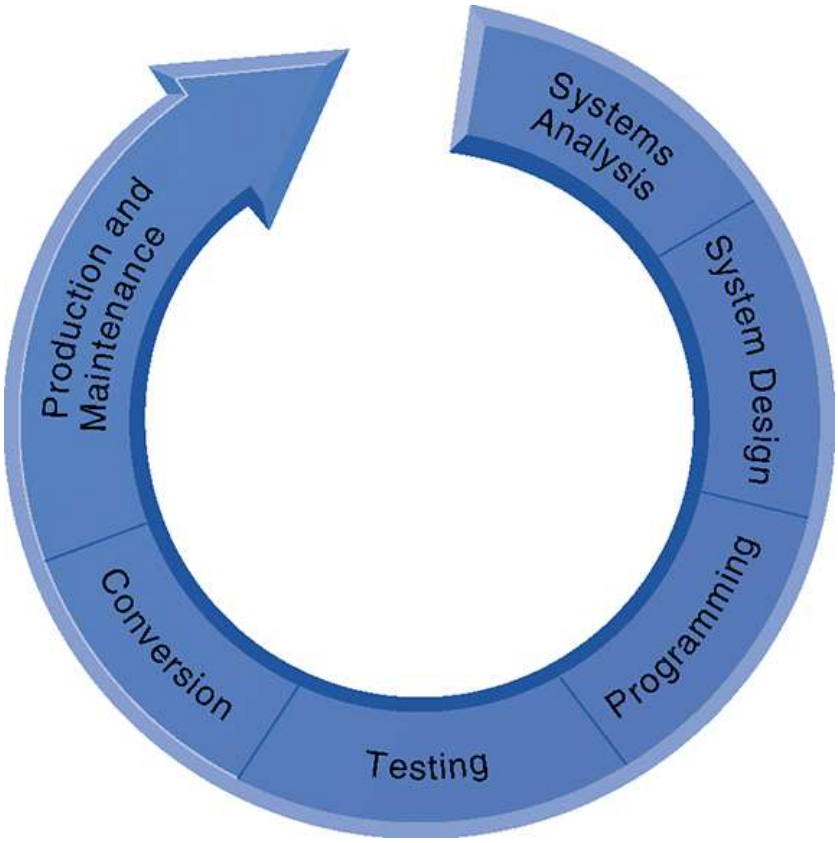
Building Information Systems

Note: In addition to the following main points, the students should refer to the detailed study material which has already been e-mailed to them.

- **The Systems Development Cycle**



- **Systems development:** Activities that go into producing an information system solution to an organizational problem or opportunity
    - Systems analysis
    - Systems design
    - Programming
    - Testing
    - Conversion
    - Production and maintenance

- **Systems analysis**
    - Analysis of problem that will be solved by system
        - Defining the problem and identifying causes
        - Specifying solutions
            - Systems proposal report identifies and examines alternative solutions
        - Identifying **information requirements**
    - Includes **feasibility study**
        - Is solution feasible from financial, technical, organizational standpoint
        - Is solution a good investment?
        - Is required technology, skill available?
    - Establishing information requirements
        - Who needs what information, where, when, and how
        - Define objectives of new/modified system
        - Detail the functions new system must perform
    - Faulty requirements analysis is leading cause of systems failure and high systems development cost

- **Systems design**
    - Describe system specifications that will deliver functions identified during systems analysis
    - Should address all managerial, organizational, and technological components of system solution
    - **Role of end users**
        - User information requirements drive system building
        - Users must have sufficient control over design process to ensure that system reflects their business priorities and information needs
        - Insufficient user involvement in design effort is major cause of system failure

- **Programming:**
    - System specifications from design stage are translated into software program code
    - Software may be purchased, leased, or outsourced instead

- **Testing**
    - To ensure system produces right results
    - **Unit testing:** Tests each program in system separately
    - **System testing:** Tests functioning of system as a whole
    - **Acceptance testing:** Makes sure system is ready to be used in production setting
    - **Test plan:** All preparations for series of tests

- **Conversion**
    - Process of changing from old system to new system
    - Four main strategies
        - Parallel strategy
        - Direct cutover
        - Pilot study
        - Phased approach
    - Requires end-user training
    - Finalization of detailed documentation showing how system works from technical and end-user standpoint

- **Production and maintenance**
    - System reviewed to determine if any revisions needed
    - May prepare formal **postimplementation audit** document
    - **Maintenance**
        - Changes in hardware, software, documentation, or procedures to a production system to correct errors, meet new requirements, or improve processing efficiency
        - 20% debugging, emergency work
        - 20% changes to hardware, software, data, reporting
        - 60% of work: User enhancements, improving documentation, recoding for greater processing efficiency

CORE ACTIVITY	DESCRIPTION
Systems analysis	**Identify problem(s)** **Specify solutions** **Establish information requirements**
Systems design	**Create design specifications**
Programming	**Translate design specifications into code**
Testing	**Unit test** **Systems test** **Acceptance test**
Conversion	**Plan conversion** **Prepare documentation** **Train users and technical staff**
Production and maintenance	**Operate the system** **Evaluate the system** **Modify the system**

- **Most prominent methodologies for modeling and designing systems:**
    - Structured methodologies
    - Object-oriented development

- **Structured methodologies**
    - **Structured**: Techniques are step-by-step, progressive
    - **Process-oriented**: Focusing on modeling processes or actions that manipulate data
    - Separate data from processes
- **Data flow diagram:**
    - Primary tool for representing system's component processes and flow of data between them
    - Offers logical graphic model of information flow

        - High-level and lower-level diagrams can be used to break processes down into successive layers of detail
    - **Data dictionary:** Defines contents of data flows and data stores
    - **Process specifications:** Describe transformation occurring within lowest level of data flow diagrams
    - **Structure chart:** Top-down chart, showing each level of design, relationship to other levels, and place in overall design structure

- **Object-oriented development**
    - Uses **object** as basic unit of systems analysis and design
        - **Object:**
            - Combines data and the specific processes that operate on those data
            - Data encapsulated in object can be accessed and modified only by operations, or methods, associated with that object
    - Object-oriented modeling based on concepts of class and inheritance
        - Objects belong to a certain class and have features of that class
        - May inherit structures and behaviors of a more general, ancestor class
    - More iterative and incremental than traditional structured development
        - **Systems analysis:** Interactions between system and users analyzed to identify objects
        - **Design phase:** Describes how objects will behave and interact; grouped into classes, subclasses and hierarchies
        - **Implementation:** Some classes may be reused from existing library of classes, others created or inherited
    - Because objects reusable, object-oriented development can potentially reduce time and cost of development
- **Computer-aided software engineering (CASE)**
    - Software tools to automate development and reduce repetitive work, including
        - Graphics facilities for producing charts and diagrams
        - Screen and report generators, reporting facilities
        - Analysis and checking tools
        - Data dictionaries

        - Code and documentation generators
    - Support iterative design by automating revisions and changes and providing prototyping facilities
    - Require organizational discipline to be used effectively

- **Other Alternative Systems-Building Methods**
    - **Traditional systems life-cycle**
    - **Prototyping**
    - **End-user development**
    - **Application software packages**
    - **Outsourcing**

- **Traditional systems lifecycle:**
    - Oldest method for building information systems
    - Phased approach - divides development into formal stages
        - Follows “waterfall” approach: Tasks in one stage finish before another stage begins
    - Maintains formal division of labor between end users and information systems specialists
    - Emphasizes formal specifications and paperwork
    - Still used for building large complex systems
    - Can be costly, time-consuming, and inflexible

- **Prototyping**
    - Building experimental system rapidly and inexpensively for end users to evaluate
    - **Prototype:** Working but preliminary version of information system
        - Approved prototype serves as template for final system
    - **Steps in prototyping**
        - Identify user requirements
        - Develop initial prototype
        - Use prototype
        - Revise and enhance prototype

- **Advantages of prototyping**
    - Useful if some uncertainty in requirements or design solutions
    - Often used for end-user interface design
    - More likely to fulfill end-user requirements
- **Disadvantages**
    - May gloss over essential steps
    - May not accommodate large quantities of data or large number of users
    - May not undergo full testing or documentation

- **End-user development:**
    - Uses **fourth-generation languages** to allow end-users to develop systems with little or no help from technical specialists
    - **Fourth generation languages**: Less procedural than conventional programming languages
        - PC software tools
        - Query languages
        - Report generators
        - Graphics languages
        - Application generators
        - Application software packages
        - Very high-level programming languages
    - **Advantages**:
        - More rapid completion of projects
        - High-level of user involvement and satisfaction
    - **Disadvantages**:
        - Not designed for processing-intensive applications
        - Inadequate management and control, testing, documentation
        - Loss of control over data
    - **Managing end-user development**
        - Require cost-justification of end-user system projects
        - Establish hardware, software, and quality standards

- **Application software packages**
    - Save time and money
    - Many packages offer customization features:
        - Allow software package to be modified to meet unique requirements without destroying integrity of package software
    - Evaluation criteria for systems analysis include:
        - Functions provided by the package, flexibility, user friendliness, hardware and software resources, database requirements, installation and maintenance efforts, documentation, vendor quality, and cost
    - **Request for Proposal (RFP)**
        - Detailed list of questions submitted to packaged-software vendors
        - Used to evaluate alternative software packages

- **Outsourcing**
    - **Several types**
        - **Cloud and SaaS providers**
            - Subscribing companies use software and computer hardware provided by vendors
        - **External vendors**
            - Hired to design, create software
            - Domestic outsourcing
                - Driven by firms need for additional skills, resources, assets
            - Offshore outsourcing
                - Driven by cost-savings
    - **Advantages**
        - Allows organization flexibility in IT needs
    - **Disadvantages**
        - Hidden costs, e.g.
            - Identifying and selecting vendor
            - Transitioning to vendor
        - Opening up proprietary business processes to third party

- **Rapid application development (RAD)**
    - Process of creating workable systems in a very short period of time
    - Utilizes techniques such as:
        - Visual programming and other tools for building graphical user interfaces
        - Iterative prototyping of key system elements
        - Automation of program code generation
        - Close teamwork among end users and information systems specialists

- **Joint application design (JAD)**
    - Used to accelerate generation of information requirements and to develop initial systems design
    - Brings end users and information systems specialists together in interactive session to discuss system's design
    - Can significantly speed up design phase and involve users at intense level

- **Agile development**
    - Focuses on rapid delivery of working software by breaking large project into several small sub-projects
    - Subprojects
        - Treated as separate, complete projects
        - Completed in short periods of time using iteration and continuous feedback
    - Emphasizes face-to-face communication over written documents, allowing collaboration and faster decision making

- **Component-based development**
    - Groups of objects that provide software for common functions (e.g., online ordering) and can be combined to create large-scale business applications
    - **Web services**
        - Reusable software components that use XML and open Internet standards (platform independent)
        - Enable applications to communicate with no custom programming required to share data and services

- Can engage other Web services for more complex transactions
- Using platform and device-independent standards can result in significant cost-savings and opportunities for collaboration with other companies