Unit #13

MIS 5203

Review

IT Auditor's responsibilities

- Provide assurance that enterprise objectives are being met by information systems and infrastructure management practices
- Identify which elements may represent greatest risk, and which controls are most effective at mitigating this risk
- Understand which methodologies are in use for:
  - Systems development, acquisition and maintenance
  - Identifying potential vulnerabilities and points requiring control
- Advise project team and senior management of deficiencies and best practices within each of these processes
- Produce and provide formal audit reports to the appropriate business managers including:
  1. Overall assessment of the controlled progress of the project
  2. Areas requiring improvement to complete the project, as specified, within budget and at an appropriate level of quality

System Development Life Cycle Methods

Software development lifecycle (SDLC) models are formal management processes for guiding the development of information systems

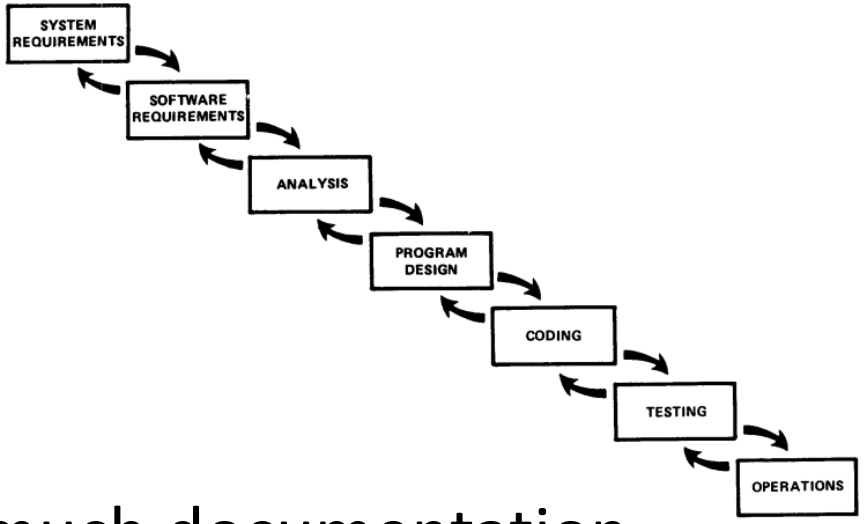
These include:

- **Waterfall Models**
  - Waterfall
  - Structured Systems Analysis and Design Method (SSADM)
- **Spiral and Iterative Models**
  - Structured Rapid Prototyping
  - Rapid Application development (RAD)
  - Agile Models
  - Rational Unified Model

If execution of the SDLC methodology is inadequate, however, the project may fail to meet business and user needs.

- *IS Auditor is responsible for verifying that the SDLC model is appropriate for the project's goals and is properly implemented*

Waterfall Model – Pros and Cons



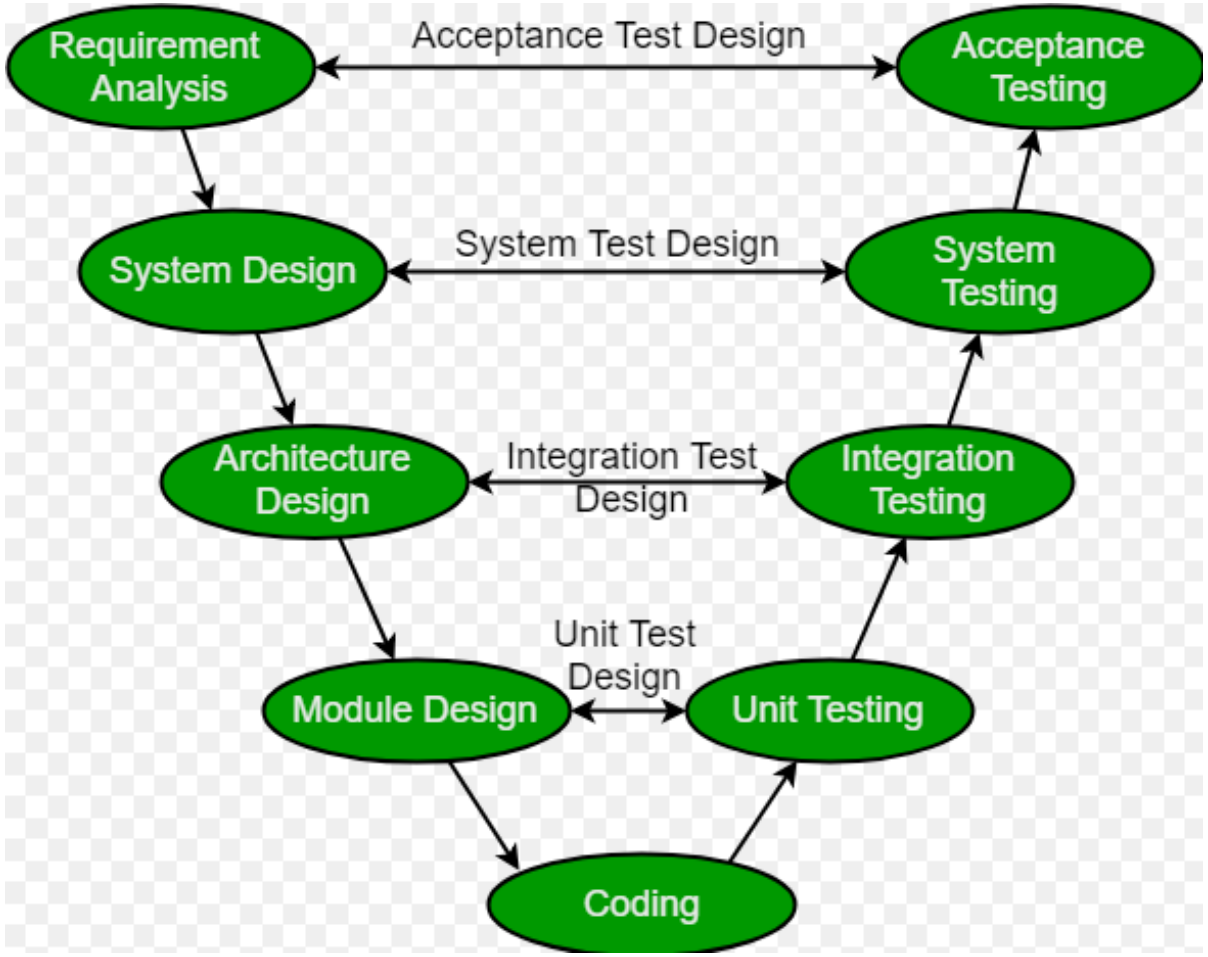
Pros

- Structured
- Worked well when requirements were well defined
- For large projects with enough time available

Cons

- Too much documentation
- Making changes becomes difficult during SDLC
- Poor speed to market
- Delayed implementation too long

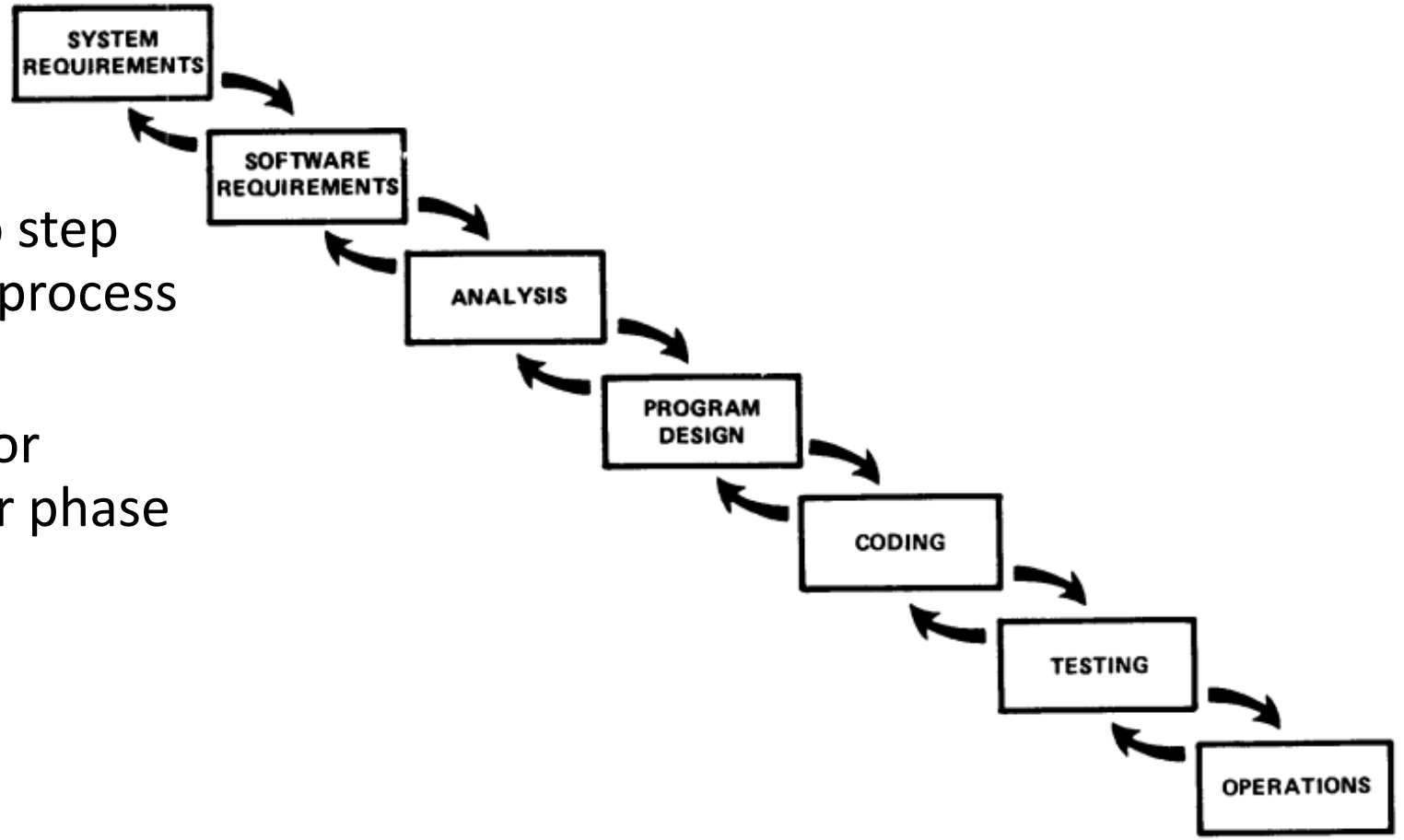
Waterfall Model with Verification and Validation



Waterfall and SSADM SDLC Models

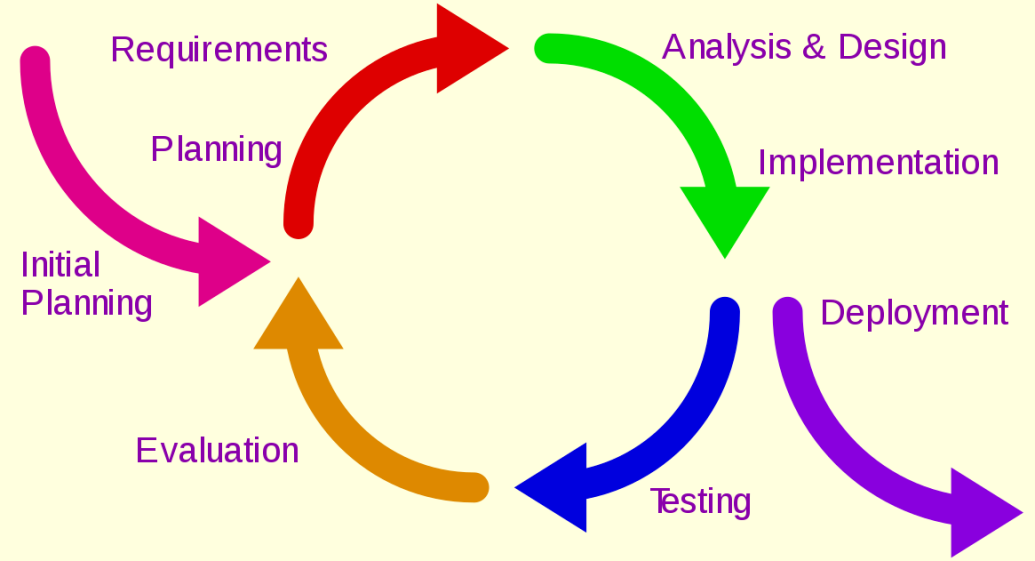
Major criticism

- Model allows developers to step back only one phase in the process
- Does not make provisions for discovery of errors at a later phase in the development cycle



Spiral, Iterative and Agile SDLC methods

- Prototyping-Evolutionary Development
- Rapid Application Development (RAD)
- Agile Development



Address difficulty of obtaining complete accurate requirements up front from users

Rapidly creates and evolves application

- Quickly deliver incremental progress
- Enable mid-course corrections
- Engage users' understanding and buy-in
- Gains user feedback and engagement in process and system improvement

Software Capability Maturity Model (CMM)

Created by Software Engineering Institute (SEI) at Carnegie Mellon University

Organizations engaged in software development move through a sequence of maturity phases:

Level 1: Initial

Hardworking people charging ahead in a disorganized fashion. Little or no defined software development process

Level 2: Repeatable

Basic lifecycle management processes introduced. Code reuse and repeatable results expected from similar projects.

Level 3: Defined

Software developers are trained and development projects take place within a standardized process model and follow formal documented software development processes

Level 4: Managed

Quantitative measures provide a detailed understanding of the development and qualitative processes

Level 5: Optimizing

Continuous improvement processes are based on feedback from one phase reaching previous phase to improve future results.

Return On Investment (ROI) Analysis

Part of feasibility study

Business benefits must justify costs and Information System investments for:

Development Projects

- *Developing, implementing and maintaining a new information system*

Acquisition Projects

- Acquiring, implementing and maintaining a new information system

Maintenance Projects

- Maintaining and improving an existing information system

IT Auditors are required to confirm information systems the business chooses to rely on:

1. Meet return on investment (ROI) expectations
2. Are planned, developed, implemented and maintained in a controlled and dependable manner

Enterprise Governance of Information and Technology

1. **Benefits Realization**
   - Creating value for the enterprise through IT:
     - Maintaining and increasing value derived from existing IT investments
     - Eliminating IT initiatives and assets that are not creating sufficient value
   - Delivery of fit-for-purpose services and solutions
     - On-time and within budget
     - That generate the intended financial and nonfinancial benefits
   - Value that IT delivers should be
     - Aligned directly with targeted business values
     - Measured in ways that show impact and contributions of IT-enabled investments

COBIT 2019 Framework: Introduction & Methodology, ISACA, 2018

Enterprise Governance of Information and Technology

2. **Risk Optimization**
   - Addresses business risk associated with use, ownership, operation, involvement, influence and adoption of IT within the enterprise
   - IT-related business risk consists of IT-related events that could potentially impact the business
   - While value delivery focuses on creation of value, risk management focuses on the preservation of value

COBIT 2019 Framework: Introduction & Methodology, ISACA, 2018

Enterprise Governance of Information and Technology

3. Resource Optimization

- Recognizes
  - Data and information are an important resources
  - Exploiting data and information to gain optimal value is a key focus
- Ensures
  - Appropriate capabilities are in place to execute the strategic plan
  - Sufficient, appropriate and effective resources are provided
  - An integrated, economical IT infrastructure is provided
  - New technology is introduced as required by the business
  - Obsolete systems are updated or replaced

COBIT 2019 Framework: Introduction & Methodology, ISACA, 2018

IT Audit's Role in a Feasibility Study

Focuses on the probability of a successful outcome of an IT project

1. Overall desirability of a information system from a corporate perspective
2. Likelihood of successful implementation

IT Auditor's role is to protect against insufficient attention paid at this stage

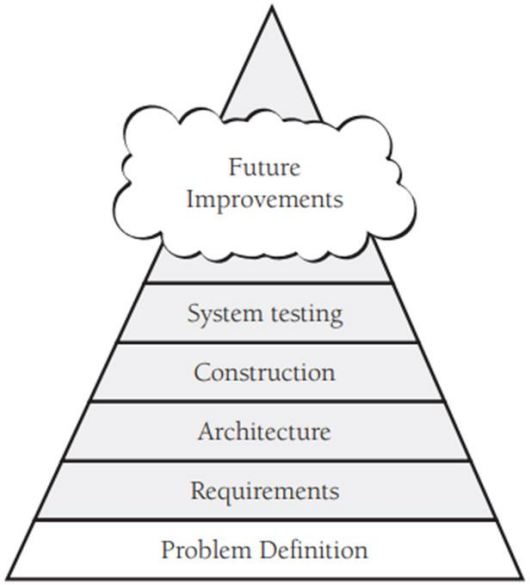
…which can result in the development or acquisition of expensive, inappropriate systems that do not fully address IT requirements of the organization

- Little expenditure has been made at this stage – a wrong decision can lead to many millions being invested to little effect and loss of significant time in gaining strategic advantage
- Many feasibility studies are conducted to support a go-head decision that has already been made whether or not there are clear benefits, tangible or intangible, and whether or not there are unacceptable risks in either the development process or implementation of the intended system

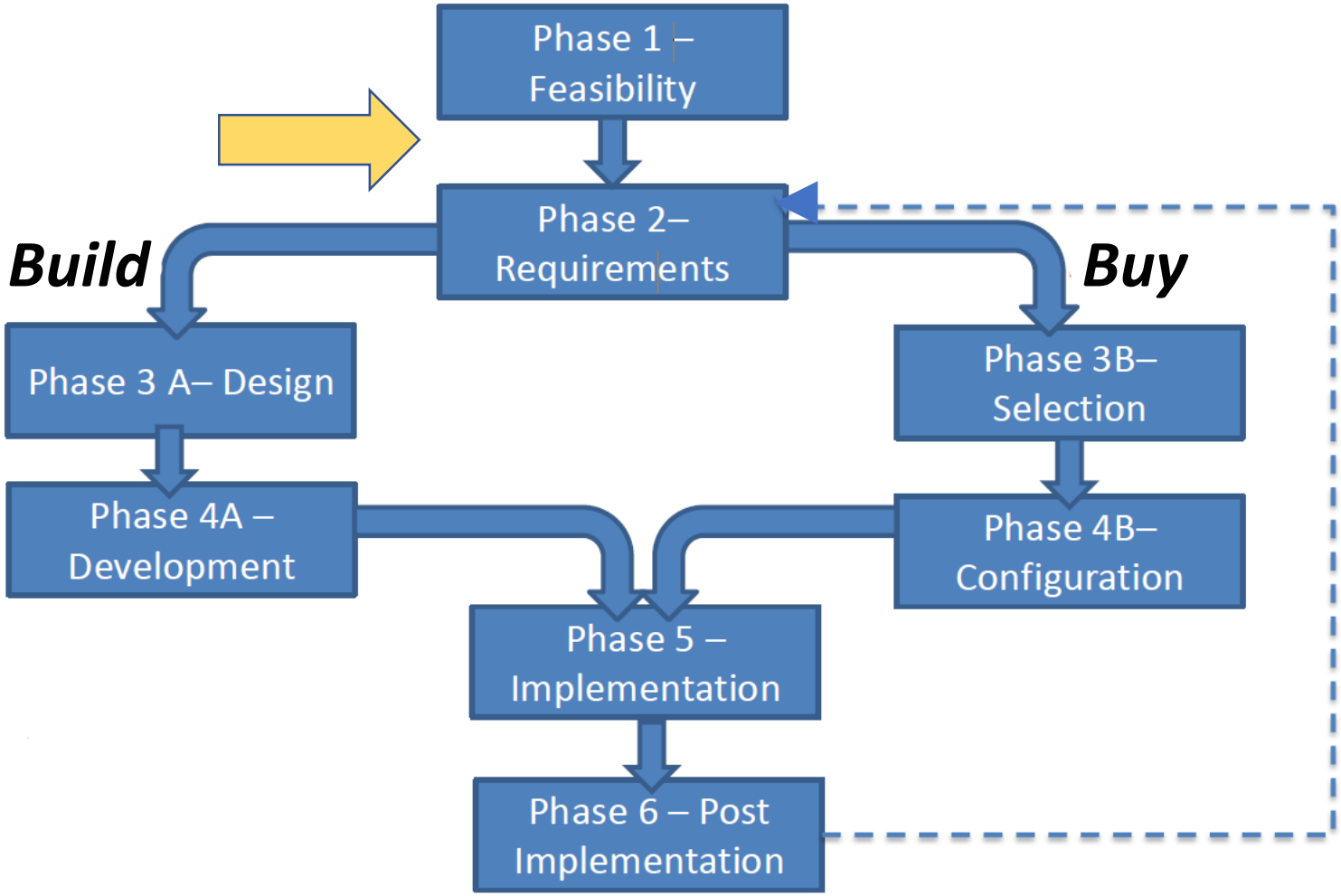
An acceptable finding of the feasibility study could be not to proceed with any system development or acquisition

Cascarino, R.E. (2012) Auditor's Guide to IT Auditing, pp. 237-240

IT Auditor reviews deliverables at the end of each phase acting as "stage gate"



Problem definition during Phase 1 lays the foundation for the rest of the process



Business Case

Should be of sufficient detail to answer:

- *Why should this project be undertaken?*

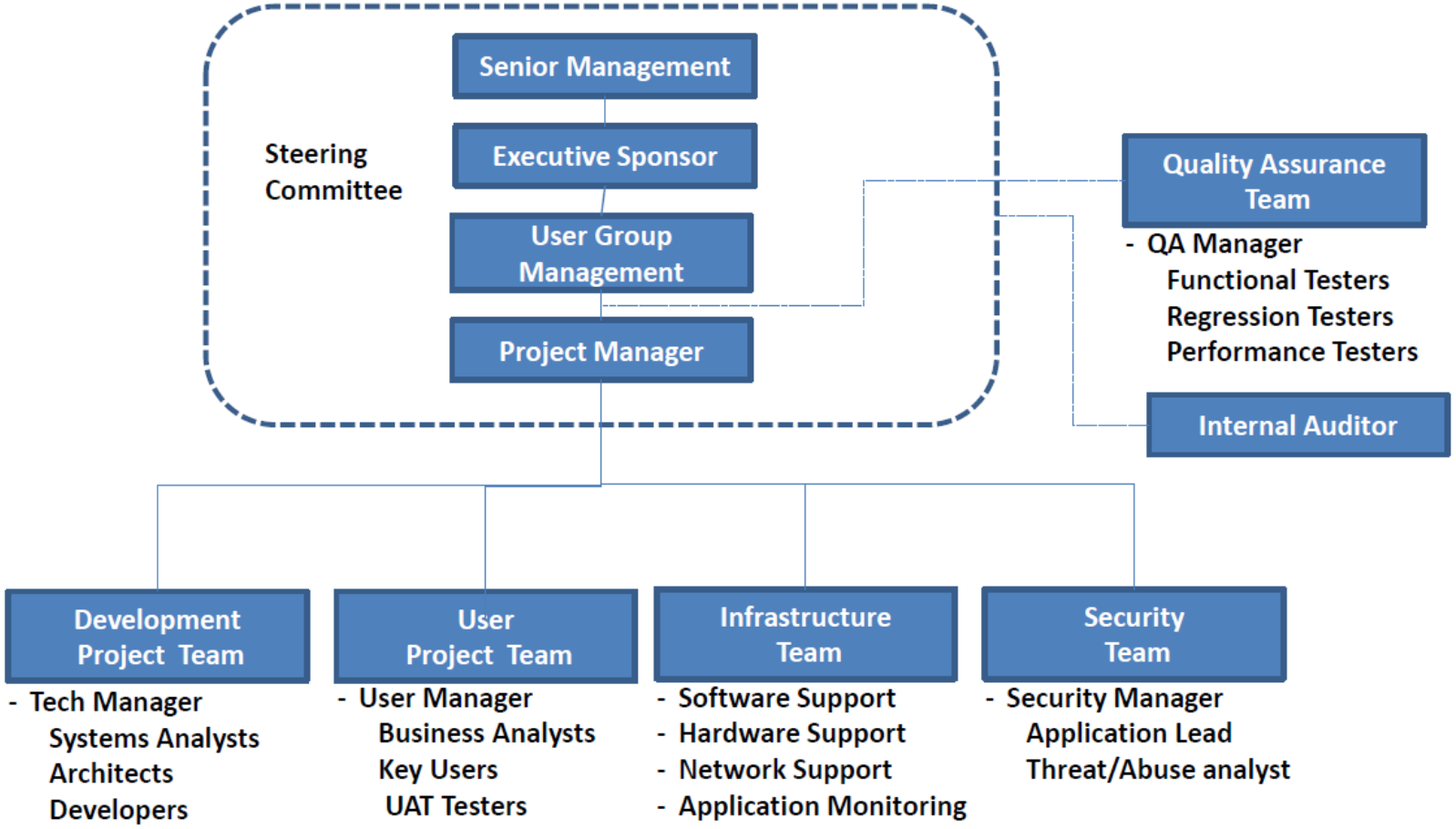
- *Why should this project continue?*

  Well-planned projects have decision points ("stage gates" or "kill-points") at which the business case is formally reviewed to ensure the project is still valid

  - Increased costs or reduction in anticipated benefits can invalidate the business case
  - IT Auditors are responsible for detecting when changes in project value require reconsidering the validity of the Business Case and continuing the project
  - IT Steering Committee needs to consider whether the project should continue

Project roles

Typical Project Management Organization Chart



Project Plan

Should define clear, discrete activities and the work needed to complete each activity. Tasks include:

1. Describing Project Scope
2. Dividing the Project into Manageable Tasks
3. Estimating Resources and Creating a Resource Plan
4. Developing a Preliminary Schedule
5. Developing a Communication Plan
6. Determining Project Standards and Procedures
7. Identifying and Assessing Risk
8. Creating a Preliminary Budget
9. Setting a Baseline Project Plan

Cost Estimation

Can be based on:

- Analogous estimating – using experience from prior projects
- Parametric estimating – extending cost models from similar prior projects
- Bottom-up estimating – building up costs from detailed Work Breakdown Structure
- Actual costs – leveraging actual historic costs from identical prior project

Software Size Estimation Techniques:

- Source Lines of Code (SLOC)
- COnstructive COst Model (COCOMO)
- Function Point Analysis (FPA)
- FPA With Feature Points

Cost Estimation – Function Point Analysis (FPA)

A technique used to determine size of a development task, based on the number of function points

Function points are factors such as inputs, outputs, queries, and logical processing units

Parameter	Simple	Average	Complex
# of Screens	5	10	5
# of Services	0	2	2
# of Database Tables	4	2	0
# of Data Files	0	4	0
# of Reports	0	2	3
# of External Interfaces	1	1	1
# of Environment Variables	3	4	0
Total:	13	25	11

Cost Estimation – Function Point Analysis (FPA)

A technique used to determine size of a development task, based on the number of function points

Function points are factors such as inputs, outputs, queries, and logical processing units

Parameter	Simple	Simple Weight	Average	Average Weight	Complex	Complex Weight	Function Points
# of Screens	5	5	10	10	5	15	200
# of Services	0	10	2	20	2	40	120
# of Database Tables	4	10	2	15	0	20	70
# of Data Files	0	5	4	10	0	15	40
# of Reports	0	20	2	30	3	40	180
# of External Interfaces	1	20	1	40	1	60	120
# of Environment Variables	3	20	4	30	0	40	180
Total:	13	90	25	155	11	230	910

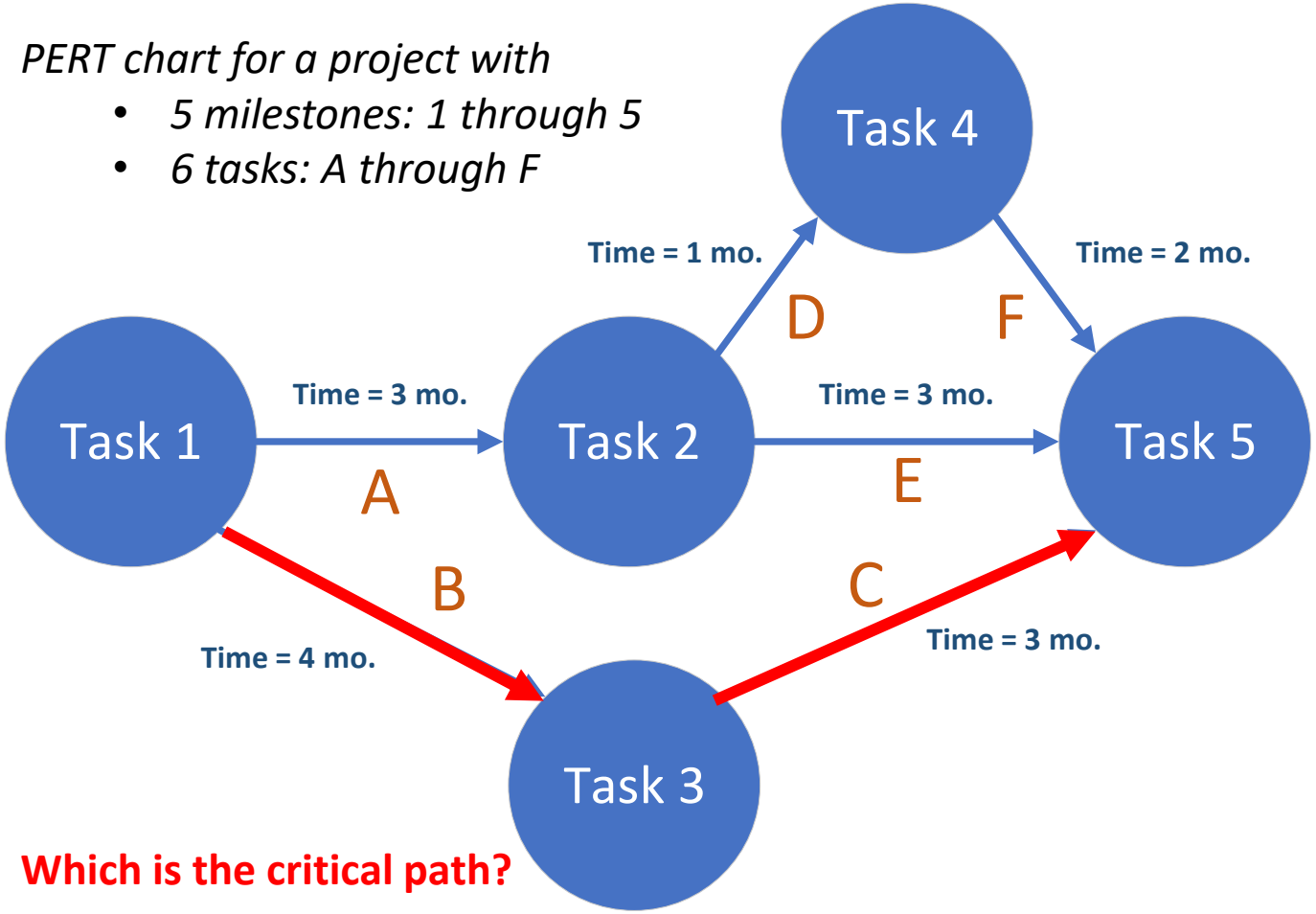
Critical Path Analysis

An algorithm for scheduling a set of project activities

- Used with Program Evaluation and Review Technique (PERT) network diagram
- Determines
  1. Critical path by identifying and measuring the time required to complete the longest path of dependent activities (i.e. tasks) from start to finish
  2. Earliest and latest that each task can start and finish without making the project longer
     - Slack time is the amount of time a task can be delayed without making the project longer
       - **Critical path has 0 slack**

PERT chart for a project with

- *5 milestones: 1 through 5*
- *6 tasks: A through F*

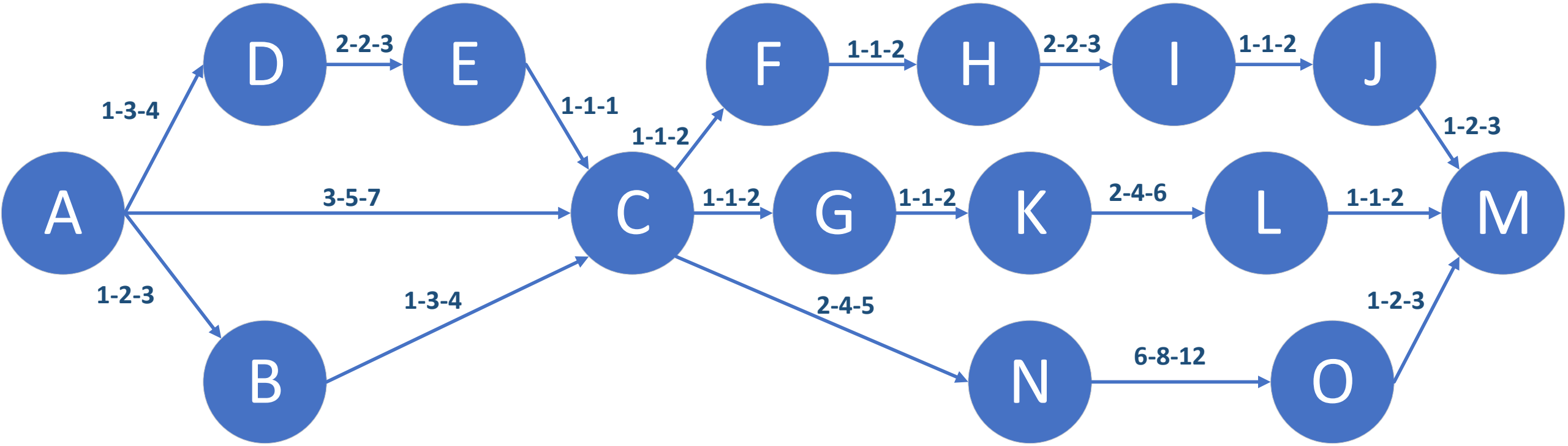


Which is the critical path?

3 Paths through the tasks:

A, D, F = 3 + 1 + 2 = 6 mo.	Slack time = 1 mo.
A, E = 3 + 3 = 6 mo.	Slack time = 1 mo.
B, C = 4 + 3 = 7 mo.	**Slack time = 0 mo.**

Calculate Expected Time Duration of Activities (Tasks)

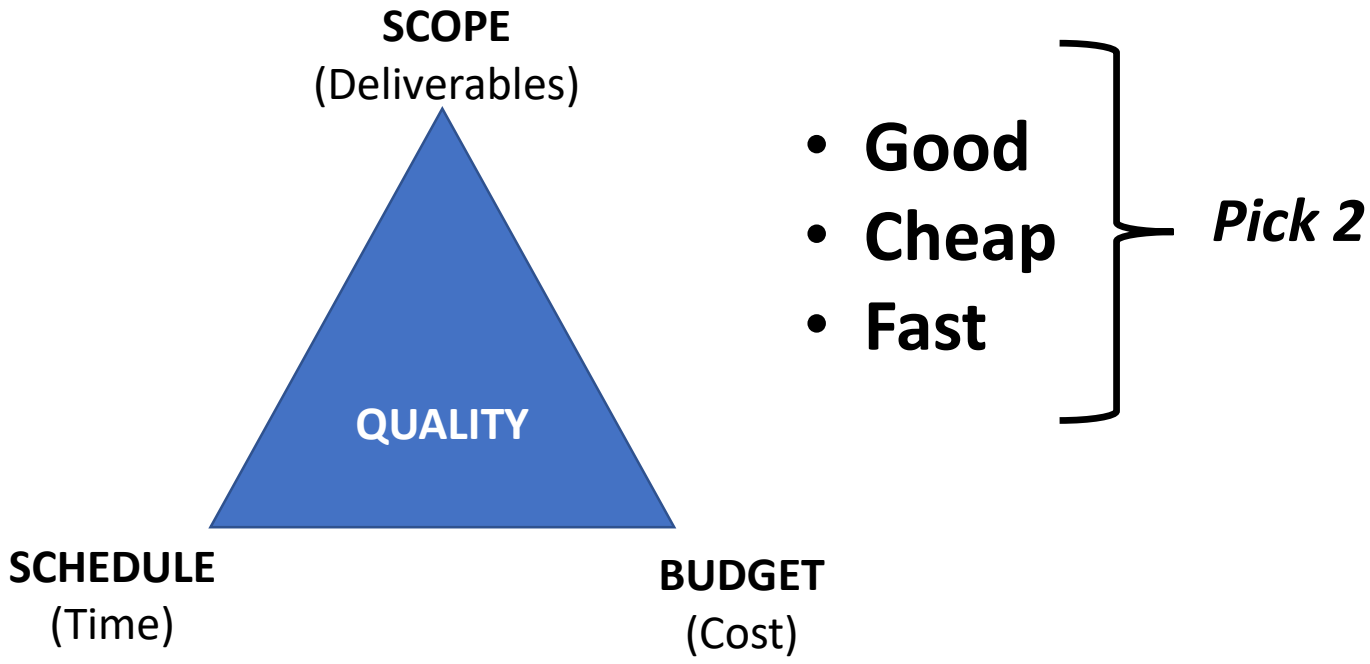


$$\text{PERT Time Estimate} = \frac{\text{Optimistic} + (4 \times \text{Most Likely}) + \text{Pessimistic}}{6}$$

Project Planning - Project Management Triangle

Projects are planned and managed in the context of 3 constraints: Scope, Schedule and Budget

- Project Managers can trade among these 3 constraints
- Changes in one constraint necessitate changes in others or quality will suffer



What are the requirement types?

1. **Functional requirement**
   - Something the system must do
   - Outcome the system must produce as part of its useful operation
2. **Nonfunctional requirement**
   - Quality or constraint for the system
   - Must be maintained as the system operates
3. **Security requirement**
   - As associated protection that must be placed on some part of the system
   - Contingency to normal operations
   - Guarantee of some constraint that would otherwise violate conditions of safe operation

IT Auditor's responsibilities during SDLC control stages

- **System requirements**
- System design
- System development
- System operation
- System utilization

System requirements questions:

- Have stakeholders/users determined that the requirements definition accurately and completely reflects the functional requirements for the proposed system?
- Is the requirements definition feasible within the technical infrastructure in place or envisioned?
- The proposed system will eventually require an IT Audit, and IT Audit has its own requirements
  - Have they been included in the requirements definition document?

Requirements focus on

- Business objectives that drive what and how work is done
- Information people need to do their jobs
- Data movement, transformation and storage processes
- Data handling/processing rules
- Dependencies and sequences
- Key events

Characteristics of good requirements specifications

- Clear and unambiguous
- Complete and Comprehensive
- Consistent
- Traceable, verifiable, testable
- Modifiable
- Prioritized

Other Methods for Determining Requirements

- **Joint Application Design (JAD)**
  - Brings together key users, managers, and systems analysts
  - Purpose: collect system requirements simultaneously from key people
  - Conducted off-site
- **System prototypes (from Spiral, RAD and Agile SDLC methods)**
  - Iterative development process
  - Rudimentary working version of system is built
  - Refine understanding of system requirements in concrete terms

Requirements Specification

With all projects it is important to understand:

- Requirements of the information system in terms of:
  - Functionality
  - Data
  - Characteristics of the environment

Without a clear definition and specification of requirements, the project is likely experience:

Scope creep

- Situation where project adds more and more functionality beyond the original specification

Good requirements answer

1. **Who are the stakeholders for this requirement?**
   - Listing of interested parties for this requirement in particular
   - Supports traceability
2. **Why should this be part of the system?**
   - Answer could be obvious or simply part of the project description
   - If no justified answer in the business case or from a stakeholder, it could be unnecessary adding cost and risk
3. **What are the dependencies for this requirement?**
   - Goal is to connect requirements to each other
     - E.g. Successful login could be precursor to executing functionality for a requirement
     - E.g. Completing actions in this requirement could facilitate other requirements
4. **What are the constraints on this requirement?**
   - Answer focuses on controls for the requirement
   - A control can be anything that monitors execution to make sure it is proceeding as expected
   - Absence of an answer may indicate additional refinement of requirement may be needed

Security Requirements

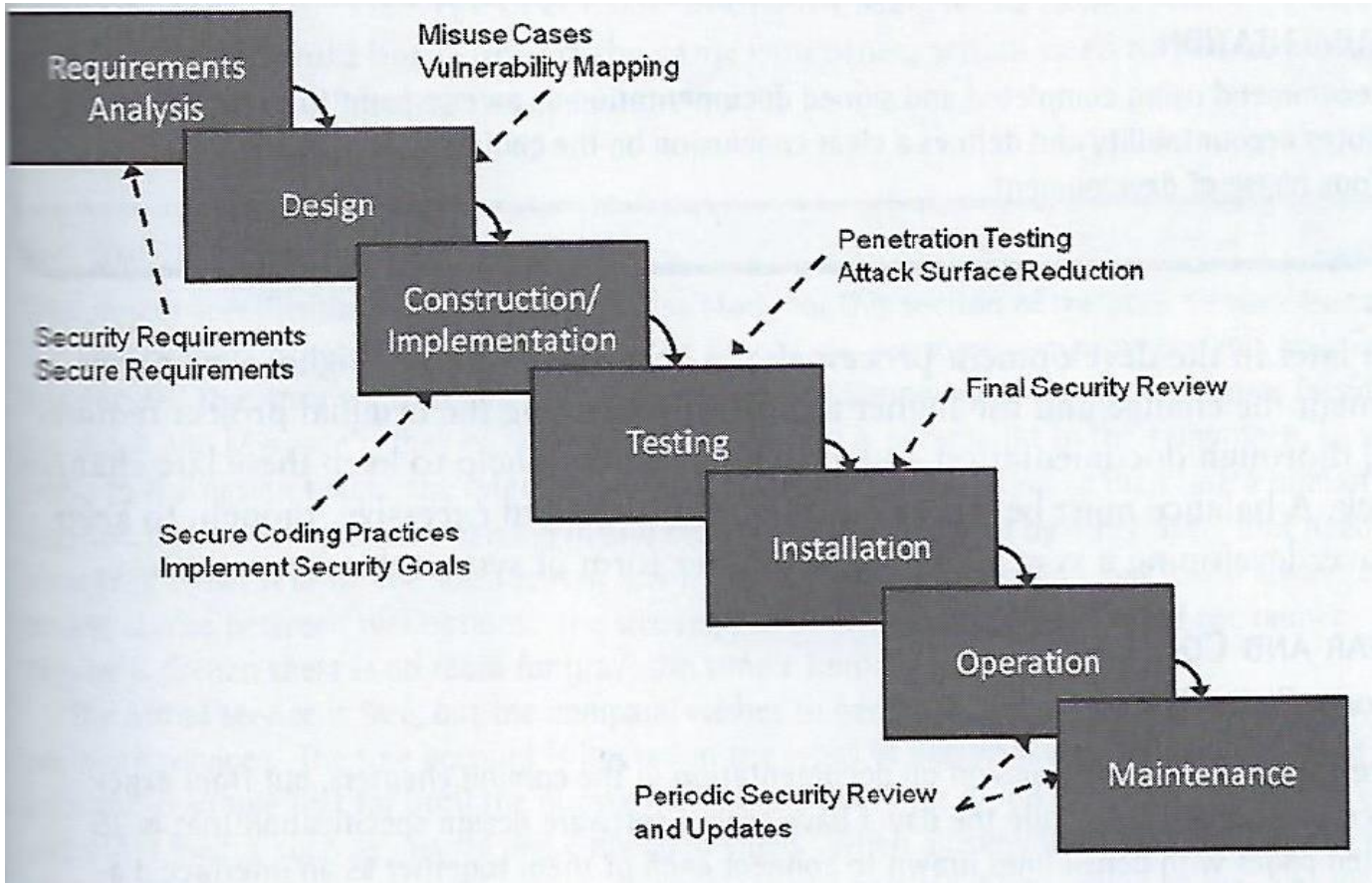
The earlier security is considered, the more likely it is to be implemented well

Baseline of security considerations are:

- Confidentiality
- Integrity
- Availability

Security requirements may also include:

- Data privacy
- Strict authentication and access control
- Uptime and reliability
- Failing safely
- Nonrepudiation



Richardson, T. and Thies, C. (2013) Secure Software Design

Requirements can be improved by answering additional information security questions

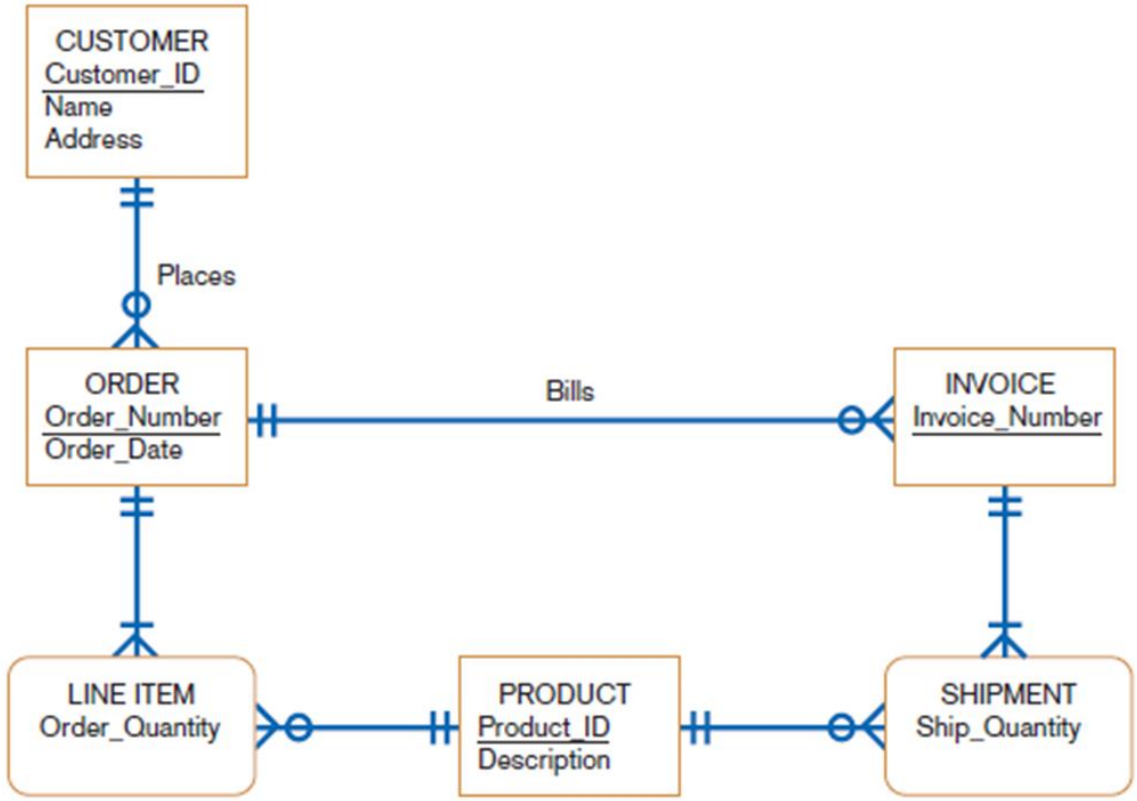
1. **What are the exceptions to the normal situation for this requirement?**
2. **What sensitive information is included in this requirement?**
3. **What are the consequences if the conditions to this requirement are violated?**
4. **What happens if this requirement is intentionally violated?**
5. ***Fail case:*** **What will happen if the requirement is not fulfilled during operation?**
6. ***Consequence of failure:*** **What is the result of the fail case?**
7. ***Associated risks:*** **What sensitive information could be revealed or compromised?**

Database Design Process - deliverables

- Logical database design
  - Accounts for every data element input and output by an information system
    - Normalized tables and relationships are the primary deliverable
- Physical database design
  - Tables and Views
  - Relationships
  - Attribute domains
  - Specified in
    - Relational database model
    - Accompanying Data Dictionary

“Relation” = a named, two-dimensional table of data

- Each relation consists of a set of named columns and an arbitrary number of unnamed rows

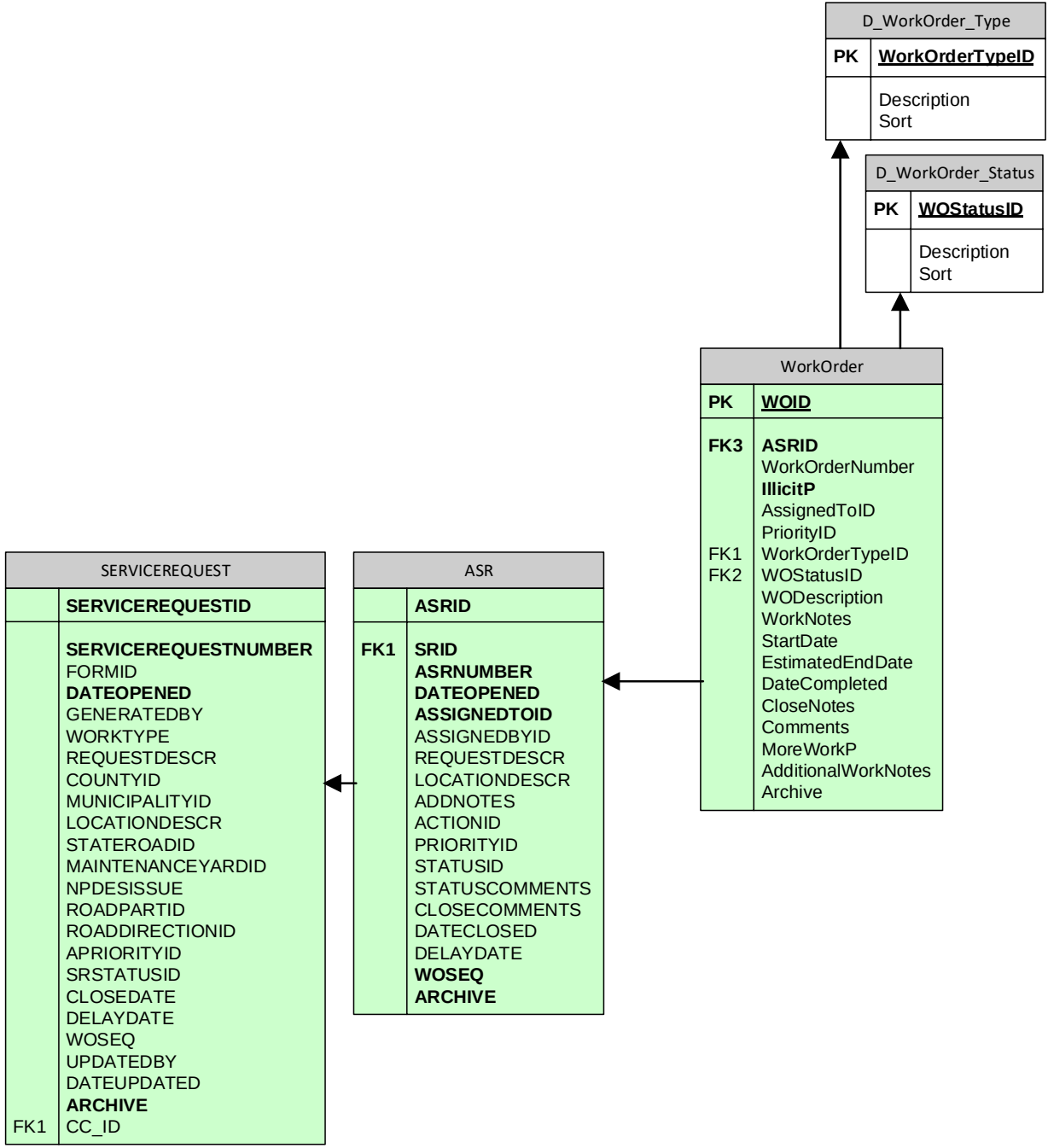


Relations:

CUSTOMER(Customer_ID,Name,Address)
PRODUCT(Product_ID,Description)
ORDER(Order_Number,Customer_ID,Order_Date)
LINE ITEM(Order_Number,Product_ID,Order_Quantity)
INVOICE(Invoice_Number,Order_Number)
SHIPMENT(Invoice_Number,Product_ID,Ship_Quantity)

Relational Tables

- **Table**
  - A relation that contains a minimum amount of redundancy
  - Allows users to insert, modify, and delete the rows without errors or inconsistencies
- **Primary Key**
  - An attribute whose value is unique across all occurrences of a relation
- All relations have a primary key
  - Ensures the rows (i.e. data records) are unique
  - A primary key often in a single attribute, but may be composed of multiple attributes (“compound key”)



Data Normalization

A structured process for organizing data into database tables in such a way that it preserves the relationships among the data

Relational Database Normalization

First Normal Form (1NF)

- Unique rows, no multivalued attributes
- All tables have primary keys
- “No multivalued attributes” means a table cannot have a field with multiple values
- All relations are in 1NF

Second Normal Form (2NF)

- Each nonprimary key attribute is identified by the whole primary key (called full functional dependency)

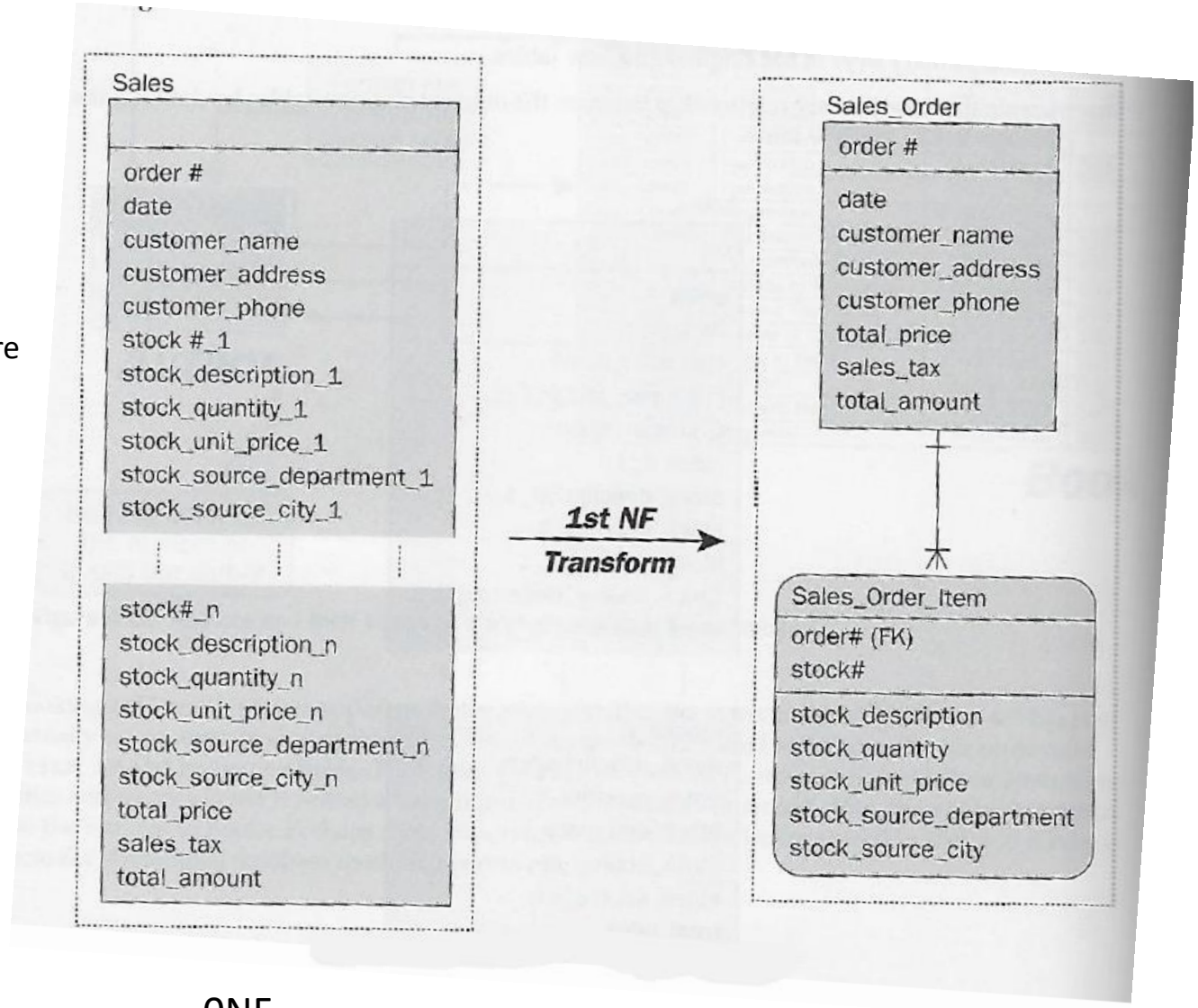
Third Normal Form (3NF)

- Nonprimary key attributes do not depend on each other (i.e. no transitive dependencies)
- **Foreign Key:** an attribute that appears as a nonprimary key attribute (or part of a primary key) in one relation and as a primary key attribute in another relation
- **Referential Integrity:** an integrity constraint specifying that the value (or existence) of an attribute in one relation depends on the value (or existence) of the same attribute in another relation

Relational Database Normalization

First Normal Form (1NF)

- Unique rows, no multivalued attributes
  - All tables have primary keys
  - "No multivalued attributes" means
    - Table cannot have a field with multiple values
    - Need to reconstruct this information with more than one table
- All relations (i.e. tables) are in 1NF



0NF

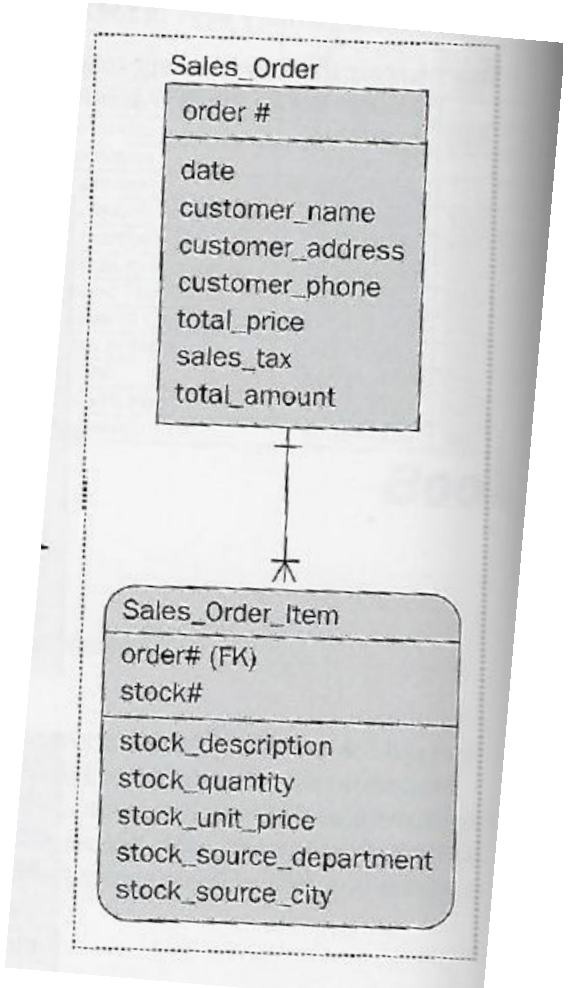
1NF

Relational Database Normalization

Second Normal Form (2NF)

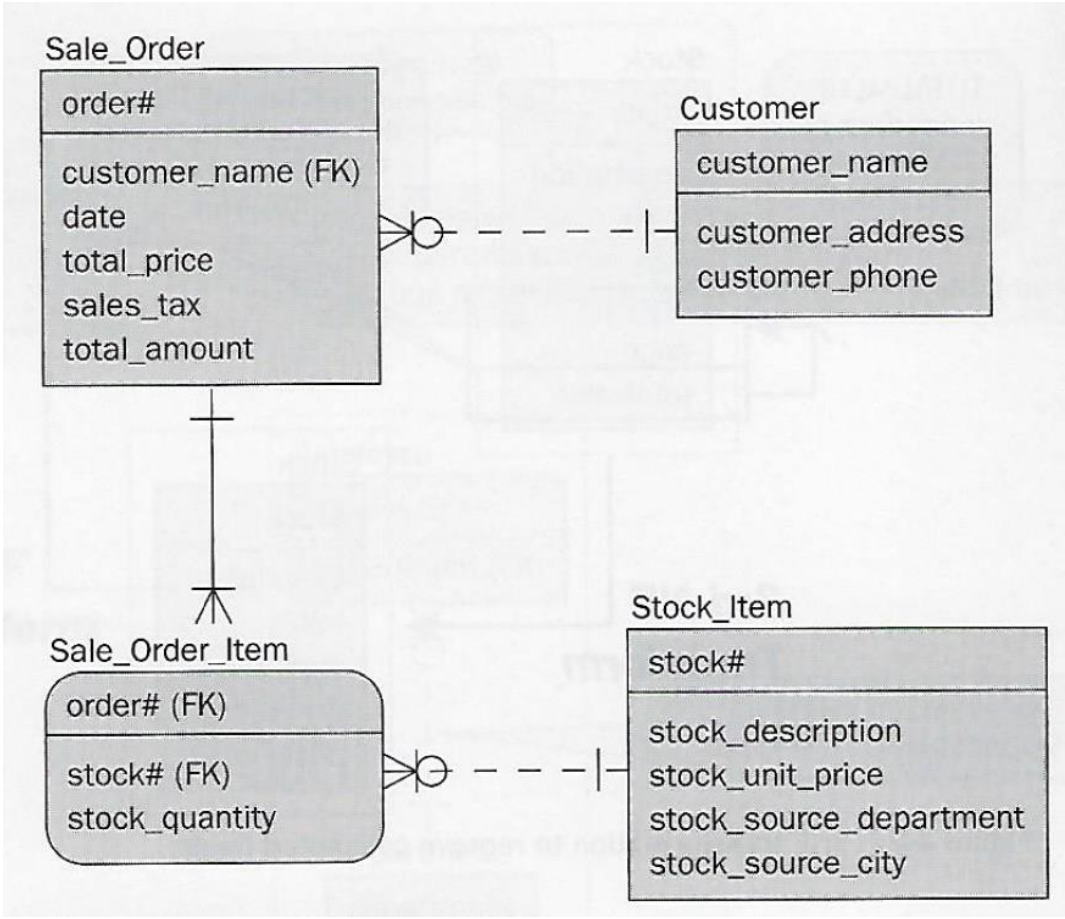
- Each nonprimary key attribute is identified by the whole primary key, called full functional dependency

2 tables in 1 NF



1NF -----> 2 NF

4 tables in 2 NF

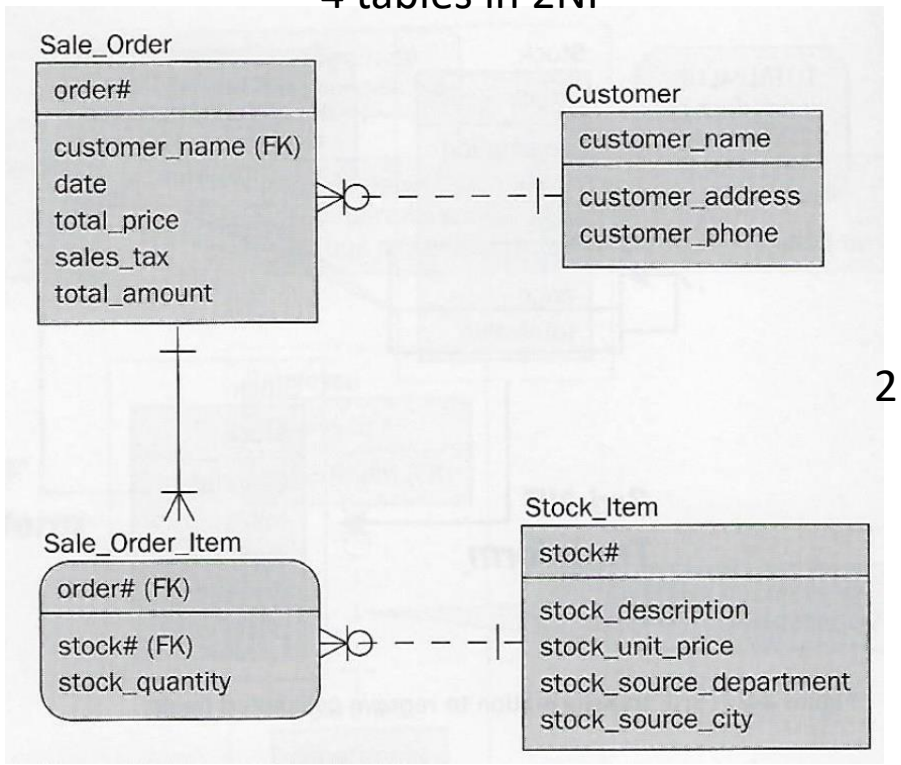


Relational Database Normalization

Third Normal Form (3NF)

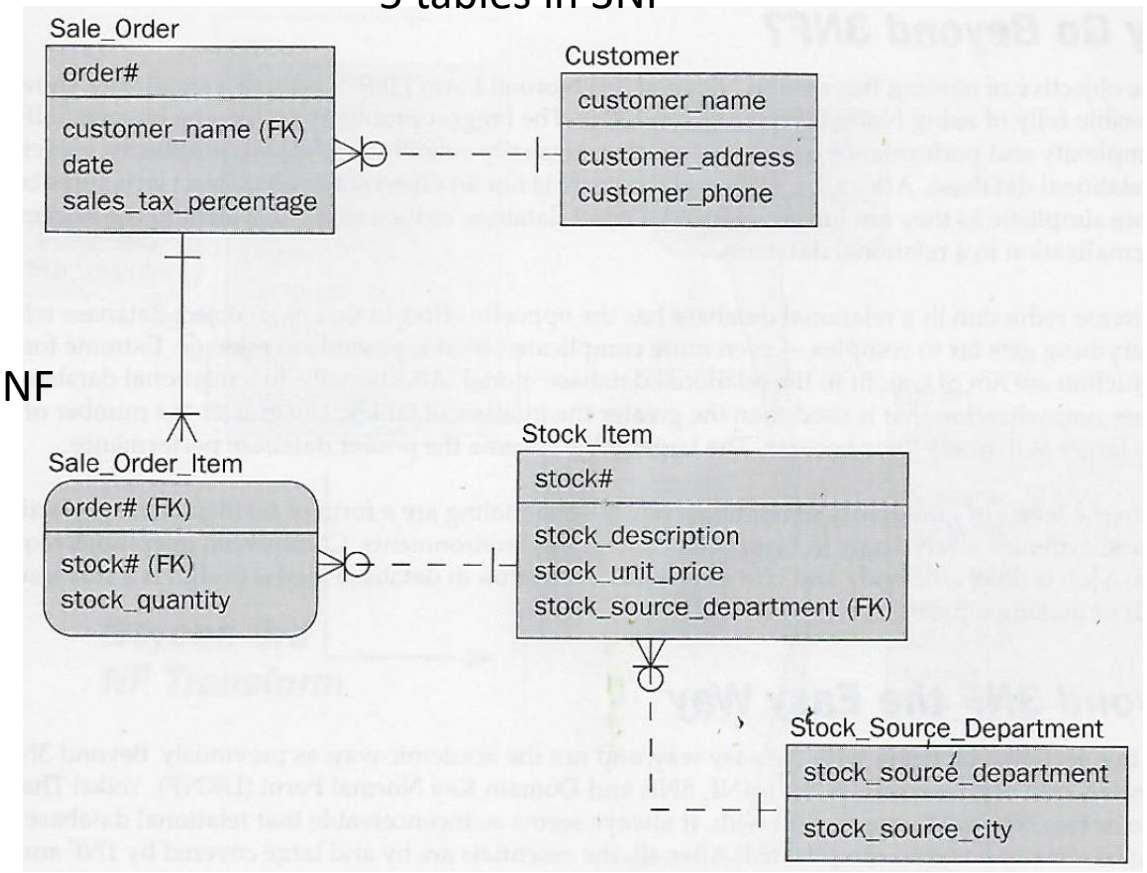
- Nonprimary key attributes do not depend on each other (i.e. no transitive dependencies)

4 tables in 2NF



2NF --------> 3NF

5 tables in 3NF



Relational Database Normalization

What foreign keys exist in the final database structure?

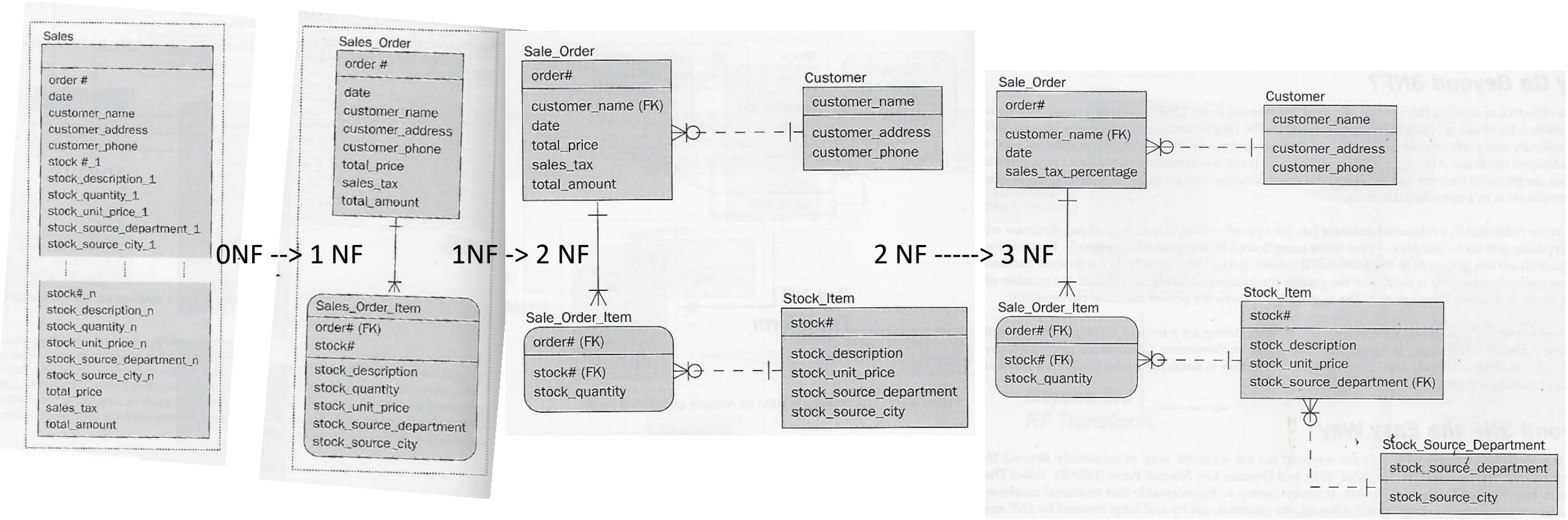
What referential integrity constraints existing in the final database structure?

1 table in 0NF

2 tables in 1NF

4 tables in 2NF

5 tables in 3NF

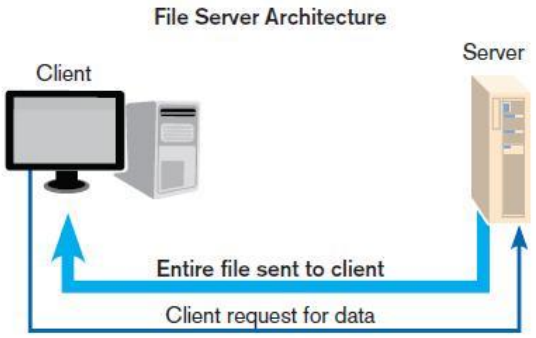


Personal data attributes can be classified based on disclosure potential:

- **Identifiers** Attributes that unambiguously identify the subject (e.g. passport no., social security no., name-surname, etc.)
- **Quasi-identifiers or key attributes** They identify the subject with some ambiguity, but their combination may lead to unambiguous identification (e.g. address, gender, age, tele-phone no., etc.)
- **Confidential outcome attributes** They contain sensitive subject information (e.g. salary, religion, diagnosis, etc.)
- **Non-confidential outcome attributes** Other attributes which contain non-sensitive subject entity information

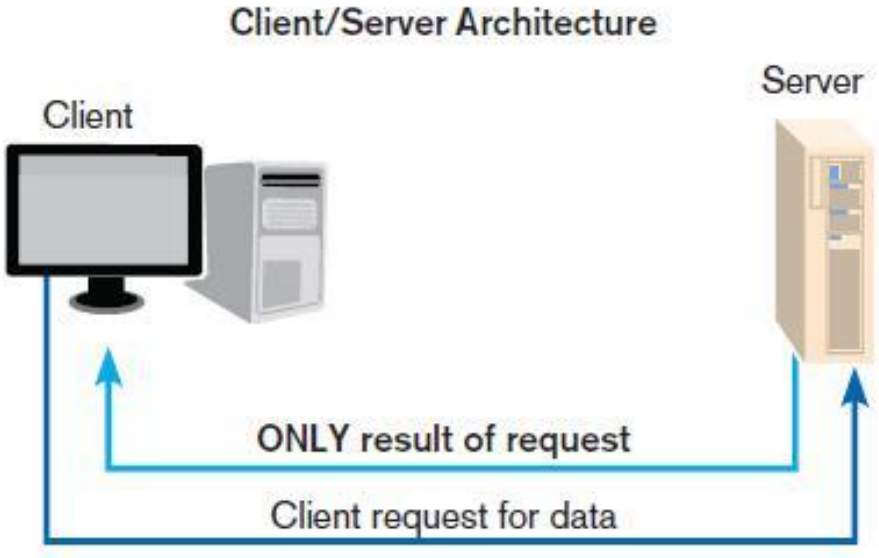
Danezis, G. et al. (2014) “Privacy and Data Protection by Design”, European Union Agency for Network and Information Security (ENISA)

Client-Server Architecture



LAN-based computing environment in which

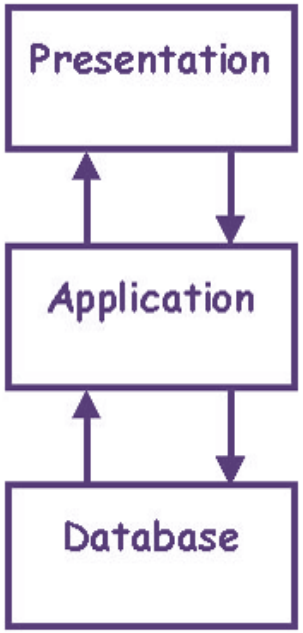
- A central database server or engine performs all database commands sent to it from client workstations
- Application programs on each client concentrate on user interface functions

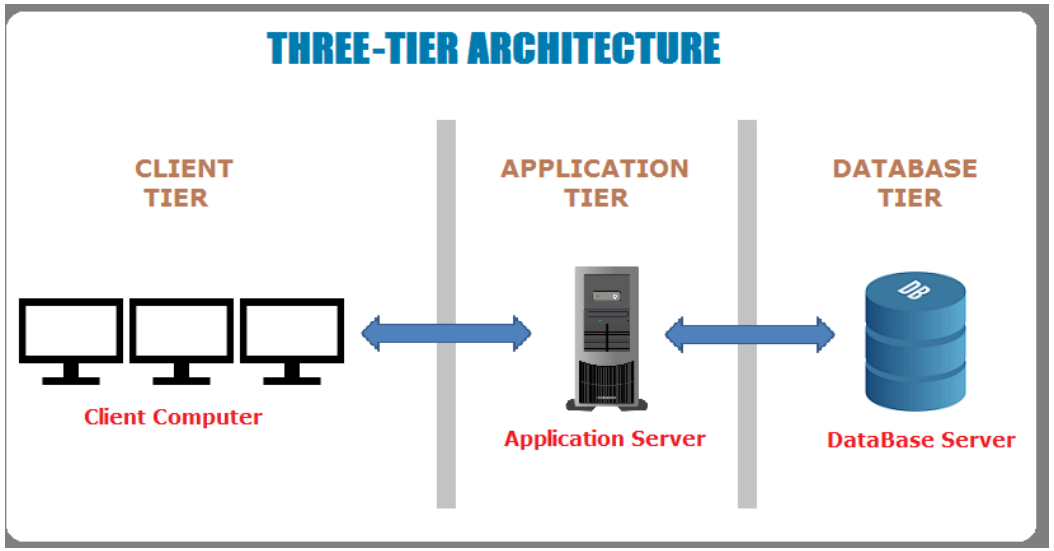


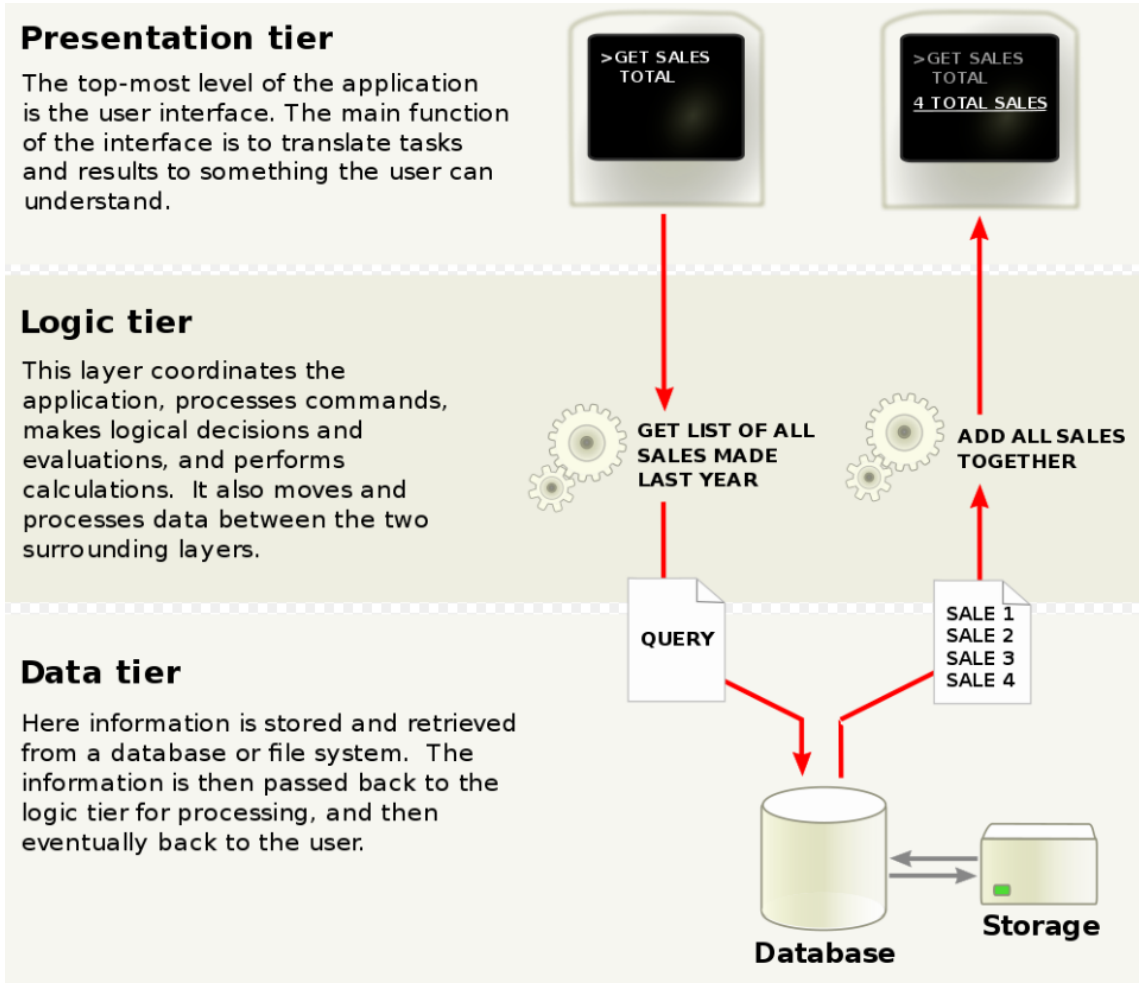
Increased efficiency and control over File server

- Server only sends specific data, not entire files, which saves on network bandwidth
- Computing load is carried out by the server
  - Increasing security
  - Decreasing computing demand on the clients

N-Tier Architecture







Cloud Computing – Service Models

- **Infrastructure as a Service (IaaS):** provides basic processing, storage, and network capabilities
- **Platform as a Service (PaaS):** customers run their own applications, using tools provided by the service provider.
- **Software as a Service (SaaS):** applications are provided by the service provider

- IaaS is most basic service model
- PaaS will include infrastructure along with platform
- SaaS will typically include both the platform and infrastructure on which the application runs

Information System Development Control Stages

Control over applications is conducted at every stage and begins at the start of the development of the information system

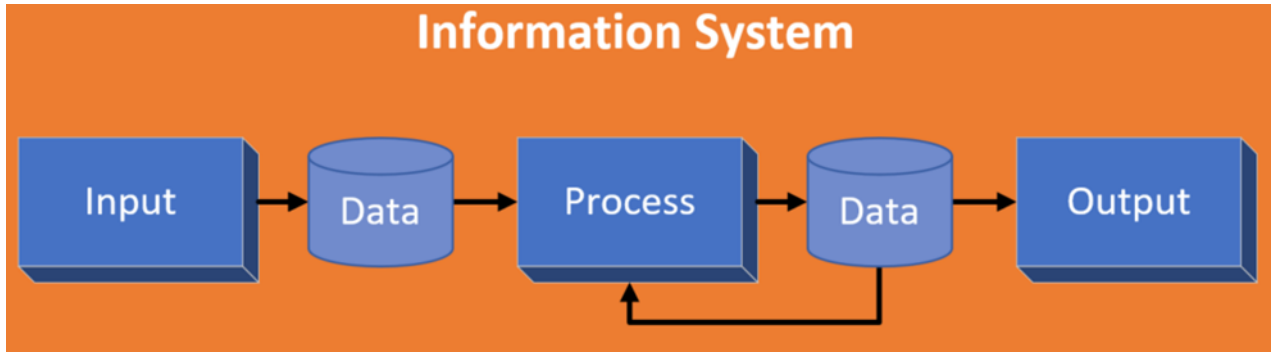
This takes 2 basic forms:

1. Control over the development process itself
2. Ensuring adequate business controls are built into the finished product

Major control stages would include:

- System design
- System development
- System operation
- System utilization

Control Objectives for Business Information Systems



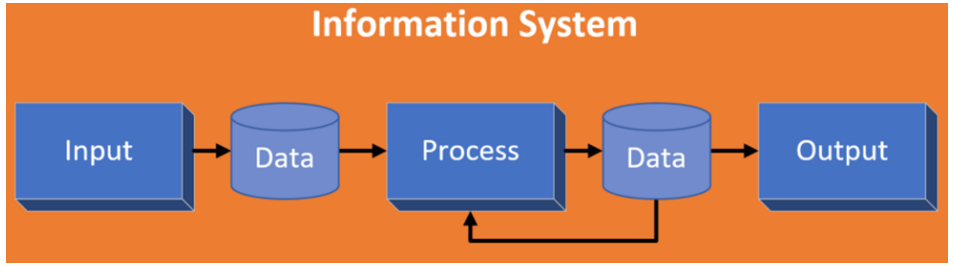
1. Input control objectives
2. Processing control objectives
3. Output control objectives

What is the greatest risk to the effectiveness of information system controls?

Collusion among employees

- *Is an active attack where users attempt to bypass controls such as separate of duties*
- *Is difficult to identify because even well thought out controls may be circumvented*

Control Objectives for Business Information Systems



Input control objectives

- All transactions are
  - initially and completely recorded
  - completely and accurately entered into the system
  - entered only once

- Controls in this area may include:
  - Pre-numbered documents
  - Control total reconciliation
  - Data validation
  - Activity logging and journaling
  - Document scanning and retention for checking
  - Access authorization
  - Document cancellation (e.g. after entry)

Control Objectives for Business Information Systems

Processing control objectives

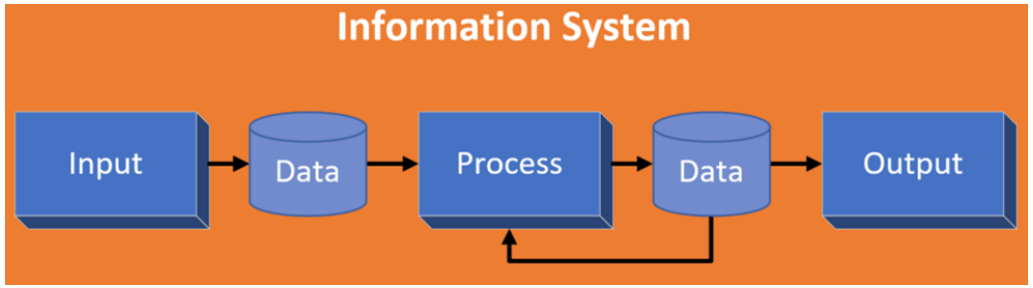
- Approved transactions are accepted by the system and processed
- All rejected transactions are reported, corrected, and re-input
- All accepted transactions are processed only once
- All transactions are accurately processed
- All transactions are completely processed

- Controls over processing may include:
  - Control totals
  - Programmed balancing
  - Reasonableness tests
  - …

Processing controls should be implemented as close as possible to the point of data entry

Data entry should check that entered values fall within valid value ranges ("limit checks")

Limit checks are one type of input validation check that provides a preventative control to ensure that invalid data cannot be entered

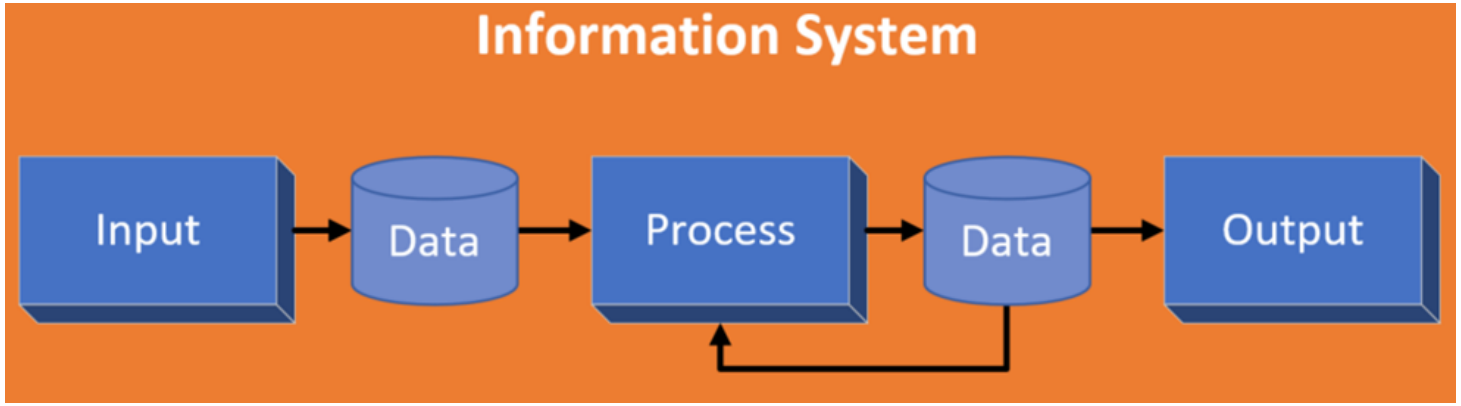


Control Objectives for Business Information Systems

Output control objectives focus on

- Hardcopy
- File outputs and output record sets stored in tables
- Online query files and outputs stored in tables

- Controls over output may include:
  - Assurance that the results of input and processing are output
  - Output is available to only authorized personnel
  - Complete audit trail
  - Output distribution logs



Benefits and risks of using production data in testing…

Using sanitized live transactions as test data enables more realistic testing of systems

Developers will seek to use production transaction files for testing, however…

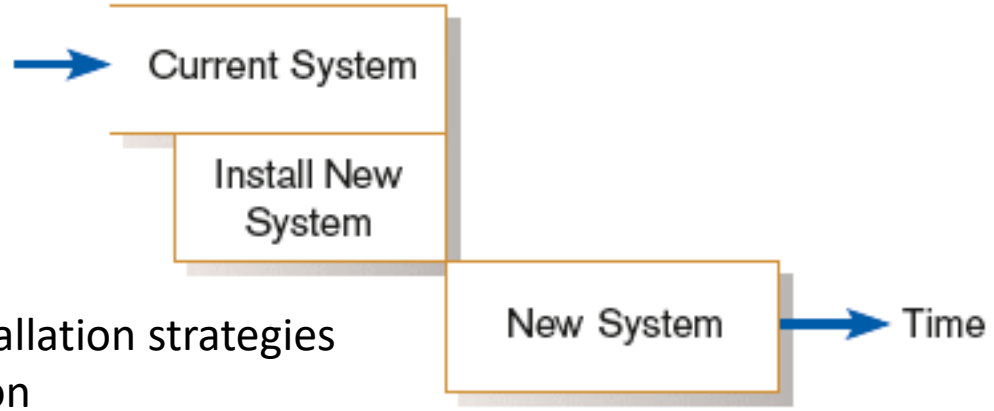
- It is important that all sensitive information in the live transaction file is sanitized to prevent improper data disclosure
- Not all transactions of functionality may be tested if there are no data that meet the requirement

Installation / Implementation

- **Installation/Implementation**: the organizational process of changing over from the current information system to a new one
- Four installation strategies:
  - Direct Installation
  - Parallel Installation
  - Single-location installation
  - Phased Installation

Direct Installation/Implemenation

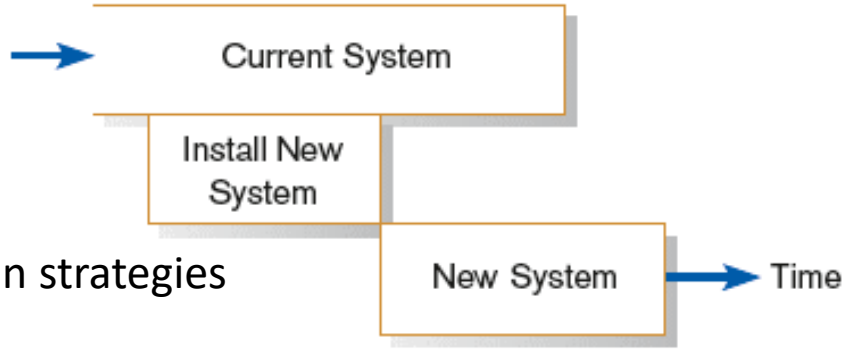
- **Direct installation**: changing over from the old system to a new one by turning off the old system when the new system is turned on



Comparison of installation strategies
(a) Direct installation

Parallel Installation/Implementation

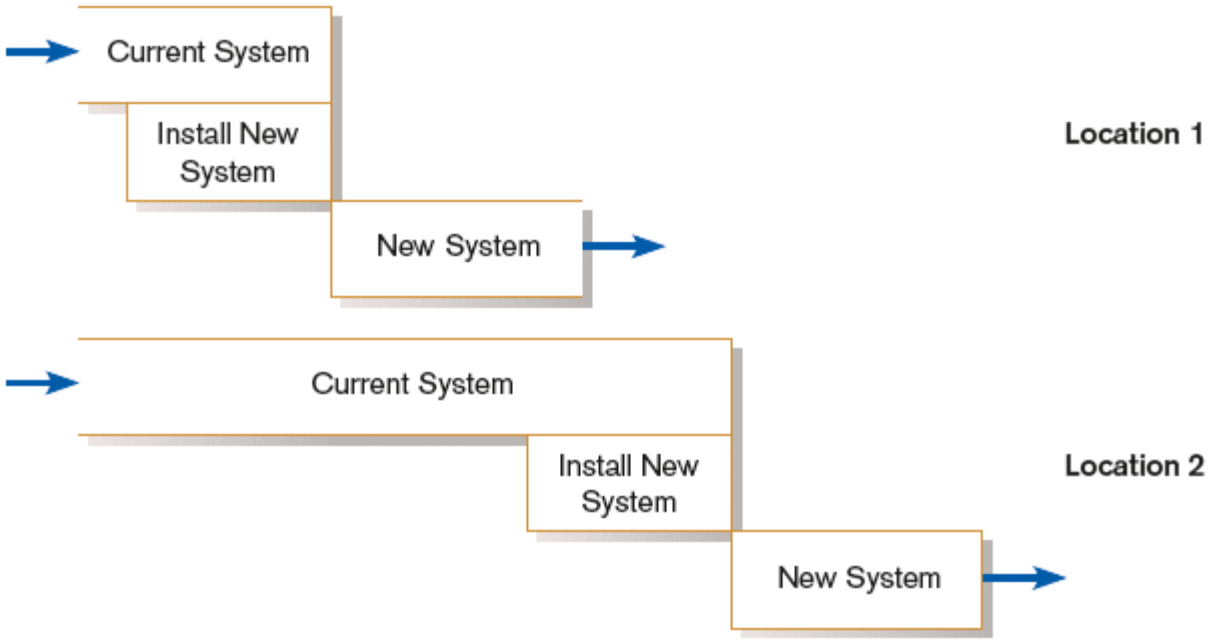
- **Parallel installation**: running the old information system and the new one at the same time until management decides the old system can be turned off



Comparison of installation strategies
(b) Parallel installation

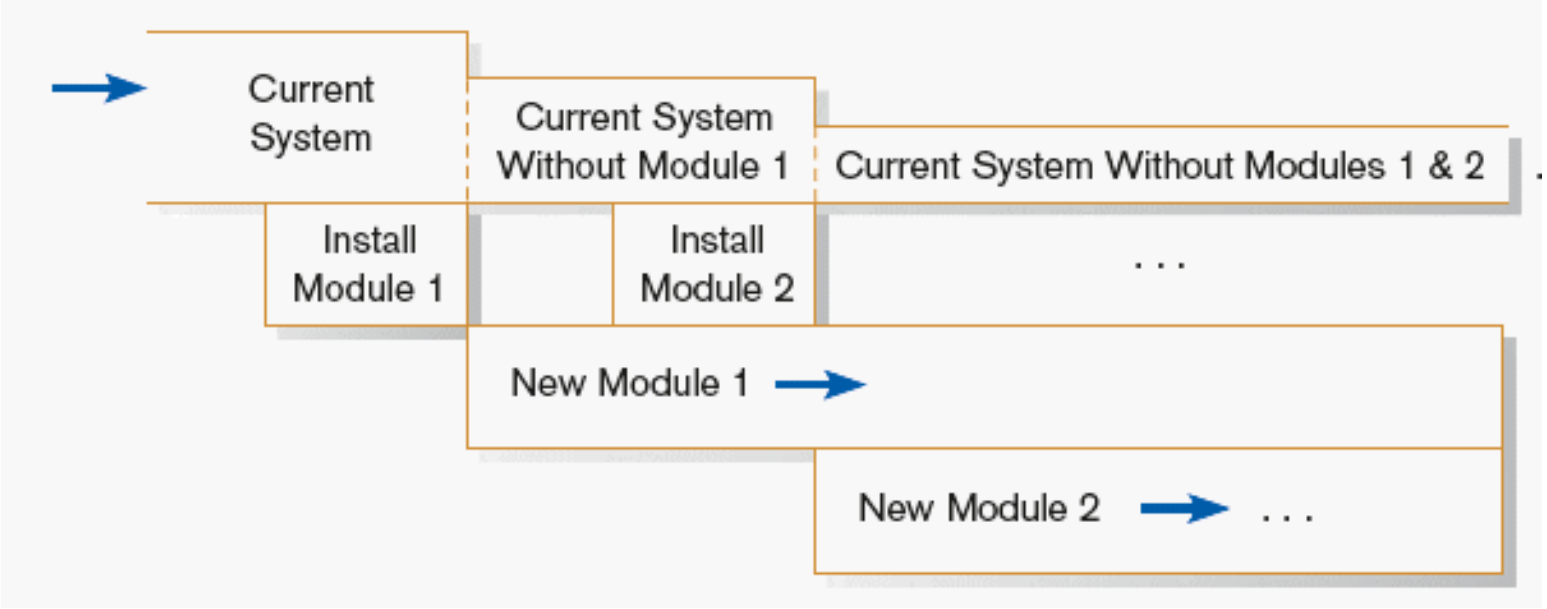
Single-Location Installation/Implementation

- **Single-location installation**: trying out an information system at one site and using the experience to decide if and how the new system should be deployed throughout the organization
- Also known as location or pilot installation



Phased Installation/Implementation

- **Phased Installation**: changing from the old information system to the new one incrementally, starting with one or a few functional components and then gradually extending the installation to cover the whole new system

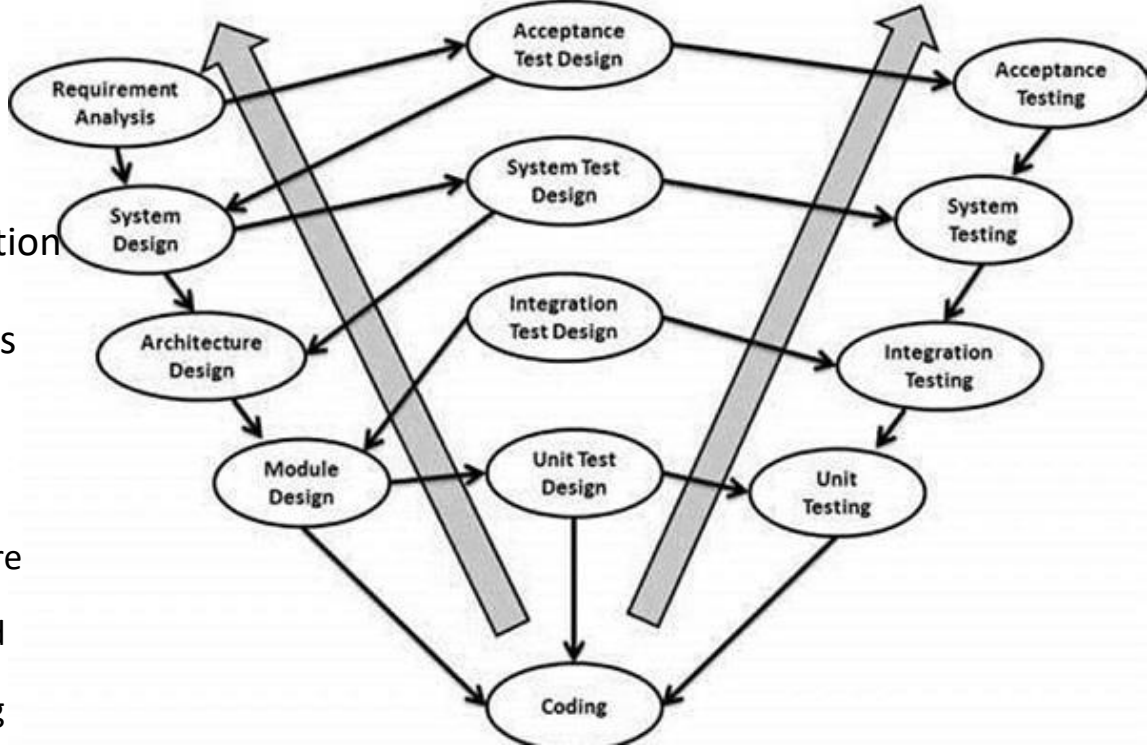


Testing Process

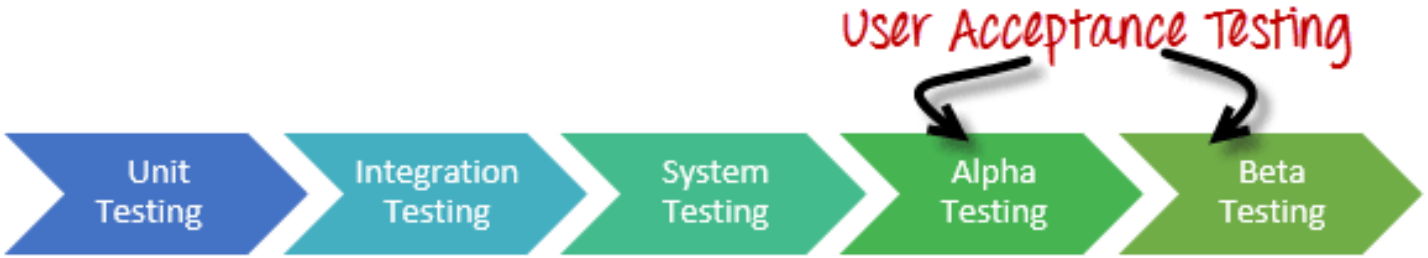
- The purpose of testing is to confirm that the system satisfies the requirements
- Testing must be planned
  - **Bottom-up testing** (most systems are tested early using bottom up testing)
    - Begins testing the smallest units of the system (e.g. programs and modules), and works upward until a the entire system has been tested
    - Advantages: Can be started before all programs are complete; Errors in critical modules can be found early
  - **Top-down testing**
    - Begins testing the breadth and works into the depth of the system
    - Advantages: Tests of major functions and processing conducted early; Interface errors can be detected sooner; Confidence raised in system by seeing a working system

Software Test Types

1. **Unit testing**: Each module is tested alone against its specification in an attempt to discover any errors in the controls structure and internal operation of the code
2. **Interface or Integration testing**: A hardware or software test that evaluates how well the application passes and exchanges information with another system
3. **System testing**: testing of programs that comprise a system to ensure that they function properly
   - **Recovery testing** – Checking system's ability to recover after a software or hardware failure
   - **Security testing** – making sure the system includes appropriate access controls and does not introduce any security holes that may compromise other systems
   - **Load testing** – Testing with large quantities of data to evaluate performance during peak use
   - **Volume testing** – Studying the impact of incremental volumes of data records to determine the maximum volume of data that the application can process
   - **Stress testing** – Studying the impact of incremental numbers of concurrent users/services to determine the maximum number that can be supported
   - **Performance testing** – Comparing the system's performance to other equivalent systems using well-defined bench marks
4. **Final acceptance Testing has 2 parts:**
   - Quality Assurance Testing (QAT) – focuses on the technical aspects of the system, done by IT staff to verify the application meets specifications and works as documented by testing the logical design and technology
   - User Acceptance Testing (UAT) – focuses on the functional aspects of the system, done by users to ensure the system is production ready and satisfies all documented requirements



Acceptance Testing



Process where actual users test a completed information system, the end goal is the users' acceptance of it

- **Alpha testing** is carried out in a lab environment and usually, the testers are internal employees of the organization
- **Beta Testing** of a product is performed by "real users" of the software application in a "real environment" and can be considered as a form of external User Acceptance Testing

Other Types of Testing

- **Pilot testing** – Preliminary test that focuses on specific predetermined aspects of the system. Not intended to replace other testing methods, but to provide a limited evaluation of basic functionalities of the system
- **White box testing** – Assesses the effectiveness of software program logic. Test data used to determine procedural accuracy or conditions of specific program logic (applies to unit and integration testing). Used in a focused manner as exhaustive white-box testing is often cost prohibitive
- **Black box testing** – Integrated testing of the an information system's functional effectiveness without regard to any specific internal program structure. Applicable to integration (interface) and user acceptance testing
- **Functional/validation testing** – A form of system testing that evaluates functionality against detailed requirements to trace back to customer requirements (did they build the right product)
- **Regression testing** – Testing against use-cases in test plan to assure that changes did not introduce new errors
- **Parallel testing** – Testing the same data within the original system (which will be replaced) and the new systems and comparing the results
- **Sociability testing** – Tests to confirm that he new system can operate in its target environment without adversely impacting existing systems. Focuses on application processing examining the interfaces with other systems that can be running at the enterprise and within the user's desktop application environment and with the user's browser

Error Detection Techniques…

No single defect-detection technique is effective by itself

- *Reviewing designs*
- *Evaluating datasets*
- *Testing the completed applications*

Post-implementation review

- **Volume testing** evaluates the impact of incremental volume of records (not users) on a system
- **Stress testing** determines the capacity of the system to cope with an abnormally large number of users or simultaneous operations
- **Load testing** evaluates performance of the system under normal and peak conditions
- **Recovery testing** evaluates the ability of the system to recover after failure
- **Lessons learned** for future projects – a project team has something to learn from each project. It is important for the organization to accumulate lessons learned and integrate them into future projects

Post-implementation

The organizational environment has a significant impact on the success of application system implementation, this includes:

- Alignment between IT and the business
- Maturity of the development process
- Use of change control and other project management tools

Post-implementation review

- During post-implementation review the Project Manager gains feedback about project deliverables and business needs
- Primary purpose of a post-implementation review is to determine if the project objectives have been met
  - Answers the question:
    - ***Does the project's deliverables meet business needs and risk acceptance criteria?***
- Evaluates projected cost-benefits, i.e. return on investment (ROI) to verify that the original business case benefits are delivered