

Information Systems Acquisition and Development

INFO 6008 Week 9



FANSHAWE

Information Systems Acquisition and Development

- **We are covering the following topics:**
- **IT Acquisition and Project Management**: This section provides an overview of IT acquisition and project management control frameworks, practices, and tools.
- **Business Application Development**: Organizations run many applications, and an auditor must prioritize them and then test the ones that are critical to core business operations.
- **Information Systems Maintenance**: This section examines the process of modifying an information system via updates and patches to continually satisfy organizational and user requirements.
- **Outsourcing and Alternative System Development**: Many development methodologies exist, and each one uses a specific approach to development. Some of them may be better fits for an organization than others. An auditor should have a basic understanding of when and how each is used.

Information Systems Acquisition and Development

- Life cycle management requires that auditors understand the systems development life cycle (SDLC).
- Auditors can become deeply involved in the SDLC process.
- Auditors are responsible for helping ensure that sufficient controls are designed during this process and that these controls work as expected. Controls must be tested, and the overall design plan must be reviewed. Not all projects use the same development method. Today many alternative development methods—such as prototyping, rapid application development, and agile development—are used.
- An auditor must understand each of these in order to fulfill his job duties. After the rollout of new applications, an auditor's job is not done. Systems require maintenance, review of changes, and review and redesign of processes.
- Throughout the life cycle, auditors play a key role.

IT ACQUISITION AND PROJECT MANAGEMENT

- IT acquisition and project management are two items that go well together.
- The acquisition of IT is by nature a temporary endeavor and as such falls under project management guidelines.
- Projects can be large or small, and they can involve hardware, software, or networks.
- A project can even be used to create a product or service.

IT ACQUISITION

- IT acquisition involves the expenditure of funds for information technology.
- IT acquisition can be a single purchase or might involve multiple purchases over many years.
- One key decision that must be made is whether the organization will build or buy a solution.
- **A business case should be made either way.**
- The decision typically comes down to time, cost, and availability of a predesigned substitute.
-

Software Escrow Agreements

- If the decision is made to make an IT acquisition, vendor management must be considered.
- What if the software developer goes bankrupt or is no longer in business?
- How is the organization supposed to maintain or update the needed code?
- These concerns can be addressed by a *software escrow* agreement, which allows an organization to maintain access to the source code of an application if the vendor goes bankrupt.
- Although the organization can modify the software for continued use, it can't steal the design or sell the code on the open market.
- A *software escrow* agreement protects you in case things go wrong and the vendor is no longer in business.

Software Licensing

- Escrow is not the only IT acquisition issue to consider.
- Another concern is licensing. Intellectual property rights issues have always been hard to enforce.
- Just consider the uproar that Pirate Bay has caused over issues of intellectual property and the rights of individuals to share music and files. The software industry has long dealt with this issue.
- From the early days of computing, some individuals have been swapping, sharing, and illegally copying computer software. The unauthorized copying and sharing of software is considered software piracy, and it is illegal.

Software Licensing

- Software piracy is big business, and accumulated loss to property owners is staggering. An auditor needs to review software licensing and usage. The cost of noncompliance can be huge.
- Major software companies formed the Software Protection Association, which is one of the primary bodies that actively fight to enforce licensing agreements.
- Microsoft and others are also actively fighting to protect their property rights. The Software Alliance (BSA) and the Federation Against Software Theft are international groups targeting software piracy. These organizations target organizations of all sizes, from small, two-person companies to large, multinational organizations.

Software Licensing

Software companies are also fighting back by making clear in their licenses what a user can and cannot do with the software. For example, Windows 10 requires users to accept all updates.

License agreements can actually be distributed in several different ways, including the following:

- **Click-wrap license agreements:** Found in many software products, these agreements require you to click through and agree to terms to install the software product. These are often called *contracts of adhesion* because such contracts are a “take it or leave it” proposition.
- **Master license agreements:** These agreements are used by large companies that develop specific software solutions that specify how the customer can use the product.
- **Shrink-wrap license agreements:** These agreements were created when software started to be sold commercially and are named for the fact that breaking the shrink wrap signifies your acceptance of the license.

Software Licensing

- Even with licensing and increased policing activities by organizations such as the BSA, improved technologies make it increasingly easy to pirate software, music, books, and other types of intellectual property.
- These factors and the need to obtain compliance with two World Trade Organization (WTO) treaties led to the passage in 1998 of the Digital Millennium Copyright Act (DMCA). Salient highlights include the following:
- The DMCA makes it a crime to bypass or circumvent antipiracy measures built into commercial software products.
- The DMCA outlaws the manufacture, sale, or distribution of any equipment or device that can be used for code-cracking or illegally copying software.

Software Licensing

- The DMCA provides exemptions from anti-circumvention provisions for libraries and educational institutions under certain circumstances; however, for those not covered by such exceptions, the act provides for penalties up to **\$1 million and 10 years in prison.**
- The DMCA provides Internet service providers exceptions from copyright infringement liability to enable transmissions of information across the Internet.

Project Management

Projects are temporary endeavors. The purpose of project management, which is a one-time effort, is to meet a defined goal of creating a specific product, service, or result. When all the objectives are met, a project is terminated. Projects have unique attributes:

- A unique purpose
- A temporary nature
- A primary customer and/or sponsor
- Uncertainty

Project Management

- Projects are constrained in a number of ways that you need to understand:
- **Scope:** How much work is defined? What do the sponsor and the customer expect from this project?
- **Time:** How long is this project scheduled to run? Does it have a defined schedule? When does the product need to be launched, or when does the service need to be operational? Answering these questions will help determine how long the project will run.
- **Cost:** How much money is this project expected to cost? Has the sponsor approved it?

Project Management

- Many approaches and standards exist to meet this triple constraint.
- Regardless of which one or ones are used, an auditor must verify that business requirements are going to be met and that the project is being achieved in a cost-effective manner while managing potential risk.
- The most well known of these approaches and standards are the Project Management Body of Knowledge (PMBOK; IEEE Standard 1490), Prince2 (Projects in Controlled Environments), and standards from the Project Management Institute (PMI).
- Each of these is somewhat different, but they share common attributes.

Roles, Responsibility, and Structure of Project Management

- An auditor should play an active part in the project management process and should know the various roles and responsibilities.
- It is important that an auditor conduct periodic reviews to determine whether a project is progressing in accordance with project plans.
- Auditors should understand who is responsible and be able to identify key stakeholders, including the following:
- **Senior management:** Managers who provide the necessary resources to complete a project.
- **Stakeholder:** A person, group, or business unit that has a share or an interest in the project activities.

Roles, Responsibility, and Structure of Project Management

- **Project steering committee or oversight board:** The group that is ultimately responsible for ensuring that the stakeholders' needs are met and overseeing the direction and scope of the project. The committee acts as project-oversight board.
- **Project sponsor:** A person who works with the project manager to ensure success and is responsible for allocating funding for the project.
- **Project manager:** A person who is responsible for day-to-day management of the project team.
- **Project team:** A person who is responsible for performing operational tasks within the project.

Roles, Responsibility, and Structure of Project Management

- **Quality assurance:** A person who is responsible for reviewing the activities of the project management team and ensuring that output meets quality standards. This role/group does not necessarily act as part of the project team as a whole. QA activities can be carried out by parties outside the project team, including auditors.

Projects must take on an organizational form or framework, which can be either loosely structured or very rigid.

In the latter case, the program manager has complete authority over the group and is assigned to the group for the duration of the project.

Project Organizational Forms

Form	Description
Pure project	Formal authority is held by the project manager. The team may have a dedicated project work area.
Influence	The project manager has no real authority, and the functional manager remains in charge.
Weak matrix	The project manager has little or no authority and is part of the functional organization.
Balanced matrix	The project manager has some functional authority, and management duties are shared with functional managers.
Strong matrix	In this more expensive model, the project has members assigned for dedicated tasks. The advantage is that this offers a greater level of authority.

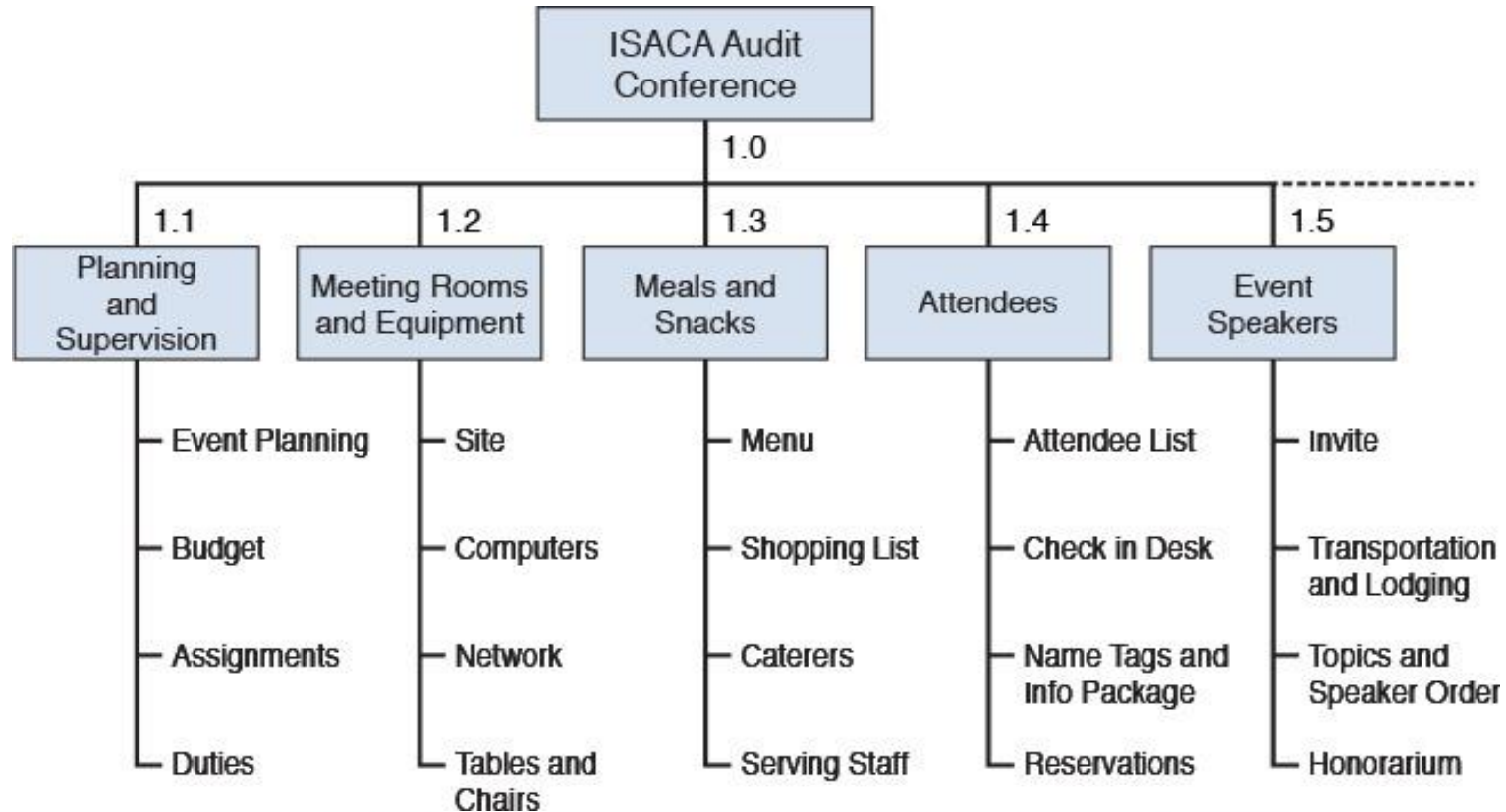
Project Culture and Objectives

- Each team has a unique culture. Project managers can play a part in developing a healthy culture by holding a kick-off meeting that allows team members to get to know each other.
- The kick-off meeting also gives the project manager a forum in which to discuss the goals of the project and the tasks that need to be completed to meet the desired goal. The program manager might also want to use this time to perform some team-building activities, such as having the group develop a team name, establish a mission statement, or create a project logo.
- As the team progresses, it typically goes through four stages:
 - 1. Forming
 - 2. Storming
 - 3. Norming
 - 4. Performing

Project Culture and Objectives

- Throughout these ups and downs, the team must stay clearly focused on the deliverables. Some deliverables are considered main objectives, and others are considered non-objectives.
- The main objectives are **directly** linked to the success of the project; non-objectives add value by clarifying and defining the main objectives.
- The project management process usually starts with the team working on an *object breakdown structure (OBS)*, which defines each component of the project and the relationships of the components to each other. The OBS can help make sure the team doesn't leave anything out and that all requirements are clearly defined.
- Next, a *work breakdown structure (WBS)* can be developed. A WBS is process oriented and shows what activities need to be completed, in a hierarchical manner. It allows for easy identification of tasks that need to be completed.
- One advantage of a WBS is that tasks are defined as achievable work units.

Work Breakdown Structure



Making the Business Case for Investment

- There are two primary ways to make a case for business investment:
 1. **Business case analysis:** A business case analysis is the more in-depth of the two. A business case analysis should contain a description of the objectives, possible alternatives, the anticipated impact in terms of improvement or dollar savings of each option, as well as a cost–benefit assessment.
 2. **Feasibility study:** A feasibility study examines a variety of items such as demographic, geographic, and regulatory factors. A feasibility study usually offers a go/no go recommendation.

Making the Business Case for Investment

- Both methods can help stakeholders determine if an investment should be made.
- If an investment is recommended, some form of requirements analysis must be conducted. This is simply the process of determining user expectations for a new or modified product.
- If a product is software, this is usually referred to as the functional specification. The requirements analysis process should occur early in a project and requires the stakeholders, auditors, users, and others to meet and discuss the requirements for the proposed product or process. This can be challenging as all the customers and their interests are brought into the process of determining what is a must-have versus what is a nice-to-have.

Making the Business Case for Investment

- **Security requirements should be determined early in a project.** It is much cheaper to build in security at the beginning of a project than to add it later.
- One way to think about the cost of security is to think of security in terms of quality. Having a product that is released and is immediately found to be vulnerable can have a real impact in tangible and intangible ways.
- Measuring cost accurately is not always easy. Some believe that the cost to correct security flaws at the requirements level is up to 100 times less than the cost to correct security flaws in fielded software.
- Auditors should be aware that the cost of not building in security up front is magnified by the expense of developing and releasing patches.
-

Return on Investment

- Historically, return on investment (ROI) has been used to determine the profitability ratio. There are several ways to determine ROI, but the most frequently used method is to divide net profit by total assets. So, if your net profit is \$10,000 and your total assets are \$50,000, your ROI is .20, or 20 percent.
- However, an auditor must also consider ROI in terms of the non-financial benefits of IT investments. Such benefits might include impacts on operations, increased sales, support for modern technology, mission performance, and improved customer satisfaction.
- Before a project begins, stakeholders must make decisions based on the perceived value of the investment. Value is examined by looking at the relationship between the organization's costs and the expected benefits. The greater the benefits in relationship to cost, the greater the value of the IT project.

Return on Investment

Besides ROI, some other measures of profitability include the following:

- **Payback period:** The amount of time required for the benefits to pay back the cost of a project.
- **Net present value (NPV):** The value of future benefits restated in terms of today's money.
- **Total cost of ownership (TCO):** A financial estimate used to help determine the direct and indirect costs of a product or system.
- **Internal rate of return (IRR):** The benefits restated as an interest rate.

When a project is finished, an auditor should perform a gap analysis. This type of analysis involves comparing actual performance with potential or desired performance. In the long term, there is even more work as the longer an item has been released or available, the more vulnerable it becomes. An auditor needs to verify that proper vulnerability management is being performed.

Return on Investment

Besides ROI, some other measures of profitability include the following:

- **Payback period:** The amount of time required for the benefits to pay back the cost of a project.
- **Net present value (NPV):** The value of future benefits restated in terms of today's money.
- **Total cost of ownership (TCO):** A financial estimate used to help determine the direct and indirect costs of a product or system.
- **Internal rate of return (IRR):** The benefits restated as an interest rate.

When a project is finished, an auditor should perform a gap analysis. This type of analysis involves comparing actual performance with potential or desired performance. In the long term, there is even more work as the longer an item has been released or available, the more vulnerable it becomes. An auditor needs to verify that proper vulnerability management is being performed.

Project Management Activities and Practices

- Good project management practices can help ensure that the goals of a project are met. Project risk is a real concern. Project management faces three constraints:
- **Scope:** The scope of a project can be better defined by understanding the areas, activities, and human resources needed to complete the project. For example, software projects must define how big the applications will be. Will the project involve a few thousand lines of code or millions of lines of code?
- **Time:** Time can be better established by building a project timeline that lists each task and specifies a time frame for each.
- **Cost:** Cost can be determined by examining the lines of code, the number of people on the project team, and the time needed for each phase of the project.

Project Initiation

- Project initiation is the first stage.
- Sometimes attention and effort are focused on the endpoint and final deliverable.
- Projects must be managed carefully during the initiation stage. At this point, a project sponsor seeks to obtain funding and approval for a project through a project initiation document (PID).
- The PID is used to obtain authorization for the project to begin.
- It justifies the project to management, clearly defines the scope of the project, documents all roles and responsibilities, and sets up and runs a project office environment.

Project Planning

- Project planning is the part of project management that relates to schedules and estimation.
- A project manager must develop a realistic estimate of how much time a project will take and determine what tasks must be accomplished.
- Project planning involves not just time but also the identification and quantification of all required resources.
- Task management is not just about handing out tasks to each member of the team; it is about determining who is most capable of accomplishing each task.
- Program Evaluation and Review Technique (PERT) charts and Gantt charts, discussed later, are two tools that help with this.

Project Planning

- Project planning requires that the sequence of tasks be determined.
- Some tasks can be performed in any order, whereas others must flow in a specific order. For example, building a house requires the foundation to be laid before the walls can be built. Each of these tasks must have a time estimate performed; the project manager must determine how long each task will take.
- The resources needed will vary depending on the task. As in the previous example, a foundation requires concrete and rebar, while walls require wood and nails. The following sections look at some of the pieces of project planning.

Software Cost Estimation

- Most of us put a lot of effort into cost estimation in our personal lives. When considering a new job offer, most people look closely at the cost of living in a different area, and when shopping for a car, most people check with several dealerships to find the best deal. The business world is constrained by similar budget factors. These components drive up the cost of software:
- **Source code language:** Using an obscure or unpopular language will most likely drive up costs.
- **Size of the application:** The size or complexity of an application has a bearing on its cost. As an example, the *level of security needed* will affect the complexity of a given application. This also has a direct correlation to the *scope* of a project.


Software Cost Estimation

- **Project time constraints:** If a project needs to be completed in one month versus the projected three months, this might mean that more overtime needs to be paid, along with fees for rushed services.
- **Computer and resource accessibility:** If resources are available only during certain times, the output of the project team will most likely be reduced.
- **Project team experience:** Every individual has a learning curve, and this means inexperienced team members cost extra.
- **Level of security needed:** A project that needs very high levels of security controls takes additional time and effort to develop.

Software Cost Estimation

- One early cost model, developed by Barry Boehm, is known as the Constructive Cost Model (COCOMO).
- It was replaced by COCOMO II. This model considers “what if” calculations to determine how changes to human and other resources affect project cost.
- The next slide shows the COCOMO II model.
- You can find a web version of the tool online at <http://csse.usc.edu/tools/cocomoii.php>, and you can also download it for use with standard spreadsheet applications such as Microsoft Excel.

COCOMO II Software Estimation


COCOMO II - Constructive Cost Model

Software Size Sizing Method Source Lines of Code ▾

SLOC % Design Modified % Code Modified % Integration Required Assessment and Assimilation (0%-8%) Software Understanding (0%-50%) Unfamiliarity (0-1)

New	<input type="text"/>					
Reused	<input type="text"/>	0	0	<input type="text"/>	<input type="text"/>	
Modified	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Software Scale Drivers

Precedentedness	Nominal ▾	Architecture/Risk Resolution	Nominal ▾	Process Maturity
Development Flexibility	Nominal ▾	Team Cohesion	Nominal ▾	

Software Cost Drivers

Product		Personnel		Platform
Required Software Reliability	Nominal ▾	Analyst Capability	Nominal ▾	Time Constraint
Data Base Size	Nominal ▾	Programmer Capability	Nominal ▾	Storage Constraint
Product Complexity	Nominal ▾	Personnel Continuity	Nominal ▾	Platform Volatility
Developed for Reusability	Nominal ▾	Application Experience	Nominal ▾	Project
Documentation Match to Lifecycle Needs	Nominal ▾	Platform Experience	Nominal ▾	Use of Software Tools
		Language and Toolset Experience	Nominal ▾	Multi-site Development
				Required Development Schedule

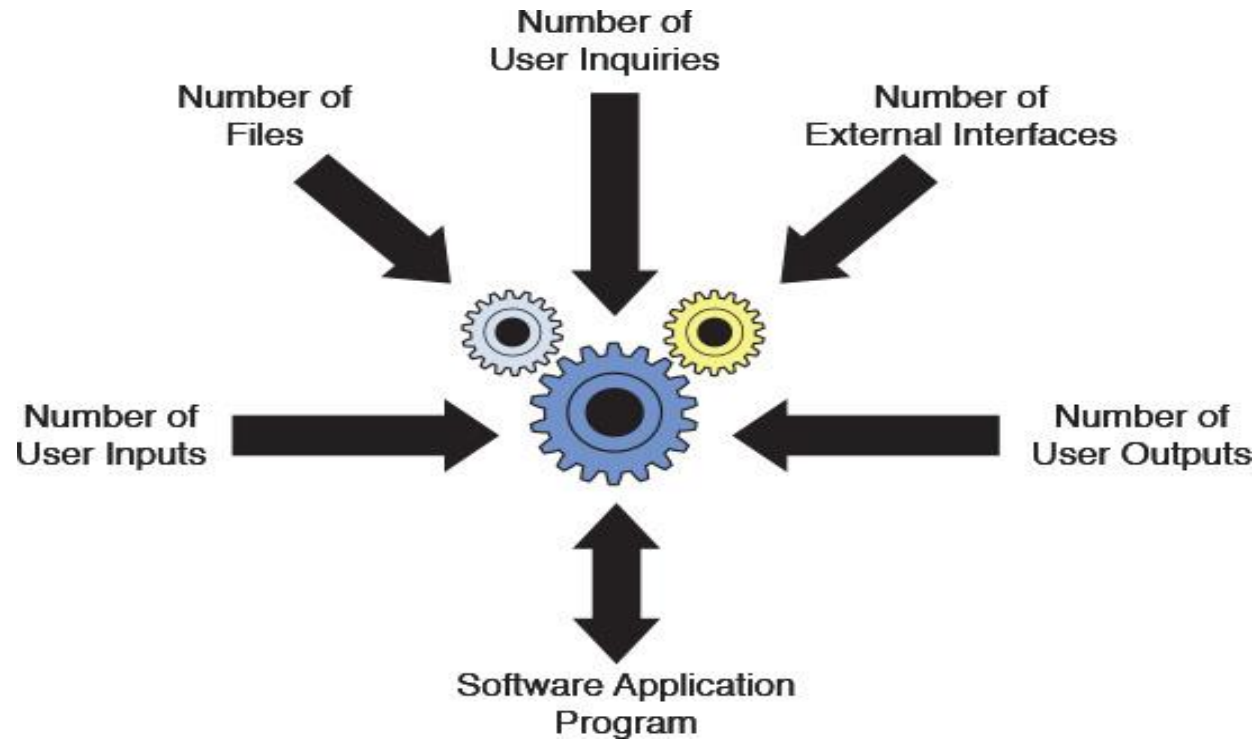
Software Size Estimation

- Originally, software sizing was done by counting *source lines of code* (SLOC). This method of software sizing was developed because early programs were written in FORTRAN and other line-oriented languages.
- To get an idea of how big programs can be, Linux 2.6 had about 5.7 million lines of code. With programs of such size, counts are usually done in *kilo lines of code* (KLOC).
- Both SLOC and KLOC determine cost solely on one factor—length of code—which does not work as well in modern development programs because additional factors affect overall costs, such as the complexity of the application/program being written.

Software Size Estimation

- Considering that development packages can generate hundreds of lines of code from only a few mouse clicks demonstrates that this model is not as useful as in years past.
- One modern way of estimating software size is to use *function point analysis (FPA)*.
- FPA is a method that the ISO has approved as a standard to estimate the complexity of software.
- FPA can be used to budget application-development costs, estimate productivity after project completion, and determine annual maintenance costs. FPA is based on the number of inputs, outputs, interfaces, files, and queries.

Software Size Estimation - Function Point Analysis



Software Size Estimation

- Per ISACA, function points are first computed by completing a table, as in the next slide.
- The purpose of the table is to determine whether the task is simple, average, or very complex.
- One way to determine this subjective weighting factor is to apply the Halstead Complexity Measures: the number of user inputs, number of user outputs, number of user inquiries, number of files, and number of external interfaces.
- Take a moment to review the following table.

Computing Metrics of Function Point Analysis

Measurement Parameter	Count	Simple	Weighing Factor Average	Complex	Results
Number of user inputs		X3	4	6	=____
Number of user outputs		X4	5	7	=____
Number of user inquiries		X3	4	6	=____
Number of files		X7	10	15	=____
Number of external interfaces		X5	7	10	=____
Total count:					

Software Size Estimation

- If an organization decides to use function point analysis, it must develop criteria for determining whether an entry is simple, average, or complex.
- When a FPA table is completed, the organization can run its computed totals through an algorithm to determine factors such as reliability, cost, and quality:
- $\text{Productivity} = \text{Function points} / \text{Person-month}$
- $\text{Quality} = \text{Defects} / \text{Function points}$
- $\text{Cost} = \$ / \text{Function points}$
- With these calculations completed, the project team can identify resources needed for each specific task.

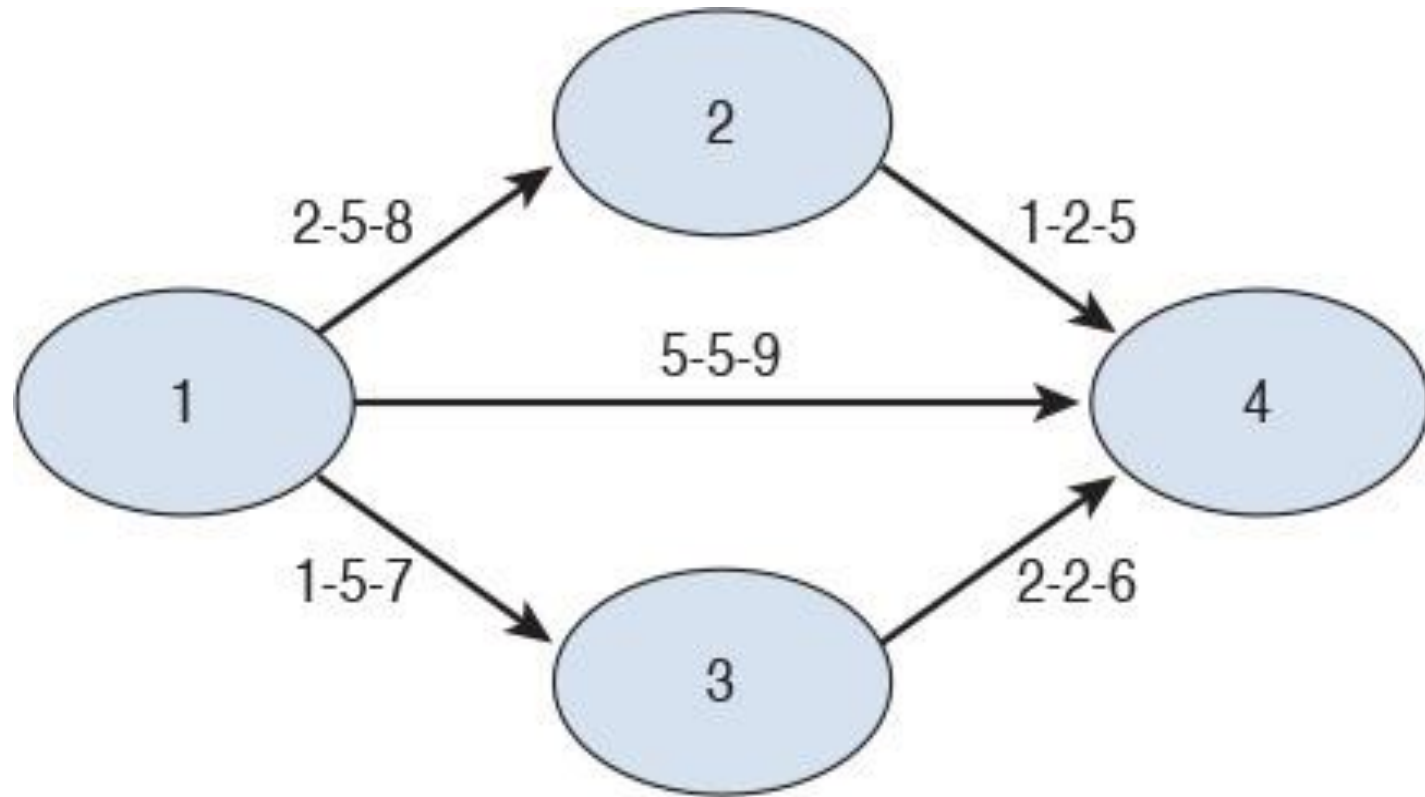
Scheduling

- With software size and cost determined, the project team can turn its attention to scheduling.
- Scheduling involves linking individual tasks. The relationship between these tasks is linked either by earliest start date or by latest expected finish date. Using a Gantt chart is one way to display these relationships.
- The Gantt chart was developed in the early 1900s as a tool to schedule activities and monitor progress.
- A Gantt chart shows the start and finish dates of each element of a project, as well as the relationship between the activities, in a calendar-like format.
- Gantt charts are one of the primary tools used to communicate project schedule information. This type of chart uses a baseline to illustrate what will happen if a task is finished early or late.

Scheduling - PERT

- Program Evaluation and Review Technique (PERT) is the preferred tool for estimating time when a degree of uncertainty exists. PERT uses a critical-path method, which applies a weighted average duration estimate.
- PERT uses a three-point time estimate to develop best, worst, and most likely time estimates. The PERT weighted average is calculated as follows:
- $$\text{PERT weighted average} = \frac{\text{Optimistic time} + 4 \times \text{Most likely time} + \text{Pessimistic time}}{6}$$
- A PERT chart is used to depict this information. Each chart begins with the first task and branches out to a connecting line that contains three estimates:

Sample PERT Chart



Scheduling - PERT

- Estimating the time and resources needed for application development is typically the most difficult part of the initial application-development activities.
- For example, suppose that a project team has been assigned to design an online user-registration system for the organization and that one task is to develop the Java input screen the user will use to enter personal data. It might be reasonable to estimate that this task will take five workdays to complete.
- With PERT, the best and worst completion time estimates would also be factored in.

Critical Paths

- In project management, anything can go wrong. Things can happen to affect the time, cost, or success of a project.
- That is why all project management techniques compute the critical path, or the sequence of tasks that must be completed on schedule for the project to be finished on time.
- We can expand on the house-building example we used earlier. Having a foundation is a task on the critical path; it must be completed before the walls and roof can be built. Exterior painting is not on the critical path and can be done at any time after the foundation, walls, and roof are completed.
- *Critical path methodology (CPM)* determines what activities are critical and what the dependencies are between the various tasks. CPM involves the following tasks:

Critical Paths

- ❖ Compiling a list of the tasks required to complete the project
- ❖ Determining the time that each task will take, from start to finish
- ❖ Examining the dependencies between the tasks
- Critical tasks have little flexibility in completion time. The critical path can be determined by examining all possible tasks and identifying the longest path.
- Even if completed on time, this path indicates the shortest amount of time in which the project can be completed.
- CPM offers real advantages over Gantt, in that it identifies this minimum time.

Critical Paths

- If the total project time needs to be reduced, one of the tasks on the critical path must be finished earlier.
- This is called **crashing**, and the project sponsor must be prepared to pay a premium for early completion—as a bonus or in overtime charges.
- The disadvantage to CPM is that the relationships of tasks are not as easily seen as with Gantt charts.

Timebox Management

- Timebox management is used in projects when time is the most critical aspect and software projects need to be delivered quickly.
- It is used to lock in specifications and prevent creep.
- For example, if given time, engineers might overengineer a system or decide to add more functionality or options.
- Although users might appreciate the added functionality, these items add time to the build phase and slow progress.
- Timeboxing counteracts this tendency by placing a very rigid time limit on the build of the system.
- When using timeboxing, the project time must never slip or be extended. If the project manager foresees time problems, he should consider taking corrective action, such as reducing the scope of the project or adding human or other resources.

Project Control and Execution

- Project control requires the collection, measurement, and dissemination of information and project performance.
- The bulk of the budget will be spent during the execution of a project.
- It is entirely possible that project changes might be needed. If so, the changes to the project must be recorded.
- Changes typically result in additional funds, human resources, or time. An auditor must be aware of any changes and must examine how they could affect any existing controls and the overall project.
- An auditor must also be concerned with end-user training. When new software products are released to users, the users must be trained on how the application works, what type of authentication is required, and how overrides or dual controls work.

Project Closing

- The last step in the process is to close the project. Projects are, after all, temporary endeavors. At the conclusion of a project, the project manager must transfer control to the appropriate individuals. The project closing includes the following tasks:

- ❖ Administrative closure
- ❖ Release of final product or service
- ❖ Update of organizational assets

At the close of a project, three items to consider are:

1. control implementation,
2. benefits realization,
3. and performance measurement.

Project Closing

- The last step in the process is to close the project. Projects are, after all, temporary endeavors. At the conclusion of a project, the project manager must transfer control to the appropriate individuals. The project closing includes the following tasks:

- ❖ Administrative closure
- ❖ Release of final product or service
- ❖ Update of organizational assets

At the close of a project, three items to consider are:

1. control implementation,
2. benefits realization,
3. performance measurement.

Project Closing

- Surveys or post-project reviews might be performed.
- This is a chance to survey the project team and end users to gauge their satisfaction with the project and examine how things could have been done differently or what changes should be implemented next time.
- A post-mortem review is similar but is usually held after the project has been in use for some time.

BUSINESS APPLICATION DEVELOPMENT

- Business application development is largely a product of the systems development life cycle (SDLC). New applications are typically created when new opportunities are discovered and when companies want to take advantage of new technology or use technology to solve an existing problem. Organizations use a structured approach for three reasons:
 - ❖ To minimize risk
 - ❖ To maximize return
 - ❖ To establish controls to increase the likelihood that the software will meet user needs

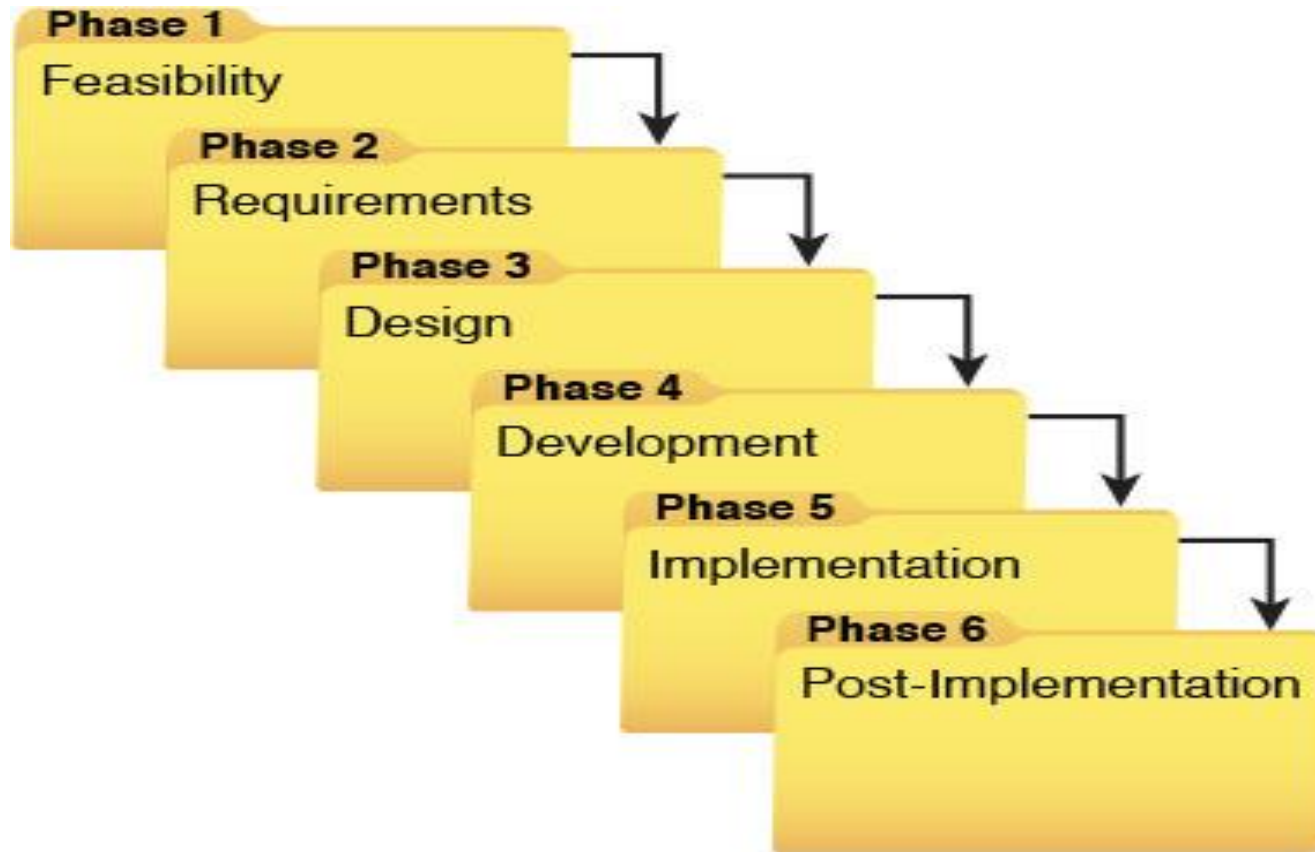
BUSINESS APPLICATION DEVELOPMENT

- As an auditor, you are not expected to be an expert programmer, instead, an auditor must know how to manage the development process so that adequate controls are developed and implemented.
- An auditor must be able to review information at each step of the process and provide input on the adequacy of controls being designed.
- Auditors are also responsible for reporting independently to management on the status of a project and the implementation of controls.
- Auditors might also become more deeply involved in a process based on their individual skills and abilities.

Systems-Development Methodology

- The SDLC is designed to produce high-quality software in a structured way that minimizes risk.
- The SDLC waterfall model is closely allied with the project management life cycle model.
- The name of this model comes from the fact that progress flows from the top to the bottom, moving through each phase.
- W.W. Royce originally described the model as having seven phases. Some variations show it as having five or six phases.
- ISACA uses a modified model that has five primary phases plus a post-implementation phase.

The Water Fall Method



Systems-Development Methodology

Waterfall Phase	Description
Initiation	Benefits and needs are determined at this phase of the SDLC.
Development / Acquisition	At this phase, the purpose of the project must be defined. The systems must be designed, developed, constructed, or purchased.
Implementation	The system is installed and end users are trained. At this point, the auditor must verify that all required controls that are in the design function as described.
Operation / Maintenance	The system or program perform the work for which it was designed. Patching and maintenance are important at this point.
Disposal	At this phase the system or program is retired and data is destroyed or archived in an approved method.

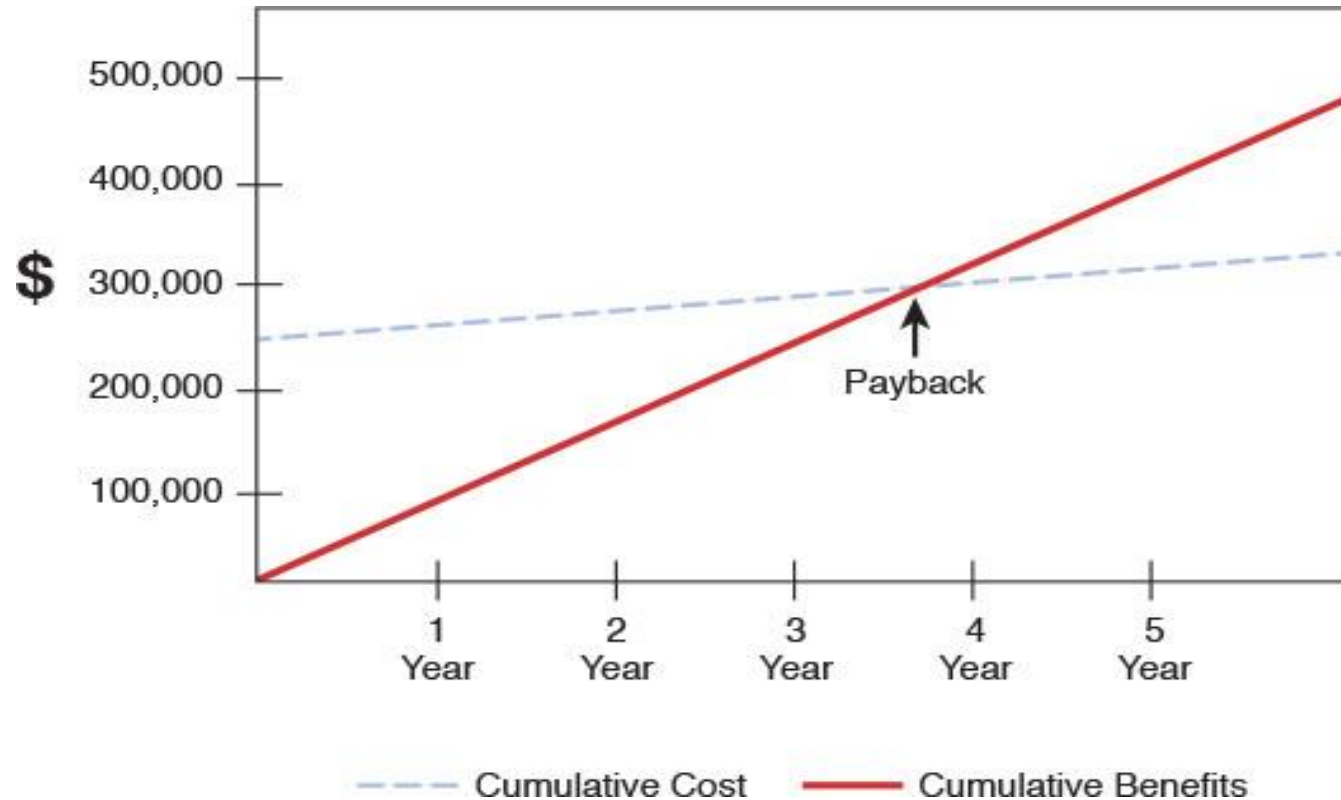
Systems-Development Methodology

- The National Institute of Standards and Technology (NIST) defines the SDLC in NIST SP 800-34 as “the scope of activities associated with a system, encompassing the system’s initiation, development and acquisition, implementation, operation and maintenance, and ultimately its disposal that instigates another system initiation.”
- Therefore, **the goal of the SDLC is to control the development process and add security checks at each phase.**
- Failure to adopt a structured development model increases risk and the likelihood that the final product may not meet the customer’s needs. A good resource for further review is http://csrc.nist.gov/publications/nistbul/april2009_system-development-life-cycle.pdf.

Phase 1: Initiation phase

- In the initiation phase of the NIST SDLC, the feasibility of the project is considered and a requirements analysis is performed.
- The cost of the project must be discussed, as well as the potential benefits that it will bring to the system's users.
- A payback analysis must be performed to determine how long the project will take to pay for itself.
- In other words, the payback analysis determines how much time will lapse before accrued benefits will overtake accrued and continuing costs.
- If it is determined that the project will move forward, the team should develop a preliminary timeline.
- During the feasibility phase, everyone gets a chance to meet and understand the goals of the project.

Phase 1: Initiation phase



Phase 1: Initiation phase

- In the initiation phase of the NIST SDLC, the feasibility of the project is considered and a requirements analysis is performed.
- The cost of the project must be discussed, as well as the potential benefits that it will bring to the system's users.
- A payback analysis must be performed to determine how long the project will take to pay for itself.
- In other words, the payback analysis determines how much time will lapse before accrued benefits will overtake accrued and continuing costs.
- If it is determined that the project will move forward, the team should develop a preliminary timeline.
- During the feasibility phase, everyone gets a chance to meet and understand the goals of the project.

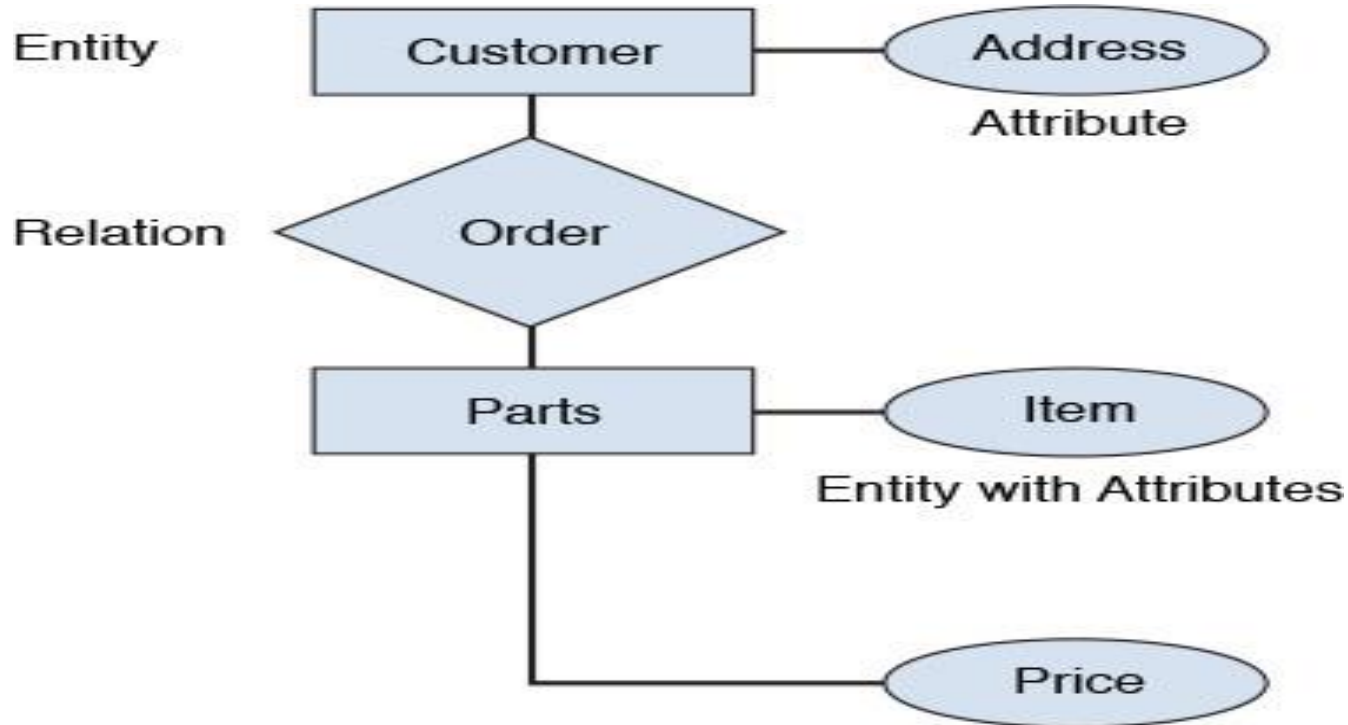
Phase 1: Initiation phase

- This is also the point at which users should be involved because they should have input into how the applications are designed.
- An *entity relationship diagram (ERD)* is often used to help map the requirements and define the relationship between elements.
- The basic components of an ERD are an entity and a relationship.
- An entity is very much like a database, in that it is a grouping of like data elements. An entity has specific attributes, which are called the entity's *primary key*.
- Entities are drawn as a rectangular box with an identifying name.
- Relationships describe how entities are related to each other and are defined as a diamond..

Phase 1: Initiation phase

- ERDs can be used to help define a data dictionary.
- When a data dictionary is designed, the database schema can be developed.
- The database schema defines the database tables and fields, as well as the relationship between them.
- The completed ERD will be used in the design phase as the blueprint for the design

Entity Relationship Diagram



Phase 1: Initiation phase

- During the requirements phase, auditors must verify the requirements and determine whether adequate security controls are being defined. These controls should include the following mechanisms:
- **Preventive:** Preventive controls can include user authentication and data encryption.
- **Detective:** Detective controls can include embedded audit modules and audit trails.
- **Corrective:** Corrective controls can include fault-tolerance controls and data-integrity mechanisms.

Phase 1: Initiation phase

- During the requirements phase, auditors must verify the requirements and determine whether adequate security controls are being defined. These controls should include the following mechanisms:
- **Preventive:** Preventive controls can include user authentication and data encryption.
- **Detective:** Detective controls can include embedded audit modules and audit trails.
- **Corrective:** Corrective controls can include fault-tolerance controls and data-integrity mechanisms.

Phase 1: Initiation phase

Before moving forward, a decision may be made to buy instead of build. With the option to buy, the project team should develop a request for proposal (RFP) to solicit bids from vendors.

Vendor responses should be closely examined to find the vendor that best meets the project team's requirements. The team should ask questions such as these:

- Does the vendor have a software product that will work as is?
- Will the vendor have to modify the software product to meet our needs?
- Will the vendor have to create a new, nonexistent software product for us?

The reputation of the vendor is also important. Is the vendor reliable, and do references demonstrate past commitment to service? When a vendor is chosen, the last step is to negotiate and sign a contract.

Auditors will want to make sure that a sufficient level of security will be designed into the product and that risks are minimized.

Phase 1: Initiation phase

Before moving forward, a decision may be made to buy instead of build. With the option to buy, the project team should develop a request for proposal (RFP) to solicit bids from vendors.

Vendor responses should be closely examined to find the vendor that best meets the project team's requirements. The team should ask questions such as these:

- Does the vendor have a software product that will work as is?
- Will the vendor have to modify the software product to meet our needs?
- Will the vendor have to create a new, nonexistent software product for us?

The reputation of the vendor is also important. Is the vendor reliable, and do references demonstrate past commitment to service? When a vendor is chosen, the last step is to negotiate and sign a contract.

Auditors will want to make sure that a sufficient level of security will be designed into the product and that risks are minimized.

Phase 2: Development

- During the development phase, users might not be involved, but the auditor is still working in an advisory role.
- The auditor must again check that security controls are still in the design and test documents. Test plans should detail how security controls will be tested.
- Tests should be performed to validate specific program units, subsystems, interfaces, and backup/recovery. Change-control procedures should be developed to prevent uncontrolled changes.

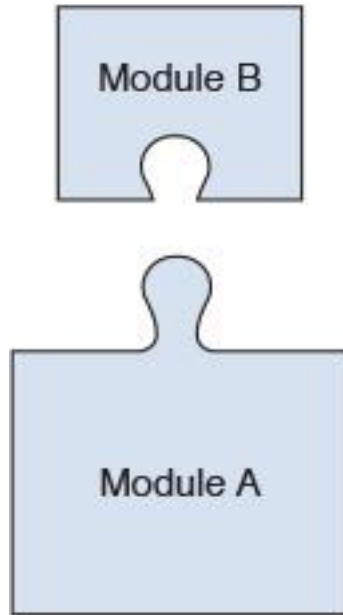
Phase 2: Development

- *Scope creep* is the addition of products, features, or items to an original design so that more and more items are added on. This is sometimes referred to as the “kitchen sink syndrome.”
- Scope creep is most likely to occur in the design phase. Little changes might not appear to have a big cost impact on a project, but they will have a cumulative effect and increase the length and cost of the project.
- There are ways to decrease design and development time. Reverse engineering is one such technique. It converts executable code into a human-readable format and can be performed with tools such as IDA Pro.
- This is a somewhat controversial subject because, although reverse engineering has legitimate uses, a company could use it to disassemble another company’s program. Most software licenses make this illegal. Reverse engineering is also sometimes used to bypass access-restriction mechanisms.

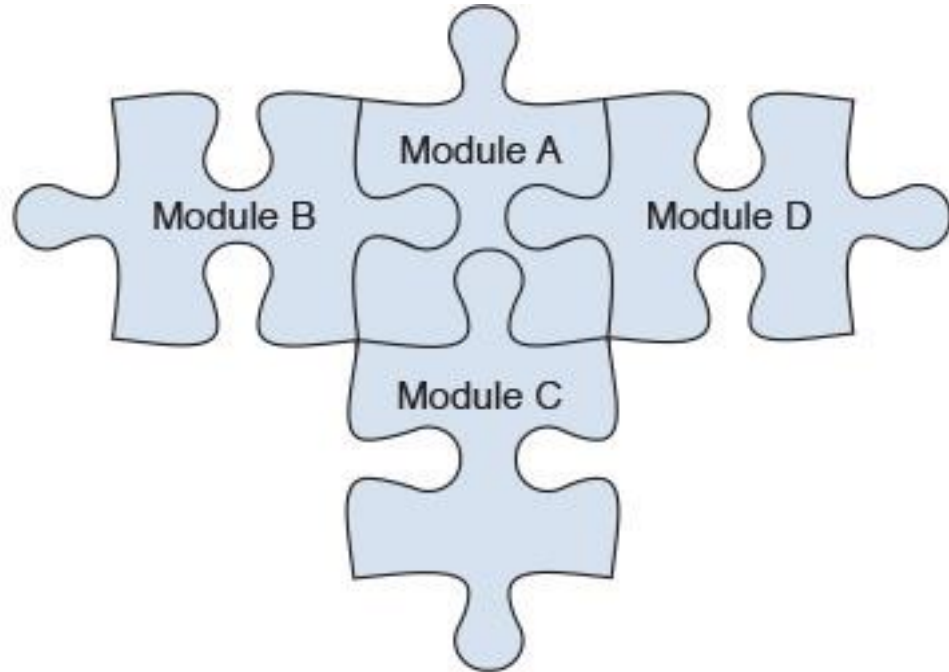
Phase 2: Development

- Programmers might use online programming facilities so that many programmers can access the code directly from their workstation.
- Although this typically increases productivity, it also increases risk because someone might gain unauthorized access to the program library.
- Programmers should strive to develop modules that have high cohesion and low coupling.
- **Cohesion** addresses the fact that a module is focused on a single task.
- **Coupling** is the degree of interconnection between modules.
- Low coupling means that a change to one module should not affect another.

High and Low Coupling



Low Coupling



High Coupling

Phase 2: Development

- Programmers might use online programming facilities so that many programmers can access the code directly from their workstation.
- Although this typically increases productivity, it also increases risk because someone might gain unauthorized access to the program library.
- Programmers should strive to develop modules that have high cohesion and low coupling.
- **Cohesion** addresses the fact that a module is focused on a single task.
- **Coupling** is the degree of interconnection between modules.
- Low coupling means that a change to one module should not affect another.

Phase 2: Development

- **Programmers should strive to develop modules that have high cohesion and low coupling.**
- During development, auditors must verify that input and output controls, audit mechanisms, file-protection schemes, and a software version control system are being used.
- Examples of input controls include dollar counts, transaction counts, error detection, and correction.
- Examples of output controls include validity checking and control authorization. One way to manage and track changes to source code is by using a source control software package.
- A source control software package is used to secure source code and for version control. Examples include Team Foundation Server (TFS) and Mercurial.
- Testing these controls and the functionality of the program is an important part of this phase. Testing can be done by using one of the following testing methods:

Phase 2: Development

- **Top down:** Top-down testing starts with a depth or breadth approach. Its advantage is that it gets programmers working with the program so that interface problems can be found sooner. It also allows for early testing of major functions.
- **Bottom up:** Bottom-up testing works up from the code to modules, programs, and systems. The advantage of bottom-up testing is that it can be started as soon as modules are complete; work does not have to wait until the entire system is finished. This approach also allows errors in modules to be discovered early. Most application testing follows the bottom-up approach.
- Regardless of the chosen approach, test classifications are divided into the following categories:

Phase 2: Development

- **Unit testing:** Examines an individual program or module.
- **Interface testing:** Examines hardware or software to evaluate how well data can be passed from one entity to another.
- **System testing:** Involves a series of tests that can include recovery testing, security testing, stress testing, volume testing, and performance testing. Although unit and interface testing focus on individual objects, the objective of system testing is to assess how well the system functions as a whole.
- **Final acceptance testing:** When the project staff is satisfied with all other tests, *final acceptance testing*, or user acceptance testing, must be performed. This occurs before the application is implemented into a production environment.
- One big reason all this testing is performed is to verify the completeness, accuracy, validity, and authorization of transactions and data.

Testing Types

Pilot test	Used as an evaluation to verify functionality of the application.
White-box test	A type of test that verifies inner program logic. This testing is typically cost-prohibitive on a large application or system.
Black-box test	Integrity-based testing that looks at inputs and outputs. Black-box testing can be used to ensure the integrity of system interfaces.
Function test	A type of test that validates a program against a checklist of requirements.
Regression test	A type of test that verifies that changes in one part of the application did not affect any other parts in the same application or interfaces.
Parallel test	Parallel tests involve the use of two systems or applications at the same time. The purpose of this testing is to verify a new or changed system or application by feeding data into both and comparing the results.
Sociability test	A type of test which verifies that the system can operate in its targeted environment.

Phase 2: Development

- Another name for beta testing is user acceptance testing (UAT). The idea of this type of testing is to put the software through a real-world test.
- One important item that must be verified during development is exception handling. When programs don't work as required or something must be done outside the normal process, close examination is needed.
- Exception handling is the act of replying to the occurrence of exceptions or anomalies or exceptional conditions that require special processing. This can change the normal flow of program execution.
- Before coding can begin, programmers must decide what programming language they will use. To some extent, the organization will decide this. For example, if the company has used C++ for engineering projects for the past 5 years, it might make sense to do so for the current project. Programming has evolved through five generations of programming languages to this point:

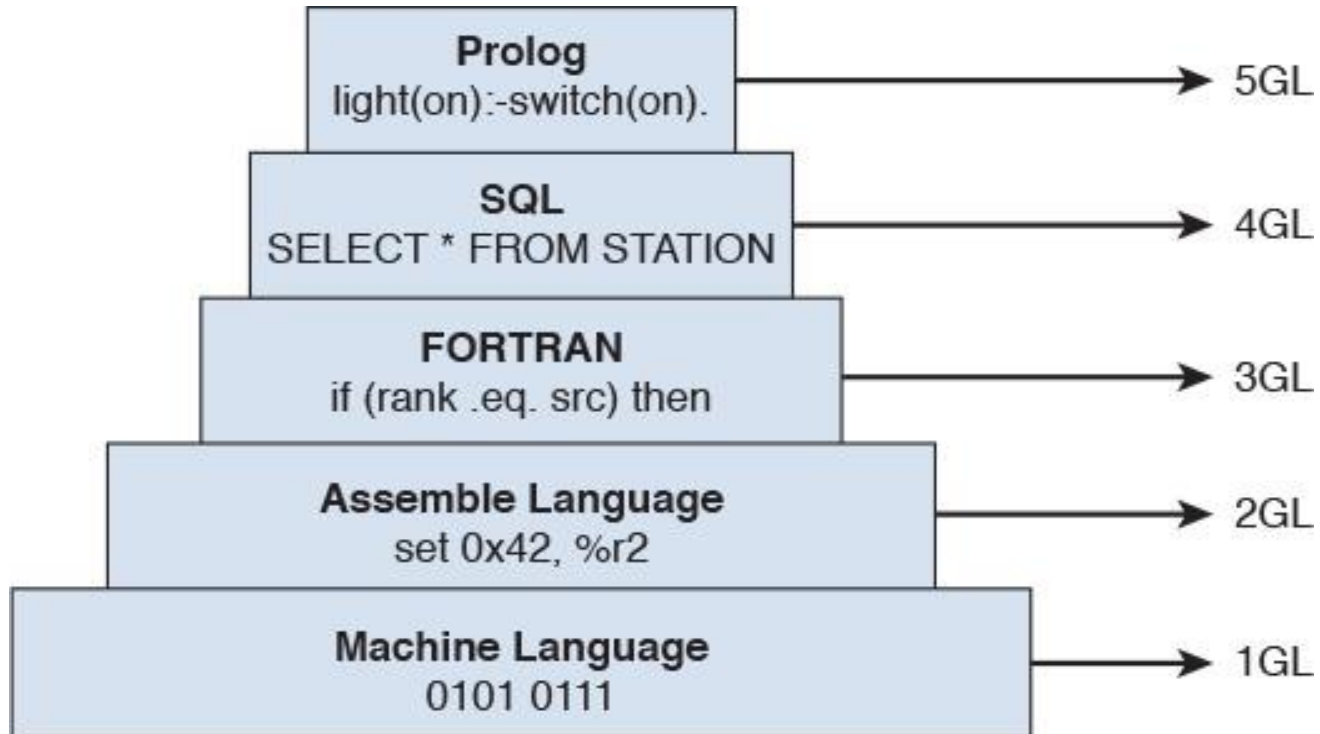
Phase 2: Development

- **First generation (1GL):** Machine language
- **Second generation (2GL):** Assembly language
- **Third generation (3GL):** High-level language
- **Fourth generation (4GL):** Very high-level language
- **Fifth generation (5GL):** Natural language
- In 6GPLs, the operating system uses a natural language processor to analyze a command and determine its meaning. After determination of meaning, the natural language processor invokes an interlanguage decompiler to rewrite the command in a common high-level language. Once the ILD decompiles the command, a low decompiler rewrites the command into assembly language or machine code. Then, the central processing unit executes the command.

Phase 2: Development

- **First generation (1GL):** Machine language
- **Second generation (2GL):** Assembly language
- **Third generation (3GL):** High-level language
- **Fourth generation (4GL):** Very high-level language
- **Fifth generation (5GL):** Natural language
- In 6GPLs, the operating system uses a natural language processor to analyze a command and determine its meaning. After determination of meaning, the natural language processor invokes an interlanguage decompiler to rewrite the command in a common high-level language. Once the ILD decompiles the command, a low decompiler rewrites the command into assembly language or machine code. Then, the central processing unit executes the command.

Phase 2: Development



Phase 2: Development

- Organizations might have many individuals who are able to write code, but this does not mean they are *authorized* to do so.
- End-user computing (EUC) refers to systems in which nonprogrammers can create working applications.
- Such applications and the citizen programmers who create them can have a detrimental effect on security if not properly managed.
- No single user should ever have complete control over the development of an application program.
- There should always be checks and balances to prevent fraud and maintain control.

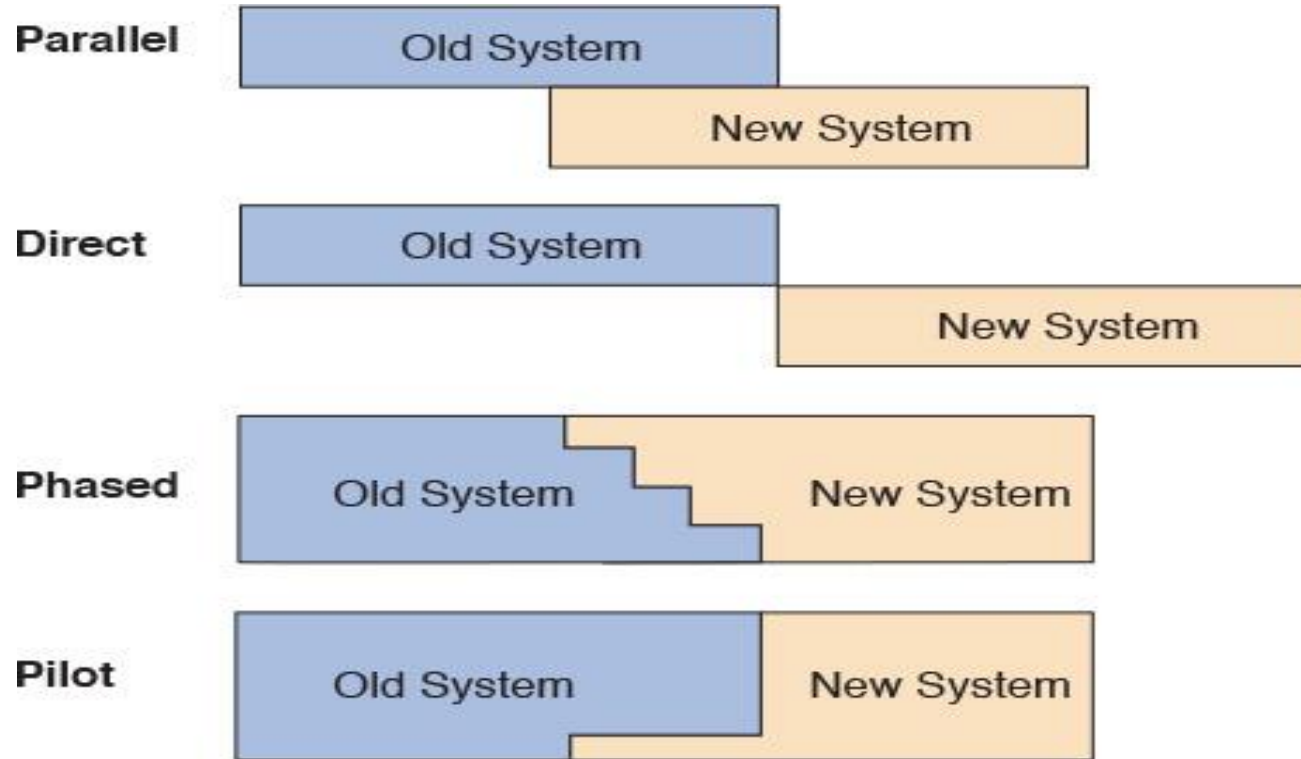
Phase 3: Implementation

- In the implementation phase, the application is prepared for release into its intended environment. Final user acceptance is performed, as are certification and accreditation. This is typically the final step in accepting the application and agreeing that it is ready for use.
- *Certification* is the technical review of the system or application. Certification testing might include an audit of security controls, a risk assessment, or a security evaluation.
- Accreditation is management's formal acceptance of a system or application. Typically, the results of the certification testing are compiled into a report, and management's acceptance of the report is used for accreditation.

Phase 3: Implementation

- Management might request additional testing, ask questions about the certification report, or accept the results as is. Once the results are accepted, a formal acceptance statement is usually issued.
- Data conversion tools may be used to migrate data from one system to another. Data conversion is simply conversion of computer data from one format to another.
- Note that final user acceptance testing is performed during the implementation phase.
- The rollout of the application or system migration might be all at once or phased in over time.
- Changeover techniques, include the following:

Phase 3: Changeover Techniques



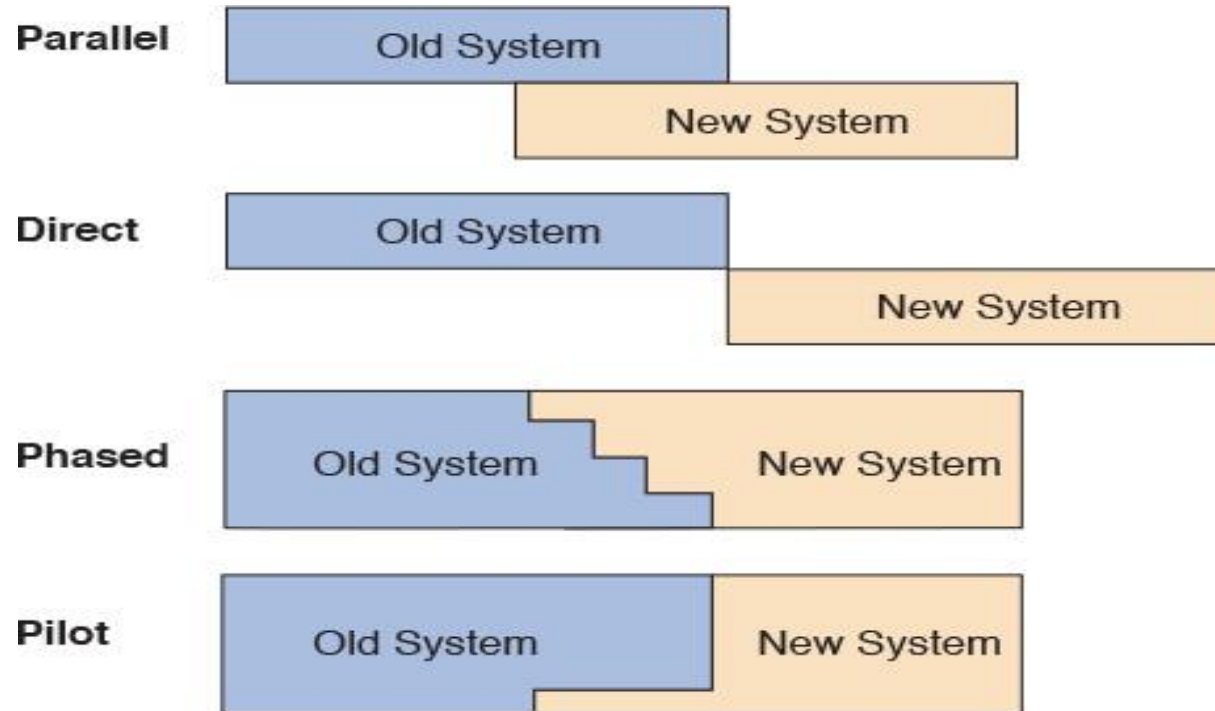
Phase 3: Implementation

- Management might request additional testing, ask questions about the certification report, or accept the results as is. Once the results are accepted, a formal acceptance statement is usually issued.
- Data conversion tools may be used to migrate data from one system to another. Data conversion is simply conversion of computer data from one format to another.
- Note that final user acceptance testing is performed during the implementation phase.
- The rollout of the application or system migration might be all at once or phased in over time.
- Changeover techniques, include the following:

Phase 3: Implementation

- **Parallel operation:** Both the old and new systems are run at the same time. Results between the two systems can be compared. Fine-tuning can also be performed on the new system as needed. As confidence in the new system improves, the old system can be shut down. The primary disadvantage of this method is that both systems must be maintained for a period of time.
- **Direct changeover:** This method establishes a date at which users are forced to change over. The advantage of a direct, or hard, changeover is that it forces all users to change at once. However, this introduces a level of risk into the environment because things can go wrong.
- **Phased changeover:** If the system is large, a phased changeover might be possible. With this method, systems are upgraded one piece at a time.
- **Pilot changeover:** This method requires that an entire new system be used at one location.

Phase 3: Changeover Techniques



Phase 4: Operation and Maintenance

- In the operation and maintenance phase, it is time to roll out the application to the users.
- Some support functions need to be established. Items such as maintenance, support, and technical response must be addressed.
- Data conversion might also need to be considered.
- If an existing system is being replaced, data from the system might need to be migrated to the new one. Computer-aided software engineering (CASE) is used for program and data conversions.

Phase 4: Operation and Maintenance

- Some of the users that have been involved throughout the process and can help in the training process. The training can include classroom training, online training, practice sessions, and user manuals.
- Some may think that once an application is deployed that the work of an auditor is done, but that is far from the case.
- A large part of information systems maintenance involves keeping the code up to date.
- This means someone must be checking for vulnerabilities and patching known concerns.
- One source of information on this topic is the common vulnerabilities and exposures (CVE) database, a dictionary of identifiers and details of publicly known information security vulnerabilities. It can be viewed at <https://cve.mitre.org>.

Phase 4: Operation and Maintenance

- Vulnerability assessment can be traced back to applications such as SATAN and SAINT.
- An auditor should review how vulnerability assessment and remediation are being handled.
- For example, Nessus might be used on a weekly basis, with vulnerabilities being labeled high, medium, or low. Items defined as high might require action within 24 hours of discovery, whereas items labeled low might not require action for up to 30 days.
- Closely related to vulnerability assessment is patching and updates. Once vulnerabilities are found, patch management helps get any vulnerabilities addressed in an expedient manner to reduce overall risk of a system compromise.

Phase 4: Operation and Maintenance

- Patch management is key to keeping applications and operating systems secure. The organization should have a well-developed patch management testing and deployment system in place. The most recent security patches should be tested and then installed on host systems as soon as possible. The only exception is when an immediate application would interfere with business requirements.
- Before a patch can be tested and deployed, it must first be verified. Typical forms of verification include digital signatures, digital certificates, and some form of checksum and/or integrity verification. This is a critical step that must be performed before testing and deployment to make sure the patch has not been maliciously or accidentally altered. Once testing is complete, deployment can begin.
- :

Phase 4: Operation and Maintenance

- Another post-implementation activity is a review of the overall success of the project. Actual costs versus projected costs should be reviewed to see how well cost estimating was done during the feasibility phase.
- ROI and payback analysis should be reviewed.
- A gap analysis can determine whether there is a gap between requirements that were and were not met.
- An independent group might conduct performance measurement, such as an audit. If this occurs, it should not be done by auditors who were involved in the SDLC process.
- Overall, post-implementation should answer the following questions:

Phase 4: Operation and Maintenance

- Is the system adequate?
- What is the true ROI?
- Were the chosen standards followed?
- Were good project management techniques used?

- Note

The release management step involves managing an application, maintaining traceability, and ensuring the version once the application has been released.

Phase 5: Disposal

- Applications and systems don't last forever.
- At some point, systems must be decommissioned and disposed of. This step of the process is reached when an application or a system is no longer needed.
- Those involved in the disposal process must consider the disposition of the application. Should it be destroyed or archived, or does the information need to be migrated into a new system?
- Disk sanitization and destruction are also important to ensure confidentiality.
- This important step is sometimes overlooked.

Phase 5: Disposal

Media	Wipe Standard	Description
Rewritable magnetic media (hard drive, flash drive, and so on)	Drive wiping or degaussing	DOD 5220.22-M seven-pass drive wipe or electric degaussing
Optical media (CD-RW, DVD-RW, DVD+RW, CD-R, DVD-R, and so on)	Physical destruction	Physical destruction of the media by shredding or breaking

Tools and Methods for Software Development

- Globalization has increased the pace of change and reduced the amount of time that organizations have to respond to changes. New systems must be brought online quickly. The SDLC is not the only development methodology used today.
- As an auditor, you must be knowledgeable about other development methods and have a basic understanding of their operations. Some popular models include the following:
- **Incremental development:** This method involves developing systems in stages so that development is performed one step at a time. A minimal working system might be deployed while subsequent releases build on functionality or scope.

Tools and Methods for Software Development

- **Spiral development:** The spiral model was developed based on the experience of the waterfall model and the concept that software development is evolutionary. The spiral model begins by creating a series of prototypes to develop a solution. As the project continues, it spirals out, becoming more detailed. Each step passes through planning, requirements, risks, and development phases.
- **Prototyping:** The prototyping model reduces the time required to deploy applications. Prototyping involves using high-level code to quickly turn design requirements into application screens and reports that users can review. User feedback can be used to fine-tune the application and improve it. Top-down testing works well with prototyping. Although prototyping clarifies user requirements, **it can result in overly optimistic project timelines. Also, when change happens quickly, it might not be properly documented, which is of concern for an auditor.**
- Note - The advantage of prototyping is that it can provide great savings in development time and costs.

Tools and Methods for Software Development

- **Rapid application development (RAD)**: RAD uses an evolving prototype and requires heavy user involvement. According to ISACA, RAD requires well-trained development teams that use integrated power tools for modeling and prototyping. With the RAD model, strict limits are placed on development time. RAD has four unique stages: concept, functional design, development, and deployment.
- These models share a common element: They each have a predictive life cycle. This means that when a project is laid out, costs are calculated, and a schedule is defined.
- Another category of application development is called *agile software development*. With this development model, teams of programmers and business experts work closely together. Project requirements are developed using an *iterative* approach because the project is both mission driven and component based. The project manager is much more of a facilitator in these situations. Popular agile development models include the following:

Tools and Methods for Software Development

- **Extreme programming (XP):** The XP development model requires that teams include business managers, programmers, and end users. These teams are responsible for developing usable applications in short periods of time. Issues with XP are that teams are responsible not only for coding but also for writing the tests used to verify the code. Lack of documentation is also a concern. XP does not scale well for large projects.
- **Scrum:** Scrum is an iterative development method in which repetitions are referred to as *sprints* and typically last 30 days. Scrum is typically used with object-oriented technology and requires strong leadership and a team meeting each day for a short time. The idea is for the project manager to give over more planning and directing tasks to the team. The project manager's main task is to work on removing any obstacles from the team's path.
- **Note - Reengineering** involves converting an existing business process. Reengineering means updating software by reusing as many of the components as possible instead of designing an entirely new system.

INFORMATION SYSTEMS MAINTENANCE

- When a system moves into production, the work is not yet done.
- Changes need to be made and must be done in a controlled manner.
- The integrity of the application and source code must be ensured.
- Most organizations use a change-control board that includes a senior manager as the chairperson and individuals from various organizational groups.
- The change-control board is responsible for developing a change-control process and also for approving changes.
- Although the types of changes vary, change control follows a predictable process:

INFORMATION SYSTEMS MAINTENANCE

1. Request the change.
2. Approve the change request.
3. Document the change request.
4. Test the proposed change.
5. Present the results to the change-control board.
6. Implement the change, if approved.
7. Document the new configuration.

INFORMATION SYSTEMS MAINTENANCE

- Documentation is key to a good change-control process. All system documents should be updated to indicate any changes that have been made to the system or environment.
- The system maintenance staff or department responsible for requesting the change should keep a copy of the change approval.
- An auditor should ensure that backup copies of critical documents are created.
- These documents should be kept offsite in case of a disaster or other situation. The auditor should also watch for the possibility of unauthorized changes due to poor oversight or lack of proper security controls. Items to look for include the following:

INFORMATION SYSTEMS MAINTENANCE

- Changes are implemented directly by the software vendor, without internal control.
- Programmers place code in an application that has not been tested or validated.
- The changed source code has not been reviewed by the proper employee.
- No formal change process is in place.
- The change review board has not authorized the change.
- The programmer has access to both the object code and the production library.

INFORMATION SYSTEMS MAINTENANCE

- Even if an auditor takes all these measures, a situation may still arise in which a change must bypass the change-control process.
- Emergency changes might have to be made because of situations that endanger production or halt a critical process.
- In such situations, it is important to maintain the integrity of the process.
- These changes should be followed up by procedures to ensure that standard controls are applied retroactively.
- If programmers are given special access or an increased level of control, the accounts and mechanisms they use should be closely monitored.

OUTSOURCING AND ALTERNATIVE SYSTEM DEVELOPMENT

Third-party outsourcing is the practice of handing over responsibility for an organization's information systems development and operations to an outside firm.

There are many reasons a firm might decide to outsource. High on the list are typically cost and problems with internal IS performance.

One common approach to outsourcing involves these steps:

- 1. Identify, select, and plan a system.
- 2. Conduct system analysis.
- 3. Develop a request for proposal.
- 4. Select a vendor.

OUTSOURCING AND ALTERNATIVE SYSTEM DEVELOPMENT

- Not all outsourcing relationships are the same.
- It is important to ensure that an organization's service levels and requisite controls are met.
- When reaching outsourcing agreements, certain stipulations may be applied.
- For example, does your company have employees, contractors, or business partners sign NDAs?
- Doing so is one way to help provide security for sensitive information and proprietary data.
- As an auditor, you should have a basic understanding of the following documents and how they are used with outsourcing partners:

OUTSOURCING AND ALTERNATIVE SYSTEM DEVELOPMENT

- **Interconnection security agreement (ISA):** An ISA is a security document that details the requirements for establishing, operating, and maintaining an interconnection between systems or networks. An ISA typically details how specific systems and networks are connected and contains a drawing of the network topology.
- **Memorandum of understanding (MOU):** An MOU typically documents conditions and applied terms for outsourcing partner organizations that must share data and information resources. To be binding, the MOU must be signed by a representative from each organization that has the legal authority to sign. Such documents are typically secured, as they are considered confidential.

OUTSOURCING AND ALTERNATIVE SYSTEM DEVELOPMENT

- **Operating level agreement (OLA):** An OLA works in conjunction with SLAs by supporting the SLA process. An OLA defines the responsibilities of each partner's internal support group. So, whereas an SLA may promise no more than five minutes of downtime, an OLA defines what group and resources are used to meet a specified goal.
- **Uptime agreement (UA):** A UA is one of the best-known types of SLAs; it details the agreed amount of uptime. For example, UAs can be used for network services such as a WAN link or equipment like servers. A UA may, for example, specify cloud server uptime of 99.9999999 percent (nine nines).

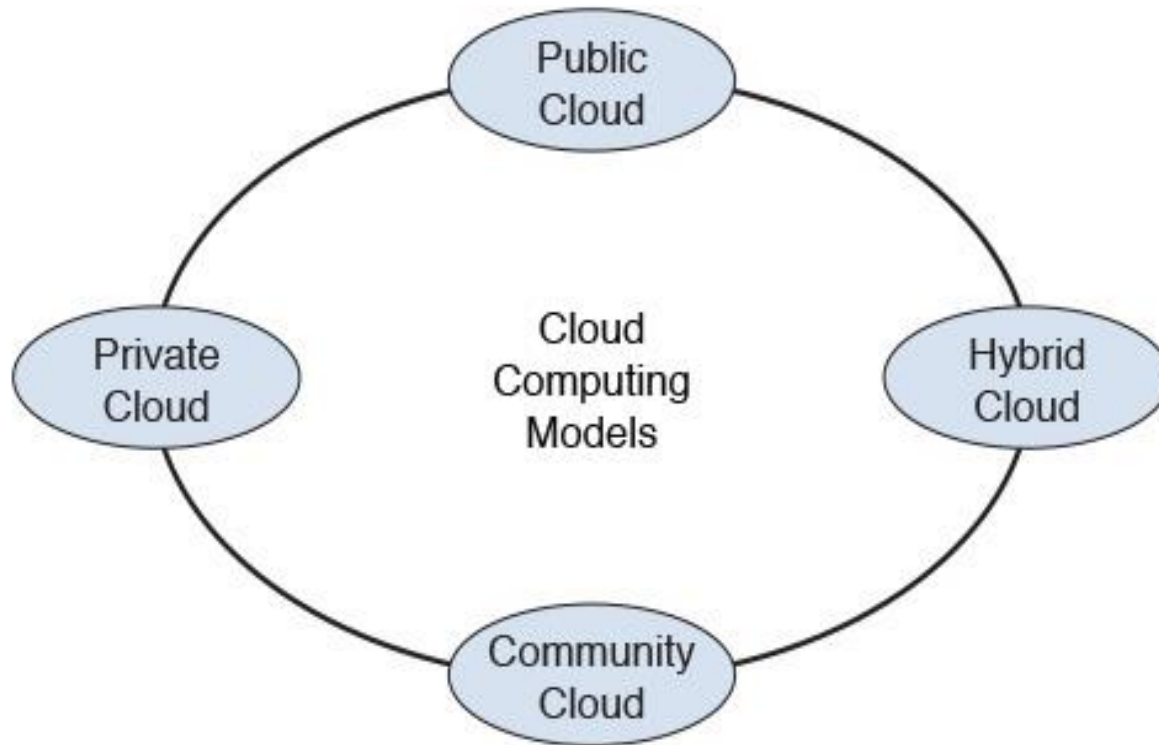
OUTSOURCING AND ALTERNATIVE SYSTEM DEVELOPMENT

- **Business Partnership Security Agreement (BPA):** A BPA is a written agreement created by lawyers with the input from the partners that contains standard clauses related to security and cooperation. A BPA is an example of a legally binding document that is designed to provide safeguards and compel certain actions among business partners in relation to specific security-related activities.
- Note - When dealing with business partners, auditors should review Statements on Standards for Attestation Engagements No. 16. SSAE 16 verifies controls and processes and requires a written assertion regarding the design and operating effectiveness of the controls being reviewed.

Cloud Computing

- Cloud computing is a type of outsourcing that involves turning over the operation of a service to an outside entity. Depending on the cloud model being used, a varying amount of control is given to the cloud service provider. There are several cloud computing models:
- **Public cloud:** This cloud model is based on the concept that the service provider makes resources, such as applications and storage, available to the general public. An example is Dropbox.
- **Private cloud:** This cloud model is based on the concept that the cloud is owned and operated by a private entity.
- **Community cloud:** This cloud model can be used by several entities.
- **Hybrid cloud:** This cloud model can be a combination of any of the other cloud models.

Cloud Computing - Cloud Models



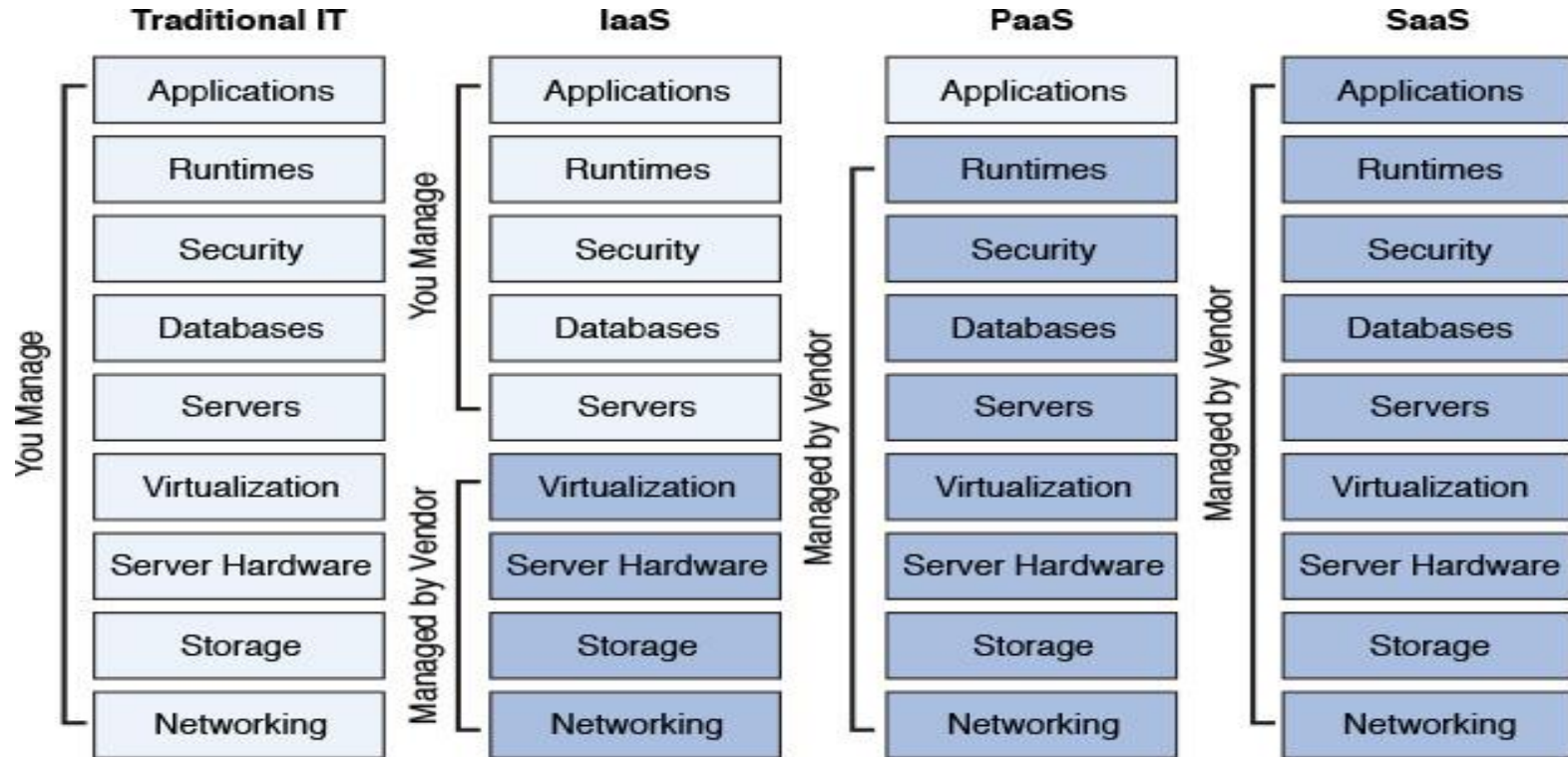
Cloud Computing

- When cloud computing is used, the organization gives up a portion or all of this activity to the cloud provider. Basic cloud services include:
- Infrastructure as a Service (IaaS),
- Platform as a Service (PaaS), and
- Software as a Service (SaaS),

Cloud Computing - Cloud Services

Service	Description
Infrastructure as a Service	A form of cloud computing services that provides virtualized computing resources over the Internet.
Platform as a Service	A form of cloud computing services in which a platform allows customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure typically associated with it.
Software as a Service	A form of cloud computing services in which a third-party provider hosts applications and makes them available to customers over the Internet.

Cloud Computing - Overview



Cloud Computing

- There are many reasons organizations are increasingly moving to cloud-based services, including cost, portability, and a desire to focus on core competencies.
- Regardless of the reason, making the transition to the cloud requires careful contract negotiation.
- An auditor should understand that, historically, the network group has had complete control; however, moving to the cloud requires the organization to relinquish some control and oversight to the cloud provider.
- This transition requires networking professionals to move beyond their traditional packet-processing mindset in order to really grasp cloud computing.

Cloud Computing

- Auditors should know that companies operating in the United States, Canada, or the European Union have many regulatory requirements by which they must abide, including ISO/IEC 27001, EU-U.S. Privacy Shield Framework, IT Infrastructure Library, and COBIT.
- It is important to have a framework for the contract that both parties can agree on, such as ISO 27001.
- The cloud provider must be able to meet these requirements and be willing to undergo certification, accreditation, and review.
- The cloud provider should also agree in writing to comply with an auditing standard, such as SSAE 16.

Cloud Computing

- With cloud computing, proving to auditors and assessors that compliance is being met is becoming more challenging and more difficult to demonstrate.
- Of the many regulations dealing with IT, few were written with cloud computing in mind.
- Auditors and assessors may not be familiar with cloud computing or with a given cloud service.

Cloud Threats

- Organizations must consider a variety of potential threats before they move to a cloud model. To that end, a cloud provider should agree in writing to provide the level of security an organization requires.
- Access control is an important issue. How will cloud authentication be managed? Insider attacks are an ongoing threat. Anyone who has been approved to access the cloud can be a potential problem. Here's an example: Say that an employee quits or is terminated, and then you find out he or she was the only person who had the password. Or, perhaps the employee was the one responsible for ensuring that the cloud provider gets paid. You need to know who has access, how he or she was screened, and how access is terminated.

Cloud Threats

- Training is another issue of concern. Knowing how your provider trains its employees is an important item to review. Most attacks are both technical and social. The steps a provider takes to address social-engineering attacks stemming from email, malicious links, phone, and other methods should be included in its training and awareness program.
- Auditors should also inquire about the standard being used to classify data and whether the provider supports it. Tokenization is a growing alternative to encryption and can help ensure compliance with regulatory requirements such as those under the Health Insurance Portability and Accountability Act (HIPAA), Payment Card Industry Data Security Standard (PCI-DSS), the Gramm–Leach–Bliley Act, and the EU data protection regulations.

Cloud Threats

- Encryption should be reviewed. Will the original data leave the organization, or will it stay internal to satisfy compliance requirements? Will encryption be used while the data is at rest, in transit, or both? You will also want to know what type of encryption is being used. There are, for example, important differences between DES and AES. And make sure you understand who maintains the encryption keys before moving forward with a contract. Encryption should always be on the list of critical cloud security tips.
- How long has the cloud provider been in business, and what is its track record? If it goes out of business, what happens to your data? Will your data be returned in its original format?

Cloud Threats

- If a security incident occurs, what support will you receive from the cloud provider? While many providers promote their services as being unhackable, cloud-based services are an attractive target to hackers. Side channel, session riding, cross-site scripting, and distributed denial of service attacks are just some of the threats to data in the cloud.
- While you may not know the physical location of your services, they are located somewhere. And all physical locations face threats from fire, storms, natural disasters, and loss of power. If any of these events occur, how will the cloud provider respond, and what guarantee of continued services does it promise?
- Some reports indicate that in the next three years, more than four-fifths of all data center traffic will be based in the cloud. This means if you have not yet audited a cloud-based service, you most likely will do so soon.

Cloud Threats

- If a security incident occurs, what support will you receive from the cloud provider? While many providers promote their services as being unhackable, cloud-based services are an attractive target to hackers. Side channel, session riding, cross-site scripting, and distributed denial of service attacks are just some of the threats to data in the cloud.
- While you may not know the physical location of your services, they are located somewhere. And all physical locations face threats from fire, storms, natural disasters, and loss of power. If any of these events occur, how will the cloud provider respond, and what guarantee of continued services does it promise?
- Some reports indicate that in the next three years, more than four-fifths of all data center traffic will be based in the cloud. This means if you have not yet audited a cloud-based service, you most likely will do so soon.

Application-Development Approaches

- Applications are written in programming languages.
- Programming languages can be low level so that the system easily understands the language; they also can be high level, enabling humans to easily understand them but requiring translation for the system.
- The programs used to turn high-level programs into object- or machine-readable code are known as *interpreters*, *compilers*, or *assemblers*.
- Most applications are compiled. The information also can be grouped for the development process in various ways, including the following:

Application-Development Approaches

- **Data-oriented system development (DOSD):** DOSD uses a process of focusing on software requirements. It helps eliminate problems with porting and conversion because the client uses the data in its predescribed format. Stock exchanges, airlines, and bus and other transit companies use DOSD.
- **Object-oriented systems development (OOSD):** OOSD uses a process of solution specifications and models, with items grouped as objects. OOSD is valued because it can model complex relationships, work with many data types, and meet the demands of a changing environment.

Application-Development Approaches

- **Component-based development (CBD):** CBD helps objects communicate with each other. The benefit of CBD is that it enables developers to buy predeveloped tested software from vendors that is ready to be used or integrated into an application. Microsoft's Component Object Model (COM), Common Object Request Broker Architecture (CORBA), and Enterprise JavaBeans (EJB) are examples of component models.
- **Web-based application development (WBAD):** WBAD uses a process to standardize code modules to allow for cross-platform operation and program integration. WBAD offers the use of application-development technologies such as Extensible Markup Language (XML). Its components include Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery, and Integration (UDDI).

Application-Development Approaches

- No matter what approach is used for development, to a large extent, success depends on how well the different groups work and communicate together.
- DevOps (development operations) seeks to address this need. DevOps, which represents a change in IT culture, emphasizes the collaboration and communication of software developers and information technology (IT) professionals while seeking to improve collaboration between operations and development teams.
- That is of huge importance because, as most auditors understand, the relationship is often adversarial. DevOps seeks to address this issue and plays a pivotal role in cloud computing, but its principles apply to on-premises deployments as well.

N-tier

- The concept behind n-tier is to provide a model by which developers can create flexible and reusable applications.
- N-tier accomplishes this by separating an application into tiers, and developers acquire the option of modifying or adding a specific layer instead of reworking the entire application.
- Think of a tier as a physical structuring mechanism for the system infrastructure. A three-tier architecture is typically composed of a presentation tier, a domain logic tier, and a data storage tier. The most common implementation of n-tier is the three-tier approach.
- With a three-tier application, the work is divided into three major parts, and each part is distributed to a different place or location in a network:

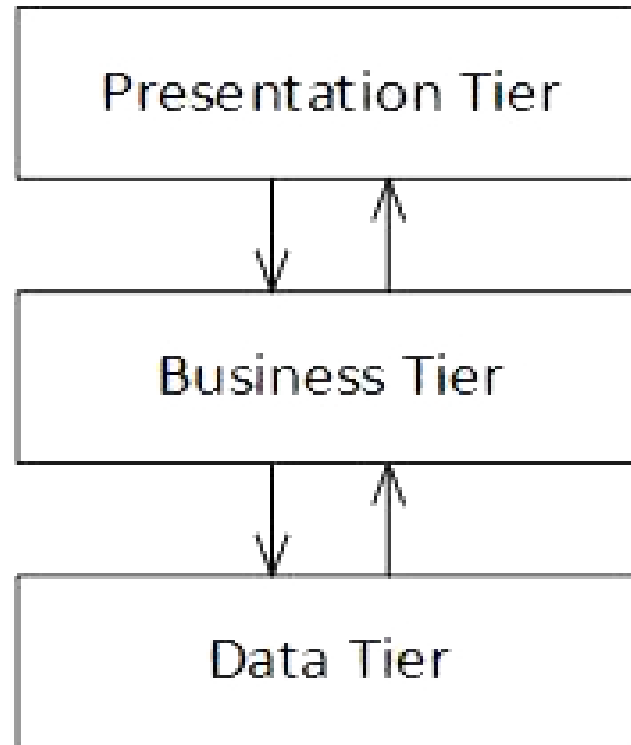
N-tier

- **The workstation or presentation interface:** This tier is located on the local workstation and may contain data that is local or unique for the workstation. It includes the coding that provides the graphical user interface (GUI) and application-specific entry forms or windows.
- **The business logic:** This tier is located on a local area network (LAN) server and acts as the server for client requests from workstations. The business logic unit determines what data is needed and acts as a client in relation to a third tier.
- **The database and programming related to managing it:** Located on a server, this third tier is the database and a program used to manage read/write access.

N-tier

- The primary advantage of the three-tier approach is that each of the three tiers can be developed concurrently by different programmers coding in different languages. One tier may have multiple layers. A layer can be thought of as a logical structuring mechanism for the elements that make up the software solution of one tier.
- The most common implementation deployment model consists of the LAMP (Linux, Apache, MySQL, and PHP/Python/Perl) stack. This is possible because the programming for a tier can be changed or relocated without affecting the other tiers. Some readers may see the similarity to the three-tier model and the OSI model in that each layer is responsible for certain items. This approach also lowers development costs and makes it easier for an enterprise or software packager to continually evolve an application as the need arises.

N-tier



Virtualization

- Cloud computing is not the only game in town. Virtualization is also an option for development or deployment of an application. Virtualized computing makes use of a virtual machine (VM).
- A *virtual server* is a virtualized computer that executes programs like a physical machine. VMware and Hyper-V are two examples of hypervisors. A hypervisor is basically the combination of software and hardware that creates and runs a virtual machine.
- Virtual machines are a huge trend and can be used for development and system administration and production, as well as to reduce the number of physical devices needed.

Virtualization

- One of the primary advantages of virtualized systems is server consolidation. Virtualization allows you to host many virtual machines on one physical server. Virtualization allows rapid deployment of new systems and offers the ability to test applications in a controlled environment. This reduces deployment time and makes better use of existing resources. Virtualization also helps with research and development.
- Virtual machine snapshots or checkpoints allow for easy image backup before changes are made and thus allow a means to quickly revert to the previous good image. This is very useful for all types of testing and production scenarios. Consider the fact that it's only a matter of time before a physical server fails or has a malfunction or even a hardware failure. In these situations, virtualization can be a huge advantage.

Virtualization

- Even with all the advantages, there are some items an auditor should review when dealing with virtualized systems.
- Virtualization adds another layer of complexity, which can cause problems that may be difficult to troubleshoot.
- Vulnerabilities associated with a single physical server hosting multiple companies' virtual machines include the fact that there is comingling of data.
- Even if none of these items are an issue, there is still the concern that the platform might be misconfigured. Such events can have devastating consequences for all the virtual systems residing on a single platform. The following are also concerns:

Virtualization

- **Privilege escalation:** This type of exploit allows an attacker to move from one user account to another either vertically or horizontally.
- **Live VM migration:** During live migration, a threat agent might attempt to capture the data as it moves over the network.
- **Data remanence:** It is entirely possible that in multi-tenant environments where VMs get provisioned and deprovisioned, residual data from previous use could be exposed.
- Virtualization systems fall into two categories: Type 1 and Type 2.
- **Type 1** hypervisors reside directly on hardware.
- **Type 2** hypervisors require an underlying OS. Examples of Type 2 systems include VirtualBox and VMware Workstation.
- Regardless of the type of VM being used, virtualized systems are an additional component that an auditor needs to consider.