System Implementation

System Implementation

- System Implementation is the critical phase of system development process where you can create executable software.
- Implementation is the stage in the software engineering process at which an executable software system is developed.
- Implementation is the process of realizing the design as a program.
- Systems implementation is the construction of the new system and the delivery of that system into production.
- **Systems implementation** is the process of:
  defining how the information system should be built (i.e., physical system design),
  ensuring that the information system is operational and used,
  ensuring that the information system meets quality standard (i.e., quality assurance).
- System implementation is the practice of creating or modifying a system to create a new business process or replace an existing business process.
- System implementation constructs the new information system and puts it into operation.
- During this phase, any new hardware and system software are installed and tested.
- Any purchased application software and databases are installed and configured.
- It is the way of carrying out a developed system into working condition.

The process of Coding, Testing and Installation

- Coding is the process whereby the physical design specifications created by the analysis team are turned into working computer code by the programming team.
- Depending on the size and complexity of the system, coding can be an involved, intensive activity.
- Once coding has begun, the testing process can begin and proceed in parallel.
- As each program module is produced, it can be tested individually, then as part of a larger program, and then as part of a larger system.
- Although testing is done during implementation, we must begin planning for testing earlier in the project.
- Planning involves determining what needs to be tested and collecting test data.
- This is often done during the analysis phase because testing requirements are related to system requirements.
- Installation is the process during which the current system is replaced by the new system.
- This includes conversion of existing data, software, documentation, and work procedures to those consistent with the new system.

The process of Documenting the system, Training Users and Supporting Users.

- Although the process of documentation proceeds throughout the life cycle, it receives, formal attention during the implementation phase because the end of implementation largely marks the end of the analysis team's involvement in systems development.
- Larger organizations also tend to provide training and support to computer users throughout the organization.
- Some of the training and support is very specific to particular application systems, whereas the rest is general to particular operating systems or off-the-shelf software packages.

Software Application Testing

Software testing is a method of assessing the functionality of a software program.

Different types of Tests

1. Inspections:
   - ✓ A testing technique in which participants examine program code for predictable language specific errors. Syntax, grammar, and some other routine errors can be checked by automated inspection software.
   - ✓ 60 to 90 percent of all software effects as well as provide programmers with feedback that enables them to avoid making the same types of errors in future work.
2. Desk checking:

- ✓ A testing technique in which the programmer or someone else who understands the logic of the program works through the code with a paper and pencil.
- ✓ The reviewer acts as the computer, mentally checking each step and its results for the entire set of computer instructions.

3. Unit testing:
   - ✓ Automated technique whereby each module is tested alone in an attempt to discover any errors that may exist in the module's code.
4. Integration testing:
   - ✓ The process of bringing together more than one modules that a program comprises for testing purposes.
5. System testing:
   - ✓ The process of bringing together of all of the programs that a system comprises for testing purposes.
   - ✓ Programs are typically integrated in a top-down incremental fashion. The system can be tested in two ways:

   i. **Black box testing**: In Black box test (also called functional test) internal code of the program are tested. It is called black box testing because the test cases are totally hidden for the general users.

   ii. **White box testing**: In white box test (also called glass box test) structure of the program is tested. It called white box testing because the test cases are totally visible to the general users and they can also make test cases.
6. Stub testing:
   - ✓ A technique used in testing modules, especially where modules are written and tested in a top down fashion, where a few lines of codes are used to substitute for subordinate modules.
   - ✓ Top-level modules contain many call to subordinate modules, we may wonder how they can be tested if the lower-level modules haven't been written yet.
   - ✓ This is called stub testing.
7. User acceptance testing:
   - ✓ Once the system tests have been satisfactorily completed, the system is ready for acceptance testing, which is testing the system in the environment where it will eventually be used.

   i. **Alpha testing:**

   User testing of a completed information system using simulated data. The types of tests performed during alpha testing include the following:

   a. Recovery testing: Forces the software to fail in order to verify that recovery is properly performed.
   b. Security testing: Verifies that protection mechanisms built into the system will protect it from improper penetration.
   c. Stress testing: Tries to break the system (eg: what happens when a record is written to the database with incomplete information or what happens under extreme online transaction loads or with a large number of concurrent users).
   d. Performance testing: Determines how the system performs in the range of possible environments in which it may be used (eg: different hardware configurations, networks, operating systems).

   iii. **Beta testing:**

   User testing of a completed information system using real data in the real user environment. The intent of the beta test is to determine whether the software, documentation, technical support and training activities work as intended. Beta testing can be viewed as a rehearsal of the installation phase.

Installation

The process of moving from the current information system to the new one is called installation. Different ways of installation are:

1. **Direct installation**:
   - Changing over from the old information system to a new one by turning off the old system when the new one is turned on.
   - Any errors resulting from the new system will have a direct impact on the users.
   - If the new system fails, considerable delay may occur until the old system can again be made operational and business transactions are reentered to make the database up to date.
   - Direct installation can be very risky.
   - Direct installation requires a complete installation of the whole system.

- For a large system, this may mean a long time until the new system can be installed, thus delaying system benefits or even missing the opportunities that motivated the system request.
- It is the least expensive installation method, and it creates considerable interest in making the installation a success.

2. **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.
   - All of the work done by the old system is concurrently performed by the new system.
   - Outputs are compared to help determine whether the new system is performing as well as the old.
   - Errors discovered in the new system do not cost the organization much, if anything, because errors can be isolated and the business can be supported with the old system.
   - Because all work is essentially done twice, a parallel installation can be very expensive, running two systems implies employing two staffs to operate and maintain.
   - A parallel approach can also be confusing to users because they must deal with both systems.
   - A parallel approach may not be feasible, especially if the users of the system cannot tolerate redundant effort or if the size of the system is large.

3. **Pilot installation:**
   - It is also known as single-location installation.
   - Rather than converting all of the organization at once, single location installation involves changing form the current to the new system in only one place or in a series of separate sites over time.
   - The single location may be a branch office, a single factory, or one department, and the actual approach used for installation in that location may be any of the other approaches.
   - The key advantage to single location installation is that it limits potential damage and potential cost by limiting the effects to a single site.
   - Once management has determined that installation has been successful at one location, the new system may be deployed in the rest of the organization, possibly continuing with installation at one location at a time.
   - Problems with the system can be resolved before deployment to other sites. Even though the single location approach may be simpler for users, it still places a large burden on information system staff to support two versions of the system.
   - Problems are isolated at one site at a time.
   - IS staff members can devote all of their efforts to success at the pilot site.
   - If different locations require sharing of data, extra programs will need to be written to synchronize the current and new systems.

4. **Phased installation:**
   - It is also called staged installation.
   - Different parts of the old and new systems are used in cooperation until the whole new system is installed.
   - By converting gradually, the organization’s risk is spread out over time and place.
   - Also phased installation allows for some benefits from the new system before the whole system is ready.
   - For example, a new data-capture methods can be used before all reporting modules are ready.
   - For a phased installation, the new and replaced systems must be able to coexist and probably share data.
   - Thus bridge programs connecting old and new databases and programs often must be built.
   - Sometimes the new and old systems are so incompatible that pieces of the old system cannot be incrementally replaced so this strategy is not feasible.
   - A phased approach requires careful version control, repeated conversions at each phase, and a long period of change, which may be frustrating and confusing to users.

Documenting the system

- Documentation is the process of collecting, organizing, storing and maintaining a complete record of system and other documents used or prepared during the different phases of the life cycle of system.
- System cannot be considered to be complete, until it is properly documented.
- Proper documentation of system is necessary due to the following reasons:

1. It solves the problem of indispensability of an individual for an organization. Even if the person, who has designed or developed the system, leaves the organization, the documented knowledge remains with the organization, which can be used for the continuity of that software.
2. It makes system easier to modify and maintain in the future. The key to maintenance is proper and dynamic documentation. It is easier to understand the concept of a system from the documented records.
3. It helps in restarting a system development, which was postponed due to some reason. The job need not be started from scratch, and old ideas may still be easily recapitulated from the available documents, which avoids duplication of work, and saves lot of times and effort.

Types of documentation

a. System documentation
   System documentation records detailed information about a system's design specification, its internal workings and its functionality. System documentation is intended primarily for maintenance programmers. It contains the following information:
   1. A description of the system specifying the scope of the problem, the environment in which it functions, its limitation, its input requirement, and the form and type of output required.
   2. Detailed diagram of system flowchart and program flowchart.
   3. A source listing of all the full details of any modifications made since its development.
   4. Specification of all input and output media required for the operation of the system.
   5. Problem definition and the objective of developing the program.
   6. Output and test report of the program.
   7. Upgrade or maintenance history, if modification of the program is made.

   There are two types of system documentation. They are:

   i. Internal documentation
      Internal documentation is part of the program source code or is generated at compile time.
   ii. External documentation
      External documentation includes the outcome of structured diagramming techniques such as dataflow and entity-relationship diagrams.
b. User documentation
   User documentation consists of written or other visual information about an application system, how it works and how to use it. User documentation is intended primarily for users. It contains the following information:
   1. Set up and operational details of each system.
   2. Loading and unloading procedures.
   3. Problems which could arise, their meaning reply and operation action.
   4. Special checks and security measures.
   5. Quick reference guides about operating a system in a short, concise format.

Training and supporting users

The type of training needed will vary by system type and user expertise. Types of training methods are:

a. Resident expert
b. Traditional instructor-led classroom training
c. E-learning/distance learning
d. Blended learning (combination of instructor-led and e-learning)
e. Software help components
   Electronic performance support system: component of a software package or an application in which training and educational information is embedded.
f. External sources, such as vendors

Computing supports for users has been provided in one of a few forums:

Automating support: online support forums provide users access to information on new releases, bugs and tips for more effective users access to information on new releases, bugs and tips form more effective usage. Forums are offered over the internet or over company intranets.

Providing support through a help desk: A help desk is an information systems department function and is staffed by IS personnel. The help desk is the first place users should call when they need assistance with an information system. The help desk staff members either deal with the users questions or refer the users to the most appropriate person.

Quality assurance

- A process or procedure for minimizing errors and ensuring quality in products.
- Poor quality can result from inaccurate requirements, design problems, coding errors, faulty documentation, and ineffective testing.
- A quality assurance (QA) team reviews and tests all applications and systems changes to verify specifications and software quality standards.
- A successful organization must improve quality in every area, including its information systems.
- Top management must provide the leadership, encouragement, and support needed for high-quality IT resources.
- The main objective of quality assurance is to avoid problems or to detect them as soon as possible.
- In an effort to achieve high standards of quality, software systems developers should consider software engineering concepts, internationally recognized quality standards, and careful project management techniques.

Software Engineering:

- Because quality is so important, you can use an approach called software engineering to manage and improve the quality of the finished system.
- Software engineering is a software development process that stresses solid design, accurate documentation, and careful testing.

International Organization for Standardization (ISO):

- International Organization for Standardization (ISO) is a worldwide body that establishes quality standards for products and services.
- ISO standards include everything from internationally recognized symbols, to the ISBN numbering system that identifies textbook.
- In addition, ISO seeks to offer a global consensus of what constitutes good management practices — practices that can help firms deliver consistently high-quality products and services.
- Because software is so important to a company’s success, many firms seek assurance that software systems either purchased or developed in-house, will meet rigid quality standards.
- In 1991, ISO established a set of guidelines called ISO 9000-3, which provided a quality assurance framework for developing and maintaining software.
- A company can specify ISO 9000-3 standards when it purchases software from a supplier or use ISO guidelines for in-house software development to ensure that the final result measures up to ISO standards.
- ISO requires a specific development plan, which outlines a step-by-step process for transforming user requirements into a finished product.
- ISO standards can be quite detailed.
- For example, ISO requires that a software supplier document all testing and maintain records of test results.
- If problems are found, they must be resolved, and any modules affected must be retested. Additionally, software and hardware specifications of all test equipment must be documented and included in the test records.

Levels of assurance: Analysts use four levels of quality assurance: testing, verification, validation, and certification.

a. **Testing:**
    - Systems testing is an expensive but critical process that can take as much as 50 percent of the budget for program development.
    - The common view of testing held by users is that it is performed to prove that there are no errors in a program.
    - However, this is virtually impossible, since analysts cannot prove that software is free and clear of errors.
    - Therefore, the most useful and practical approach is with the understanding that testing is the process of executing a program with explicit intention of finding errors that is, making the program fail.
    - The tester, who may be an analyst, programmer, or specialist trained in software testing, is actually trying to make the program fail.
    - A successful test, then, is one that finds an error.

- Analysts know that an effective testing program does not guarantee systems reliability.
- Reliability is a design issue.
- Therefore, reliability must be designed into the system. Developers cannot test for it.

b. **Verification and Validation**:

- Like testing, verification is also intended to find errors.
- Executing a program in a simulated environment performs it.
- Validation refers to the process of using software in a live environment in order to find errors.
- When commercial systems are developed with the explicit intention of distributing them to dealers for sale or marketing them through company – owned field offices, they first go through verification, some-times called alpha testing.
- The feedback from the validation phase generally produces changes in the software to deal with errors and failures that are uncovered.
- Then a set of user sites is selected that puts the system into use on a live basis.
- These beta test sites use the system in day- to - day activities; they process live transactions and produce normal system output.
- The system in live is very sense of the word, except that the users are aware they are using a system that can fail.
- But the transactions that are entered and the persons using the system are real. Validation many continue for several months.
- During the course of validating the system, failure may occur and the software will be changed.
- Continued use may produce additional failures and the need for still more change.

c. **Certification**:

- Software certification is an endorsement of the correctness of the program, an issue that is rising in importance for information systems applications.
- There is an increasing dependence on the purchase or lease of commercial software rather than on its in-house development.
- However, before analysts are willing to approve the acquisition of a package, they often require certification of the software by the developer or an unbiased third party.
- For example, selected accounting firms are now certifying that a software package in fact does what the vendor claims it does and in a proper manner.
- To so certify the software, the agency appoints a team of specialists who of specialists who carefully examine the documentation for the system to determine what the vendor claims the system does and how it is accomplished.
- Then they test the software against those claims.
- If no serious discrepancies or failures are encountered, they will certify that the software does what the documentation claims.
- They do not, however, certify that the software is the right package for a certain organization.
- That responsibility remains with the organization and its team of analysts.

Maintenance

- Correcting and upgrading process of the system is called system maintenance.
- Maintenance is necessary to eliminate errors in the working system during its working life and to tune the system to any variations in its working environment.
- Different types of maintenance are:

1. Corrective maintenance: Corrective maintenance refers to changes made to repair defects in the design , coding or implementation of the system. For example if we had recently purchased a new home , corrective maintenance would involve repairs made to things that had never worked as designed, such as a faulty electrical outlet or misaligned door. More corrective maintenance problems surface soon after installation. When corrective maintenance problems surface, they are typically urgent and need to be resolved to curtail possible interruptions in normal business activities. Of all types of maintenance, corrective accounts for as much as 75 percent of all maintenance activity. It simply focuses on removing defects from an existing system without adding new functionality.

2. Adaptive maintenance: Adaptive maintenance involves making changes to an information system to evolve its functionality to changing business needs or to migrate it to a different operating environment. Within a home, adaptive maintenance might be adding storm windows to improve the cooling performance of an air conditioner. Adaptive maintenance is less urgent than corrective maintenance because business and technical changes typically occur over

some period of time. Adaptive maintenance is generally a small part of an organizations maintenance effort, but it adds value to the organization.

3. Perfective maintenance: Perfective maintenance involves making enhancements to improve processing performance or interface usability or to add desired, but not necessarily required, system features. In home perfective maintenance would be adding a new room. Many systems professionals feel that perfective maintenance is not really maintenance but rather new development. Perfective maintenance usually is cost effective during the middle of the system's operational life. Early in systems operation, perfective maintenance usually is not needed. Later, perfective maintenance might be necessary, but have a high cost.
4. Preventive maintenance: Preventive maintenance involves changes made to a system to reduce the chance of future system failure. An example of preventive maintenance might be to increase the number of records that a system can process far beyond what is currently needed or to generalize how a system sends a report information to a printer so that the system can easily adapt to changes in printer technology. In our home example, preventive maintenance could be painting the exterior to better protect the home from severe weather conditions.

The process of maintaining information system

Maintenance phase is the last phase of SDLC. The major activity occur within maintenance:

1. Obtaining maintenance requests
   Obtaining maintenance requests requires that a formal process be established whereby users can submit system change requests. When developing the procedures for obtaining maintenance requests, organizations must also specify an individual within the organization to collect these requests and manage their dispersal to maintenance personnel.
2. Transforming requests into changes
   Once a request is received, analysis must be conducted to gain an understanding of the scope of the request. It must be determined how the request will affect the current system and how long such a project will take. As with the initial development of a system, the size of a maintenance request can be analyzed for risk and feasibility.
3. Designing changes
   Change request can be transformed into a formal design change, which can then be fed into the maintenance implementation phase.
4. Implementing changes

The cost of Maintenance

Information systems maintenance costs are a significant expenditure. For some organizations, as much as 60 to 80 percent of their information systems budget is allocated to maintenance activities. On average, 52 percent of a company's programmers are assigned to maintain existing software. Only 3 percent are assigned to new application development.
Numerous factors influence the maintainability of a system. Of these factors, three are most significant; the number of latent defects, the number of customers, and documentation quality. The others are personnel, tools and software structure.

Latent defects: This is the number of unknown errors existing in the system after it is installed. Because corrective maintenance accounts for most maintenance activity, the number of latent defects in a system influences most of the cost associated with maintaining a system.
Number of customers for a given system: Greater the number of customers, the greater the maintenance costs. For example, if a system has only one customer, problem and change requests will come from only one source. Training, error reporting and support will be simpler.
Quality of system documentation: Without quality documentation, maintenance efforts can increase exponentially. Quality documentation makes it easier to find code that needs to be changed and to understand how the code needs to be changed. Good documentation also explains why a system does what it does and why alternatives were not feasible.
Maintenance personnel: In some organizations, best programmers are assigned to maintenance. Highly skilled programmers are needed because the maintenance programmer is typically not the original programmer and must quickly understand and carefully change the software.
Tools: Tools that can automatically produce system documentation where none exists can also lower maintenance costs. Tools that can automatically generate new code based on system specification changes can dramatically reduce maintenance time and costs.
Well structured programs: Well designed programs are easier to understand and fix.

Managing Maintenance

Maintenance activities consume more and more of the systems development budget, maintenance management has become increasingly important.

Managing Maintenance Personnel

Many organizations had a “maintenance group” that was separate from the “development group”. With the increased number of maintenance personnel, the development of formal methodologies and tools, changing organizational forms, end user computing, and the widespread use of very high level languages for the development of some systems, organizations have rethought the organization of maintenance and development personnel. One key issue is that many systems professionals don’t want to perform maintenance because they feel that it is more exciting to build something new than change an existing system.

Measuring Maintenance Effectiveness

To measure effectiveness, we must measure the following factors:

Number of failures
Time between each failure
Type of failure

Measuring the number of and time between failures will provide us with the basis to calculate a widely used measure of system quality. This metric is referred to as the mean time between failures (MTBF).

A more revealing method of measurement is to examine the failures that are occurring. Over time, logging the types of failures will provide a very clear picture of where, when and how failures occur. Tracking the types of failures also provides important management information for future projects. For example, if a higher frequency of errors occurs when a particular development environment is used, such information can help guide personnel assignments, training courses, or the avoidance of a particular package, language, or environment during future development. Without measuring and tracking maintenance activities, we cannot gain the knowledge to improve or know how well we are doing relative to the past. To effectively manage and to continuously improve, we must measure and assess performance over time.

Formal Technical Review (FTR)
A formal technical review is a software quality assurance activity performed by software engineers (and others).
The objectives of the FTR are

- to uncover errors in function, logic, or implementation for any representation of the software;
- to verify that the software under review meets its requirements;
- to ensure that the software has been represented according to predefined standards;
- to achieve software that is developed in a uniform manner;
- to make projects more manageable.

In addition, the FTR serves as a training ground, enabling junior engineers to observe different approaches to software analysis, design, and implementation. The FTR also serves to promote backup and continuity because a number of people become familiar with parts of the software that they may not have otherwise seen.

The FTR is actually a class of reviews that includes walkthroughs, inspections, round-robin reviews and other small group technical assessments of software. Each FTR is conducted as a meeting and will be successful only if it is properly planned, controlled, and attended. In the sections that follow, guidelines similar to those for a walkthrough are presented as a representative formal technical review.

Walkthrough

In software engineering, a **walkthrough** or **walk-through** is a form of software peer review "in which a designer or programmer leads members of the development team and other interested parties through a software product, and the participants ask questions and make comments about possible errors, violation of development standards, and other problems".