

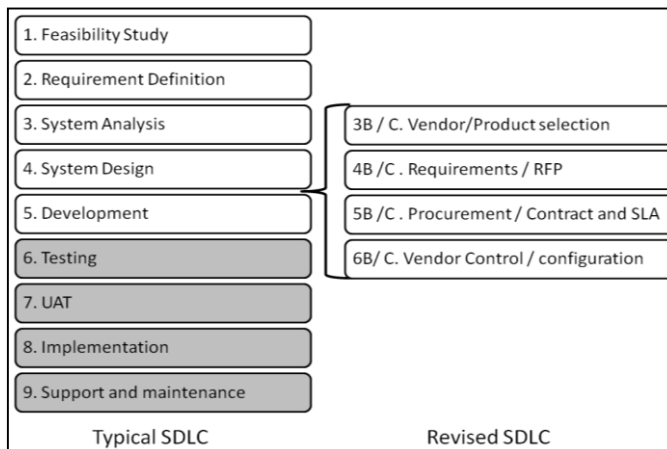
Chapter 6: Implementation and Maintenance.....	2
6.1 Learning objectives	2
6.1 Introduction	2
6.2 Testing	3
6.2.1 Unit Testing	3
6.2.2 Integration Testing	6
6.2.3 Regression Testing.....	7
6.2.4 System Testing	7
6.2.5 Final Testing	8
6.2.6 Other types of Testing	9
6.3 Implementation	11
6.3.1 Implementation Strategies	12
6.3.2 Preparing for implementation	14
6.3.3 Training	15
6.3.4 Conversion	16
6.4 Change management process	17
6.4.1 Emergency Changes	19
6.4.2 Implementing Changes into Production	20
6.4.3 Segregation of Duties	20
6.4.4 Configuration Management.....	21
6.5 Summary.....	22
6.6 Questions.....	22
6.7 Answers and Explanations	23

CHAPTER 6: IMPLEMENTATION AND MAINTENANCE

6.1 Learning objectives

This chapter focuses on providing information on implementation methodologies that organisation may adopt depending upon the nature and complexity of solution developed or acquired and nature of services offered by the solutions. The chapter also focuses on providing information on activities that need to be performed before and after implementation (maintenance of application).

6.1 Introduction



This chapter discusses phases 6 to 9 of SDLC

Phase 6 is testing of developed or configured software.

Phase 7 is conducting user acceptance testing (UAT)

and sign-off from users. (UAT is discussed in testing section.)

Phase 8 is implementation methods being used by various organisations.

Phase 9 is change management process required for support and maintenance activities

Application software developed or acquired requires appropriate process for implementing and maintenance in order to realize benefits effectively and efficiently. The best solution to ensure smooth implementation, configuration, change and release management processes need to be implemented. These processes should be focused on reliability, availability and security of production systems. Maintenance generally involves updating and changing IT systems (application and infrastructure) to suit to changing requirement of business. Every change must

be assessed, planned, tested, approved, documented, communicated and carried out without any undesirable consequences and minimum downtime for business processes. The IS auditor should be aware of the processes for implementation and change management including controls to ensure segregation of duties between development, test and the production environment.

The change management processes discussed in section 6.4 and configuration management process discussed section 6.5 are common processes used for implementation of changes as well as new application systems.

6.2 Testing

The success of information systems depends upon the quality of software that supports the system. Testing of software before deploying in production to ensure it delivers as per requirements is most essential aspect of quality apart from documentation, compliance with coding standards, version control discipline and user training.

Although testing phase comes much later in the life cycle, planning for testing starts with the commencement of System Development life cycle i.e. during requirement gathering phase. Testing is a process that focuses on correctness, completeness and quality of developed computer software. Testing should systematically uncover different classes of errors in a minimum amount of time with a minimum amount of efforts. The data collected through testing can also provide an indication of the software's reliability and quality. However, testing cannot show the absence of defect, it can only show that software defects are present. There are various types of testing performed during development of life cycle that are discussed in ensuing paragraphs.

6.2.1 Unit Testing

In computer programming, unit testing is a verification and validation method in which a programmer tests whether individual units of source code are fit for use. A unit is the smallest functional part of an application often called as module. It can be an individual program, function, procedure, or may belong to a base/super class, abstract class or derived/child class.

Unit tests are typically written based on requirement specifications and run by testing professionals or software developers to ensure that code meets these requirements and behaves as intended. The goal of unit testing is to isolate each component of the program and

show that they are correct. A unit test provides a strict, written contract that the piece of code must satisfy.

There are five categories of tests typically performed on a program unit. Such typical tests are described as follows:

1. **Functional Tests:** Functional Tests check 'whether programs do, what they are supposed to do or not'. The test plan specifies operating conditions, input values, and expected results, and as per this plan, programmer checks by inputting the values to see whether the actual result and expected result match. These test data values are prepared in advance for all possible permutations the data can acquire during live run. This can have two types:

- a. **Positive test:** Where tester collects the expected values the data can possess. Sometimes tester may use sanitized live data for testing.
- b. **Negative test:** Where tester provides value sets that data should not possess anytime. Here the program should flash the error with suitable message.

For example, if data field is used to store amount and can acquire a value between 0 and 1 crore, positive test should provide expected results and negative test should flash error if values are beyond the specified range.

2. **Performance Tests:** Performance tests are designed to verify the expected performance criteria of program.

There are different performance parameters like response time (time required to receive input and deliver confirmation), execution time (processing of single data value should be less than 100 microseconds), throughput (1000 values must be processed in once second), primary (RAM/CPU) and secondary memory (Storage) utilization and rate of traffic flow on data channels and communication links (number of messages per second).

3. **Stress Tests:** Stress testing is a form of testing that is used to determine the stability of a given system or entity. It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results. These tests are designed to overload a program in various ways. The purpose of a stress test is to determine the limitations of the program. (For example if access to web application is expected to be 10000 hits per second, if the program can stand this load and how it behaves when load exceeds or during a sort/search operation, available memory can be reduced to find out whether the program is able to handle the situation.).

4. **Structural Tests:** Structural Tests are concerned with examining the internal processing logic of a software system. Particularly when the program is expected to behave differently depending upon value of data set. Programmer may code for known values and might forget to code for unknown values where program might misbehave. For example tax calculation where depending upon value different rates is applied or if

division operation is involved and data set gets value of zero, program may terminate abruptly or go in loop without response.

5. **Parallel Tests:** These are applicable during change management or reengineering where the same test data is used in the new and old system and the output results are then compared.

Methods of Unit Testing

- A. Static analysis testing and
- B. Dynamic testing.

A. Static Testing: Static analysis tests are conducted on source programs and do not normally require executions in operating conditions. Typical static analysis techniques include the following:

1. **Desk check:** This is done by the programmer to check the logical syntax errors, and deviation from coding standards. As name suggests programmer uses paper and pen to verify the logic of code by jotting down values of data sets and thinking like computer to arrive at possible values.
2. **Structured Walk-through:** Desk check performed with team or peers who scan through the text of the program and explanation try to uncover errors.
3. **Code Inspection:** The program is reviewed by a formal committee. Review is done with formal checklists.

B. Dynamic Analysis Testing: Such testing is normally conducted through execution of programs in operating conditions. Typical techniques for dynamic testing and analysis include the following:

1. **Black Box Testing:** This testing primarily focuses on functionalities as specified in requirements. It assume code set (program/module/function etc.) as a black box without knowledge of internal structure, and focuses on input values and compares expected output for each value. The test designer selects typical inputs including simple, extreme, valid and invalid input-cases and executes to uncover errors.
This method of test design is applicable to all levels of software testing i.e. unit, integration, functional testing, system and acceptance. While this method can uncover unimplemented parts of the specification, one cannot be sure that all existent paths are tested. If a module performs a function, which it is not supposed to perform, for example: running of a Trojan code, the black box test does not identify it.
2. **White Box Testing:** This testing goes further than black box testing and also tests internal perspective of the code/system/function to design test cases. The

tester chooses test case inputs to exercise paths through the code and determines the appropriate outputs. Since the tests are based on the actual implementation, if the implementation changes, the tests probably will need to change, too. It is applicable at the unit, integration and system levels of the testing process. It is typically applied to the unit. While it normally tests paths within a unit, it can also test paths between units during integration, and between subsystems during a system level test. After obtaining a clear picture of the internal workings of a product, tests can be conducted to ensure that the internal operation of the product conforms to specifications and all the internal components are adequately exercised. There are automated tools (like test tools/debuggers etc.) available to conduct this type of testing.

3. **Grey Box Testing:** It is a technique that is in between uses black box testing and white box testing. Tester applies a limited number of test cases to the internal workings of the software under test. In the remaining part of the grey box testing, one takes a black box approach in applying inputs to the software under test and observing the outputs.

6.2.2 Integration Testing

Unit testing focuses on testing of different modules/functions and programs that are small part of entire information system being developed. These modules are expected to work together to achieve objectives of information system. For example Internet banking is a system consisting of various functions like saving account management, time deposit management, loan account management, third-party fund transfer, standing instruction, getting statements of accounts etc. While developing programs/functions, each service function is developed separately and tested in unit testing. Now it is necessary to test if these modules/functions work together seamlessly and communicate appropriately during execution. Objective is to evaluate the validity of integration of two or more components that pass information to one another. Integration testing puts together modules that have been unit tested and applies tests defined. There are two approaches for integration testing:

1. **Bottom-up Integration:** It is the traditional strategy used to integrate the components of a software system starting from smallest module/function/program. It consists of unit testing, followed by sub-system testing. Bottom-up testing is easy to implement as at the time of module testing, tested subordinate modules are available. The disadvantage; however is that testing of major decision / control points is deferred to a later period. For example in above example of internet banking it will test communication between different modules using smallest level of module like saving bank account, fund transfer and then statement of accounts to ensure previous

transaction reflects in statement, and so on, however it might not ensure the overall control on passing parameters required for session time out or inactive session

2. Top-down Integration: This starts with the main routine followed by the stubs being substituted for the modules which are directly subordinate to the main module. Considering above example, the testing will start from opening login screen and then login, then selecting function one by one. An incomplete portion of a program code is put under a function (called stub) to allow the function. Here a stub is considered as black box and assumed to perform as expected, which is tested subsequently. Once the main module testing is complete, stubs are substituted with real modules one by one, and these modules are tested. This process continues till the atomic (smallest) modules are reached. Since decision-making processes are likely to occur in the higher levels of program hierarchy, the top-down strategy emphasizes on major control decision points encountered in the earlier stages of a process and detects any error in these processes. The difficulty arises in the top-down method, because the high-level modules are tested with stubs and not with actual modules.

6.2.3 Regression Testing

It is a testing performed during change management when a function/module/program is changed or added to existing software, in order to ensure that new/changed functions executes properly and integrates with other modules as expected. It is required since new data flow paths are established, new I/O may occur and new control logic is invoked. These changes may cause problems with functions that previously worked flawlessly. In the context of the integration testing, the regression tests ensure that changes or corrections have not introduced new faults. The data used for the regression tests should be the same as the data used in the original test.

6.2.4 System Testing

It is a process in which software and other system elements are tested as a whole. System testing begins either when the software as a whole is operational or when the well-defined subsets of the software's functionality have been implemented. The purpose of system testing is to ensure that the new or modified system functions properly. These test procedures are often performed in a non-production test environment. The types of testing that might be carried out with various other objectives described below:

1. **Recovery Testing:** This is the activity of testing 'how well the application is able to recover from crashes, hardware failures and other similar problems'. Recovery testing

is the forced failure of the software in a variety of ways to verify that recovery is able to be performed properly, in actual failures

2. **Security Testing:** This is the process to determine that an Information System protects data and maintains functionality as intended. The three basic security concepts that need to be covered by security testing are – confidentiality, integrity, availability. In addition the software may further be tested for user management requirements require i.e. authentication, authorization, and non-repudiation and log maintenance.
3. **Stress or Volume Testing:** Stress testing is a form of testing that is used to determine the stability of a given system or entity based on the requirements and expected data growth. It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results. Stress testing may be performed by testing the application with large quantity of data during peak hours to test its performance.
4. **Performance Testing:** Software performance testing is performed on various parameters like response time, speed of processing, effectiveness use of a resources (RAM, CPU etc.), network, etc. This testing technique compares the new system's performance with that of similar systems using available industry benchmarks.

6.2.5 Final Testing

It is conducted if results of system testing are satisfactory and when the system is just ready for implementation. This testing is performed at two levels:

1. At technical level quality assurance testing is performed
2. At functional level user acceptance testing is performed.

Quality Assurance Testing (QAT): QAT focuses on conforming to the quality standards of the organisation accepted before development. It includes documented specifications, technology employed, use of coding standards, and the application meets the documented technical specifications and deliverables. QAT is performed primarily by the technical (IT) department. The participation of the end user is minimal and on request. QAT does not focus on functionality testing.

User Acceptance Testing (UAT): It is a user extensive activity and participation of functional user is a primary requirement for UAT. The objective of UAT is to ensure that the system is production-ready and satisfies all accepted (baselined) requirements. UAT is a formal process and may include:

- i. Definition of test strategies and procedures
- ii. Design of test cases and scenarios
- iii. Execution of the tests
- iv. Utilization of the results to verify system readiness

Acceptance criteria defined along with requirement specifications includes that deliverables must satisfy the predefined needs of the user. A UAT plan must be documented for the final test of the completed system. The tests are written from a user's perspective and should test the system in a manner as close to production as possible. For example, tests may be based around typical predefined, business process scenarios. If new business processes have been developed to accommodate the new or modified system they should also be tested at this point. A key aspect of testing should also include testers seeking to verify that supporting processes integrate into the application in an acceptable fashion. Successful completion would generally enable a project team to hand over a complete integrated package of application and supporting procedures.

1. **UAT is a stage in SDLC where end users finally accept the developed application system. This is required for all situations of acquiring software i.e. software developed in-house, or by outsourced team or purchased and configured by vendor. A formal sign-off generally marks end of development process.**
2. **UAT should be performed in a secure testing or staging environment where both source and executable code are protected to ensure that unauthorized or last-minute changes are not made to the system unless authorized and the standard change management process is followed. In the absence of controls, the risk of introducing unauthorized changes/malicious patches/Trojan horse programs is very high.**
3. **Users should develop test cases or use data of live operations of a specified period to confirm whether the processing of data by new application is providing correct results, has required controls and the reports meet the management requirements.**

Many organizations expects report from IS auditor after tests are completed. The IS auditor should issue an opinion to management as to whether the system meets documented business requirements, has incorporated appropriate controls, and may be migrated to production. This report should also identify and explain the risk that the organization might be exposed by implementing the system.

6.2.6 Other types of Testing

When any complex application/software is intended for general and wide spread use developers want to make sure that product delivers diverse requirements of general users, organisations may consider alpha and beta testing. For example Microsoft performs this type of testing on new product before making it available commercially.

Alpha Testing: This is the first stage, often performed by users within the organisation by the developers, to improve and ensure the quality/functionalities as per users' satisfaction.

Beta Testing: This is the second stage, generally performed after the deployment of the system. It is performed by the external users, during the real life execution of the project. It normally involves sending the product outside the development environment for real world exposure and receives feedback for analysis and modifications, if any.

Automated testing: In software testing, automation of testing is performed using special software (separate from the software being tested) to control the execution of tests and the comparison of actual outcomes with predicted outcomes. Test automation can automate some repetitive but necessary tasks in a formalized testing process already in place, or add additional testing that would be difficult to perform manually.

Integrated Testing: Some organisations rely on integrated test facilities. Test data usually are processed in production-like systems. This confirms the behaviour of the new application or modules in real-life conditions. These conditions include peak volume and other resource-related constraints. In this environment, IS will perform their tests with a set of fictitious data whereas client representatives use extracts of production data to cover the most possible scenarios as well as some made-up data for scenarios that would not be tested by the production data.

Some organisations use a subset of production data in a test environment, such production data may be altered or scrambled to mask the confidential data. This is often the case where the acceptance testing is done by team members who, under usual circumstances, would not have access to such production data. These tools help in building test cases and also generate test data based on conditions. However, using production data may not help in identifying negative test cases.

Accreditation of software: Although it is not type of testing, many organisations insist on certification and accreditation of software. Generally this is done by the software development houses before taking product to market. In case of tailor-made software certification/accreditation should only be performed once the system is implemented and in operation for some time to produce the evidence needed for certification/accreditation processes. This process includes evaluating program documentation and testing effectiveness. The process will result in a final decision for deploying the business application system.

Security testing: (Application scans and penetration testing) For information security issues, the evaluation process includes reviewing security plans, the risk assessments results

along with response decision, and the evaluation of processes to be deployed. The result of security assessment focuses on measuring effectiveness of the security controls. Security testing provides assurance to the business owner.

Security testing of web application for identified external threats (like SQL injection, cross site scripting etc.) is necessary to ensure that the application can sustain an attack by the hacker who is trying to breach the security.

While reviewing testing process IS auditors focus on getting answers to following questions:

- 1. Whether the test-suite prepared by the testers includes the actual business scenarios?**
- 2. Whether test data used covers all possible aspects of system?**
- 3. Whether CASE tools like 'Test Data Generators' have been used?**
- 4. Whether test results have been documented?**
- 5. Whether tests have been performed in their correct order?**
- 6. Whether modifications needed based on test results have been done?**
- 7. Whether modifications made have been properly authorized and documented?**

Testers generally perform black box testing (Penetration test) by trying to simulate attacks on hosted application. This is then followed by performing grey box and/or white box testing that includes code review to identify the issues in coding practices that might introduce the vulnerabilities in the application. These can be avoided by including secure coding practices in coding standard developed by the organisations.

6.3 Implementation

Application software developed shall be implemented once it is tested and UAT has been signed off. However the planning for implementation must start much earlier in SDLC, many times after feasibility study. Planning involves:

- Selecting Implementation Strategies
- Preparing for implementation
 - Deciding on hardware and ordering (if required) in advance so as to be available in time
 - Deciding on site where infrastructure to be made available

- Training
- Conversion of data to suit to the requirements of new application.

6.3.1 Implementation Strategies

Considering the nature of business operation appropriate implementation strategy must be decided, much earlier in SDLC. Generally it is decided once the design is finalized or in case of acquisition once the application is selected. Organisation can adopt one of the four strategies, which are described below:

Cut-off or Direct Implementation / Abrupt Change-Over: This is achieved through an abrupt takeover – an all or no approach. With this strategy, the changeover is done in one operation, completely replacing the old system in one go. Fig 6.1 depicts Direct Implementation, which usually takes place on a set date, often after a break in production or a holiday period so that time can be used to get the hardware and software for the new system installed without causing too much disruption.

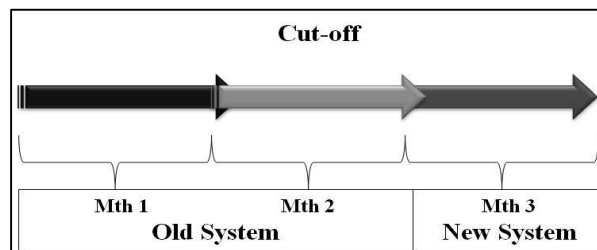


Fig. 6.1: Cut-off or Direct implementation

The challenge in cut-off implementation is that roll-back is most difficult and hence planning must be meticulously done. Also conversion activity must start well in advance and must be properly planned.

Phased Changeover: With this strategy, implementation can be staged with conversion to the new system taking place gradually. This is done based on business operations. For example, converting one function (e.g. marketing) on new system, wait for the same be stabilized and then take another function (Finance/HR/production etc.) If a phase is successful then the next phase is started, eventually leading to the final phase when the new system fully replaces the old one as shown in Fig. 6.2.

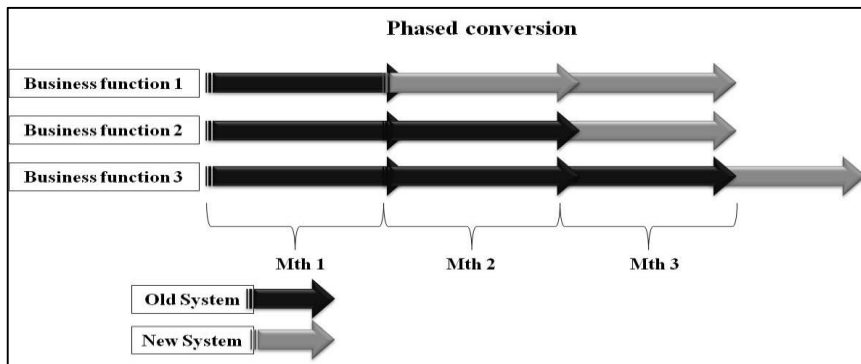


Fig. 6.2: Phased changeover

Phase changeover might require more time to implement however it helps in stabilizing one function before starting another.

Pilot Changeover: With this strategy, the new system replaces the old one in one operational area or with smaller scale. Any errors can be rectified and new system is stabilized in pilot area, this stabilized system is replicated in operational areas throughout the whole system. For example converting banking operations to centralized systems are done at one branch and stabilized. The same process is replicated across all branches. Fig. 6.3 depicts Pilot Implementation.

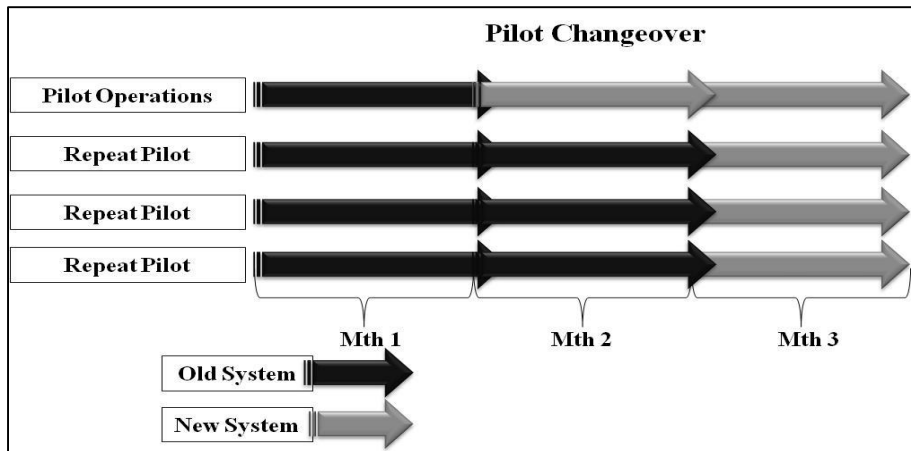


Fig. 6.3: Pilot Changeover

Advantage of pilot implementation is that issues and problems are identified and rectified during pilot run and a stabilized system is implemented thus saving cost and enabling faster implementation.

Parallel Changeover: This is considered the most secure method, time and resource consuming implementation. The new systems is implemented, however the old system also continues to be operational. The output of new system is regularly compared with old system. If results matches over period of time and issues observed with new system are taken care of, the old system is discontinued. Fig. 6.4 shows parallel implementation.

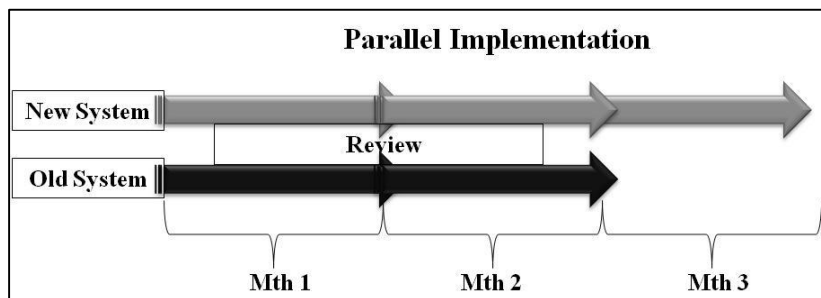


Fig. 6.4: Parallel Changeover

However it is costly and may not be feasible in large and complex systems, since all transactions must be processed twice. Many times users may not be conformable in duplicating the work.

6.3.2 Preparing for implementation

In order to finally deploy or implement the new system in the operating environment, several activities are undertaken. A fully functional as well as documented system is a prerequisite for implementation to begin. Moreover, many other issues like defect removal, maintenances, reengineering may need to be addressed to assure the desirable quality control of the system in operational environment.

The process of ensuring that the information system is operational and then allowing users to take over its operation for use and evaluation is called Systems Implementation. Implementation includes all those activities that take place to convert from the old system to the new. The new system may be totally new, replacing an existing manual or automatic system. Some of the generic key activities involved in system Implementation are:

- Site preparation and hardware installation
- Conversion of data to the new system files;

- Training of end users;
- Completion of user documentation;
- System changeover
- Post implementation review and evaluation

Site preparation and Installation: The hardware required to support the new system is selected prior to the implementation phase. The necessary hardware should be ordered in time to allow for installation and testing of equipment during the implementation phase. An installation checklist should be developed at this time with operating advice from the vendor and system development team. In situations, where people are not experienced in the installation of similar hardware/platform/equipment, adequate time should be planned to allow completion of required activities.

Site Preparation: An appropriate location as required to provide an operating environment for system (temperature, humidity and dust control specifications) has to be prepared in time.

Installation of New Hardware / Software: Site preparation also includes receiving, installing and connecting the hardware and supporting software (like operating systems, middleware etc.). In case the hardware is available need to be commissioned for the new application as per design requirements.

Equipment Checkout: The equipment must be turned on for testing under normal operating conditions. Though the routine 'diagnostic tests' should be run by the vendor, the in-house implementation team should test the equipment functionalities in actual working conditions.

6.3.3 Training

Operational training to users about system must be planned along with implementation and before going live and depending upon the complexity of application. There are three ways to provide training:

1. **Simulated training:** Where a training instance of application is installed along with live environment and users are requested to explore the application. This type of training is generally performed on-site
2. **Classroom training:** This is controlled simulated training conducted in classroom and is facilitated by trainer.
3. **On the job training:** Conducted in live environment with appropriate guidance from experts.

The quality of training received by the personnel involved with the system in various capacities helps or hinders the successful implementation of information system. When a new system is

acquired, which often involves new hardware and software, both users and computer professionals generally need some type of training.

Effectiveness of training can be measured based on changes in number of help desk requests before and after training.

6.3.4 Conversion

Conversion of data is most important activity while implementing new application system or when there is significant change in technology requiring conversion. The conversion activity involves converting data, procedures, documentation from old system to new system. Most important being data conversion.

Data Conversion: The requirement of data conversion depends upon the change. If the new application is replacing manual operation to automated operation it involves:

1. Capturing of data into electronic form
2. Verification of data
3. Uploading into database

In case change is from old system to new system, it involves:

1. Converting electronic data from old format to new format
2. Verification
3. Uploading into new database.

Since data conversion is a type of input, controls on conversions are essential to ensure integrity of data. These controls generally include:

1. **Completeness check:** Using number of records, control totals, batch totals, hash totals. For example verifying number of employees record, checking trial balance before and after conversion etc.
 2. **Accuracy check:** Manual verification or key verification (manual to electronic conversion)
- Unauthorized changes during conversion are one of the sources of frauds.

Procedure Conversion: Changes in application systems may require changes in operating procedures and associated controls. Operating procedures should be carefully completed with sufficient documentation pertaining to operations on how to use the new system. It applies to both computer-operations and functional area operations. Before conversion activities can start, conversion procedures must be defined and personnel involved must be trained to cover input, data files, methods, procedures, output, and internal control.

For example, during manual operation in banking every transaction is verified before being posted to account and then the effect of transaction is reflected in general ledger. However in electronic banking system transactions are flagged with type of transaction and posted to general ledger. Hence verification of transaction is most essential in new system.

System conversion: After on-line and off-line files have been converted and the reliability of the new system has been confirmed for a functional area, daily processing can be shifted from the existing information system to the new one. All transactions initiated after this time are processed on the new system. System development team members should be present to assist and to answer any questions that might develop. Consideration should be given to operating the old system for some more time to permit checking, matching and balancing the total results of both systems.

Scheduling Personnel and Equipment: Scheduling data processing operations of a new information system for the first time is a difficult task for the system manager. As users become familiar with the new system, the job becomes easier to perform and becomes part of the routine work. Schedules should be set up by the system manager in conjunction with departmental managers of operational units serviced by the equipment. The master schedule for next period/month should provide sufficient computer time to handle all required processing.

6.4 Change management process

Application maintenance refers to the process of managing changes in the application and IT triggered or prompted due to changes in processes, regulatory compliances, and strategic changes in business, technology changes and so on. Changes also arise due to issues, problems, incidents faced. In order to handle changes organisation should have defined process. This process generally includes:

1. **Raising change request:** Formal process for requesting change. Anyone can raise the change request with reason for the change, a cost justification analysis, if possible and the expected benefits of the change. An automated process for raising change requests helps in capturing all associated changes and maintaining record of change requests.
2. **Defining requirements:** Defining details of changes required, like functional changes, appearance changes, processing changes. (e.g. change in tax structure may require processing changes, if tax slabs are displayed then appearance changes and so on)
3. **Analysing requirements:** Getting answers to the questions such as: why change is required, when it should be effective, who needs it, where the changes are required,

what programs/modules/function is affected, how the changes will be carried out and so on.

4. **Impact analysis:** What will be impact of changes on processes and other related programs that interface with application that need to be changed or how changes in technology shall affect the processing.
5. **Approval of change:** Changes must be approved by the asset owners, i.e. application owners in case of application change and other stakeholders that might be impacted. Sometimes it is difficult to decide who has appropriate authority to approve change due to impact on multiple processes. To overcome such situations organisations forms a change approval board or committee (CAB) consisting of representatives from multiple business functions.
6. **Prioritizing the change requests:** this is required to resolve the conflict due to multiple change requests from different users.
7. **Carrying out changes:** System analyst shall review the changes and decide appropriate resources to carry out changes. Records of all program changes should be maintained. Library management software may help in automating this process and also maintaining audit trail. The maintenance information usually consists of the programmer ID, time and date of change, project or request number associated with the change, and before and after images of the lines of code that were changed. This also helps in preventing and/or detecting unauthorized changes.
8. **System document maintenance:** All relevant system documentation updating sometimes is neglected area during change management. It is essential to ensure the effective utilization and future maintenance of a system, Documentation requiring revision may consist of program and/or system flowcharts, program narratives, data dictionaries, entity relationship models, data flow diagrams (DFDs), operator run books and end-user procedural manuals. In case of infrastructure changes network diagrams. Data centre block diagrams, electrical and facility diagrams etc. are likely to undergo changes.
9. **Testing the changes:** Changes will be tested as per testing process (Please refer subsection on testing). However for testing changes, following points must be considered:
 - Existing functionalities are not affected by the change
 - System performance is as expected
 - Security vulnerabilities are not introducedThis also includes conducting user acceptance testing and formal sign-off from users/owners. (E-mail, electronic approval in automated system, document etc.)
10. **Releasing changes:** Changes shall be released to production once approved by stakeholders (UAT). Ensure fall-back procedures in place in case operations are affected due to change. Automation of this process shall help management in restricting one person requiring access to production, test and development environment.
11. **Review:** Post implementation/release review may be conducted.

12. **Record maintenance:** Change requests should be maintained in a format that will ensure that all changes associated with primary change requests are considered. This allows the management to easily track the changes to change requests. The process must be formal and maintain record of all approvals and rejections.

For acquired systems vendor may distribute periodic updates, patches or new version of the software. User and systems management should review such changes for impact and appropriateness before implementing.

While reviewing change management IS auditor should ensure that:

- **Access to source program is restricted.**
- **Change requests are approved and record is maintained to trace and track the changes.**
- **Impact assessment is performed before approving changes.**
- **The change request should be documented in standard format covering at the minimum:**
 - **Change specifications, benefit analysis developed and a target date.**
 - **Change form has been reviewed and impact assessment is recorded.**
 - **change request has been approved formally**
- **Verify records of changes for sample changes made and trace end-to-end (from request till closure) confirm that the changes are authorized, approved, and moved to production after UAT.**

Ensure segregation of duties during change management

6.4.1 Emergency Changes

In exceptional situations there may be need to make changes to production to resolve issues in time. This requirement can arise due to one or more reasons like:

- Events/incidents
- Short notice requirement changes (due to external incidents/events: terrorist attacks, natural disasters, etc.)
- Infrastructure failure
- Production issues due to unexpected data conditions

Procedures should focus to ensure that emergency changes can be performed without compromising the integrity of the system. Organisation should have a process for carrying out emergency changes. The process may consist of following steps:

1. Identify need for emergency change (process issue, incident/event etc.)
2. Determine activities involved. Generally it may involve providing all accesses to one person. A special user ID may be created with higher privileges for this purpose and all activities are logged and reviewed.

3. A post-facto change management process must be followed, so as to ensure consistency in documents, source-code library, network diagram etc. as applicable.

The IS auditor has to ensure that emergency changes are handled in a controlled manner.

6.4.2 Implementing Changes into Production

Changes are implemented into production environment once they are approved by the user management (UAT sign-off). The best practices suggest that this implementation should be done by independent team not involved in development or testing of changes. In case of client-server applications or distributed systems, such as point-of-sale systems, the process should be properly documented and implemented over a period of time to ensure:

- Conversion of data
- Training of users
- Support process for changes
- Rollback plan
- All points are updated

6.4.3 Segregation of Duties

Uncontrolled change management has risk associated with unauthorized changes. An unauthorized change occurs due to various reasons:

- Developer has access to production libraries containing programs and data including object code.
- User has not approved change or not aware of the change
- A change procedure has not formally established.
- A change was updated into production without user approval.
- The changes had not been reviewed or tested.
- Developer inserted extra logic for personal benefit (i.e., committed fraud).
- In case of vendor software, changes received were not tested.

In order to control unauthorized changes segregation of duties has to be implemented at organisation level. The typical segregation includes following controls at the minimum:

- Development, test and production environments to be physically separated.
- Developers' team, testing team and production user should not have access to other areas. I.e. developers should not have access to test and production and so on.
- Source code must be maintained by librarian. At least control must be in place to prevent or detect insertion of unauthorized code.
- A separate change control team or release team should be appointed to move source/object code from development to test and from test to production.

It may not be possible for some organisations to implement strict segregation of duties, in such situation appropriate compensating controls should be present to prevent or detect and correct unauthorized changes. Some such situations may arise due to:

1. The developer is also the operator due to small IT department. In this case user management required to ensure proper authorization and monitoring of changes and upgrades made by the programmer.
2. Emergency changes to resolve the issues in production.
3. In case separate release team is not possible, compensating control of enabling user ID of user who moves changes from development to test and/or test to production only after approval and monitoring activities may work as compensating control.
4. Developers should not have write, modify or delete access to production data. Depending on the type of information in production, programmers may not have read-only access to personally identifiable information.

6.4.4 Configuration Management

Configuration management refers to automated process organisations install to maintain all information assets and work-flows required to maintain them. The backend of such system is a data base called a Configuration management Data Base (CMDB); hence sometimes the system is also referred to as CMDB.

Configuration management system helps in maintaining information about system as collection of configuration items (CI). A CI can be a module/function/program or database instance of IT asset associated with a system and referenced with an ID. Workflows around the CMDB consist of workflows for Change Management, configuration management etc. Change management requests must be formally documented and approved by a change control group within CMDB. CMDB then manages the change process via checkpoints, reviews and sign-off procedures that generates audit trails.

Configuration management sometimes may provide procedures throughout the software life cycle (from requirements analysis to maintenance) to identify, define and baseline software items in the system and thus provide a basis for problem management, change management and release management. However though it sounds easy, proper implementation of CMDB is a necessary requirement (which must follow SDLC process for acquired Software).

Software configuration management requires following tasks to be performed:

1. Develop configuration management plan.
2. Baseline application and associated assets.
3. Analyze results of configuration control.

4. Develop monitoring of configuration status.
5. Develop release procedures.
6. Define and implement configuration control activities (such as identification and recording of change requests.)
7. Update the configuration status accounting database.

A configuration management tool supports change and release management by supporting following activities:

1. Identification of items affected by a proposed change
2. Help in impact assessment by providing information
3. Recording configuration items affected by changes
4. Implementation of changes as per authorization
5. Registering of configuration item changes when authorized changes and releases are implemented
6. Recording of original configuration to enable rollback if an implemented change fails
7. Preparing a release to avoid human errors and resource costs

6.5 Summary

Implementation of new application follows a similar process required for change and release management. Although change management primarily refers to the maintenance activity in SDLC phases, a large change may have to be initiated to implement change by implementing a SDLC project. In other words a change management process by itself may be considered as a mini-SDLC project. Controls required during implementation and change management are similar and IS auditor has to understand the process of controlling unauthorized changes in software whether it is being developed or configured or maintained.

6.6 Questions

1. Which of the following process during implementation of banking software is most critical to minimize the risk associated with frauds?

- A. Training and awareness
- B. Data conversion
- C. Hardware configuration
- D. Procedure conversion

2. Which of the following is a major concern for IS auditor while auditing implementation phase of SDLC project?

- A. Requirement of resilient infrastructure are captured during implementation
- B. Hardware acquisition has been delayed due delay in procurement process

- C. Organization has contracted multiple network service providers for connectivity
 - D. Organization has decided to use existing site for implementing new solution
3. Who among the following should be primarily responsible for approving changes to application system?
- A. Head of IT department
 - B. Business process users
 - C. Application administrator
 - D. Affected business stakeholders
4. Which of the following method is most useful in implementing an ERP application at multiple locations of same organization?
- A. Parallel
 - B. Pilot
 - C. Phased
 - D. Cut off
5. An organization has developed a web based application for the use of internal users to be hosted on intranet. Before finalizing and making it live it was decided to make it available to users for providing feedback. This is an example of:
- A. Internal audit
 - B. Alfa testing
 - C. Beta testing
 - D. User training
6. A major concern associated with using sanitized old production data for testing new application is that:
- A. User may not provide sign off.
 - B. Production data may be leaked.
 - C. Integration testing cannot be performed.
 - D. All conditions cannot be tested.

6.7 Answers and Explanations

1. B. Fraudster manipulated the data being entered into system to commit fraud by exploiting vulnerabilities in the process. Hence, data conversion process must be monitored more closely as compared to other processes.

2. A. Requirement of resilient infrastructure must have been considered during requirement phase, since it affects software development as well as procuring redundant infrastructure.

Module 5

Capturing this requirement during implementation phase will have major impact on benefit realization. Other options are not major concerns, but in fact C is an advantage.

3. D. Changes to application must be primarily approved by application owner, who is also business process owner. However change in application may also require change in dependent systems hence it is best to get approval from all. To facilitate this process generally change approval board is formed that represents all stake holders.

4. B. Pilot is method is most useful, as organization can implement application at one location and run for few days. This will help in finalizing the basic configuration and also learn from post implementation issues. Once successful the same can be replicated at other locations.

5. C. Beta testing is making product available for user for feedback before launching.

6. D. Production data generally may not cover all paths the data can take and hence system cannot be tested for all possible cases. Data leakage is not a major concern since data is sanitized. Options A and C are not concerns.