UNIT 156 – UPSC - Systems Development - Overview of systems and design, System development management life-cycle

Systems development is the procedure of defining, designing, testing, and implementing a new software application or program. It comprises of the internal development of customized systems, the establishment of database systems, or the attainment of third party developed software. In this system, written standards and techniques must monitor all information systems processing functions. The management of company must describe and execute standards and embrace suitable system development life cycle practise that manage the process of developing, acquiring, implementing, and maintaining computerized information systems and associated technology.

System development methodologies are promoted in order to improve the management and control of the software development process, structuring and simplifying the procedure, and standardizing the development process and product by stipulating actions to be done and methods to be used. It is often implicitly presumed that the use of a system development methodology will increase system development output and excellence.

System Development Management Life-cycle

It is maintained in management studies that effectual way to protect information and information systems is to incorporate security into every step of the system development process, from the initiation of a project to develop a system to its disposition. The manifold process that begins with the initiation, analysis, design, and implementation, and continues through the maintenance and disposal of the system, is called the System Development Life Cycle (SDLC). Walsham (1993) stated that system development life cycle is an approach to developing an information system or software product that is characterized by a linear sequence of steps that progress from start to finish without revisiting any previous step. It is one of the oldest systems development models and is

commonly used (Walsham, 1993). According to Dennis, Wixom, and Tegarden(2009) "the systems development life cycle is the process of understanding how an information system (IS) can support business needs by designing a system, building it, and delivering it to users" .

The SDLC model is basically a project management device that is used to plan, execute, and control systems development projects (Whitten and Bentley, 1998). System development life cycles are usually deliberated in terms of the conventional development using the waterfall model or the prototyping development spiral model. Major objectives of systems development lifecycle are to ensure that high quality systems are delivered, provide strong management controls over the projects, and maximize the productivity of the systems staff. In order to fulfil these objectives, the systems development lifecycle has many specific requirements that include being able to support projects and systems of various scopes and types, supporting all of the technical activities, supporting all of the management activities, being highly usable, and providing guidance on how to install it.

Phases of System Development

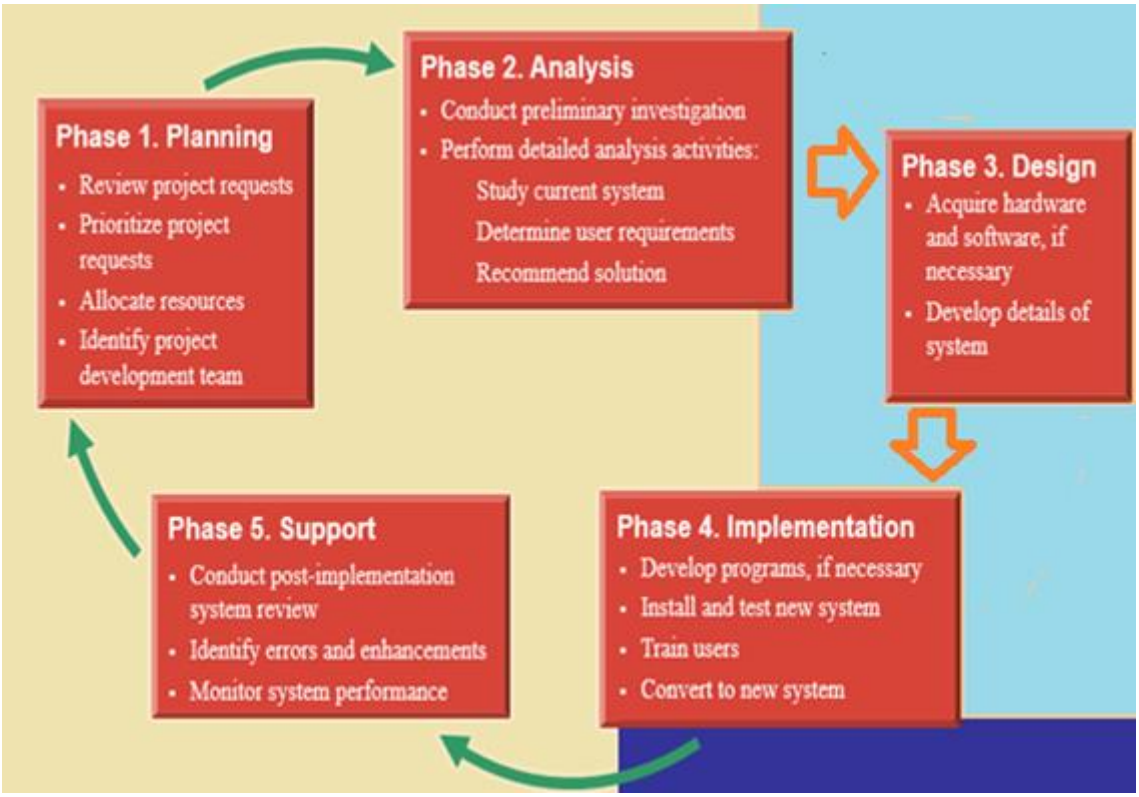
A system development project comprises of numerous phases, such as feasibility analysis, requirements analysis, software design, software coding, testing and debugging, installation and maintenance.

1. A feasibility study is employed to decide whether a project should proceed. This will include an initial project plan and budget estimates for future stages of the project. In the example of the development of a central ordering system, a feasibility study would look at how a new central ordering system might be received by the various departments and how costly the new system would be relative to improving each of these individual systems.
2. Requirement analysis recognises the requirements for the system. This includes a detailed analysis of the specific problem being addressed or the expectations of a particular system. It can be said that analysis will coherent what the system is supposed to do. For the central ordering system, the analysis would cautiously scrutinize existing ordering systems and how to use the best aspects of those systems, while taking advantage of the potential benefits of more centralized systems.
3. The design phase consist of determining what programs are required and how they are going to interact, how each individual program is going to work, what the software interface is going to look like and what data will be required. System design may use tools such as flowcharts and pseudo-code to develop the specific logic of the system. For this central ordering system, the design phase would lay out the comprehensive steps of how orders would take place and who in the organization would be involved at each step.
4. Implementation stage include the design which is to be translated into code. This requires choosing the most suitable programming language and writing the actual code needed to make the design work. In this stage, the central ordering system is essentially coded using a particular programming language. This would also include developing a user interface that the various departments are able to use efficiently.
5. Testing and debugging stage encompasses testing individual modules of the system as well as the system as a whole. This includes making sure the system actually does what is expected and that it runs on intended platforms. Testing during the early stages of a

project may involve using a prototype, which meets some of the very basic requirements of the system but lacks many of the details. Testing of the central ordering system could take place in one department or use only a few key individuals. That makes it possible to recognise needed improvements before execution in all departments.

6. In Installation phase, the system is implemented so that it becomes part of the workflows of the organization. Some training may be needed to make sure employees get happy with using the system. At this stage, the central ordering system is installed in all departments, replacing the older system.
7. All systems need some types of maintenance. This may consist of minor updates to the system or more drastic changes due to unexpected circumstances. As the organization and its departments evolve, the ordering process may require some modifications. This makes it possible to get the most out of a new centralized system.

phases of the system development cycle



Whitten and Bentley (1998) recommended following categories of system development project lifecycle:

1. Planning
2. Analysis
3. Design
4. Implementation
5. Support

There are many different SDLC models and methodologies, but each usually consists of a series of defined steps such as Fountain, Spiral, rapid prototyping, for any SDLC model that is used, information security must be integrated into the SDLC to ensure appropriate protection for the information that the system will transmit, process, and store.

System development life-cycle models (Source: Conrick, 2006))

Fountain Model	Recognizes that there is considerable overlap of activities throughout the development cycle.
Spiral model	Emphasis the need to go back and reiterate earlier stages like a series of short water fall cycle, each producing an early prototype representing the part of entire cycle.
Build and fix model	Write some programming code, keeps modifying it until the customer is happy. Without planning this is very open ended and risky.
Rapid prototyping model	Emphasis is on creating a prototype that look and act like the desired product in order to test its usefulness. Once the prototype is approved, it is discarded and real software is written.
Incremental model	Divides the product into builds, where sections of the projects are created and tested separately.
Synchronize and stabilise model	Combines the advantages of spiral models with technology of overseeing and managing source code. This method allows many teams to work efficiently in parallel. It was defined by David Yoffe of Harvard University and Michael Cusumano of Massachusetts institute of technology who studied Microsoft corporation developed internet explorer and how the Netscape communication corporation developed communicator finding common thread In the ways the two companies worked.

Waterfall Model

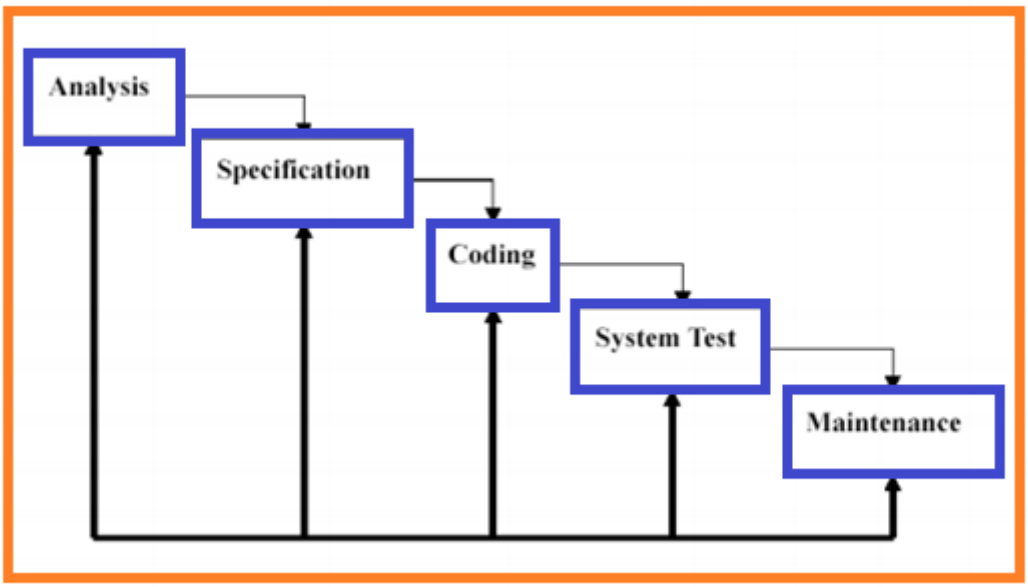
The Waterfall Model signifies a traditional type of system development project lifecycle. It builds upon the basic steps associated with system development project lifecycle and uses a top-down development cycle in completing the system.

Walsham (1993) outlined the steps in the Waterfall Model which are as under:

1. A preliminary evaluation of the existing system is conducted and deficiencies are then identified. This can be done by interviewing users of the system and consulting with support personnel.
2. The new system requirements are defined. In particular, the deficiencies in the existing system must be addressed with specific proposals for improvement.
3. The proposed system is designed. Plans are developed and delineated concerning the physical construction, hardware, operating systems, programming, communications, and security issues.
4. The new system is developed and the new components and programs are obtained and installed.
5. Users of the system are then trained in its use, and all aspects of performance are tested. If necessary, adjustments must be made at this stage.

6. The system is put into use. This can be done in various ways. The new system can be phased in, according to application or location, and the old system is gradually replaced. In some cases, it may be more cost-effective to shut down the old system and implement the new system all at once.
7. Once the new system is up and running for a while, it should be exhaustively evaluated. Maintenance must be kept up rigorously at all times.
8. Users of the system should be kept up-to-date concerning the latest modifications and procedures.
   On the basis of the Waterfall Model, if system developers find problems associated with a step, an effort is made to go back to the previous step or the specific step in which the problem occurred, and fix the problem by completing the step once more.

the model's development schedule



Fountain model: The Fountain model is a logical enhancement to the Waterfall model. This model allows for the advancement from various stages of software development regardless of whether or not enough tasks have been completed to reach it.

Prototyping Model: The prototyping paradigm starts with collecting the requirements. Developer and customer meet and define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory. The prototype is appraised by the customer/user and used to improve requirements for the software to be developed. Iteration occurs as the prototype is tuned to satisfy the needs of the customer, while at the same time enabling the developer to better understand what needs to be done.

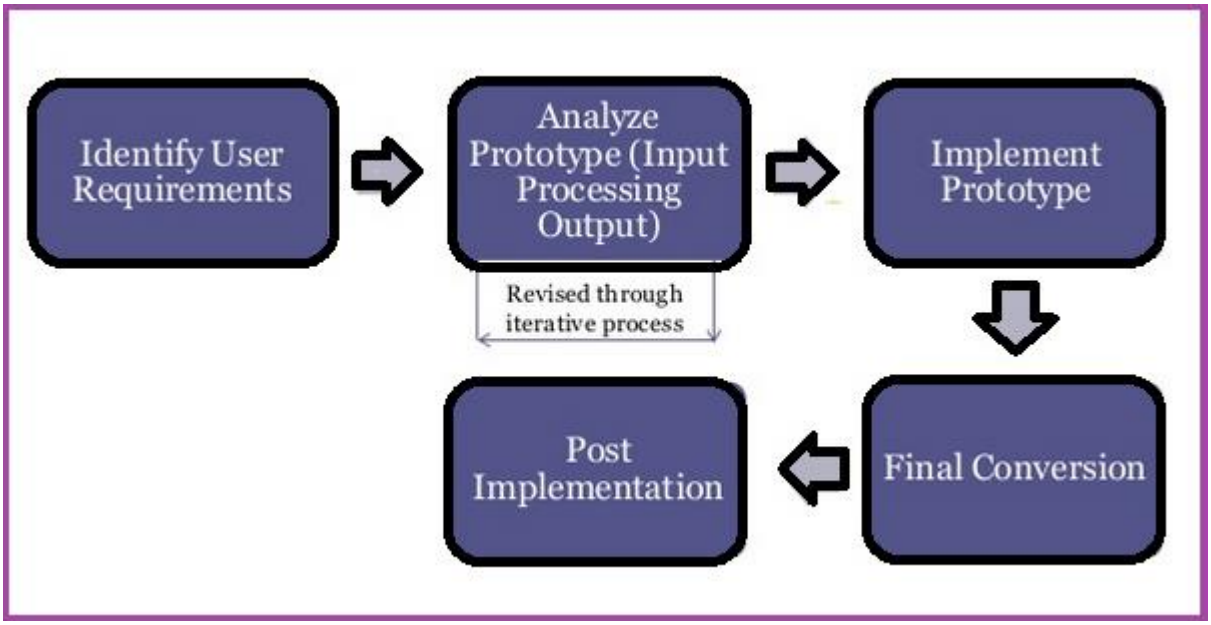
Major Advantages of this Model include

i. When prototype is presented to the user, he gets a proper clearness and functionality of the software and he can suggest changes and modifications.

ii. It determines the concept to prospective investors to get funding for project and thus gives clear view of how the software will respond.
iii. It decreases risk of failure, as potential risks can be recognized early and alleviation steps can be taken thus effective elimination of the potential causes is possible.
iv. Iteration between development team and client provides a very good and conductive environment during project. Both the developer side and customer side are coordinated.
v. Time required to complete the project after getting final the SRS reduces, since the developer has a better idea about how he should approach the project.

Main drawbacks of this model are that Prototyping is typically done at the cost of the developer. So it should be done using nominal resources. It can be done using Rapid Application Development tools. Sometimes the start-up cost of building the development team, focused on making the prototype is high. Once developers get proper requirements from client after showing prototype model, it may be useless. It is a slow process and too much involvement of client is not always favoured by the creator.

Figure: different phases of Prototyping model



Uses of prototyping:

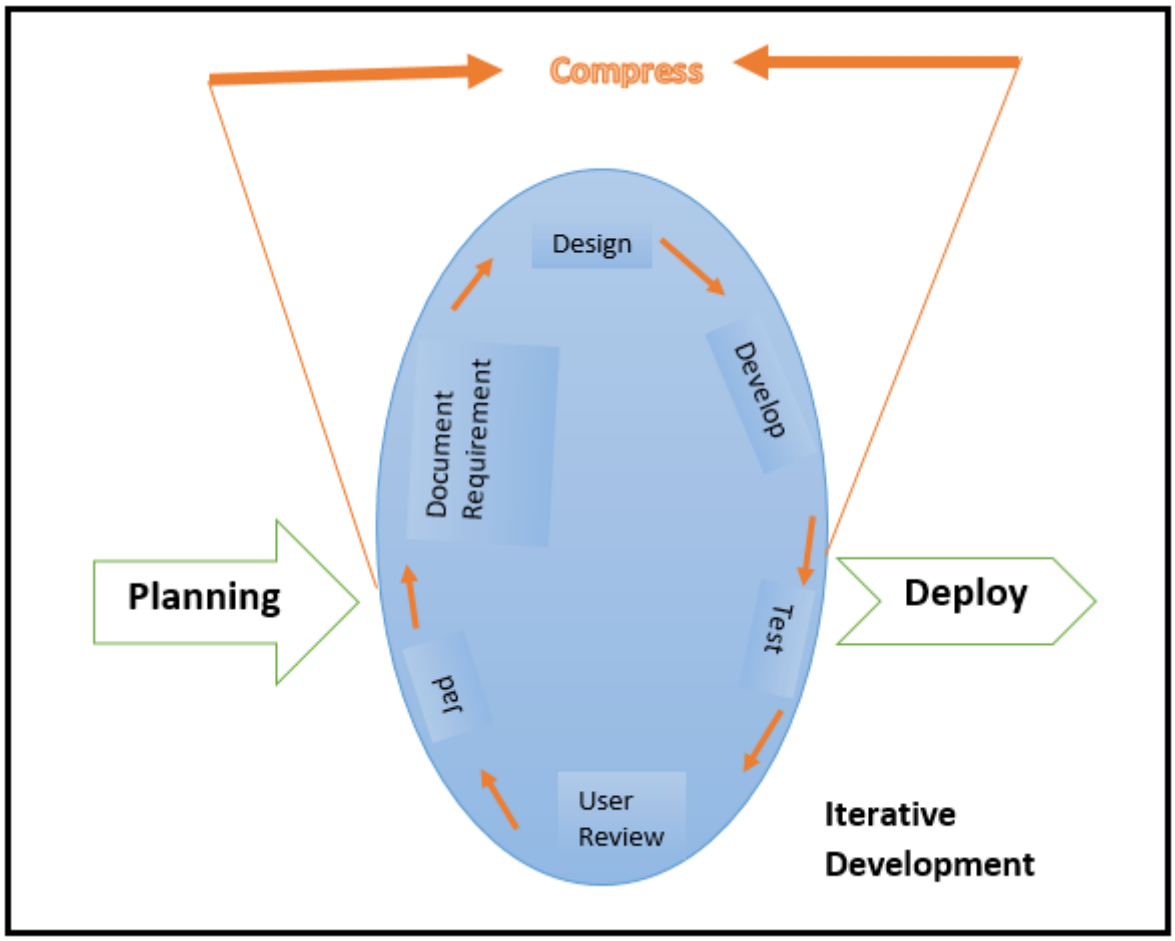
i. Verifying user needs
ii. Verifying that design = specifications
iii. Selecting the “best” design
iv. Developing a conceptual understanding of novel situations
v. Testing a design under varying environments
vi. Demonstrating a new product to upper management
vii. Implementing a new system in the user environment quickly

Rapid Application Development

This model is based on prototyping and iterative development with no detailed planning involved. The process of writing the software itself involves the planning required for developing the product. Rapid Application development focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using iterative concept, reuse of the existing prototypes (components), continuous integration and rapid delivery. There are three main phases to Rapid Application Development:

i. Requirements planning
ii. RAD design workshop
iii. Implementation

RAD Model



RAD is used when the team includes programmers and analysts who are experienced with it, there are pressing reasons for speeding up application development, the project involves a novel ecommerce application and needs quick results and users are sophisticated and highly engaged with the goals of the company.

Spiral Model: The spiral model was developed by Barry Boehm in 1988 (Boehm, 1986). This model is developed to Spiral Model to address the inadequacies of the Waterfall Model. Boehm stated that "the major distinguishing feature of the Spiral Model is that it creates a risk-driven approach to the software process rather than a primarily document-driven or code-driven process. It incorporates many of the strengths of other models and resolves many of their difficulties" (1988). A Spiral Model the first model to elucidate why the iteration matters. Spiral model is an

evolutionary software process model which is a grouping of an iterative nature of prototyping and controlled and systematic aspects of traditional waterfall model. As originally proposed, the iterations were usually 6 months to 2 years long. Each phase starts with a design goal and ends with the client reviewing the progress. Analysis and engineering efforts are done at each phase of the project. The spiral model consists of four phases:

i. Planning
ii. Risk Analysis
iii. Engineering
iv. Evaluation

Major benefits of this model include:

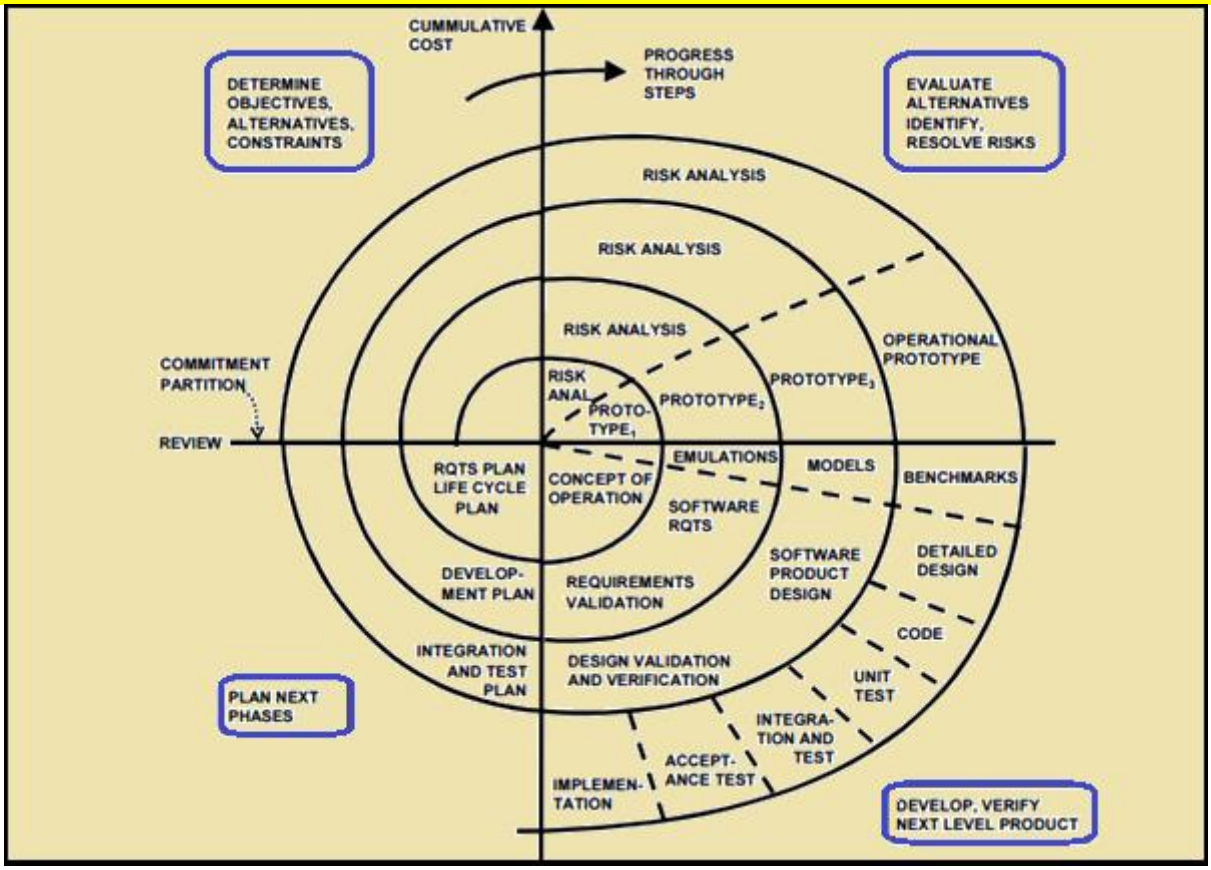
i. Changing requirements can be accommodated.
ii. Allows for extensive use of prototypes.
iii. Requirements can be captured more accurately.
iv. Users see the system early.
v. Development can be divided in to smaller parts and more risky parts can be developed earlier which helps better risk management.

Main drawbacks of this model are as under:

1. Management is more complex.
2. Conclusion of project may not be recognized early.
3. Not suitable for small or low risk projects (expensive for small projects).
4. Process is difficult
5. Spiral may go indeterminately.
6. Large numbers of intermediate stages require unnecessary documentation.

The spiral model is normally used in huge projects. For example, the military had adopted the spiral model for its Future Combat Systems program. The spiral model may suit small software applications.

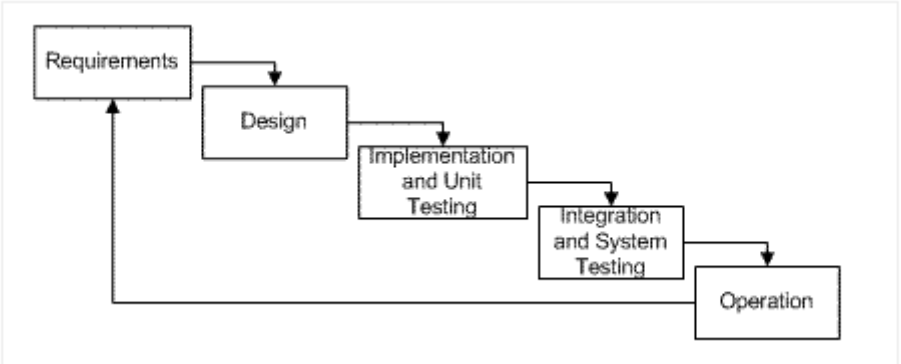
Phases of spiral model



Incremental model: Incremental model is a technique of software development in which the model is analysed, designed, tested, and implemented incrementally. Some benefits of this model are that it handles large projects, it has the functionality of the water fall and the prototyping model. It is easier to manage the project as it is broken down into smaller pieces, changes can be done through the development stages and errors are easy to be identified.

Disadvantages of this model are that when remedying a problem in a functional unit, then all the functional units will have to be corrected thus taking a lot of time. It needs good planning and designing.

Increment model of SDLC



There are numerous benefits of integrating security into the system development life cycle that are as under:

1. Early documentation and alleviation of security vulnerabilities and problems with the configuration of systems, resulting in lower costs to implement security controls and mitigation of vulnerabilities;
2. Awareness of potential engineering challenges caused by mandatory security controls.
3. Identification of shared security services and reuse of security strategies and tools that will reduce development costs and improve the system's security posture through the application of proven methods and techniques.
4. Assistance of informed executive decision making through the application of a comprehensive risk management process in a timely manner.
5. Documentation of important security decisions made during the development process to inform management about security considerations during all phases of development.
6. Enhanced organization and customer confidence to facilitate adoption and use of systems, and improved confidence in the continued investment in government systems.
7. Improved systems interoperability and integration that would be difficult to achieve if security is considered separately at various system levels.

Strengths of System Development Life Cycle

1. Methodologies incorporating this approach have been well tried and tested.
2. This cycle divides development into distinct phases.
3. Makes tasks more manageable.
4. It Offers opportunity for more control over development process.
5. It Provides standards for documentation.
6. It is better than trial and error.

Weaknesses of System Development Life Cycle

1. It fails to realise the "big picture" of strategic management.
2. It is too inflexible to cope with changing requirements.
3. It stresses on "hard" thinking (which is often reflected in documentation that is too technical).
4. It unable to capture true needs of users.

To summarize, the systems development life cycle is a theoretical model which is used in project management. It explained various phases involved in an information system development project, from an initial feasibility study through maintenance of the completed application. SDLC system life cycle is a step by step systematic approach from planning to testing and deployment of the project. There are some rudimentary phases that are firmly followed in the order as analysis, designing, coding, testing and implementation. Different SDLC models are used to develop numerous projects.