SYSTEM METHODOLOGY:

The need for a Systems Methodology was perceived in the second half of the 20th Century, to show how and why systems engineering worked and was so effective.

Systems engineering was seen as a powerful method for solving complex problems, particularly in respect of major projects in the space program and the defense industry. These early successes were based on systems science and system methods that were themselves relatively new, having emerged in response to perceived limitations in the hard sciences, notably their inability to explain life and the counterintuitive behavior of wholes, or gestalt.

Systems methods concerned themselves with the synthesis of whole open systems and with emergence, the latter caused by interactions between the parts within a system. Such methods, although effective, seemed alien (unknown) and arcane (deep) to engineers concerned with methods, based on Cartesian reduction, for creating tangible and material end products.

The need was perceived for a systems methodology, one that was accessible to engineers along with other disciplines from the applied and life sciences, so that the whole process, from addressing the problem to creating the optimum solution, could be understood and pursued.

System Methodology:

"Has mankind evolved to a point that there exists, or that with creative additions and re-combinations of modest proportions, there can be shown to be available, a common systems methodology, in terms of which we can conceive of, plan, design, construct, and use systems (procedures, machines, teams of people) of any arbitrary type in the service of mankind, and with low rates of failure?"

Systems methodology looks like:

By definition, if it is a methodology, it is context free - it can be used for any kind of problem and to create any kind of systems solution - provided one exists. (In the real world, every problem need not have a solution).

OBJECTIVES OF THE METHODOLOGY:

- ✓ Systems methodology shows how they should be work
- ✓ It examines how various components work together to produce a particular outcome

- ✓ By diagramming the linkages between each system activity. Systems methodology makes it easier to understand the relationships among various activities and the impact of each on the other.
- ✓ It shows the processes as part of larger systems whose objectives are to serve a specific client need.
- ✓ Systems methodology is valuable when an overall picture is needed.
- ✓ It shows direct and support services interact, where critical inputs come from, and how products or services are expected to meet the needs in the community.
- ✓ When the teams do not know where to start, Systems methodology can help in locating problems areas or in analyzing the problem by showing the various parts of the systems.
- ✓ Systems methodology can also reveal data collection needs : indicators of inputs, process , and outcomes
- ✓ It can be helpful in monitoring performance.

Aspects of the systems methodology:

Some of the most important of aspects of the Systems methodology are:

- ➢ Human dimension
- ➢ Logic dimension
- ➢ Knowledge dimension
- ➢ Time dimension

TIME DIMENSION:

The systems methodology contains processes and tasks at its heart; some processes necessarily involve sequence. Without knowing the problem first, e.g., it is not sensible to predicate a remedy .so, for some aspects there is a natural, inescapable, logical sequence, which will take time to perform.

Complexity appears to emanate (originate) from at least three aspects: connectivity, variety and tangling, or the degree of interweaving of strands and parts of any systems. The greater the complexity of an issue, the more time may be needed to unravel the knot, to identify and characterize the various “open systems, and to understand how their various ramifications interact and interdependent.”

LOGIC DIMENSION:

The systems methodology is necessarily logical and rational where both terms indicate, inter, an absence of cultural bias.

The thinking /behavior aspect is of concern, epistemologically—the 'do you know what you think you know,' and 'how do you know what you know' considerations. Many engineers' education and training. Moreover, psychologists, anthropologists, and others are not in total agreement themselves about human behavior, society, psyche, etc.

That which applies to the remedial system in the way of psychology, anthropology, and others soft factors, applies equally to the systems methodology in operation, where it is an undoubted complex system in its own right. The systems methodology is comprised of methods, tools and processes being used and conducted by individuals, teams and teams of teams , all with behavioral characteristics , all operating in a dynamic, interactive creative cauldron.

With such a minefield, to talk of logic is, perhaps, tantamount to treading on a mine. The approach taken in the systems methodology, however, has been to limit the extent of systems methodology such that its content is logical.

HUMAN DIMENSION:

Social systems become progressively more complex as we diversify activities : create new generations , business and industries : bring new technologies online , enjoy freedom to evolve , become ever-more materialistic ; as populations continue to increase , as the hunger for power and energy continues to grow ; and as the waste products of lifestyles accumulate to pollute.

World has been made smaller by enhanced communication, by easy speed of travel, by improved infrastructures etc. this has had the effect of coupling many open systems that were previously only on the loosest indirect contact. Close coupling increases the rate of interchange between systems, causing their behavior to become more dynamic, even chaotic.

As many complex systems dynamically interact, change, and, evolve, issues and problems arise. Many of these are familiar to us; even if there seems to be no way of addressing them. Meeting the global demand for energy; preventing biosphere becoming even more polluted; stemming the observed diminution of species diversity; accommodating the burgeoning human population of the planet; and so on.

The systems methodology is necessarily founded in systems science, in that its operations recognizes systems, works with systems, configures and re-configures systems, recognizes dysfunction in systems, and finds provable ways to 'repair' those dysfunctions.

To be credible, the systems methodology 'adopts' the scientific method, i.e., the scientific method is 'built-in.' the systems methodology incorporates the four steps of the scientific method , which are generally presented as:

- Observe and describe a phenomenon or group of phenomena.
- Formulate one or more hypotheses to explain the phenomena.
- Use of the hypotheses to predict the existence of other phenomena, or to predict quantitatively the results of new observations.

Perform experimental tests of the predictions by several independent experimenters and properly performed experiments.

SOFTWARE LIFECYCLE MODELS:

A software lifecycle model is a standardized format for planning, organizing and running a new development project.

A software lifecycle model is a description of the sequence of activities carried out in an software engineering project, and the relative order of these activities.

It provides a fixed generic framework that can be tailored to a specific project. Project specific parameters will include:

- Size (person-years)
- Budget
- Duration

Project plan = lifecycle model + project parameters

Software Development Life Cycle is a process used by software industry to design, develop and test high quality software's. It aims to produce high quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

A typical Software Development life cycle consists of the following stages:

Planning and Requirement Analysis: Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from

the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational, and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Defining Requirements: Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through '**SRS**' – **S**oftware **R**equirement **S**pecification document which consists of all the product requirements to be designed and developed during the project life cycle.

Designing the product architecture: SRS is the reference for product architects to come out with the best architecture for the product to be developed; based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a **DDS** - **D**esign **D**ocument **S**pecification. This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity , budget and time constraints , the best design approach is selected for the product.

Building or Developing the Product: In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle. Developers have to follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers etc are used to generate the code. Different high level programming languages are used for coding. The programming language is chosen with respect to the type of software being developed.

Testing the Product: This stage is usually a subset of all the stages as in the modern SDLC models; the testing activities are mostly involved in all the stages of SDLC. However this stage refers to the testing only stage of the product where products defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Deployment in the Market and Maintenance: Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometime product deployment happens in stages as per the organizations' business strategy. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

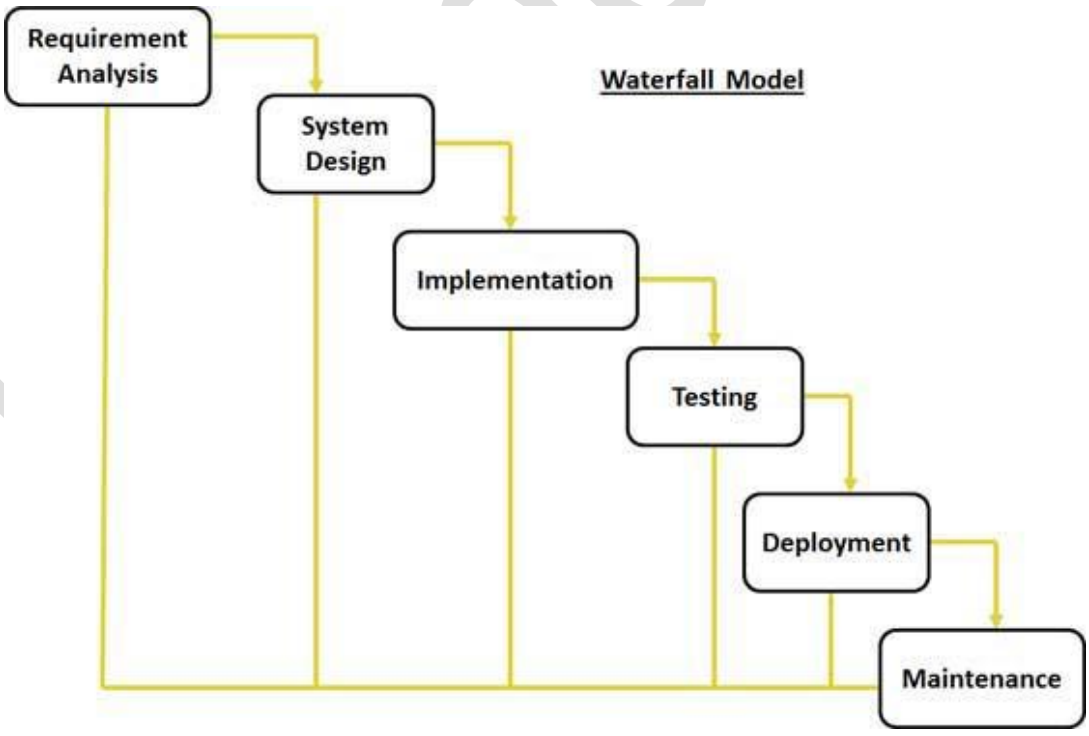
There are various kinds of lifecycle models to choose from, e.g.:

- Waterfall Model
- Iterative Model
- Spiral Model

The other related methodologies are Agile Model, RAD Model – Rapid Application Development and Prototyping Models.

WATERFALL MODEL:

The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. Following is a diagrammatic representation of different phases of waterfall model.



The sequential phases in Waterfall model are:

- **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
- **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system:** Once the functional and non functional testing is done, the product is deployed in the customer environment or released into the market.
- **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

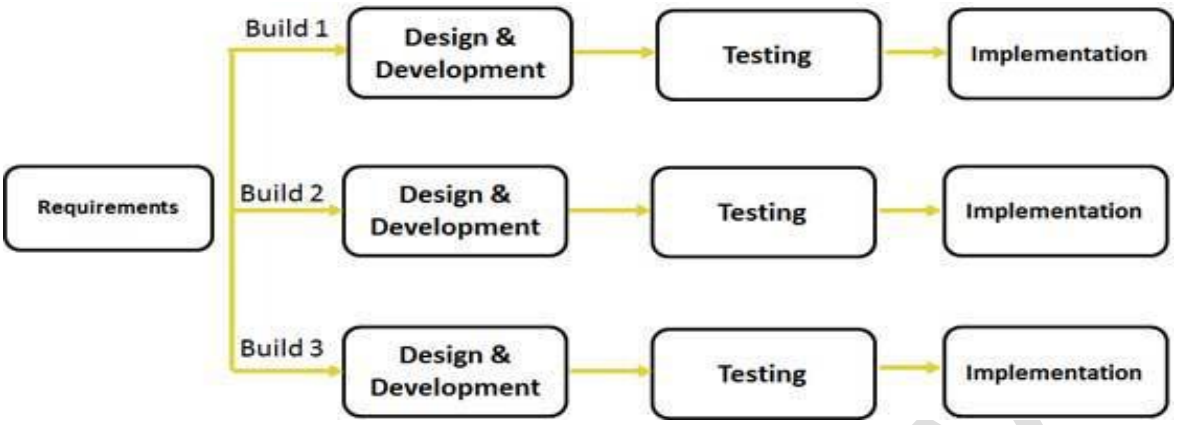
ITERATIVE MODEL:

In Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model.

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

Following is the pictorial representation of Iterative and Incremental model:



SPIRAL MODEL:

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model.

The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

Identification:

This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase.

This also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral the product is deployed in the identified market.

Design:

Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and final design in the subsequent spirals.

Construct or Build:

Construct phase refers to production of the actual software product at every spiral. In the baseline spiral when the product is just thought of and the design is being developed a **POC** (Proof of Concept) is developed in this phase to get customer feedback.

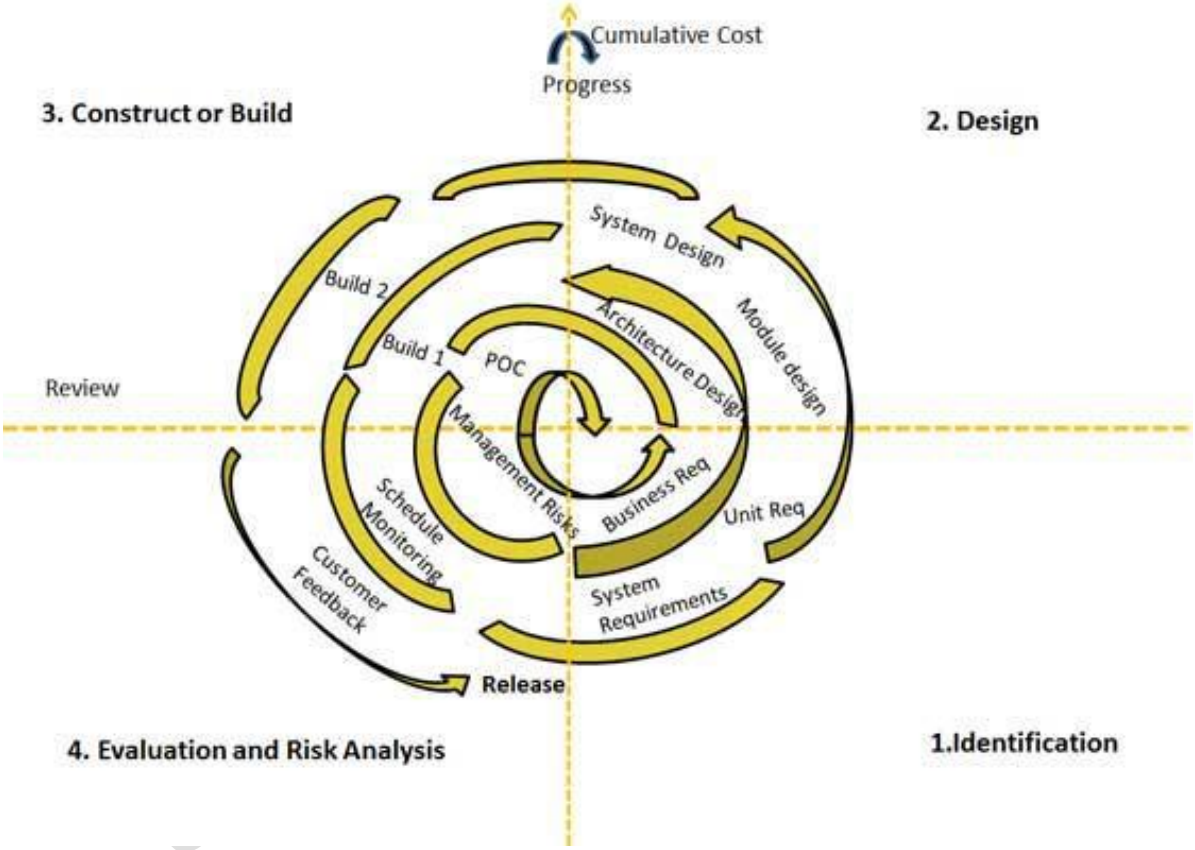
Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to customer for feedback.

Evaluation and Risk Analysis:

Risk Analysis includes identifying, estimating, and monitoring technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

Based on the customer evaluation, software development process enters into the next iteration and subsequently follows the linear approach to implement the feedback suggested by the customer. The process of iterations along the spiral continues throughout the life of the software.

Following is a diagrammatic representation of spiral model listing the activities in each phase:



VERIFICATION AND VALIDATION:

During and after the implementation process, the program being developed must checked to ensure that it meets its specifications and delivers the functionality expected by the people paying for the software. Verification and activities take place at each stage of the software process. V &V starts with requirements reviews and continues through design reviews and code inspections to product testing.

As per the Boehm, **Validation** is: "Are we building the right product." **Verification** is: "Are we building the product right."

- ✓ **Verification** is the process of determining whether or not the products of a given phase of software development fulfill the specifications established during the previous phase
- ✓ **Verification** involves checking of functional and non-functional requirements to ensure that the software confirms to its specification.
- ✓ **Validation** is the process of evaluating software at the end of the software development to ensure compliance with the software requirements. Clearly, for high reliability we need to perform both activities. Together they are often called V&V activities.
- ✓ **Validation** is the process of determine whether a fully developed system confirms to its requirements specifications
- ✓ **Validation** is an analysis process that is done after checking the confirms of the system to its specifications.

The goal of verification process is to asses and improves the quality of the work products generated during development and modifications of software.

The ultimate goal of the verification and validation process is to establish confidence that the software system is 'fit for purpose'. This means that the system: must be good enough for its intended use. The level of required confidence depends on the system's purpose, the expectations of the system users and the current marketing environment for the system.

GOALS AND REQUIREMENTS OF VERIFICATION:

Everything must be verified: In principle, all design processes and all of these processes must be verified.

In fact, once we have tested a system for proper behave our, we should check whether our tests were executed properly themselves. Verifying the validity of experiments is standard practice in established scientific disciplines.

The results of verification may not be binary: It cannot be stated that a piece of software is absolutely error free, but an approximation of ideal correctness is often considered satisfactory and may in some way be certificated. It is common to hear or read sentences such as "the new release of this product corrects several errors."

Verification may be objective or subjective: some instances of verification may be the result of an objective activity, such as performing a test by supplying some input data to the

system and checking the output, or measuring the response time of an interactive system to given stimuli.

Even implicit qualities must be verified: the desired software qualities should be stated explicitly in the requirements specification document. Some requirements, however, may have been left out, either because they are implicit or because they were forgotten.

Verification and Validation process approaches:

1) **Software testing** involves running an implementation of the software with test data. You examine the outputs of the software and its operational behavior to check that it is performing as required; testing is a dynamic technique of verification and validation.

2) **Software inspections** or peer reviews analyze check system representations such as the requirements document, design diagrams and the programs and the program source code. You cause inspections at all stages of the process. Inspections may be supplemented by some automatic analysis of the source code. You can use inspections at all stages of the process. Inspections may be supplemented by some automatic analysis of source text of a system or associated documents.