Table of Contents

History of Software

- The first higher level programming language: “Plankalkuel” developed by Konrad Zuse in 1945 A.D.
- Other language developments: 1950 onwards, FORTRAN, ALGOL, LISP, COBOL, SIMULA etc.
- First programmer: lady Ada Lovelace

Definition of Software Engineering

- building of software systems
- application of engineering discipline to software
- IEEE defines – “Application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software
- “Multi-person construction of multi-version software”

Evolution of SE

- Started in 1960
- Mathematical tools and theories have not been well developed yet.
- SE is based mainly on judgment & experience.
- Programming activity is the first stage of SE, where programming is defined as “the sequence of instructions put together”. One person used to write the whole program. There were misinterpretations of user requirements by programmers.
- The first ever documented software is OS 360 operating system for IBM 360. The term “Software Engineering” was invented at that time. Also introduced the term “*Software Crisis.*” A lot of new challenges came then after e.g. cost, people, project, delivery, compatibility etc.
- Brooks, in 1987, suggested there are two kinds of challenges in SW development: -
    (1) **Essential** : Requiring a lot of intellectual effort, creativity and time
    (2) **Accidental** : To do with current tools & techniques e.g. syntactic problems
- He also said that there was no silver bullet in the SE approach. And he gave answer to the term "Software Crisis" saying that it was challenge that should be taken in a positive way.
- Other condition to evolution of SE is "Cost-to-Performance ratio".

Evolution of Object Oriented (OO) concept

- Around 1990, Air Traffic Navigation System was analyzed by two different teams - DFD team and ER team. The first team went of analyzing the system according to the requirements whereas the second team concentrated in the concept and hence focused in identifying new real world entities and raised a lot of practical concerns to the problem. Hence the DFD team was awarded the project and the ER team was declared as the troublemaker.
- The similar incident happened also with the second project that was initiated by the government. Coad & Yourdon, who were watching both the events were attracted by the new findings of the ER team, started to conceptualize the idea and gave the name “Object Oriented.”
- But again, the OO concept is not the "Silver” bullet.

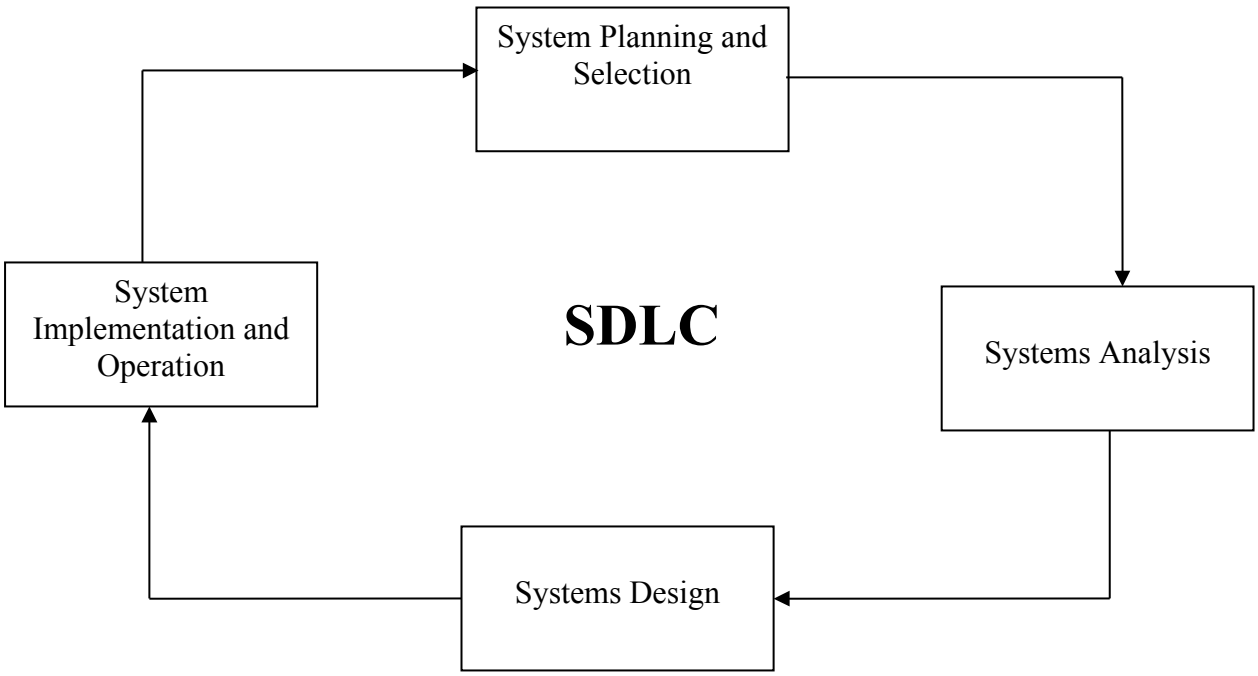
SDLC Stages

- Requirement analysis and specification
- System design and Specification
- Coding and Module testing
- Interaction and System Testing
- Delivery and Maintenance

OO Process Cycle

- Business Modeling
- Domain Modeling
- Requirements Gathering
- Analysis
- Design
- Coding & Testing
- Deployment
- Change management
- Project management

Software Development Life Cycle



- SDLC is central to the development of an efficient information system
- The steps may vary in each organization depending on its goals and type of systems like- Transaction Processing System, Management Information System, Decision Support System, Expert System etc.

Phase 1: System Planning and Selection

The major activities are -

- Identifying the need for a new or enhanced system, in which the feasibility study is also included.
- Investigating the system and determining the proposed system's scope.

Phase 2: Systems Analysis

- Analyst studies the organizations' current procedures and information systems
- The different sub-phases include:-
- Determining the requirements of the system - by careful study of any current existing systems; or according to user's/actor's interrelationships.
- Generating alternative initial designs to match the requirements and compare these alternatives to determine the appropriate solution

Phase 3: Systems Design

- Convert the description of recommended alternative solution into logical and then physical system specifications.
- Design all aspects of the system from input and output screens to reports, databases, and computer processes.
- Logical design concentrates on the business aspects and hence could be implemented or any hardware & system software
- In physical design, logical design is turned into physical (technical) specifications
- Analysts decide the programming language, database systems and file structures, hardware platform, OS, networking environment etc.

Phase 4: Systems Implementation and Operation

- Turn system specifications into working system; it is tested and put into use
- Implementation includes coding, testing and installation.
- Initial user support, documentation, training & orientation are also done.
- During operation, system maintenance is also done including change management

Generic View of Software Engineering:

- Engineering is Analysis, Design, Construction, Verification and Management of technical (and social) entities.
- A development process must be defined to adequately engineer the software.
- Works associated with SE can be broadly categorized in to three phases:

(1) Definition Phase:

Here the key requirements for the system and the s/w are identified such as:

- What information is to be processed?
- What system behaviors can be expected?
- What function and performance are desired?
- What design constraint exists?
- What interface are to be established?
- What validation criteria are to be employed for successful system definition?

(2) Development Phase:

Here, the major phase is code generation and s/w. testing. Activation includes defining:

- How data are structured?
- How the function is to be implemented as s/w architecture?
- How the procedural details are to be implemented?
- How the interfaces are to be characterized?
- How the design is going to be translated in programming language?
- How the testing will be performed?

(3) Maintenance Phase:

In this phase, changes that are associated with error correction, adaptations required as the s/w's environment evolves; changes required due to changing costumer's requirements are handled.

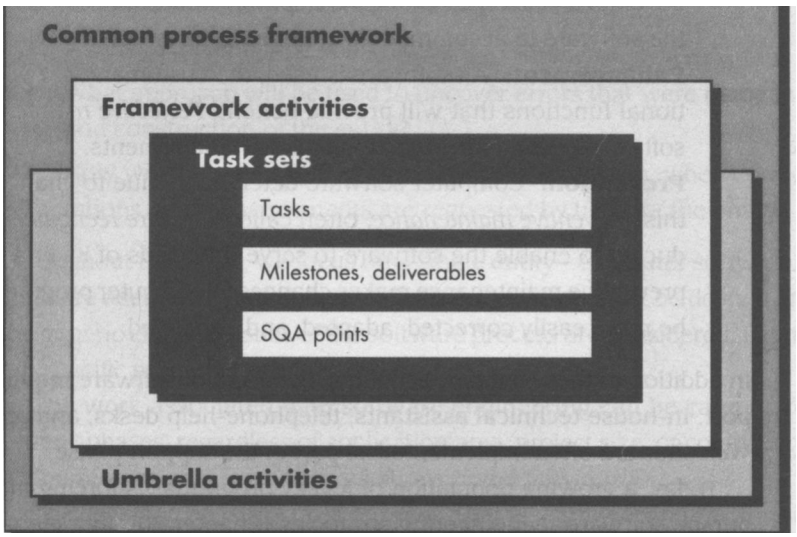
Changes are classified into four categories:

- ***Correction***:- changes made due to defects uncovered.
- ***Adaptation***: - changes made to accommodate change in the external environment (OS, s/w., govt. policies, etc.)
- ***Enhancement***:-changes made to extend the s/w beyond its original functions. (Also called Perfective Maintenance).
- ***Prevention***:- Preventive maintenance or s/w. Reengineering: changes are made to help better and easy correction, Adaptation and Enhancement.
- All these SE phases are complemented by a number of "Umbrella Activities" such as:
- S/W. Project Tracking And Control
- Formal Technical Review
- Software Quality Assurance
- Software Configuration Management
- Document Preparation And Production
- Reusability Management
- Measurement
- Risk Management.

Software Process

Established by defining few Framework Activities

Includes a number of Task Sets, each is a collection of Tasks, milestones s/w. work products, deliverables, and SQA Points-Adapted to the characterization of the s/w. project team.



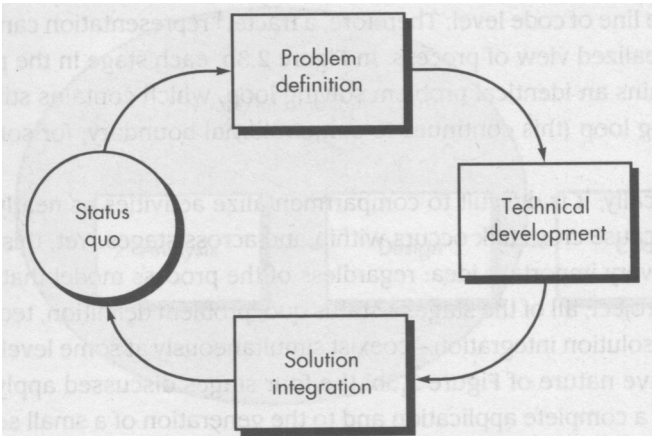
- The umbrella activities (SQA, SCM and measurement) overlay the s/w process model. These are independent of any one framework activity and takes place throughout the project.

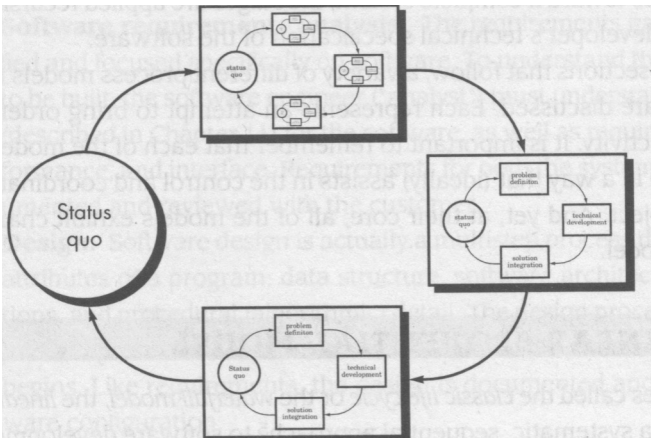
- SEI (software Engg. Institute) has developed comprehensive model, called Capability Maturity Model (CMM), which should be present when the organization reaches different levels of process maturity. This model uses a five-point grading scheme to determine compliance with the model that defines the key activities required at different levels of process maturity.
- This model provides a measure of global effectiveness of the company's SE practices and establishes five maturity levels:
- ***Level-1: Initial: -*** Few processes are defined. Success depends on individual effort.
- ***Level-2: Repeatable:*** - Repeat earlier success on projects with similar activities.
- ***Level-3: Defined: -*** Management and Engg. Activities are defined, standardized and integrated into organization wide s/w. process.
- ***Level-4: Managed:*** - S/w. process and the product are quantitatively understood and controlled using detailed measures.
- ***Level-5: Optimizing: -*** Continuous process improvement is achieved lay quantitative feed back too the process and from the testing.
- SEI has associated KPAs (key process Areas) with each maturity levels of CMM. Each KPA is described by identifying the following characteristics: -
    1. Goals (Objectives of KPA)
    2. Commitments (Requirements to meet goals)
    3. Abilities (of the organization.)
    4. Method for monitoring implementations (Are activities going on right?)
    5. Methods for verifying implementations (Is practice adapted perfect?)
    6. Activities- (Tasks to meet KPA)

Software Process Models (or Software Engineering Paradigm)

- To solve real life problems in industry settings, Software Engineers or a team of engineers must incorporate development strategy that covers the process, methods and tools.
- This strategy is called a software process model or software Engineering Paradigm, which is sleeted on the basis of the nature of the project and the applications, development methods and tools to be used, the controls and the deliverables that are required.

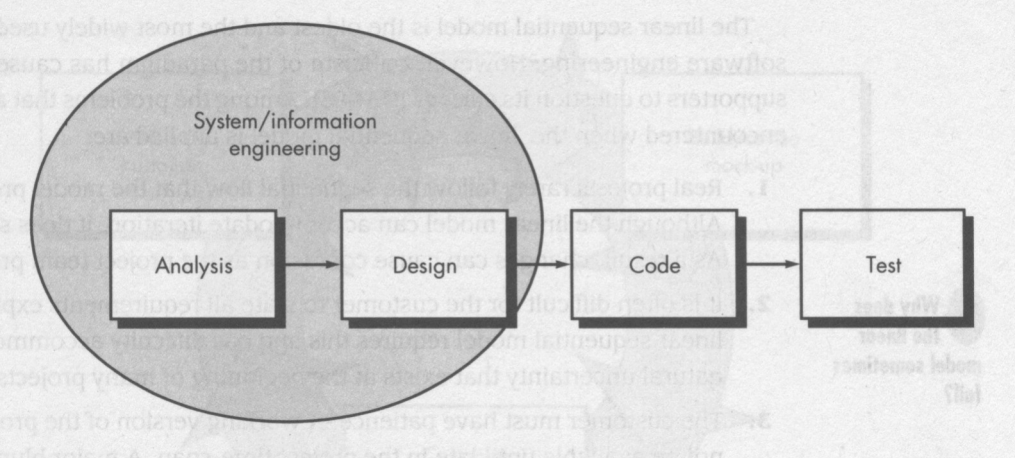
- Software development can be considered as a Problem Solving Loop (PSL). Problem solving loop (PSL) comprises of four stages:





- ***Status Quo***: Represents current state of affairs.
- ***Problem Definition***: Identifies specific problem to be solved
- ***Technical Development***: Solves the problem by application of come technology
- ***Solution Integration***: Delivers the result. (Document, program, Data, New Business Function, New product) to the client.
- Problem Solving loop applies to SE Process at different levels of resolution:
- ***Macro-Level*** - When entire application is considered.
- ***Middle-Level*** - When program components are considered.
- ***Line of Code Level*** - At the level of source program code.
- Neat Compartmentalization is not possible since some level of cross-talk occurs within and across the stages.
- These are various Software Process modes in existence:
    1. Linear-Sequential model (or waterfall model)
    2. Prototyping Model
    3. RAD model.
    4. Evolutionary models: Incremental, model, Spiral model, WINWIN Spiral model, Concurrent Development model
    5. Component Based Development
    6. Formal Method Model
    7. Fourth Generation Technology Model

1. Liner Sequential Model (or Waterfall Model)



Also Called Classical life cycle of Software Engineering.

- It suggests a systematic, sequential approach to software Engineering, which starts at system/information engineering level and progresses through analysis, design, coding, testing and maintenance phase.
- This is modeled after conventional engineering cycles.

1. System/Information Engineering and Modeling:

- Starts by establishing requirements for all system elements and then allocating some subset of these requirements to software.
- Interfaces with h/w, people, database, etc. apart form s/w. elements.
- In some eases, this level alto takes up some level of top-level analysis and design activities.

2. Software Requirement Analysis:

- Requirement gathering process is intensified and focused specially on software
- Study of information to be built, required functions, behavior, performance and interfacing requirements.
- Requirements for system and software must be determined and reviewed with the customer.

3. Design

- It is a multi-step process to address various aspects to be implemented, such as: Data structures, s/w Architecture, Interface Representation, Procedural /Algorithmic Details etc.
- It translates requirements into representations of the s/w, which can be assessed before the code generation.
- Design Document is also a part of the software configuration

4. Code generation:

- Here, the design is translated into a machine-readable form.

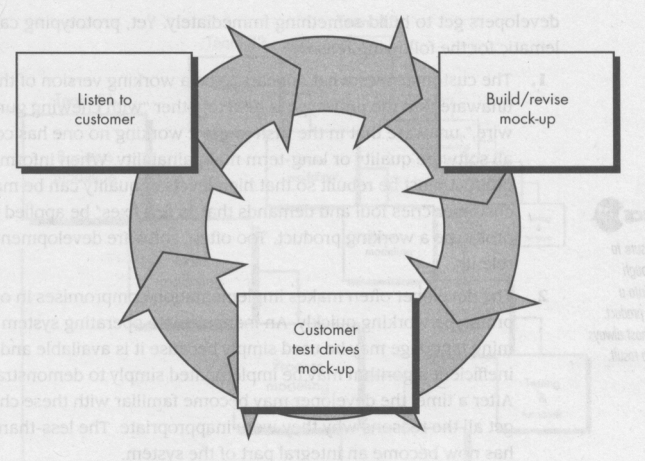
5. Testing:

- It focuses on the logical internals of the software (all statements) and Functional externals of the software - test to uncover errors.
- Basic objective is defined - input should produce desired output

6. Maintenance and Support:

- Maintenance of changes to accommodate changing user requirement
- ***SOFTWARE CONFIGURATION:*** Outputs (deliverables) of a large s/w development project consists of source code, executable design documents, SRS, test cases, etc. During development, they are changed by many. State of these at any point of time is referred to as software configuration.

2 Prototyping Model



- Customer is not sure of what is to be done or what to explain (objective of s/w) i.e. can not explain what i/p, o/p and processing. Developer is not sure of efficiency of algorithm, adaptability, or interface, etc.
- Prototyping Paradigm starts with Requirements gathering (involves customer and developer)
- Define overall system objective
- Track all known requirements
- Cutlive areas of further development.
- Quick Design Approach
- Result in all user visible aspects (Input approach, output formats, ets.)
- Construction of a prototype
- Evaluated be the customer/user.
- Specify refinements in the requirements.
- It is an interactive process. Prototype serves as a tool to identify software requirements. Working prototypes, if needed, are built form existing program fragments or tolls, libraries, etc. Once done, what to do with working prototypes? The first system built is barely usable - May be too slow, may be too big, awkward in use, etc. Usually a new system is built (i.e, redesigned revision). Prototyping leads to the development of the first system.
- Prototyping can be problematic too.
- ***Customer's View:*** Customer pressurizes the developer to get the working version of the S/W. immediately. Customer demands a few fixes be applied to the prototype to make it a working product.
- ***Developer's View:*** Developer often makes implementation compromise to get the prototype work quickly.
- Inappropriate OS or programming language may be used because it is available and known to the developer.
- Inefficient algorithms are implemented to demonstrate the capabilities. Due to these practices, the developer becomes used to it and targets all the reason why they are inappropriate.
- The less-than-ideal choice becomes an integral part of the system.
- However, apart from all these factors, Prototyping can be an effective paradigm. The customer and the developer, both should agree from the beginning that prototypes are to serve the

mechanizing of defining requirements. It is then discarded and the actual software is engineered keeping quality and maintainability aspects in view.

3. The RAD Model (Rapid Application Development)

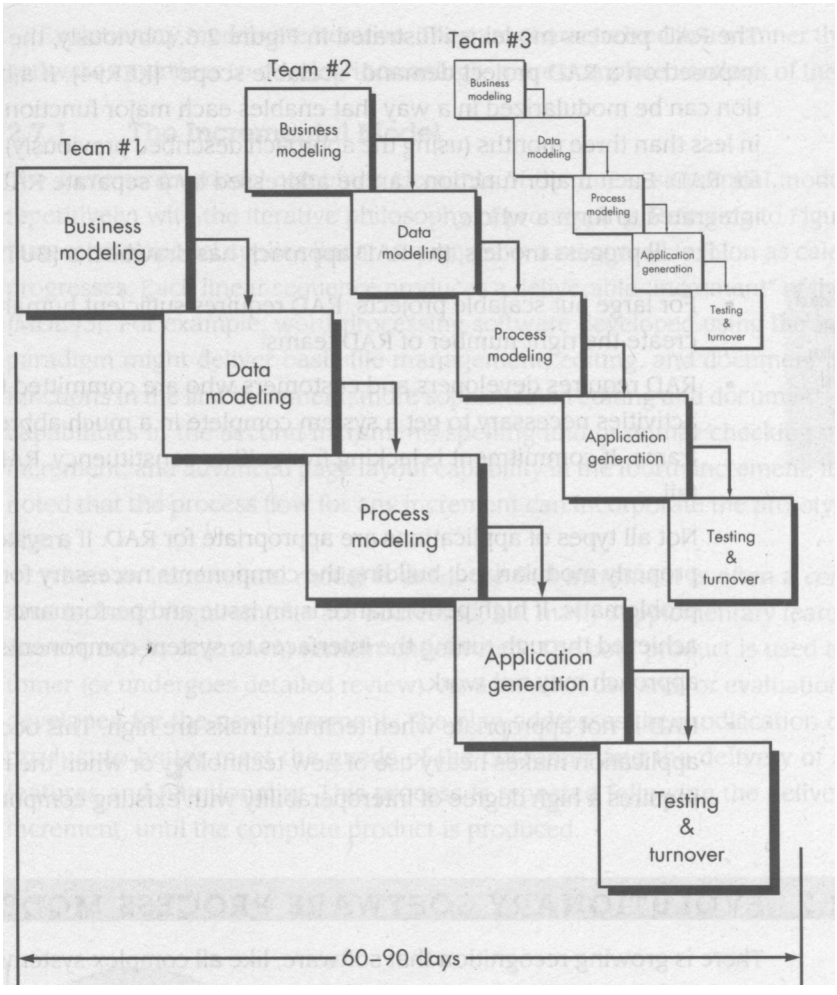
- RAD is a linear-sequential software development process model made for very short development cycle.
- It is "high speed" adaptation of the linear-sequential S/W development process which is achieved by using component based construction approach.
- If the requirements are well understood, RAD process develops the "fully functional" system within 60 to 90 days.
- RAD is primarily used for information system applications.
- Phases of RAD Process are:
- ***1. Business Modeling:*** models information flow among business functions, various inputs, processes, output aspects of information.
- ***2. Data Modeling:*** Phase-1 is redefined into a set of data objects that takes part in business activity (i.e. information process).
- ***3. Process Modeling:*** The data objects are transformed through process to achieve the information flow.
- ***4. Application Generation:*** RAD model makes use of 4th Generation techniques, reuses existing program components wherever possible, any other automated tools speed-up the development process.
- ***5. Testing and Turnover:*** Testing overhead is reduced due to reusability. Only the new components need testing. All new interfaces need to be exercised.

The RAD Process Model:

- ***Key Aspect***: If a business application can be modularized in a way that each major function can be completed with 60/90 days, than it is a candidate for consideration under RAD model.

***Drawbacks*:**

- Human resource requirement overhead is very high.
- Customer and the developer, both should be committed.
- All types of application are not appropriate for development under RAD strategy.



- RAD is not applicable when technical risk is high, i.e.:
- Performance is to be achieved through learning.
- New technologies are involved for development.
- High degree of inter-operability with existing system is required.

4. Evolutionary Software Process Models

- Software engineers are required to take up process models to accommodate a product evolution (a maintenance aspect) over time due to the following factors:
  - Business and product requirement changes as the development proceeds→straight path to end product becomes unrealistic.
  - Tight market deadline→limited version to be introduced.
  - System requirements and set of core products are well understood - but details of product extensions or system extensions are not known.
- In contrast to these factors:
  - Linear-sequential model is meant for straight line (linear) development approach.
  - Water-fall model delivers a complete product.
  - Prototyping helps the customer to understand the system requirements.
- Other overloads of Evolutionary Software Process Models:
  - Evolutionary models are inherently interactive in nature.
  - It helps to develop increasingly more complete versions of the target software.
- These are many software process models falling under the category of "Evolutionary Software Process Models". Some of these models discussed further in greater detail are:

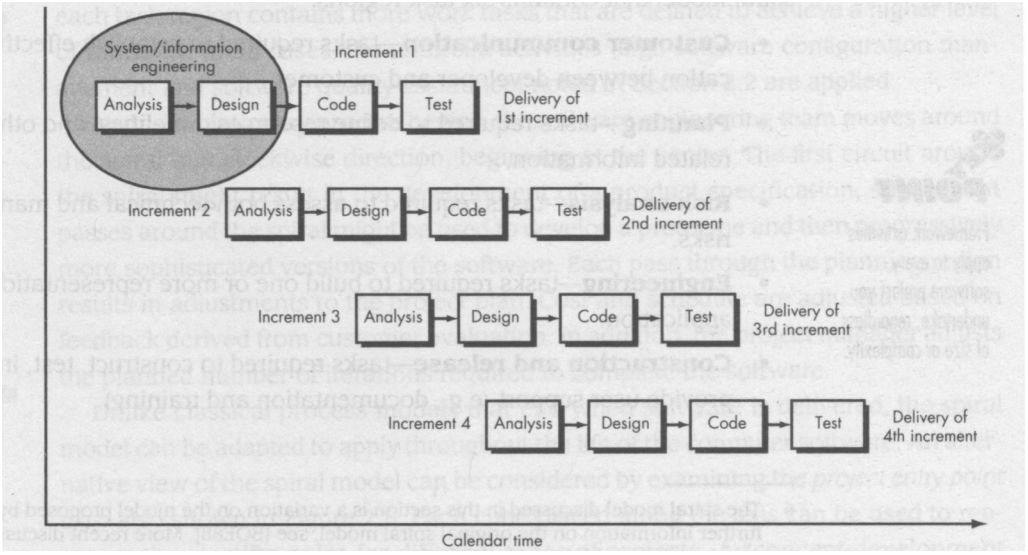
  4(a). Incremental Model,

  4(b). Spiral Model,

  4(c). WINWIN Spiral Model and

  4(d). Concurrent Development Model.

4(a). Incremental Model



- It is combination of linear-sequential Model philosophy with the iterative philosophy of Prototyping paradigm.
- The model is outlined pictorially below:

Example: Word Processing Software:

* 1st Increment: The core product (Basic Word Processing Application Software)
  - Basic WP requirements are addressed.
  - Supplementary features (known+unknown) are not delivered.
  - Core is used by the customer(Undergoes detailed review).
  - Helps to plan next development in order to betterment customer's need and delivery of additional features and functionality.

* Subsequent Increments: –This process is repeated with delivering each time until the complete product is produced (final version).

–Each increment is a stripped-down version of the final product.

–Each version fulfills users need and provides a platform for evaluation by the user, and hence further development.

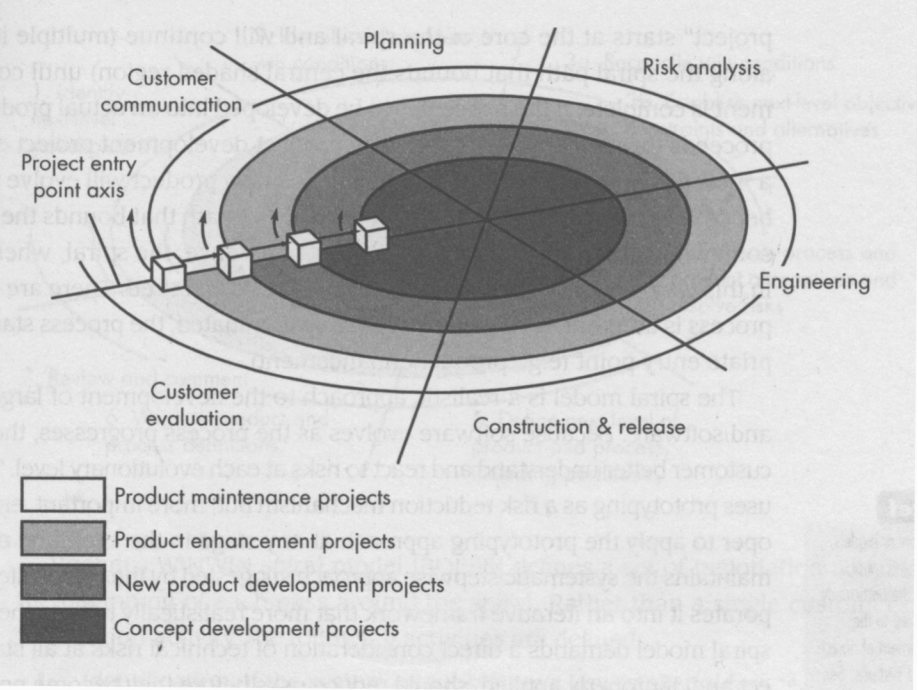
Benefits:

– Low manpower requirement

– Early increments can be implemented with fewer people.

– Increments can be planned to manage various technical risk (Change in hardware platform, OS features, etc.)

4(b). Spiral Model



- Proposed by Boehm
- This Evolutionary S/W process model combines the interactive nature Prototyping Model plus the control and systematic aspect of linear sequential model.
- It has potential for rapid development of incremental versions of the software.
- Software is developed in a series of incremental releases.

  → ***Early stage increments:*** Paper Model or Prototype.

  → ***Subsequent Stage releases:*** More complete versions of required software.
- Spiral Model is divided into a number of Framework Activities or Task Regions. These are six task regions or frame work activities:

  1. Customer Communication: Task for effective communication between customer and developer.
  2. Planning: Task is to define resources, timeline and other project related information.
  3. Risk Analysis: The task is to assess technical and management visit.
  4. Engineering: Task required to build one or more representations of the application.
  5. Construction and release: Task to construct, test, install and provide support (documentation, training, etc.)
  6. Customer Evaluation: Task is to obtain customer feedback of evaluation (Engg. Stage Vs. Implementation Stage).
- Spiral Model handles the software development process in the phase manner, each phase being treated as a Project Work.
- Spiral Model divides the development process into 4 projects:
- Each region is populated by a serious of tasks specific to the nature of the project.
- In all cases, umbrella activities are applied (SCM and SQA).
- All the stages are iterative in nature:

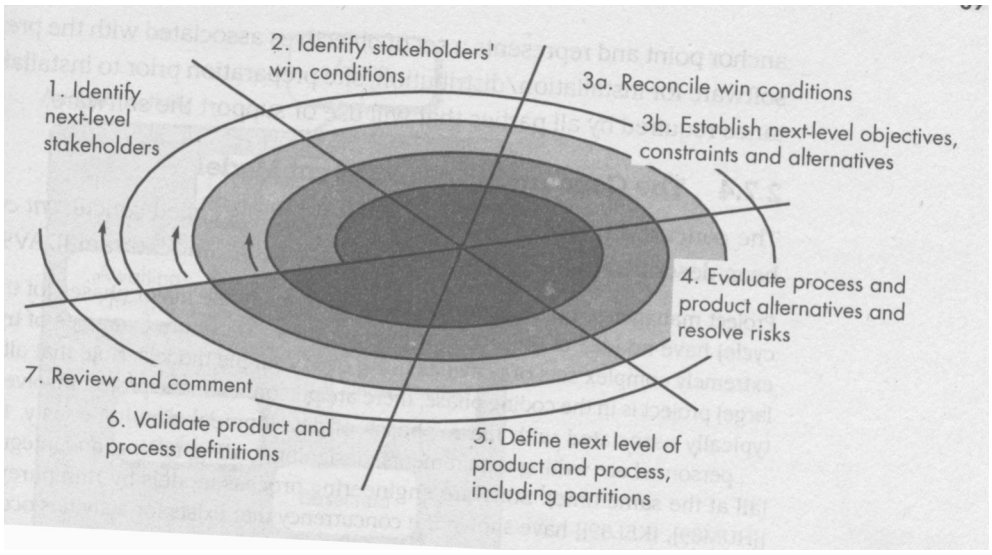
* ***1st iteration*** results in production of product specification.
* ***2nd iteration*** results in production of product prototype.
* ***Next iteration*** Results in production of progressively more sophisticated versions of the software.

- Each pass through the planning region two (2) results in adjustments to the project plan.
- Cost and schedule adjusted on the basis of customer evaluation.
- Project manager adjusts the no. of iterations to complete the software.
- Classical model ends when the S/W. is delivered. Spiral model can be applied throughout the life of the software.
- Each project in the Spiral model has a starting point on the project entry point axis which represents the start of a different type of project.
- Spiral model remains active until the software retires.
- Spiral model is a realistic approach to development of large scale projects (S/W).
- Spiral model uses Prototyping as a Risk Reduction Mechanism. Prototyping is applied at any stage of the product.
- It incorporates systematic approach as suggested by Classical life-Cycle of Software in an iterative way (Frame-work)
- It demands direct consideration of technical risk.

Discussions:

- Difficult to convince customer that evolutionary approach is controllable.
- High expertise is required to assess considerable risk.
- This is a new model, not used wisely as linear- sequential Development Approach.
- It will take number of years before the effectives of this model is known.

4(c) WINWIN Spiral Model



- Spiral Model suggests customer communication to decide upon project requirements from customer:
- Developer asks what is required–Customer provides necessary details.

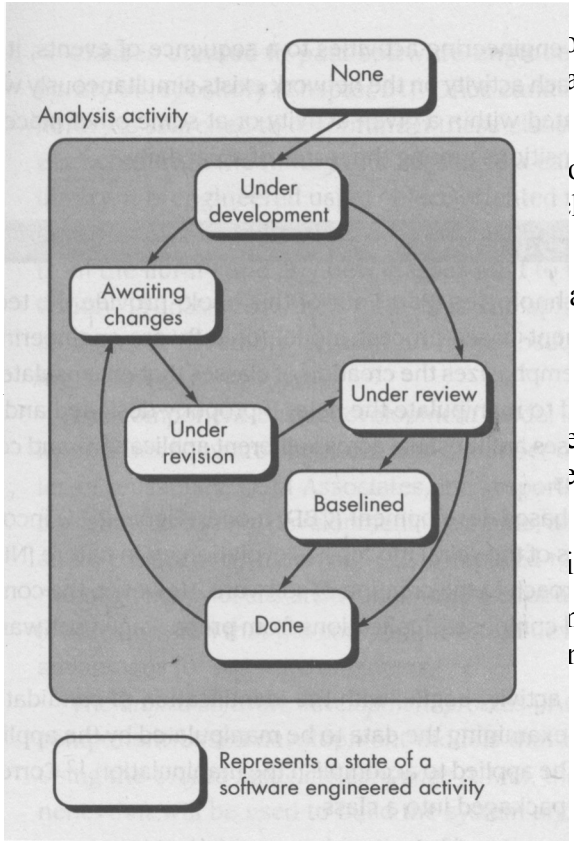
- In reality, developer negotiates with customer for functionality, performance and other product/ system features against cost and time-to-mark etc.
- Negotiation is successful at Win-Win State:
  - * Customer wins by getting a product/system that satisfies majority of his requirements.
  - * Developer wins by deadline target and achievable budget.
- Negotiation takes place at the beginning of each pass around the spiral, involving the following activities:
  1. Identification of the key stakeholders of the system/sub-systems.
  2. Determination for the stockholder's win condition.
  3. Negotiation of the stockholder's win condition to fit into a set of win-win conditions for all concerned (including S/W Development Project team)
- Win-win Condition is the key criterion for the S/W and system definition to proceed.
- This model introduces 3 process milestones (anchor points) to establish completion of one cycle around the spiral and provides decision milestones before the S/W project proceeds.
- Anchor points (AP) gives different view of progress or the project moves through the spiral.
- The three Anchor points are:

A. LCO- life Cycle Objective: Defines objective of each major SE Activity. Defines top-level systems requirements.

B. LCA-Life Cycle Activities: Establishes objectives that the system and S/W architecture must meet (Use of off-the-shelf and reusable components).

C. IOC- Initial Operation at Capabilities: Establishes the log of preparation for the S/W installation and distribution, site preparation prior to installation.

4(d). Concurrent Development Model



or phases of a project have no idea of project status since an one activity - might be writing SRS, doing design,

of activities occurring in any one phase (Requirement h can be represented by notations to represent state of a

affects the time-bound nature of software development

el can be represented schematically as a series of major ed states. e.g. Analysis Activity can be represented as

ide in different states.

nmunication activity completed the 1st iteration and is in nsition from state 1 to state 2. Now as a part of customer

communication activity, the customer signals change in requirement; analysis activity makes a move to state 3.

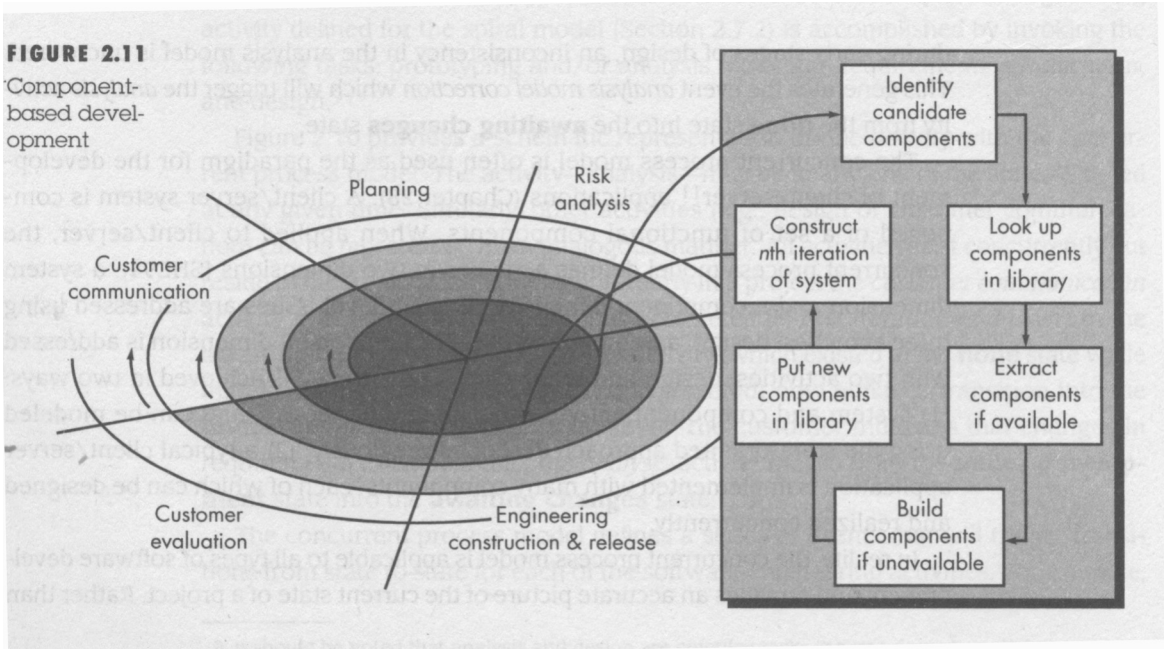
- Concurrent process model defines a serious of events that will trigger transition from state to state for each software engineering activity.
- In general, this model is used as a paradigm for client/server applications which comprises of a set of functional components.
- Concurrent Development Model defines Client/Server applications in two dimensions:

  1. System and component activities can be concurrently taking place (A state oriented approach.)
  2. Design and Realization of many components can take place concurrently.

Comments:

- Applicable to all types of S/W developments.
- Helps to figure out the actual picture of the state of the project.
- Instead of showing S/W engg. activities as a sequence of tasks, it defines a network of activities exiting simultaneously with other activities.
- Events generated in one activity may trigger a state transition of an activity.

5. Component Assembly Model



- Component Assembly model is based on the principle of object oriented technology.
- OO technology emphasizes creation of class that emasculates both data and the algorithm (code) that are used to manipulate the data.
- OO technology provides very high level of reusability.
- Component assembly model resembles spiral model.
- It is evolutionary in nature and uses iterative approach to develop software.
- It composes applications from pre-packaged software components (class)
- The Phase Engineering, Construction and Release is

- Accomplished by examining the data that are to be manipulated by the application and the algorithms that will be applied to accomplish manipulation. These data and code are packaged to form a software component (or a class).
- Old classes are stored in class-libraries.

- This model leads to software reuse
- Reusability provides measurable benefits to software engg. process
- QSM Inc. Report indicates:
  - 70% reduction in development cycle time.
  - 84% reduction in project cost
  - 26.2% productivity index Vs. 16.9 as industry harm.

6. Formal Methods Model:

- This model covers a set of activities that leads to a formal mathematical specification of computer software. It adopts mathematical notations for specification, development and verification of computer based systems.
- Clean room Software Engineering is a variation of this approach.
- Mathematical analysis can help detection and removal of ambiguity, incompleteness and inconsistency while specifying a system
- At the design stage, this model helps to detect and correct several errors during program verification that would have gone undetected.
- During development, it helps the software engineer to overcome many problems, which are not possible in other paradigms.
- This model can guarantee production of detect-free software. It is a must for building safely-critical software.

Bottlenecks:

- Development of formal model is time-consuming and expensive.
- Needs the S/W engineers to have adequate background and training.
- Also needs the customer to be technically sound to participate on the customer-communication / feedback mechanism.

7. Fourth Generation Techniques:

- It covers a broad array of software tools. Each tool helps the S/W. engineer to specify some characteristics of the S/W at high level. These tools can automatically generate source code based on the developer's specification.
- At present, it is one of the dominant approach to software development.
- More higher the level of specification, faster the development.
- Ability lies in specifying the S/W. using specialized language forms or graphic tools to describe the problem to be solved. At present, 4GT Tools include Non-procedural language for db query;

Report Generation; Data manipulation; Screen Interaction and Definition; Code-Generation; High-level graphics capability; Spread-Sheets, etc.

- Makes prototyping quite flexible and easier - suitable for both small projects as well as longer system projects. 4GT transfers implementation into a product. Minimal Testing is done by the developer

Advantages:

- Less time, High Productivity, Reduced cost etc.

Disadvantages:

- Poor code quality, Poor program efficiently etc.

Sources: -

- Ghezzi, Jazayeri and Mandrioli, *Fundamentals of Software Engineering*
- Roger S. Pressman, *A modern approach to Software Engineering*
- Valacich, George & Hoffer, *Essentials of Systems Analysis and Design*