SOFTWARE ENGINEERING FUNDAMENTALS

Rudra Nepal

SOFTWARE PROJECT MANAGEMENT CONCEPTS

- SOFTWARE?

Instruction which when executed provided desired features,function and performance

Data structure that enables the programs to adequately manipulate information

Documents describing operations and use of programs

Characteristics of software that make it different from other thing that human being build

- Software is logical rather than a physical system element
- Software is developed or engineered,it is not manufactured in classical sense.
- Software does not wear out :makes it considerably different from hardware

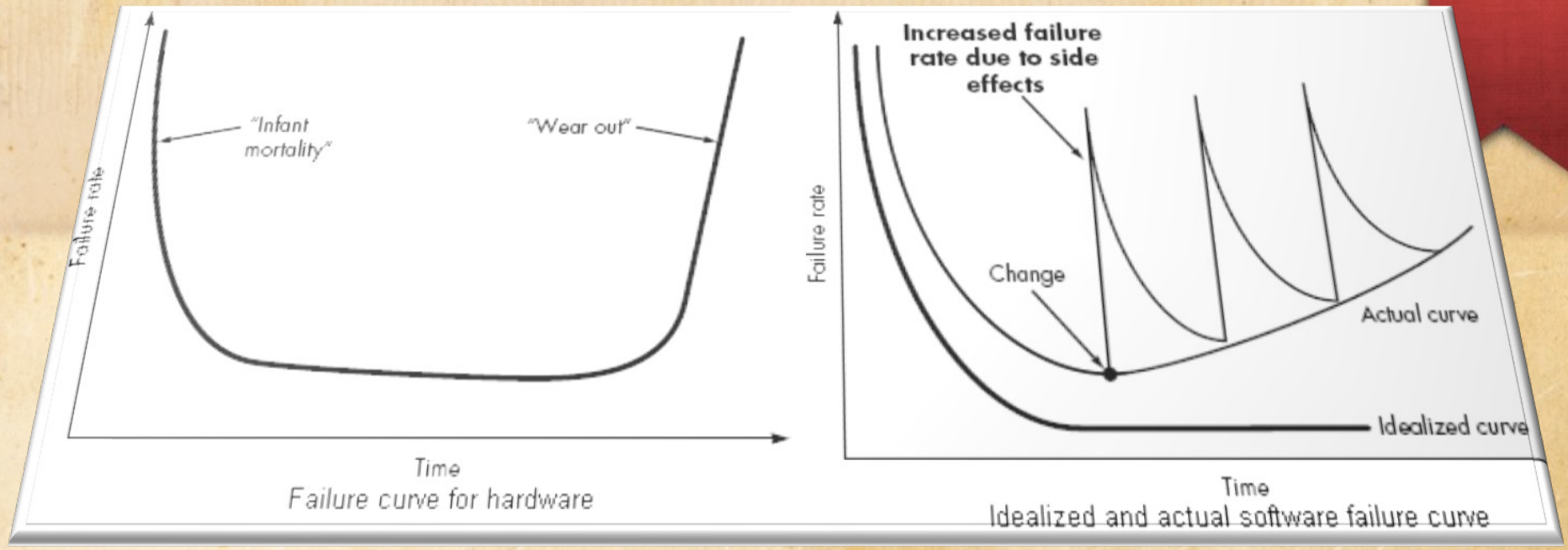


Fig: failure curve for hardware and software

Software is not susceptible to the environmental maladies that cause hardware to wear out. But it does deteriorate.
During the life time software undergo change.
As change are made it is likely that error will be introduced,causing the failure rate curve to spike and soon.
When hardware component wears out, it is replaced by spare part but there is no software spare part.

What are the attributes of good software?

- The software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.

Maintainability

- Software **must evolve** to meet changing needs

Dependability

- Software must be **trustworthy**

Efficiency

- Software should not make **wasteful use** of system resources

Usability

- Software **must be usable** by the users for which it was designed

Why need software engineering?

Analogy with bridge building:

Over a stream = easy, one person job

Over River Severn ... ? (*the techniques do not scale)*

Many sources, but ***size*** is key:

UNIX contains 4 million lines of code

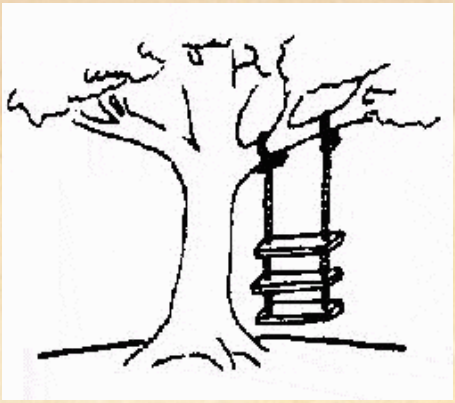
Windows 2000 contains 10^8 lines of code

Problem : ***complexity***

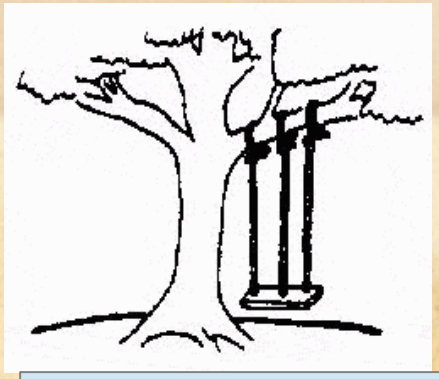
Software engineering is about managing this complexity!!

Definition:The application of systematic,disciplined,quantifiable approach to the development,operation and maintenance of software.
i.e application of engineering to software.

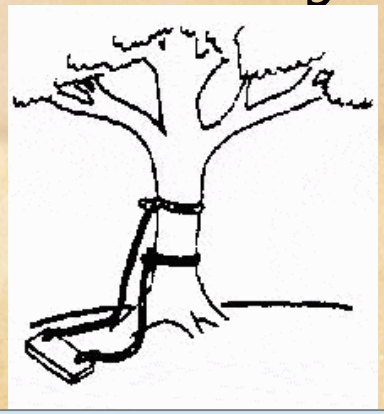
How Programs Are Usually



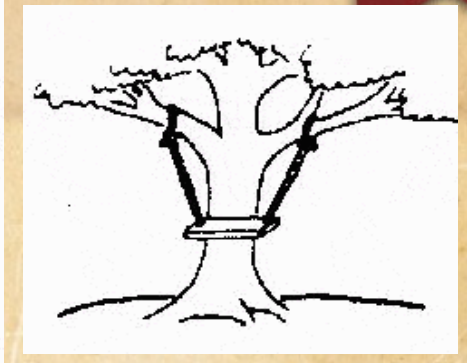
specification was defined like this



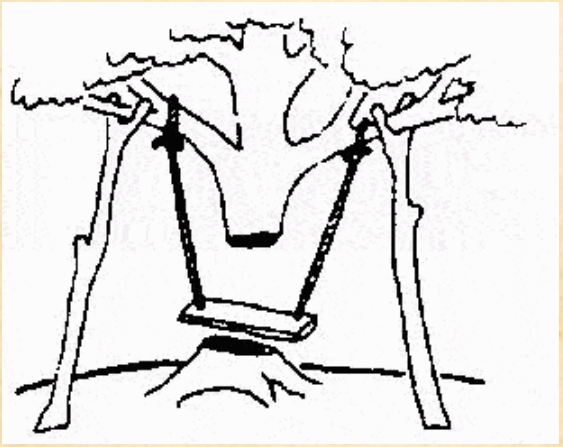
The developers understood it in that way



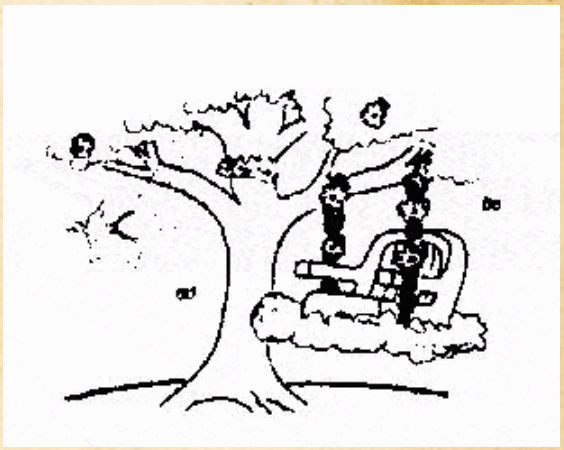
This is how problem was solved before.



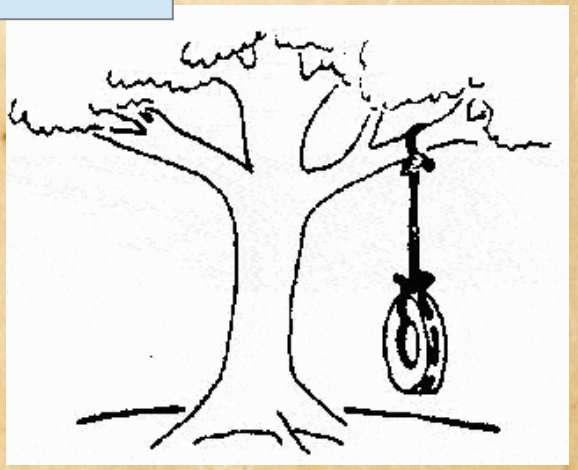
This is how problem is solved now



That is program after debugging



This is how program is described by marketing department



This, in fact, is what the customer wanted … ;-)

Software engineering layered technology

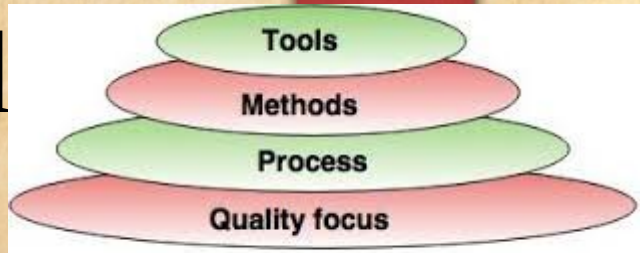


Fig. - Software Engineering Layers

- Software engineering is a layered technology.
- An engineering approach must have a focus on quality which provides a continuous process improvement culture.
- Process layer is the foundation that defines a framework with activities for effective delivery of software engineering technology.
- Method provides technical how-to's for building software. It encompasses many tasks including communication, requirement analysis, design modeling, program construction, testing and support
- Tools provide automated or semi-automated support for the process and methods.

The main aim of software engineering are:
producing good quality,
 maintainable software,
 on time,
 Within budget

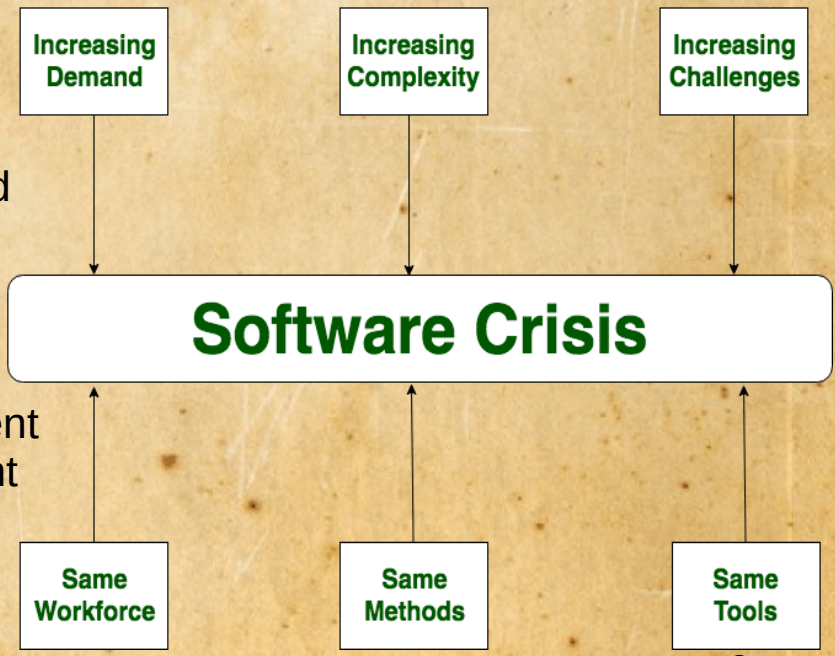
Software Crisis

Software Crisis is a term used in computer science for the difficulty of writing useful and efficient computer programs in the required time .software crisis was due to using same workforce, same methods, same tools even though rapidly increasing in software demand, complexity of software and software challenges. With increase in the complexity of software, many software problems arise because existing methods were insufficient.

use same workforce, same methods and same tools after fast increasing in software demand, software complexity and software challenges, then there arose some problems like software budget problem, software efficiency problem, software quality problem, software managing and delivering problem etc. This condition is called software crisis.

Reasons for s/w crisis

- Lack of communication between s/w developers and users
- Increase in size of software
- Increased complexity of problem area
- Project management problem
- Lack of understanding of problem and its environment
- High optimistic estimates regarding s/w development time and cost.



SOFTWARE MYTHS

Myths :Traditional story accepted as history.

-**Software standards provide software engineers with** all the guidance **they need.** *The reality is that the standards may be* ***outdated*** *and* ***rarely referred to***

- **People with** modern computers **have all the software development tools .** *The reality is that CASE tools are more important than hardware to producing high quality software, yet* ***they are rarely used*** *effectively*

- Adding people **is a good way to catch up when a project is behind schedule.** *The reality is that adding people only helps the project schedule when it is done in a planned well coordinated manner*

- **Giving software** projects to outside parties **to develop solves software project management problems.** *The reality is people who cant manage internal software development problems will struggle to manage or control the external development of software too*

- **A** general statement of objectives **from the customer is all that is needed to begin a software project**. *The reality is without constant communication between customer and developers it is impossible to build a software product that meets the customers real needs*

- Once a program is written **, the software engineers work is finished.** *The reality is that* ***maintaining a piece of software is never done****, until the software product is retired form service*

- **There is** no way to assess the quality **of a piece of software until it is actually running on some machine**. The reality is that one of the most effective quality assurance practices (FTR) can be applied to any software design product and can serve as a quality filter very early in the product life cycle

- **The only deliverable from a successful software project is the** working program**.** *The reality is the* ***working program is only one of several deliverables*** *that arise from a well managed software project.*

- **Software engineering is all about the creation of large and unnecessary documentation**

The reality is that software engineering is concerned with creating quality.

- Project requirements change continually and change is easy to accommodate in the software *design.*

The reality is that every change has far reaching an unexpected consequence.

Software process and process model

What is process?

- A road map for building the software.
- Software process is a framework for the task that are required to build high quality software
- Also provides a framework for managing development activities.
- A software process help to produce program,documentation and data.
- Measure of success
  - quality(less error,find all errors)
  - timeliness(don' miss deadline)
  - Long-term viability(platform independent,language independent)

FIVE ACTIVITIES OF A GENERIC PROCESS FRAMEWORK

 A General Process framework consists of five major activities.

 These activities can be used for all software development regardless of the application domain, size of the project, complexity of the efforts etc.

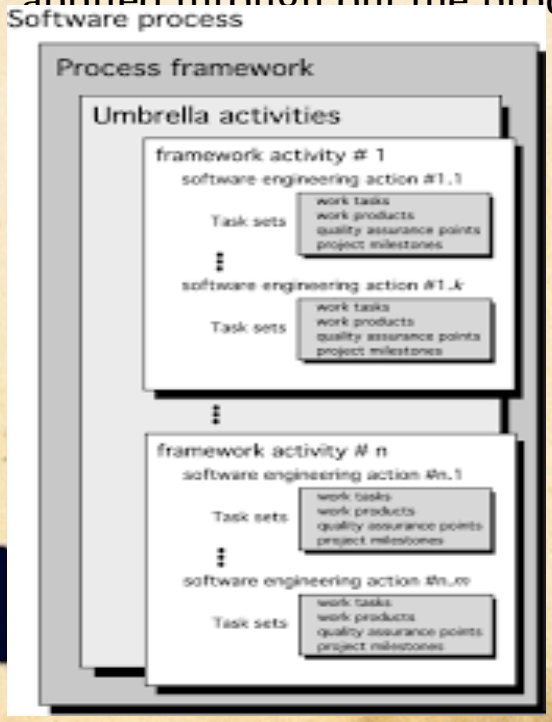
 For many software projects, these framework activities are applied iteratively as a project progresses.

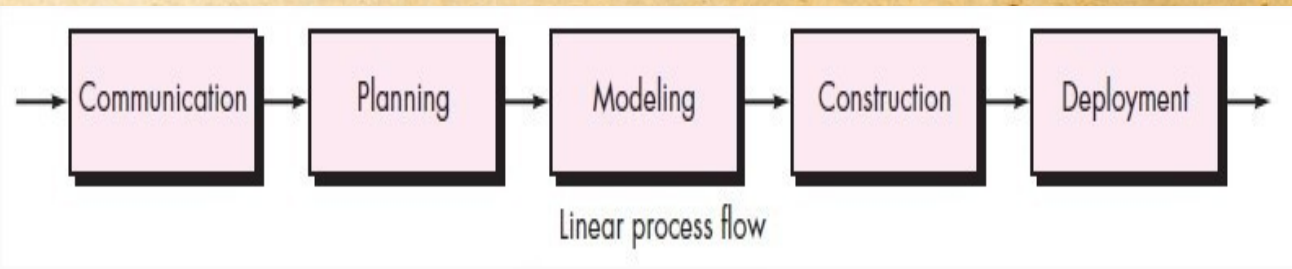
 Each iteration produces a software increment that provides a subset of overall software features and functionality.

FIVE ACTIVITIES OF A GENERIC PROCESS FRAMEWORK

- Communication: communicate with customer to understand objectives and gather requirements
- Planning: creates a “map” defines the work by describing the tasks, risks and resources, work products and work schedule.
- Modeling: Create a “sketch”, what it looks like architecturally, how the constituent parts fit together and other characteristics.
- Construction: code generation and the testing.
- Deployment: Delivered to the customer who evaluates the products and provides feedback based on the evaluation.

In addition a set of umbrella activities -project tracking and control,risk management,quality assurance ,configuration management,quality assurance,technical reviews and others are applied through out the process.





Software Process Models (or Software Engineering Paradigm)

- **It is descriptive and diagrammatic model of software life cycle.**
- **Identifies all the activities required for product development.**
- **Captures the order in which these activities are to be undertaken.**
- **Divides the life cycle into phases,where several different activities may be carried out in each phase.**
- **A process covers all the activities beginning from product inception through delivery and retirement,but a methodology covers only single activity or a few individual steps in the development.**
- **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 selected 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.**

WHY USE A LIFE CYCLE MODEL?

- **Primary advantage is it helps in development of software in a systematic and disciplined manner.**
- **When a program is developed by a single programmer,he has the freedom to decide his exact step.**
- **When the product is being developed by a team,there must be a precise understanding among team member as to “when to do what” otherwise it would lead to chaos and project failure.**

These are various Software Process models 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

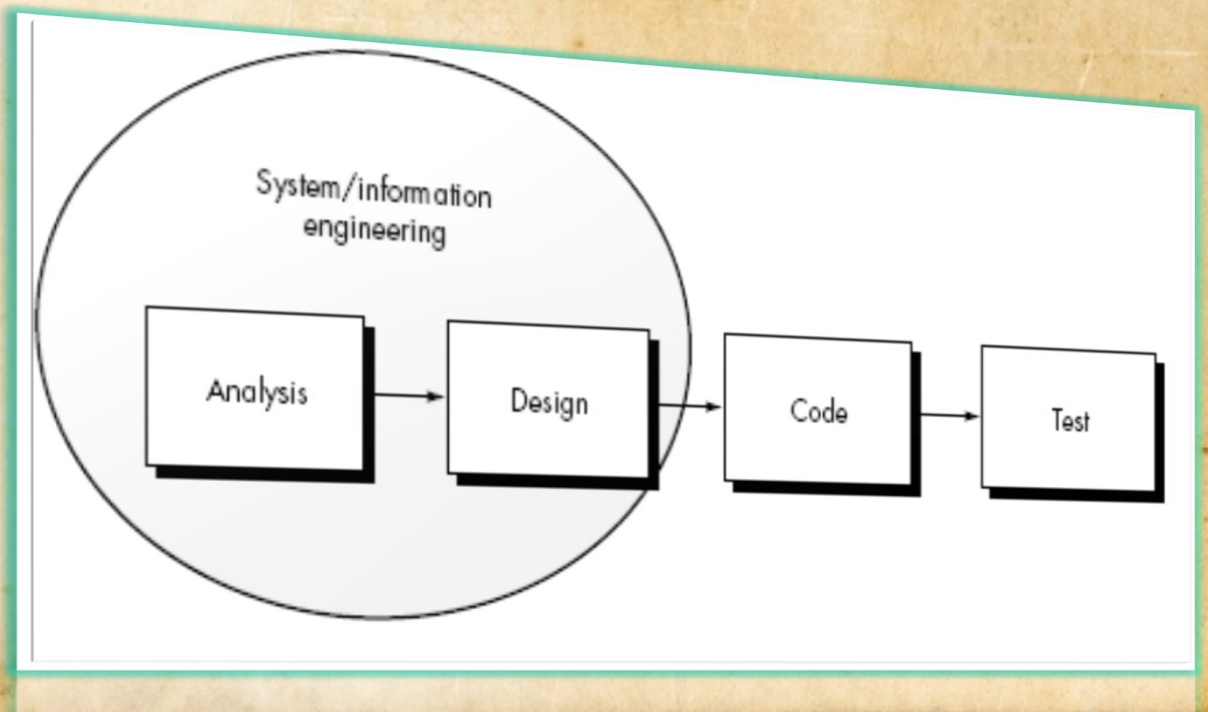
5. Component Based Development

6. Formal Method Model

7. Fourth Generation Technology Model

Waterfall model

- This model assumes that everything is carried out and taken place perfectly as planned in the previous stage and there is no need to think about past issues that may arise in next phase .
- This model doesn't work smoothly if there are some issues left at the previous step.
- The sequential nature of model doesn't allow us to go back and undo or redo our actions.
- This model is best suited when developers already have designed and developed similar software in the past and are aware of all its domains.



Drawback : Difficulty of accommodating change **after the process is underway**

Appropriate when requirements are well understood

Problems of waterfall model

1. It is difficult to define all requirements at the beginning of a project
2. This model is not suitable for accommodating any change
3. A working version of the system is not seen until late in the project's life
4. It does not scale up well to large projects.
5. Real projects are rarely sequential.

Prototyping Model

Customer

What is to be done or what to explain
(objective of software?)
Input? Output?
?

Developer

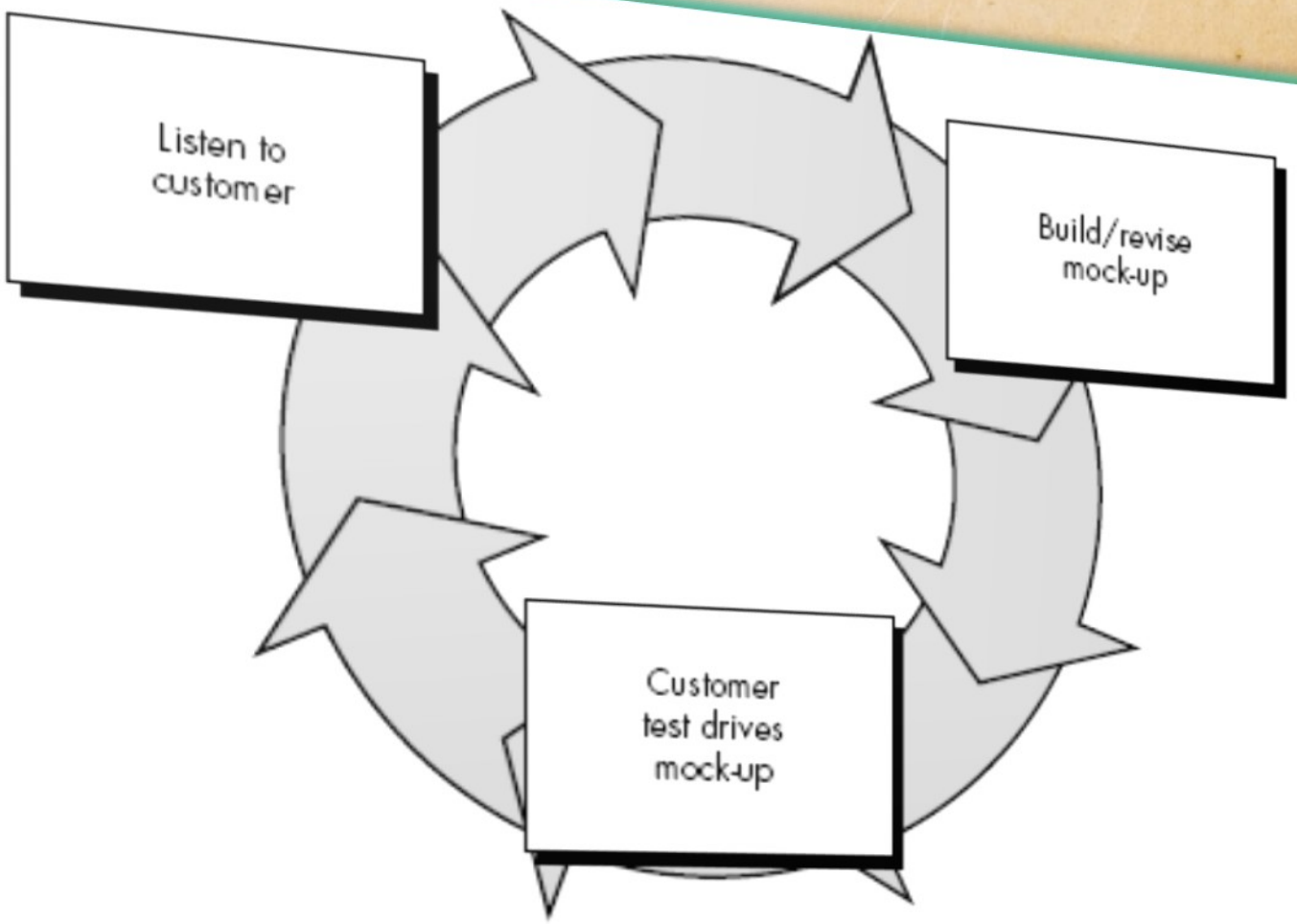
Efficiency of algorithm, adaptability , interfaces ?
?

- **Software prototyping refers to the activity of creating prototypes of software applications i.e incomplete versions of the software program being developed.**

- **The prototyping model is a systems development method in which a prototype is built,tested and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed.**

- **Customer sets general objective for software but doesn't properly identify detailed system behavior. Prototyping models tries to capture requirements of customer in detail through a series of quick design and evolution.**

- **This model beings with requirement gathering,then a quick design occurs which leads to development of prototype. Customer evaluates the prototype and uses it to refine requirements. Then iteration occurs as a prototype is modified to satisfy customers need.**


Listen to customer
Build/revise mock-up
Customer test drives mock-up

Limitations:

- Customer cries foul and demands some “few fixes” be applied to make the prototype a working product
- Developers may make a prototype for small database , use algorithm for few and for specific operating system ,so it may create problem in future.

I want a working version of my software immediately!!

Customer

I am not done yet! I only have a prototype ready!!

Developer

OK!! Just apply a few fixes **to the prototype to make it a working product!!**

Customer

I'll have to make some implementation compromise **to get the prototype work quickly**

Developer

Advantages:

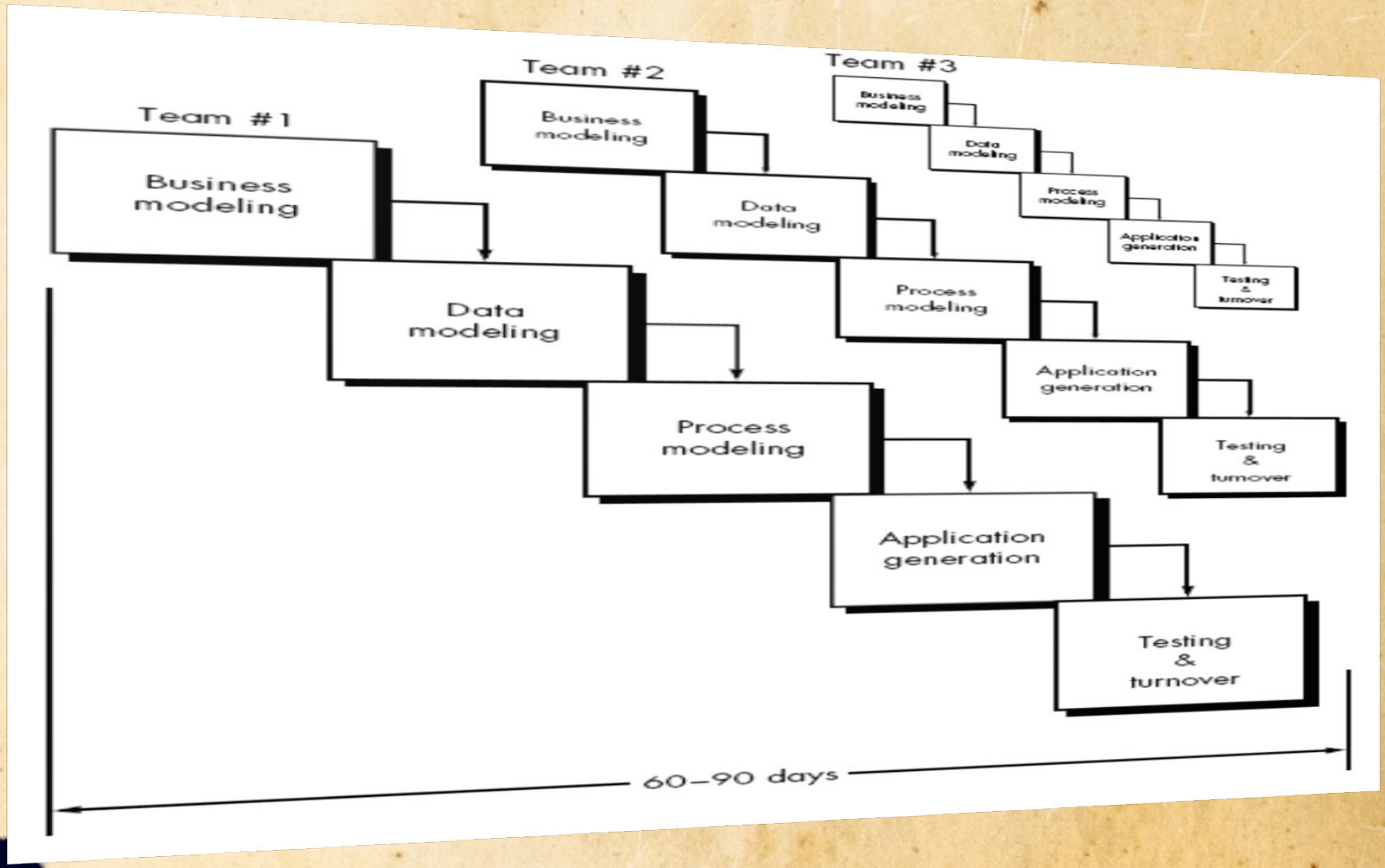
- Users are actively involved in development.
- Since in this methodology a working model of the system is provided, the user get a better understanding of the system being developed.
- Errors can be detected much earlier as the system is made side by side.
- Quicker user feedback is available leading to better solution.

The RAD Model(Rapid Application Development)

Primarily used for information system **applications**

If requirements are well understood **, and** modularized **, RAD process develops fully functional system within** 60 **to** 90 days

Phases of RAD



Business Modeling: Models information flow **among business functions, various input process, output aspects of information**

Data Modeling: Phase one is redefined into a set of data objects **that takes part in business activity (information process)**

Process Modeling: The data objects are transformed **thru processing to achieve the information flow**

Application Generation : RAD model makes use of fourth generation techniques reuses existing a program comp0nents wherever possible any other automated tools **to speed up the development process**

Testing and turn over: Testing overhead is reduced **due to reusability only the new components need testing**

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.
- Performance is to be achieved through learning.
- New technologies are involved for development.
- High degree of inter-operability with existing system is required.

Evolutionary Software Process Models:

Evolutionary models are inherently iterative **in nature**

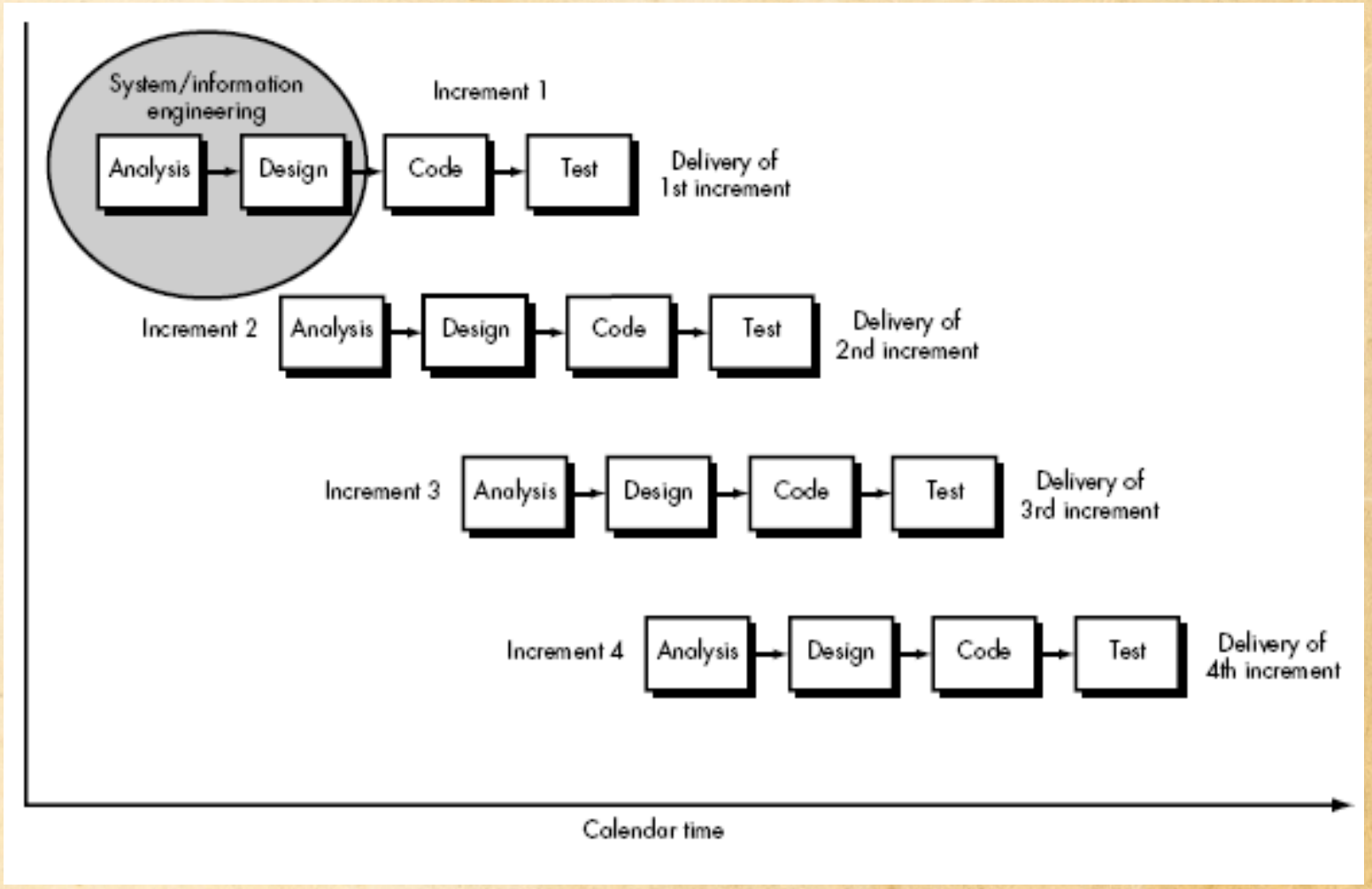
It helps to develop increasingly more complete versions **of the target software**

Types of Evolutionary Software Process Models:

- Incremental Model
- Spiral Model
- Win Win Spiral Model
- Concurrent development Model

Incremental model

It is a combination of linear sequential model philosophy with the iterative philosophy of prototyping paradigm
The model is outlined pictorially as below
Example of Incremental Model: Word processing Softrware
First 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 better meet customers need and delivery of additional features and functionality


System/information engineering
Increment 1
Analysis
Design
Code
Test
Delivery of 1st increment
Increment 2
Analysis
Design
Code
Test
Delivery of 2nd increment
Increment 3
Analysis
Design
Code
Test
Delivery of 3rd increment
Increment 4
Analysis
Design
Code
Test
Delivery of 4th increment
Calendar time

- **Subsequent increments:This process is repeated 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 for their 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)**

Spiral Model (Proposed by Bohem)

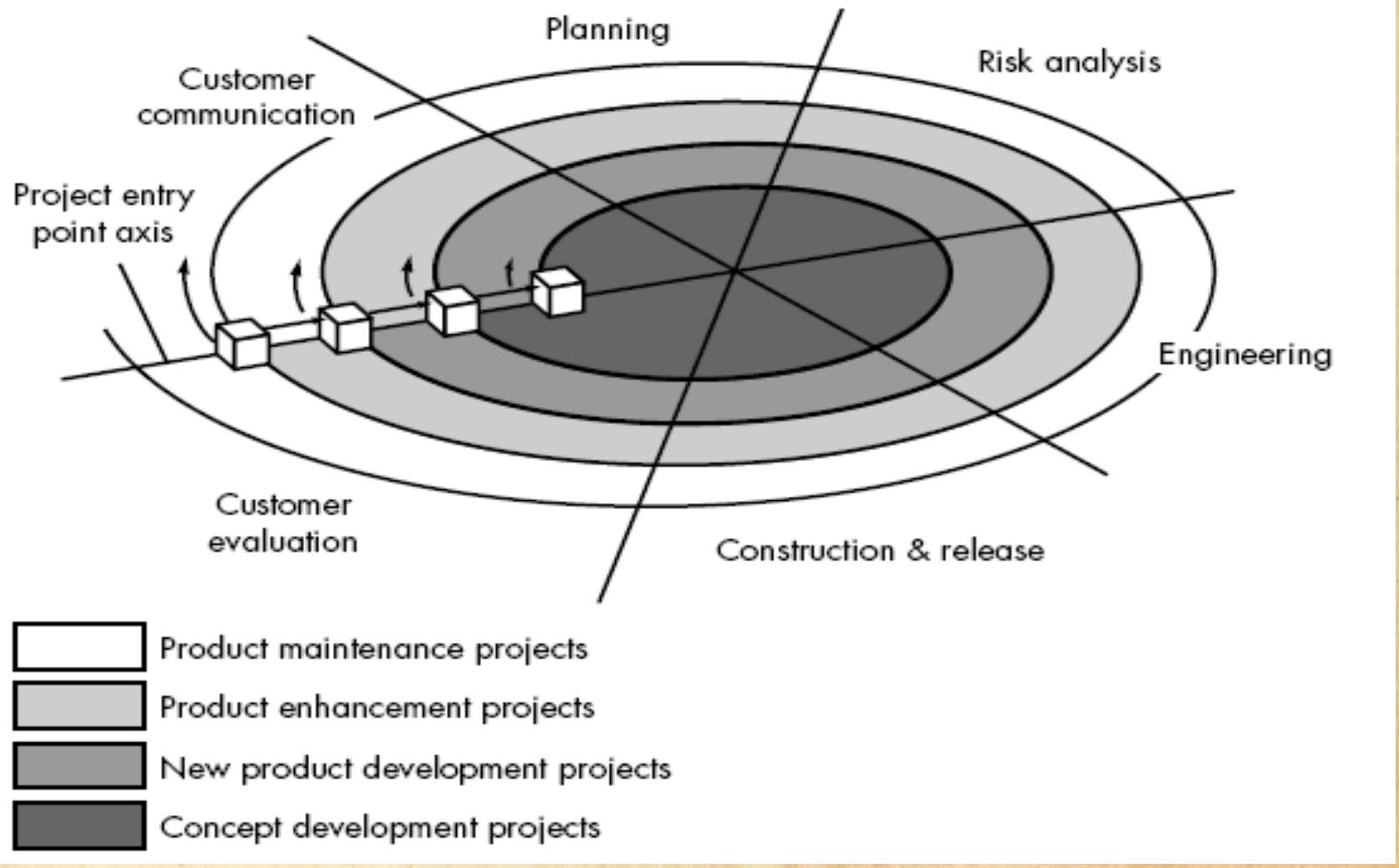
This evolutionary software process model combines the iterative nature of prototyping model , the control and systematic aspect of linear sequential model

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 version of required software


Planning
Risk analysis
Customer communication
Project entry point axis
Engineering
Customer evaluation
Construction & release
Product maintenance projects
Product enhancement projects
New product development projects
Concept development projects

Spiral model handles the software development process in phase manner, each phase being treated as a project work

Spiral model divides the development process into four projects:

Each region is populated by a series of tasks specific to the nature of the project

In all cases , umbrella activities are applied (SCM and SQA)

All the stages iterative in nature:

First Iteration: Results in production of product specification

Second Iteration:Results in production of product prototype

Next Iteration:Results in production of progressively more sophisticated versions of the software

Each pass thru the planning region two results in adjustment to the project plan

Cost and schedule adjusted on the basis of customer evaluation

Project manager adjusts the number of iterations to complete the software

Classical model ends when the software is delivered. Spiral model can be applied thru out the life of the software

Each project in the Spiral model has a starting point in 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

Spiral model uses Prototyping as a Risk Reduction mechanism. Prototyping is applied at any stage of the product

It incorporated 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 widely as linear sequential development approach
It will take number of years before the effectiveness of this model is known

WINWIN Spiral Model

Spiral model suggests customer communication to decide upon project requirements form customer

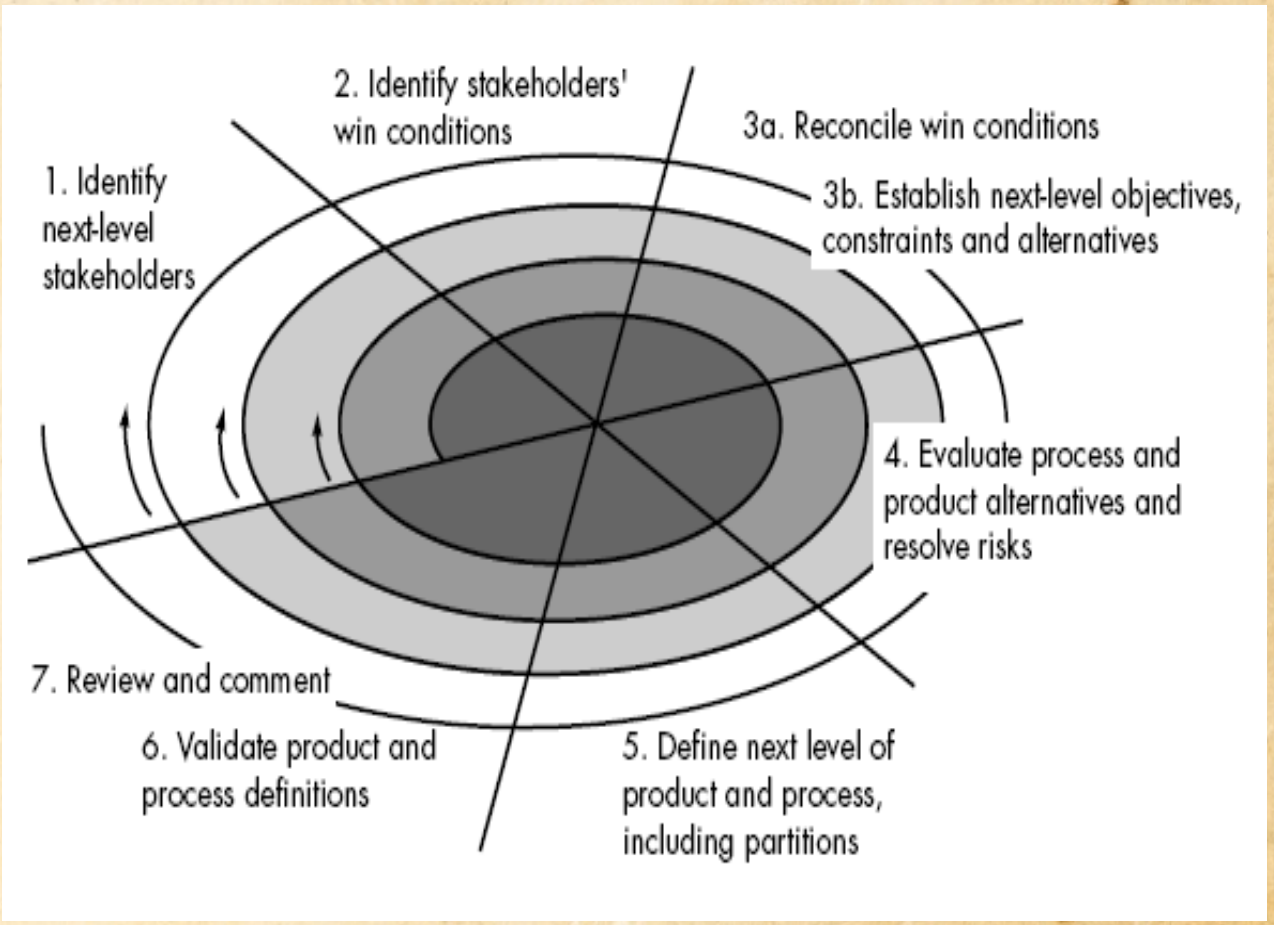
Developer asks what is required and customer provides necessary details

In reality, developer negotiates with customer for functionality , performance, and other product /system features against cost and time to market

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


2. Identify stakeholders' win conditions
3a. Reconcile win conditions
1. Identify next-level stakeholders
3b. Establish next-level objectives, constraints and alternatives
4. Evaluate process and product alternatives and resolve risks
7. Review and comment
6. Validate product and process definitions
5. Define next level of product and process, including partitions

Negotiation takes place at the beginning of each pass around the spiral , involving the following activities:

Identification of the key stake holders of the system/sub systems

Determination of the stake holder's win condition

Negotiation of the stake holder's win condition to fit into a set of Win-Win conditions

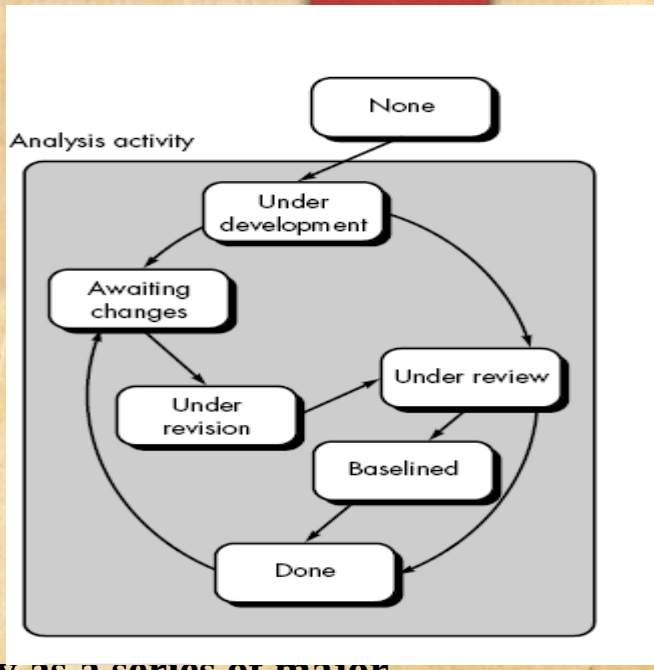
for all concerned (including software development project team)

Concurrent Development Model (by Davis and Sitaram)

Project managers tracking status of major phases of a project have no idea of project status since personnel are associated with more than one activity - might be writing SRS , doing design , coding , testing etc. all simultaneously

This shows existence of concurrency of activities occurring in any one phase (requirements change during late development) which can be represented by notations to represent the state of a process (state chart)

Existence of concurrency of activities affect the time bound nature of software development process



Any state of a concurrent process model can be represented schematically as a series of major technical activities, tasks and associated states e.g. analysis, activity can be represented as shown

All activities resides concurrently but resides in different states

For a Spiral Model :

When customer communication activity completed the first iteration and is in state three the analysis activity makes a transition from state one to state two.

Now as a part of customer communication activity , the customer signals a change in requirement , analysis activity makes a move to state three

Concurrent process model defines a series 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

- **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.**

Concurrent development model defines client/server applications in two dimensions:

System Dimensions: involves three activities (design, assembly, use)

Component dimensions: involves two activities (design and realization)

Concurrency is achieved in two ways:

i. System and component activities can be concurrently taking place (a state oriented approach)

ii. Design and realization of many components can take place concurrently

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.**

Drawbacks:

- **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.**

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.

References:

Software engineering A PRACRITIONER’S APPROACH,Roger S.PRESSMAN ,Bruce R.MAXIM