UNIT-1

Introduction to software engineering

Software :- Software is Instructions (computer programs) that provide desired features, functions & performance when executed

Data structures that Enable the programs to adequately manipulate Information

Documents that describe the operation and use of the Programs

Characteristics of software :

Software is developed or Engineered ; it is not manufactured in the classical sense

Software does not - wear out

Although the industry is moving toward component based construction, most software continues to be custom built

Software Engineering :

The systematic, disciplined quantifiable approach to the development, operation and maintenance of software ; that is, application of Engineering to software

The study of approaches as in (1)

EVOLVING ROLE OF SOFTWARE

Software takes dual role. It is both a product and a vehicle for delivering product.

As a product : It delivers the computing potential Embodied by Computer hardware or by a network of computers

As a Vehicle : It is Information Transformer - producing managing, acquiring, modifying, displaying, or Transmitting information that can be as simple as single bit or as complex as a multimedia presentation.

Software delivers the most important product of our time - Information

- It transforms personal data
- It manages business Information to Enhance Competitiveness
- It provides a gateway to world wide information networks
- It provides the means for acquiring information

The Role of Computer software has undergone significant change over a span of little more than 50 years

Dramatic Improvements in hardware performance
Vast increases in memory & storage capacity
A wide variety of Exotic input & Output options

1970s and 1980s

Osborne characterized a - new industrial Revolutional

- Toffler called the advent of micro electronics part of - the third wave of change" in human history
- Naisbitt predicted the transformation from an "industrial society to an - information society"
- Feigenbaum and mc corduck suggested that information and knowledge would be the focal point for power in the Twenty - first century
- Stoll Argued that the - Electronic community" created by networks and software was the key to knowledge interchange throughout the world

1990s began :

- Toffler described a - power shift in which old power structures disintegrate as computers and software lead to a - democratization of knowledge
- Yourdon worried that U.S companies might lose their competitive edge in software related business and predicted - the decline and fall of the American programmer
- Hammer and champy argued that information Technologies were to play a Pivotal role in the - reengineering of the corporation

Mid-1990s:

The pervasiveness of computers and software spawned a rash of books by neo-luddites

Later 1990s:

Yourdan reevaluated the prospects of the software profession and suggested - the rise and resurrection of the American programmer

The Impact of the Y2K-time bomb was at the end of 20th century

2000s progressed

- Johnson discussed the power of-Emergence a phenomenon that explains what happens when interconnection among relatively simple entities result in a system that - self organizes to form more intelligent, more adaptive behavior
- Yourdon revisited the tragic events of 9/11 to discuss the continuing impact of global Terrorism on IT community
- Wolfram presented a treatise on a - new kind of science that posits a unifying theory based primarily on sophisticated software simulations

Daconta and his colleagues discussed the Evolution of the Semantic web

Today a huge software industry has become a dominant factor in the economics of the industrialized world.

THE CHANGING NATURE OF SOFTWARE

The 7 broad categories of computer software present continuing challenges to software engineers:

System software

Application software

Engineering / scientific software

Embedded software

Product-line software

web-applications

Artificial Intelligence software

System software : System software is a collection of programs written to service other programs. The system software is characterized by

- Heavy interaction with computer hardware
- heavy usage by multiple users
- concurrent operation that requires scheduling, resource sharing and sophisticated process management
- Complex data structures
- multiple external interfaces

Eg:- Compilers, Editors and file management utilities

Application software :- Application software consists of stand alone programs to solve a specific business need.

- It facilitates business operations or management / technical decision making.

- It is used to control business functions in real time

Eg:- point of sale transaction processing, real time manufacturing process control

Engineering/scientific software :- Engineering and scientific applications range

- from astronomy to volcanology
- from automative stress analysis to space shuttle orbital dynamics
- from molecular biology to automated manufacturing

Eg:- Computer aided design, system simulation and other interactive applications

Embedded software :

- Embedded software resides within a product & system and is used to implement and control features and functions for the end user and for the system itself
- It can perform limited and esoteric functions & provide significant function and control capability

  Eg:- Digital function in automobile, dashboard display, braking system etc.

Product line software :- Designed to provide a specific capability for use by many different customers, product-line software can focus on a limited and esoteric market place & address mass of consumer markets

Eg:- word processing, spread sheets, Computer graphics multimedia, Entertainment, data base management personal and business financial applications

Web applications : web apps are Evolving into sophisticated Computing Environments that not only provide standlone features, computing functions and content to the end user but also are integrated with Corporate data bases and business applications

Artificial intelligence software : AI software makes use of non numerical algorithms to solve complex problems that are not amenable to computation or straight forward analysis. Application within this area includes robotics, expert systems, Pattern recognition, artificial neural networks theorem proving and game playing.

The following are the new challenges on the horizon:

- Ubiquitous Computing
- Net Sourcing
- Open Source
- The - new Economy

Ubiquitous Computing : The challenge for software engineers will be to develop systems and application software that will allow small devices, personal Computers and Enterprise System to Communicate across Vast networks

Net sourcing : The challenge for software Engineers is to architect simple and sophisticated applications that provide benefit to targeted end-user market world wide

Open source : The challenge for software Engineers is to build source that is self descriptive but more importantly to develop Techniques that will Enable both customers and developers to know what changes have been made and how those changes manifest themselves within the software

The new-Economy :- The challenge for software Engineers is to build applications that will facilitate mass communication and mass product distribution

SOFTWARE MYTHS

Beliefs about software and the process used to build it- Can be traced to the Earliest days of computing myths have a number of attributes that have made them insidious

management myths :- managers with software responsibility like managers in most disciplines are often under pressure to maintain budgets, Keep schedules from slipping and Improve Quality

myth :- we already have a book that's full of standards and procedures for building software - won't that provide my people with Everything they need to know

Reality: Software development book of standards may very well exist but, it is used? Are software practitioners aware of its existence? Does it reflect modern software engineering practice?

myth: If we get behind schedule, we can add more programmers and catch up

Reality: Software development is not a mechanistic process like manufacturing. As new people are added, people who were working must spend time educating the newcomers, there by reducing the amount of time spend on productive development effort. People can be added but only in a planned and well coordinated manner

myth: if I decide to outsource the software project to a third party, I can just relax and let that firm built it

Reality: If an organization does not understand how to manage and control software projects internally, it will invariably struggle when it outsources software projects

customer myths: The customer believes myths about software because software managers and practitioners do little to correct misinformation. myths lead to false expectations and ultimately, dissatisfaction with the developer

myth: A general statement of objectives is sufficient to begin with writing programs - we can fill in the details later.

Reality: Although a comprehensive and stable statement of requirements is not always possible, an ambiguous statement of objectives is recipe for disaster

myth: Project requirements continually change, but change can be easily accomodated because software is flexible

Reality: It is true that software requirements change, but the impact of change varies with the time at which it is introduced and change can cause upheaval that requires additional resources and major design modification

Practitioner's myths: myths that are still believed by software practioners: during the early days of software, programming was viewed as an art from old ways and attitudes die hard

myth: once we write the program and get it to work, our jobs are done

Reality: some one said that the sooner you begin writing code, the longer it'll take you to get done. industry data indicate that between 60 and 80 percent of all effort expended on software will be expended after it is delivered to the customer for the first time

myth: The only deliverable work product for a successfull project is the working program.

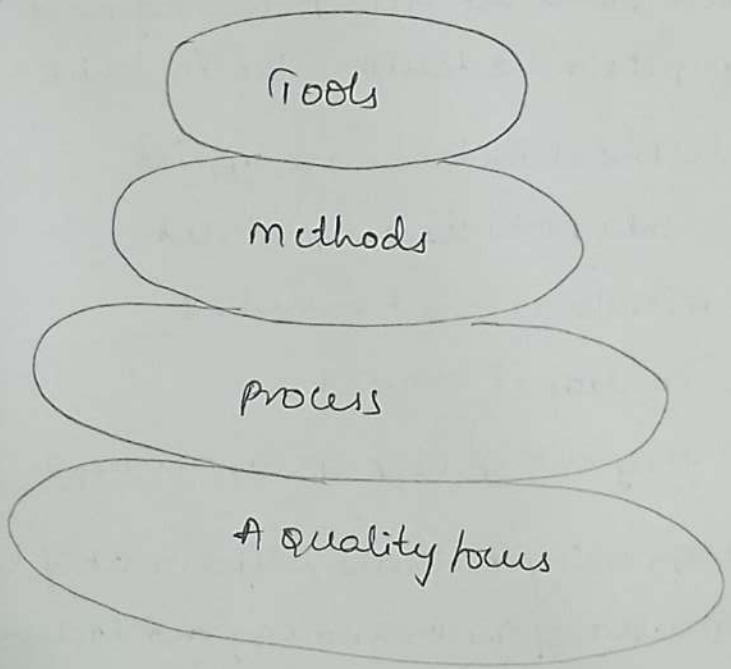
Reality: A working program is only one part of a software configuration that includes many elements. Documentation provides guidance for software support

Myth : Software Engineering will make us create voluminous and unnecessary documentation and will invariably slows down.

Reality : Software Engineering is not about creating documents it is about creating quality. Better quality leads to reduced rework. And reduced rework results in faster delivery times.

A GENERIC VIEW OF PROCESS

SOFTWARE ENGINEERING - A LAYERED TECHNOLOGY



SOFTWARE ENGINEERING

Software Engineering is a layered Technology. Any Engineering approach must rest on an organizational commitment to quality. The bed rock that supports software Engineering is a Quality focus

The foundation for software Engineering is the process layer. Software Engineering process is the glue that holds the Technology layers. process defines a frame work must be established for effective delivery of software Engineering Technology

The software forms the basis for management control of software projects and establishes the in which

- Technical methods are applied
- works products are produced
- milestones are established
- Quality is ensured
- Any change is properly managed

Software engineering methods rely on a set of basic principles that govern area of the Technology and include modeling activities

methods Encompass a broad array of Tasks that include

- Communication,
- requirements analysis,
- design modeling,
- Program Construction,
- Testing and supporting

Software Engineering tools provide automated or semi automated support for the process and the methods. When tools are Integrated so that Information created by one tool can be used by another, a system for the support of software development, called computer-aided software engineering is established.

A PROCESS FRAMEWORK :

- Software process must be established for effective delivery of software engineering Technology
- A process framework establishes the foundation for a complete software process by identifying a small number of framework activities that are applicable to all software projects, regardless of their size or complexity
- The process framework encompasses a set of umbrella activities that are applicable across the entire software process
- Each framework activity is populated by a set of Engineering actions
- Each engineering action is represented by a number of different Tasks set- Each a collection of software engineering work tasks, related work products, Quality assurance points, and project milestones.

In Brief

"A process defines who is doing what, when & how to reach a certain goal"

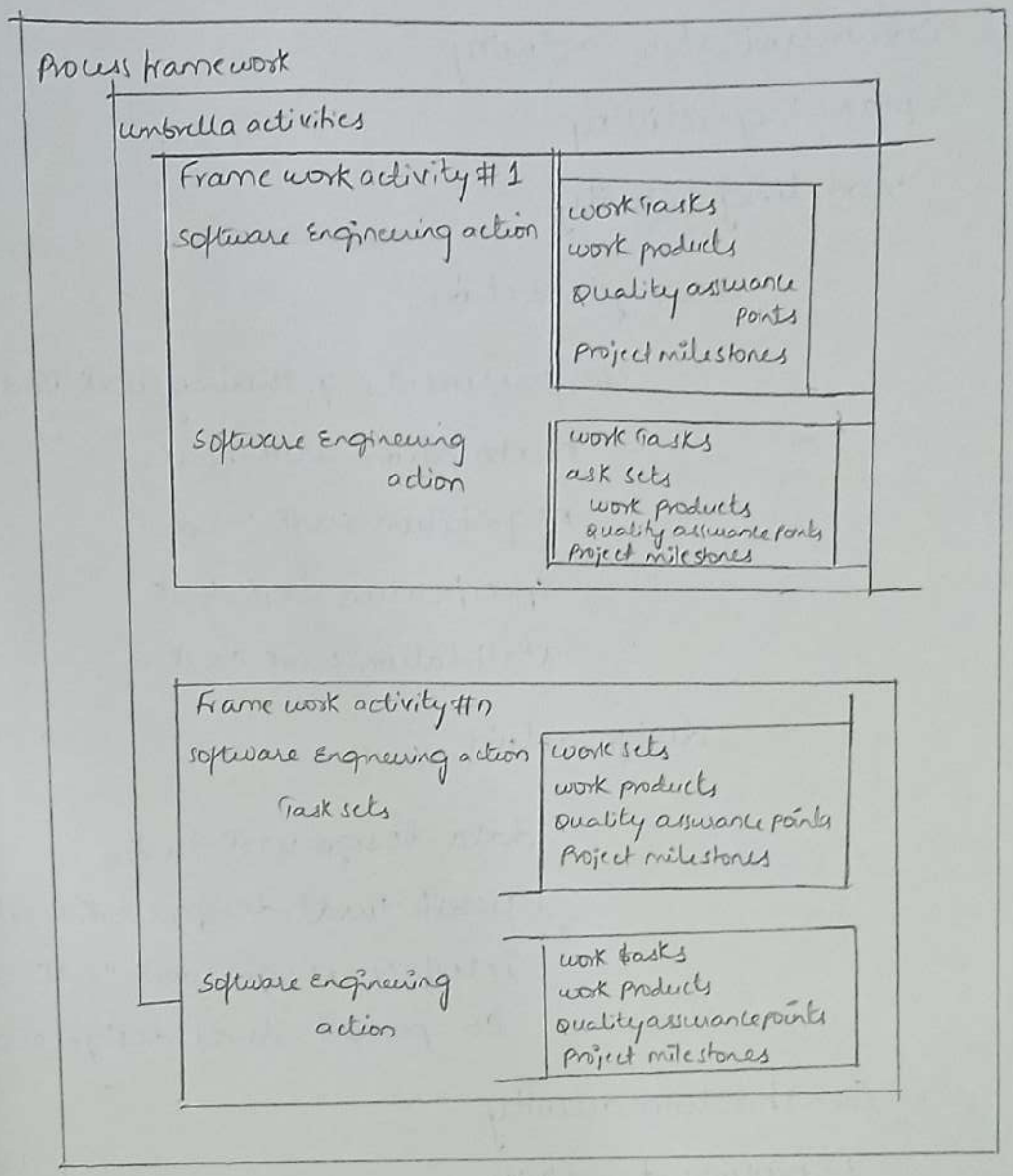
A process frame work

- Establishes the foundation for a complete software process
- identifies a small number of frame work activities
- applies to all s/w ~~objects~~ projects, regardless of size / complexity
- also, set of umbrella activities
- applicable across entire s/w process.

Each frame work activity has set of s/w Engineering actions

Each s/w engineering action (e.g. design) has

- collection of related tasks (called task sets):
  - work tasks
  - work products (deliverables)
  - Quality assurance points
  - Project milestones


Process framework
umbrella activities
Frame work activity #1
software engineering action
work tasks
work products
Quality assurance points
Project milestones
software engineering action
work tasks
task sets
work products
Quality assurance points
Project milestones
Frame work activity #n
software engineering action
Task sets
work sets
work products
Quality assurance points
Project milestones
software engineering action
work tasks
work products
Quality assurance points
Project milestones

Generic process Frame work : It is applicable to the vast majority of software projects

- Communication activity
- planning activity
- modeling activity

analysis action

requirements gathering work Task
Elaboration work Task
negotiation work Task
Specification work Task
Validation work Task

design action

data design work Task
architectural design work Task
interface design work Task
Component - level design work Task

Construction activity
Deployment activity

Communication : This frame work activity involves heavy communication and collaboration with the customer and encompasses requirements gathering and other related activities

Planning : This activity establishes a plan for the software Engineering work that follows. it describes the Technical Tasks to be conducted, the risks are that likely, the resources that will be required, the work products to be produced, and a work schedule

modeling : This activity encompasses the creation of models that allow the developer and customer to better understand software requirements & the design that will achieve those requirements. The modeling activity is composed of 2 software Engineering actions - analysis and design

Analysis encompasses a set of work tasks

Design encompasses works that create a design model

Construction : This activity combines core generation & the Testing that is required to uncover the Errors in the code

deployment : The software is delivered to the customer who evaluates the delivered product and provides feedback based on the evaluation

These 5 generic frame work activities can be used during the development of small programs, the creation of large web applications & for Engineering of large, complex computer-based systems

The following are the set of umbrella activities

Software project tracking and control - allows the software team to assess progress against the project plan and take necessary action to main schedule

Risk management - assesses risks that may effect the outcome of the project or the quality of the product

Software Quality Assurance - defines & conducts the activities required to ensure software Quality

Formal Technical Reviews - assess software engineering work products in an effort to uncover & remove errors before they are propagated to the next action or activity

measurement - define & collects process, project and product measures that assit the team in delivering software that needs customers needs, can be used in conjunction with all other frame work and umbrella activities

Software configuration management : manages the effects of change throughout the software process

Reusability management - defines criteria for work product reuse & Establishes mechanisms to achieve reusable components

work product preparation and production - Encompasses the activities required to create work products such as models, documents, logs, forms and lists.

Intelligent application of any software process model must recognize that adaption is essential for success but process models to differ fundamentally in:

- The overall flow of activities and Tasks & the inter-dependencies among activities and Tasks
- The degree through which work Tasks are defined within each frame work activity
- The degree through which work products are identified & required
- The manner which Quality assurance activities are applied
- The manner in which project tracking & control activities are applied
- The overall degree of the detailed and rigor with which the process is described
- The degree though which the customer and other stake holders are involved with the project
- The level of autonomy given to the software project team
- The degree to which team organization & roles are prescribed

THE CAPABILITY MATURITY MODEL INTEGRATION (CMMI):

The CMMI represents a process meta-model in two different ways:

As a continuous model
As a staged model

Each process area is formally assessed against specific goals and practices and is rated according to the following capability levels

level 0 : Incomplete. The process area is either not performed or does not achieve all goals and objectives defined by CMMI for level 1 capability

Level 1: performed. All of the specific goals of the process area have been satisfied. work Tasks required to produce defined work products are being conducted

level 2 : managed. A level 1 criteria have been satisfied. In addition, all work associated with the process area conforms to an organizationally defined policy; all people doing the work have access to adequate resources to get the job done; stakeholders are actively involved in the process area as required; all work Tasks and work products are monitored, controlled and reviewed;

level 3 : Defined. All level 2 criteria have been achieved. In addition, the process is Tailored from the organization set of standard processes according to the organization tailoring guidelines and contributes and work products, measures and other process-improvement information to the organizational process assets

Level 4: Quantitatively managed: All level 3 criteria have been achieved. In addition, the process area is controlled & Improved using measurement & Quantative assessment. Quantative Objectives for quality and process performance are established & used as criteria in managing the process

Level 5: Optimized: All level 4 criteria have been achieved. In addition, the process area is adapted and optimized using Quantiative means to meet changing customer needs and to continually improve the efficacy of the process area under Consideration

- The CMMI defines each process area in terms of - Specific goals and the - specific practices required to achieve these goals specific practices refine a goal into a set of process-related activities
- The specific goals (SG) and the associated specific practices (SP) defined for project planning are

SG 1 Establish Estimates

SP 1.1 Estimate the scope of the project

SP 1.2 Establish Estimates of work product & task attributes

SP 1.3 Define project life cycle

SP 1.4 Determine Estimates of Effort & Cost

SG 2 Develop a project plan

SP 2.1 Establish the budget & schedule

SP 2.2 Identify project risks

SP 2.3 plan for data management

SP 2.4 plan for needed knowledge & skills

SP 2.5 plan stakeholder involvement

SP 2.6 Establish the project plan

SG 3 Obtain Commitment to the plan

SP 3.1 Review plans that affect the project

SP 3.2 Reconcile work and resource levels

SP 3.3 Obtain plan commitment

In addition to specific goals and practices, the CMMI also defines a set of five generic goals and related practices for each process area. Each of the five generic goals corresponds to one of the five capability levels. Hence to achieve a particular capability level, the generic goal for that level and the generic practices that correspond to that goal must be achieved. To illustrate the generic goals (GG) and practices (GP) for the project planning process area are

GG 1 Achieve specific goals

GP 1.1 perform base practices

GG 2 Institutionalize a managed process

GP 2.1 Establish and organizational policy

GP 2.2 plan the process

GP 2.3 Provide resources

GP 2.4 Assign responsibility

GP 2.5 Train people

GP 2.6 manage Configuration
GP 2.7 Identify & involve relevant stakeholders
GP 2.8 monitor and control the process
GP 2.9 objectively evaluate adherence
GP 2.10 Review status with higher level management

GG3 Institutionalize a defined process

GP 3.1 Establish a defined process
GP 3.2 collect improvement information

GG4 Institutionalize a Quantitatively managed process

GP 4.1 Establish Quantitative objectives for the process
GP 4.2 stabilize sub process performance

GG5 Institutionalize and optimizing process

GP 5.1 Ensure continuous process Improvement
GP 5.2 Correct root causes of problems

PROCESS PATTERNS

The software process can be defined as a collection patterns that define a set of activities, actions, work, Tasks, work products and/or related behaviours required to develop computer software

A process pattern provides us with a template - a consistent method for describing an important characteristic of software process. A pattern might be used to describe a complete process & a Task within frame work

Pattern Name: The pattern is given a meaningful name that describes its function within the software process

Intent: The objective of the pattern is described briefly

Type: The pattern type is specified. There are three types

Task pattern: define a software engineering action or work task that is part of the process and relevant to successful software engineering practice. Ex:- Requirement gathering

Stage pattern:- define a frame work activity for the process This pattern incorporates multiple Task patterns that are relevant to the stage Ex:- Communication

Phase pattern: define the sequence of frame work activities that occur with the process, even when the overall flow of activities is iterative in nature Ex: Spiral model or prototyping

Initial content: The conditions under which the pattern applies are described prior to the Initiation of the pattern we ask

- what organizational or team related activities have already occured
- what is the entry state for the process
- what software engineering information a project information already exists

Problem: The problem to be solved by the pattern is described

solution: The Implementation of the pattern is described

This section describes how the Initial state of the process is modified as a Consequence the initiation the pattern

It also describes how software engineering information or project information that is available before the initiation of the pattern is transformed as a consequence of the successful execution of the pattern

Resulting context: The conditions that will result once the pattern has been successfully implemented are described upon completion of the pattern we ask

- what organizational or team-related activities must have occured
- what is the exit state for the process
- what software engineering information or project information has been developed?

Known uses: The specific instances in which the pattern is applicable are indicated

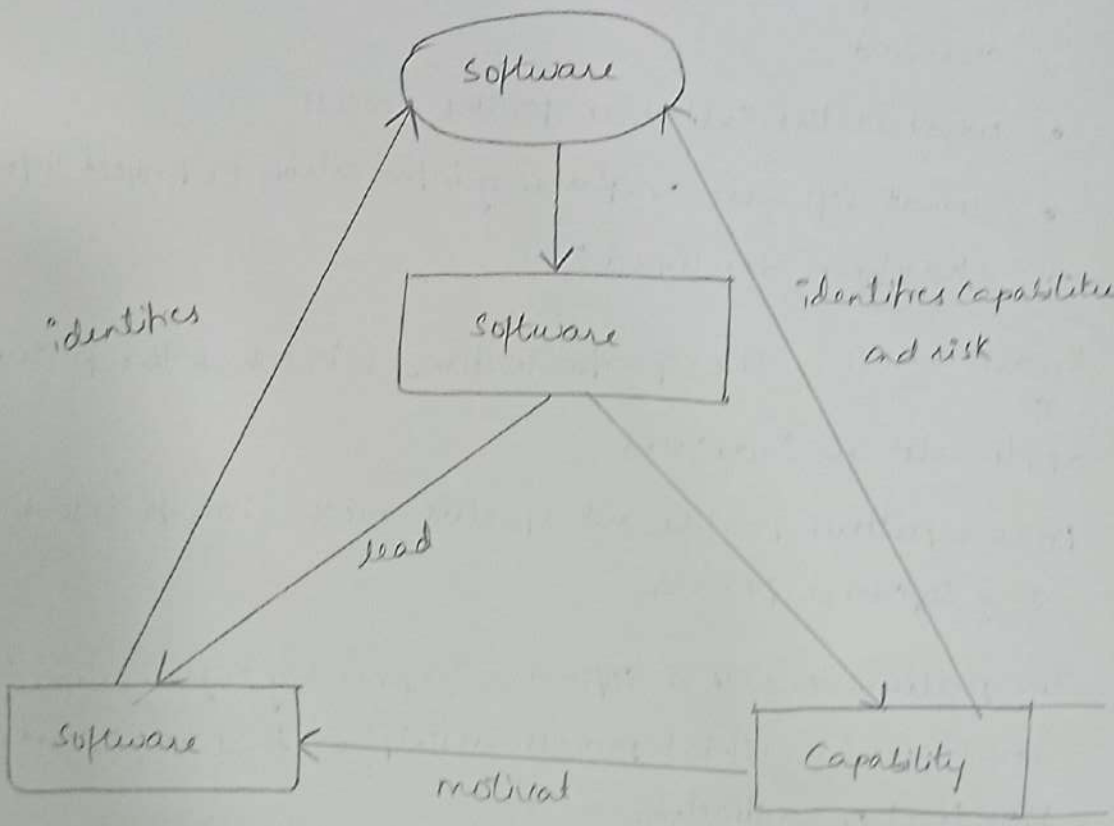
Process patterns provide and effective mechanism for describing any software process

The patterns enable a software engineering organization to develop a hierarchial process description that begins at a high level of abstraction

Once process pattern have been developed, they can be reused for the definition of process variants that is a customized process model can be defined by a software team using the pattern as building blocks for the process models

PROCESS ASSESSMENT

The existence of a software process is no guarantee that software will be delivered on time that it will meet the customer's needs, or that will exhibit the technical characteristics that will lead to long-term quality characteristics. In addition, the process itself should be assessed to be essential to ensure that it meets a set of basic process criteria that have been shown to be essential for a successful software engineering



A number of different approaches to software process assessment have been purposed over the past few decades.

Standards CMMI Assessment method for process Improvement (SCAMPI): provides a five step process assessment model that incorporates initiating, diagnosing, establishing, acting and learning. The SCAMPI method uses the SEI CMMI as the basis for assessment.

CMM Based Appraisal for Internal process Improvement (CBA IPI): provides a diagnostic technique for assessing the relative maturity of a software organization, using the SEI CMM as the basis for the assessment

SPICE: (ISO/IEC15504) standard defines a set of requirements for software process assessments. The intent of the standard is to assist organization in developing an objective evaluation of the efficiency of any defined software process

ISO 9001:2000 for Software: is a generic standard that applies to any organization that wants to improve the overall Quality of the products, system, or services that it provides. Therefore, the standard is directly applicable to software organization and companies.

PERSONAL AND TEAM PROCESS MODELS:

The best software process is one that is close to the people who will be doing the work, Each software engineer would create a process that best fits his or her needs and at the same time meets the broader needs of the team and the

organization. Alternatively, the team itself would create its own process. and at the same time meet the narrower needs of individuals and the broader needs of the organization

Personal software process (PSP): The PSP Emphasizes personal measurement of both the work product that is produced & the resultant quality of the work product

The PSP process model defines five framework activities planning, high level design, review, development and post mortem

planning: This activity isolates requirements and, base on these develops both size and resource estimates, In addition, a defect Estimate is made. All metrics are recorded on worksheets or templates. Finally, development Tasks are identified and a project schedule is created.

high level design: External specifications for each component to be constructed are developed and a component design is created. prototypes are built when uncertainity exists. All issues are recorded and tracked

high level design Review: Formal verification methods are applied to uncover errors in the design. metrics are maintained for all Important Tasks and work results

Development: The component level design is refined and reviewed. code is generated, reviewed, compiled and Tested metrics are maintained for all important task and work results

Post mortem : using the measures and metrics collected the effectiveness of the process is determined measures and metrics should provide guidance for modifying the process to improve effectiveness

Psp stresses the need for each software Engineer to identify errors early and, as important, to understand the types of errors that he is likely to make

Psp represents a disciplined, metrics-based approach to software engineering

Team software process (TSP): The goal of TSP is to build a self directed project team that organizes itself to produce high-quality software. The following are the objectives for TSP:

Build self-directed teams that plan and track their work, establish goals and own their processes & plans These can be pure software teams or integrated product teams (IPT) of 3 to about 20 engineers

show managers how to coach and motivate their teams and how to help them sustain peak performance

Accelerate software process improvement by making CMM level 5 behaviour normal and expected

Provide improvement guidance to high-maturity organizations

Facilitate university teaching of industrial-grade team skills. self directed team defines

Roles and Responsibilities for each team member

Tracks quantiative project data

identifies a team process that is appropriate for the project

a strategy for implementing the process

defines local standards that are applicable to the teams software engineer work;

continually assesses risk and reacts to it

Tracks, manages and reports project status

TSP defines the following frame work activities: launch, high-level design, implementation, integration and test and post mortem

TSP makes use of a wide variety of scripts, forms, and standards that serve to guide team members in their work

Scripts define specific process activities and other more detailed work functions that are part of the team process

Each project is - launched using a sequence of Tasks

The following launch script is recommended

- Review project ~~(or)~~ objectives with management and agree on and document team goals
- Establish team roles
- Define the teams development process
- make a quality plan and set quality Targets
- plan for the needed support facilities

PROCESS MODELS

Prescriptive process models define a set of activities, actions Tasks, milestones, and work products that are required to Engineer high-Quality Software. These process models are not perfect, but they do provide a useful roadmap for Software Engineering work

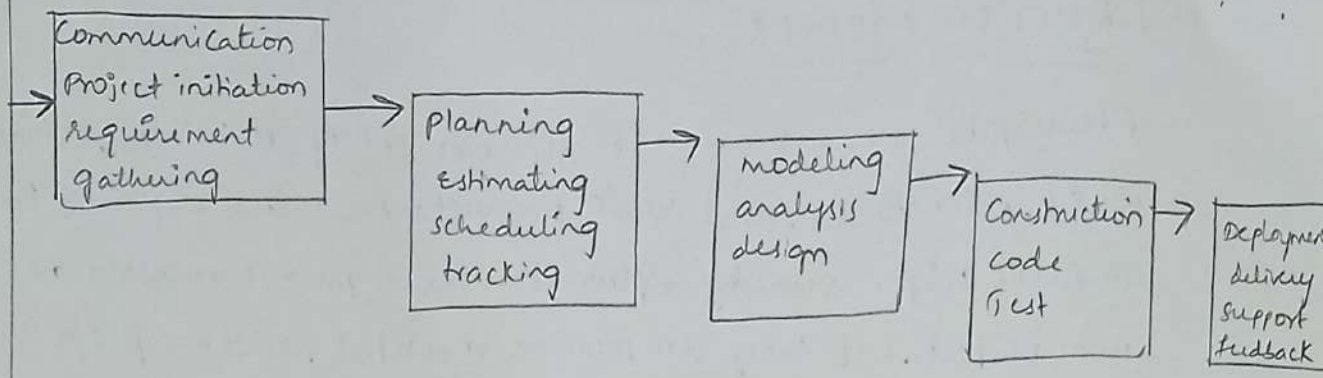
A prescriptive process model populates a process frame work with Explicit Task sets for Software Engineering actions

THE WATERFALL MODEL

The waterfall model, Sometimes called the classic life cycle, Suggests a systematic sequential approach to Software development that Begins with customer Specification of requirements and progresses through planning, modelling, construction and deployment

Context: used when requirements are reasonably well understood.

Advantage: It can serve as a useful process model in situations where requirements are fixed and work is to complete in a linear manner



The problems that are sometimes encountered when the waterfall model is applied are :

Real projects rarely follow the sequential flow that the model purposes. Although the linear model can accomodate iteration, it does so indirectly. As a result, changes can cause confusion as the project team proceeds

It is often difficult for the customer to state all requirements explicitly. The waterfall model requires this & has difficulty accomodating the natural uncertainty that exist at the beginning of many projects.

The customer must have patience. A working version of the programs will not be available until late in the project time-span. If a major blunder is undetected then it can be disastrous until the program is reviewed

INCREMENTAL PROCESS MODELS:

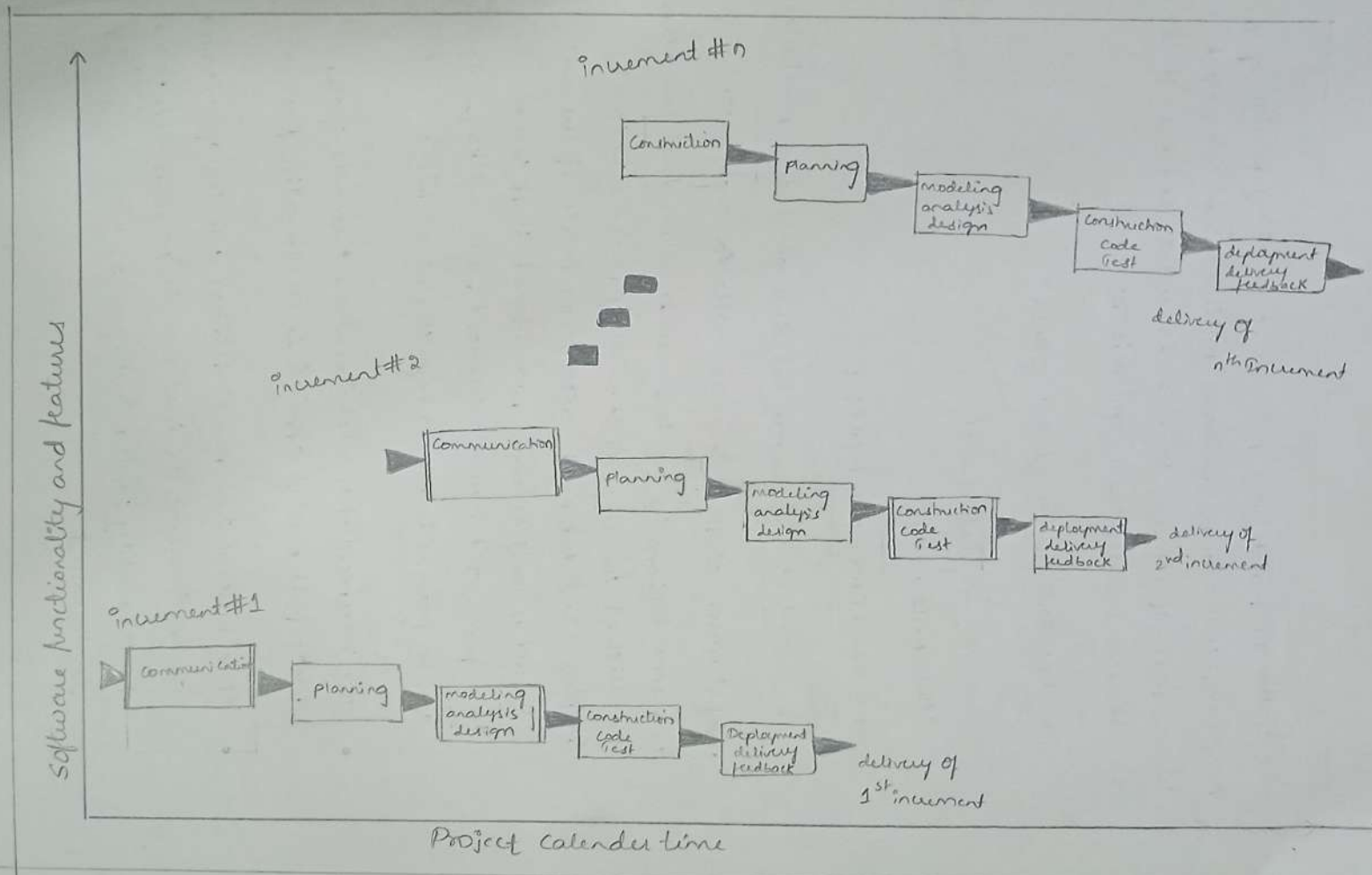
The incremental model

The RAD model

THE INCREMENTAL MODEL:

Context: Incremental development is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project. Early increments can be implemented ~~(the next)~~ with fewer people. If the core product is well received, additional staff can be added to implement the next increment. In addition, increments can be planned to manage technical risks

- The incremental model combines elements of the waterfall model applied in an iterative fashion
- The incremental model delivers a series of releases called increments that provide progressively more functionality for the customer as each increment is delivered


Software functionality and features
Project calender time
increment #n
Construction
Planning
modeling analysis design
Construction code Test
deployment delivery feedback
delivery of nth increment
increment #2
Communication
Planning
modeling analysis design
construction code Test
deployment delivery feedback
delivery of 2nd increment
increment #1
Communication
Planning
modeling analysis design
construction code Test
Deployment delivery feedback
delivery of 1st increment

- When an increment model is used, the first increment is often a core product. That is basic requirements are addressed. The core product is used by the customer As a result, a plan is developed for the next increment
- The plan addresses the modification of the core product to better meet the needs of the computer and the delivery of additional features and functionality

This process is repeated following the delivery of each increment, until the complete product is produced

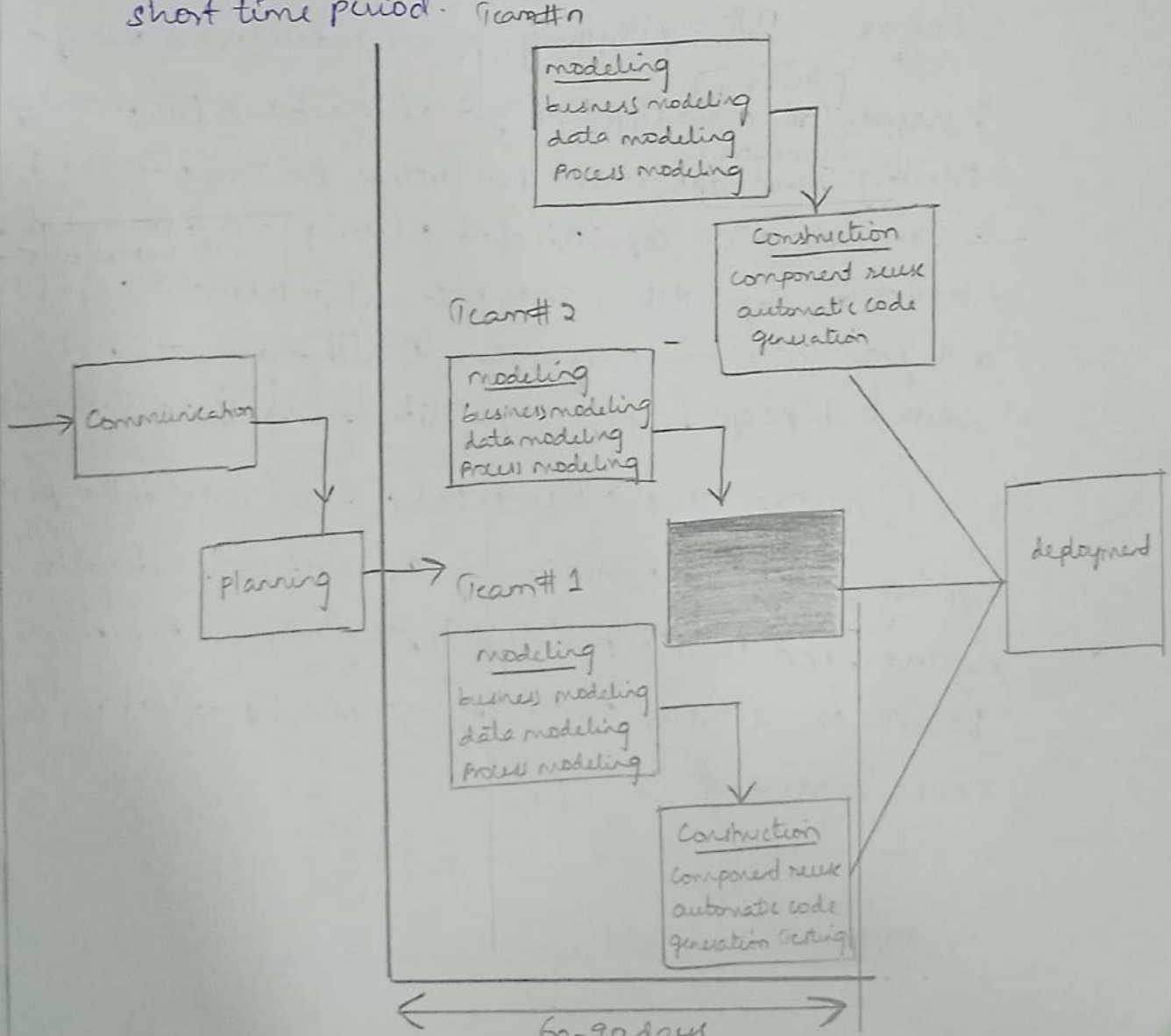
For Ex:- word processing software developed using the incremental paradigm might deliver basic file management Editing, and document production functions in the first increment; more sophisticated editing, and document production capabilities in the second increment; spelling and grammar checking in the third increment and advanced page layout capability in the fourth increment

Difference: The incremental process model, like prototyping and other evolutionary approaches is iterative in nature. But unlike prototyping, the incremental model focuses on delivery of an operational product with each increment

THE RAD MODEL :

Rapid application development (RAD) is an incremental software process model that Emphasizes a short development cycle. The RAD model is a high speed adaption of the waterfall model, in which rapid model development is achieved by using a component base construction approach

Context : if requirements are well understood and project scope is constrained, the RAD process enables a development team to create a fully functional system within a very short time period.



The RAD approaches maps into the generic framework activities

Communication: works to understand the business problem and the information Techniques that the software must accomodate

Planning: is essential because multiple software systems teams works in parallel on diffrent system functions

modeling: Enlompasses three major phases - business modeling, data modeling and process modeling. and establishes design representation that serve existing software components and the application of automatic code generation

deployment: Establishes a basis for subsequent iterations. The RAD approach has drawbacks:

For large, but scalable projects, RAD requires sufficient human resources to create the right number of RAD Teams

If a system cannot be properly modularized, building the components neleisary for RAD will be problematic

RAD may not be appropriate when Technical risks are high

EVOLUTION PROCESS MODELS :

Evolutionary process models produce with each iteration Produce an increasingly more complete version of the software with every iteration

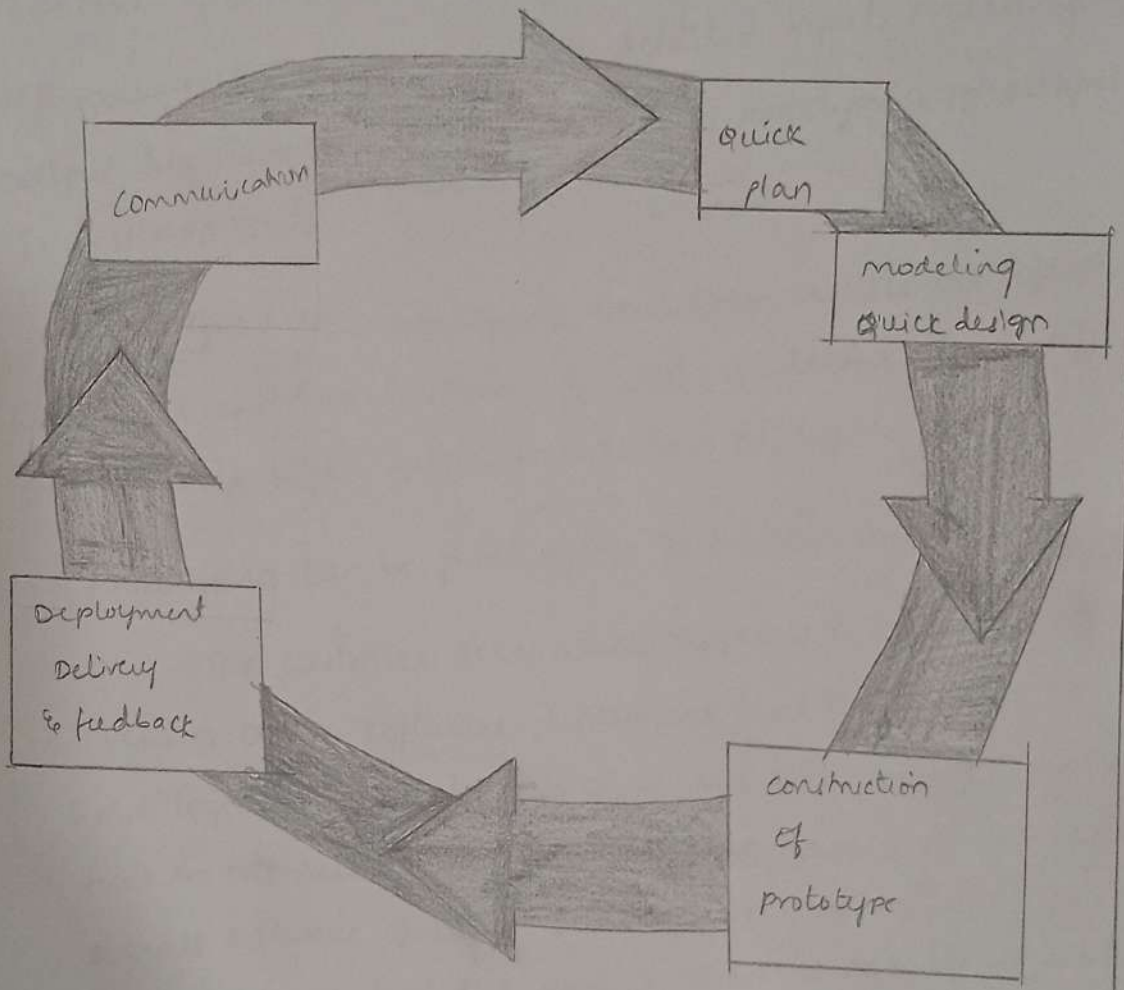
Evolutionary models are iterative. They are characterized in a manner that enables software engineers to develop increasingly more complete versions of the software

PROTOTYPING:

Prototyping is more commonly used as a Technique that can be implemented within the context of anyone of the process model

The prototyping paradigm begins with communication. The software engineer and customer meet and define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory.

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


Communication
Quick plan
modeling Quick design
construction of prototype
Deployment Delivery & feedback

Context: if a customer defines a set of general objectives for software, but does not identify detailed input, processing or output requirements, in such situation prototyping paradigm is best approach

Advantages: The prototyping paradigm assists the software engineer and the customer to better understand what is to be built when requirements are hazy

Prototyping can be problematic for the following reasons

The customer sees what appears to be a working version of the software, unaware that the prototype is held together - with chewing gum and bailing wire, unaware that in the rush to get it working we haven't considered overall software quality or long term maintainability. When informed that the product must be rebuilt so that high levels of quality can be maintained, the customer cries foul of demands that - a few tixes be applied to make the prototype a working product. Too often, software development relents

The developer often makes implementation compromises in order to get a prototype working quickly. An inappropriate operating system or programming language may be used simply because it is available & known; an inefficient algorithm may be implemented simply to demonstrate capability. After a time, the developer may become comfortable with these choices and forget all the reasons why they

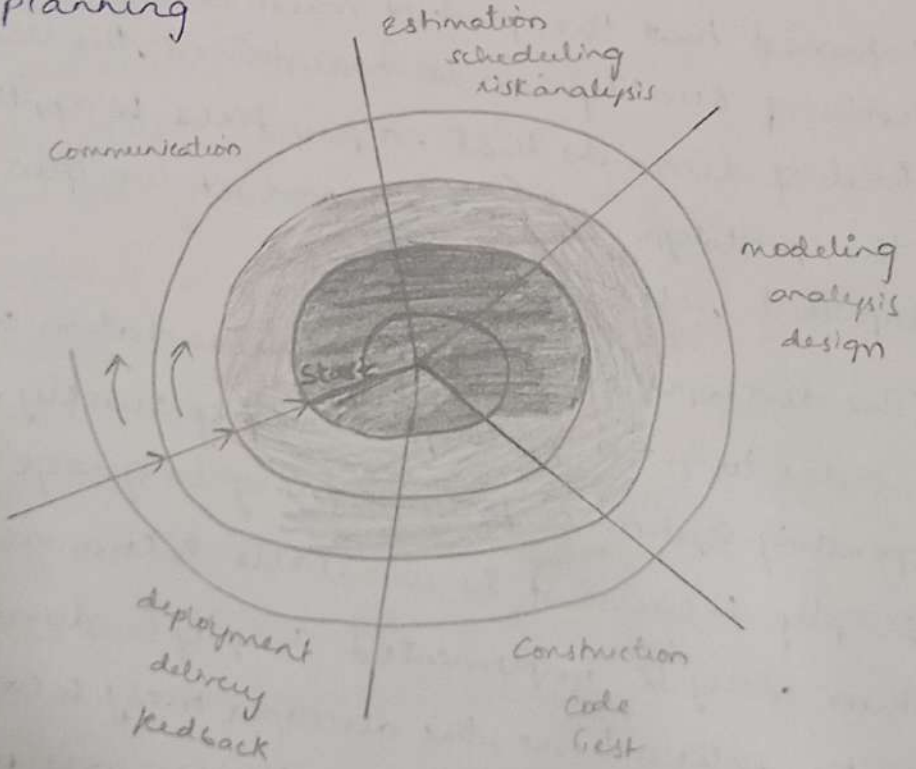
were inappropriate. The less than-ideal choice has now become an integral of the system

THE SPIRAL MODEL

The spiral model, originally proposed by Boehm, is an Evolutionary software process model that couples the iterative nature of prototyping with the controlling and systematic aspects of the waterfall model

The spiral model can be adapted to apply throughout the entire life cycle of an application from concept development to maintenance

using the spiral model, software is developed in a series of Evolutionary releases. during early iterations, the release might be a paper model or prototype. During later iterations increasingly more complete versions of engineered systems are planning



Spiral, when characterized ~,

Anchor point milestones : A combination of work products and conditions that are attained along the path of the spiral - are noted for each evolutionary pass.

The first circuit around the spiral might result in the development of product specifications subsequent products around the spiral might be used to develop a prototype and then progressively more sophisticated versions of the software

Each pass through the planning region results in adjustments to the project plan. Cost and schedule are adjusted based on feedback derived from the customer after delivery. In addition, the project manager adjusts the planned number of Iteration required to complete the software

It maintains the systematic stepwise approach suggested by the classic life cycle but incorporates it into an iterative frame work that more realistically reflects the real world

The first circuit around the Signal might represents a concept development project which starts at the core of the spiral and continues for multiple Iterations until concept development is complete.

if the concept is to be developed into an actual product - the process proceeds outward on the spiral and a new product development project commences

later, a circuit around the spiral might be used to represent a product enhancement project in essence, the spiral, when characterized in this way, remains operative

until the software is retired

Context: The spiral model can be adopted to apply throughout the entire life cycle of an application, from concept development to maintenance

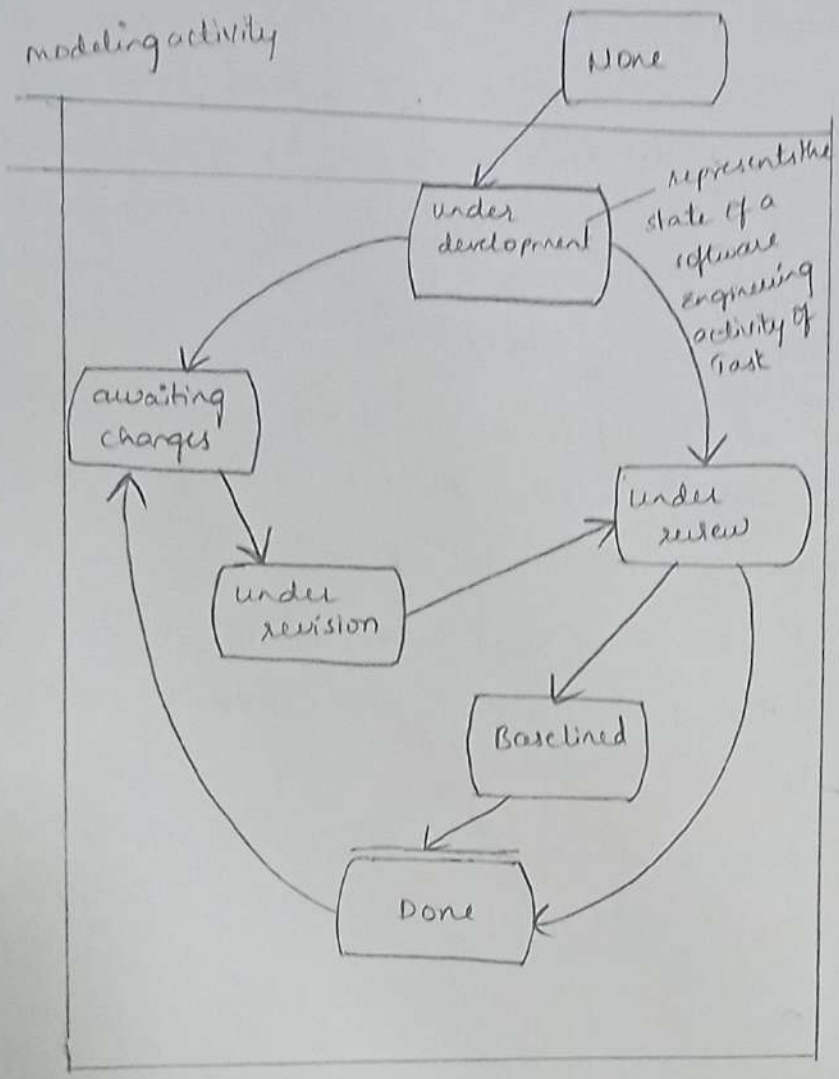
Advantages: It provides the potential for rapid development of increasingly more complete versions of the software

The spiral model is a realistic approach to the development of large-scale systems and software. The spiral model uses prototyping as a risk reduction mechanism but, more importantly enables the developer to apply the prototyping approach at any stage in the evolution of the product.

Drawbacks: The spiral model is not a panacea. It may be difficult to convince customers that the evolutionary approach is controllable. It demands considerable risk assessment Expert sc and relies on this expertise for success if a major risk is not uncovered and managed, problems will undoubtedly occur

THE CONCURRENT DEVELOPMENT MODEL:

The concurrent development model, sometimes called Concurrent engineering, can be represented schematically as a series of framework activities, software engineering actions and tasks, and their associated states



The activity modeling may be in anyone of the states noted at any given time. similarly other activities or tasks can be represented in an analogous manner. All activities exist concurrently but reside in different states

Any of the activities of a project may be in a particular state at any one time

under development
awaiting changes
under revision
under review

In a project the communication activity has completed its first iteration and exists in the awaiting change

state. The modeling activity which existed in the none state while Internal communication was completed, now makes a transition into the under development state if, however, the customer indicates the changes in requirements must be made the modeling activity moves from the under development state into the awaiting changes state

The concurrent process model defines a series of events that will trigger transitions from state to state for each of the software engineering activities, actions or tasks.

The event analysis model correction which will trigger the analysis action from the done state into the awaiting changes state

Context: The concurrent model is often more appropriate for system engineering projects where different engineering teams are involved

Advantages:

The concurrent process model is applicable to all types of software development and provides an accurate picture of the current state of a project

it defines a network of activities rather than each activity action, or task on the network exists simultaneously with other activities, action and tasks

A FINAL COMMENT ON EVOLUTIONARY PROCESSES:

The concerns of evolutionary software processes are:

The first concern is that prototyping poses a problem to project planning because of the uncertain number of cycles required to construct the product

Second, evolutionary software process do not establish the maximum speed of the evolution. if the evolution occurs too fast, without a period of relaxation, it is certain that the process will fall into chaos

Third, software processes should be focused on flexibility and extensibility rather than on high quality

THE UNIFIED PROCESS:

The unified process is an attempt to draw on the best features and characteristics of conventional software process models, but characterize them in a way that implements many of the best principles of agile software development

The unified process recognizes the importance of customer communication and streamlined methods for describing the customers view of a system. it emphasizes the important role of software architecture and helps the architect focus on the right goals, such as understandability, reliance to future changes and reuse - it suggests a process flow that is iterative and incremental, providing the evolutionary feel that is essential in modern software development.

A BRIEF HISTORY

During the 1980's & into early 1990's, Object Oriented methods and programming languages gained a widespread audience throughout the software engineering community. A wide variety of Object-Oriented Analysis (OOA) and design (OOD) methods were proposed during the same time period.

During the early 1990's James Rumbaugh, Grady Booch, and Ivar Jacobson began working on a unified method that would combine the best features of each of OOD & OOA. The result was UML - a unified modeling language that contains a robust notation for the modeling and development of OO systems.

By 1997, UML became an industry standard for Object-Oriented software development. At the same time, the Rational corporation and other vendors developed automated tools to support UML methods.

Over the next few years, Jacobson Rumbaugh and Booch developed the unified process, a frame work for Object-Oriented software engineering using UML. Today, the unified process and UML are widely used on OO projects of all kinds. The iterative, incremental model proposed by the UP can and should be adopted to meet specific project needs.

PHASES OF THE UNIFIED PROCESS

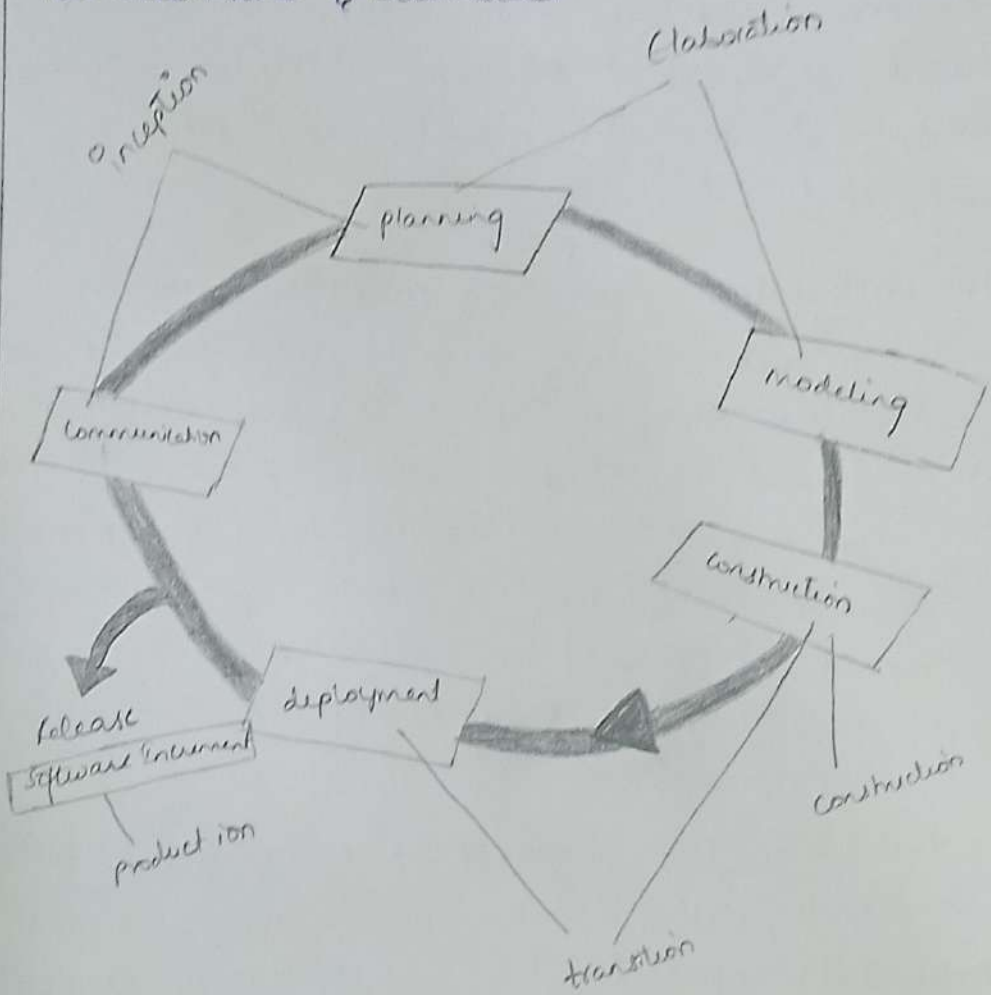
The Inception phase of the UP encompasses both customer communication and planning activities. By collaborating with the customer and end-users, business requirements for the software are identified, a rough architecture for the system is proposed and a plan for the iterative, incremental nature of ensuing project is developed.

The Elaboration phase encompasses the customer communication and modeling activities of the generic process model. Elaboration refines and expands the preliminary use-cases that were developed as part of the inception phase and expands the preliminary use-cases that were developed as a phase & expands the architectural (model as input) representation to include five different views of the software - the use case model the analysis model, the design model, the implementation model & the deployment model

The construction phase of the UP is identical to the construction activity defined for the generic software process. Using the architectural model as input, the construction phase develops or acquires the software components that will make each use-case operational for end users. To accomplish this, analysis and design models that were started during the elaboration phase are completed to reflect the final version of the software increment

The transition phase of the UP encompasses the latter stages of the generic construction activity & the first part of the generic deployment activity. Software given to end-users for beta testing and user feedback reports both defects and necessary changes

The production phase of the UP coincides with the deployment activity of the generic process. During this phase the on-going use of the software is monitored, support for the operating environment is provided, & defect reports & requests for changes are submitted & evaluated



A Software Engineering work flow is distributed across all UP phases. In the context of UP, a work flow is analogous to a Task Test. That is, a work flowing identifies the tasks required to accomplish an Important software Engineering action & the work products that are produced as a Consequence of successfully completing the tasks

UNIFIED PROCESS WORK PRODUCTS

During the Inception phase, the Intent is to Establish an Overall - vision for the project,

identify a set of business requirements
make a business case for the software, and
define project and business risks that may represent a threat to success

The most Important work product produced during the Inception is the use-case model - a collection of use-cases that describe how outside actors interact with the system and gain value from it. The use-case model is a collection of software features and functions by describing a set of preconditions, a flow of Events & a set of post-conditions for the Interaction that is depicted.

The use-case model is refined & elaborated as Each UP phase is conducted and serves as an important input for the creation of subsequent work products. During the inception phase only 10 to 20 Percent of the use-case model is completed. After Elaboration, s/w 80 to 90 percent of the model has been created

The Elaboration phase produces a set of work products that Elaborate requirements and produce and architectural description and preliminary design. The UP analysis model is the work product that is developed as a consequence of this activity. The classes and analysis package defined as a part of the analysis model are refined further into a design model which identifies design classes subsystems and interfaces b/w subsystems. Both the analysis & design models expand & refine an evolving representation of software architecture. In addition the Elaboration phase revisits risks and the project plan to ensure that each remains valid

The construction phase produces an implementation model that translates design classes into software components into the physical computing environment. Finally, test model describes tests that are used to ensure that use cases are properly reflected in the software that has been constructed

The transition phase delivers the software increment and assess work products that are produced as end-users work with the software. Feedback from beta testing & qualitative requests for change is produced at this time

Inception phase

Vision document
initial use-case model
initial project glossary
initial business case
initial risk assessment
Project plan, phases & iterations
Business model, if necessary one or more prototypes

Elaboration phase

use-case model
supplementary requirements including non-functional Analysis model software architecture description
Executable architectural prototype
Preliminary design model revised risk list
Project plan including iteration plan adapted workflow milestones
technical work products
Preliminary user manual

Construction in phase

Design model
software components
integrated software increment
Test plan and procedure Test cases
support documentation user manuals
installation manuals
description of current increment

Transition phase

Delivered software increment
Beta Test reports General user feedback