WHAT IS COMPUTER SOFTWARE?

Software is:

(1) instructions (computer programs) that when executed provide desired features, function, and performance;
(2) data structures that enable the programs to adequately manipulate information and (3) documentation that describes the operation and use of the programs.

CHARACTERISTICS OF A SOFTWARE:

1. Software is developed or engineered and not manufactured.
2. Software doesn't wear out
3. Most Software is custom built despite approaches like component based construction.

It is a product that software professionals build and maintain over a long period of time.
It is built using a software engineering approach which consists of an agile , adaptable process to build a high quality product.
The software should be high quality, easy to use, faster to build and less·expensive.

WHAT IS A WORK PRODUCT?

A work product is a report, diagram, or collection of documents used by the business analyst
during the requirements development process. A work product can be used to share information with stakeholders, elicit requirements, provide status, etc.Examples of a work product may include: Meeting minutes ,Diagrams Recorded discussions ,Status reports ,Presentations, Prototypes

WHAT IS SOFTWARE ENGINEERING ?

IEEE definition: Software Engineering:
(1) The application of a systematic, disciplined, quantifiable approach to the development,
operation, and maintenance of software; that is, the application of engineering to software.
(2) The study of approaches as in (1).
Definition 2: Software engineering is a process , a set of methods and array of tools to build a
software.

THE EVOLVING ROLE OF SOFTWARE

Today, software takes on a dual role. It is a product and, at the same time, the vehicle for delivering a product.
 As a product, it delivers the computing potential embodied by computer hardware or,more broadly, a network of computers that are accessible by local hardware. Whether it resides within a cellular phone or operates inside a mainframe computer,
 software is an information transformer—producing, managing, acquiring, modifying,

displaying, or transmitting information that can be as simple as a single bit or as complex as a multimedia presentation.

 As the vehicle used to deliver the product, software acts as the basis for the control of the computer (operating systems), the communication of information (networks), and the creation and control of other programs (software tools and environments). Software delivers the most important product of our time—information.

 Software transforms personal data (e.g., an individual's financial transactions) so that the data can be more useful in a local context; it manages business information to enhance competitiveness; it provides a gateway to worldwide information networks (e.g., Internet)
and provides the means for acquiring information in all of its forms.

 The role of computer software has undergone significant change over a time span of little more than 50 years. Dramatic improvements in hardware performance, profound
changes in computing architectures, vast increases in memory and storage capacity, and a wide variety of exotic input and output options have all precipitated more sophisticated and complex computer-based systems.

 The lone programmer of an earlier era has been replaced by a team of software specialists, each focusing on one part of the technology required to deliver a complex application.

 And yet, **the same questions asked of the lone programmer are being asked when**
modern computer-based systems are built:

1)Why does it take so long to get software finished?

2) Why are development costs so high?

3) Why can't we find all the errors before we give the software to customers?

4) Why do we continue to have difficulty in measuring progress as software is being
developed?

Characteristic of software:

There is some characteristic of software which is given below:

1. Functionality
2. Reliability
3. Usability
4. Efficiency
5. Maintainability
6. Portability

CHANGING NATURE OF SOFTWARE

The software is an instruction or computer program that when executed provides desired features, function, and performance. A data structure that enables the program to adequately manipulate information and document that describes the operation and use of the program.

Changing Nature of Software:

Nowadays, seven broad categories of computer software present continuing challenges for software engineers .which is given below :

1. **System Software:**

System software is a collection of programs which are written to service other programs.Some system software processes complex but determinate, information structures other system application processes largely indeterminate data. Sometimes the system software area is characterized by the heavy interaction with computer hardware that requires scheduling, resource sharing, and sophisticated process management.

Examples of system software include operating systems like macOS, GNU/Linux and Microsoft Windows, computational science software, game engines, industrial automation, and software as service applications.

2. **Application Software:**

Application software is defined as programs that solve a specific business need.Application in this area processes business or technical data in a way that facilitates business operation or management technical decision making. In addition to convention data processing application, application software is used to control business functions in real time.

Examples of applications include word processors, database programs, web browsers,development tools, image editors and communication platforms. Applications use the computer's operating system (OS) and other supporting programs, typically system software, to function.

3. **Engineering and Scientific Software:**

This software is used to facilitate the engineering function and task, however modern application within the engineering and scientific area is moving away from the conventional numerical algorithms. Computer-aided design, system simulation, and other interactive applications have begun to take real-time and even system software characteristics.

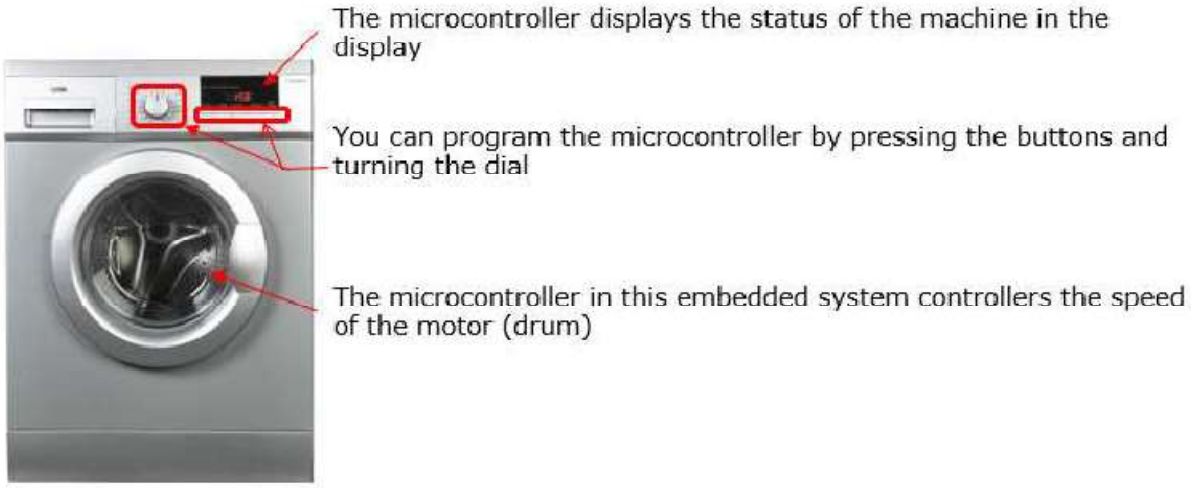
Examples are software like MATLAB, AUTOCAD, PSPICE, ORCAD, etc.

4. **Embedded Software:**

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

Example: Motion detection systems in security cameras.

This washing machine has an embedded system in it.



5. **Product-line Software:**

Software Product Lines · Any organization that develops software creates multiple software applications that have some characteristics in common. ·Some software has the same application architecture, some run on the same execution platforms, and others support the same segment of the business.

Designed to provide a specific capability for use by many different customers, product line software can focus on the limited marketplace or address the mass consumer market.

Examples of Product Lines *Microsoft Corporation (MSFT) as a brand sells several highly recognized product lines including Windows, Office, and the Xbox. Nike Inc. (NKE)*

6. **Web Application:**

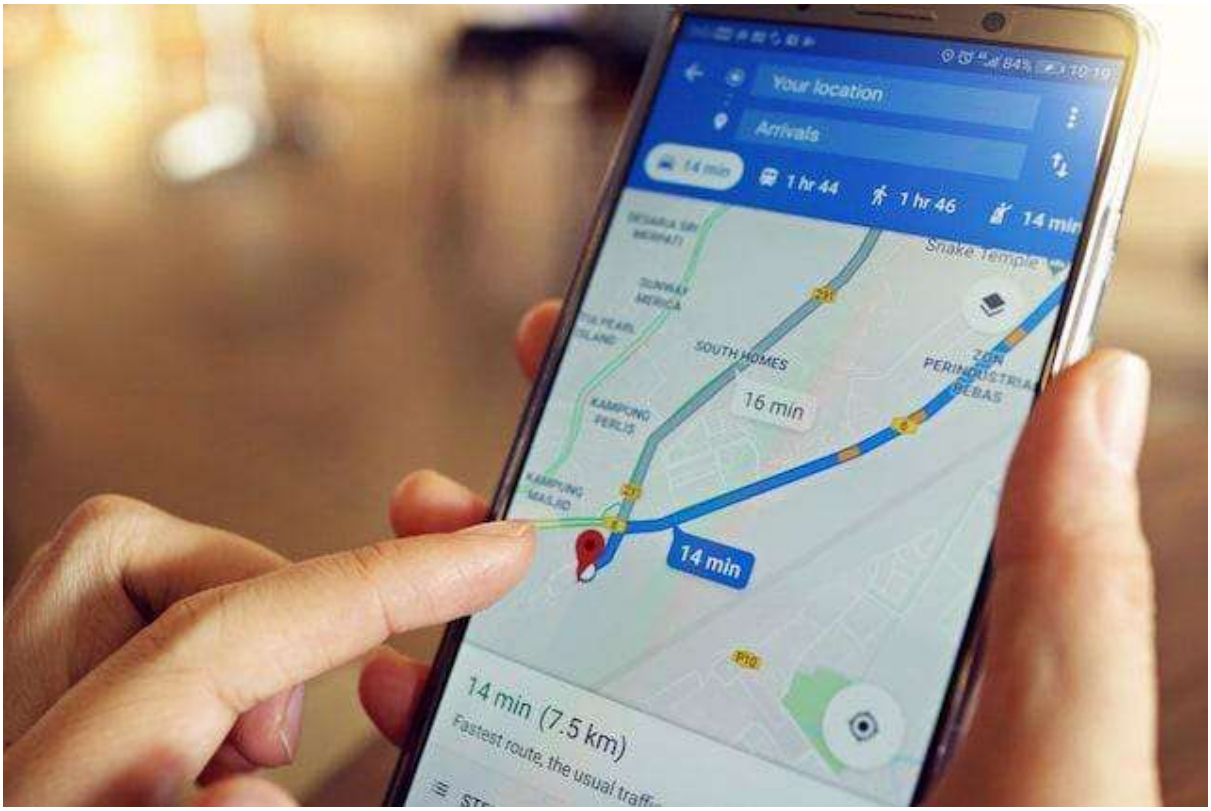
It is a client-server computer program which the client runs on the web browser. In their simplest form, Web apps can be little more than a set of linked hypertext files that present information using text and limited graphics. However, as e-commerce and B2B applications grow in importance. Web apps are evolving into a sophisticated computing environment that not only provides a standalone feature, computing function, and content to the end user.

Example: Quora is web-based software, for example. So is Facebook, Google, eBay, Amazon, etc. If normal websites that can be accessed from browser

7. **Artificial Intelligence Software:**

Artificial intelligence software makes use of a non numerical algorithm to solve a complex problem that is not amenable to computation or straightforward analysis. Application within this area Includes robotics, expert system, pattern recognition,artificial neural network, theorem proving and game playing.

Examples :Smart Phones, Smart Home, Smart Cars , Alexa Etc



CHALLENGES FOR SOFTWARE ENGINEERS:

Ubiquitous computing: Developers have to build software that allows PC’ s, small devices, enterprise systems to communicate across vast networks all around the world. [*Ubiquitous computing (or "ubicomp") is a concept in software engineering and computer science where computing is made to appear anytime and everywhere. In contrast to desktop computing, ubiquitous computing can occur using any device, in any location, and in any format*.

Some of the examples are: Apple Watch, Electronic Toll Systems, Smart Traffic Lights, Self Driving Cars, Home Automation.]

Net sourcing: software designed must be simple and for targeted end user market. Net sourcing is the pattern of leasing or “paying as you use” entree to supplier-managed concern applications, made available to multiple clients over the Internet or other webs

Open source: The software today needs to be open source (Open-source software is a type of computer software in which source code is released under a license in which the copyright holder grants users the rights to use, study, change, and distribute the software to anyone

and for any purpose. Open-source software may be developed in a collaborative public manner.)

New Economy: The software must be able to communicate with other existing software and must be able to be distributed across networks.

Software must be built within schedule, cost and resources

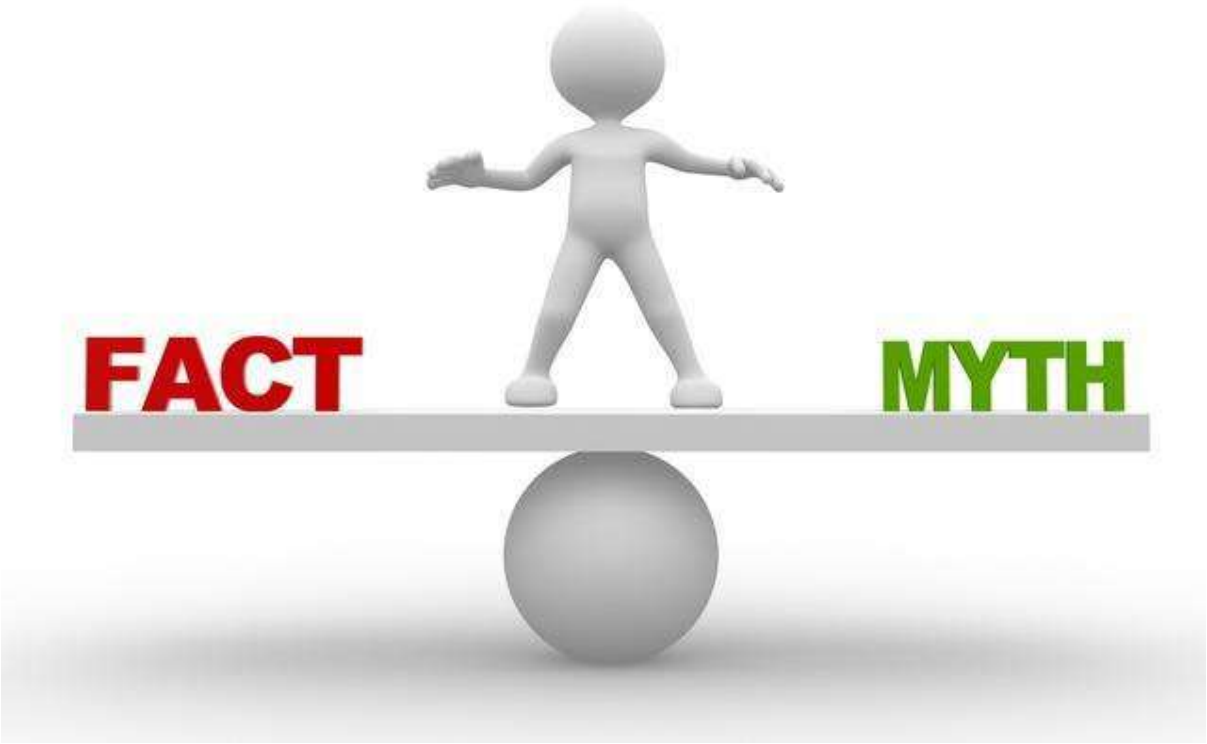
Software must be high quality and maintainable

What is legacy software?

 Legacy software is software that has been around a long time and still fulfills a business need. It is mission critical and tied to a particular version of an operating system or hardware model (vendor lock-in) that has gone end-of-life. Generally the lifespan of the hardware is shorter than that of the software. As time goes on, the hardware gets harder to maintain but is kept because it is installed and (for now) working and has proven too complex and/or expensive to replace

An example of legacy software is a factory's computer system running on an old version of Windows because there is not a need to invest in the most updated software.

SOFTWARE MYTHS: (a widely held but false belief or idea)



 Software Development Myths. Pressman (1997) describes a number of common beliefs or myths that software managers, customers, and developers believe falsely. He describes these myths as ``misleading attitudes that have caused serious problems.

 1) Management myths :

 a) Myth: We already have a book full of standards and procedures for building software's. Isn't that enough?

 Reality: Although a standard operating procedure book is available, but it is hardly used and adapted.

 b) Myth: If we get behind schedule, we can add up more programmers and catch up.
 Reality: New programmers must be trained by existing programmers about the project and hence more time and effort is required.
 c) Myth: If we outsource the software project to a third party, we can relax.
 Reality: If it is difficult to control and manage the software by the organization itself, then there will be issues managing it by third party outsourcing also.

2) Customer Myths:

a) Myth: A general statement of objectives is sufficient to begin writing programs and details can be filled later.

Reality: If we begin to develop with outline requirements, then it will require a lot of changes in the code later when details in requirements are added later

b) Myth: Project requirements change continually, but changes can be accommodated easily because software is flexible.

Reality: The earlier the changes are made, it is better. Because changes in later stages of
development will cost a lot of effort, time and resources.

3) Practitioner's Myths:

a) Myth: Once we write a program and get it to work, our job is done.

Reality: More effort is required to maintain the software after it is delivered.

b) Myth: Until I get the program running, there is no way of assessing its quality.

Reality: Software can be reviewed early in the phases during requirement engineering itself.

c) Myth: The only deliverable work product for a project is the working program.

Reality: Other work products like documentation, system models are also important support for
software development.

d) Myth: Software engineering will make us create too much documentation and slow down the project.

Reality: Documentation helps to improve quality and avoid rework.

HOW DOES SOFTWARE PROJECT STARTS:

A software project starts with a business need to adapt a legacy software, correct errors in existing software ,adding new features / functions, new product / service / system, Initially the
business needs are informally expressed and later these are revised and expressed in more technical terms.

A GENERIC VIEW OF PROCESS

In order to build software that meets the challenges of the twenty first century, a few simple realities must be recognized:

 A concerted effort should be made to understand the problem because every before a software solution is developed Because stakeholders have a different idea of what features and functions are to be developed.

 Design is a pivotal software engineering activity because today software is embedded in almost all systems (medical, weapons, electronic devices etc) . hence it is complex and has to interact with all system elements. The software should exhibit high quality because today software is used by individuals,business and governments in day to day lives. Hence its failure can cause inconvenienceto catastrophic problems.

 Software should be maintainable and enhancements. because It must adapt to changes

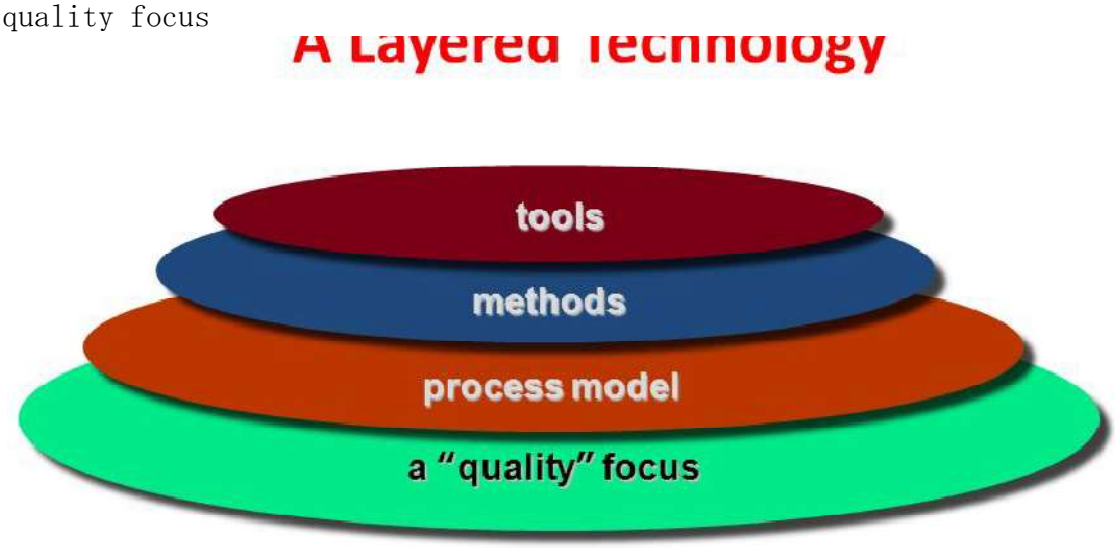
EXPLANATION OF Software Engineering - A Layered Technology

Fritz Bauer defined Software engineering as the "establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines."

Software engineering encompasses a process, the management of activities, technical methods, and use of tools to develop software products.

.Software Engineering is a layered technology as shown below.

Quality focus : Any engineering approach must rest on an organizational commitment to quality. The bedrock that supports software engineering is a quality focus



The foundation for S/W eng is the ***process*** **layer.** It is the glue that holds the technology layers together and enables rational and timely development of computers .S/W.Process defines a framework that must be established for effective delivery of S/W eng technology.

 The software process forms the basis for management control of software projects and establishes the context in which technical methods are

applied, work products (models, documents, data, reports, etc.) are produced, milestones are established, quality is ensured, and change is properly managed.

- *Methods layer:* provide the technical "how to's" for building S/W. Methods encompass a broad array of tasks that include communication, *req. analysis, design, coding, testing, and support.*
- *Tools layer* provides automated or semi-automated support for the process and the methods. When tools are integrated so that info. Created by one tool can be used by another, a system for the support of S/W development called *computer-aided software engineering* **is established.**

Explain software Process Framework , umbrella activities:

- *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 size or complexity. **EX (HOUSE CONSTRUCTION PLAN)**
- It also includes a set of *umbrella activities* that are applicable across the entire software process.
- E *ach framework activity is populated by a set of software engineering actions* - (phases _ example: CPMCD)
- A collection of related tasks that produces a major software engineering work product (e.g. design is a software engineering action).
- *Each action is populated with individual work tasks* that accomplish some part of the work implied by the action.

Framework activities are:

The following generic process framework is applicable to the vast majority of software projects:

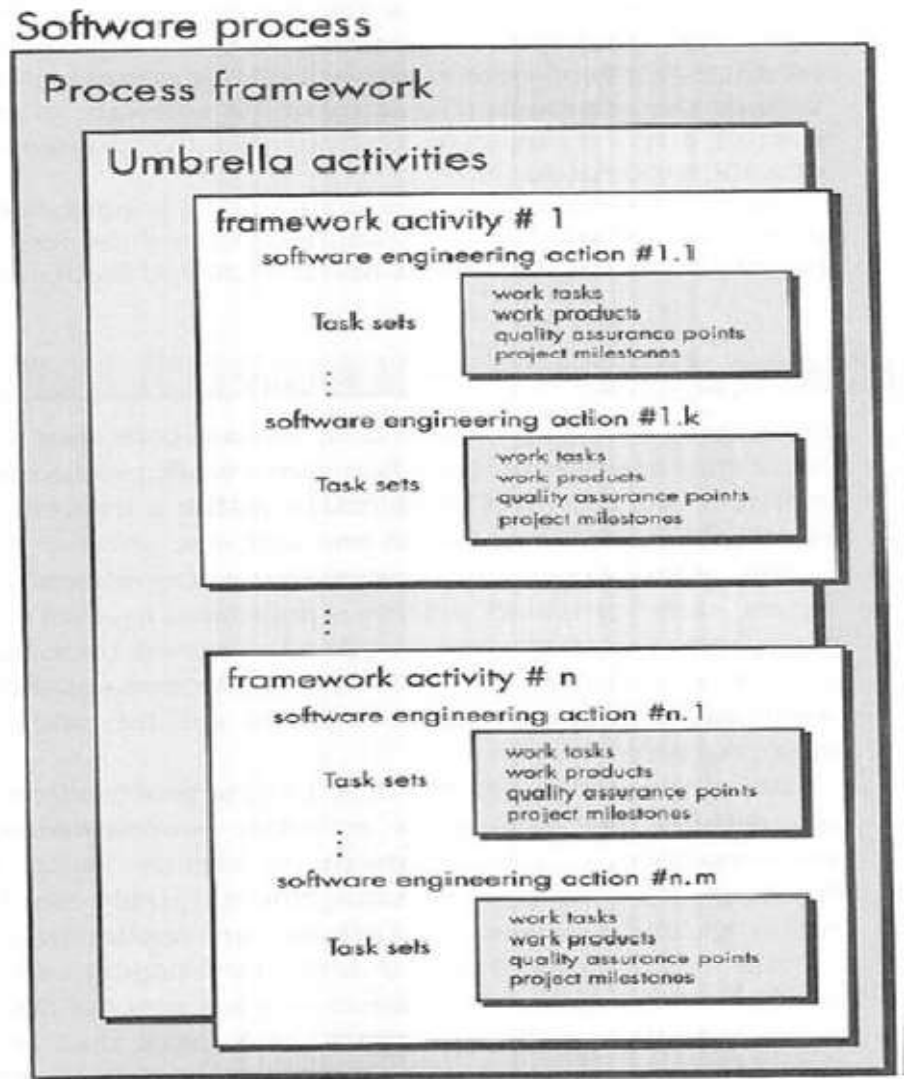
1. Communication: This framework activity involves heavy communication and collaboration with the customer (and other stakeholders, **Business Roles: Company Owner, CEO, Business Analyst, CIS,** Product analyzer, Quality Manager, Operational Managers, Team Leader, Developers, Technical & Non Technical staffs) and encompasses requirements gathering and other related activities.

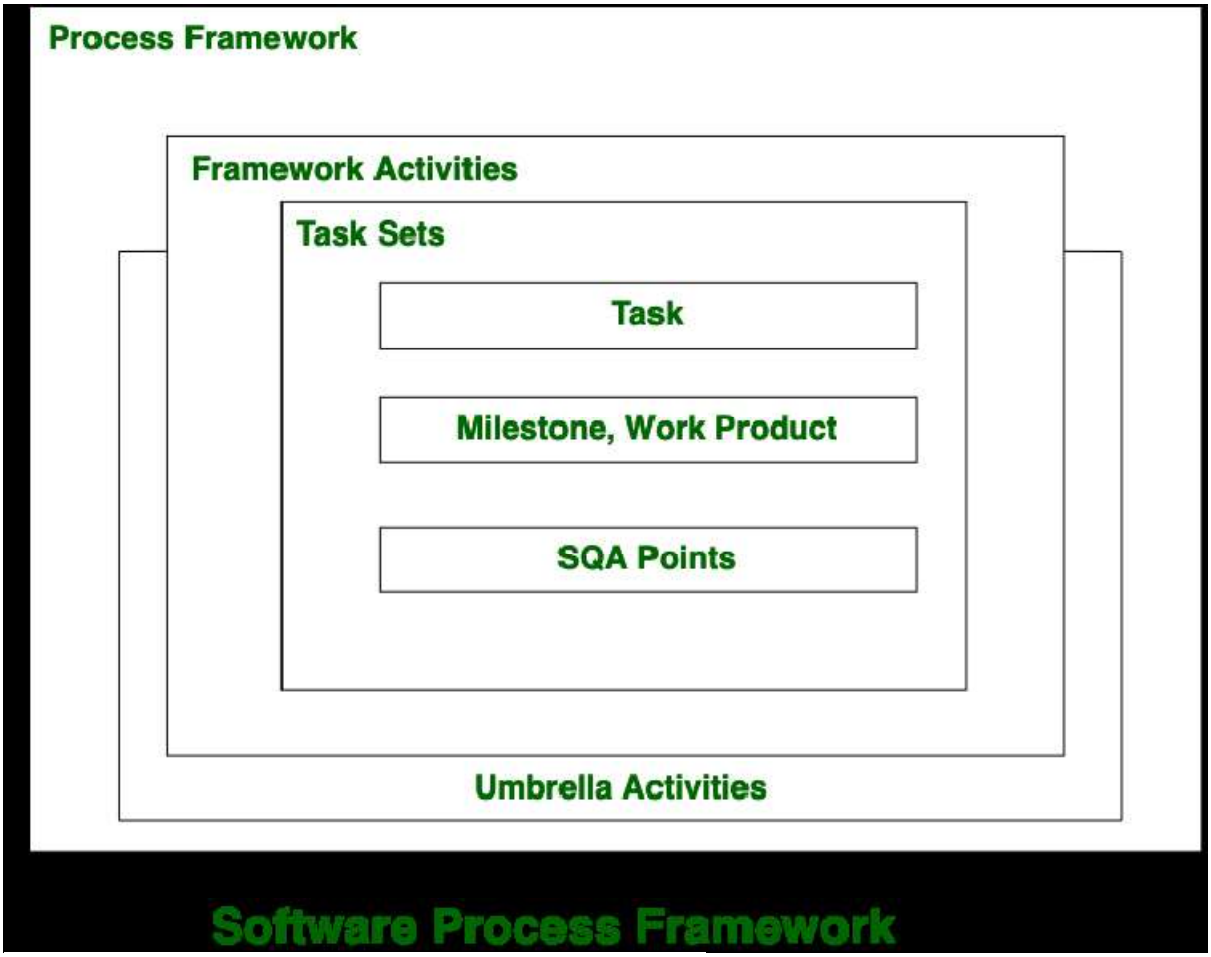
2. Planning: This activity establishes a plan for the software engineering work that follows. It describes the technical tasks to be conducted, the risks that are likely, the resources that will be required, the work products to be produced and a work schedule.

3. Modeling: This activity encompasses the creation of models that allow the developer and the customer to better understand software requirements and the design that will achieve those requirements.

4. Construction: This activity combines code generation (either manual or automated) and the testing that is required uncovering errors in the code.

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





Umbrella activities include:

- Risk management
- Software quality assurance(SQA)
- Software configuration management(SCM)

- Measurement
- Formal technical reviews(FTR)

EXPLANATION OF UMBRELLA ACTIVITIES

Software project tracking and control: When plan, tasks, models all have been done then a network of software engineering tasks that will enable to get the job done on time will have to be created.

Formal technical reviews: This includes reviewing the techniques that have been used in the project.

Software quality assurance : This is very important to ensure the quality measurement of each part to ensure them.

Software configuration management: Software configuration management (SCM) is a set of activities designed to control change by identifying the work products that are likely to change, establishing relationships among them, defining mechanisms for managing different versions of these work products.

Document preparation and production: All the project planning and other activities should be hardly copied and the production gets started here.

Re-usability management: This includes the backing up of each part of the software project that can be corrected or any kind of support can be given to them later to update or upgrade the software at user/time demand.

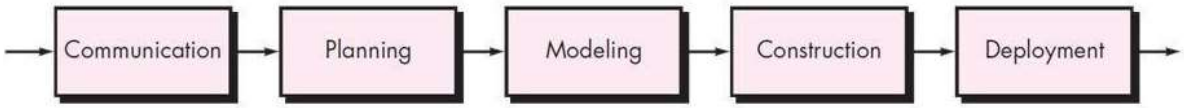
Measurement & Metrics: This will include all the measurement of every aspect of the software project.

Risk management: Risk management is a series of steps that help a software team to understand and manage uncertainty. It's a really good idea to identify it, assess its probability of occurrence, estimate its impact, and establish a contingency plan that— 'should the problem actually occur'.

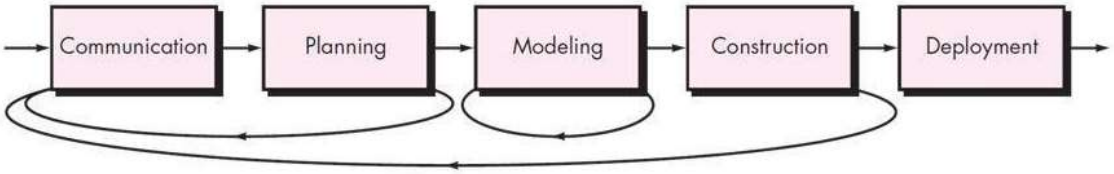
Explain Process Flow :

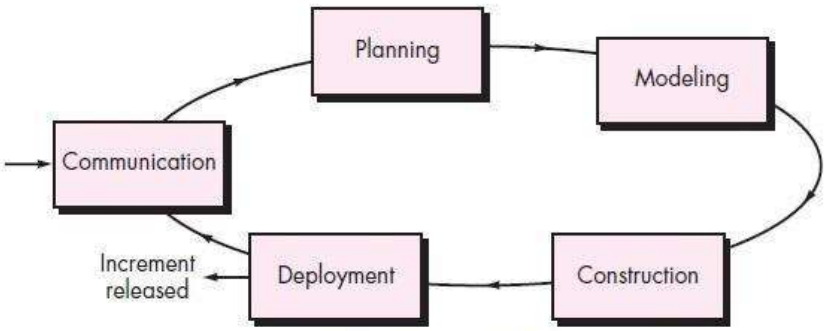
- *It describes how the framework activities and the actions and tasks are organized with respect to sequence and time.*
- *A linear process flow* executes each of the five framework activities in sequence, beginning with communication and culminating with deployment
- *An iterative process flow* repeats one or more of the activities before proceeding to the next
- *An evolutionary process flow* executes the activities in a "circular" manner. Each circuit through the five activities leads to a more complete version of the software
- *A parallel process flow* executes one or more activities in parallel with other activities (e.g., modeling for one aspect of the software might be executed in parallel with construction of another aspect of the software).

- Linear process flow

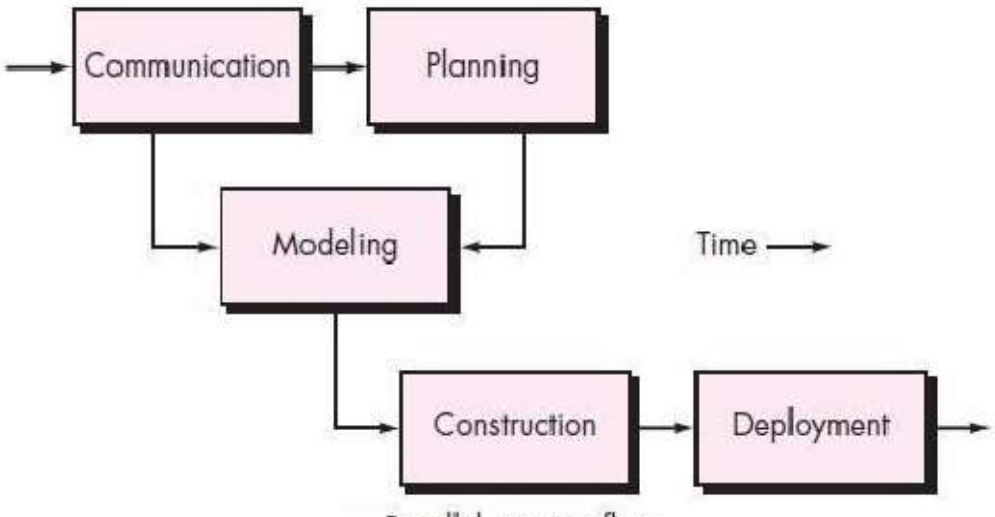


- Iterative process flow





Evolutionary process flow



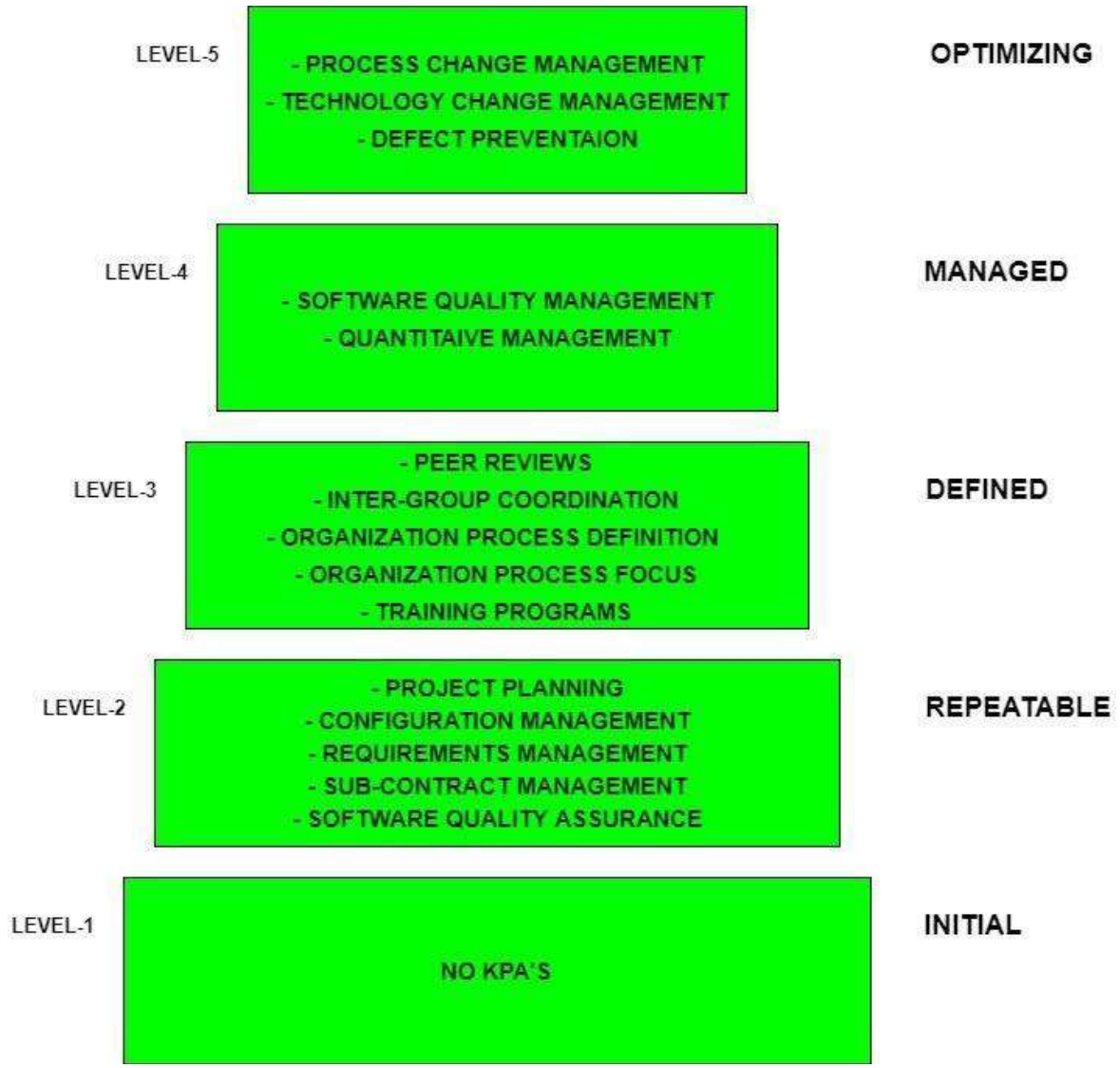
Parallel process flow

Defining a F ramework Activity :

To define a framework activity, the following points must be kept in mind:

- What *actions* are appropriate for the activity depending on the nature of the project.
- *characteristics* of the people doing the work
- *stakeholders* responsible for the activity

- *Identifying a Task Set :* Each software engineering action (Communication, Planning,Modeling, etc) can be represented by a task set, or series of tasks to do to complete the action.
  - *Depending upon the size of the project* , and the needs of the project, the number of tasks could vary.
  - When doing *initial communication, a small project* with a single stakeholder might be conducted via an informal meeting in an office.
  - *A medium project* might need several stakeholders with several meetings both individually and collectively until a process is ready to be rolled out.
  - *Large projects* might need a series of meetings, interviews with individual stakeholders, meetings for revisions, defining time lines, etc. With each action, this will be true, and it makes the "engineering" part difficult as there is no such formal definition for what is small, medium, or large, nor for the specific tasks.
  - *Company culture and the type of tasks* being done may play a big part in the correct way to handle things, which is why it is important to go over projects at the end to determine what should continue, and what should be changed to make them work more effectively.



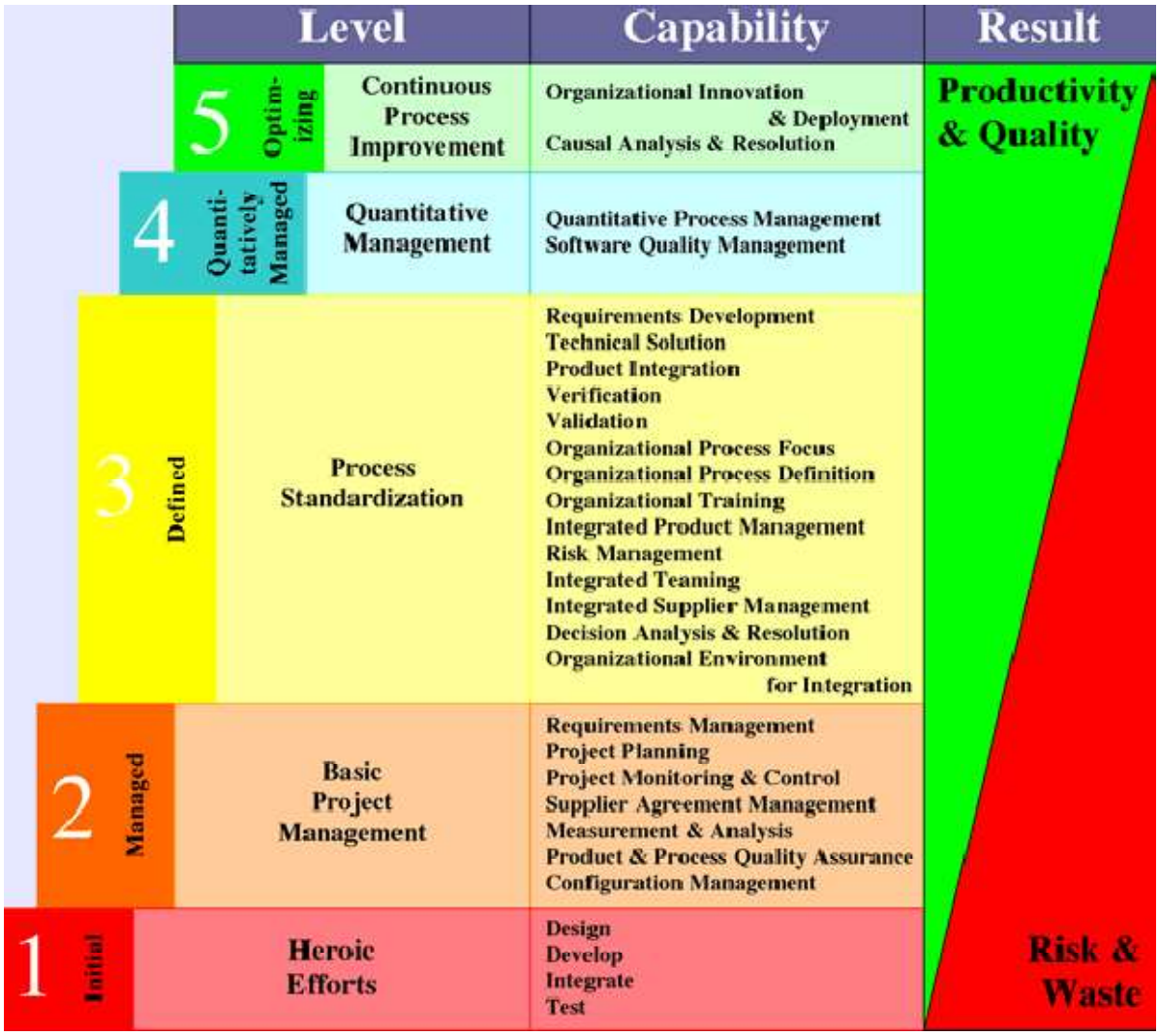
CMM was developed by the Software Engineering Institute (SEI) at Carnegie Mellon University in 1987.

1. *It is not a software process model. It is a framework* which is used to analyse the approach and techniques followed by any organization to develop a software product.
2. *It also provides guidelines* to further enhance the maturity of those software products.
3. *It is based on profound feedback* and development practices adopted by the most successful organizations worldwide.
4. *This model describes a strategy* that should be followed by moving through 5 different levels.
5. *Each level of maturity s*hows a process capability level. All the levels except level-1 are further described by Key Process Areas (KPA's)
6. *Key Process Areas (KPA's) :* Each of these KPA's defines the basic requirements that

should be met by a software process in order to satisfy the KPA and achieve that level of maturity.

What is KPA?

 Conceptually, key process areas form the basis for management control of the software project and establish a context in which technical methods are applied, work products like models, documents, data, reports, etc. are produced, milestones are established, quality is ensured and change is properly managed.



The 5 levels of CMM are as follows:

Level-1: Initial –

 No KPA ' s defined.

 Processes followed are ad hoc and immature and are not well defined.

 Unstable environment for software development.

 No basis for predicting product quality, time for completion, etc.

Level-2: Repeatable –

 Focuses on establishing basic project management policies.

 Experience with earlier projects is used for managing new similar natured projects.

 KPA' s:

 Project Planning - It includes defining resources required, goals, constraints, etc. for the project. It presents a detailed plan to be followed systematically for successful completion of a good quality software.
 Configuration Management- The focus is on maintaining the performance of the software product, including all its components, for the entire lifecycle.
 Requirements Management- It includes the management of customer reviews and feedback which result in some changes in the requirement set. It also consists of accommodation of those modified requirements.
 Subcontract Management- It focuses on the effective management of qualified software contractors i.e. it manages the parts of the software which are developed by third parties.
 Software Quality Assurance- It guarantees a good quality software product by following certain rules and quality standard guidelines while development.

Level-3: Defined - At this level, documentation of the standard guidelines and procedures takes place.
 It is a well defined integrated set of project specific software engineering and
management processes.

KPA's:

 Peer Reviews - In this method, defects are removed by using a number of review methods
like walkthroughs, inspections, buddy checks, etc.
 Intergroup Coordination- It consists of planned interactions between different development teams to ensure efficient and proper fulfillment of customer needs.
 Organization Process Definition- It's key focus is on the development and maintenance of the standard development processes.
 Organization Process Focus- It includes activities and practices that should be followed to improve the process capabilities of an organization.
 Training Programs- It focuses on the enhancement of knowledge and skills of the team members including the developers and ensuring an increase in work efficiency.

Level-4: Managed -

 At this stage, quantitative quality goals are set for the organization for software products as well as software processes.
 The measurements made help the organization to predict the product and process quality within some limits defined quantitatively.

KPA's:

 Software Quality Management- It includes the establishment of plans and strategies to develop a quantitative analysis and understanding of the product's quality.

 Quantitative Management - It focuses on controlling the project performance in a quantitative manner.

Level-5: Optimizing -

 This is the highest level of process maturity in CMM and focuses on continuous process improvement in the organization using quantitative feedback.

 Use of new tools, techniques and evaluation of software processes is done to prevent recurrence of known defects.

 Process Change Management - Its focus is on the continuous improvement of organization's software processes to improve productivity, quality and cycle time for the software product.

 Technology Change Management - It consists of identification and use of new technologies to improve product quality and decrease the product development time.

 Defect Prevention - It focuses on identification of causes of defects and to prevent them
from recurring in future projects by improving project defined processes.

 Each process area is defined in terms of Specific Goals (SG) and Specific Practices(SP) .

 Specific goals specify the characteristics that must be present if the process area activities are to be effective.

 Specific practices refine a goal into a set of process-related activities.

For example, project planning is one of eight process areas defined by the CMMI for "project management" category.**6 The specific goals (SG)** and the associated specific practices (SP)
defined for project planning are [CMM07]:

 SG 1 Establish Estimates

 SP 1.1-1 Estimate the Scope of the Project

 SP 1.2-1 Establish Estimates of Work Product and Task Attributes

 SP 1.3-1 Define Project Life Cycle

 SP 1.4-1 Determine Estimates of Effort and Cost

 SG 2 Develop a Project Plan SP 2.1-1 Establish the Budget and Schedule

 SP 2.2-1 Identify Project Risks

 SP 2.3-1 Plan for Data Management

 SP 2.4-1 Plan for Project Resources

 SP 2.5-1 Plan for Needed Knowledge and Skills

 SP 2.6-1 Plan Stakeholder Involvement

 SP 2.7-1 Establish the Project Plan

 SG 3 Obtain Commitment to the Plan

 SP 3.1-1 Review Plans That Affect the Project

 SP 3.2-1 Reconcile Work and Resource Levels

 SP 3.3-1 Obtain Plan Commitment

 2) A Staged CMMI model : Here the process improvement is measured based on the maturity levels.
 Process Maturity : It indicates how close a process is to being complete (mature) and how capable it is for further improvements through feedback and qualitative measures.Maturity levels apply to the organization as a whole.
used to describe an important framework activity (e.g., planning) or a task within
framework activity (e.g., project- estimating). Ambler has proposed the following
template for describing a process pattern:
Pattern Name. The pattern is given a meaningful name that describes its functions
within the software process (e.g., customer-communication).
Intent. The objective of the pattern is described briefly. For example, the intent of
customer-communication is “to establish a collaborative relationship with the customer in
an effort to define project scope, business requirements, and other project constraints”.
Type. The pattern type is specified. Ambler suggests three types:
· Task pattern defines a software engineering action or work task that is part of the
process and relevant to successful software engineering practice (e.g., requirements
gathering is a task pattern).
· Stage patterns represent a framework activity for the process. Since a framework
activity encompasses multiple work tasks, a stage pattern incorporates multiple task
patterns that are relevant to the stage. An example of a stage pattern might be
communication. This pattern would incorporate the task pattern requirements gathering
and others.
· Phase patterns define the sequence of framework activities that are iterative in nature.
An example of a phase pattern might be a spiral model or prototyping.
Initial context. The conditions under which the pattern applies are described. Prior to the
initiation of the pattern, we ask (1) what organizational team created activities have

already occurred (2) what is the entry state for the process? And (3) what software
engineering information or project information already exists?
For example, the planning pattern (a stage pattern) requires that
(1) customers and software engineers have established a collaborative communication:
(2) successful completion of a number of task patterns (specified) for the customercommunication
pattern has occurred: and
(3) project scope, basic business requirements, and project constraints are known.

An Example Process Pattern:

The following abbreviated process pattern describes an approach that may be applicable
when stakeholders have a general idea of what must be done, but are unsure of specific
software requirements.

Pattern name. Prototyping.

Intent: The objective of the pattern is to build a model (a prototype) that can be assessed
iteratively by stakeholders in an effort to identify or solidity software requirements.

Type: Phase pattern.

Initial context: The following conditions must be met prior to the initiation of this
pattern:
(1) stakeholders have been identified;
(2) a mode of communication between stakeholders and the software team has been established;
(3) the overriding problem to be solved has been identified by stakeholders:
(4) an initial understanding of project scope, basic business requirements, and project constraints
has been developed.

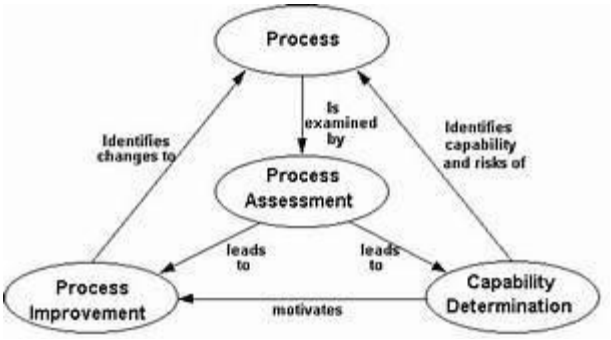
*Problem: Requirements are hazy or nonexistent, yet there is clear recognition that there is a
problem, and the problem must be addressed with a software solution. Stakeholders are
unsure of what they want; that is, they cannot describe software requirements in any detail.*

*Solutions: A description of the prototype that identifies basic requirements (e.g., modes of
interaction, computational features, processing functions) is approved by stakeholders.
Following this,*

(1) the prototype may evolve through a series of increments to become the production software
or
(2) the prototype may be discarded and the production software built using some other process
pattern.
Related Patterns: The following patterns are related to this pattern: **customer-communication; iterative design; iterative development; customer assessment; requirement extraction**.
Known uses/examples: Prototyping is recommended when requirements are uncertain.

PROCESS ASSESSMENT

The process assessment is 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 proposed over the past few decades:
PROCESS ASSESSMENT: A software process has to be assessed (evaluated) in order to ensure that it meets basic process criteria of being efficient and effective In process assessment the methods, tools and practices of a process are evaluated. The aim is to identify the areas for improvement and suggest a plan for making these improvements. It determines the strength, weakness and risks of the process.



A software process assessment consists of identifying the capabilities and risks of a process. This
leads to identify the changes or improvements in the process. Different approaches to process
improvement are:

1. Standard CMMI Assessment Method for Process Improvement (SCAMPI)

provides a five
-step process assessment model that incorporates initiating, diagnosing, establishing, acting,
and learning. The SCAMPI methods use the SEI CMMI as the basis for assessment.

2) CBA IPI(CMM based appraisal for Internal Process Improvement) It provides a diagnostic technique for assessing the relative maturity of software organization It is based on SEI CMM

3) **SPICE (ISO/IEC 15504)** Standard defines a set of requirements for software process assessment. The intent of the standard is to assist organization in developing an objective evaluation of the efficacy of any defined software process.

4) **ISO9001: 2000 for Software** is a generic standard that applies to any organization that wants to improve overall quality of the products, systems, or services that it provides. Therefore, the standard is directly applicable to software organizations and companies.

Personal Software Process (PSP) : Every developer uses some process that suits his needs to build computer software. This is called as The Personal Software Process(PSP). Here the practitioner responsible for project planning (e.g., cost estimation and schedule etc) and to control the quality of all software work products that are developed.

The PSP model defines five framework activities:

1) Planning

- The requirements are separated into components
- The size and resource estimates are calculated.
- Also, the defect estimate (the number of defects expected for the work) is made.
- development tasks are identified
- A project schedule is created.
- All metrics are recorded on worksheets or templates.

2) High-level design

External specifications for each component to be constructed are developed
A component design is created.
Prototypes are built if required.
All issues are recorded and tracked.

3) High-level design review The high level design is formally verified to uncover errors in the design. Metrics for all important tasks and work results are maintained

4) Development The component-level design is further refined and reviewed. Code is generated,reviewed, compiled, and tested. Metrics for all important tasks and work results are maintained

5) Postmortem Using the measures and metrics collected the processes is evaluated to check and improve its effectiveness

- Team Software Process (TSP): The goal of TSP is to build a "self-directed" project team that organizes itself to produce high-quality s/w.
- This team has a clear understanding of the goals and objectives,roles of each member,appropriate team process and strategy, local standards, project status and risks.

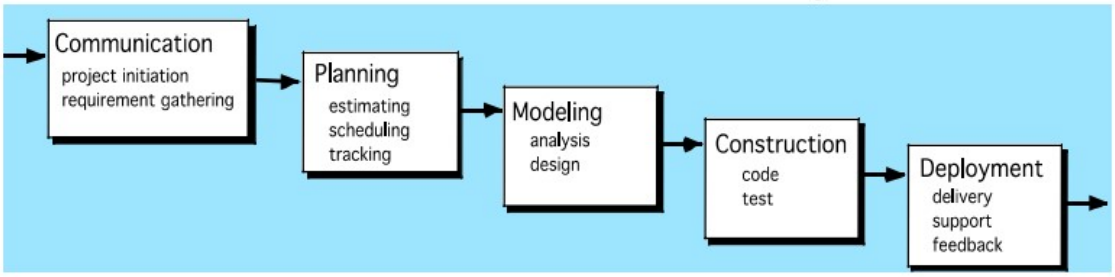
● It helps improve the performance and approach of the team towards software engineering. Show managers how to coach and motivate their teams and how to help them sustain peak performance.
● TSP defines the following framework activities:
● project launch
● high level design implementation
Integration and test
● post-mortem
● Measures are analyzed for improving the team process.
● Each project is "launched" using a "script" that defines the tasks to be accomplished. Scripts define the specific process activities and detailed work functions that are part of the team process.

SOFTWARE PROCESS MODELS

1. Linear Sequential Model/Waterfall Model/Classic Life Cycle

The Waterfall Model

1970 ,Winston W. Royce,



It is the oldest paradigm for SE. When requirements are well defined and reasonably stable, it leads to a linear fashion.

(problems: 1. rarely linear, iteration needed. 2. hard to state all requirements explicitly. Blocking state. 3. code will not be released until very late.)

The classic life cycle suggests a systematic, sequential approach to software development.

19

● *The linear sequential model sometimes called the classic life cycle or the waterfall model* suggests a systematic, sequential approach to software development that begins at the system level and progresses through communication, planning, modeling, construction, and deployment.

● The Waterfall model is the *earliest SDLC approach* that was used for software development.
● The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap. *The following given figure illustrates the waterfall model for software engineering.*

The sequential phases in Waterfall model are –

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

Why is it called a waterfall model?

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for the previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

Advantages of Classical Waterfall Model

Classical waterfall model is an idealistic model for software development. It is very simple, so it can be considered as the basis for other software development life cycle models. Below are some of the major advantages of this SDLC model:

✔ *This model is very simple and is easy to understand.*
✔ *Phases in this model are processed one at a time.*

✔ *Each stage in the model is clearly defined.*
✔ *This model has very clear and well understood milestones.*
✔ *Process, actions and results are very well documented.*
✔ *Reinforces good habits: define-before- design, design-before-code.*
✔ *This model works well for smaller projects and projects where requirements are well understood*

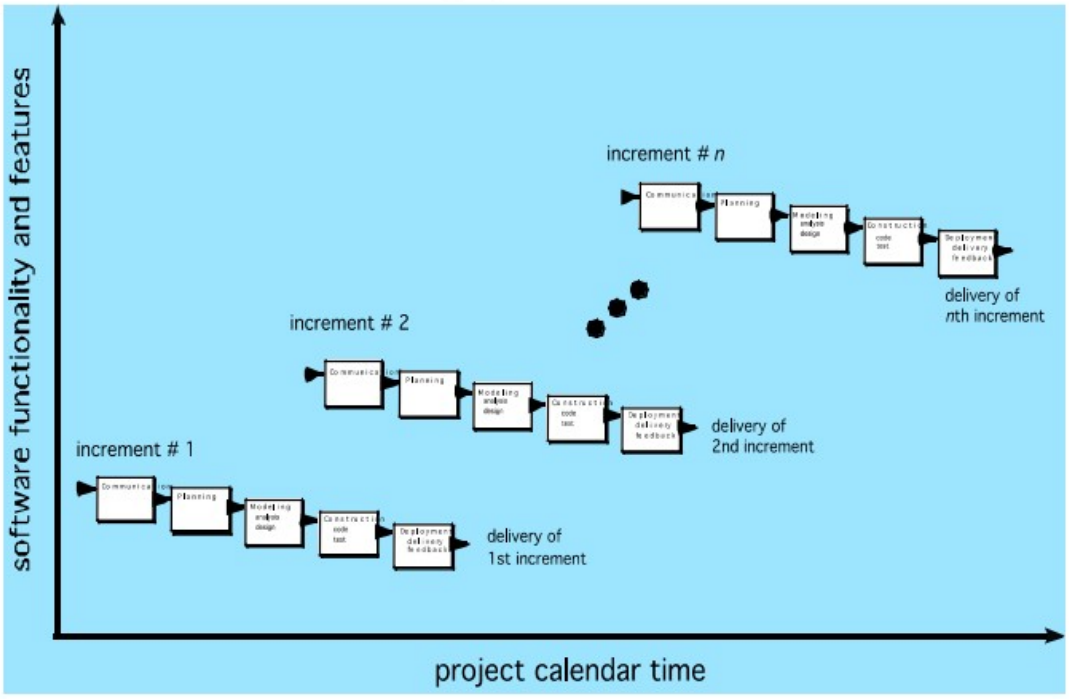
Disadvantages

✔ *Real projects rarely follow the linear sequential model. Although the linear model can accommodate iteration, it does so indirectly. As a result, changes can cause confusion as the project team proceeds.*
✔ *It is often difficult for customers to state all the requirements explicitly. This model requires this and has difficulty accommodating the natural uncertainty.*
✔ *The customers must have patience. A working version of the program will not be available until late in the project time-span. A major mistake, if undetected until the working program is reviewed, can be disastrous.*
· *It assumes that the requirements of a system can be frozen (i.e., baselined) before the design begins.* · *Since requirements are freezed at an early stage, the hardware required for the project must be chosen well in advance.*
· *A formal document is produced at the end of each phase.*
· *we cannot backtrack from one phase to another*
· *It follows the "big bang" approach: the entire software is delivered in one shot at the end. Hence the user does not know until the very end what they are getting.*
· *There can be blocking states where one team member has to wait until another member completes the dependent task.*
✔ *Today, software work is fast-paced and subject to a never-ending stream of changes.*
Linear sequential model is inappropriate for this, but when the requirements are fixed

Incremental Model

Incremental Model is a process of software development where requirements are divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system is achieved

The Incremental Model
- Baryy Boehm



✔ *Requirement Analysis*: In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

✔ *Design & Development:* In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development
phase.

✔ *Testing:* In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task.

✔ *Implementation:* Implementation phase enables the coding phase of the development system. It involves the final coding that is in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

When do we use the Incremental Model?

✔ *When the requirements are superior.*

✔ *A project has a lengthy development schedule.*
✔ *When Software teams are not very well skilled or trained.*
✔ *When the customer demands a quick release of the product.*
✔ *You can develop prioritized requirements first.*

✔ **Advantage of Incremental Model**

✔ *Errors are easy to recognize.*
✔ *Easier to test and debug*
✔ *More flexible.*
✔ *Simple to manage risk because it is handled during its iteration.*
✔ *The Client gets important functionality early.*

✔ *Disadvantage of Incremental Model*

✔ *Need for good planning*
✔ *Total Cost is high.*
✔ *Well defined module interfaces are needed.*

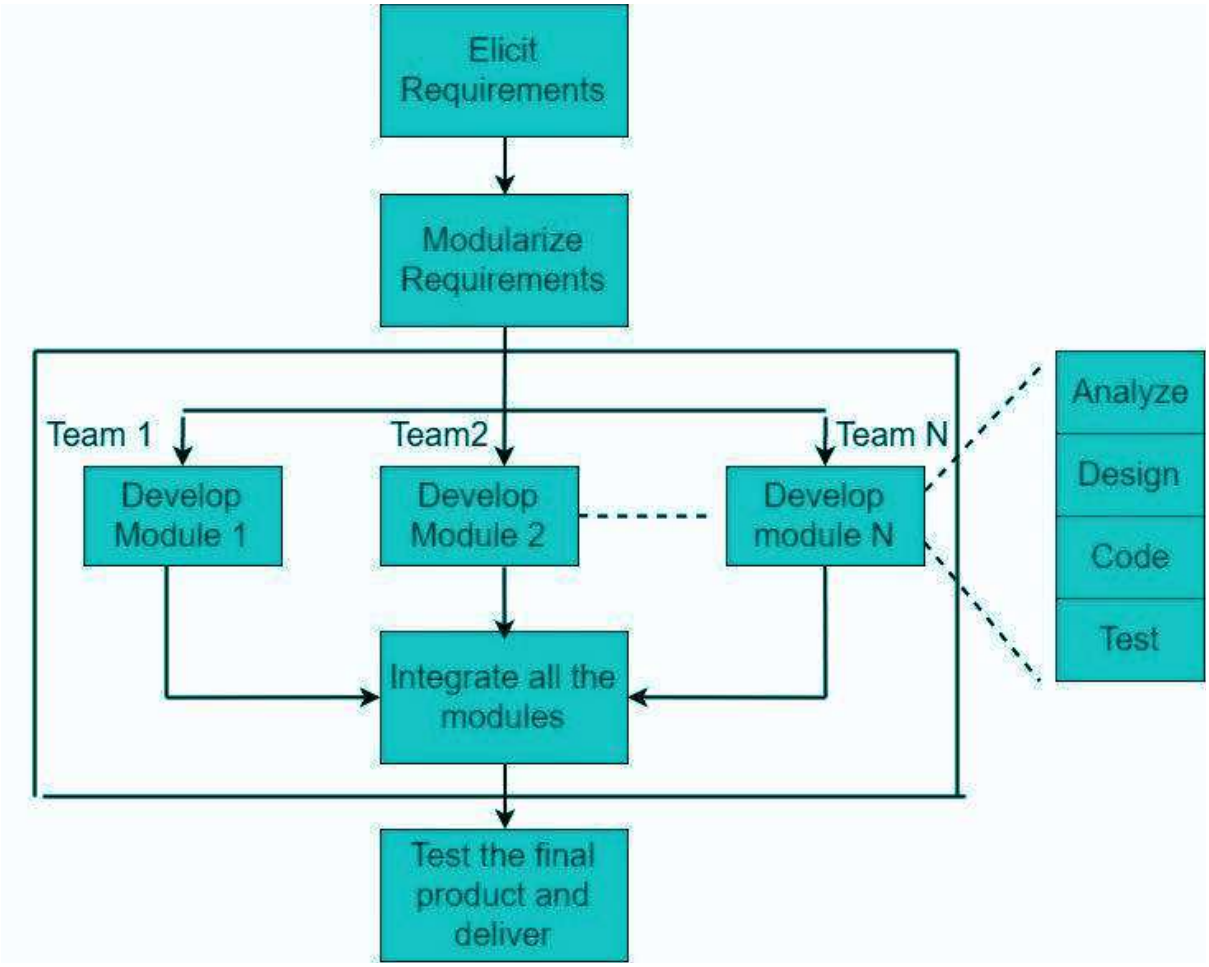
Rapid application development model (RAD)

● The Rapid Application Development Model was first proposed by IBM in the 1980's. The critical feature of this model is the use of powerful development tools and techniques.

● *A software project can be implemented using this model if the project can be broken down into small modules wherein each module can be assigned independently to separate teams. These modules can finally be combined to form the final product.*

● Development of each module involves the various basic steps as in waterfall model i.e analyzing, designing, coding and then testing, etc. as shown in the figure.

● *Another striking feature of this model is a short time span i.e the time frame for delivery(time-box) is generally 60-90 days.*



This model consists of 4 basic phases:

1. *Requirements Planning* – It involves the use of various techniques used in requirements elicitation like brainstorming, task analysis, form analysis, user scenarios, FAST (Facilitated Application Development Technique), etc. It also consists of the entire structured plan describing the critical data, methods to obtain it and then processing it to form a final refined model.

2. *User Description* – This phase consists of taking user feedback and building the prototype using developer tools. In other words, it includes re-examination and validation of the data collected in the first phase. The dataset attributes are also identified and elucidated in this phase.

3. *Construction* – In this phase, refinement of the prototype and delivery takes place. It includes the actual use of powerful automated tools to transform process and data models into the final working product. All the required modifications and enhancements are too done in this

phase.

4. *Cutover -*

All the interfaces between the independent modules developed by separate teams have to be tested properly. The use of powerfully automated tools and subparts makes testing easier. This is followed by acceptance testing by the user.

The process involves building a rapid prototype, delivering it to the customer and taking feedback. After validation by the customer, an SRS document is developed and the design is finalised.

Advantages -

- *Use of reusable components helps to reduce the cycle time of the project.*
- *Feedback from the customer is available at initial stages.*
- *Reduced costs as fewer developers are required.*
- *Use of powerful development tools results in better quality products in comparatively shorter time spans.*
- *The progress and development of the project can be measured through the various stages.*
- *It is easier to accommodate changing requirements due to the short iteration time spans.*

Disadvantages -

- *The use of powerful and efficient tools requires highly skilled professionals.*
- *The absence of reusable components can lead to failure of the project.*
- *The team leader must work closely with the developers and customers to close the project in time.*
- *The systems which cannot be modularized suitably cannot use this model.*
- *Customer involvement is required throughout the life cycle.*

Applications -

1. *This model should be used for a system with known requirements and requiring short development time.*
2. *It is also suitable for projects where requirements can be modularized and reusable components are also available for development.*
3. *The model can also be used when already existing system components can be used in developing a new system with minimum changes.*

4. This model can only be used if the teams consist of domain experts. This is because relevant knowledge and ability to use powerful techniques is a necessity.

5. The model should be chosen when the budget permits the use of automated tools and techniques required.

EVOLUTIONARY PROCESS MODELS :

Evolutionary models are iterative type models.·With each iteration a more complete version of
the software is·developed.

Following are the evolutionary process models.

1. *The prototyping model*
2. *The spiral model*
3. *Concurrent development model*

I) The prototyping model

What is a Prototype Model?

The Prototype model is one of the software development life cycle models in which a prototype is built with minimal requirements, which is then tested and modified based on the feedback received from the client until a final prototype with desired functionalities gets created. This final prototype also acts as a base for the final product.

- *As mentioned earlier, this model is useful when all the detailed requirements are not known to the client before starting the project.*
- *It is also useful when the product to be developed is a complex one and similar products do not exist in the market.*
- *In such a scenario, the client can ask the developers to start working on the basic prototype with limited requirements. Once the basic prototype is ready, the client can see and check the prototype to decide what all changes are required.*
- *The client can also use the prototype to do market research and gather end-user or customer feedback. When the client has decided about the changes that need to be made, the client will give these requirements to the requirements gathering team, which eventually reach the development team.*
- *Developers can then start working on the modifications to the basic prototype. This cycle will be repeated until the client is satisfied with the prototype which reflects the final product.*

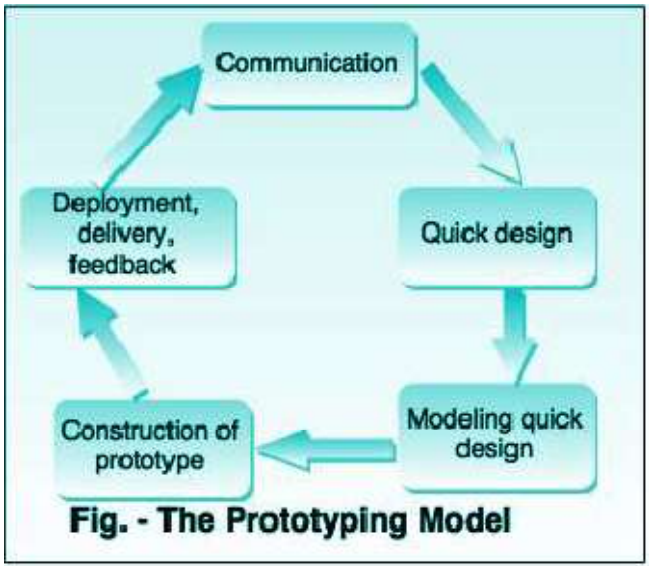


Fig. - The Prototyping Model

Phases of Prototype Model

The following are the primary phases involved in the development cycle of any prototype model.

- *Initial Communication - In this phase, business analysts and other individuals responsible for collecting the requirements and discussing the need for the product, meet the stakeholders or clients.*
- *Quick Plan - Once basic requirements have been discussed, a quick plan of the initial prototype is made.*
- *Modeling Quick Design - User interface part i.e. designing part of the prototype is carried out in this phase.*
- *Development of the Prototype: - In this phase, the designed prototype is coded and developed.*
- *Deployment, Delivery, and Feedback of the Prototype - In this phase, the initial prototype is deployed and is accessible to clients for its use. Clients review or evaluate the prototype and they provide their feedback to the requirements gathering and development teams.*

Above mentioned phases keep repeating until the replica of the final product is deployed.

- *Final Product Design, Implementation, Testing, Deployment, and Maintenance - Once*

the client finalizes a prototype, on the basis of the prototype, the final product is designed and developed. This developed product is tested by the testing team and if it is ready to go LIVE, the product is deployed and is available for end-users.

Advantages of Prototype Model

Prototype model offers the following benefits-

- Quick client feedback is received which speeds up the development process. Also, it helps the development team to understand the client's needs.
- Developed prototypes can be used later for any similar projects.
- Any missing functionality and any error can be detected early.
- It is useful when requirements are not clear from the client's end, even with limited requirements, the development team can start the development process.

Disadvantages of Prototype Model

Apart from appealing advantages, the prototype model has many disadvantages that are listed below-

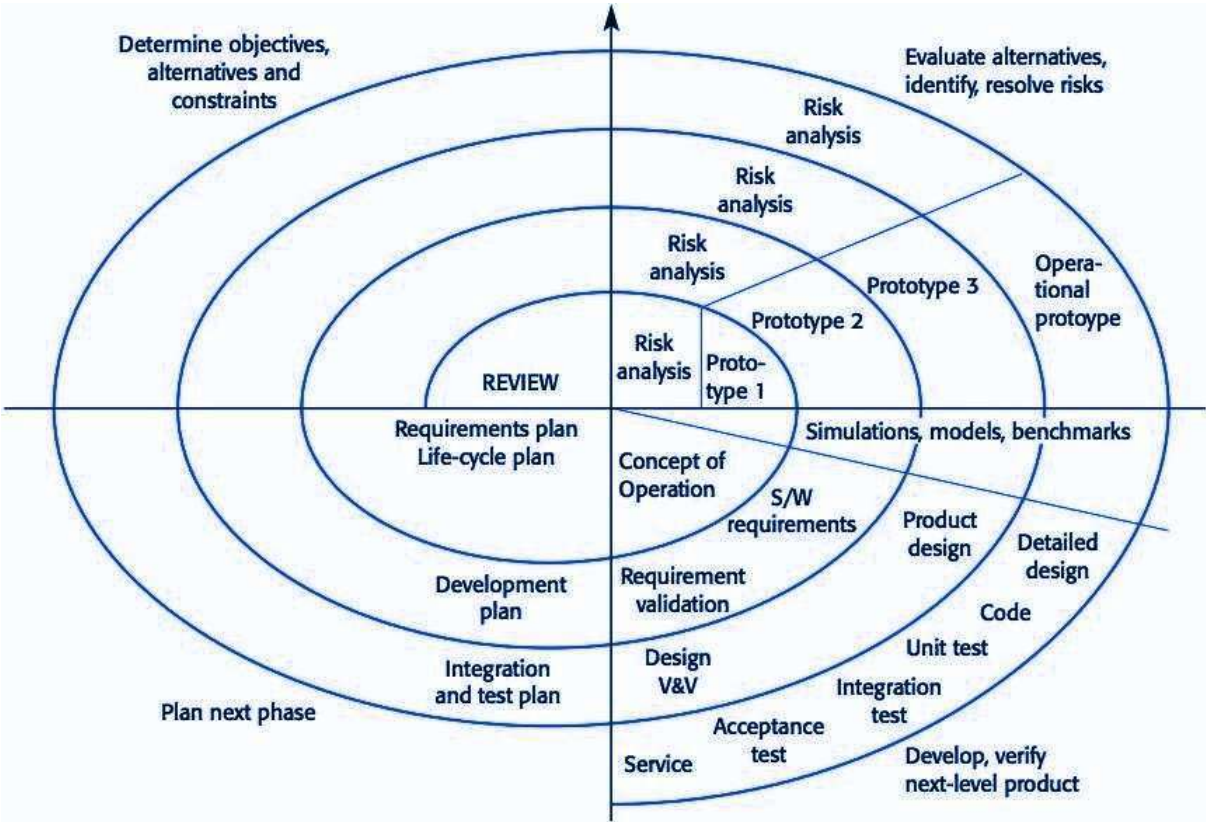
- It is a time-consuming process or method as multiple prototypes might be needed until the client reaches the final requirements. The Client may not have an explicit idea about what they want.
- This method involves too much client interaction and involvement, which can be done only with a committed client.
- In the beginning, it is a bit difficult to predict the exact amount of time needed to reach the final product.
- While coding, developers do not have a broad perspective of what is coming, because of which they might use an underlying architecture that is not suitable for a final product.
- To produce the quick prototype, developers might make weak decisions during the development process (especially implementation decisions), and compromise on quality which might eventually affect the product.

Software Engineering | Spiral Model

Spiral model is one of the most important Software Development Life Cycle models, which provides support for **Risk Handling**. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. **Each loop of the spiral is called a Phase of the software development process.**

- *The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using a spiral model.*

● *The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase*



The spiral model has 3 main features

1. Cyclic approach wherein the system's degree of definition and implementation grows with
every increment and its degree of risk goes on decreasing.
2. Anchor Point Milestones which ensure that stakeholder commitment creates feasible and mutually satisfactory system solutions.
3. Risk Identification

● *Using spiral models, the software is developed as a series of evolutionary releases (like paper model , product specification, prototype , design, code etc after checking risk factors).*

● *A spiral model is divided into a set of framework activities defined by the software engineering team. Each framework activity represents one segment*

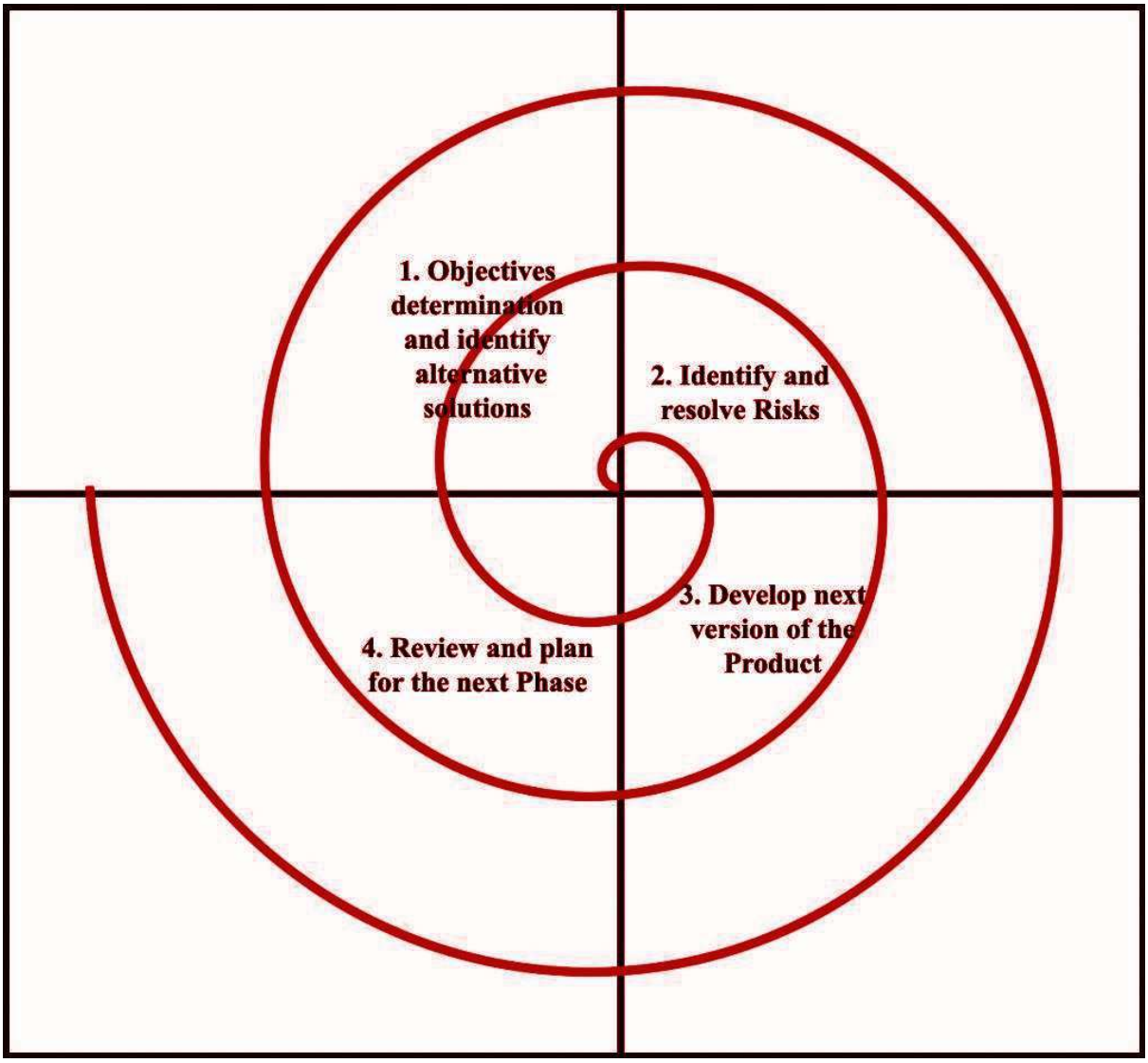
● *The evolutionary process begins, the software team performs activities in a circuit around the spiral in a clockwise direction, beginning at the center. Risk is considered during each revolution.*

Milestone which is a combination of work product and condition that is reached along
the path of the spiral is noted after each evolutionary pass. Each pass results in
adjustments to the project plan.

● *Cost and schedule are adjusted based on feedback derived from the customer.*

● *first circuit – "concept development and product specification starts with core 2nd circuit: prototype ; 3rd : design; 4th : code ; 5 th : testing ; Last spiral – represents "project enhancement"*

Spiral Model



Each phase of the Spiral Model is divided into four quadrants as shown in the above figure.

The functions of these four quadrants are discussed below-

1. *Objective determination and identify alternative solutions: Requirements are gathered from the customers and the objectives are identified, elaborated and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.*
2. *Identify and resolve Risks: During the second quadrant all the possible solutions are evaluated to select the best possible solution. Then the*

risks associated with that solution is identified and the risks are resolved using the best possible strategy. At the end of this quadrant, Prototype is built for the best possible solution.

3. Develop the next version of the Product: During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.

4. Review and plan for the next Phase: In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.

A risk is any adverse situation that might affect the successful completion of a software project. The most important feature of the spiral model is handling these unknown risks after the project has started. Such risk resolutions are easier done by developing a prototype. The spiral model supports coping up with risks by providing the scope to build a prototype at every phase of the software development

Advantages of Spiral Model:

Below are some of the advantages of the Spiral Model.

- Risk Handling: The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.
- Good for large projects: It is recommended to use the Spiral Model in large and complex projects.
- Flexibility in Requirements: Change requests in the Requirements at later phase can be

1. incorporated accurately by using this model.
2. ● Customer Satisfaction: Customers can see the development of the product at the early
3. phase of the software development and thus, they become habituated with the system by
4. using it before completion of the total product.

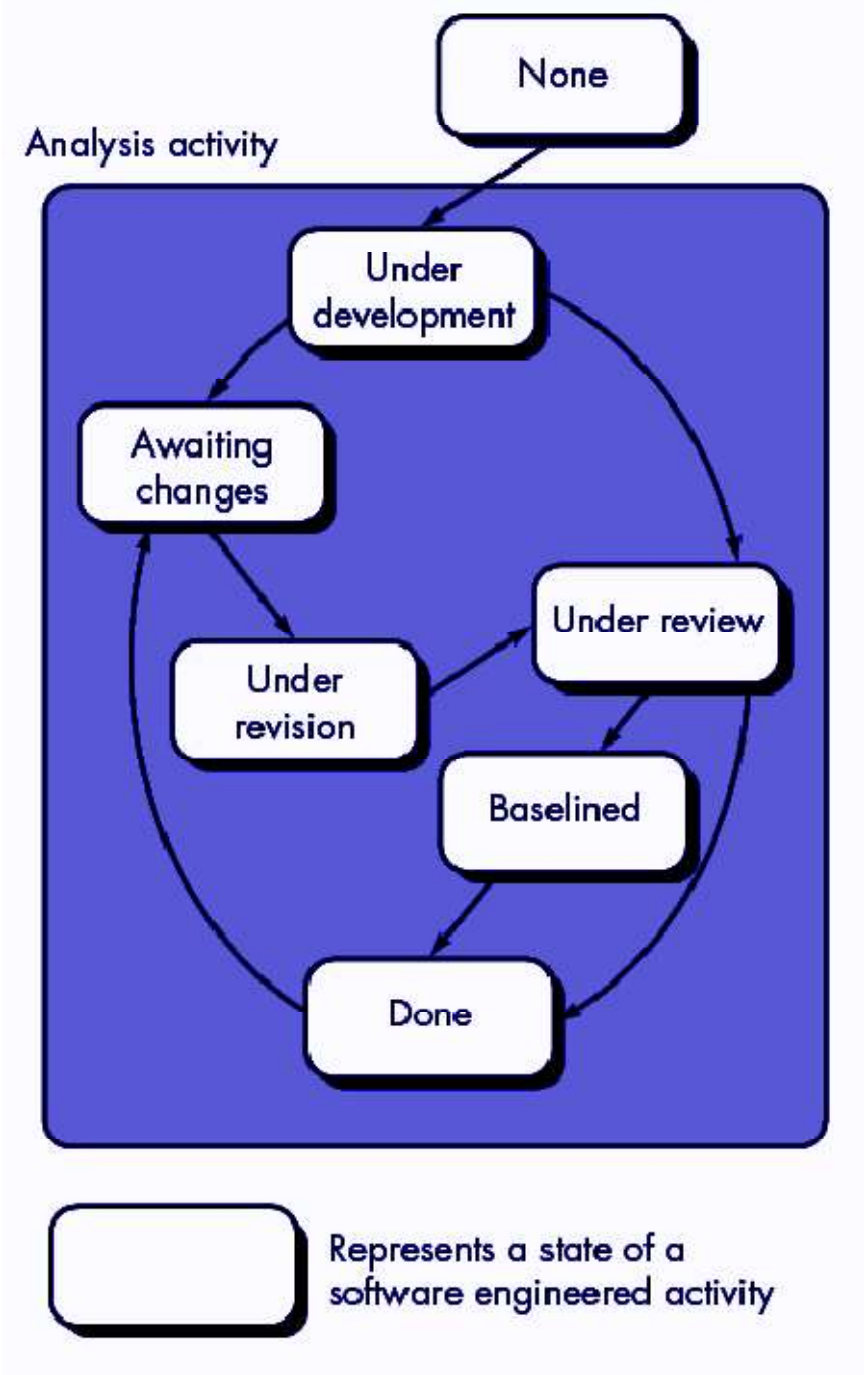
Disadvantages of Spiral Model:

Below are some of the main disadvantages of the spiral model.

- Complex: The Spiral Model is much more complex than other SDLC models.
- Expensive: Spiral Model is not suitable for small projects as it is expensive.
- Too much dependable on Risk Analysis: The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced expertise, it is going to be a failure to develop a project using this model.
- Difficulty in time management: As the number of phases is unknown at the start of the project, so time estimation is very difficult.

CONCURRENT MODEL

The concurrent process model can be represented schematically as a series of major technical activities, tasks, and their associated states. For example, the engineering activity defined for the spiral model is accomplished by invoking the following tasks: prototyping and/or analysis modeling, requirements specification, and design.



● *The activity-analysis*-may be in any one of the states noted at any given time.

Similarly, other activities (e.g., design or customer communication) can be represented in an analogous manner. All activities exist concurrently but reside in different states.

● *For example*, early in a project the customer communication activity (not shown in the figure) has completed its first iteration and exists in the awaiting changes state. The analysis activity (which existed in the non state while initial customer communication was completed) now makes a transition into the under development state. If, however, the customer indicates that changes in requirements must be made, the analysis 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. For example, during early stages of design, an inconsistency in the analysis model is uncovered. This generates the event analysis model correction which will trigger the analysis activity from the done state into the awaiting changes state.

● *The concurrent process model is often used as the paradigm for the development of client/server applications.* A client/server system is composed of a set of functional components. When applied to client/server, the concurrent process model defines activities in two dimensions : a system dimension and a component dimension. System level issues are addressed using three activities: design, assembly, and use. The component dimension is addressed with two activities: design and realization.

● *Concurrency is achieved in two ways*:

● System and component activities occur simultaneously and can be modeled using the state-oriented approach described previously;

● A typical client/server application is implemented with many components, each of which can be designed and realized concurrently.

● In reality, the concurrent process model is applicable to all types of software development and provides an accurate picture of the current state of a project. Rather than confining software engineering activities to a sequence of events, it defines a network of activities.

● Each activity on the network exists simultaneously with other activities. Events generated within a given activity or at some other place in the activity network trigger transitions among the states of an activity.

Specialized Process Models

Component Based Development

Commercial off-the-shelf (COTS) Software components, developed by vendors who offer them as products, can be used when Software is to be built. These components provide targeted functionality with well-defined interfaces that enable the component to be integrated into the Software.

The *component-based development* model incorporates many of the characteristics of the spiral model.

The *component-based development* model incorporates the following steps:

- Available component-based products are researched and evaluated for the application domain in question.
- Component integration issues are considered.
- Software architecture is designed to accommodate the components.
- Components are integrated into the architecture.
- Comprehensive testing is conducted to ensure proper functionality.

The component-based development model leads to Software reuse, and reusability provides Software engineers with a number of measurable benefits.

Advantages:

Less development cycle

time Less

project cost increase in productivity

Promotes software reuse

Limitation: Needs a robust component library

The Formal Methods Model

The Formal Methods Model encompasses a set of activities that leads to formal mathematical specifications of Software.

Formal methods enable a Software engineer to specify, develop, and verify a computer-based system by applying a rigorous, mathematical notation.

A variation of this approach, called clean-room Software engineering is currently applied by some software development organizations.

Although not a mainstream approach, the formal methods model offers the promise of defect free Software. Yet, concern about its applicability in a business environment has been voiced:

- The development of formal models is currently quite time-consuming and expensive.
- B/C few software developers have the necessary background to apply formal methods, extensive training is required.
- It is difficult to use the methods as a communication mechanism for technically unsophisticated customers.

Advantage: Promises a defect free software can be used to develop safety critical systems

Limitations: Quiet time-consuming and expensive model Very few software developers have the necessary skill, Cannot be used to communicate with technically unsophisticated customers

Aspect-Oriented Software Development

Regardless of the software process that is chosen, the builders of complex software invariably implement a set of localized features, functions, and information content.

- *Aspect-oriented software development or Aspect oriented Programming(AOP) “ Aspect-oriented software development (AOSD) is a*

software design solution that helps address the modularity issues that are not properly resolved by other software approaches.

- ***" Aspects are modules so that the crosscutting concerns can be encapsulated separately modularized and localized. This promotes better modularization. Aspect oriented Programming(AOP) provides a process and· methodological approach for defining, specifying, designing, and constructing aspects.***

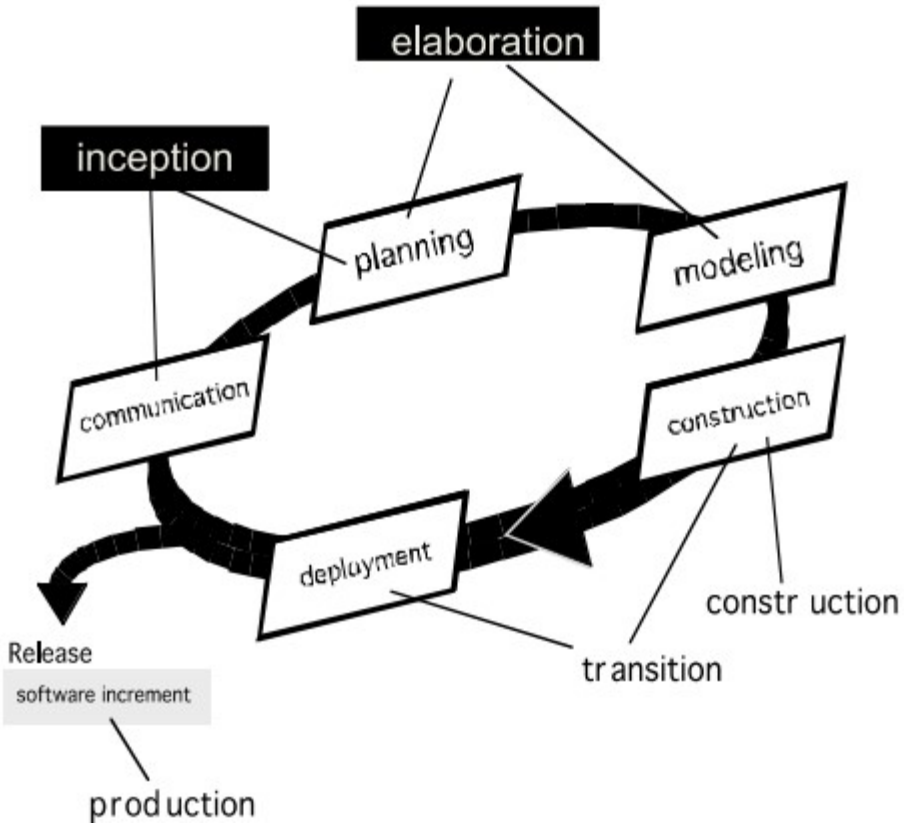
Every complex software is made up of components which are· constructed within the context of a system architecture. Such software have concerns like customer required properties· or areas of technical interest etc. Some concerns are systemic like synchronization issues, logging,· recovery etc.

- ***When concerns cut across multiple system functions, features,· and information they are often referred to as crosscutting concerns. Such crosscutting concerns have an impact across the software· architecture.***
- *AOP is used for separation of concerns and it makes available specific concepts for better modularization of crosscutting concerns which are used for the composition of the program components. · It adopts characteristics of both The aspects are identified and spiral model (iterative nature) then constructed iteratively. The aspects are concurrent process models (parallel nature) engineered independently of localized software components and they have a direct impact on these components.*

The Unified Process

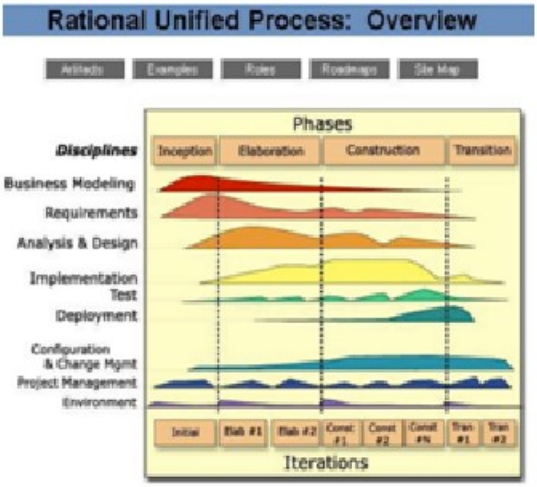
- A "use-case driven, architecture-centric, iterative and incremental" software process closely aligned with the Unified Modeling Language (UML).
- The UP is an attempt to draw on the best features and characteristics of conventional software process models, but characterize them in a way that implements many of the best principles of agile software development.

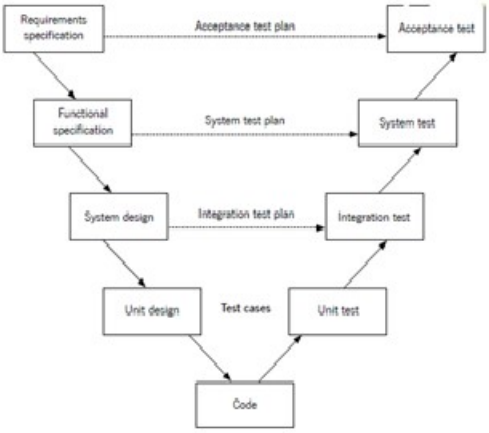
The Unified Process (UP)



Phases of the Unified Process

The figure below depicts the phases of the UP and relates them to the V-Model activities.





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 the ensuing project is developed.

A use-case describes a sequence of actions that are performed by an actor (person, machine, another system) as the actor interacts with the Software.

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 architectural representation to include five different views of the software - the use-case model, the analysis model, the design model, the implementation model, and 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.

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

At the conclusion of the transition phase, the software increment becomes a usable software release “user manuals, trouble-shooting guides, and installation procedures.)

The production phase of the UP coincides with the development activity of the generic process.

The on-going use of the software is monitored, support for the operating environment is provided and defect reports and requests for changes are submitted and evaluated.