UNIT-1

What is Software

Computer Software is the product that Software professional design and build and also it is instructions (Computer programs) that when executed provide desired features, function, and performance.

* Nature of the Software

Software's Dual Role

* Software is a product
* Software is a vehicle for delivering a product.

Software is a product

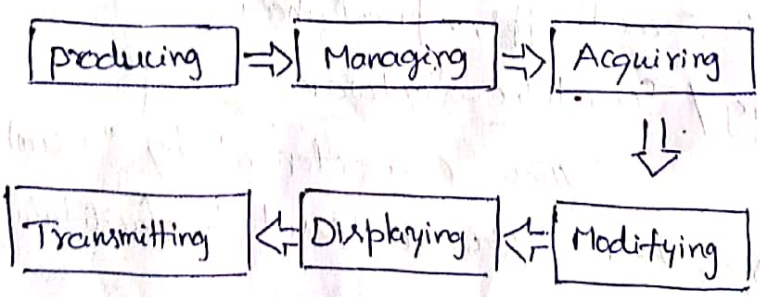
→ Delivers Computing potential
→ produces, manages, acquires, modifies, displays, or transmits information

Software is a vehicle for delivering a product

→ Supports or directly provides System functionally
→ Controls other programs (e.g operating System)
→ Effects communications (e.g networking Software)
→ Helps build other Software (e.g Software tools)

* Defining Software

Software is an information transformer — producing, managing, acquiring, modifying, displaying or transmitting info. That can be simple as a single bit or as complex as a multimedia presentation derived from data acquired from dozens of Indep-endent Sources.



* Software Application Domains / changing Nature of Software

Seven broad categories of computer software present challenges for software Engineers

* System Software
* Application Software
* Engineering / Scientific Software
* Embedded Software
* product - line Software
* web - application
* AI Software

System Software It is a collection of programs. Usually provide services to other programs.

⇒ System Softwares are two types.

Determinate data & Indeterminate data

* Determinate data → These procss complex but determinate Ex: compilers
* Indeterminate data → These procss easy but Indeterminate Ex: O/S

⇒ Heavy Intiraction with n/w, multiple users, resource sharing, Scheduling, procss management.

⇒ Example operating System, compilers etc.

Application Software

Stand alone programs that solve a specific business need. It is group of programs designed for end users.

Example : Email apps, Spread sheets

Engineering / Scientific Software

These characterized by "number crunching" algorithms. These Software used to facilitate the engineering functions & tasks

Example CAD (computer Aided Design)

Embedded Software

It resides in read only memory

→ It is used to control products & Systems for consumer & Industrial markets

Example: GPS devices, factory robots, modern smart watches microwave oven control.

Product-line Software:

⇒ Designed to provide a specific capability for use by many different Customers.

⇒ These Softwares can focus on a limited and hard marketplace. (Eg. inventory control products) (or) mass consumer marketplace. (Eg. word processing, spreadsheets, Computer graphics, multimedia, entertainment, database management & personal & business financial applications).

Web applications: It is a client-server computer program in which the client runs in a web browser.

common web applications include webmail, online retail sales, wikis, and other many functions

Artificial Intelligence Software

Software makes use of non numerical algorithms to solve complex problems That are not open for computation or straightforward analysis

⇒ AI is The branch of Computer Science That emphasizes the development of intelligence machines, Thinking and working like humans. for Example, Speech recognition, problem solving etc.

* Characteristics of Software

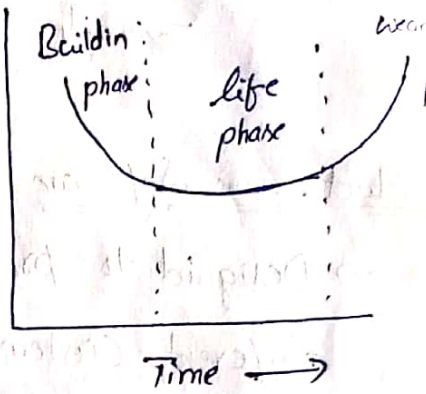
* Software does not wear out

* Software is not manufactured.

* Reusability of component
* Software is flexible.

⇒ Software does not wear out

Software becomes unreliable overtime instead of wearing out.



⇒ Software is not manufactured.

It is not manufactured in classical sense. Making 1000 copies is not an issue and it does not involve any cost. as in case of hardware product, every product cost us due to raw material and other expenses

⇒ Reusability of Components

* In Software, every project is a new project we start from Scratch & design every unit of the Software product which will increase the cost.

⇒ Hence developers are concentrate on truly innovative element of design

* Eg: GUI are built using reusable components That enable the creation of graphics & windows.

⇒ Software is flexible

A program can be build to do anything. It is helpful to accomodate anything of change.

Legacy Software :-

Legacy Software is an old and outdated program That is still used to perform a task for a user, even though newer and more efficient options are available.

time passes legacy systems develop gradually because of some reasons

* The Software must be adapted to meet the needs of new computing environments or technology.
* The Software must be enhanced to implement new business requirements.
* The Software must be re-architected to make it Viable within a network environment.

The goal of modern Software engineering is to "Software systems continually change, new Software systems are built from the old ones, all must interoperate and cooperate with each other."

Software Engineering

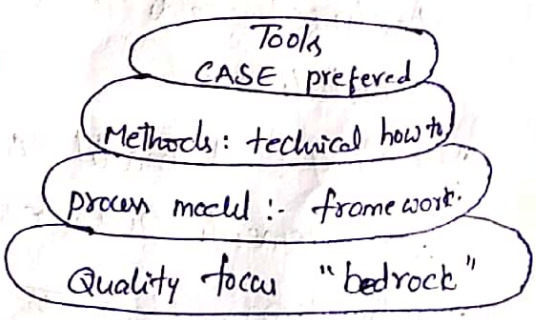
This is an Engineering Approach how to develop a Software Systematically.

Defi: The Application of Systematic, disciplined, quantifiable approach to development, operation, and maintenance of Software

→ The foundation for Software Engineering is the process layer.
→ process defines a framework for a set of key process areas.

CASE - Computer Aided Software Engg.

Layered Technology



Software Process:

A process is a collection of activities, actions, and tasks that are performed when some work product is to be created.

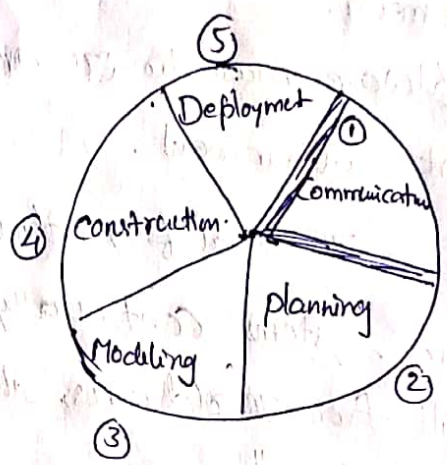
⇒ An Activity strives to achieve a broad objective (e.g communication) is applied regardless of the application domain.

=> An Action consists of a set of tasks that produces work product (e.g an architectural design model)

=> A Task focuses on a small, but well-defined objective (e.g -cting a unit test) that produces a outcome.

* A Generic process framework for software engineering encompa-sses five activities

- Communication
- planning
- Modeling
- Construction
- Deployment



Communication :
The Activity involves heavy Communication with customers and other stakeholders in order to gather require-ments and other related Activities

planning : A Software project is a complicated journey, and the planning activity creates a "map" that guide the team. A plan to be followed will be created which will describe the technical tasks to be conducted, risk, required resources, work schedule etc.

Modeling : A model will be created to better understand the requirements and design to achieve these requirements.

Construction : Here the code will be generated and tested.

Deployment : Here, a complete or partially complete Version of the Software is represented to the customers to evaluate and they give feedbacks based on the evaluation.

* Software Engineering process framework activities are done by a number of Umbrella activities.

Software Myths

Erroneous belief about software and the process that is used to build it - can be traced to the earliest days of computing.

Today, professionals recognize myths for what they are misleading attitudes that have caused serious problems for managers and practioners alike.

Myths →
- Management myths
- Customer myths
- practitioner's

Management myths :

* Members acquires all the information
* Adding more people can reduce the time gap
* The management can relax themselves by outsourcing its project.

Myth Members acquires all the information

Generally, there is a myth that the members of the organization acquire all the information containing procedures, principles and standards.

Reality: The Developers don't have information about all the established standards because they are often outdated, incomplete and unadaptable.

Myth Adding more people can reduce the time gap

Reality More the number of programmers, lesser will be the time gap. If a project is behind the schedule, adding more manpower will further delay it because new workers will take more time to learn about the ongoing project.

Myth The management can relax themselves by outsourcing its project

Reality When the organization outsources the software project, they suffer invariably.

These activities are applied Throughout a Software proje... help a Software team manage and control progress, Quality and risk.

* Software project tracking and control
* Risk management
* Software Quality assurance
* Technical reviews
* Measurement
* Software Configuration management
* Reusability management
* work product preparation and production

Software project tracking and control
It allows the Software team to assess progress against the project plan and take any necessary action to maintain the Scheduled

Risk management Assesses risks that may affect the outcome & the project or the quality of the product.

Software quality assurance :-
Defines and conducts the activities required to ensure Software Quality

Technical reviews. Assess Software engineering work products in an effort to uncover and remove errors before they are propagated to the next activity.

Measurement :- Defines and collects process, project, and product measures that assist the team in delivering S/w that meets Stakeholders.

Software Configuration management
Manages the effects of change Throughout the S/w process.

Reusability Management :- Defines criteria for work product reuse and establishes mechanisms to achieve reusable Components

Work product preparation and production :- encompasses the activities required to create work products such as models, documents, logs & lists.

mers myths

In many cases, the customer believes myths about Software because software managers and practitioners do little to correct misinformation.

Myths lead to false expectations (by the customer)

* Software is malleable as a result of which changes are easy to accommodate.
* To start coding, a general statement of need is enough.

Myth Software is malleable as a result of which changes are easy to accommodate

There is an enormous amount of labor required to have a change in the software after the release. It is not so easy to accommodate these changes.

Myth To start coding, a general statement of need is enough

The Developers can't read the customer's mind and requires detailed descriptions of the requirements, in order to start coding

Practitioner's myths:

Myths that are still believed by Software practitioners have been fostered by 50 years of programming culture

* once the code is delivered, the Software can be called complete
* Software's Success depends on the product's produced quality.
* The unnecessary documentation is required in S/w engineering, which further slows down the growth rate of a project.
* The Assessment of the Software quality can be addressed after the execution of the program.
* The product, which is delivered after the project's completion can be called working program.

M:- Once the code is delivered, the software can be called

R:- In really, more than 60% of the efforts are expended in the delivery of the software to the earlier user.

M:- The software's success depends on the product's quality

R:- The project does not become successful on the quality of the programs because both the software configuration and documentation also play an important role in its success.

M- The unnecessary documentation is required in software engineering, which further slows down the growth rate of a project

R:- The proper documentation enhances the project's quality and results in reduction of the rework. Also this field is just about creating quality at all the level of the project.

Myth:- The assessment of the software quality can be addressed after the execution of the program.

R:- During any phase of the soft development process, the software's quality can be measured just by applying the mechanism of quality assurance.

Myth:- The product, which is delivered after the project's completion can be called working program.

Reality:- The deliverables of a successful project don't only consist working program, but also the documentation which can guide the users about how to use the software.

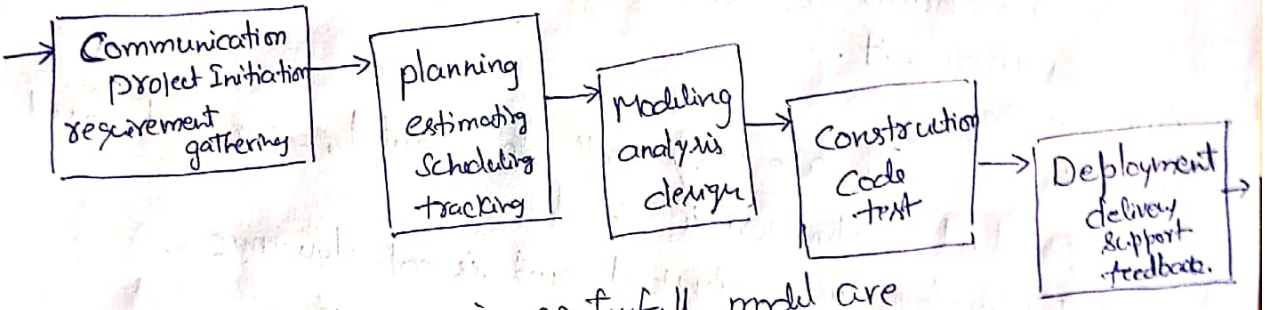
ware Development procen models

Waterfall Model
- classical waterfall model
- Iterative waterfall model

classical waterfall model

⇒ In waterfall model, each phase must be completed before the next phase can begin and there is no ~~phasing~~ overlapping in the phases.

⇒ This is a classical model which follows systematic and sequential approach (linear sequential flow). The outcome of one phase acts as the input for the next phase.



The Sequential phases in waterfall model are
* Requirement Gathering & Analysis
* System Design
* Implementation (coding)
* Integration & Testing
* Deployment of System
* Maintainance

Requirement Gathering & Analysis
All possible requirements of the system to be developed are captured in this phase. These are prepared like document i.e SRS (Software Requirement Specification)

System Design : This system design helps in specifying H/w and System requirements and helps in defining the overall system architecture.

Implementation : The system is first developed in small prog. called units, which are integrated in the next phase. Each

unit developed and tested for its functionality, which is referred to as Unit Testing.

Testing → All the units developed in the implementation are integrated into a System after testing of each unit called integration testing.

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.

⇒ Maintenance is done to deliver these changes in the customer environment.

Applications:

* Requirements are very well documented, clear and fixed
* product definition is stable
* Technology is understood and is not dynamic
* There are no ambiguous requirements
* The project is short.

Advantages

→ It allows for departmentalization & control
→ Development of each stage one by one.

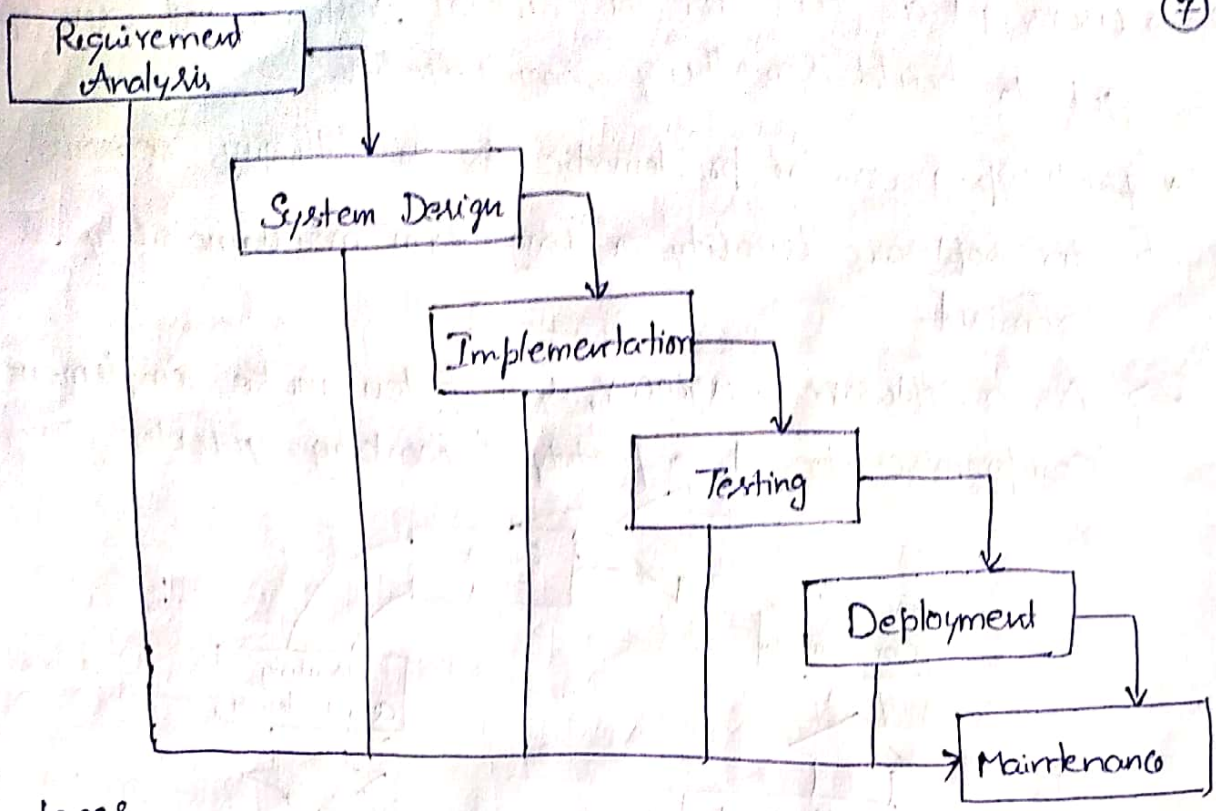
Disadvantages

→ No back tracking
→ If change something that was not well-documented communication phase.

Iterative Waterfall Model

Development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements.

⇒ This process is then repeated, producing a new version of the software for each cycle of the model.



Advantages :

-> Improving the product step by step
-> Less time is spent on documenting.

Disadvantages :

-> Each phase of an iteration is rigid with no overlaps
-> Design issues may arise because not all requirements are gathered up front for the entire lifecycle.

✓ Evolutionary Process Models

This approach is based on the idea of developing an initial software from few specifications and modifying this according to user requirements.

* prototyping
* Spiral model

Prototyping : Customer defines a set of general objectives for software, but does not clearly specify requirements.

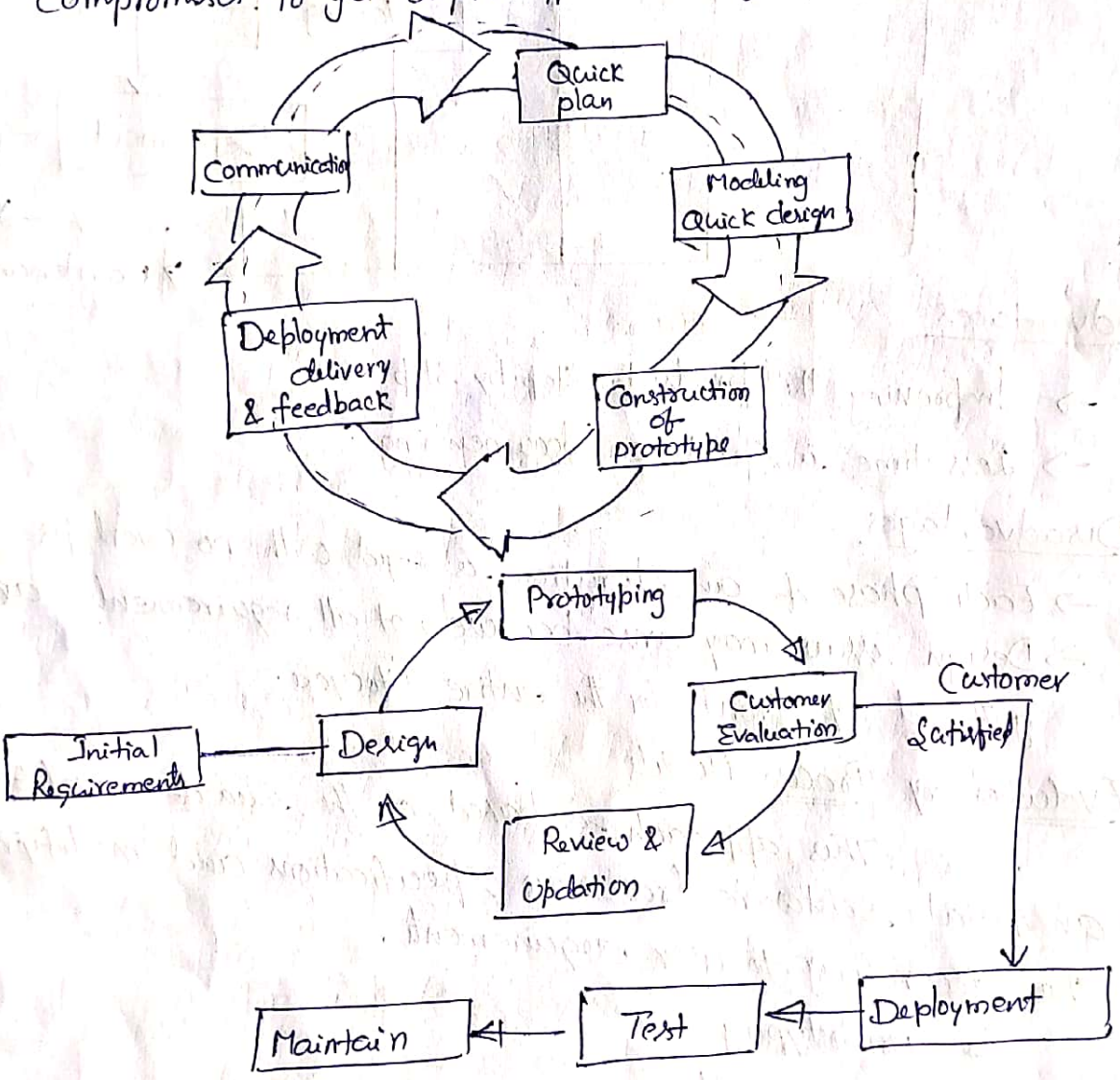
=> The Developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system (or) human machine interaction should take. In this situations, a prototyping may offer the best approach.

⇒ users get a feel for the actual system, and develop get to build something immediately.

* Prototyping can be problematic for the following reasons.

⇒ for software quality or long-term maintainability is compromised.

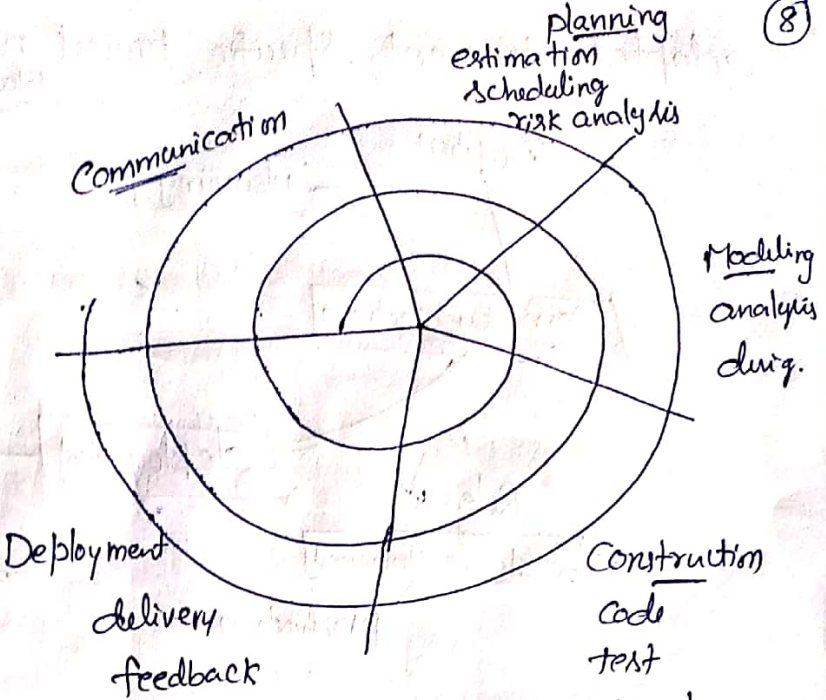
⇒ As a software engineer, you often make implementation compromises to get a prototype working quickly



Spiral Model :

This model is suitable throughout the life of the computer software that combines iterative nature of prototyping with the systematic aspects of the water fall model.

⇒ The Spiral model stresses on risk analysis. As evolutionary process starts, the software team perform activities around spiral in a clockwise direction, beginning at the centre.



=> Each pass in planning region results in project plan adjustments

=> project manager adjusts the no. of iterations to complete the software

=> The Spiral model reduces risks before they become problematic.

Advantages

* changing requirements can be accommodated
* Requirements can be captured more accurately
* Development can be divided into smaller parts and the risky parts can be developed earlier

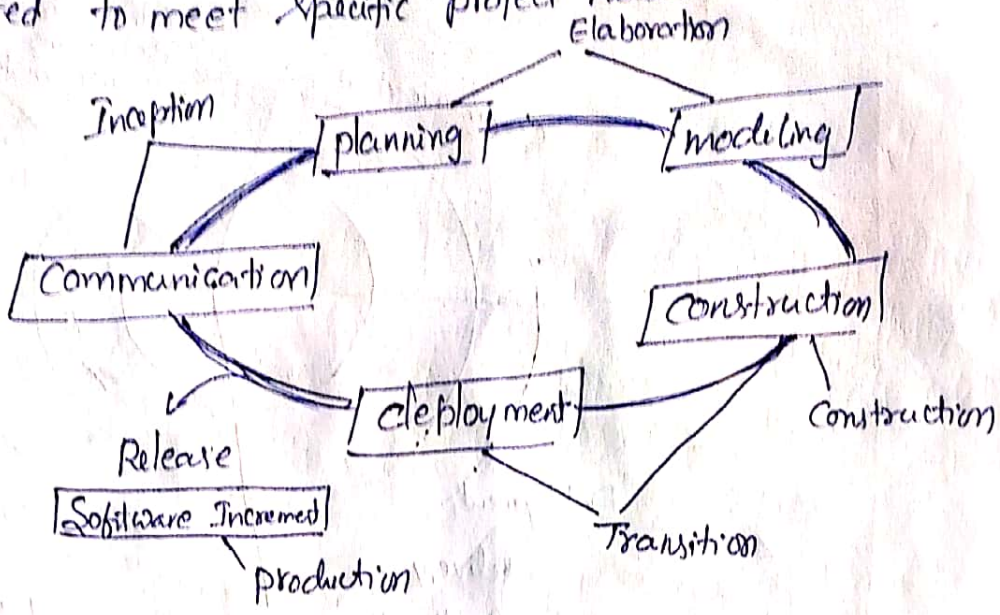
Disadvantages:

* Management is more complex
* End of the project may not be known early
* process is complex
* Large no. of intermediate stages requires excessive docu.

Unified process

* Jacobson, Rumbaugh and Booch developed the unified process,
* The iterative, incremental model proposed by the UP should be

adapted to meet specific project needs.



Inception phase :-
This phase of the UP encompasses both customer communication and planning activities. By collaborating with stakeholders, business requirements for the software are identified.

Elaboration phase :- It encompasses the communication and modelling activities of the generic process model
=) Elaboration refines and expands the preliminary use cases that were developed as part of the inception phase

Construction phase
The 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 software components for end users.

transition phase :-
The phase of the UP includes the later stages of the generic construction activity and the first part of the generic deployment activity

production phase :- The phase of the UP coincides with the deployment activity of the generic process

eme programming and agile proces - Kent Beck, ward & jeffries in 1991

What is Extreme programming

xp is a lightweight, efficient, low risk, flexible, predictable, Scientific, and fun way to develop a software.

⇒ Extreme programming (xp) was conceived and developed to address the specific needs of software development by small teams in the face of vague and changing requirements

⇒ Extreme programming is one of the Agile pro software develop -ment methodologies.

⇒ It provides values and principles to guide the team behaviours

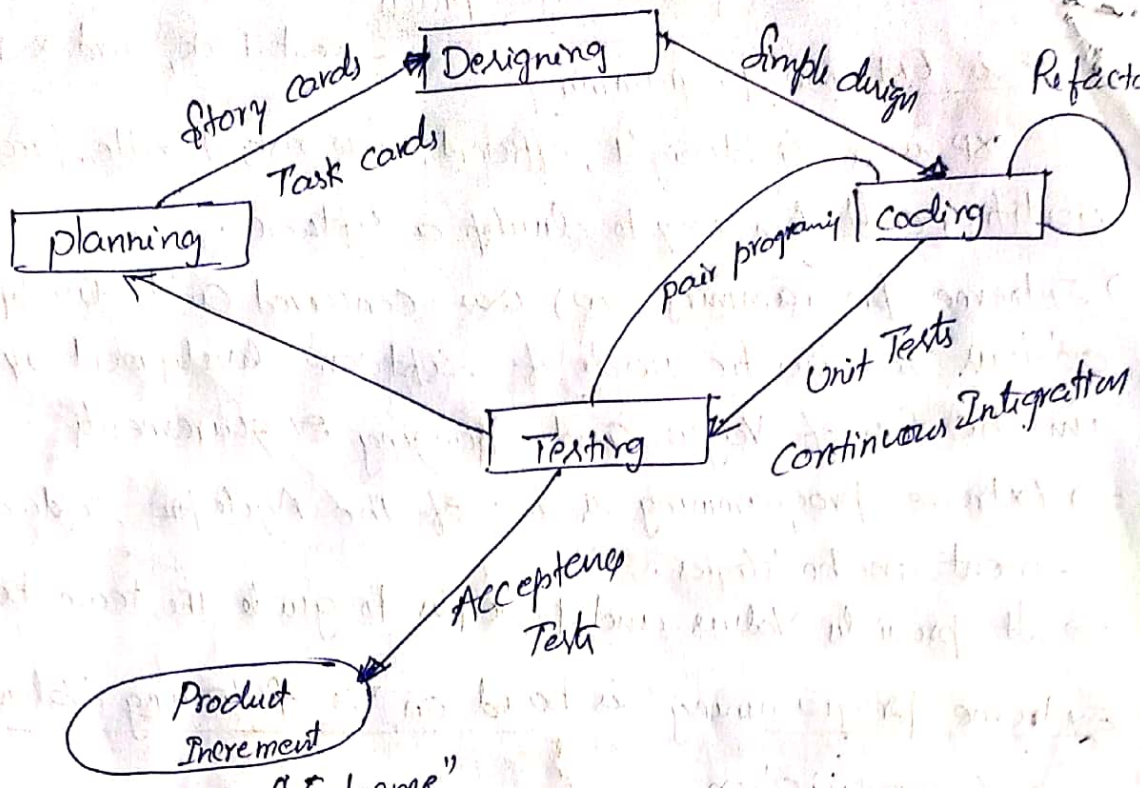
Extreme programming is based on the following values

* Communication
* Simplicity
* Feedback
* Courage
* Respect

Extreme programming in a NUTSHELL

* Starting with a simple design just enough to code the features at hand and redesigning when required
* programming in pairs (called pair programming) with two programmers at one screen, taking turns to use the keyboard. The other constantly reviews and provides inputs.
* Integrating and testing the whole system several times a day
* putting a minimal working system into the production quickly and upgrading it whenever required
* keeping the customer involved all the time and obtaining constant feedback.

Iterating ~~facilitates~~ facilitates the accommodating changes as the software evolves with the changing requirements



Why is it called "Extreme"

Extreme programming takes the effective principles and practices to extreme levels

* Code reviews are effective as the code is reviewed all the time
* Testing is effective as there is continuous regression and Testing
* Design is effective as everybody needs to do refactoring daily
* Integration testing is important as integrate and test several times a day

Advantages

* Cancelled project
* costs incurred in changes
* Rapid development
* Immediate responsiveness to the Customer's Changing requirements
* High customer Satisfaction
* Team cohesion and employee satisfaction