MANAGEMENT OF SOFTWARE SYSTEMS

MODULE 1

Important questions

1. What is software Process? Why is it important ?

Software Process is a sequence of activities that lead to the production of software Product . The necessity of selecting and following a formal process for software development is to provide desired discipline to deliver a quality product for business success and to avoid wastage of time, money, demoralization in developers etc.

2. List the Fundamental Activities of software Process

a. These are four key process activities, which are common to all software processes. These activities are:

b. **Software specifications:** The functionality of the software and constraints on its operation must be defined.

c. **Software development:** The software to meet the requirement must be produced.

d. **Software validation:** The software must be validated to ensure that it does what the customer wants.

e. **Software evolution:** The software must evolve to meet changing client needs.

3. Define Software engineering?List two types of software products .

Software engineering is an engineering discipline that is concerned with all aspect of software production.

- **Engineering discipline**
  - Using appropriate theories and methods to solve problems within the organizational and financial constraints.
- **All aspects of software production**
  - Not just technical process of development. Also project management and the development of tools, methods etc. to support software production.

Products: Generic products and customized products.

Generic Products

- These are stand-alone systems that are developed by a software-company.
- These are sold to any customer who is able to buy them.
- The company which develops the software controls the software-specification.
- For example:

→ databases

→ word processors &
→ drawing packages.

Customized Products
 These are systems which are developed for a particular customer.
 A software-contractor develops the software especially for that customer.
 The company which is buying the software controls the software-specification.
 The software-developers must work to that specification.
 For example:
→ electronic-device control systems &
→ air-traffic control systems.

4. **Why professional software that is developed for a customer is not simply the programs that have been developed and delivered.**

Software is not just theprograms themselves but also all associated documentation, libraries, support websites, and configuration data that are needed to make these programs useful. A professionally developed software system is often more than a single program. A system may consist of several separate programs and configuration files that are used to setup these programs. It may include system documentation, which describes the structure of the system, user documentation, which explains how to use the system, and websites for users to download recent product information. This is one of the important differences between professional and amateur softwaredevelopment

5. **Software engineering is intended to support professional software development rather than individual programming. Justify?**

- The key distinctions are that professional software is intended for use by someone apart from its developer and that teams rather than individuals usually develop the software. It ismaintained and changed throughout its life.
- It includes techniques that support program specification, design, and evolution, none of which are normally relevant for personal software development.

6. **How software engineering is different from Computer Science and system engineering?**
   - **Computer science** is concerned with the theories and methods that underlie computers and software systems, whereas software engineering is concerned with the practical problems of producing software. Some knowledge of computer science is essential for software engineers in the same way that some knowledge of physics is essential for electrical engineers.

- **System engineering** is concerned with all aspects of the development and evolutionof complex systems where software plays a major role. System engineeringis therefore concerned with hardware development, policy and process design,and system deployment, as well as software engineering.

7. What is Software Engineering Code of Ethics ?

The Software Engineering Code of Ethics focuses on providing high-quality software. They are committed to analyzing, specifying, designing, developing, testing, and maintaining software that is beneficial and effective for the company or clientAccording to the Association for Computer Machinery, the Software Engineering Code of Ethics is as follows:

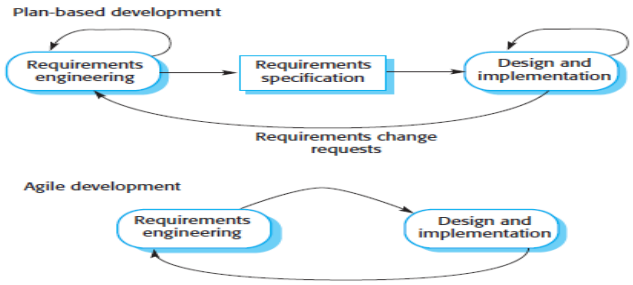
1. **PUBLIC** - Software engineers shall act consistently with the public interest.
   2. CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
   3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
   4. JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.
   5. MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
   6. PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
   7. COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.
   8. SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

8. Compare waterfall model incremental model and spiral model for software development.(write and elaborate according to the weightage of marks)

Waterfall Model vs Incremental Model vs Spiral Model:

Parameter	Waterfall Model	Incremental Model	Spiral Model
1. Handle Large Project	Not Appropriate	Not Appropriate	Appropriate
2. Detailed Documentation	Necessary	Yes, but not much	Yes
3. Cost	Low	Low	Expensive
4. Risk	High	Low	Medium
5. Time-frame	Very long	Long	Long
6. Testing	After completion of coding phase	After every iteration	At the end of the engineering phase
7. Framework	Linear	Linear+Iterative	Linear+Iterative

9. Difference between agile and plan driven development ?



Plan-driven processes are processes where all of the **process** activities are planned in advance and progress is measured against this plan

In agile processes, planning is incremental and it is easier to change the plan and the software to reflect changing customer requirements

10 . State Agile manifesto ?

- Individuals and interactions over processes and tools Working software over comprehensive documentation Customer collaboration over contract negotiation Responding to change over following a plan .

10. List Principles of agile methods?

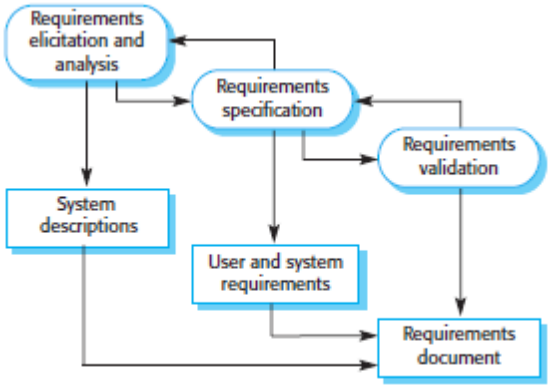
Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

11. List the different approaches to reduce the cost of rework?

Change anticipation, where the software process includes activities that cananticipateor predict possible changes before significant rework is required. Forxample, a prototype system may be developed to show some key features ofthe system to customers. They can experiment with the prototype and refine their requirements before committing to high software production costs.

Change tolerance, where the process and software are designed so that changescan be easily made to the system. This normally involves some form of incrementaldevelopment. Proposed changes may be implemented in increments thathave not yet been developed.

12. What are the different elements of requirement engineering Process?



There are three main activities in the requirements engineering process:

1. *Requirements elicitation and analysis* This is the process of deriving the system requirements through observation of existing systems, discussions with potential users and procurers, task analysis, and so on. This may involve the development of one or more system models and prototypes. These help you understand the system to be specified.
2. *Requirements specification* Requirements specification is the activity of translating the information gathered during requirements analysis into a document that defines a set of requirements. Two types of requirements may be included in this document. User requirements are abstract statements of the system requirements for the customer and end-user of the system; system requirements are a more detailed description of the functionality to be provided.
3. *Requirements validation* This activity checks the requirements for realism, consistency, and completeness. During this process, errors in the requirements document are inevitably discovered. It must then be modified to correct these problems.

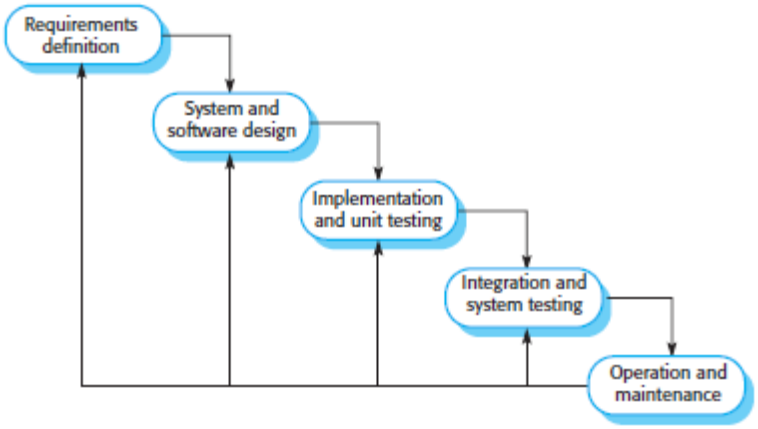
13.Define throw away prototype?

Prototypes should be discarded after development as they are not a good basis for a production system:

a. It may be impossible to tune the system to meet non-functional requirements;
b. Prototypes are normally undocumented;
c. The prototype structure is usually degraded through rapid change;
d. The prototype probably will not meet normal organizational quality standards

PART B

1. Explain the phases of Waterfall model ?



The stages of the waterfall model directly reflect the fundamental software development

activities:

1. ***Requirements analysis and definition***The system's services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.
2. ***System and software design***The systems design process allocates the requirements to either hardware or software systems. It establishes an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.
3. ***Implementation and unit testing***During this stage, the software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.
4. ***Integration and system testing***The individual program units or programs are integrated and tested as a complete system to ensure that the softwarerequirements have been met. After testing, the software system is deliveredto the customer.
5. ***Operation and maintenance:***Normally, this is the longest life-cycle phase. The system is installed and put into practical use. Maintenance involves correctingerrors that were not discovered in earlier stages of the life cycle, improving theimplementation of system units, and enhancing the system's services as newrequirements are discovered.

Water Fall model problems

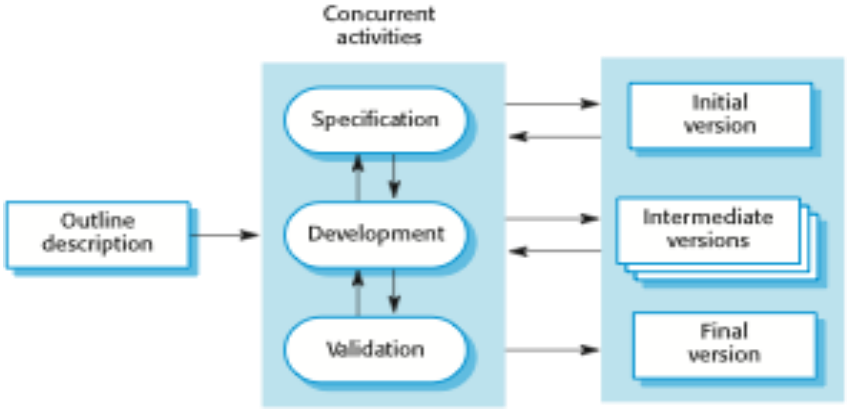
- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
  - Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
  - Few business systems have stable requirements.
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.

- In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

Advantages of Waterfall model

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model phases are processed and completed one at a time. Phases do not overlap.
- Waterfall model works well for smaller projects where requirements are clearly defined and very well understood.

2. Explain about incremental development Model ?



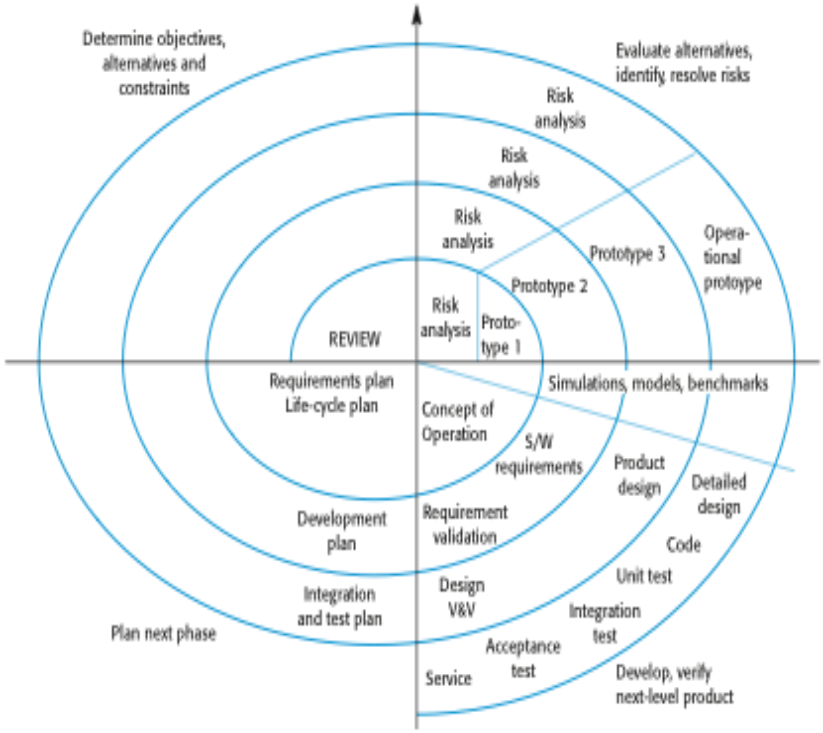
- Incremental Development is based on the idea of developing an initial implementation, exposing this to user comment and evolving it through several versions until an adequate system has been developed. Specification, development and validation activities are interleaved rather than separate, with rapid feedback across activities.
- It is better for business,e-commerce and software systems.
- The customer can evaluate the system at a relatively early stage in the development to see if it delivers what is required. If not, then only the current increment has to be changed and possibly new functionality defined for later increments.
- **Incremental development Benefits**
- The cost of accommodating changing customer requirements is reduced.
  - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
- It is easier to get customer feedback on the development work that has been done.
  - Customers can comment on demonstrations of the software and see how much has been implemented.
- More rapid delivery and deployment of useful software to the customer is possible.

- Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Incremental development problems

- The process is not visible.
  - Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
- System structure tends to degrade as new increments are added.
  - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

3. Explain about Boehm spiral model ? (model deals with risk)



- **Objective setting**
  - Specific objectives for the phase are identified.
  - Constraints on the process and the product are identified and a detailed management plan is drawn up.
  - Project risks are identified.
- **Risk assessment and reduction**
  - Risks are assessed and activities put in place to reduce the key risks.
- **Development and validation**

- A development model for the system is chosen which can be any of the generic models.
- **Planning**
  - The project is reviewed and the next phase of the spiral is planned.
- Spiral model has been very influential in helping people think about iteration in software processes and introducing the risk-driven approach to development.
- In practice, however, the model is rarely used as published for practical software development.

Advantages of Spiral Model

1. **Risk Managing Feature:** The projects including several undiscovered risks that occur with the proceeding development can use Spiral Model in this case since it is the best development model to grasp concerning the risk analysis feature and administration at every phase.
2. **Suitable for more extensive projects:** Using the Spiral Model in long and intricate projects is highly recommended.
3. **Docile in terms of Requirements:** Replacement demands in the project requirements at a much later phase can be included accurately using the Spiral Model.
4. **Client Satisfaction:** Clients can observe the evolution of the product from the early stage of the software development, and thus, get habituated with the method by utilizing it before finishing the entire project.
5. **Early Estimation of Cost:** Project estimations regarding programming, cost, and other factors become more accessible since the project progresses forward in a spiral while reaching completion.

4. Discuss the role of software prototyping in software development ? .

Software prototyping is the activity of creating prototypes of software applications, i.e., incomplete versions of the software program being developed.

- A prototype is an initial version of a system used to demonstrate concepts and try out design options, and find out more about the problem and its possible solutions.
- Where a version of the system or part of the system is developed quickly to check the customer requirements .
- Rapid ,iterative development of the prototype is essential ,so that costs are controlled and system stakeholders can experiment with the prototype early in the software process.
- A prototype can be used in:
  - The requirements engineering process to help with requirements elicitation and validation;
  - In design processes to explore particular software solutions options and develop a UI design;
  - In the testing process to run back-to-back tests.

Advantages

- Improved system usability.
- A closer match to users' real needs.
- Improved design quality.
- Improved maintainability.
- Reduced development effort.

4.Explain about Process activities ?

The four basic process activities of **specification, development, validation, and evolution** are organized differently in different development processes

- **The process of establishing what services are required and the constraints on the system's operation and development.**
- **Requirements engineering process**
  - Feasibility study
    - Is it technically and financially feasible to build the system?
    - Developed within the existing budgetary constraints.(cost effective)
  - Requirements elicitation and analysis
    - What do the system stakeholders require or expect from the system?
    - Observations from existing systems, discussions with potential users ,task analysis.
    - This may involve the development of one or more models and prototypes
  - Requirements specification
    - Is the activity of translating the information gathered during the analysis activity into a document
    - Two types of requirements
    - User requirements :are abstract statements of the system requirements for the customer and end user of the system.
    - System requirements are a more detailed description of the functionality to be provided.
  - Requirements validation
    - Checking the validity of the requirements(consistent/complete) .

Software Design and implementation

- The process of converting the system specification into an executable system.
- Software design
  - Design a software structure that realises the specification;
- Implementation

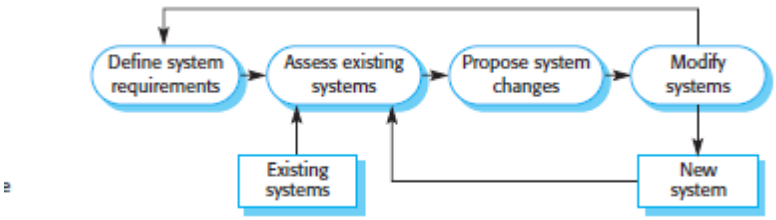
- Translate this structure into an executable program;

- The activities of design and implementation are closely related and may be inter-leaved.
- *Architectural design,* where you identify the overall structure of the system, the principal components (sometimes called sub-systems or modules), their relationships and how they are distributed.
- *Interface design,* where you define the interfaces between system components. This interface specification must be unambiguous.
- *Component design,* where you take each system component and design how it will operate.
- *Database design,* where you design the system data structures and how these are to be represented in a database. ,depends on whether an existing database is to be reused or a new database is to be created.

Software Validation

- System components are tested(component defects are discovered early in the process)
- then the integrated system is tested,(interface problems are found when the system is integrated)
- Finally the system is tested with the customers data.
- Stages in testing
    a. Development testing
    b. System testing
    c. Acceptance testing .

Software Evolution :



5. Explain about XP (Extreme Programming) - refer notes AND ppt

Extreme Programming -agile method.

Features of XP

EXTREME PROGRAMMING (XP)

- Extreme programming is perhaps the best known and most widely used of the agile methods.

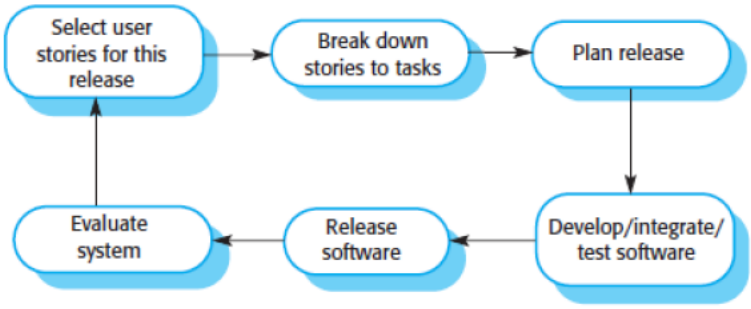


Figure 17.4 The extreme programming release cycle

- Extreme programming practices are(Figure 17.4):

1) Incremental Planning

- Requirements are recorded on Story Cards.
- The Stories to be included in a release are determined by the time available and their relative priority.
- The developers break these Stories into development 'Tasks'.

2) Small Releases

- The minimal useful set of functionality that provides business value is developed first.
- Releases of the system are frequent and incrementally add functionality to the first release.

3) Simple Design

- Enough design is carried out to meet the current requirements and no more.

4) Test-first development

- An automated unit test framework is used to write tests for a new piece of functionality.
- Then, that functionality itself is implemented.

5) Refactoring

- All developers are expected to refactor the code continuously as soon as possible code improvements are found.
- This keeps the code simple and maintainable.

6) Pair Programming

- Developers work in pairs.
- They check each other's work.
- They provide the support to always do a good job.

7) Collective Ownership

- The pairs of developers work on all areas of the system.
- Thus, all the developers own all the code.
- Anyone can change anything.

8) Continuous Integration

- As soon as work on a task is complete it is integrated into the whole system.
- After any such integration, all the unit tests in the system must pass.

9) Sustainable Pace

- Large amounts of overtime are not acceptable.
- This is because the net effect is often to reduce code quality and medium-term productivity.

10) On-site Customer

- A representative of the end-user of the system (the Customer) should be available full time for the use of the XP team.
- The customer is a member of the development-team.
- The customer is responsible for bringing system requirements to the team for implementation.

XP programming practices

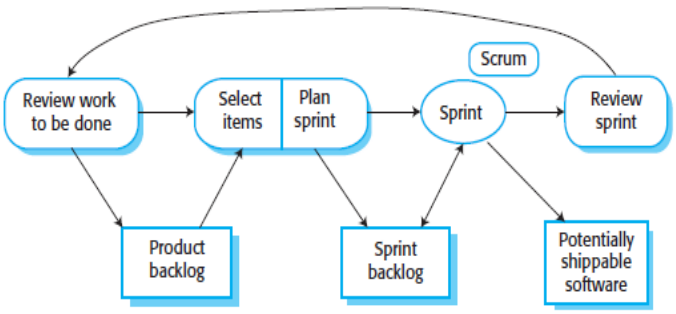
Principle or practice	Description
Incremental planning	Requirements are recorded on story cards and the stories to be included in a release are determined by the time available and their relative priority. The developers break these stories into development 'Tasks'. See Figures 3.5 and 3.6.
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.

Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site customer	A representative of the end-user of the system (the customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

6. Explain about agile project management technique ?

SCRUM

- The Scrum approach is a general agile method and focus is on managing iterative development rather than specific agile practices.
- The Scrum Process (refer ppt or notes)



SCRUM BENEFITS

- The product is broken down into a set of manageable and understandable chunks.
- Unstable requirements do not hold up progress.
- The whole team have visibility of everything and consequently team communication is improved.
- Customers see on-time delivery of increments and gain feedback on how the product works.
- Trust between customers and developers is established and a positive culture is created in which everyone expects the project to succeed.

7. What is Pair programming ? List its advantages ?

Pair programming is a style of programming in which two programmers work side-by-side at one computer, sharing one screen, keyboard and mouse, continuously collaborating on the same design, algorithm, code or test.

- Many mistakes are detected at the time they are typed, rather than in QA Testing or in the field.
- The end defect content is statistically lower.
- The designs are better and code length shorter.
- The team solves problems faster.
- People learn significantly more about the system and about software development.
- The project ends up with multiple people understanding each piece of the system