Software Engineering Fundamentals

Bachelors in Software Engineering

III semester

Chapter one

Introduction to Software Engineering

OUTLINES:

1.1 Overview of software engineering and its application domain
1.2 Software process models (waterfall, iterative, spiral, rapid application development, reuse oriented)
1.3 Introduction to agile development (extreme programming, scrum)
1.4 Software Crisis and Myths
1.5 Four P's of software

Software

- *Software is a set of*

  (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.
- *Software is developed or engineered, it is not manufactured in the classical sense.*
- *Software doesn't "wear out."*
- *Although the industry is moving toward component-based construction, most software continues to be custom-built.*

- *Various attributes that are associated with a software decides whether it is good or bad.*
- *Specific set of attributes which one expects from software, depends on application*
  - *Banking system must be secure*
  - *Telephone system must be reliable*
  - *Interactive game must be responsive*
- *Some attributes are:*
  - *Maintainability*
  - *Dependability*
  - *Efficiency*
  - *usability*

Maintainability

- Maintenance is very important and is critical to both software engineer and its user.
- Software should be written in such a way that it may evolve to meet the changing need of customer.
- Change in the software is required because of changing business environment, which are unavoidable.

Dependability

- Software dependability has a range of sub-attributes:
  - Reusability through assured performance
  - Security
  - Safety
- Dependable software should not cause physical or economical damage in the event of system failure.

Efficiency

- Software should not make the wasteful use of the system resources such as memory, processor and storage.
- Efficiency therefore includes:
    a) Responsiveness
    b) Processing time
    c) Memory utilization

Usability

- Software becomes useful if it does not call for extra effort to be learnt.
- Usability increases with good documentation and an appropriate user interface.

Software Crisis

- **Software Crisis** is a term used in computer science for the difficulty of writing useful and efficient computer programs in the required time .
- software crisis was due to using same workforce, same methods, same tools even though rapidly increasing in software demand, complexity of software and software challenges.
- With increase in the complexity of software, many software problems arise because existing methods were insufficient.
- If we will use same workforce, same methods and same tools after fast increasing in software demand, software complexity and software challenges, then there arose some problems like software budget problem, software efficiency problem, software quality problem, software managing and delivering problem etc. This condition is called software crisis.

Reasons of Software Crisis

- Lack of communication between software developers and users.
- Increase in size of software.
- Increase in cost of developing a software.
- Project management problem.
- Duplication of efforts due to absence of automation in most of the software development activities.

Software Myths

- Software myths propagated misinformation and confusion.
- Software myths propagate false beliefs and confusion in the minds of management, users and developers
- Myths lead to false expectations and ultimately develop dissatisfaction among the users.

Software Myths – Management Perspective

- "We already have a book that is full of standards and procedures for building software. Won't that provide my people with everything they need to know?"
  - Not used, not up to date, not complete, not focused on quality, time, and money
- "If we get behind, we can add more programmers and catch up"
  - Adding people to a late software project makes it later
  - Training time, increased communication lines
- "If I decide to outsource the software project to a third party, I can just relax and let that firm build it"
  - Software projects need to be controlled and managed

Software Myths – Customer Perspective

- "A general statement of objectives is sufficient to begin writing programs – we can fill in the details later"

  – Ambiguous statement of objectives spells disaster

- "Project requirements continually change, but change can be easily accommodated because software is flexible"

  – Impact of change depends on where and when it occurs in the software life cycle (requirements analysis, design, code, test)

Software Engineering

- *Field of Computer Science*
- *Deals with building software systems which are so large*
- *IEEE defines SE as:*

  Application and approach of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software.
- *SE uses tools, methods and process with quality focus on system to be developed.*

Role of SE

- *SE researches, designs and develops software systems to meet with client requirements. Once the system has been fully designed, Software engineers then test, debug and maintain the system.*
- *SE translates vague (unclear) requirements and derives into precise specification.*
- *SE develops model for the system or application and understands the behavior and performance of the system.*
- *SE provides methods to schedule work, operate systems at various levels and obtain the necessary details of the software.*
- *SE promotes interpersonal and communication skills and management skills.*

Software Process

- A road map
- Helps to create a timely and high quality result.
- A software process (also knows as software methodology) is a set of related activities that leads to the production of the software. These activities may involve the development of the software from the scratch, or, modifying an existing system.

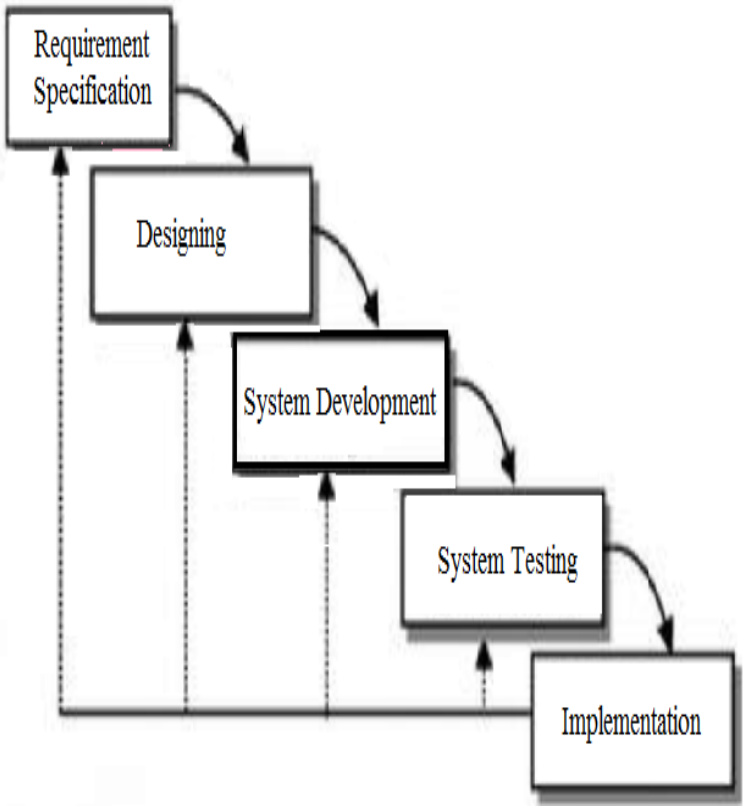
a) Communication	Understand objectives of the project, define software features and functions.
b) Planning	Initial study of: technical tasks to be conducted, risk that are likely, work schedule, etc.
c) Modeling	Create a sketch or models using Use-Case diagrams, DFD, ERD, etc.
d) Construction	Actual coding, testing, debugging and integration
e) Deployment	Delivery to customer and get feedback

Software Process Models

- A software process model is a simplified representation of a software process. Each model represents a process from a specific perspective.
- they prescribe a set of process elements such as framework activities, software engineering actions, tasks, and work products, quality assurance, and change control mechanism for each project.

Waterfall Model

- This model was first introduced by Dr. Winston W. Royce in a paper published in 1970.
- The waterfall model is a classical model used in system development life cycle to create a system with a linear and sequential approach.
- It is termed as waterfall because the model develops systematically from one phase to another in a downward fashion.
- This model is divided into different phases and the output of one phase is used as the input of the next phase.
- Every phase has to be completed before the next phase starts and there is no overlapping of the phases.



Waterfall Model (contd.)

1. **Requirement Analysis and Specification**
   - This phase identifies and documents the exact requirements of the system through feasibility study.
   - This is done by combination of customers, developer and organization.
2. **Design**
   - From the specified requirement, software engineers develop a design.
   - It can be split into two phases:
   - High Level Design and Detailed Design.
   - The High-level design deals with the overall module structure and organization.
   - Then, it is refined by designing each module (i.e. detailed design). Screen layout, business rules, process diagrams, DFD, ERD, etc. are used in this phase.

Waterfall Model (contd.)

3. System Development

- Produces the actual code that will be developed to the customers.
- Also develops prototypes, modules, test drivers, etc.
- Testing is done at program modules and at various levels.
- Includes unit testing, black-box testing, white-box testing, etc.

4. System Testing

- All modules are tested individually in the previous phase are then integrated to make a complete system,
- and performance test is done on the whole system.

5. Implementation

- Once the system passes all the tests, it is delivered to the customers
- and enters the maintenance phase.
- Any modifications made to the system, after initial delivery, are usually attributed to this phase.

Waterfall Model (contd.)

Advantages:

- Easy to understand and implement
- Suitable for simple project
- Well understood milestones.
- Process and results are well documented.

Disadvantages:

- Difficult to trace back
- No working software is produced until late during the life cycle.
- Not suitable for large product
- Risk analysis is not performed.
- Cannot accommodate changing requirements.
- Product may be outdated at the end.

Waterfall Model (contd.)

When to use it?

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

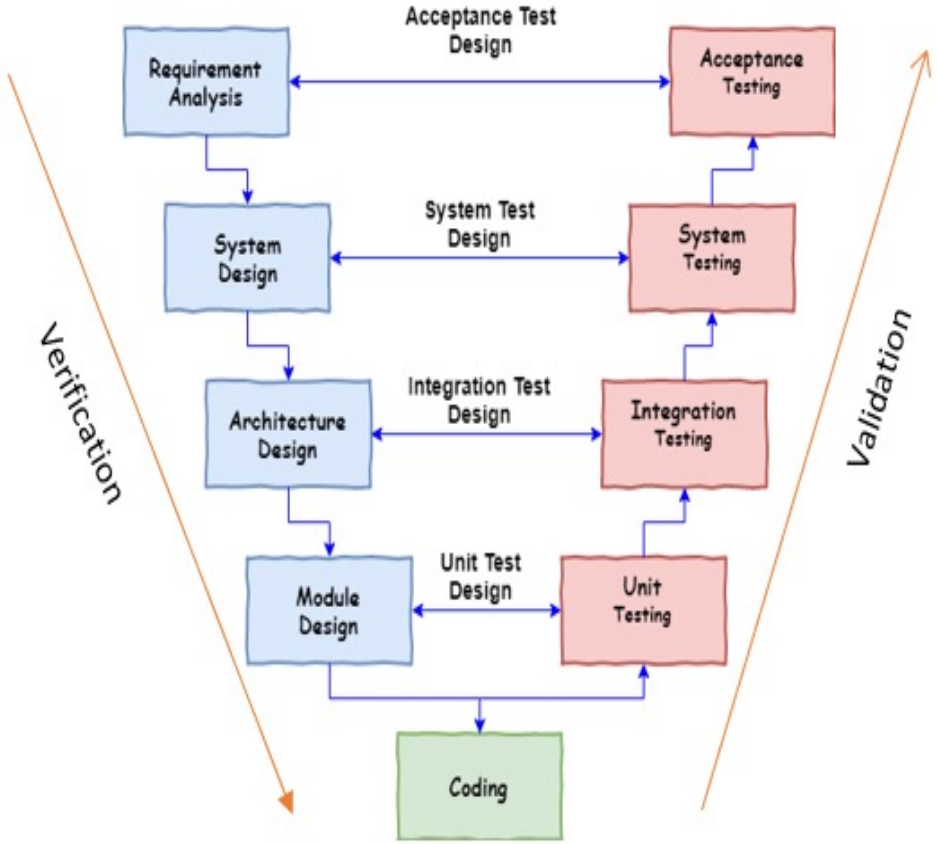
Major Drawback:

The major drawback of the Waterfall model is we move to the next stage only when the previous one is finished and there is no chance to go back if something is found wrong in later stages.

The V Model

- The major drawback of the Waterfall model is we move to the next stage only when the previous one is finished and there is no chance to go back if something is found wrong in later stages.
- V-model provides means of testing of software at each stage in reverse manner.
- The V-Model is SDLC model where execution of processes happens in a sequential manner in V-shape.
- It is also known as Verification and Validation Model.
- V-Model is an extension of the Waterfall Model and is based on association of a testing phase for each corresponding development stage.
- This means that for every single phase in the development cycle there is a directly associated testing phase.
- This is a highly disciplined model and next phase starts only after completion of the previous phase.

The V Model (Contd.)



- The left-hand side shows the development activities and the right-hand side shows the testing activities.
- In the development phase, both verification and validation are performed along with the actual development activities.
- Verification and Validation phases are joined by coding phase in V-shape.
- Thus it is called V-Model.
- This demonstrates that testing can be done in all phase of development activities as well

The V Model (Contd.)

Design Phases:

1. Business Requirement Analysis:

- product requirements are understood from the customer perspective.
- The users are interviewed and a document called the User Requirements Document is generated.
- this document would serve as the guideline for the system designers in the system design phase.

2. System Design:

- system developers analyze and understand the business of the proposed system by studying the user requirements document.
- They figure out possibilities and techniques by which the user requirements can be implemented.
- System design comprises of understanding and detailing the complete hardware and communication setup for the product under development.
- System test plan is developed based on the system design.

The V Model (Contd.)

3. Architecture Design:

- This is also referred to as *High Level Design (HLD)*.
- This phase focuses on system architecture and design.
- It provides overview of solution, platform, system, product and service/process.
- An integration test plan is created in order to test the pieces of the software systems ability to work together.

4. Module Design:

- also be referred to as low-level design.
- In this phase the actual software components are designed.
- It defines the actual logic for each and every component of the system.
- The designed system is broken up into smaller units or modules and each of them is explained so that the programmer can start coding directly.
- Unit test plan created and performed.

The V Model (Contd.)

Validation Phases:

- Unit testing
- Integration testing
- System testing
- User Acceptance testing

When to use?

- Where requirements are clearly defined and fixed.
- The V-Model is used when ample technical resources are available with technical expertise.
- Project is short.

The V Model (Contd.)

Advantages:

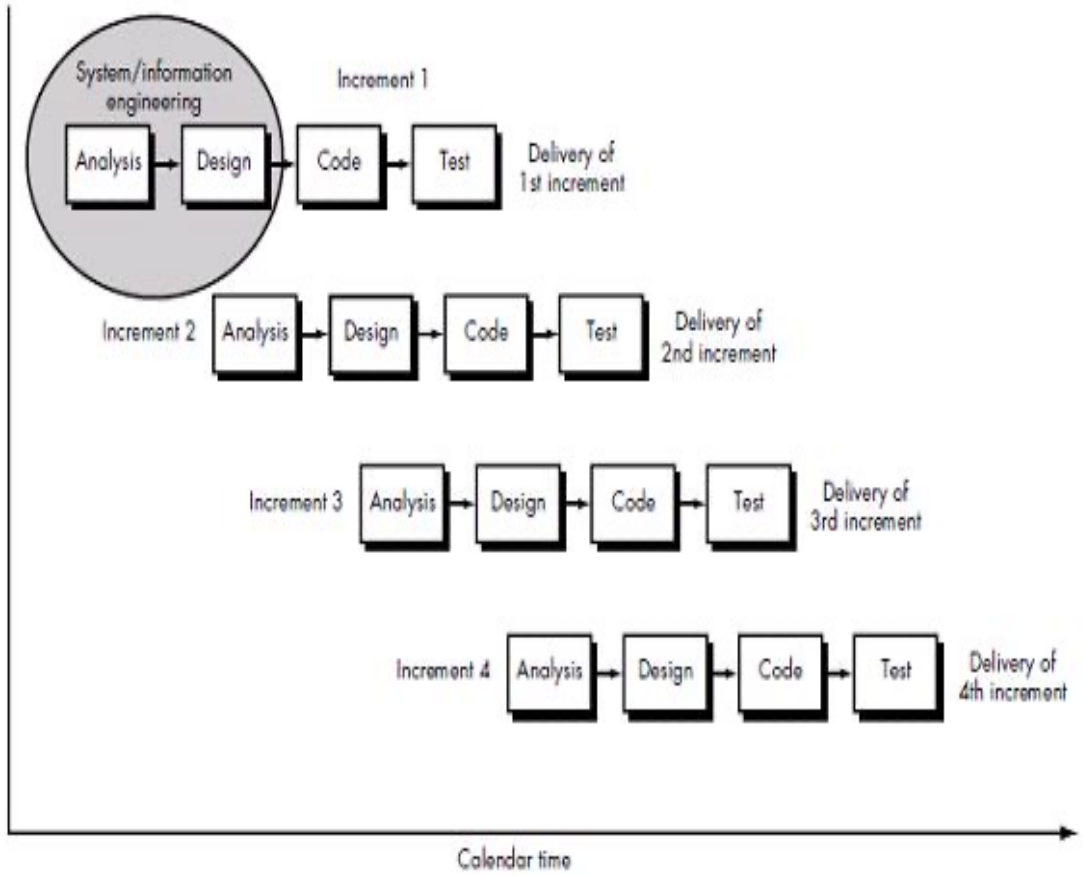
- This is a highly disciplined model and Phases are completed one at a time.
- V-Model is used for small projects where project requirements are clear.
- Simple and easy to understand and use.
- This model focuses on verification and validation activities early in the life cycle thereby enhancing the probability of building an error-free and good quality product.
- It enables project management to track progress accurately.

Disadvantages:

- High risk and uncertainty.
- It is not a good for complex and object-oriented projects.
- It is not suitable for projects where requirements are not clear and contains high risk of changing.
- This model does not support iteration of phases.
- It does not easily handle concurrent events.

Incremental Model

- product is designed, implemented and tested incrementally.
- Little more is added each time until the product is finished. It involves both development and maintenance.
- This is also known as the **Iterative Model**
- The basic idea of this model is that the software should be developed in increments, where each increment adds functional capability to the system until the full system is implemented.



Incremental Model (Contd.)

- The product is decomposed into a number of components, each of which is designed and built separately.
- Multiple development cycles take place here, making the life cycle a "multi-waterfall" cycle.
- Cycles are divided up into smaller, more easily managed modules.
- Each module passes through the requirements, design, implementation and testing phases.

For example: word processing software developed using this model might deliver:

- file management, editing and printing in the first increment,
- most sophisticated editing and document production capabilities in the second increment,
- Spelling and grammar checking in the third increment and so on.

Incremental Model (Contd.)

Advantages:

- Incremental Model allows partial utilization of the product and avoids a long development time.
- Supports changing environment
- Generates working software quickly and early during the software life cycle.
- This model is more flexible and less costly to change scope and requirements.
- It is easier to test and debug as smaller changes are made during each iteration.
- In this model, the customer can respond to each built. During the life cycle, software is produced early which facilitates customer evaluation and feedback
- As testing is done after each iteration, faulty elements of the software can be quickly identified because few changes are made within any single iteration.

Incremental Model (contd.)

Disadvantages:

- As additional functional is added to the product at every stage, problems may arise related to system architecture which was not evident in earlier stages.
- It needs good planning and design at every step, more management attention is required.
- Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
- Total cost is higher than waterfall, more resources may be required.
- End of project may not be known, which is a risk.

Incremental Model (contd.)

When to use Iterative Enhancement model?

The below are the cases when we need to use Iterative Enhancement Model:

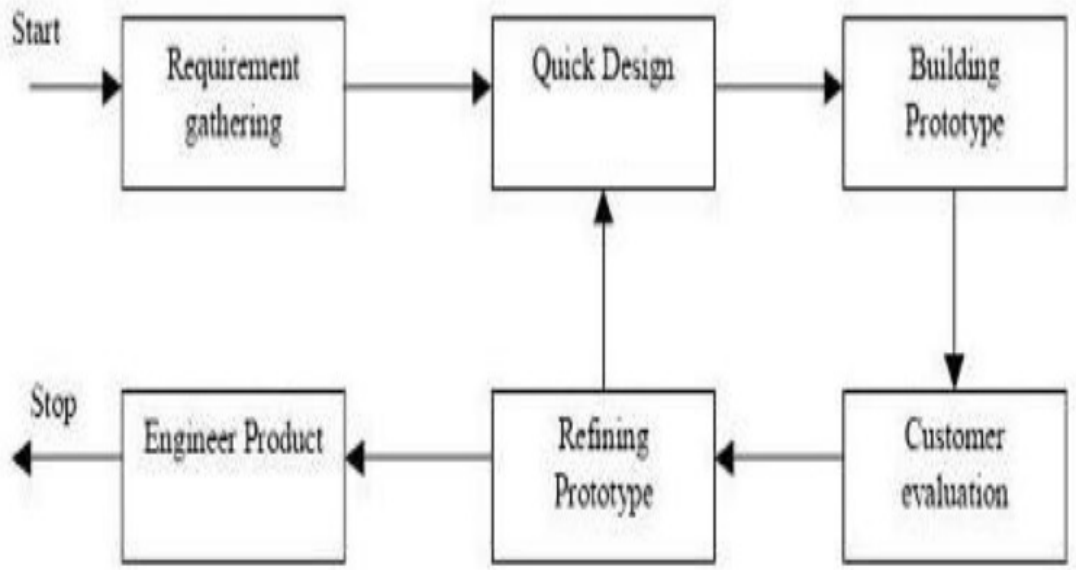
- Some functionalities or requested enhancements may evolve with time.
- There is a time to the market constraint.
- New technology is being adapted.
- There are some high-risk features and goals which may change in the future.

Evolutionary Model

- Evolutionary models are iterative
- suitable for new systems where no clear idea of the requirements, inputs and outputs parameters.
- produces an increasingly more complete version of the software with each iteration.
- The two common evolutionary process models are:
    1. Prototyping model
    2. Spiral Model

1. Prototype Model

- prototype (an early approximation of a final system or product)
- In this model, a prototype is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved.
- Then, the complete system is developed.
- The analyst works with users to determine the initial or basic requirements for the system.
- The analyst then quickly builds a prototype and gives it to the user.
- The analyst uses the feedback to improve the prototype and takes the newer version back to users.
- The process is repeated until the user is relatively satisfied



Prototyping Model

1. Prototype Model (Contd.)

Advantages:

- Users are actively involved in the development.
- Users get better understanding of the system being developed.
- Errors can be detected much earlier.
- Missing functionality can be identified quickly.
- Reduces project risk and less documentation.

Disadvantages:

- May increase the complexity of the system since the user may demand more functionality,
- Integration can be very difficult.
- Frequent improvement and rebuilding of software.
- Slower process.

When to use Prototyping model?

- When requirement is not clear.
- Useful in development of systems having high level of user interactions such as online systems.

2. Spiral Model

- This Spiral model, proposed by Barry Boehm (1985),
- is a combination of iterative development process model and sequential linear development model
- i.e. the waterfall model with a very high emphasis on risk analysis.
- It allows incremental releases of the product or incremental refinement through each iteration around the spiral.
- The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation.
- A software project repeatedly passes through these phases in each iteration/ spiral.

2. Spiral Model (Contd.)

i) Planning:

- This phase starts with gathering the business requirements in the baseline spiral.
- In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase.

ii) Risk Analysis:

- Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks,
- such as schedule slippage and cost overrun.
- If any risk is found during the risk analysis then alternate solutions are suggested and implemented.

2. Spiral Model (Contd.)

iii) Software Engineering / Development

- This phase refers to production of the actual software product at every spiral.
- Actual development and testing of the software takes place in this phase.
- It includes testing, coding and deploying software at the customer site.
- A POC (Proof of Concept) is developed in this phase to get customer feedback.
- In this phase software is developed, along with testing at the end of the phase.
- Hence in this phase the development and testing is done

iv) Evaluation:

- This phase allows the customer to evaluate the output of the project before the project continues to the next spiral.

2. Spiral Model (Contd.)

Advantages:

- Risk monitoring is one of the core parts which makes it pretty attractive, especially when you manage large and expensive projects. Moreover, such approach makes your project more transparent because, by design, each spiral must be reviewed and analyzed
- Customer can see the working product at the early stages of software development lifecycle
- Different changes can be added at the late life cycle stages
- Project can be separated into several parts, and riskier of them can be developed earlier which decreases management difficulties
- Project estimates in terms of schedule, costs become more and more realistic as the project moves forward, and loops in spiral get completed
- Strong documentation control

2. Spiral Model (Contd.)

Disadvantages:

- Since risk monitoring requires additional resources, this model can be pretty costly to use. Each spiral requires specific expertise, which makes the management process more complex. That's why this SDLC model is not suitable for small projects
- A large number of intermediate stages. As a result, a vast amount of documentation
- Time management may be difficult. Usually, the end date of a project is not known at the first stages

2. Spiral Model (Contd.)

When to use Spiral model:

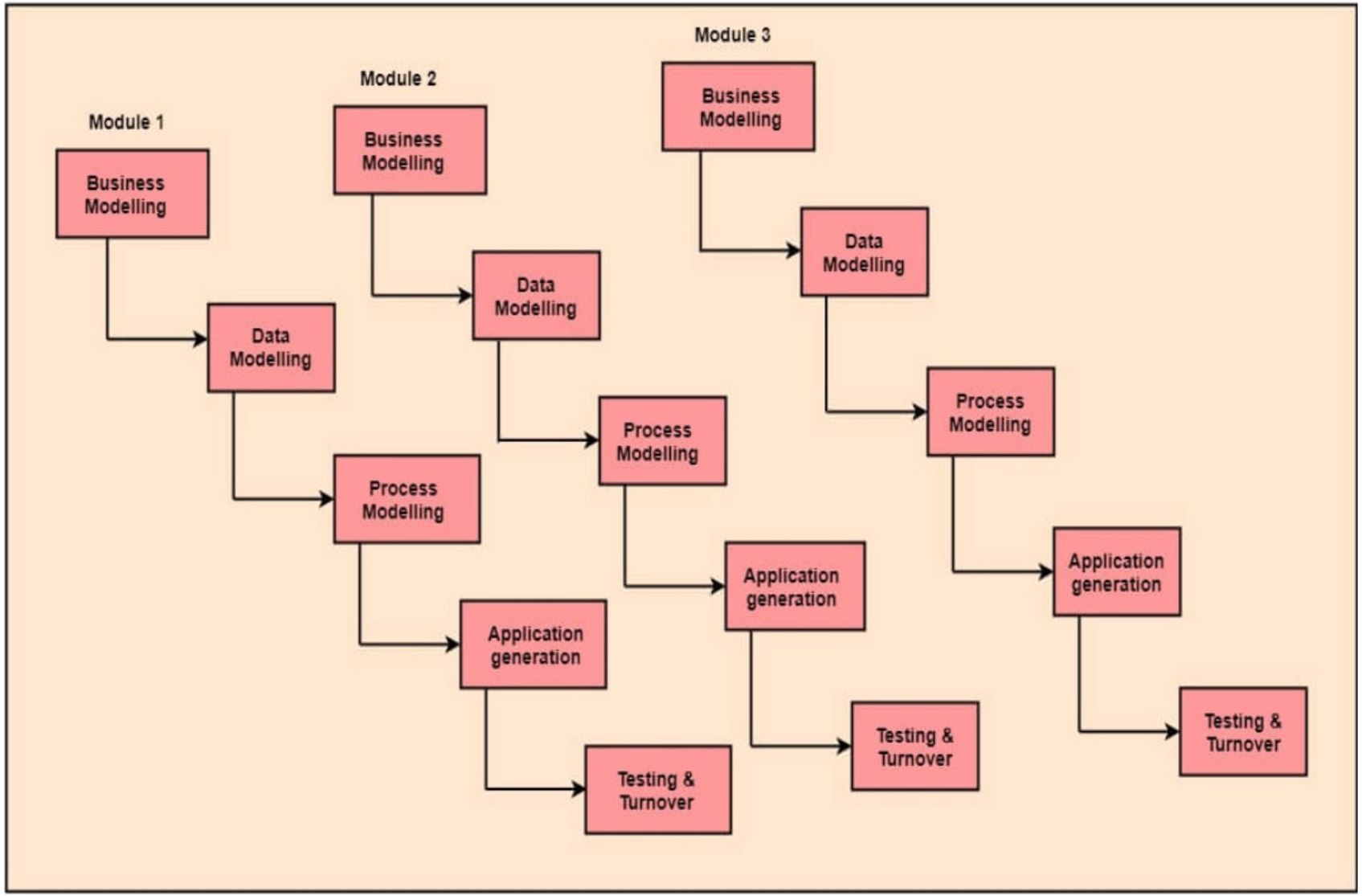
Spiral Model can be used in the below scenario:

- When costs and risk evaluation is important
- For medium to high-risk projects
- Long-term project commitment unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex
- Significant changes are expected (research and exploration)

RAD Model

- RAD or Rapid Application Development process is an adoption of the waterfall model;
- it targets at developing software in a short span of time.
- RAD follow the **iterative approach**. In RAD model the components or functions are developed in parallel as if they were mini projects.
- The developments are time boxed, delivered and then assembled into a working prototype.
- This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.
- Development of each module involves the various basic steps as in waterfall model i.e. analyzing, designing, coding and then testing.

RAD Model (Contd.)



RAD Model (Contd.)

1. **Business Modelling:**
   - The information flow among business functions is defined by answering questions like what data drives the business process, what data is generated, who generates it, where does the information go, who process it and so on.
2. **Data Modelling:**
   - The data collected from business modeling is refined into a set of data objects (entities) that are needed to support the business. The attributes (character of each entity) are identified, and the relation between these data objects (entities) is defined.

RAD Model (Contd.)

3. **Process Modelling:**
   - The information object defined in the data modeling phase are transformed to achieve the data flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.
4. **Application Generation:**
   - Automated tools are used to facilitate construction of the software; even they use the 4th GL techniques.
5. **Testing & Turnover:**
   - Many of the programming components have already been tested since RAD emphasis reuse. This reduces the overall testing time. But the new part must be tested, and all interfaces must be fully exercised.

RAD Model (Contd.)

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.

RAD Model (Contd.)

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.
- It is not meant for small scale projects as for such cases, the cost of using automated tools and techniques may exceed the entire budget of the project.

RAD Model (Contd.)

Applications –

- This model should be used for a system with known requirements and requiring short development time.
- It is also suitable for projects where requirements can be modularized and reusable components are also available for development.
- The model can also be used when already existing system components can be used in developing a new system with minimum changes.
- 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.
- The model should be chosen when the budget permits the use of automated tools and techniques required.

Agile Development

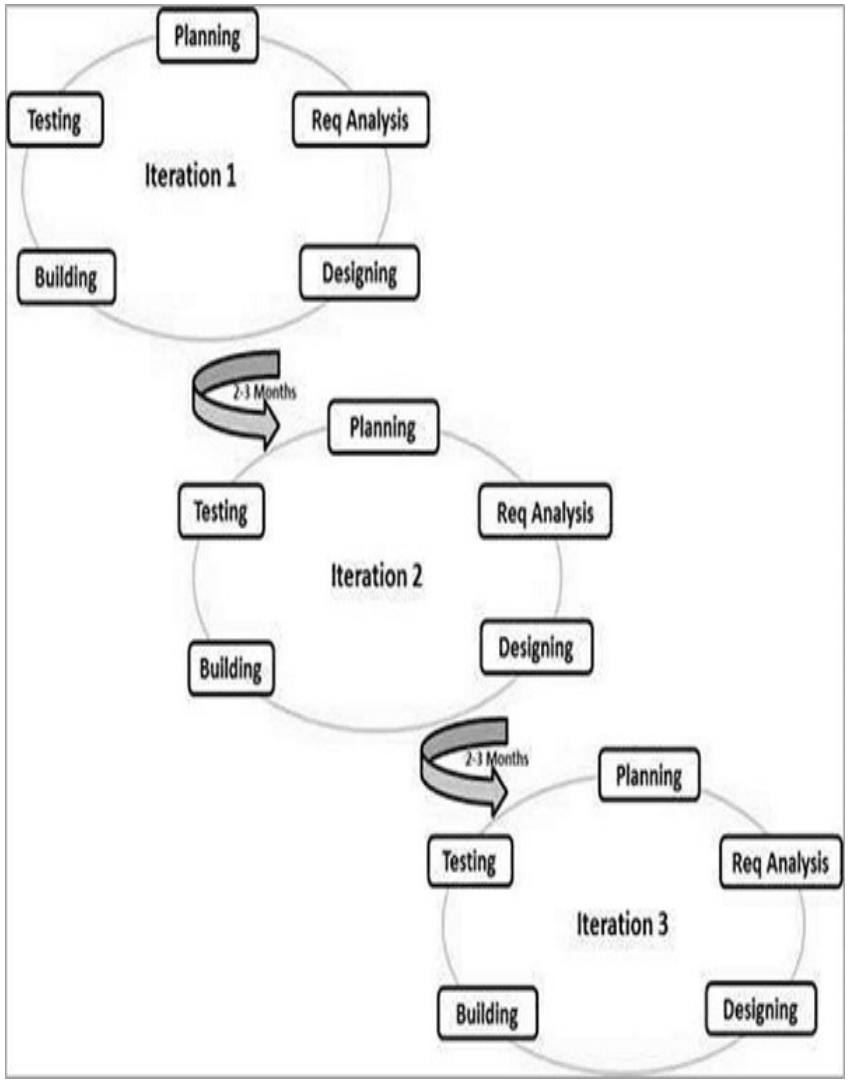
- combines a philosophy and a set of development guidelines to develop the software project as customer need.
- The philosophy encourages customer satisfaction and early incremental delivery of software; small, highly motivated project teams; informal methods; minimal software engineering work products; and overall development simplicity.
- The development guidelines stress delivery over analysis and design, and active and continuous communication between developers and customers.
- The change management is a primary goal of this philosophy. The changes may be from employ turnover, change of customer need, technology change, stakeholder decision change etc.

Agile Development (Contd.)

- Software engineers and other project stakeholders (managers, customers, end users) work together on an agile team—a team that is self-organizing and in control of its own destiny.
- An agile team fosters communication and collaboration among all who serve on it.
- The basic framework activities—communication, planning, modeling, construction, and deployment— remains the same for agile development but they transform into a minimal task set that pushes the project team toward construction and delivery.

Agile Development (Contd.)

- a process for managing a project characterized by constant iteration and collaboration in order to more fully answer a customer's needs.



Agility Principles

The Agile Alliance defines 12 agility principles for those who want to achieve agility:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Agility Principles (Contd.)

7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Not every agile process model applies these 12 principles with equal weight, and some models choose to ignore (or at least downplay) the importance of one or more of the principles. However, the principles define an *agile spirit* that is maintained in each of the process models.

Agile Process

Agile is a process by which a team can manage a project by breaking it up into several stages and involving constant collaboration with stakeholders and continuous improvement and iteration at every stage. The Agile methodology begins with clients describing how the end product will be used and what problem it will solve. This clarifies the customer's expectations to the project team. Once the work begins, team cycle through a process of planning, executing, and evaluating — which might just change the final deliverable to fit the customer's needs better. Continuous collaboration is key, both among team members and with project stakeholders, to make fully-informed decisions.

The agile software development emphasizes on four core values.

1. Individual and team interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

Agile Models

1. Extreme Programming (XP)
2. Scrum
3. Dynamic System Development Method
4. Feature Driven Development
5. Lean Software Development
6. Adaptive Software Development

1. Extreme Programming

- Extreme programming (XP) is one of the most important software development framework of Agile models.
- It is used to improve software quality and responsive to customer requirements.
- XP is a flexible and pragmatic Agile methodology that encourages continuous improvement and responsiveness to changing requirements. It has been successfully used in a wide range of software development projects

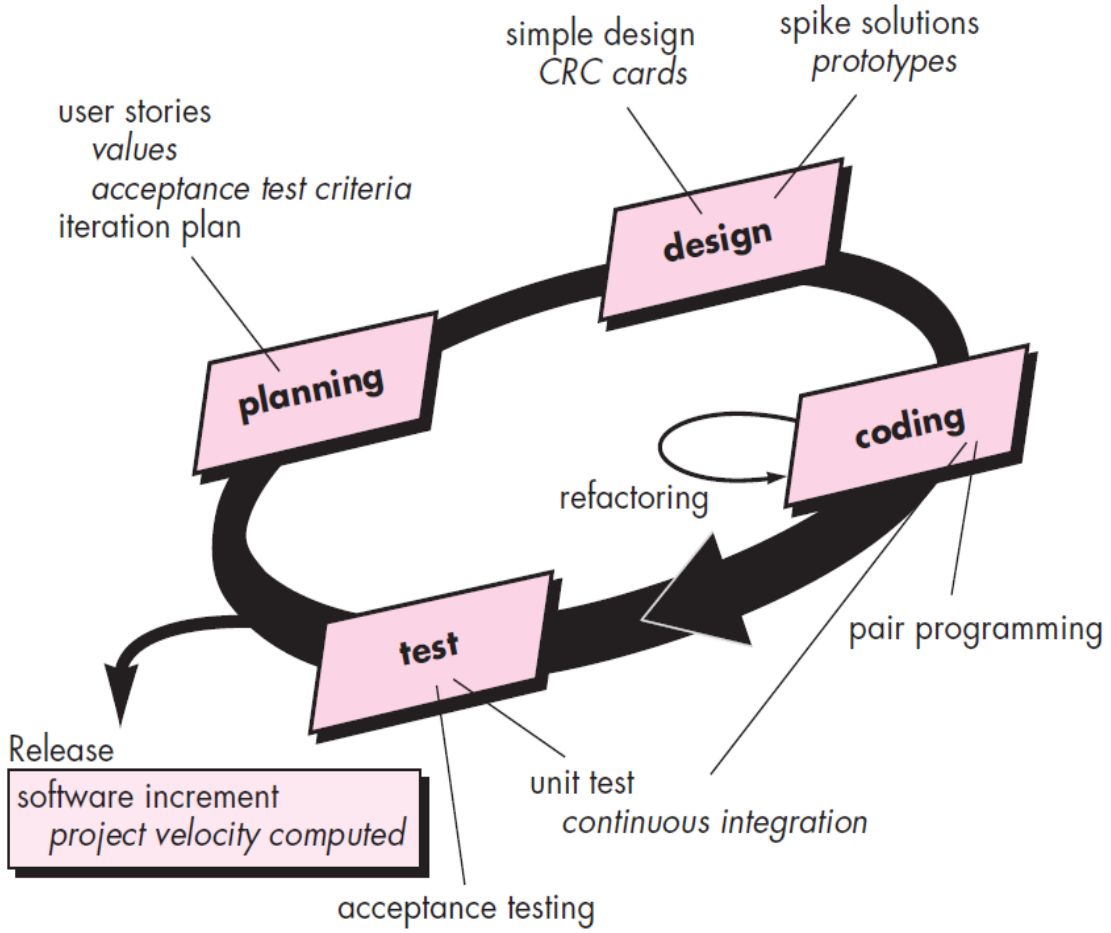


Fig. The Extreme Programming process

1. Extreme Programming (Contd.)

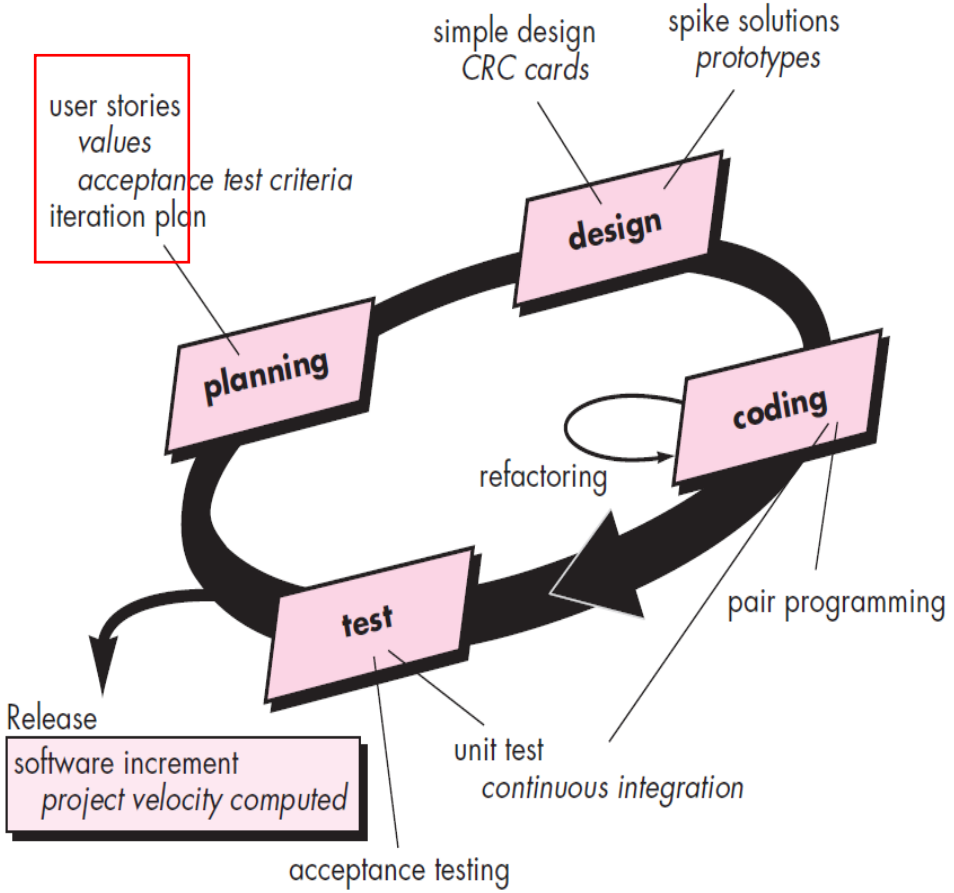


Fig. The Extreme Programming process

- XP is based on the frequent iteration through which the developers implement User Stories.
- User stories are simple and informal statements of the customer about the functionalities needed.
- A User story is a conventional description by the user about a feature of the required system.
- It does not mention finer details such as the different scenarios that can occur.

1. Extreme Programming (Contd.)

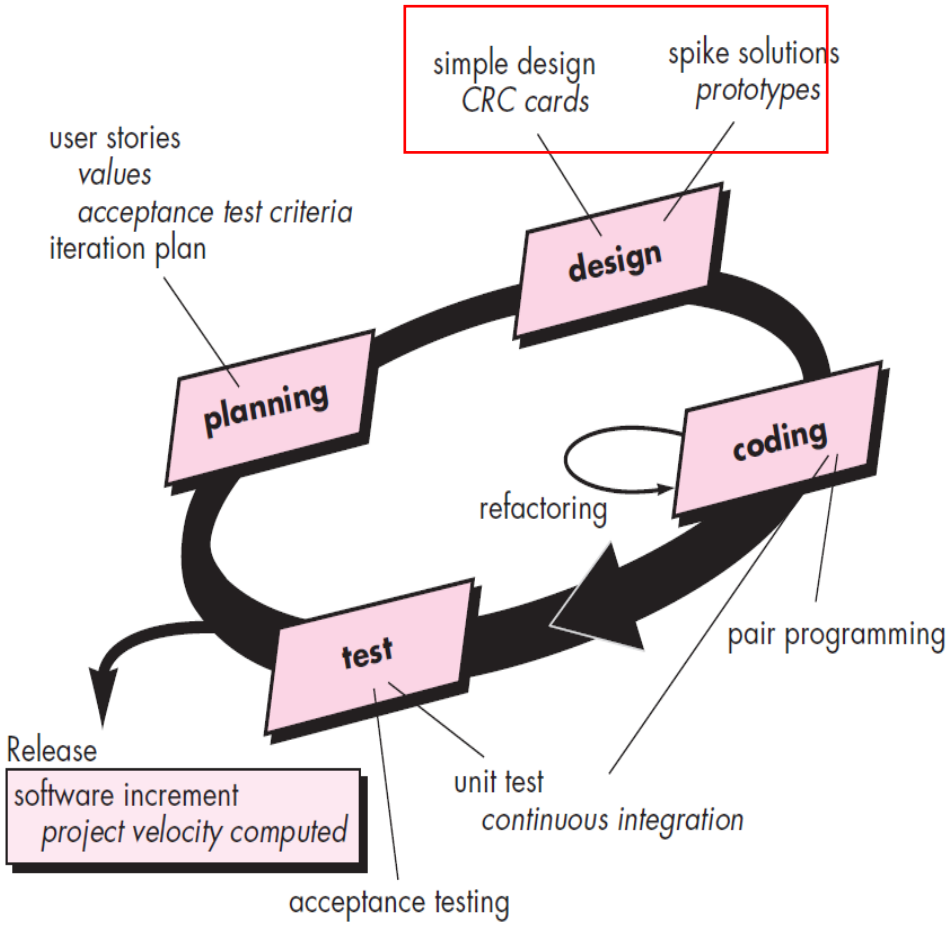


Fig. The Extreme Programming process

- On the basis of User stories, the project team proposes Metaphors.
- Metaphors are a common vision of how the system would work.
- The development team may decide to build a Spike for some feature.
- A Spike is a very simple program that is constructed to explore the suitability of a solution being proposed.
- It can be considered similar to a prototype.

1. Extreme Programming (Contd.)

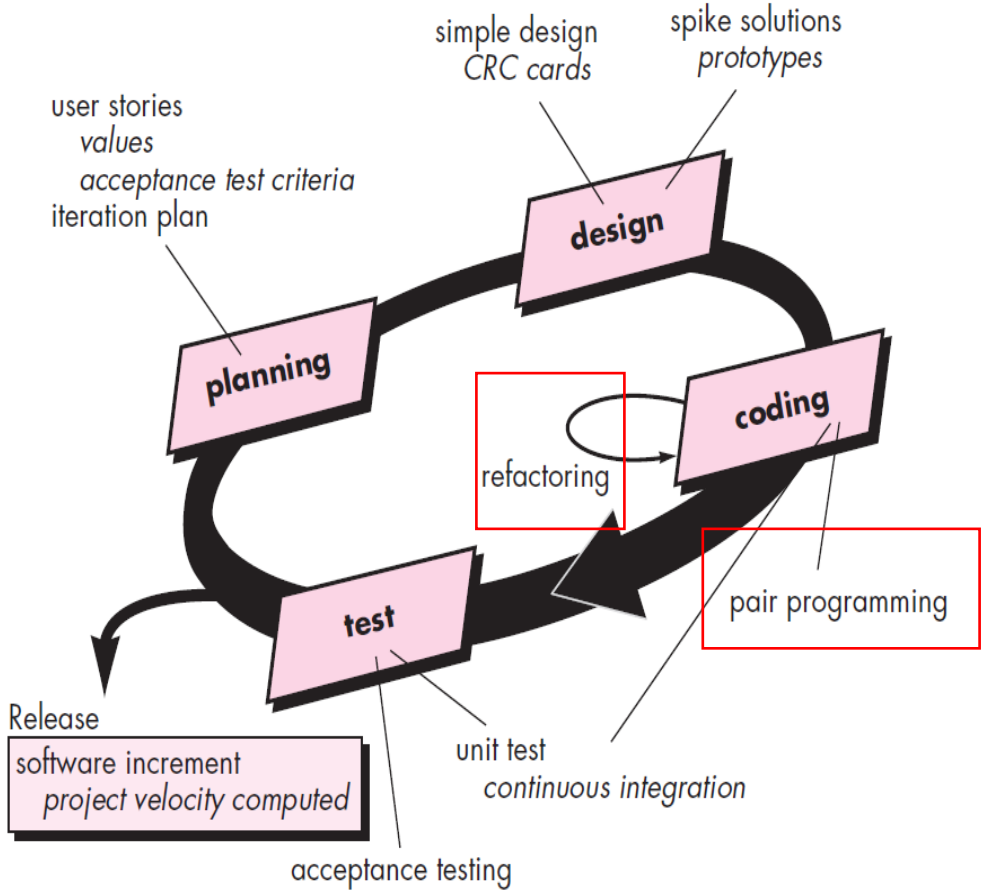


Fig. The Extreme Programming process

- A key concept of "Pair Programming" is done during the coding activity. Code review detects and corrects errors efficiently.
- XP recommends two people work together at one computer workstation to provide better solution for the problem.
- It suggests pair programming as coding and reviewing of written code carried out by a pair of programmers who switch their works between them every hour.
- Refactoring is always considered. Refactoring is the technique of improving code without changing functionality.

1. Extreme Programming (Contd.)

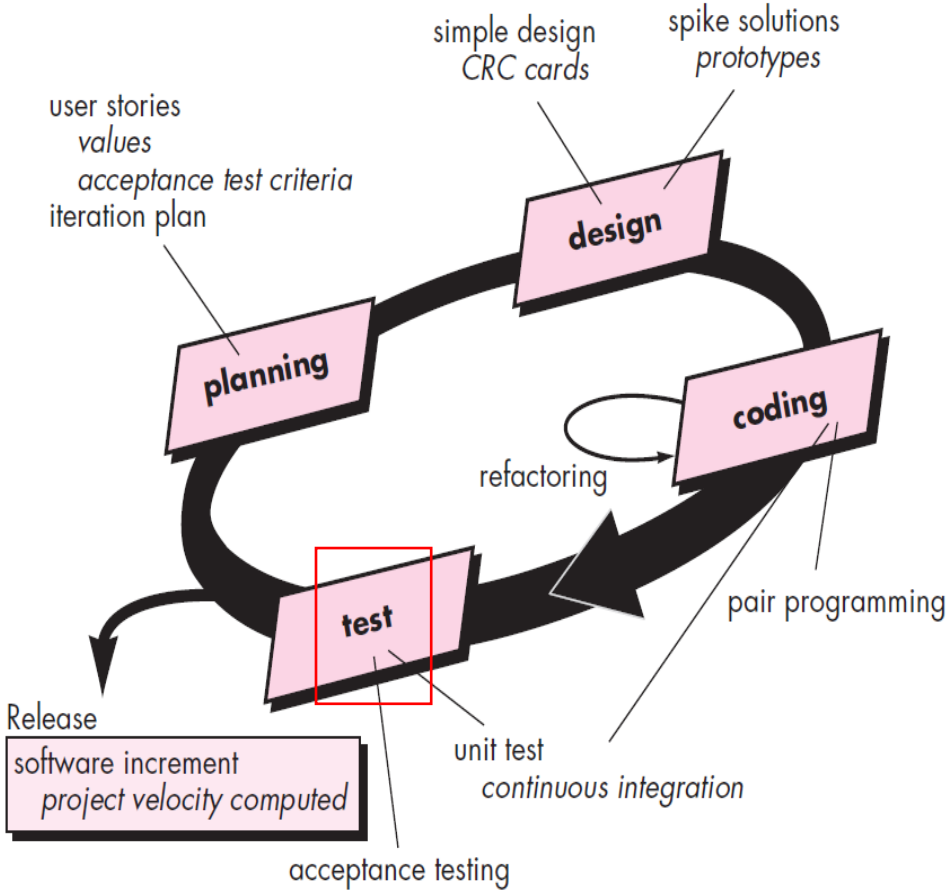


Fig. The Extreme Programming process

- XP model gives high importance on testing and considers it be the primary factor to develop a fault-free software.
- XP suggests test-driven development (TDD) to continually write and execute test cases.
- In the TDD approach test cases are written even before any code is written.

1. Extreme Programming (Contd.)

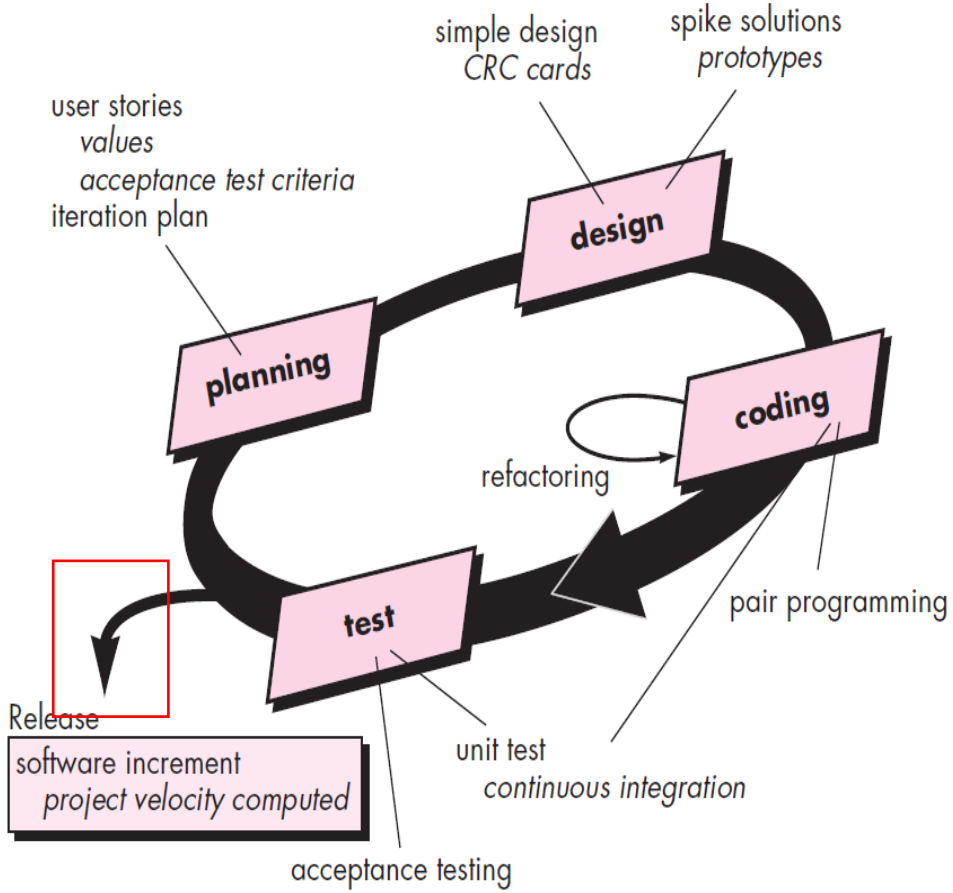


Fig. The Extreme Programming process

- Incremental development is very good because customer feedback is gained
- and based on this development team come up with new increments every few days after each iteration.

2. Adaptive Software Development

- Adaptive Software Development (ASD) is a direct outgrowth of an earlier agile framework, Rapid Application Development (RAD).
- It aims to enable teams to quickly and effectively adapt to changing requirements or market needs by evolving their products with lightweight planning and continuous learning.
- The ASD approach encourages teams to develop according to a three-phase process:

  speculate, collaborate, learn.

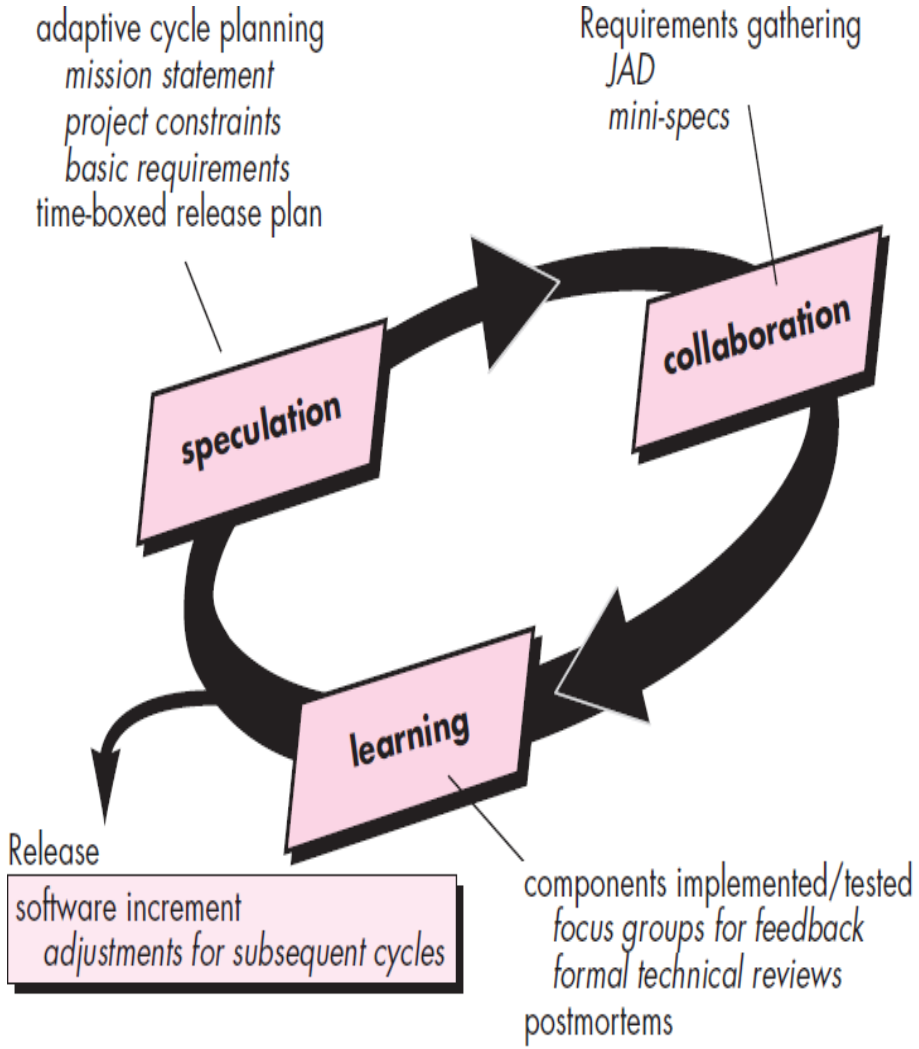


Fig. Adaptive software development

3. Scrum

Scrum is the type of **Agile framework**. It is a framework within which people can address complex adaptive problem while productivity and creativity of delivering product is at highest possible values. Scrum uses **Iterative process**.

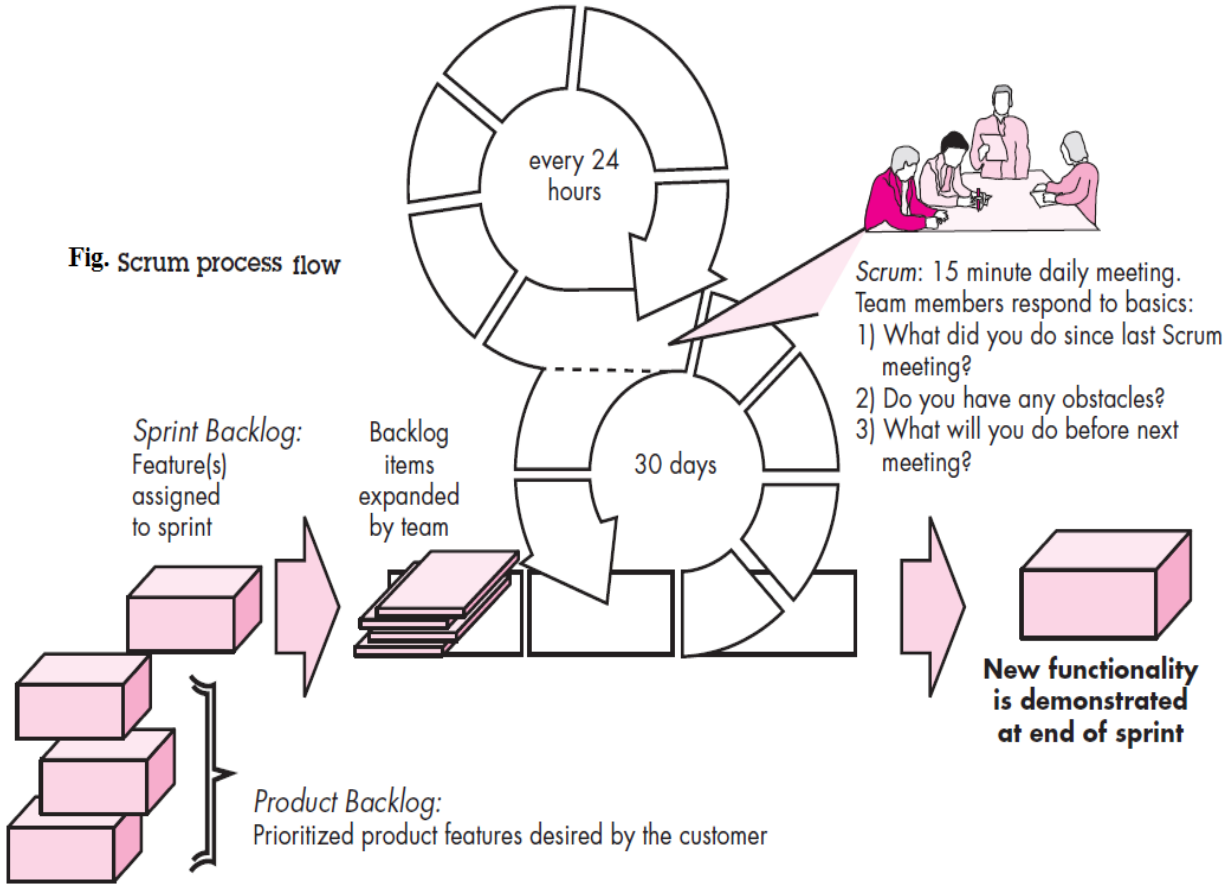


Fig. Scrum process flow

3. Scrum (Contd.)

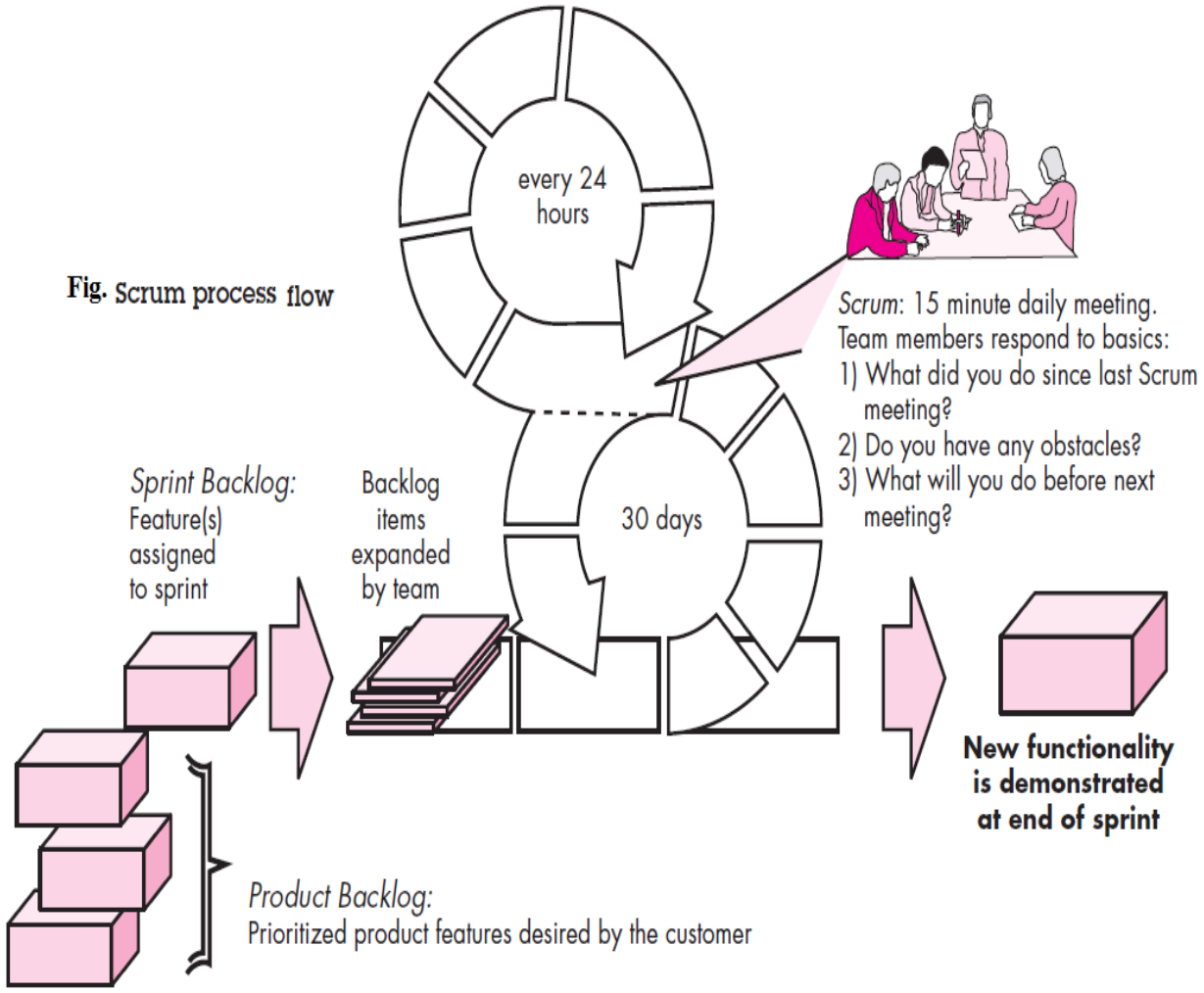


Fig. Scrum process flow

Product Backlog

- It is the master list of work that needs to get done maintained by the product owner or product manager.
- Items can be added to the backlog at any time.
- The product backlog is constantly revisited, re-prioritized and maintained by the Product Owner.

3. Scrum (Contd.)

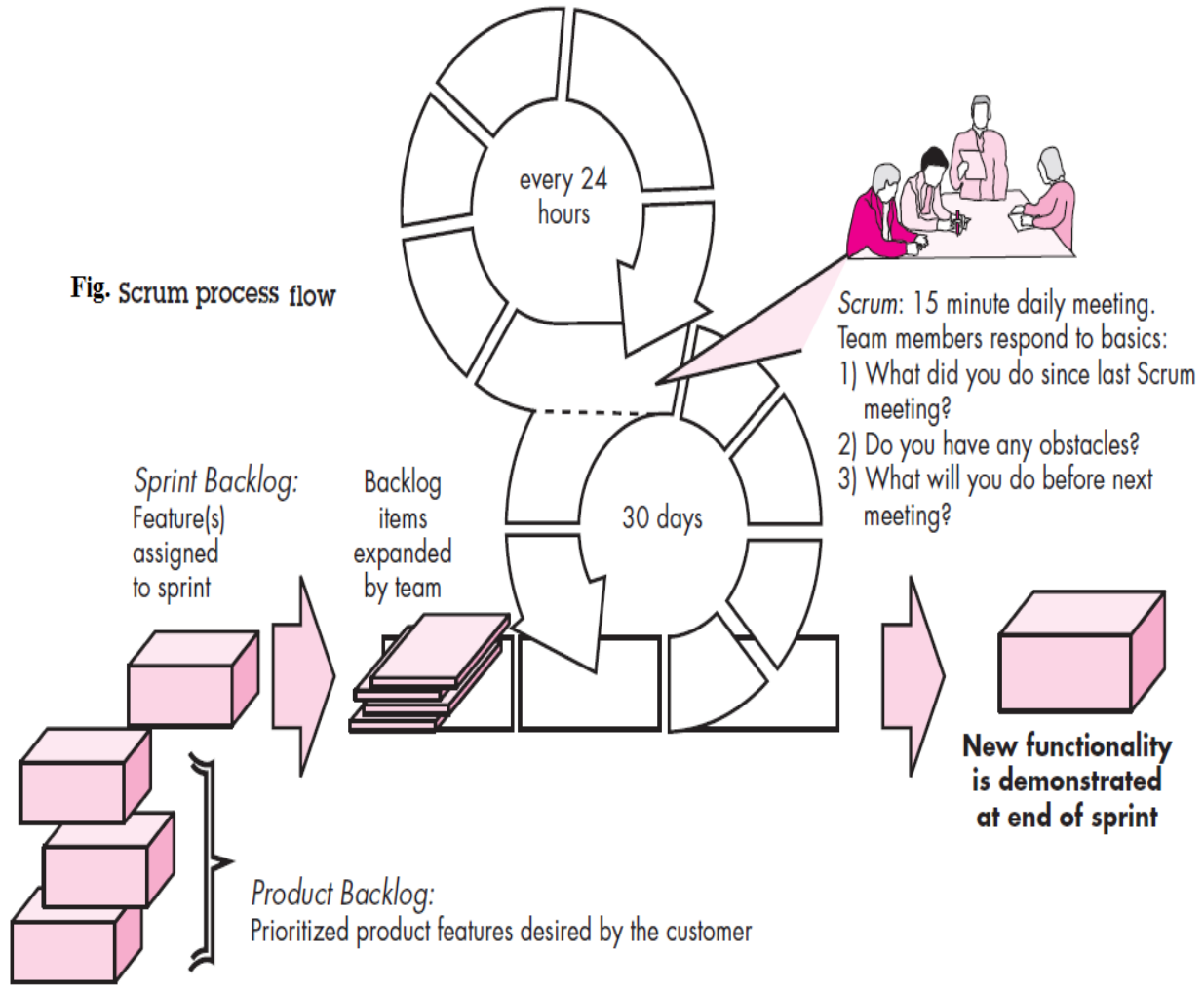


Fig. Scrum process flow

Sprint Backlog:

- It is the list of items, user stories, or bug fixes, selected by the development team for implementation in the current sprint cycle.
- Before each sprint, in the sprint planning meeting, the team chooses which items it will work on for the sprint from the product backlog.

3. Scrum (Contd.)

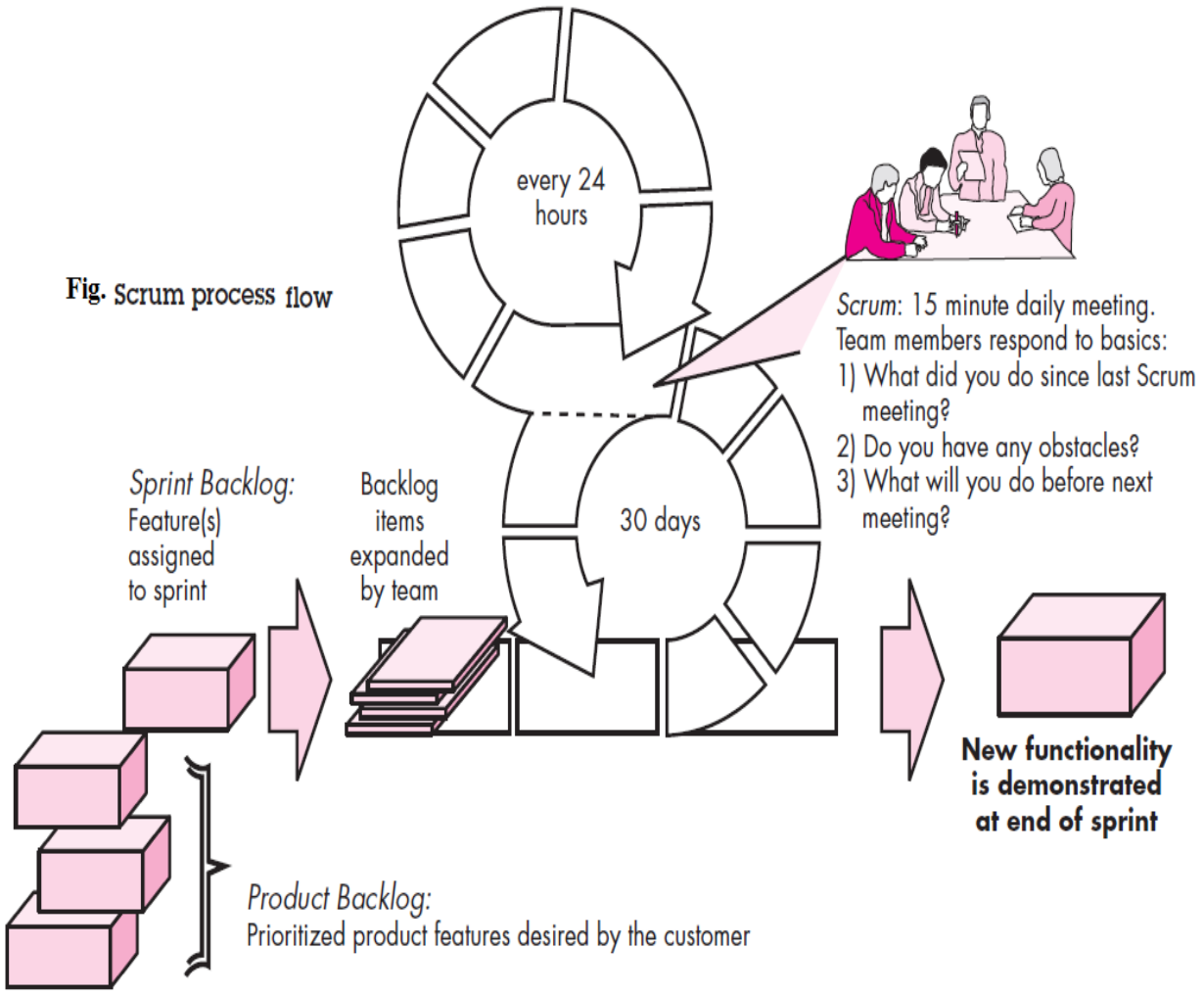


Fig. Scrum process flow

Scrum Meetings:

- These are short (typically 15 minutes) meetings held daily by the Scrum team.
- A team leader, called a Scrum master, leads the meeting and assesses the responses from each person.
- The Scrum meeting helps the team to uncover potential problems as early as possible.
- Also, these daily meetings lead to "knowledge socialization" and thereby promote a self-organizing team structure.

3. Scrum (Contd.)

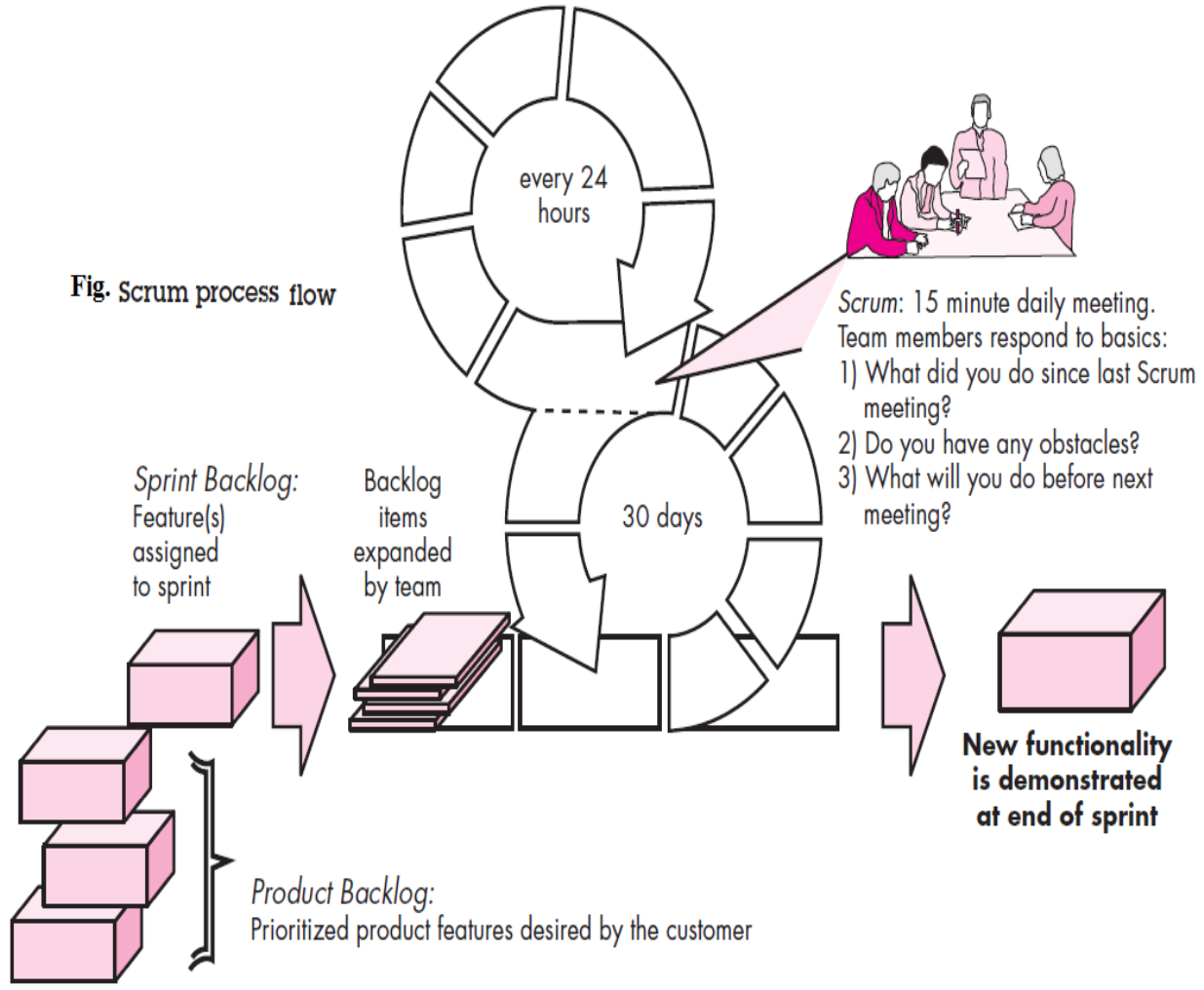


Fig. Scrum process flow

Demos:

- Demos deliver the software increment to the customer so that functionality that has been implemented can be demonstrated and evaluated by the customer.

Four P's of Software Project Management

- The effective software project management Focuses on four P's.
  - The People
  - The Product
  - The Process
  - The Project
- Project manager has to control all these P’s to have a smooth flow in the project progress and to reach the goal.

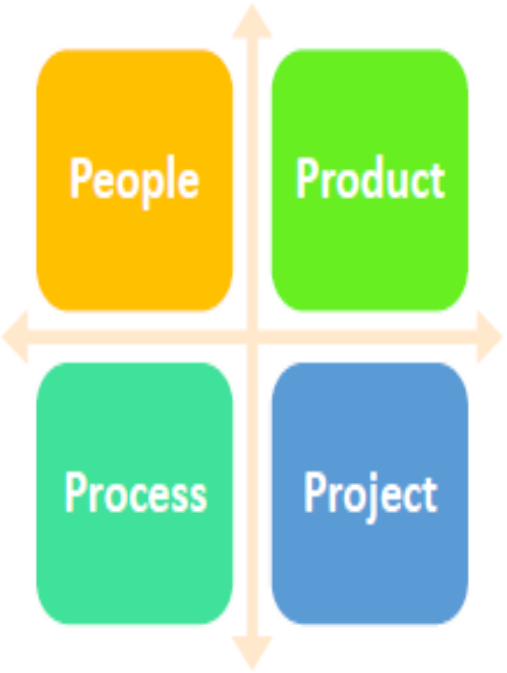
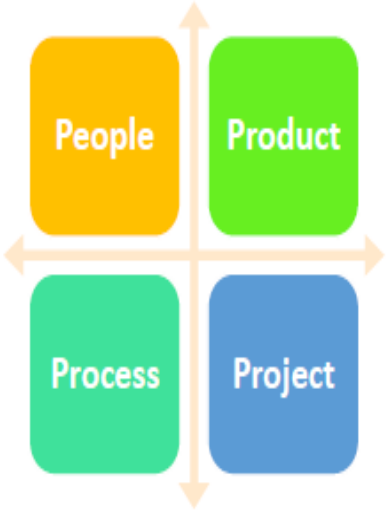
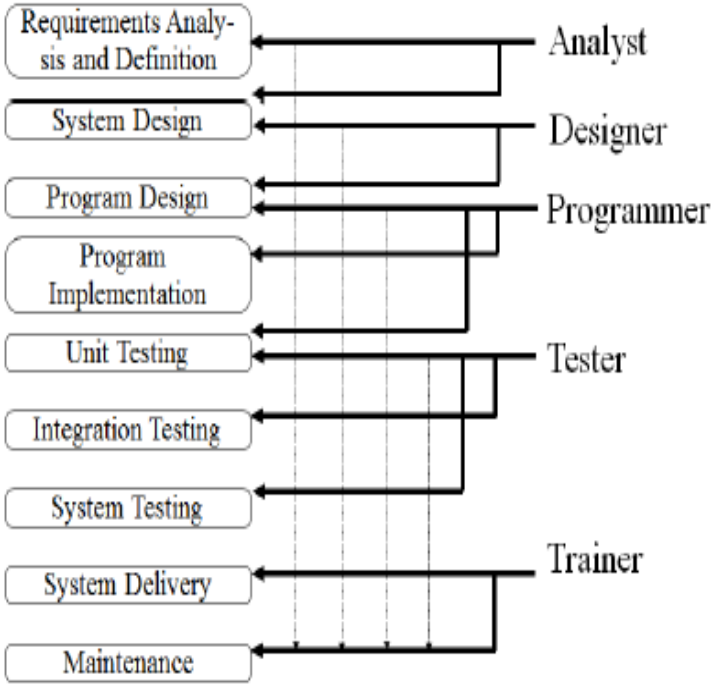


Figure: The 4 P’s

The People:



- are the primary key factor for successful organization.
- For the successful software production environment, any organization must perform the proper staffing, communication and coordination, work environment, performance management, training, compensation (or reward), competency analysis and development, career development, workgroup development, team/culture development, and others.
- The people involve: The Senior managers, Project managers, Analysts, Designers, Software developers, testers and customers



The Product:

- Product is any software that has to be developed.
- To develop successfully, product objectives and scope should be established, alternative solutions should be considered, and technical and management constraints should be identified.
- Without this information, it is impossible to define reasonable and accurate estimates of the cost, an effective assessment of risk, a realistic breakdown of project tasks or a manageable project schedule that provides a meaningful indication of progress.

The Process:

- It is the set of framework activities and software engineering tasks to get the job done.
- The project manager is responsible for deciding what process to follow for doing the project and may use appropriate SDLC model.
- A software process provides the framework from which a comprehensive plan for software development can be established.
- A number of different tasks sets— tasks, milestones, work products, and quality assurance points—enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team.
- Finally, umbrella activities overlay the software process model.
- Umbrella activities are independent of any one framework activity and occur throughout the process.

The Project:

- The project is the complete software project that includes requirement analysis, development, delivery, maintenance and updates.
- The project manager of a project or sub-project is responsible for managing the people, product and process.
- The responsibilities or activities of software project manager would be a long list but that has to be followed to avoid project failure.

Thank You