SOFTWARE PROJECT MANAGEMENT

Module-2: PROJECT LIFE CYCLE AND EFFORT ESTIMATION

CSE4016

PRIYANKA SINGH

Software Process

- A software process 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.

- Any software process must include the following four activities:
    - ***Software specification (or requirements engineering):*** Define the main functionalities of the software and the constrains around them.
    - ***Software design and implementation:*** The software is to be designed and programmed.
    - ***Software verification and validation:*** The software must conforms to it's specification and meets the customer needs.
    - ***Software evolution (software maintenance):*** The software is being modified to meet customer and market requirements changes.

- In practice, they include sub-activities such as requirements validation, architectural design, unit testing, …etc.
- There are also supporting activities such as configuration and change management, quality assurance, project management, user experience.
- Along with other activities aim to improve the above activities by introducing new techniques, tools, following the best practice, process standardization (so the diversity of software processes is reduced), etc.

- Software process also includes the process description, which includes:
  - Products: The outcomes of the an activity. For example, the outcome of architectural design maybe a model for the software architecture.
  - Roles: The responsibilities of the people involved in the process. For example, the project manager, programmer, etc.
  - Pre and post conditions: The conditions that must be true before and after an activity. For example, the pre condition of the architectural design is the requirements have been approved by the customer, while the post condition is the diagrams describing the architectural have been reviewed.

- Software process is complex, it relies on making decisions. There's no ideal process and most organizations have developed their own software process.
- For example, an organization works on critical systems has a very structured process, while with business systems, with rapidly changing requirements, a less formal, flexible process is likely to be more effective.

Choice of Process models

- The word “process’ is sometimes used to emphasize the idea of a system in action. In order to achieve an outcome, the system will have to execute one or more activities: this is its process.
- This idea can be applied to the development of computer-based systems where a number of interrelated activities have to be undertaken to create a final product. These activities can be organized in different ways and we can call these process models.
- A major part of the planning method will be the choosing of the development methods to be used and the slotting of these into an overall process model.
- The planner needs not only to select methods but also to specify how the method is to be applied.

Software Process Models

- A software process model is a simplified representation of a software process. Each model represents a process from a specific perspective.
- Definition- A (software/system) process model is a description of the sequence of activities carried out in an SE (software engineering) project, and the relative order of these activities.

- It provides a fixed generic framework that can be tailored to a specific project.
- Project specific parameters will include:
    - Size, (person-years)
    - Budget,
    - Duration.

project plan = process model + project parameters

- There are hundreds of different process models to choose from, e.g:
    - waterfall,
    - code-and-fix
    - spiral
    - rapid prototyping
    - unified process (UP)
    - agile methods, extreme programming (XP)
    - COTS …
- But most are minor variations on a small number of basic models.

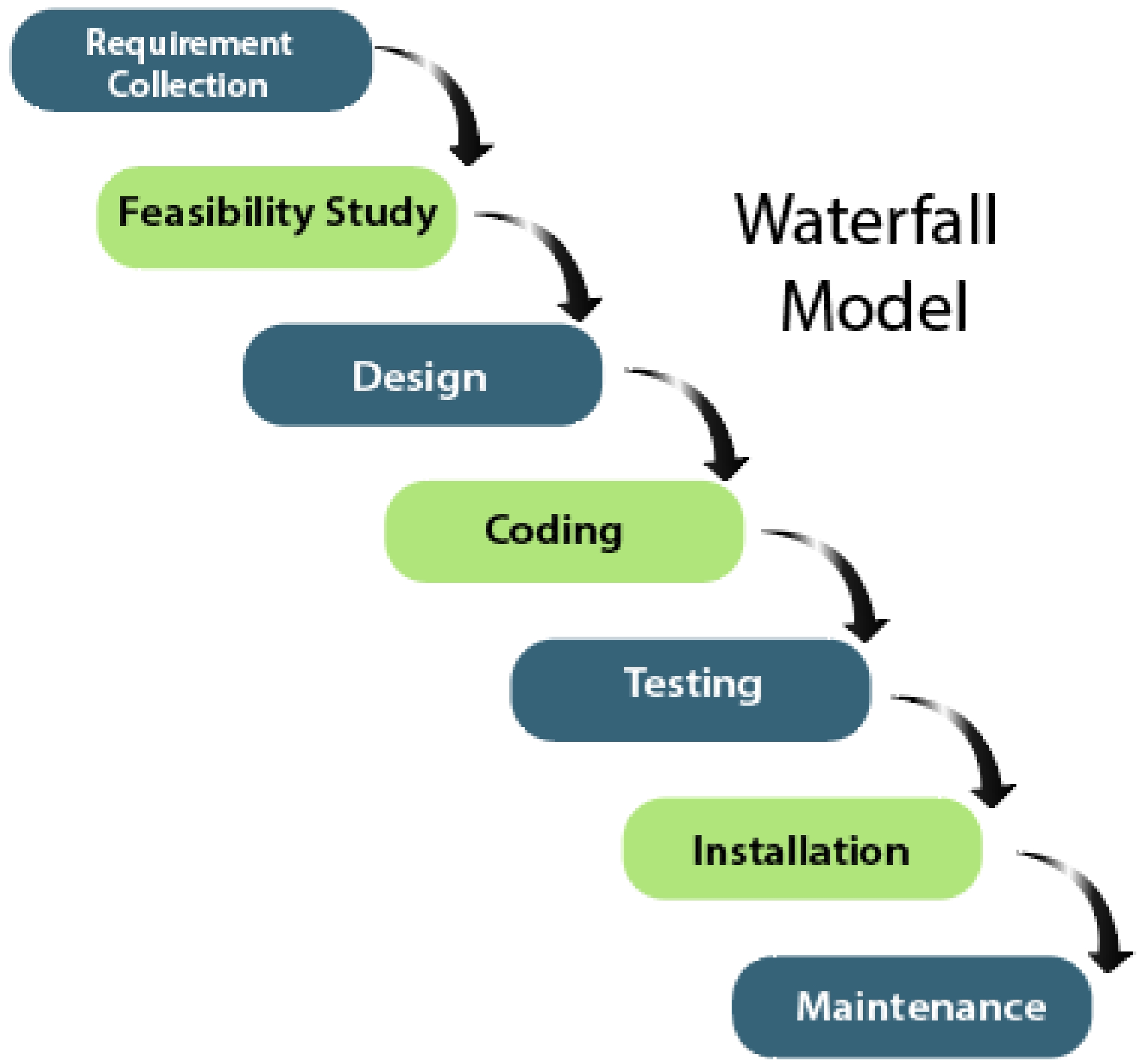
By changing the process model, we can improve and/or tradeoff:

- Development speed (time to market)
- Product quality
- Project visibility
- Administrative overhead
- Risk exposure
- Customer relations, etc, etc.

Normally, a process model covers the entire lifetime of a product. From birth of a commercial idea to final de-installation of last release.

Waterfall Model

- The waterfall model is the classic process model – it is widely known, understood and used.
- In some respect, waterfall is the ”common sense” approach.
- The waterfall model is a sequential approach, where each fundamental activity of a process represented as a separate phase, arranged in linear order.
- In waterfall model, you must plan and schedule all of the activities before starting working on them (plan-driven process).


Requirement Collection
Feasibility Study
Design
Coding
Testing
Installation
Maintenance
Waterfall Model

Advantages

1. Easy to understand and implement.
2. Widely used and known (in theory!)
3. Fits other engineering process models: civil, mech etc.
4. Reinforces good habits: define-before- design, design-before- code
8. Identifies deliverables and milestones
9. Document driven: ***People leave, documents don’t***

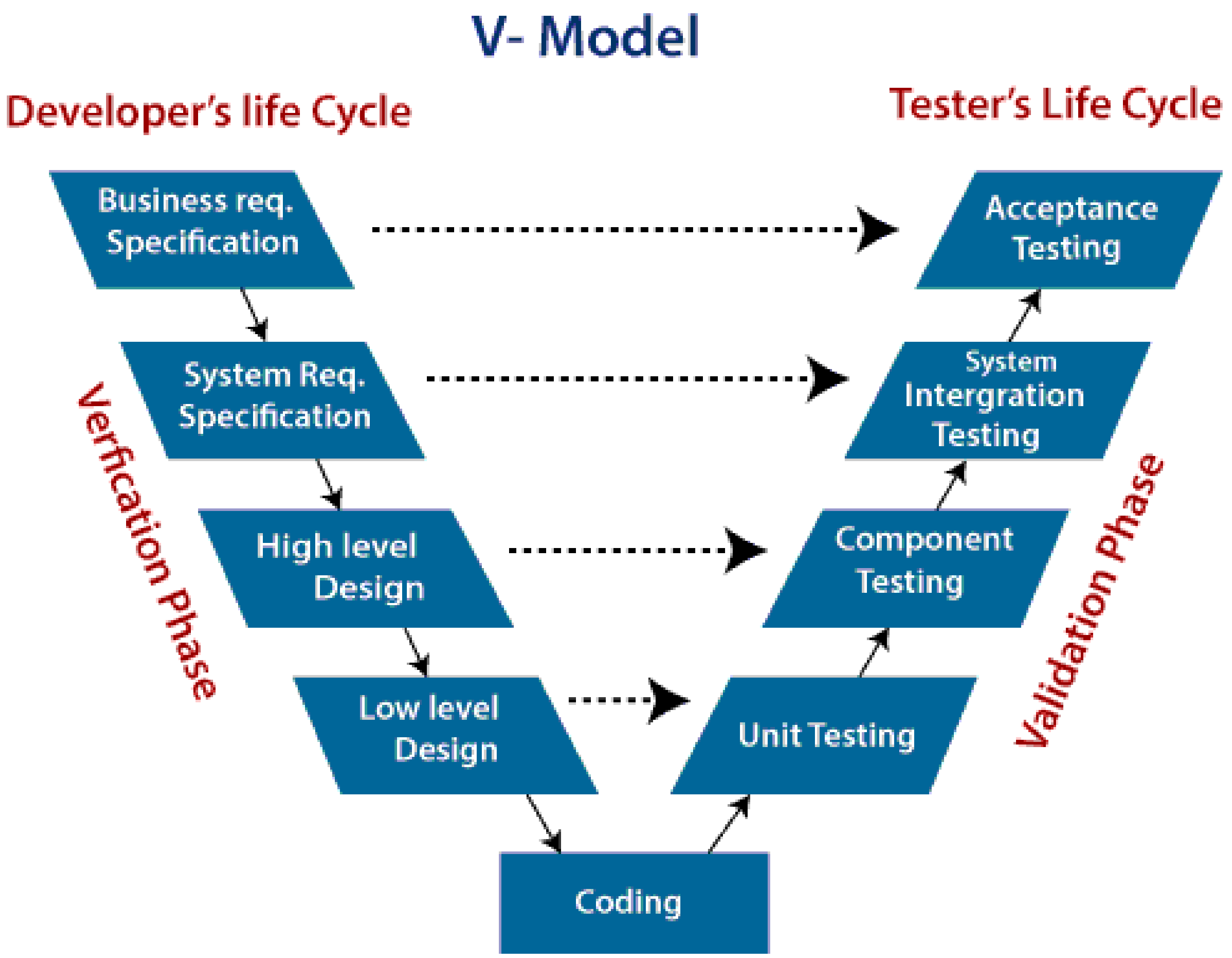
   Published documentation standards: URD, SRD, … etc. , e.g. ESA PSS-05.
10. Works well on large/mature products and weak teams.

Disadvantages

1. Doesn’t reflect iterative nature of exploratory development.
2. Sometimes unrealistic to expect accurate requirements early in a project
3. Software is delivered late, delays discovery of serious errors.
4. No inherent risk management
5. Difficult and expensive to change decisions, ”swimming upstream”.
6. Significant administrative overhead, costly for small teams and projects.

The V-process model

- This is an elaboration of the waterfall model and stresses the necessity for validation activities that match the activities that create the products of the project.
- The V-model is a type of SDLC model where process executes in a sequential manner in V-shape. It is also known as Verification and Validation model.
- It is based on the association of a testing phase for each corresponding development stage.
- Development of each step directly associated with the testing phase. The next phase starts only after completion of the previous phase i.e. for each development activity, there is a testing activity corresponding to it.


V- Model
Developer's life Cycle
Tester's Life Cycle
Business req. Specification
System Req. Specification
High level Design
Low level Design
Coding
Unit Testing
Component Testing
System Intergration Testing
Acceptance Testing
Verfication Phase
Validation Phase

- ***Verification***: It involves static analysis technique (review) done without executing code. It is the process of evaluation of the product development phase to find whether specified requirements meet.
- ***Validation***: It involves dynamic analysis technique (functional, non-functional), testing done by executing code. Validation is the process to evaluate the software after the completion of the development phase to determine whether software meets the customer expectations and requirements.
- So V-Model contains Verification phases on one side of the Validation phases on the other side. Verification and Validation phases are joined by coding phase in V-shape. Thus it is called V-Model.

Spiral Model

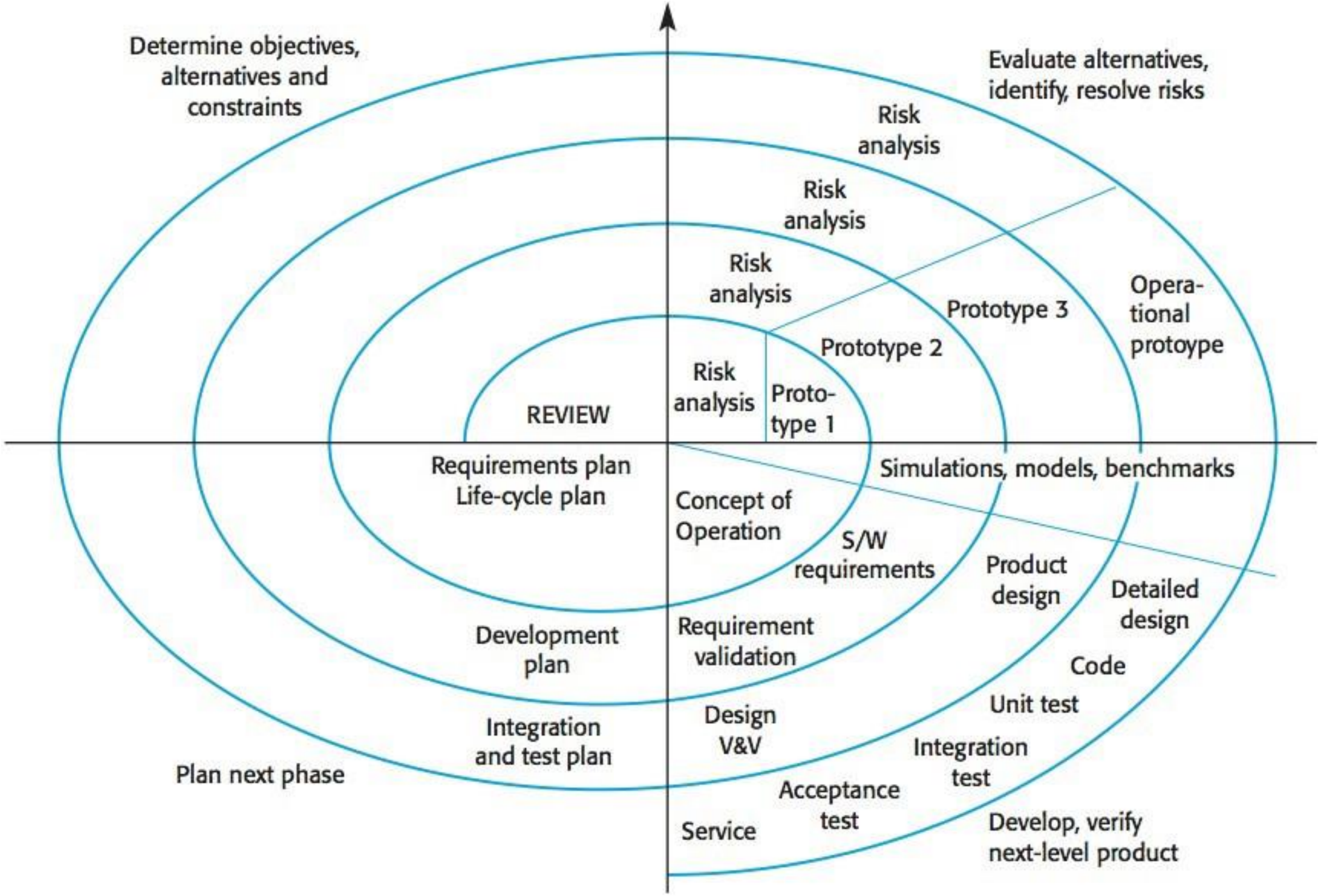
- Extends waterfall model by adding iteration to explore /manage risk
- Project risk is a moving target. Natural to progress a project cyclically in four step phases

1. Consider alternative scenarios, constraints
2. Identify and resolve risks
3. Execute the phase
4. Plan next phase: e.g. user req, software req, architecture

… then goto 1

Key idea: on each iteration identify and solve

the sub-problems with the highest risk


Determine objectives, alternatives and constraints
Evaluate alternatives, identify, resolve risks
Risk analysis
Risk analysis
Risk analysis
Risk analysis
Operational protoype
Prototype 3
Prototype 2
Proto-type 1
REVIEW
Requirements plan Life-cycle plan
Simulations, models, benchmarks
Concept of Operation
S/W requirements
Product design
Detailed design
Code
Unit test
Integration test
Acceptance test
Service
Development plan
Requirement validation
Design V&V
Integration and test plan
Plan next phase
Develop, verify next-level product

Advantages

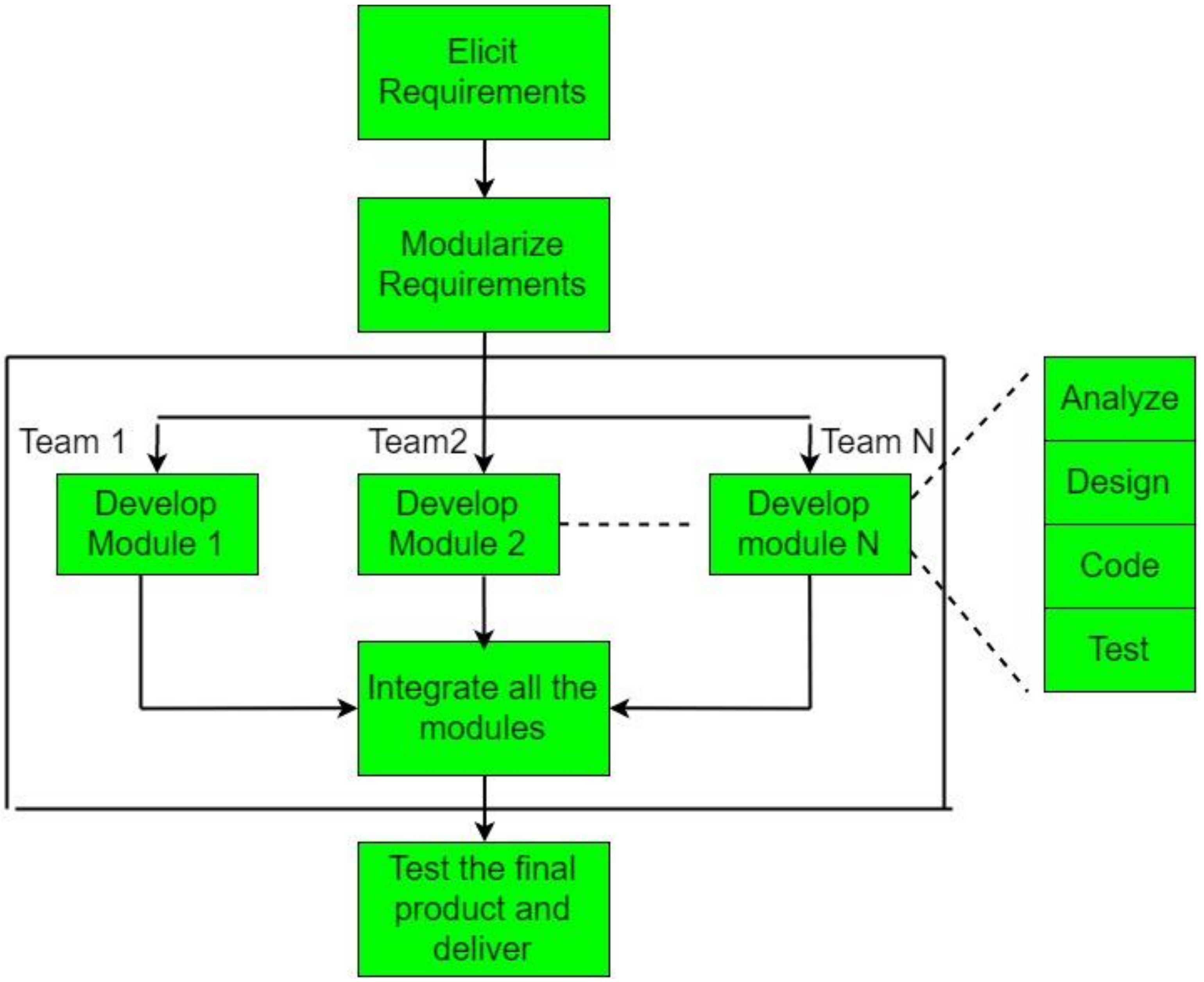
- ***Realism***: the model accurately reflects the iterative nature of software development on projects with unclear requirements
- ***Flexible***: incoporates the advantages of the waterfall and evolutionary methods
- ***Comprehensive model decreases risk***
- ***Good project visibility***.

Disadvantages

1. Needs technical expertise in risk analysis and risk management to work well.
2. Model is poorly understood by nontechnical management, hence not so widely used
3. Complicated model, needs competent professional management. High administrative overhead.

Rapid Application Development (RAD)

- The Rapid Application Development Model was first proposed by IBM in 1980's. The critical feature of this model is the use of powerful development tools and techniques.
- A software project can be implemented using this model if the project can be broken down into small modules wherein each module can be assigned independently to separate teams. These modules can finally be combined to form the final product.
- Development of each module involves the various basic steps as in waterfall model i.e analyzing, designing, coding and then testing, etc. as shown in the figure.
- Another striking feature of this model is a short time span i.e the time frame for delivery(time-box) is generally 60-90 days.


Elicit Requirements
Modularize Requirements
Team 1
Team2
Team N
Develop Module 1
Develop Module 2
Develop module N
Analyze
Design
Code
Test
Integrate all the modules
Test the final product and deliver

Advantages –

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

Disadvantages –

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

Agile methods

- Agile software development refers to software development methodologies centered round the idea of iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams.
- The ultimate value in Agile development is that it enables teams to deliver value faster, with greater quality and predictablity, and greater aptitude to respond to change. Scrum and Kanban are two of the most widely used Agile methodologies.
- Originally created for software development, it was established as a response to the inadequacies of the Waterfall method, the processes of which did not meet the demands of the highly competitive and constant movement of the software industry.

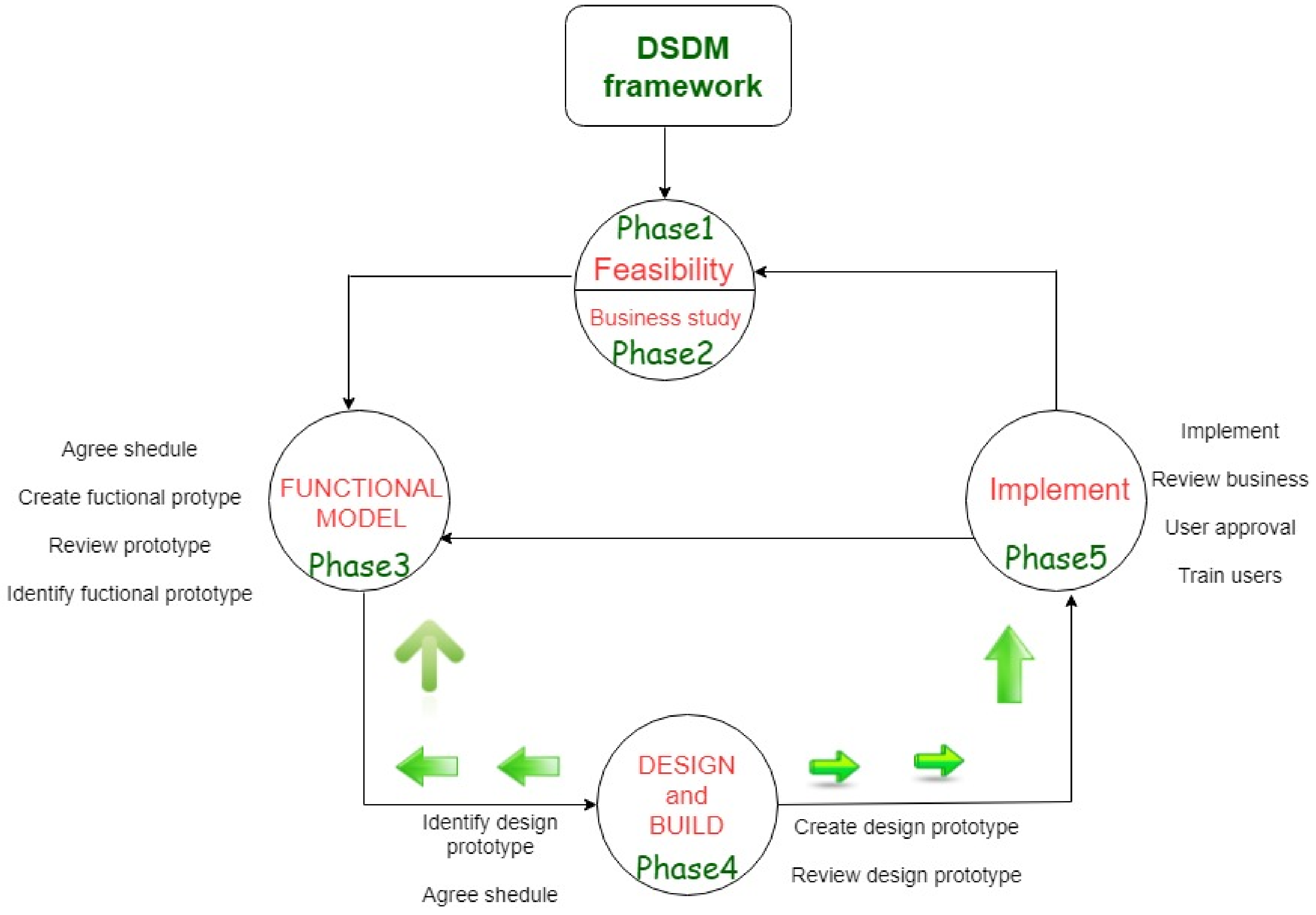
- In Agile development, Design and Implementation are considered to be the central activities in the software process.
- Design and Implementation phase also incorporate other activities such as requirements elicitation and testing into it.
- In an agile approach, iteration occurs across activities. Therefore, the requirements and the design are developed together, rather than separately.
- The allocation of requirements and the design planning and development as executed in a series of increments.
- An agile process focuses more on code development rather than documentation.

Principles:

1. Customer satisfaction through early and continuous software delivery
2. Accommodate changing requirements throughout the development process
3. Frequent delivery of working software
4. Collaboration between the business stakeholders and developers throughout the project
5. Support, trust, and motivate the people involved
6. Enable face-to-face interactions
7. Working software is the primary measure of progress
8. Agile processes to support a consistent development pace
9. Attention to technical detail and design enhances agility
10. Simplicity
11. Self-organizing teams encourage great architectures, requirements, and designs
12. Regular reflections on how to become more effective`

Dynamic System Development Method

- The Dynamic Systems Development technique (DSDM) is an associate degree agile code development approach that provides a framework for building and maintaining systems.
- DSDM is an iterative code method within which every iteration follows the 80% rule that simply enough work is needed for every increment to facilitate movement to the following increment. The remaining detail is often completed later once a lot of business necessities are noted or changes are requested and accommodated.



Dynamic Systems Development Method life cycle

Principles:

1. Focus on the business need
2. Deliver on time
3. Collaborate
4. Never compromise quality
5. Build incrementally from firm foundations
6. Develop iteratively
7. Communicate continuously and clearly
8. Demonstrate control

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.
- The extreme programming model recommends taking the best practices that have worked well in the past in program development projects to extreme levels.

Good practices needs to practiced extreme programming:

- ***Code Review:*** Code review detects and corrects errors efficiently. 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.
- ***Testing***: Testing code helps to remove errors and improves its reliability. XP suggests test-driven development (TDD) to continually write and execute test cases.
- ***Incremental development***: 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.
- ***Simplicity***: Simplicity makes it easier to develop good quality code as well as to test and debug it.
- ***Design***: Good quality design is important to develop a good quality software. So, everybody should design daily.
- ***Integration testing***: It helps to identify bugs at the interfaces of different functionalities.

Basic principles of Extreme programming:

- 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.
- 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.

Some of the basic activities that are followed during software development by using XP model are given below:

- **Coding**: Here, coding activity includes drawing diagrams (modeling) that will be transformed into code, scripting a web-based system and choosing among several alternative solutions.
- **Testing**: XP model gives high importance on testing and considers it be the primary factor to develop a fault-free software.
- **Listening**: The developers needs to carefully listen to the customers if they have to develop a good quality software.
- **Designing**: Without a proper design, a system implementation becomes too complex and very difficult to understand the solution, thus it makes maintenance expensive. A good design results elimination of complex dependencies within a system.
- **Feedback**: One of the most important aspects of the XP model is to gain feedback to understand the exact customer needs. Frequent contact with the customer makes the development effective.
- **Simplicity**: The main principle of the XP model is to develop a simple system that will work efficiently in present time, rather than trying to build something that would take time and it may never be used.