SDLC Models

Software Development Life Cycle Models

- A software life cycle model (also called process model) is a descriptive and diagrammatic representation of the software life cycle. A life cycle model represents all the activities required to make a software product transit through its life cycle phases. It also captures the order in which these activities are to be undertaken.
- Different life cycle models may map the basic development activities in different ways.
- Thus, no matter which life cycle model is followed, the basic activities are included in all life cycle models though the activities may be carried out in different orders in different life cycle models

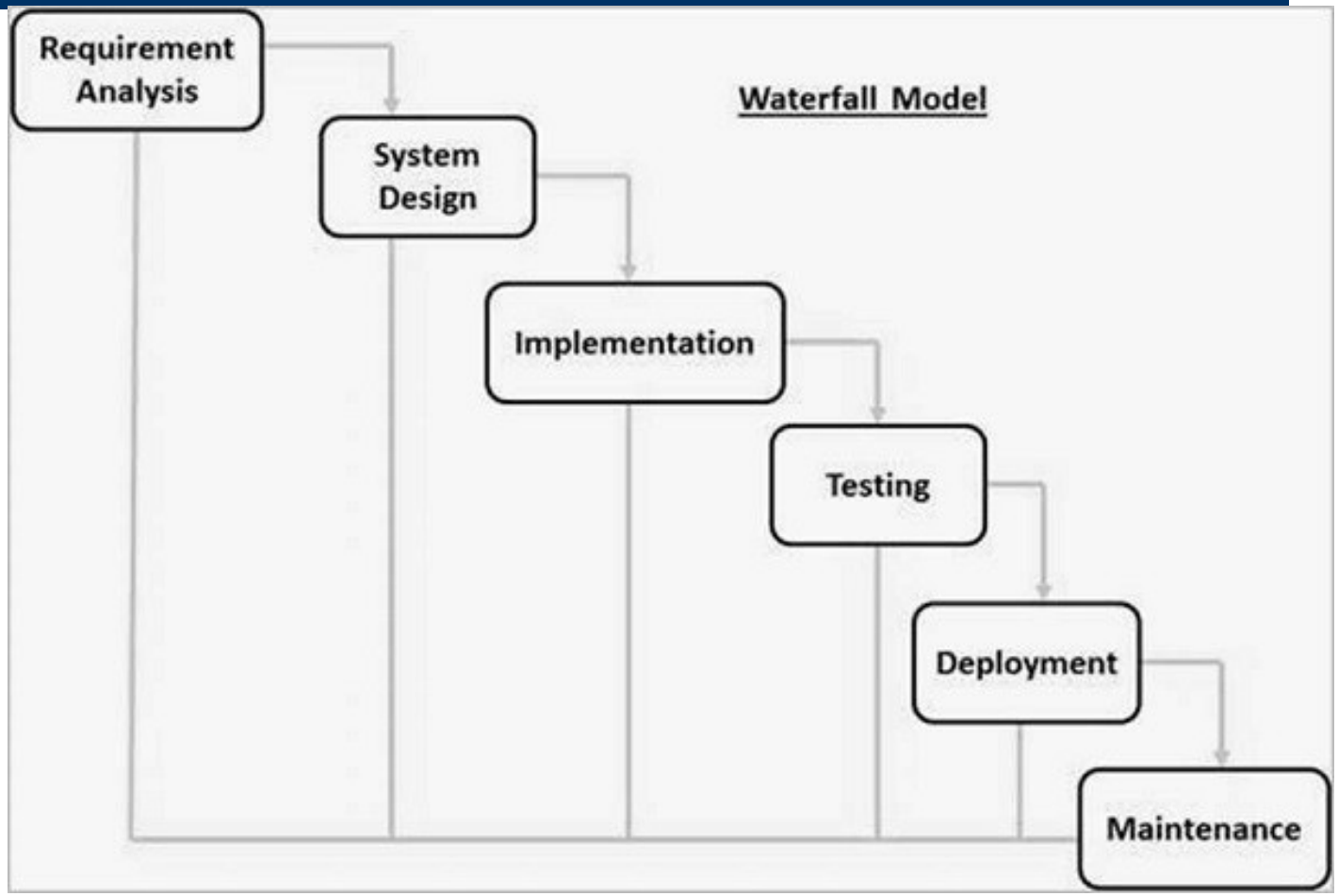
SDLC Models

- Waterfall Model
- The Iterative Model
- Spiral Model
- Agile Model

Waterfall model

- The Waterfall Model was the first **SDLC Model** to be introduced.
- It is also referred to as a **linear-sequential life cycle model**.
- In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.
- The outcome of one phase acts as the input for the next phase sequentially.

Waterfall model



Waterfall Model Phases

- **Requirement Gathering and analysis** – All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** – The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** – With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

Waterfall Model Phases

- **Integration and Testing** – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** – There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Advantage & Disadvantage of Waterfall model

Advantages:

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Advantage & Disadvantage of Waterfall model

Disadvantage:

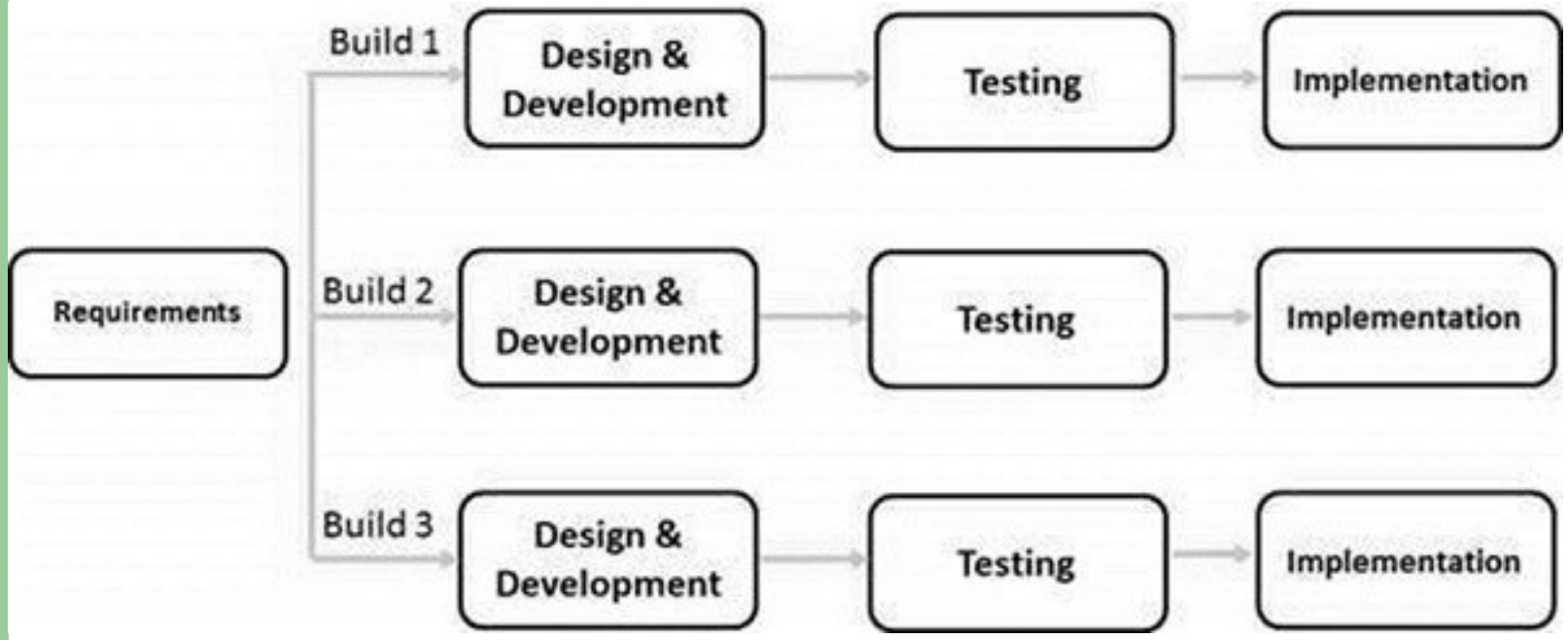
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

The Iterative model

- In the Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.
- Development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements.
- This process is then repeated, producing a new version of the software at the end of each iteration of the model.

The Iterative model

The Iterative model



Iterative Model - Design

- Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented.
- At each iteration, design modifications are made and new functional capabilities are added.
- The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

Advantages of The Iterative model

- Develop high risk or major function first.
- Each release delivers an operational product.
- Customer can respond to each build.
- Uses “divide and Conquer” break down of task.
- Lower initial delivery cost.
- Initial product delivery is faster.
- Customers get important functionality early.
- Risk of changing requirements is reduced.

Disadvantages of
The Iterative model

- More resources may be required.
- Although cost of change is lesser, but it is not very suitable for changing requirements.
- More management attention is required.
- System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.

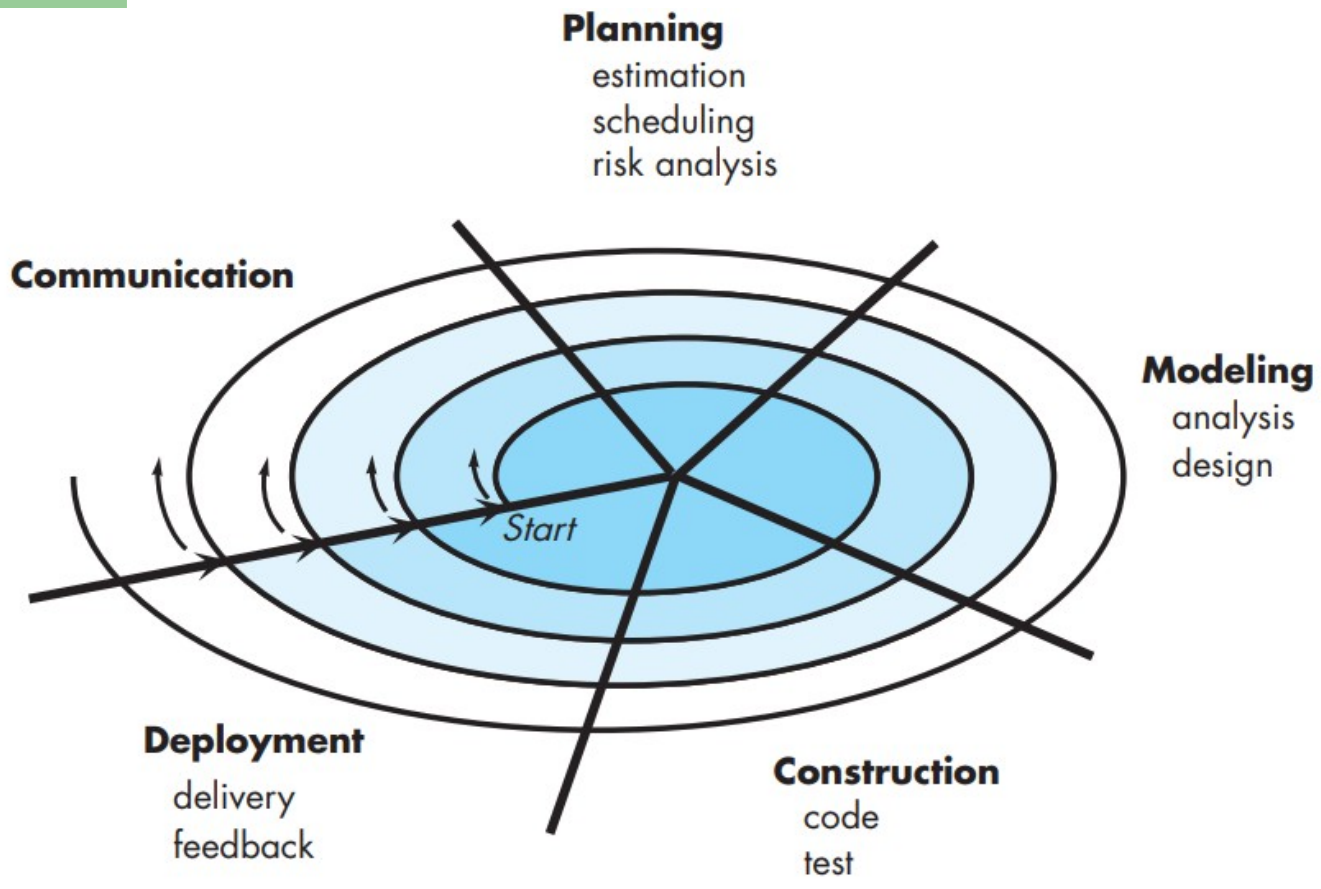
When to use Iterative Model

- Most of the requirements are known up-front but are expected to evolve over time.
- A need to get basic functionality to the market early.
- On projects which have lengthy development schedules.
- On a project with new technology.
- Mostly such model is used in web applications.

Spiral Model

- Spiral model 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**
- Using the spiral model, software is developed in a series of evolutionary releases
- During early iterations, the release might be a model or prototype
- During later iterations, increasingly more complete versions of the engineered system are produced

A typical spiral model



Spiral Model

- As this evolutionary process begins, the software team performs activities that are implied by a circuit around
  the spiral in a clockwise direction, beginning at the center
- **Risk analysis** is considered as each revolution is made
- The first circuit around the spiral might result in the development of a product specification
- Subsequent passes around the spiral might be used to develop a prototype and then progressively more sophisticated versions of the software

Spiral Model

- Each pass through the planning region results in adjustments to the project plan.
- Cost and schedule are adjusted based on feedback derived from the customer after delivery.
- In addition, the project manager adjusts the planned number of iterations required to complete the software
- Unlike other process models that end when software is delivered, the spiral model can be adapted to apply throughout the life of the computer software.

When to Use?

- When the project is large.
- Where the software needs **continuous risk evaluation**.
- Requirements are a bit complicated and require continuous clarification.
- Where enough time frame is there to get end user feedback.
- Where releases are required to be frequent.

Advantages of using Spiral Model:

- Development is fast
- Larger projects / software are created and handled in a strategic way
- Risk evaluation is proper.
- Control towards all the phases of development.
- More and more features are added in a systematic way.
- Software is produced early.
- Has room for customer feedback and the changes are implemented faster.

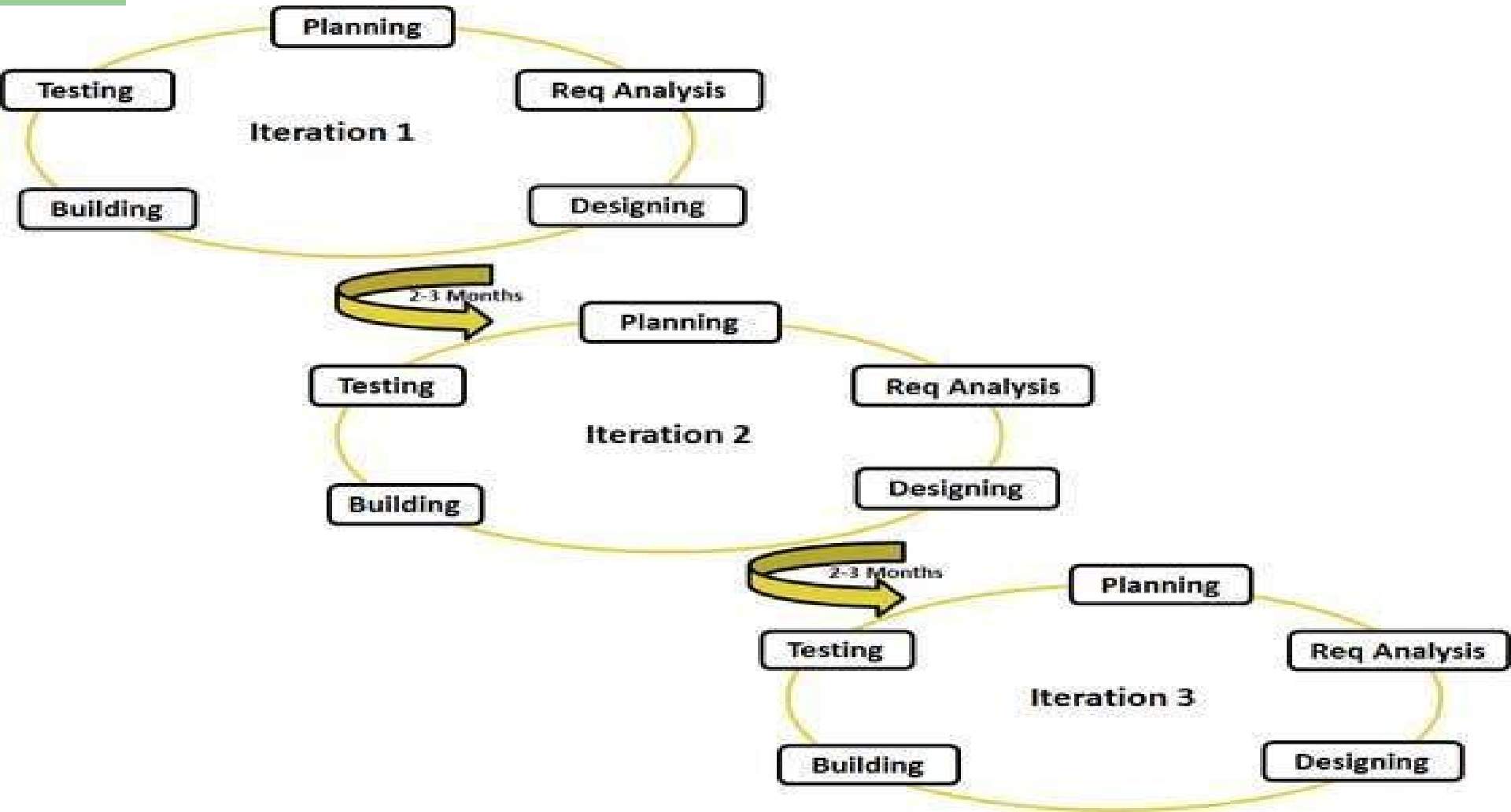
Disadvantages of using Spiral model:

- Risk analysis is important phase so requires expert people.
- Is not beneficial for smaller projects.
- Spiral may go infinitely.
- Documentation is more as it has intermediate phases.
- It is costly for smaller projects.

Agile Model

- Agile SDLC model is a combination of iterative and incremental process models with focus on **process adaptability and customer satisfaction** by rapid delivery of working software product.
- Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing.
- At the end of the iteration a working product is displayed to the customer and important stakeholders.

Agility Diagram



Advantages of Agile model:

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication. .
- Regular adaptation to changing circumstances.

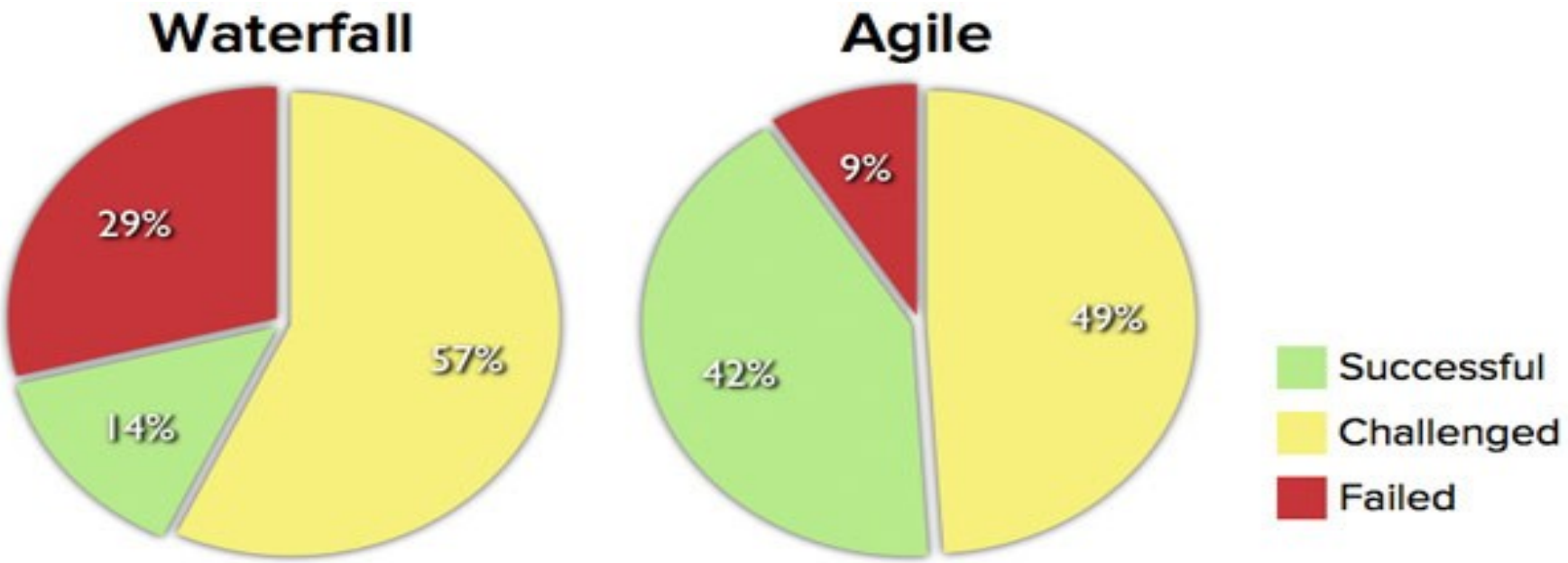
Disadvantages of Agile model:

- In case of some large software deliverables, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers.

When to use Agile model

- When new changes are needed to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
- To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
- Unlike the waterfall model in agile model very limited planning is required to get started with the project. Agile assumes that the end users' needs are ever changing in a dynamic business

Waterfall VS Agile



Source: The CHAOS Manifesto, The Standish Group, 2012.

Waterfall VS Agile

Suppose Google is working on project to come up with a competing product for MS Word:

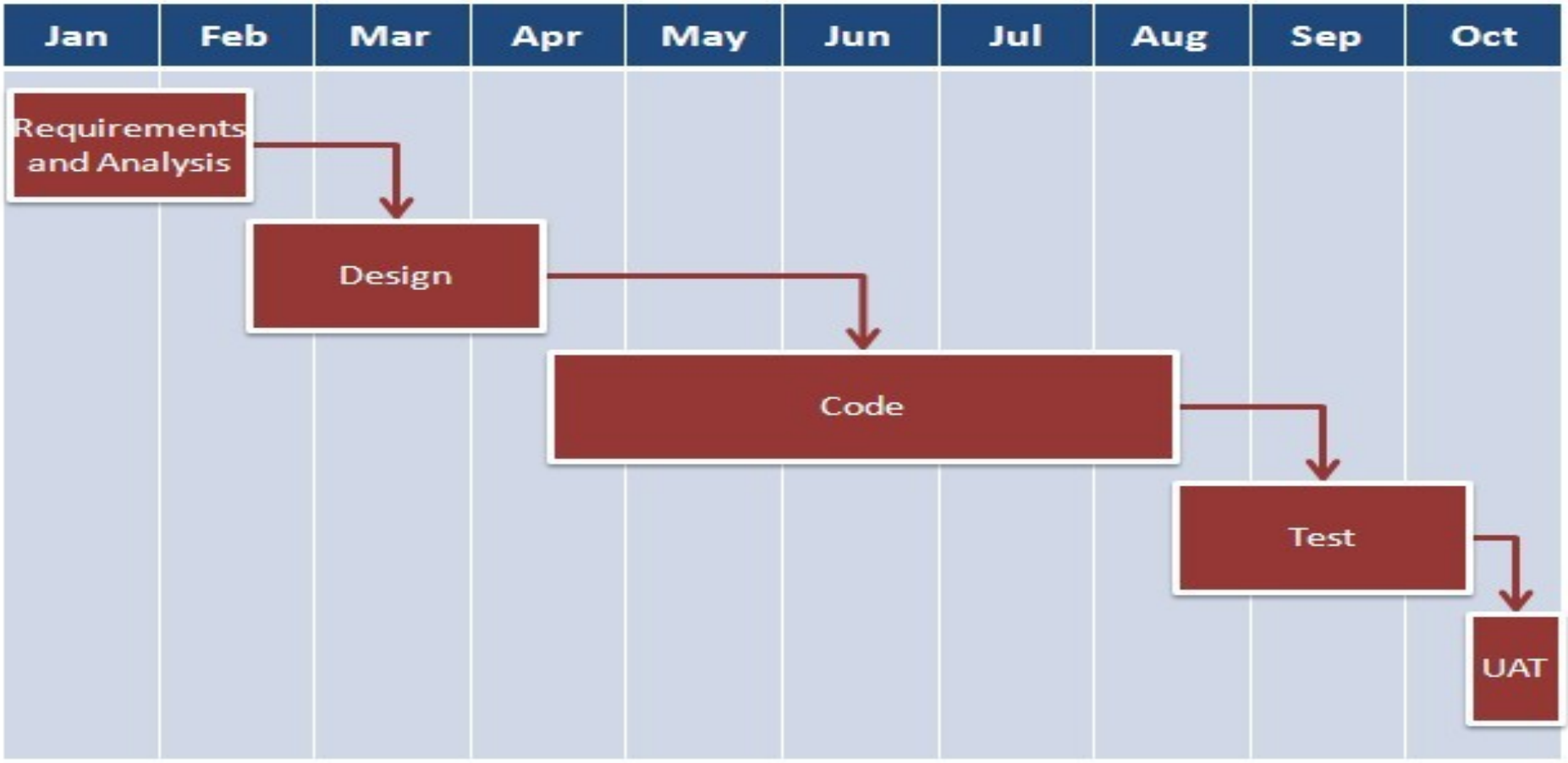
- That provides all the features provided by MS Word and any other features requested by the marketing team.
- The final product needs to be ready in 10 months of time.

Let us see how this project is executed in traditional and Agile methodologies.

In traditional Waterfall model –

- At a high level, the project teams would spend 15% of their time on gathering requirements and analysis (1.5 months)
- 20% of their time on design (2 months)
- 40% on coding (4 months) and unit testing
- 20% on System and Integration testing (2 months).
- At the end of this cycle, the project may also have 2 weeks of User Acceptance testing by marketing teams.
- In this approach, the customer does not get to see the end product until the end of the project, when it

Project Schedule in Waterfall Model

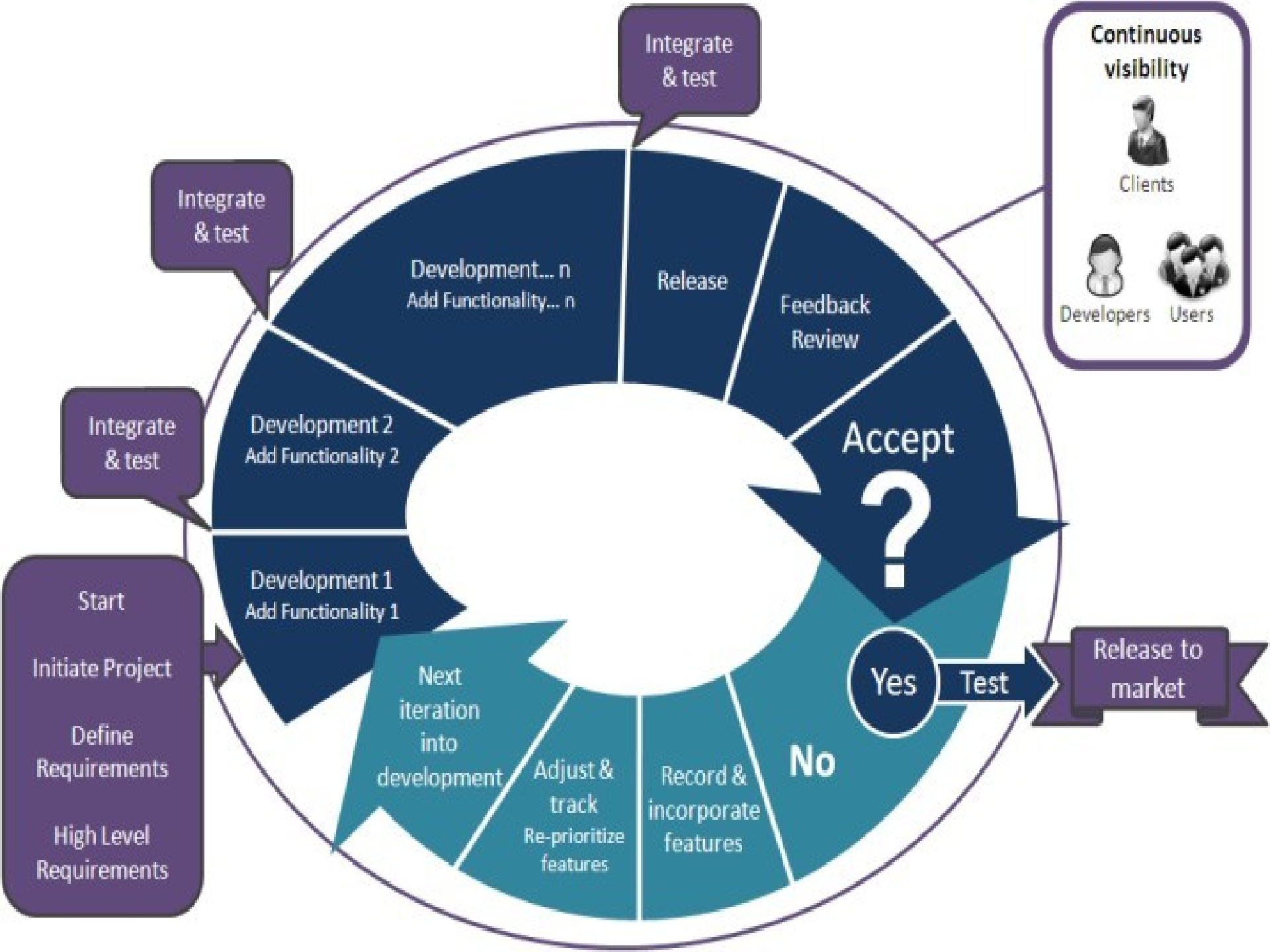


With Agile development methodology –

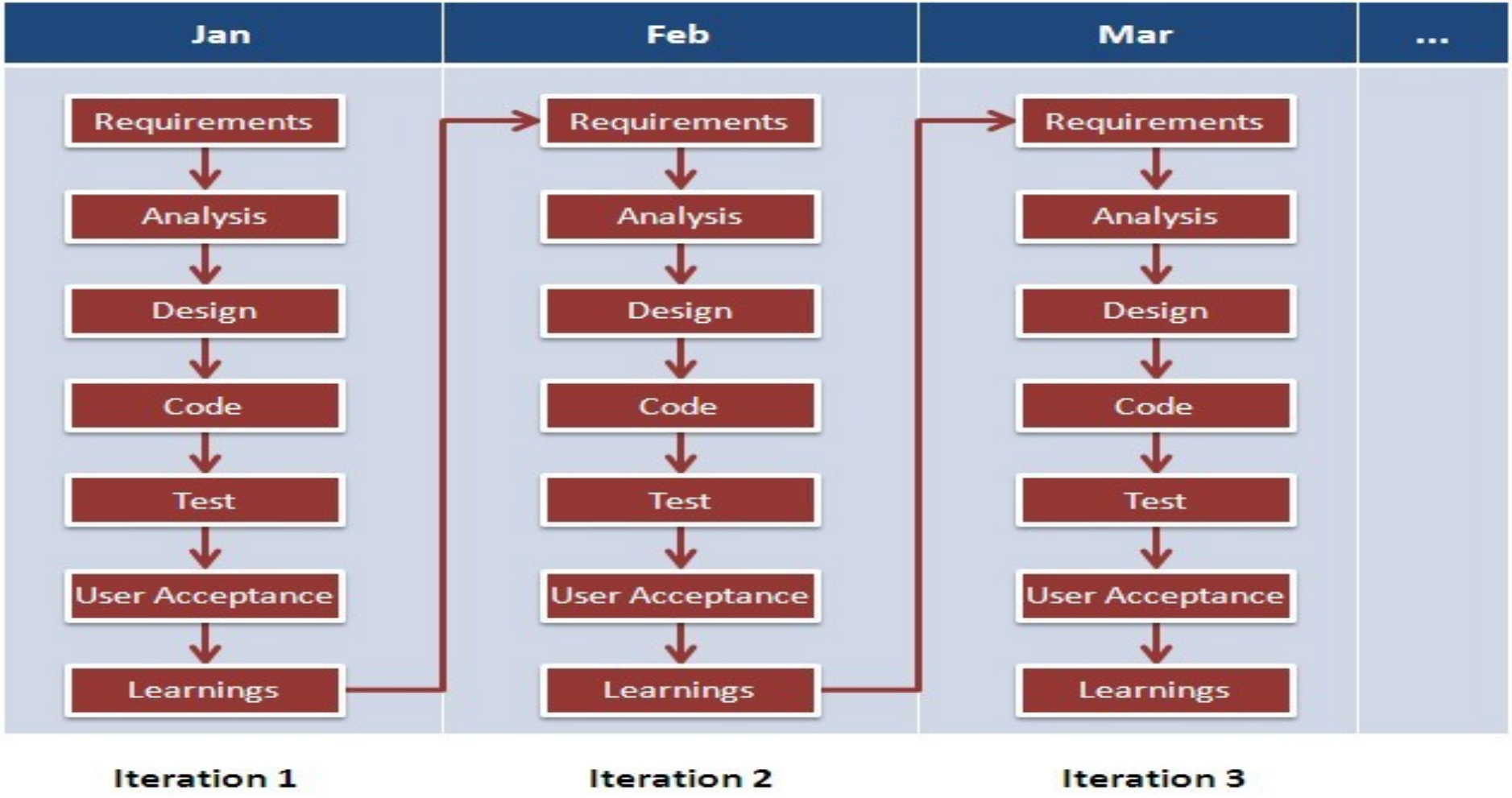
- In the Agile approach, each project is broken up into several ‘Iterations’.
- All Iterations should be of the same time duration (between 2 to 8 weeks).
- At the end of each iteration, a working product should be delivered.
- In simple terms, in the Agile approach the project will be broken up into 10.
- Rather than spending 1.5 months on requirements gathering, in Agile software development, the team will decide the basic core features

With Agile development methodology –

- Any remaining features that cannot be delivered in the first iteration will be taken up in the next iteration or subsequent iterations, based on priority.
- At the end of the first iterations, the team will deliver a working software with the features that were finalized for that iteration.
- There will be 10 iterations and at the end of each iteration the customer is delivered a working software that is incrementally enhanced and updated with the features that were shortlisted for that iteration.


Integrate & test
Integrate & test
Integrate & test
Development... n
Add Functionality... n
Release
Feedback
Review
Development 2
Add Functionality 2
Development 1
Add Functionality 1
Accept
?
Yes
Test
No
Release to market
Next iteration into development
Adjust & track
Re-prioritize features
Record & incorporate features
Start
Initiate Project
Define Requirements
High Level Requirements
Continuous visibility
Clients
Developers
Users

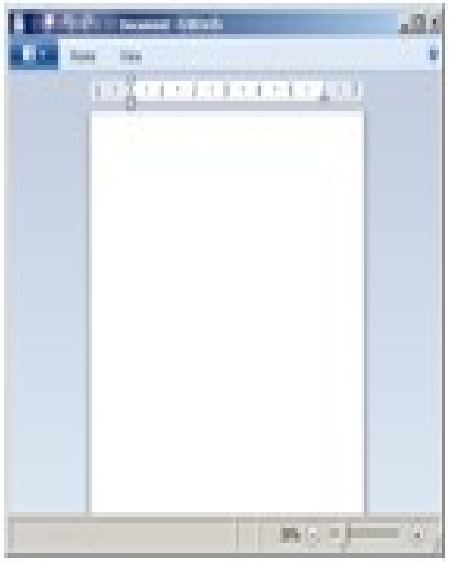
Project Schedule in Agile Model



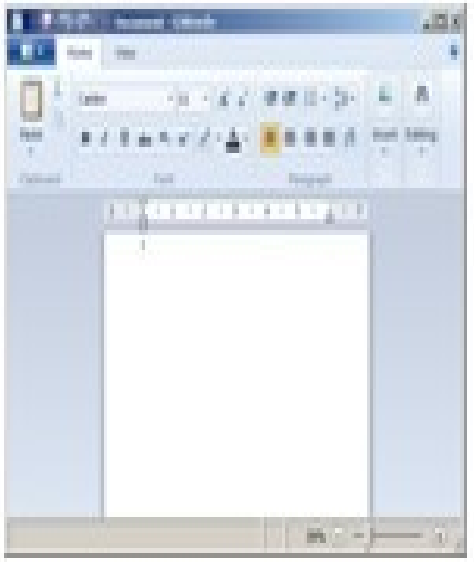
Software evolve through Iteration

Iteration 1

Iteration 2



Iteration 3



Comparison

Waterfall	Iterative	Spiral	Agile
• Basic • Rigid • Inflexible • Not for Real projects	• Module by module delivery • Easy to test and debug	• Risk evaluation • Not for Small Projects • No early lock on Requirements	• Flexible • Advanced • Parallel Process divided into builds