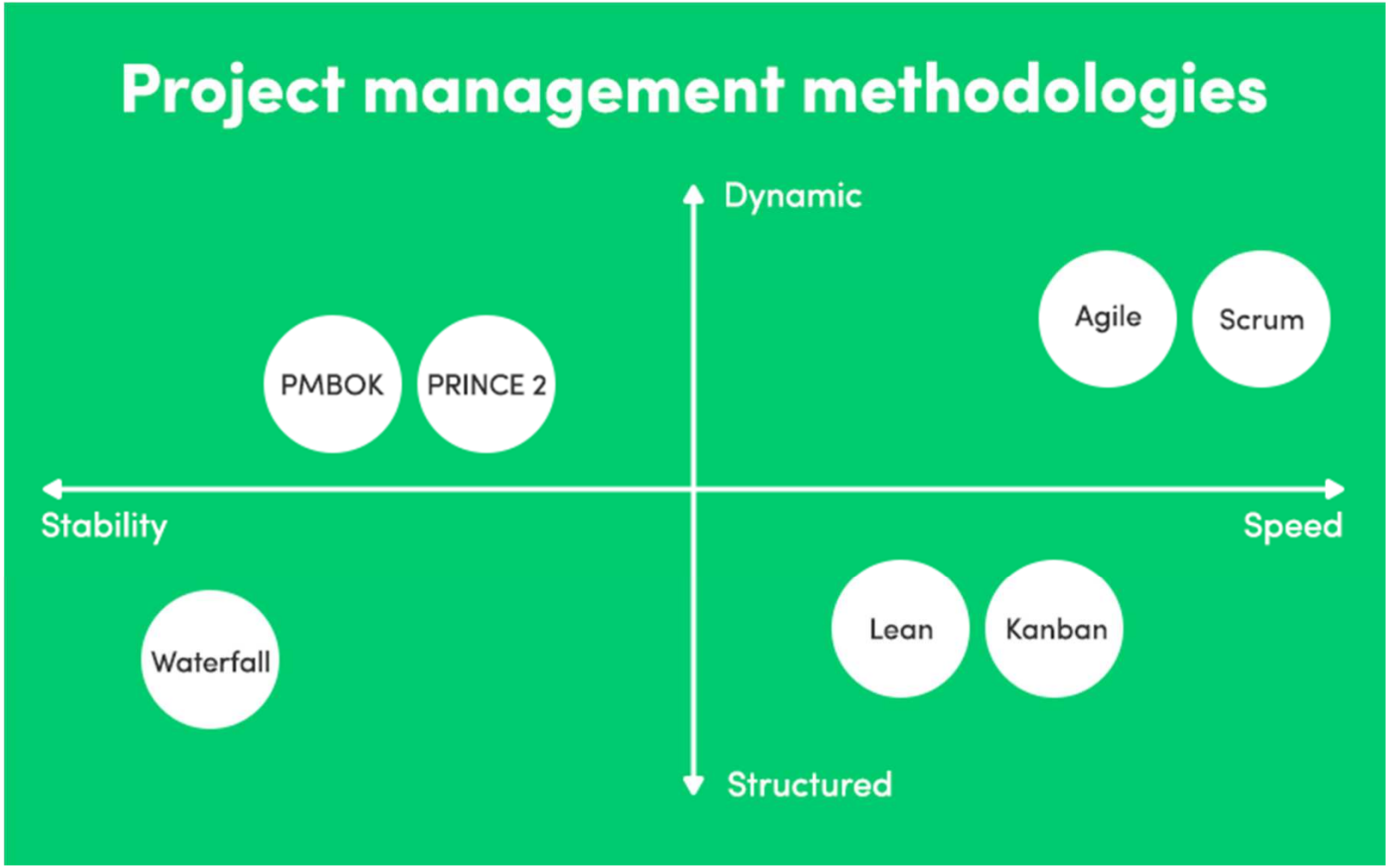
TRADITIONAL SOFTWARE PROJECT MANAGEMENT METHODOLOGIES: WATERFALL

Dr. SELİN METİN
Netaş Telekomünikasyon A.Ş.

Overview

- The Traditional Waterfall Approach
  - Phases of Waterfall Methodology
  - How Waterfall Deals with Software Requirements
  - Advantages of Using Waterfall
  - Disadvantages of Using Waterfall
  - Agile and Waterfall
- Iterative Waterfall Development
  - Feedback Paths Introduced by the Iterative Waterfall Model
  - Advantages of Iterative Waterfall Model
  - Drawbacks of Iterative Waterfall Model
- Incremental Models: Spiral Model
  - Phases of Spiral Model
  - Advantages of Spiral Model
  - Disadvantages of Spiral Model
- What is AgileFall?
  - AgileFall Phases

7 Top Project Management Methodologies



The Traditional Waterfall Approach

- The first formal description of the waterfall model is often cited as a 1970 article by Winston W. Royce.
  - Royce did not use the term "waterfall" in this article.
  - Royce presented this model as an example of a flawed, non-working model.

- The first established modern approach to building a system
  - Originated in the manufacturing and construction industries, both highly structured environment where changes can be too expensive or sometimes impossible

What is Waterfall Method?

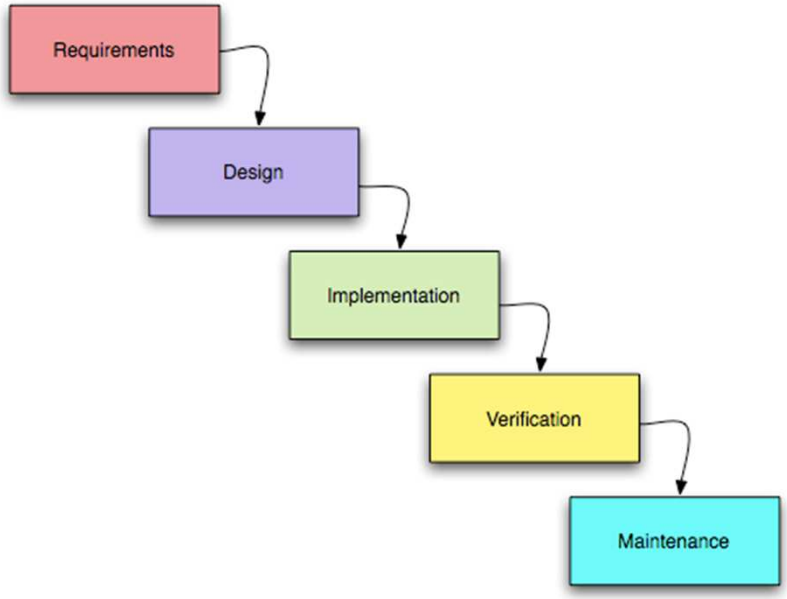
- A sequential design process for project management
- Everything flows logically from the beginning of a project through the end
- As each of the methodology stages is completed, the developers can move on to the next step.
- There's no chance for changes or errors, so your project outcome and a detailed plan must be set in the beginning.
- While less flexible than agile management, the waterfall method provides a clear roadmap of desired outcomes for both programmers and clients.
  - It can save time and money by offering a well-defined strategy.
  - It's not the most popular method currently, but it is a worthwhile system, and can be implemented with almost any project management software.

Reminder: Linear PMLC - 1

- Traditional PMLC
- The five process groups are each executed once in the order shown in the figure.
- Major weakness: knowledge gained from one process group cannot be used to revise and improve the deliverables from a previously completed process group.
- A scope change request from the client upsets the balance in the Linear PMLC model schedule and the resource schedule as well.
  - **The Linear PMLC model is change intolerant.**

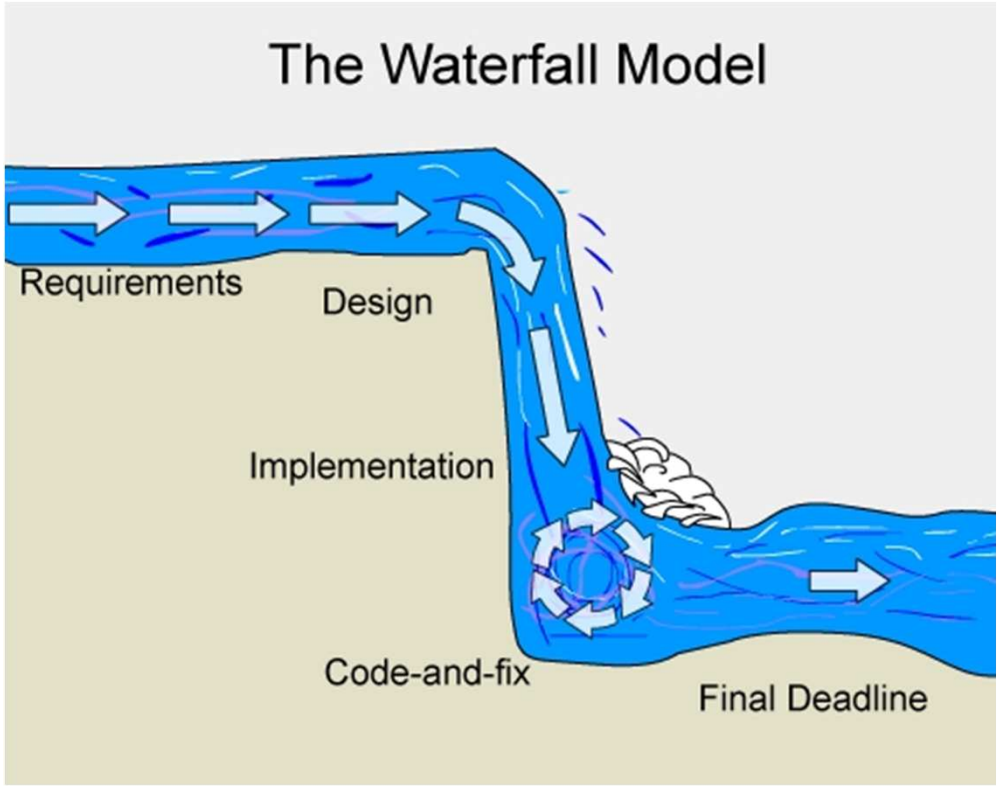
Reminder: Linear PMLC - 2

- Clearly defined solution and requirements
- Not many scope change requests
- Routine and repetitive projects
- Uses established templates



Phases of Waterfall Methodology - 1

- Requirements
- Analysis
- Design and construction
- Testing phase
- Installation/ deployment
- Support and maintenance



Phases of Waterfall Methodology - 2

- Requirement Gathering and Analysis:
  - The basement of any development process where all the requirements are gathered.
  - All this information is documented in a requirement specification document.
- Analysis
  - When all requirements are gathered and documented, it's time to analyze whether they are valid or not.
- Design and construction
  - This stage may contain implementation, development, and coding processes.
  - The design specifications created in this phase are used in the coding phase.
  - The requirements are studied and evaluated, and the design of the system is prepared.
  - Goal is to understand what actions need to be taken and what they should look like.

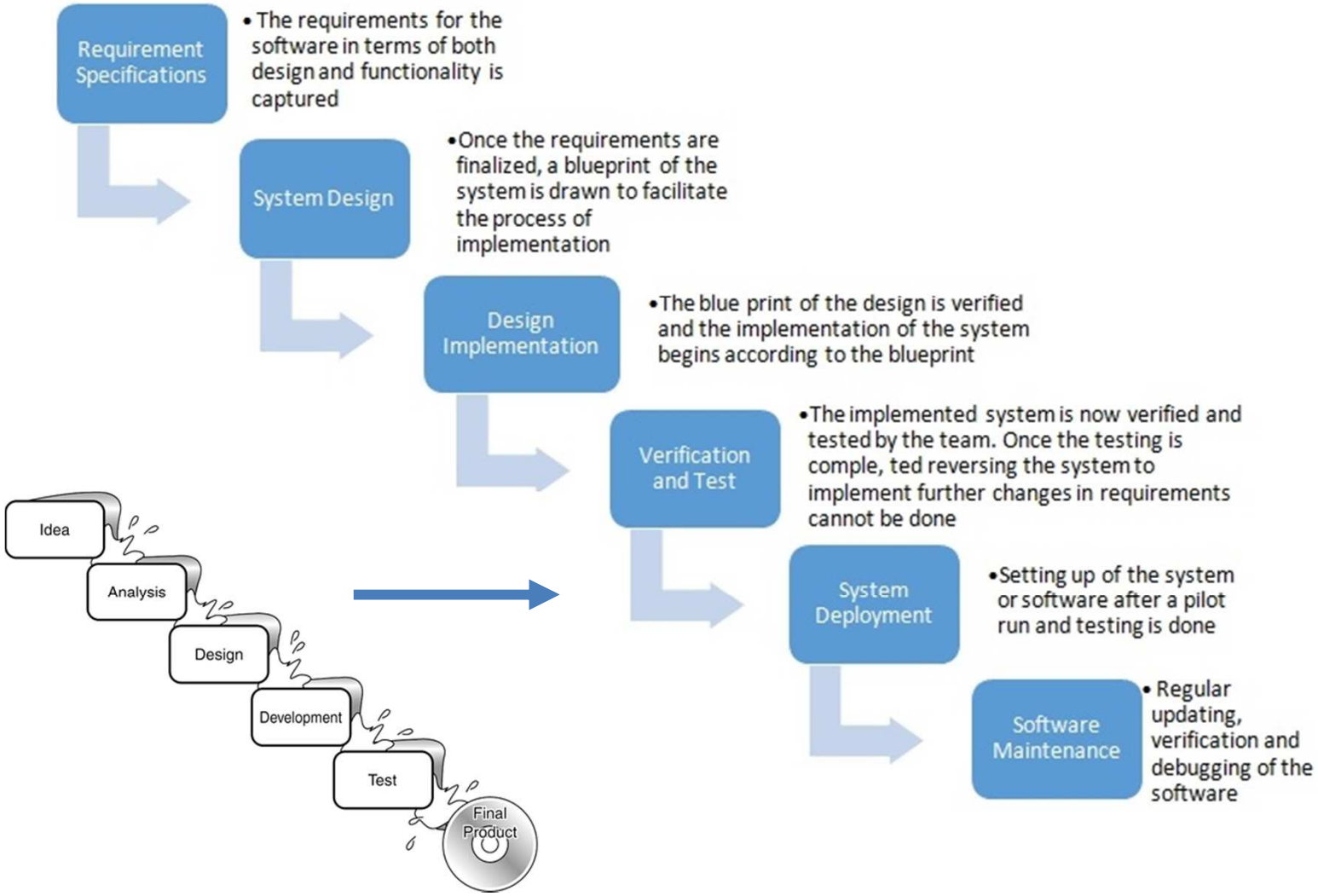
Phases of Waterfall Methodology - 3

- Testing phase
  - Once the code is complete, the software needs to be tested for any errors.
  - It's crucial to make sure the product meets all the customer's requirements.
  - When the testing is finished, the software is delivered to the customer.
  - Some teams may choose to include user acceptance testing (UAT), where users test the software before it is deployed to the general public.

- Installation/ deployment
  - Installation is a phase when the product is implemented according to all requirements.
  - Some testing processes may take place at this stage.

- Support and maintenance
  - The final product is delivered to the customer.
  - Maintenance and support are put into place.
  - The development team will need to resolve, change, or modify the software to continue to be effective.

Hence the Term “Waterfall”



The Role of Project Managers in Waterfall Methodology

- Create the requirements and foresee all pertinent questions

- Requirements must be given as complete as possible because the team works with the research and design on the initial stages

  → efforts of product managers are much heavier at the beginning of the project

How Waterfall Deals with Software Requirements

- Waterfall projects define all software requirements upfront.
  - The project cannot proceed unless these requirements have been identified and documented.

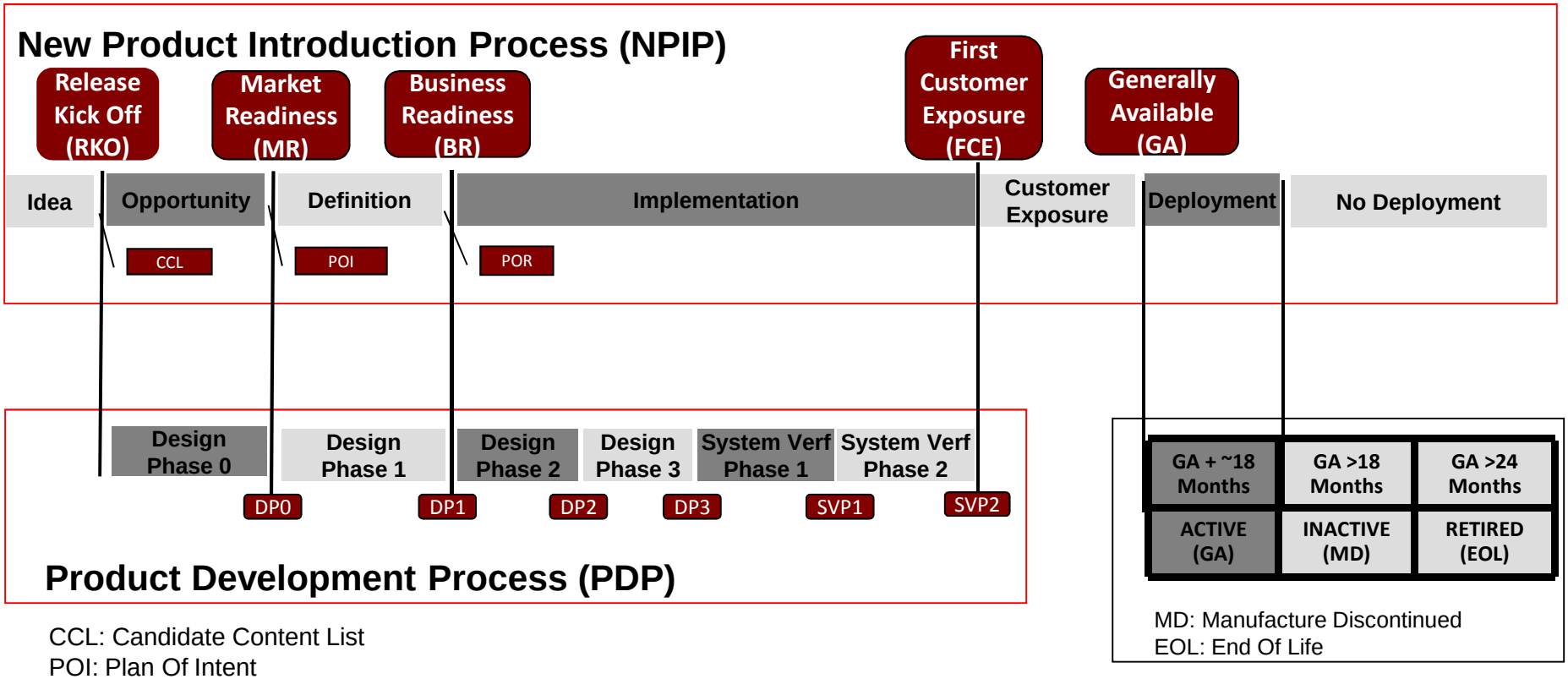
- Some Waterfall projects may have a dedicated team to capture, collect, and gather the requirements.
  - Questionnaires, face-to-face or phone interviews, white boards, and modeling tools may be used to capture stakeholder and customer requirements.

- Once the initial requirements are defined, the team should produce a requirements specification document (sometimes they may create more than one).

  → defines what needs to be delivered so everyone understands the scope of the project.

Waterfall Example for a S/W Project

- Waterfall phases for software product release



CCL: Candidate Content List
POI: Plan Of Intent
POR: Plan Of Record

Example Phase Definitions

IDEA: Define the project concept, capture the customer and business value proposition and ensure that the proposition is aligned to company's core strategic vision.

OPPORTUNITY: Define and capture the market opportunity, finalize the requirements specifications, identify design requirements, and to approve initial project plan.

DEFINITION: Define and capture the product solution and to ensure all key functional requirements have been captured in the product design

IMPLEMENTATION: Develop and validate the product. All product delivery processes and tools are confirmed/established and systems are confirmed/established to support the product.

CUSTOMER EXPOSURE: Perform specific activities with customers and/or channel partners in a controlled fashion prior to the start of the deployment phase

DEPLOYMENT: Deliver product to meet, or exceed, market and business requirements, establish and/or maintain customer relationships, drive for continuous improvement in quality, efficiency and cost and identify requirements to stop production

NO DEPLOYMENT: Execute plans to stop production, maintain customer relationships, continue cost-effective product maintenance and identify/execute plans that support the decision to end the product's life

Waterfall Milestone Deliverables Example

- Business decision points (RKO, MR, BR, FCE,GA) key deliverables

BDP	Owner	Key Deliverable
RKO	PLM	Candidate Content List (CCL)
	PLM	Feature Requirements Specification (Available)
MR	Design	Design Phase 0 (DP0) Exit Criteria Met
	Marketing	Marketing & Sales Readiness Plan - Approved
	PLM	Plan Of Intent (POI) [Approved]
	PLM	Product Business Plan (including Business Case Financials) @ MR [Approved]
BR	CustDoc & Train	Customer Documentation & Training Release Management Plan (PMP) [Approved]
	Design	Design Phase 1 (DP1) Exit Criteria Met
	PLM	Plan Of Record (POR) [Approved]
	Pricing	Pre-Commercial Pricing Strategy Package
	Proj Office	Approach Proclaimed: First Customer Exposure BDP & Customer Exposure Phase
	SPLM	Services Definition & Delivery Plan [Approved]
	Supply Chain	Supply Chain Readiness Plan [Approved]
FCE	CustDoc & Train	Training Plan For Channel Partners [Approved]
	CustDoc & Train	Customer Documentation Available for customer on portal
	Design	Design Phase 2 (DP2), DP3, System Verification Phase 1(SVP1) & SVP2 Exit Criteria Met
	Marketing	Marketing & Sales Readiness Plan - Executed
	Proj Office	Customer Exposure Phase Plan Approved
GA	CustDoc & Train	Customer Documentation Updated for customer on portal; DVD available
	CustDoc & Train	Training Courses available for customers
	Proj Office	Customer Exposure Phase Plan Executed [If Applicable]
	Supply Chain	Supply Chain Readiness Plan Achieved

Advantages of Using Waterfall - 1

- The phases do not overlap. They are processed and completed one at a time.
- Waterfall software development methodologies are good for small projects that contain clear requirements.
- Excellent technical documentation is part of the deliverables
  - Because you can't go back to a previous activity, you're forced to create a comprehensive documentation from the start, listing all the requirements you can think of, resulting in better understanding of the logic behind the code and tests.
  - Easier for new programmers to ramp up during the maintenance phase
  - If your company has employee turnover, the strong documentation allows for a minimal project impact.
- Client input not required
  - After requirements phase, client input is minimal (save for occasional reviews, approvals, and status meetings). This means you don't have to coordinate with them and wait for when they're available.

Advantages of Using Waterfall - 2

- Easy to use and manage
  - Waterfall is also a rigid model; each phase has specific deliverables and review, so it's easy to manage and control. It is easier to measure progress by reference to clearly defined milestones
  - Discipline is enforced: By focusing on requirements and design before writing code, the team can reduce the risk of a missed deadline.
  - The emphasis on requirements and design brings minimal wastage of time and effort and reduces the risk of schedule slippage, or of customer expectations not being met.

- Easy to understand
  - Divided in discrete and easily understandable phases→ project management is straightforward and the process is easily understandable even to non-developers.
  - Follows the same sequential pattern for each project → the team doesn't need any prior knowledge or training before working on a Waterfall project
  - Because developers and customers discuss end goals in extensive detail, both parties have a solid understanding of the primary expectations and desired outcomes. As a result, it is much easier to guarantee that the project is on the right track.

- Testing is easier as it can be done by reference to the scenarios defined in the functional specification

Advantages of Using Waterfall - 3

- Better design
  - Design errors are captured before any software is written saving time during the implementation phase.
  - Products have a higher cohesion because during the design phase you know everything that must be taken into account. There is no one-feature-at-a-time problem that leads to usability problems down the road.
- In the maintenance phase, programmers and clients can observe the developed application in order to determine whether it operates properly.
  - If there are problems, the programmers can correct the errors before handing the finished product over to the clients.
- Can save both time and money
  - Programmers only develop the code that is actually needed, which saves both time and effort during the initial stages.
  - Developers can correct any errors or mistakes early on in the process, which allows the implementation phase to unfold smoothly.
  - Because the end goal is clearly defined in the beginning stages, all parties understand the financial and time constraints of the project.
  - Finalizing each cycle before moving on to the next guarantees success in the end.

Advantages of Using Waterfall - 4

- Client knows what to expect
  - Clients can know in advance the cost, size and timeline of the project so they can plan their business activities and manage cash flow according to the plan.
- The total cost of the project can be accurately estimated after the requirements have been defined (via the functional and user interface specifications).
- Team members can better plan their time
  - Because everyone knows in advance on what they'll work, they can be assigned on multiple projects at the same time.
  - Workers can easily determine how much progress they are making, since expectations are clearly defined before a project begins.
- Ideal for teams who are working closely together, since it offers clear-cut tasks and goals.
- Enables developers to continue working on the project without needing constant supervision from the manager.

Disadvantages of Using Waterfall - 1

- Changes can't be easily accommodated
  - If you find a requirement error or need to change something, your project has to be started from the beginning with a new code.
- QA too late to be useful
  - Testing is done at the end of the project which means that developers can't improve how they write code based on QA feedback.
  - It is not easy to go back and change something that is unclear and not well thought out in the concept phase.
  - Designers can't foresee all the problems that will arise from their design, and once those problems surface, it's very difficult to fix them.
- You cannot solve some essential problems, using Waterfall for object-oriented and complex projects.
  - It is also not a good idea to use it for long and ongoing projects.
- Software isn't delivered until late
  - The project has to complete two to four phases before the coding actually begins.
  - Stakeholders won't see working software until late in the life cycle.

Disadvantages of Using Waterfall - 2

- Customers can be not satisfied with the delivered product.
  - Clients will often find it difficult to state their requirements at the abstract level of a functional specification and will only fully appreciate what is needed when the application is delivered.
  - As all tasks and deliverables are based on documented requirements, customers may not see what will be delivered until it's almost finished.
  - If the client realizes they need more than they initially thought, the project plan will need a major overhaul (as well as the budget).
  - Then it becomes very difficult (and expensive) to re-engineer the application.

- No going back
  - Because Waterfall is a linear, sequential model, you can't bounce between phases, even if unexpected changes occur. Once you're done with a phase, that's it.
  - Once you're finished with one activity, it's difficult and expensive to go back and make changes. This puts a huge pressure on the planning.
  - The model does not cater for the possibility of requirements changing during the development cycle.

- No room for error during requirements phase
  - Everything relies heavily on the requirements phase and if you make an error, the project is doomed.

Disadvantages of Using Waterfall - 3

- Gathering accurate requirements can be challenging
  - It can be difficult to pinpoint exactly what the customers and stakeholders want early in the project.
  - Often customers don't know what they want early on and identify requirements as the project progresses.
- The method isn't appropriate for the requirements that have the high risk of changing.
- A project can often take substantially longer to deliver than when developed with an iterative methodology such as the agile development method.
- Deadline creep
  - Once one activity is late, all the other activities are late too, including the project deadline.
- Bug ridden software
  - Because the testing is done at the end, most teams tend to rush the testing in order to deliver the project on time and hit their incentives. This short-term wins lead to sub-par quality and long-term problems.

Waterfall is suited for projects where

- Budget, requirements, and scope are fixed
  - Eg. you're building a one-off project which doesn't need further development
- You can accurately estimate the work
  - You're familiar with technology and you've done the same work before
- You can't afford to iterate
- Project is innately low-risk
  - You're building a clone of something that already works
- Project has a hard ship date
  - Eg. you have to ship a video game by Christmas
- Your users can't or won't update software
  - Doesn't apply to web applications where updates are seamless

You shouldn’t use waterfall

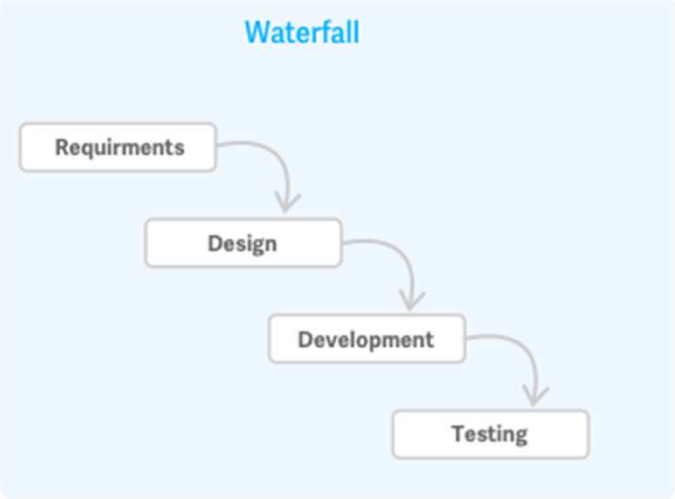
- Where a working prototype is more important than quality
  - Eg. you first need to test if there’s a market demand
- When you don’t know what the final product should look like
- Where client doesn’t know exactly what they need
- When the product is made for an industry with rapidly changing standards
- When you know you won’t get the product first the right time and have to incorporate user feedback
- When your users are happy with v1.0 and you can ship additional features as time goes on

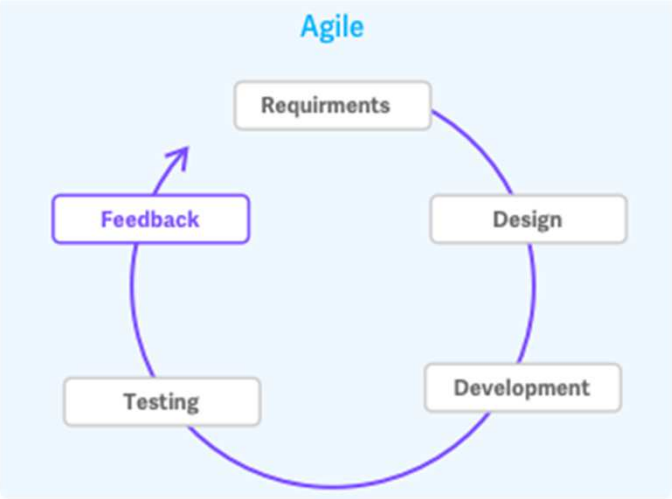
Waterfall is not that different from Agile

- Waterfall is always mentioned as the antithesis to Agile, which makes sense.
  - Waterfall projects have a hard time dealing with changes while agile projects welcome change.
- No matter what methodology you use, change is not a good thing.
  - Change always means additional scope, delay, and expenses.
  - Agile is better at minimising the effects of change, but they still happen.
  - Also, agile teams have the culture where change is OK, which is maybe the most important benefit of being agile.
- But once you scratch behind the surface and look both from purely process perspective, waterfall is very similar to agile.

Agile and Waterfall - 1

- Once you break down any agile workflow, you'll still get a set of activities that follow one another
- If you treat waterfall projects as smaller phases within a big project, you'll end up with agile.
- Agile still fits in the traditional project management, only the point of view changes
  - Activities on a project are waterfall and if you treat the whole project as a series of iterations, it's agile.



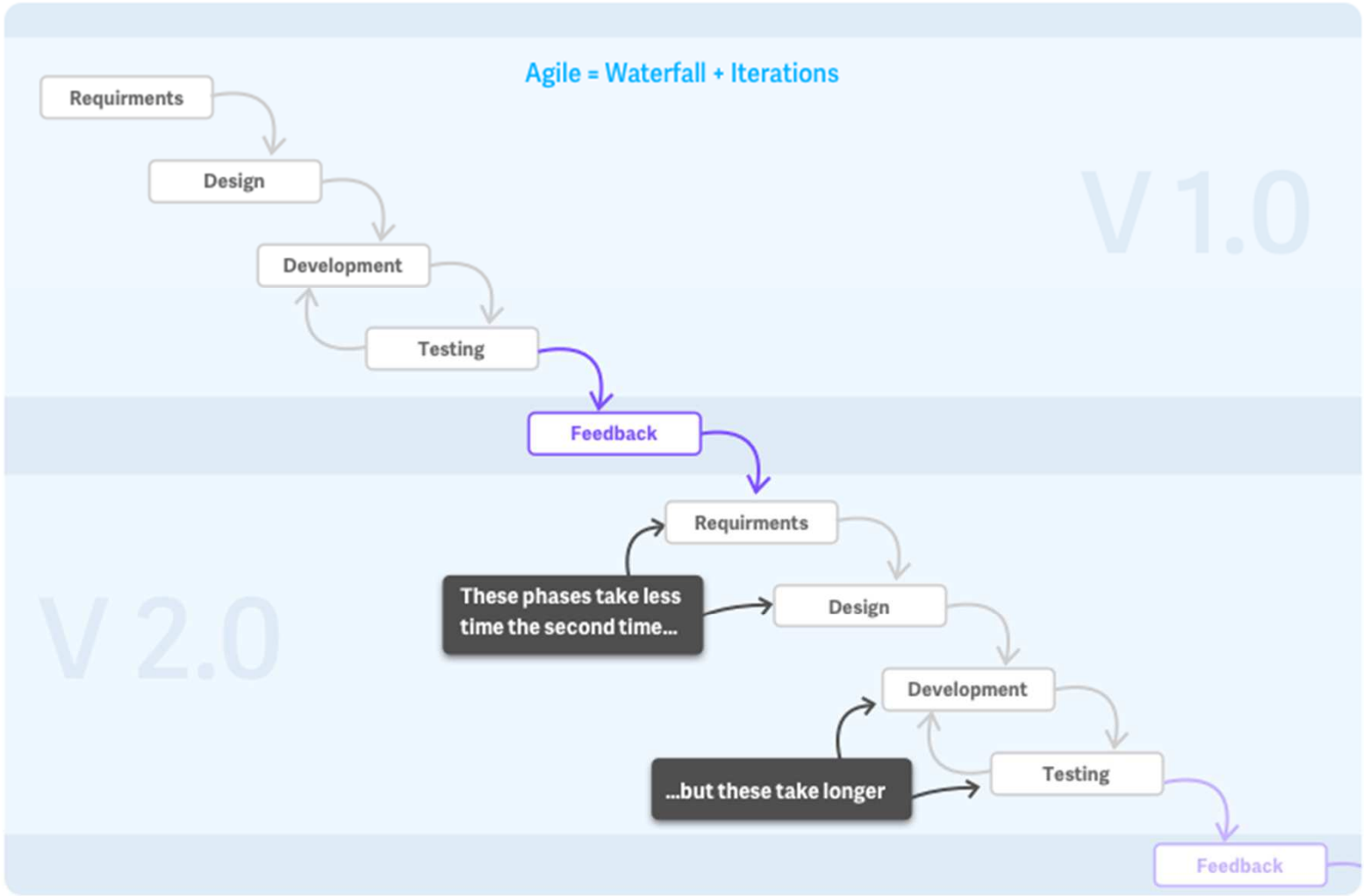


Agile and Waterfall - 2

- Waterfall projects are projects where the client decided on zero iterations.
- In agile projects, the number of iterations is decided on by the customer. Because things are all done within an iteration in agile, the logical assumption was that an iteration equaled a project. But an iteration is more properly referred to as a phase or subphase of the project.

  - PMI
- Agile still fits in the traditional project management, only the point of view changes.
  - Instead of treating each iteration as a separate project, iterations are just phases in one big project.
- **The real difference between the waterfall and agile: in waterfall the clients are heavily engaged at the beginning and then their engagement declines; while in agile, the client is constantly engaged.**
- No organization is purely agile or waterfall.
  - Agile and waterfall are more about the culture and type of work the organization does than how they do it.
  - Most organizations use waterfall milestones but work according to agile principles between those.

Agile and Waterfall - 3



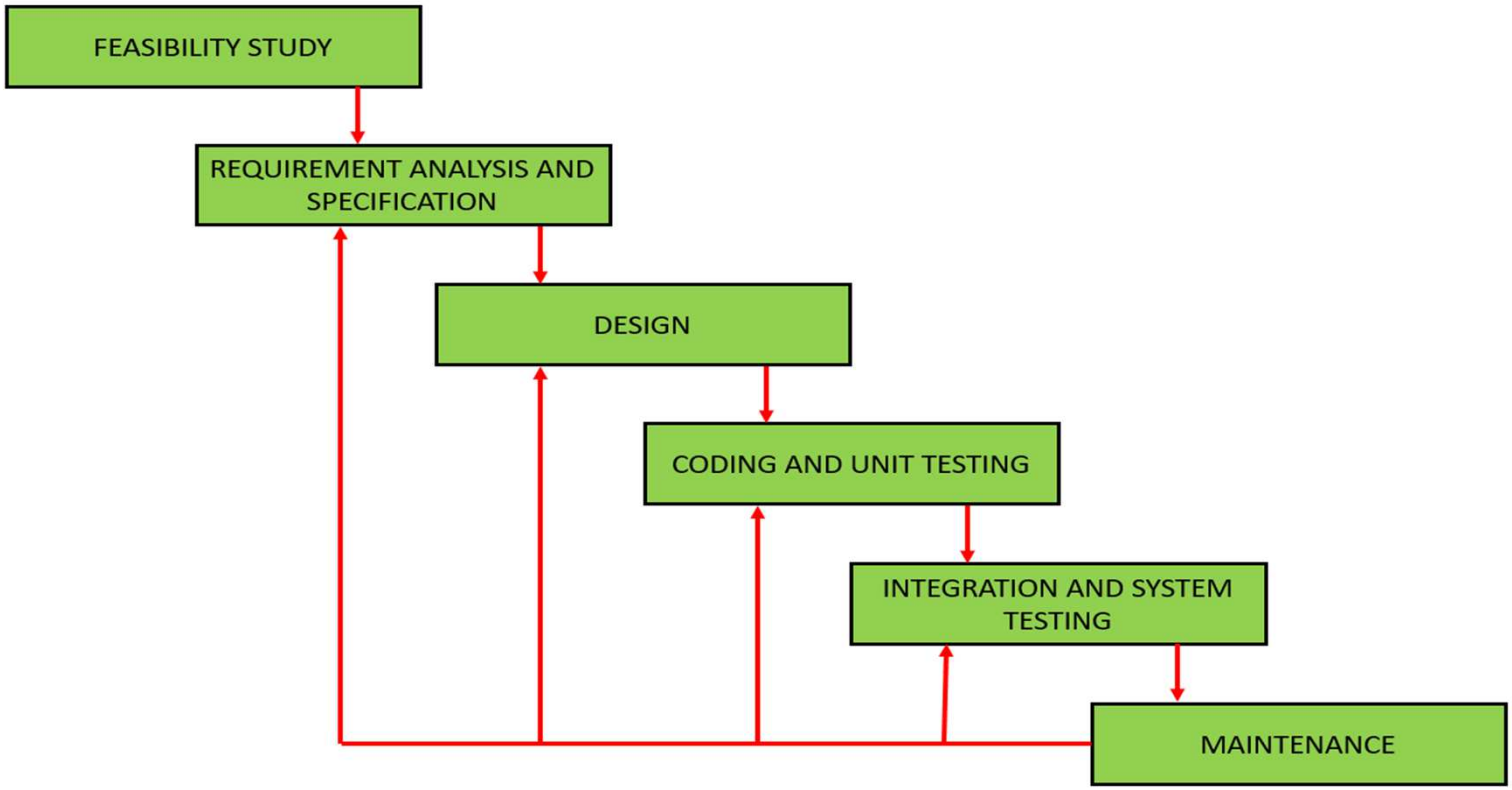
Iterative Waterfall Development

- In the traditional Waterfall model, the team goes through each phase for the entire project.
  - For example, they do the analysis for the entire project, then they do the design for the entire project, etc.
- In an iterative waterfall model, there is still a lot of upfront planning required.
  - Once the plan is in place, the team follows the same pattern as traditional Waterfall but does it for each story.
  - They do the analysis for one story, then all the design for one story, then all the coding and testing for one story.
  - Then they repeat the process for another story.
- The work is broken up into chunks that benefit the development team.

Iterative Waterfall

- This approach carries less risk than a traditional waterfall approach but is still far more risky and less efficient than a more Agile approaches.
- 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.
- The iterative waterfall model provides feedback paths from every phase to its preceding phases, which is the main difference from the classical waterfall model.
  - It is good to detect errors in the same phase in which they are committed. It reduces the effort and time required to correct the errors.
- **Phase Containment of Errors**: The principle of detecting errors as close to their points of commitment as possible is known as Phase containment of errors.

Feedback Paths Introduced by the Iterative Waterfall Model



- When errors are detected at some later phase, these feedback paths allow the phase to be reworked and correct errors committed by programmers during some phase.
- **There is no feedback path to the stage** – feasibility study, because once a project has been taken, does not give up the project easily.

Advantages of Iterative Waterfall Model

- Feedback Path
  - In the classical waterfall model, there are no feedback paths, so there is no mechanism for error correction.
  - In iterative waterfall model feedback path from one phase to its preceding phase allows correcting the errors that are committed and these changes are reflected in the later phases.

- Simple
  - Iterative waterfall model is very simple to understand and use. That's why it is one of the most widely used software development models.

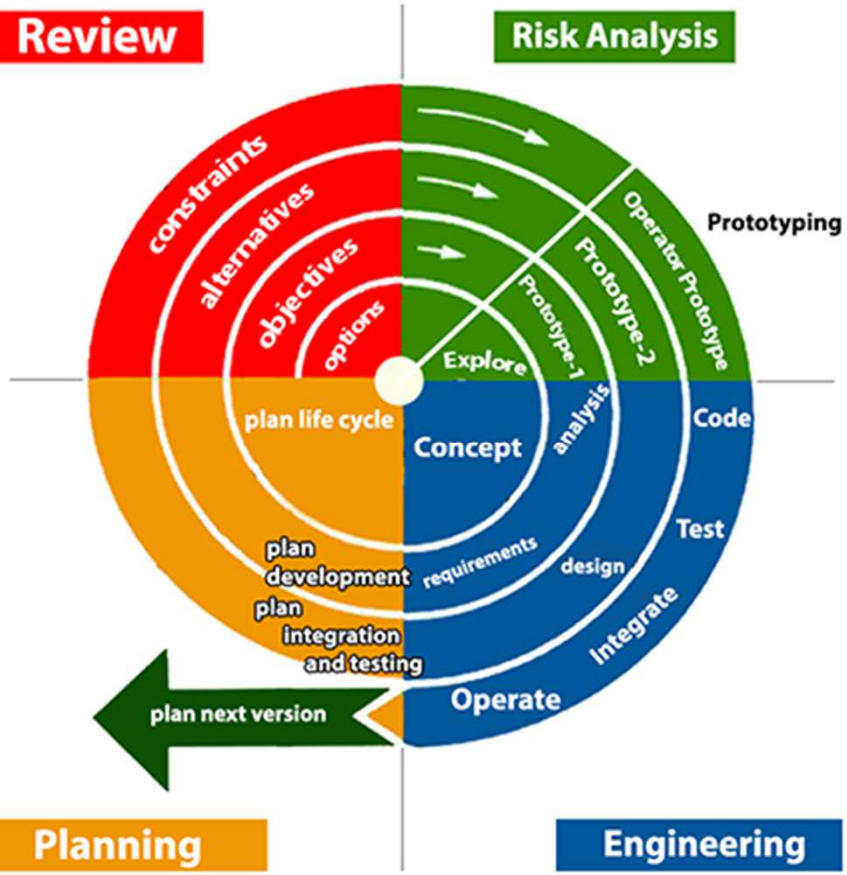
Drawbacks of Iterative Waterfall Model

- Difficult to incorporate change requests
  - The **major drawback** is that all requirements must be clearly stated before starting the development phase.
  - Does not leave any scope to incorporate change requests that are made after development phase starts.
- Incremental delivery not supported
  - The full software is completely developed and tested before delivery to the customer.
  - Customers have to wait long for getting the software.
- Overlapping of phases not supported
  - Iterative waterfall model assumes that one phase can start after completion of the previous phase, but in real projects, phases may overlap to reduce the effort and time needed to complete the project.
- Risk handling not supported
  - Projects may suffer from various types of risks but there is no mechanism for risk handling.
- Limited customer interactions
  - Customer interaction occurs at the start of the project at the time of requirement gathering and at project completion at the time of software delivery.
  - This may lead to many problems as the final software may differ from the customers' actual requirements.

Incremental Models: Spiral Model

- It is a systems development method used in information technology
- This model of development combines the features of the prototyping model and the waterfall model.
  - The spiral model is similar to the incremental model, with more emphasis placed on risk analysis
- The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation.
  - Software project repeatedly passes through these four phases in iterations (called Spirals).
  - The baseline spiral: starting in the planning phase, requirements are gathered and risk is assessed.
  - Each subsequent spirals builds on the baseline spiral.

Spiral Software Development Life Cycle Model



Phases of Spiral Model

- Planning
  - Requirements are studied and gathered. Feasibility study is done.
  - Reviews and walkthroughs to streamline the requirements
  - Finalized list of requirements.
  - The project is reviewed and a decision made whether to continue with a further loop of the spiral.
  - If it is decided to continue, plans are drawn up for the next phase of the project.
- Risk Analysis
  - Requirements are studied and brain storming sessions are done to identify the potential risks
  - Once the risks are identified, risk mitigation strategy is planned and finalized
  - Document which highlights all the risks and its mitigation plans
- Engineering Works
  - Actual development and testing if the software takes place in this phase
  - Code
  - Test cases and test results, test summary report and defect report.
- Review and Evaluation
  - Customers evaluate the software and provide their feedback and approval
  - Features implemented document

Spiral Model is Metamodel

- Spiral model is also called as meta-model because in a way it comprises of other models of SDLC.
- Both waterfall and prototype models are used in it.
- In spiral model, software development is done systematically over the loops (adhering to waterfall approach) and at the same time we make a prototype and show it to user after completion of various phase (just in case of prototype model).
- This way we are able to reduce risks as well as follow systematic approach

When to Use Spiral Model?

- When costs and risk evaluation is important.
- For medium to high-risk projects.
- Users are unsure of their needs.
- Requirements are complex.
- Significant changes are expected.

Advantages of 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.
- Risk reduction mechanisms are in place.
- Supports iteration and reflects real-world practices –Systematic approach

Disadvantages of 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.
- Complex, relatively difficult to follow strictly.
- Applicable only to large systems
- It is costly for smaller projects.

What is AgileFall?

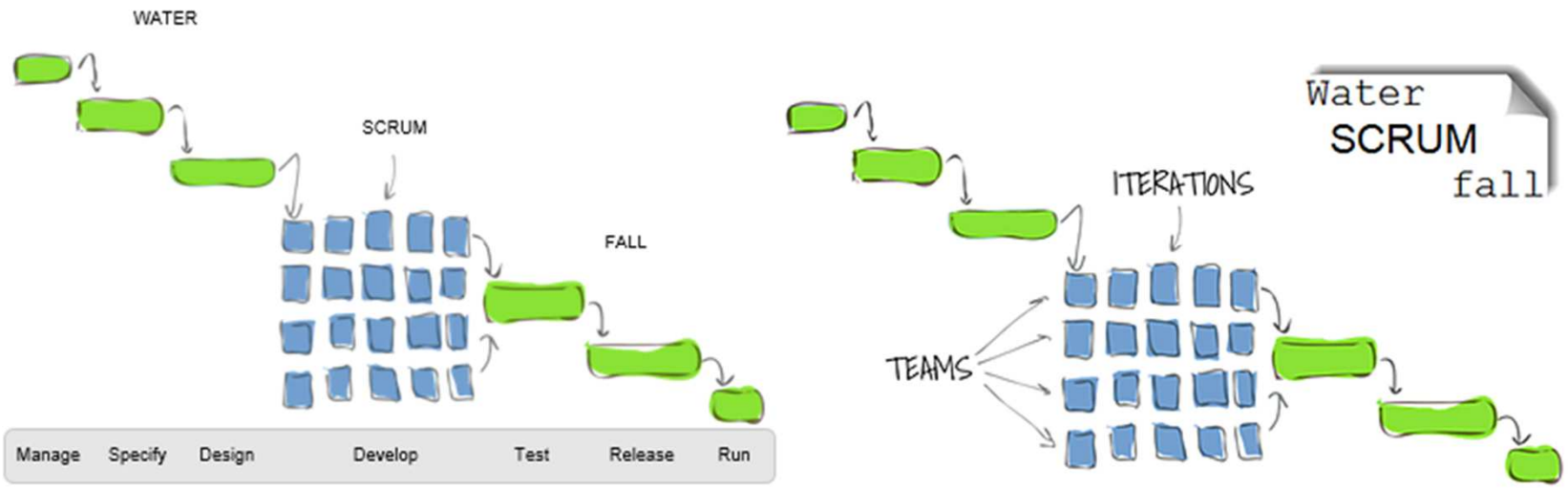
- Agile-fall is the process of applying Agile practices in Waterfall-like steps.

- Teams working in sprints or iterations can still structure these in measured steps:
  - Sprint 1: Gather requirements.
  - Sprint 2: Design your tests.
  - Sprint 3: Run those tests.
  - Sprint 4: Fix bugs.
  - Sprint 5: Regress those bugs.

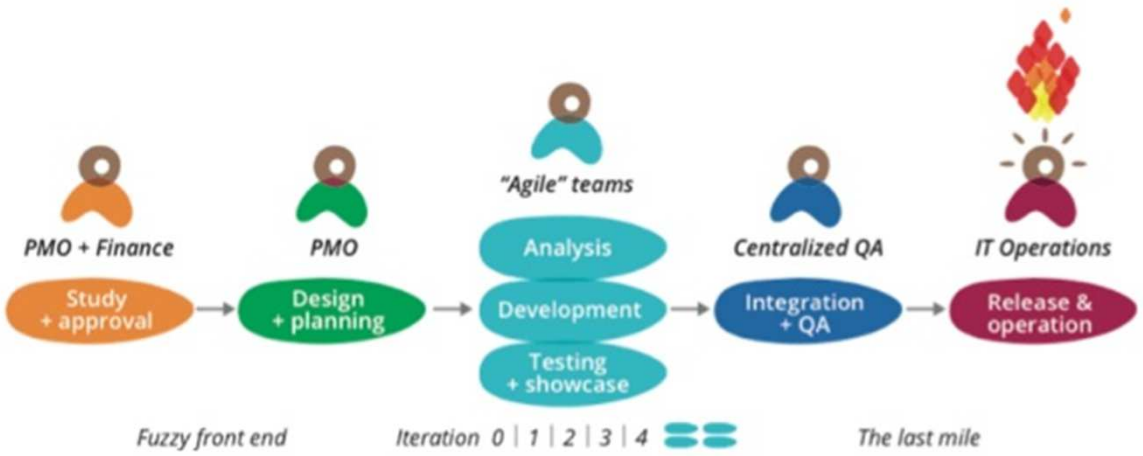
When AgileFall is Used?

- A software development model where you are trying to be agile, but you keep falling into waterfall development habits. For example:
  - We have sprints, but they are four or six weeks long.
  - We try to do most of the development at the beginning of the sprint and most of the testing at the end of the sprint.
  - We have a product backlog with priorities, but we start working on a release with a long list of features already committed to the business.
  - We know our product release dates several months in advance. There's a long list of target dates that must be met before the release date.
  - We test new features as they are developed in each sprint, but we also have a long system / globalization / translation / integration test cycle at the end of the release.
  - We sometimes spend too much time up front designing the software before we even start prototyping it.
  - We do our initial estimates in person-weeks, because we're not comfortable with story points and velocities yet.
  - When management asks us for a rough sizing, we might spend days working on that sizing effort, because we feel like we need to really understand the tasks involved before we "commit" to a sizing.
  - People get upset (or defensive) when stories are not completed in the sprint they were planned for.
  - We demo our software in development to customers, but only after we're happy with it. By the time we get around to having a demo, it might be too late to change much.

AgileFall Phases - 1



AgileFall Phases - 2



Summary

- **Waterfall Approach**: A sequential design process for project management. As each of the methodology stages is completed, the developers can move on to the next step. There's no chance for changes or errors, so your project outcome and a detailed plan must be set in the beginning.
  - **The real difference between the waterfall and agile: in waterfall the clients are heavily engaged at the beginning and then their engagement declines; while in agile, the client is constantly engaged.**
- **Iterative Waterfall Development**: 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.
  - The iterative waterfall model provides feedback paths from every phase to its preceding phases, which is the main difference from the classical waterfall model.
  - The major drawback is that all requirements must be clearly stated before starting the development phase.
- **Spiral Model**: waterfall and prototype models are used in it.
  - Software development is done systematically over the loops (adhering to waterfall approach) and at the same time we make a prototype and show it to user after completion of various phase (just in case of prototype model).
- **Agile-fall** is the process of applying Agile practices in Waterfall-like steps.