CITY UNIVERSITY

Assignment No -01

Department Name: Computer Science & Engineering

Course Code : CSE-325

Course Name: System Analysis & Design

Submitted BY

Name: Nitu Akter
#ID: 171442639
Program: CSE(Evn)
Batch: 44th

Submitted To

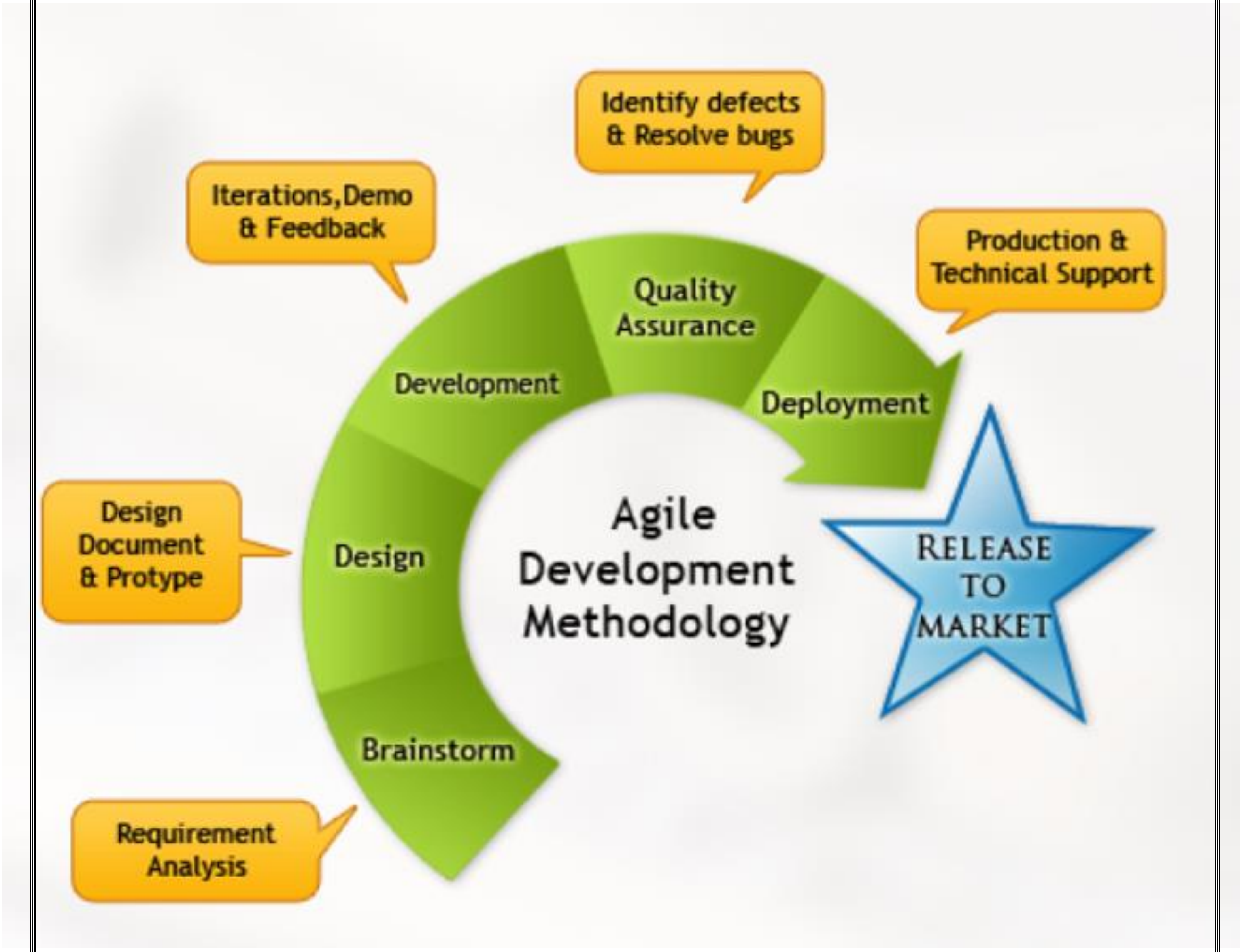
Rechard Philip
Sr.LECTURER ,DEPARTMENT OF CSE
CITY UNIVERSITY, BANGLADESh

Date of Submission: 15 May 2019

Agile Development Methodology Process

Contents

AGILE DEVELOPMENT METHODOLOGY PROCESS

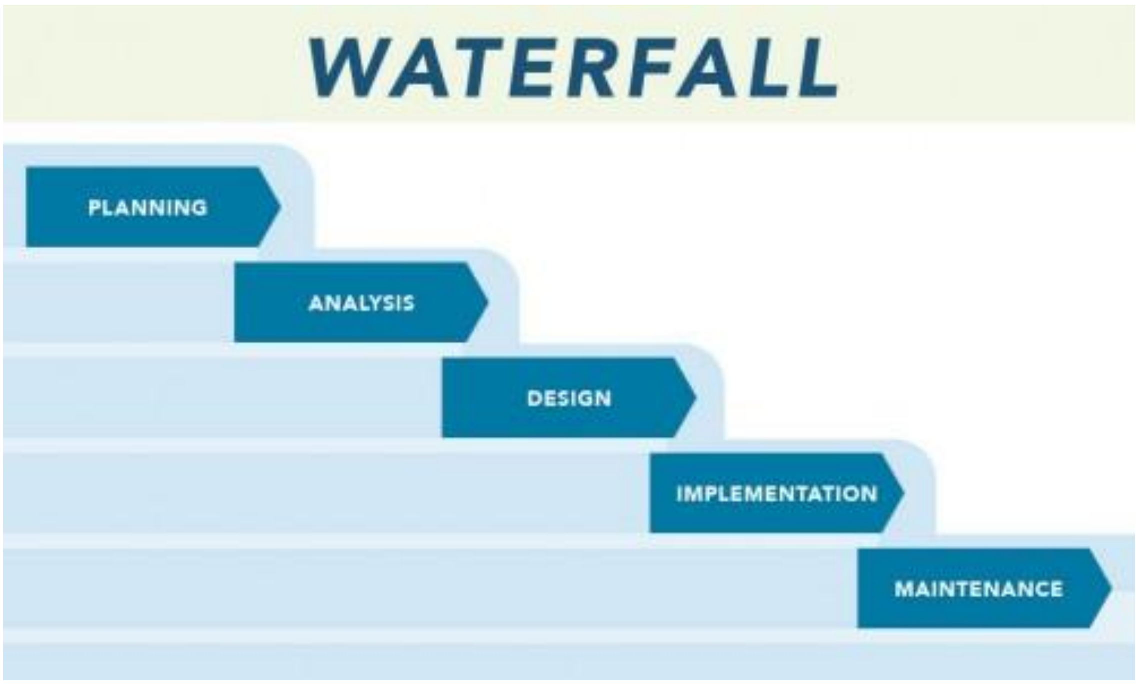

Identify defects & Resolve bugs
Iterations,Demo & Feedback
Production & Technical Support
Quality Assurance
Development
Deployment
Design Document & Protype
Design
Agile Development Methodology
RELEASE TO MARKET
Brainstorm
Requirement Analysis

About Agile Methodology:

Agile methodology has taken the software development world by storm and rapidly cemented its place as “the gold standard.” Agile methodologies all started based on four core principles as outlined in the Agile Manifesto. These methodologies are rooted in adaptive planning, early delivery and continuous improvement, all with an eye toward being able to respond to change quickly and easily. As a result, it’s no surprise that 88% of respondents in Version One’s 2017 State of Agile Report ranked “ability to adapt to change” as the number one benefit of embracing agile. However, as more and more development teams adopt an agile philosophy, testers have struggled to keep pace. That’s because the widespread adoption of Agile has led teams to issue releases and totally undocumented software on a more frequent basis. This frequency has forced testers to shift when they conduct testing, how they work with developers and BAs and even what tests they conduct, all while maintaining quality standards.

What is the waterfall development methodology:

The waterfall model is a linear, sequential approach to the software development life cycle (SDLC) that is popular in software engineering and product development. The waterfall model emphasizes a logical progression of steps. Similar to the direction water flows over the edge of a cliff, distinct endpoints or goals are set for each phase of development and cannot be revisited after completion. The term was first introduced in a paper published in 1970 by Dr. Winston W. Royce and continues to be used in applications of industrial design.



The waterfall methodology is composed of seven non-overlapping stages:

1. Requirements: Potential requirements, deadlines and guidelines for the project are analyzed and placed into a functional specification. This stage handles the defining and planning of the project without mentioning specific processes.
2. Analysis: The system specifications are analyzed to generate product models and business logic_ that will guide production. This is also when financial and technical resources are audited for feasibility.
3. Design: A design specification document is created to outline technical design requirements such as programming language, hardware, data sources, architecture and services.
4. Coding/Implementation: The source code is developed using the models, logic and requirements designated in the prior stages. Typically, the system is designed in smaller components, or units, before being implemented together.
5. Testing: This is when quality assurance, unit, system and beta tests take place to report issues that may need to be resolved. This may cause a forced repeat of the coding stage for debugging. If the system passes the tests, the waterfall continues forward.
6. Operation/Deployment: The product or application is deemed fully functional and is deployed to a live environment.
7. Maintenance: Corrective, adaptive and perfective maintenance is carried out indefinitely to improve, update and enhance the final product. This could include releasing patch_ updates or releasing new versions.

The agile manifesto has four important values:

1. Individuals and interactions over processes and tools

The value #1 in the Agile Manifesto is known as “Individuals and interactions over processes and tools.” It’s crucial to value people more highly than processes or tools. It is the people who respond to business needs and drive the development process. When the development is driven by the process or tools, then the team is less responsive to changes and less likely to meet customer needs.

2. Software over documentation

Historically, documenting the product for development and ultimate delivery required enormous amounts of time. Technical requirements, specifications, prospectus, testing plans, interface design documents, and different approvals were required for each. There were long delays in development. Agile streamlines documentation, not eliminating it. In this form, it gives developers what they need to do without getting bogged down in minutiae. Agile values documentation, but it values working software more.

3. Customer collaboration over contract negotiation

Negotiations are important. It's about the period when customers and a product manager work out the details of a delivery with all the details. However, collaboration is a different creature entirely and it really matters. According to Waterfall, customers negotiate the requirements for the product, often in great detail, prior to any work starting. The Agile Manifesto describes a customer who collaborates throughout the development process, making. It's easier for development to meet their needs of the customer. According to Agile methods, the customer may be included at intervals for periodic demos, but a project could just as easily have an end-user as a daily part of the team.

4. Responding to changes over following a plan

Traditional software development regarded change as an expense. It was to be avoided. According to Agile, the shortness of an iteration means priorities can be shifted from iteration to iteration and new features can be added into the next iteration. Changes always improve a project and provide additional value.

12 principles of Agile development:

These principles can be your litmus test to define whether or not you're being Agile. So, always remember them in a short form:

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.

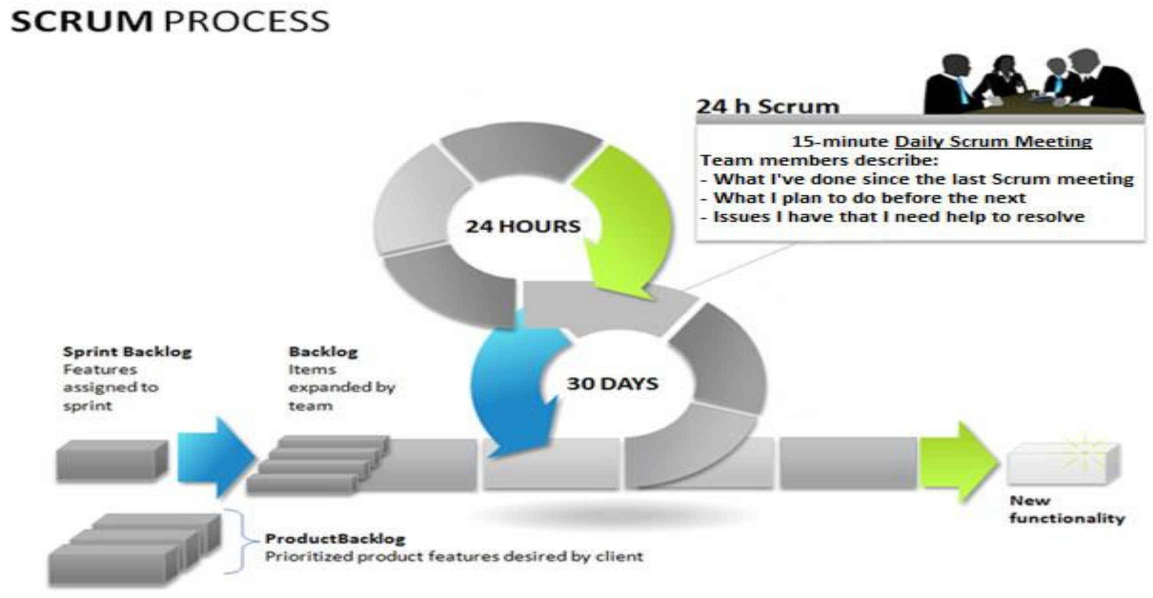
4. Business people and developers must work together daily throughout the project.

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.
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.

How an agile development methodology works?

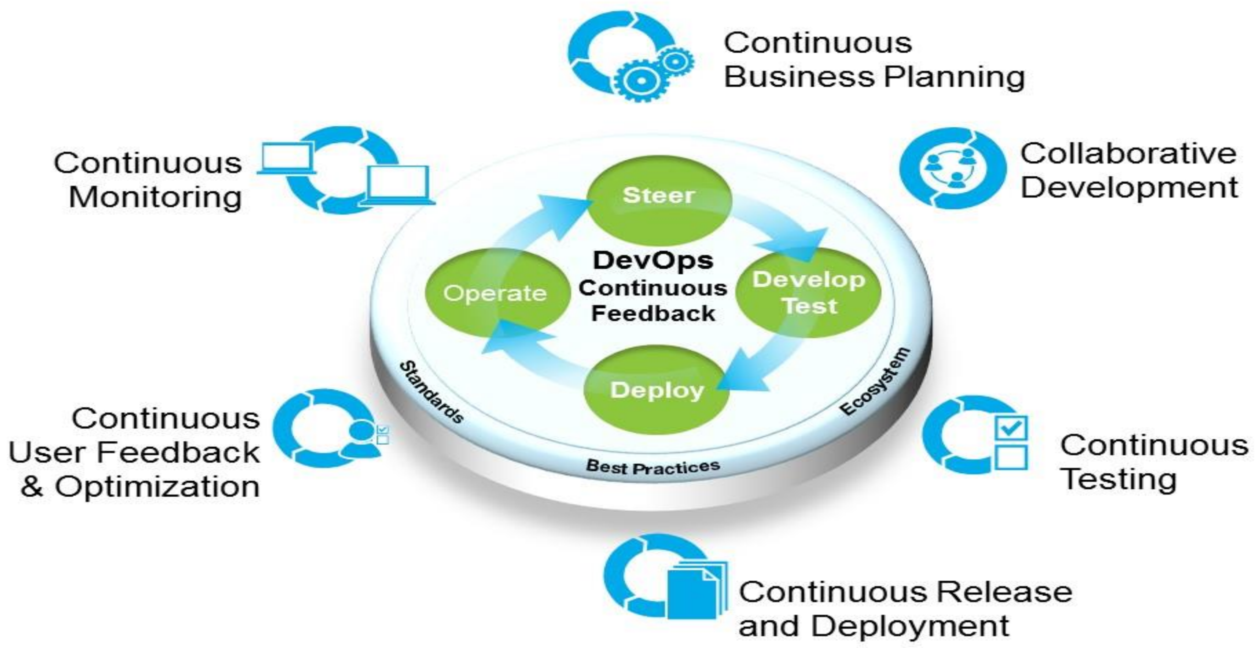
In the agile Scrum world, instead of providing complete, detailed descriptions of how everything is to be done on a project, much of it is left up to the Scrum software development team. This is because the team will know best how to solve the problem they are presented. This is why in Scrum development, for example, a sprint planning meeting is described in terms of the desired outcome (a commitment to a set of features to be developed in the next sprint) instead of a set of Entry criteria, Task definitions, Validation criteria, Exit criteria (ETVX) and so on, as would be provided in most methodologies. Scrum relies on a self-organizing, cross-functional team. The scrum team is self-organizing in that there is no overall team leader who decides which person will do which task or how a problem will be solved. Those are issues that are decided by the team as a whole. And in Scrum, a team is cross functional, meaning everyone is needed to take a feature from idea to implementation. Within agile development, Scrum teams are supported by two specific roles. The first is a Scrum Master, who can be thought of as a coach for the team, helping team members use the Scrum process to perform at the highest level. The product owner (PO) is the other role, and in Scrum software development, represents the business, customers or users, and guides the team toward building the right product.


SCRUM PROCESS
24 h Scrum
15-minute Daily Scrum Meeting
Team members describe:
- What I've done since the last Scrum meeting
- What I plan to do before the next
- Issues I have that I need help to resolve
24 HOURS
30 DAYS
Sprint Backlog
Features assigned to sprint
Backlog
Items expanded by team
ProductBacklog
Prioritized product features desired by client
New functionality

Scrum Development: What’s Involved?

The Scrum model suggests that projects progress via a series of sprints. In keeping with an agile methodology, sprints are time boxed to no more than a month long, most commonly two weeks. Scrum methodology advocates for a planning meeting at the start of the sprint, where team members figure out how many items they can commit to, and then create a sprint backlog – a list of the tasks to perform during the sprint. During an agile Scrum sprint, the Scrum team takes a small set of features from idea to coded and tested functionality. At the end, these features are done, meaning coded, tested and integrated into the evolving product or system. On each day of the sprint, all team members should attend a daily Scrum meeting, including the Scrum Master and the product owner. This meeting is time boxed to no more than 15 minutes. During that time, team members share what they worked on the prior day, will work on that day, and identify any impediments to progress. The Scrum model sees daily scrums as a way to synchronize the work of team members as they discuss the work of the sprint. At the end of a sprint, the team conducts a sprint review during which the team demonstrates the new functionality to the PO or any other stakeholder who wishes to provide feedback that could influence the next sprint. This feedback loop within Scrum software development may result in changes to the freshly delivered functionality, but it may just as likely result in revising or adding items to the product backlog. Another activity in Scrum project management is the sprint retrospective at the end of each sprint. The whole team participates in this meeting, including the Scrum Master and PO. The meeting is an opportunity to reflect on the sprint that has ended, and identify opportunities to improve.

DevOps delivers on agile's promise


Continuous Business Planning
Collaborative Development
Continuous Monitoring
Steer
DevOps Continuous Feedback
Operate
Develop Test
Deploy
Standards
Best Practices
Ecosystem
Continuous User Feedback & Optimization
Continuous Testing
Continuous Release and Deployment

In 2001, The Agile Manifesto changed the landscape of software development with the introduction of agile development. Agile methodologies taught developers to break down software development into smaller chunks known as "user stories," which accelerate feedback loops and align product features with market need. The agile movement centered on helping developers and small teams work smarter and more efficiently. The early adopters were primarily small software startups that were eager to disrupt the markets and were willing to trailblazer through trial and error. Slowly, as practices matured and the software community increasingly embraced agile-based methods, the notion of "scale" came to the forefront. Developers were now able to produce functioning software code in increasingly shorter iterations. However, when it got to downstream processes, such as test and deployment (not to mention release), fragmented processes and the dreaded "functional silos" stifled the end goal of agile efforts: to speed the time to market of quality software applications. Agile ultimately gave rise to new processes and technology breakthroughs aimed at streamlining and automating the entire software delivery lifecycle. The advent of continuous integration (CI), the practice of checking code in often, with each "chunk" tested and integrated several times a day into a shared "trunk," created a flood of smaller and more frequent releases, which strained downstream QA and Ops teams. With the release of Jez Humble's breakthrough book, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, the notion of treating the entire software lifecycle as a single process—and one that could be automated— was embraced not only by startups but also by Fortune 1000 companies. This helped elevate the value of agile initiatives that were stalled and blocked and raised the stakes for software delivery as a strategic business initiative. While agile addressed the needs of developers, the investment in DevOps initiatives and CD offers businesses much higher returns—a way to truly realize the goals outlined in the Agile Manifesto and support them throughout the software delivery lifecycle.

Conclusion

Using the systemic review process when applied to the field of nutrition allows for considerable flexibility with regard to the types of questions evaluated, studies included and information captured, as well as the nature of summary statements. Confidence in the results of systematic reviews occurs at a number of levels. These include the transparent nature of the process and involvement of a broad-based research team free of potential biases and vested interests. Confidence also derives from the involvement of trained systematic review methodologists, and, a prior formulation of key questions, search criteria, study evaluation criteria, and information captured for evidence tables, and a priori procedures for obtaining appropriate outside inputs from subject matter experts, sponsors and users while precluding the potential biases and conflicts of interest. Within these boundaries the conclusions are comprehensive in nature and objective in the assessment of the available information, without gong beyond the limits of the data. Recognition of a number of challenges not necessarily encountered in other disciplinary areas can enhance the quality and usefulness of nutrition related systematic reviews. Lastly, important to always keep in mind is that systematic reviews are a tool to be used by expert panels, funding agencies and other groups, and can not serve as a replacement for expert deliberations and organizational policy development. Users of systemic reviews often need to augment the reviews by other sources of information and where uncertainties exist, by application of expert scientific judgment. Systematic reviews are a valuable and independent component—but not the end—to decision making processes by groups responsible for developing science-based recommendations and policies.

References:

https://www.qasymphony.com/blog/agile-methodology-guide-agile-testing/

https://searchsoftwarequality.techtarget.com/definition/waterfall-model

https://medium.com/hygger-io/4-values-of-the-agile-manifesto-and-12-agile-principles-easily-explained-84cd429f69f

https://medium.com/hygger-io/4-values-of-the-agile-manifesto-and-12-agile-principles-easily-explained-84cd429f69f

https://techbeacon.com/app-dev-testing/agile-devops-continuous-delivery-evolution-software-delivery