Introduction to Software Engineering and Agile Methodologies

Dr. Mary Giblin

Content

Section 1

Understanding Software Engineering

What is software?

01

Computer programs and associated documentation such as requirements, design models and user manuals.

02

Software products may be developed for a particular customer or may be developed for a general market.

03

New software can be created by developing new programs, configuring generic software systems or reusing existing software.

The Role of Software Engineering

01

Economic Significance

Software engineering is pivotal in today's developed economies where software often surpasses hardware in terms of costs. Understanding its role is crucial for cost-effective development.

02

Defining Software Engineering

It is an engineering discipline that encompasses all aspects of software production, ensuring quality, maintainability, and efficiency.

03

Importance of Systematic Approach

A systematic and organized approach in software engineering is vital for producing dependable and secure software that meets user expectations.

Software Costs

01

Software costs often dominate computer system costs. The costs of software usually greater than the hardware cost. More of the value (e.g. AI) is in the software.

02

Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs

03

Software engineering is concerned with cost-effective software development

\"Just code it”\

01

Why not only “code it”?

02

Which happens more frequently?

Which is more serious?

Famous Software Failures

Mars Climate Orbiter (Sept. 23rd, 1999)

Hi-tech toilet swallows woman (2001)

ESA Ariane 5 Rocket Flight 501

Therac-25

Famous Software Failures

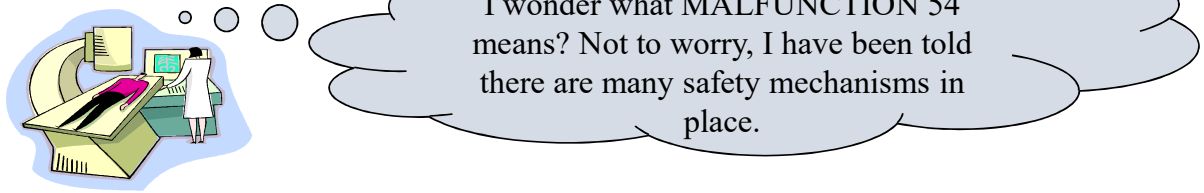
The 125 million dollar Mars Climate Orbiter is assumed lost by officials at NASA. The failure resulted because one development team used imperial measurement and one used metric system of measurement

“The toilet, which boasts state-of-the-art electronic auto-flush and door sensors, steadfastly refused to release Maureen, and further resisted attempts by passers-by to force the door”

Famous Software Failures

- **ESA Ariane 5 Rocket Flight 501**
- "at ~40 seconds into launch
- at an altitude of ~3700m
- the launcher veered off path and began to break up
- the self-destruct system was triggered
- ~$500 million (uninsured, maiden flight)
- the launcher was unmanned

- **Therac-25, computer controlled radiation therapy machine,that killed**
- Five Therac-25 machines were installed in the U.S and six in Canada.
- Between June 1985 and January 1987, Therac-25 massively overdosed six people
- The operator´s manual supplied with the machine does not explain nor even address the malfunction codes



Famous Software Failures

Toyota Prius (2010)

- 400,000 cars recalled to replace software that controls the antilock bracks.

Famous Software Failures Videos

https://www.youtube.com/watch?v=izGSOsAGIVQ (Therac 25)
https://www.youtube.com/watch?v=NSa-AbWiNS0 (toyota prius)
https://www.youtube.com/watch?v=5tJPXYA0Nec (ariane 5)
https://www.youtube.com/watch?v=54z56FbEtRk (air traffic control)

Consequences of Poor Engineering

Famous Failures

Highlighting the Mars Climate Orbiter loss and the Therac-25 accidents underscores the dire consequences of inadequate software engineering practices.

Cost of Errors

These failures exemplify the high costs associated with errors in software development, emphasizing the need for rigorous engineering.

Learning from Mistakes

Analyzing these incidents helps in understanding the importance of quality and precision in software engineering.

Key Principles of Software Engineering

01

Quality and Maintainability

Software engineering aims to create software that is not only functional but also maintainable and of high quality.

02

Dependability and Security

Ensuring that software is reliable and secure is a fundamental goal of software engineering.

03

Efficiency and Acceptability

Software must be efficient in its performance and acceptable to the end-users, meeting their requirements and expectations.

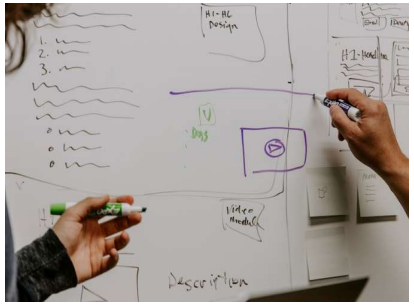
Section 2

The Software Process

Software Process Activities

Specification

The process begins with specifying the requirements and defining what the software should do.

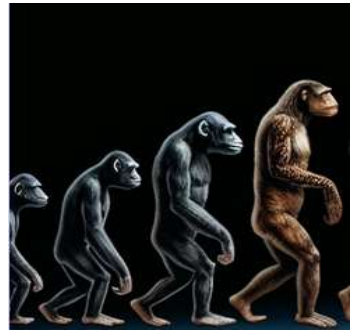


Development

Following specification, the development phase involves designing and coding the software.

Validation

Validation ensures that the software meets the required specifications and fulfills its intended purpose.



Evolution

Changing the software in response to changing demands.

Software Process Models

Waterfall Model

A traditional model where the process is linear and sequential, moving from one phase to the next without going back.

Iterative Development

This model involves repeating the software development cycle multiple times until the final product is achieved.

Component-based Software Engineering

This approach focuses on assembling pre-existing software components to build the final product.

Challenges in Software Engineering

01

Complex Requirements

Dealing with complex and often changing requirements is a significant challenge in software engineering.

02

Compatibility Issues

Ensuring that new software is compatible with existing systems can be difficult.

03

Cost Estimation and Milestones

Accurately estimating costs and setting realistic milestones are critical yet challenging aspects of software engineering.

Importance of Process and Models

Addressing Inherent Challenges

Adopting appropriate processes and models is essential to address the challenges inherent in software engineering.

Ensuring Quality and Efficiency

A well-defined software process helps in ensuring the quality and efficiency of the final product.

Facilitating Maintenance and Evolution

A systematic approach to software engineering facilitates easier maintenance and evolution of software over time.

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

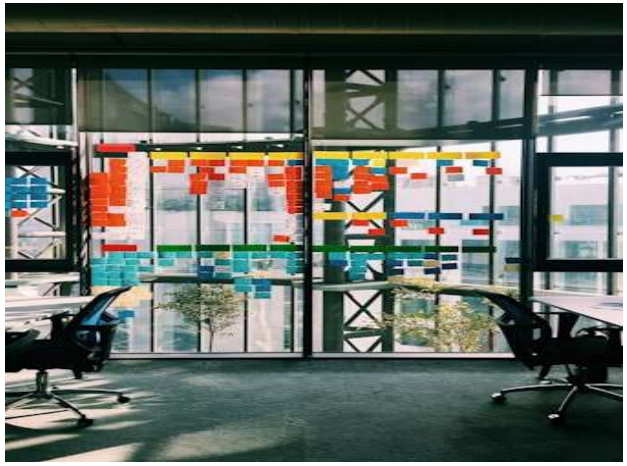
Essential attributes of good software

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

Section 3

Agile Methodology

Introduction to Agile



Defining Agile

Agile is a methodological approach to software development that emphasizes incremental and iterative work sequences.

Focus on Value Delivery

Agile prioritizes delivering value to the customer through working software over comprehensive documentation.

Empowered Teams

Agile relies on cross-functional teams that are empowered to make decisions and adapt to changes quickly.

Agile Principles

Iterative Learning Loops

Agile promotes continuous learning through iterative loops of planning, executing, observing, and adjusting.

Customer Collaboration

Agile methodology values customer collaboration over contract negotiation, ensuring that the product meets customer needs.

Responding to Change

Agile is about embracing change and being able to respond to it quickly, rather than following a fixed plan.

Benefits of Agile

01

For Customers

Customers receive better product quality and more frequent updates, ensuring that the product remains relevant to their needs.

02

For Development Teams

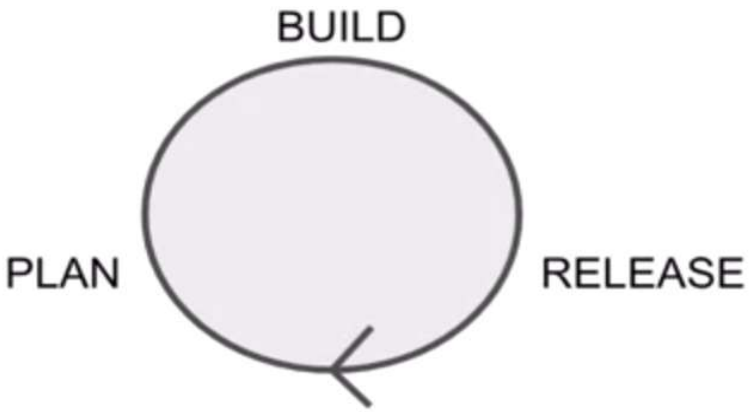
Teams enjoy higher job satisfaction due to empowerment and innovation, leading to lower costs and predictable deliveries.

03

Overall Advantages

Agile offers a more flexible and adaptive approach to project management, which can lead to better outcomes.

Iterative Learning Loops

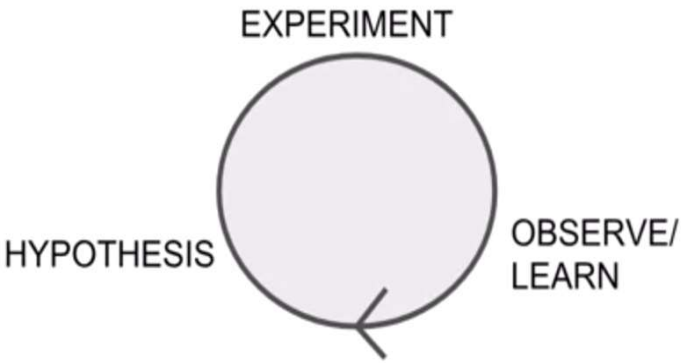


The scientific method is the foundation of many processes.

Examples:

Plan, Do, check, Act/Adjust (PDCA)

Think, Build, Ship, Tweak (Spotify)



1. Create a hypothesis
2. Build an experiment
3. Observe/learn from the results
4. Repeat/iterate

The scientific method

Agile Practices

Small Iterations

Agile involves many small iterations, allowing for frequent reassessment and adaptation of the project direction.

Continuous Feedback

Regular feedback from users and stakeholders is integrated into the development process, ensuring alignment with user needs.

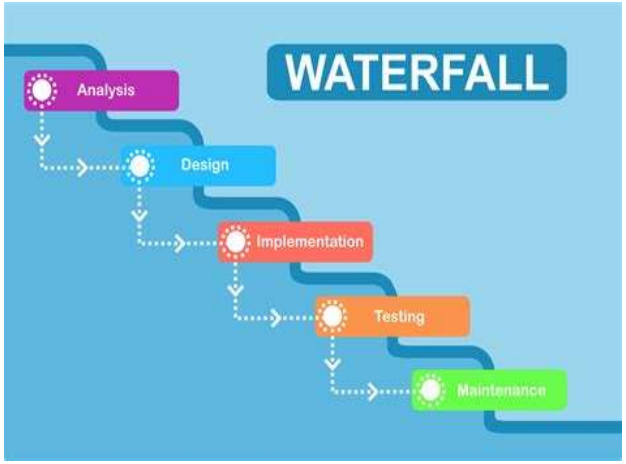
Sustainable Development

Agile promotes sustainable development practices, allowing teams to maintain a constant pace and high-quality standards.

Section 4

Agile vs. Waterfall

Traditional Waterfall Model

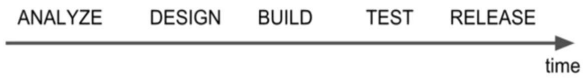


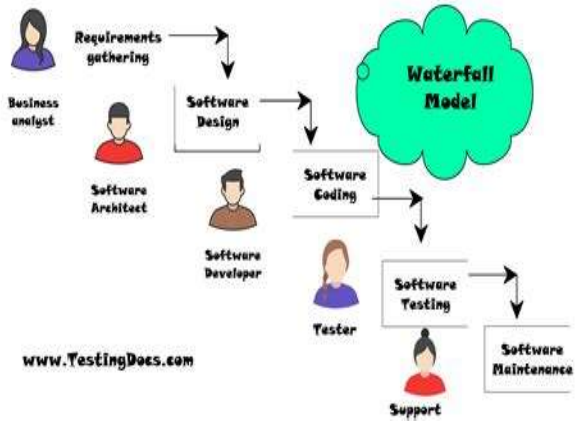
Phased Approach

The Waterfall model follows a linear and sequential approach, with distinct phases that do not overlap.

No Looping

Once a phase is completed in the Waterfall model, the process does not loop back; changes are difficult to implement. One cycle of the scientific method.

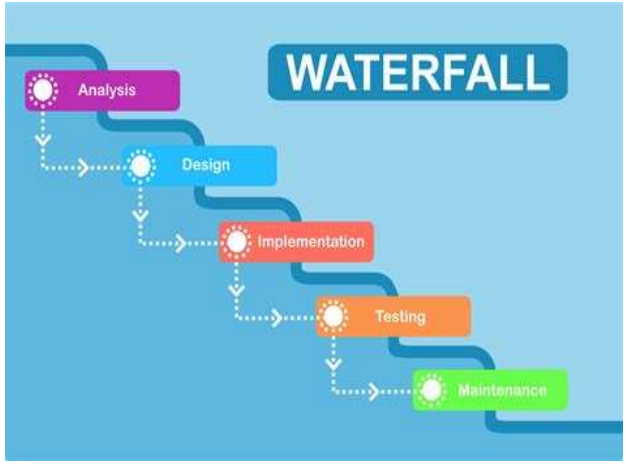




One Big Release

The Waterfall model typically involves a single, extensive planning and development phase, culminating in one big release.

Traditional Waterfall Model - Disadvantages



Your big up-front plan will be wrong because:

- You cannot predict the future
- High value features aren't correctly identified
- You may build many unnecessary features
- Things are harder, more problematic and take longer to build than you think
- The market and/or your team will change while you build
- Change is hard and expensive
- You create a lot of obsolete documents
- Feedback is drastically delayed

.

Agile's Iterative Approach

Continuous Iteration

Agile, on the other hand, involves continuous iteration, allowing for frequent reassessment and refinement.

Flexibility

Agile's iterative nature provides the flexibility to adapt to changes and incorporate feedback at any stage of the project.

Incremental Releases

Agile focuses on delivering incremental releases, providing value to customers early and often.

Documentation and Planning

01

Waterfall Documentation

Waterfall often results in extensive documentation that can become obsolete as the project evolves.

02

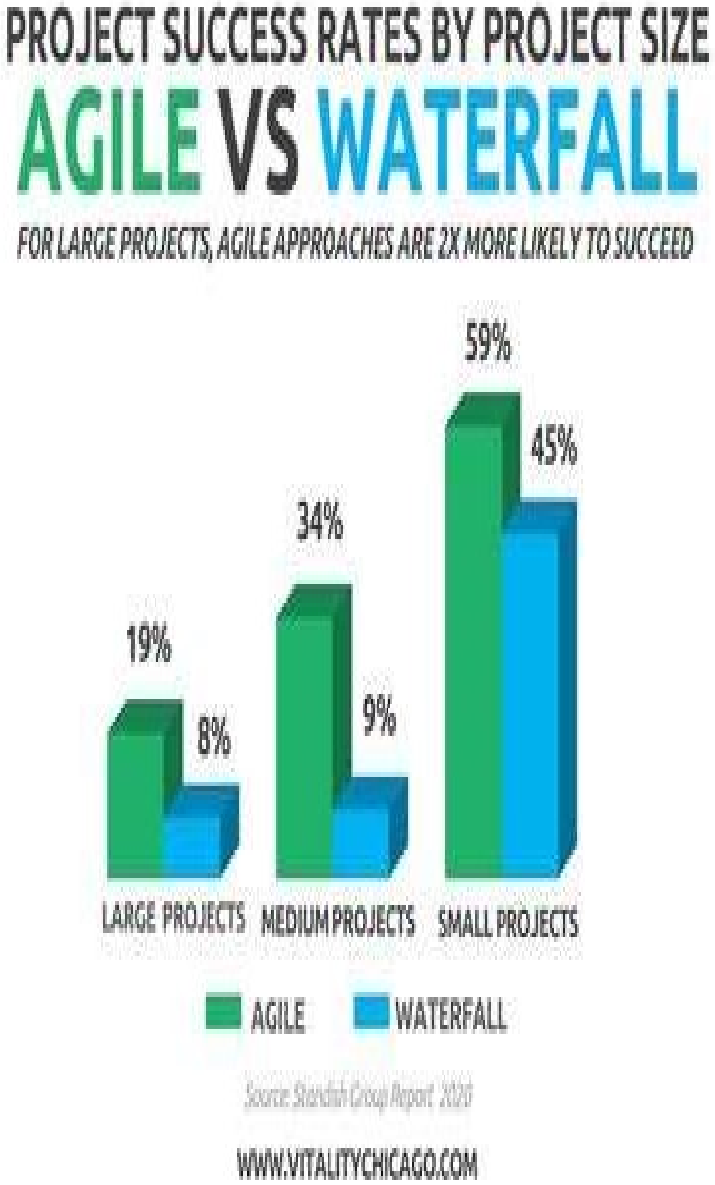
Agile Documentation

Agile emphasizes working software over comprehensive documentation, reducing the risk of obsolescence.

03

Planning in Agile

Agile involves continuous planning throughout the project, as opposed to Waterfall's single, upfront planning phase.



Comparing Outcomes

Project Management

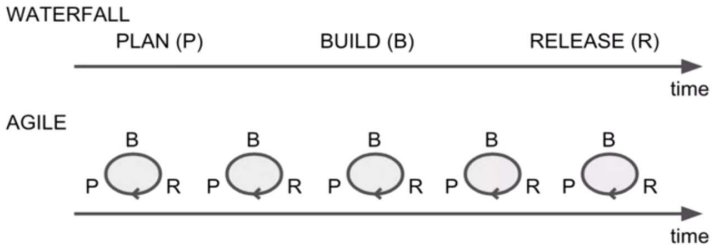
Agile's project management approach allows for more flexibility and adaptability compared to Waterfall's rigid structure.

Customer Satisfaction

Agile's focus on customer collaboration and incremental delivery often leads to higher customer satisfaction.

Risk Management

Agile's iterative nature helps in managing risks more effectively by allowing for early detection and resolution of issues.



Section 5

Agile in Practice

Agile in Different Contexts

Software Development

Agile is widely used in software development, allowing teams to respond to the rapidly changing technology landscape.

Beyond Software

Managing Service tickets

HR new hire process

Building a rocket

Agile Methodologies

Scrum

Scrum is a popular Agile framework that organizes work into time-boxed iterations called sprints, with regular reviews and retrospectives.

Kanban

Kanban focuses on visualizing work, limiting work in progress, and maximizing flow to improve efficiency.

Lean

Lean methodology emphasizes eliminating waste, optimizing value streams, and delivering value to the customer.

Agile Tools and Techniques

01

User Stories

User stories are a way to capture requirements in Agile, focusing on the user's perspective and needs.

02

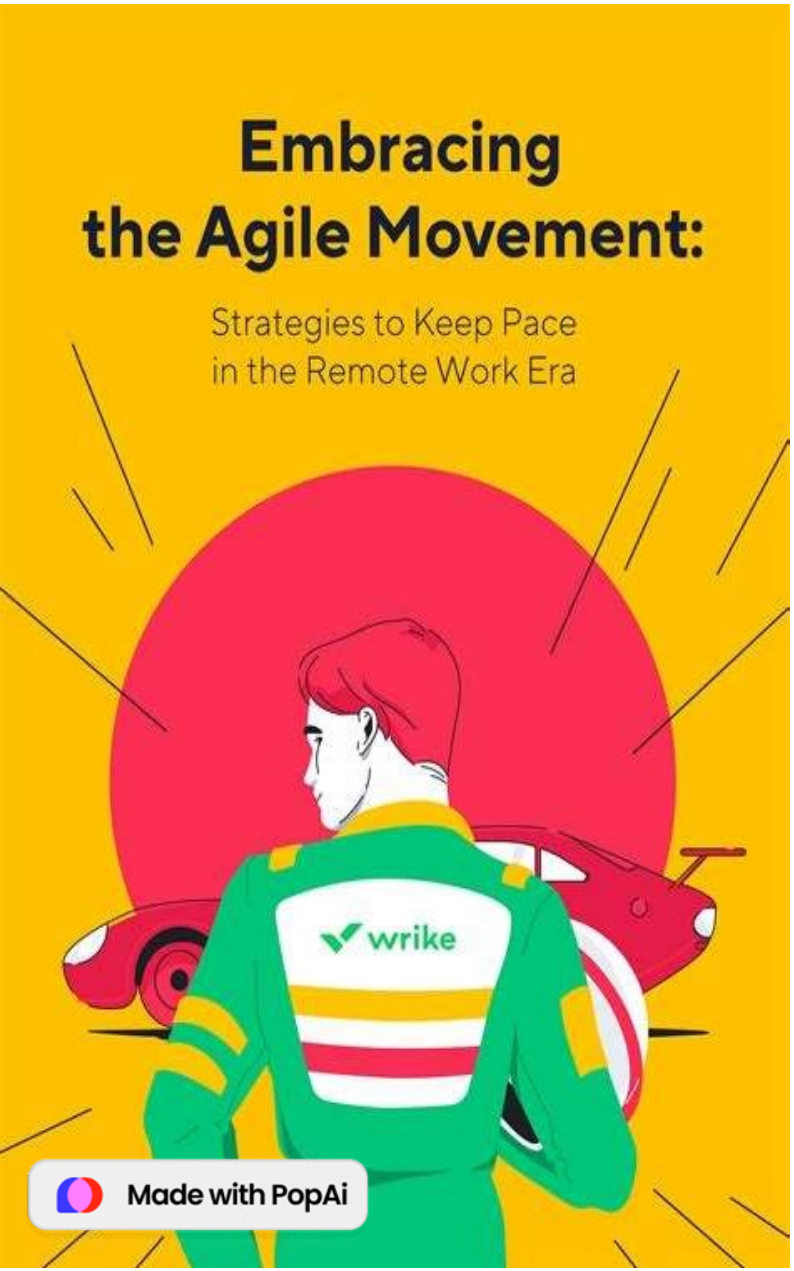
Continuous Integration/Continuous Deployment (CI/CD)

CI/CD practices enable frequent and reliable code integration and deployment, enhancing product quality.

03

Test-Driven Development (TDD)

TDD is an Agile practice where tests are written before the code, ensuring that the software is built to meet the tests' requirements.



Embracing Agile Mindset

Cultural Shift

Adopting Agile requires a cultural shift within the organization, embracing collaboration, flexibility, and continuous improvement.

Empowerment and Ownership

Agile fosters a sense of empowerment and ownership among team members, leading to more engaged and productive teams.

Continuous Learning

Agile is about continuous learning and adapting, encouraging teams to experiment, learn from failures, and strive for excellence.


Questions??