SOFTWARE ENGINEERING BEST PRACTICE

Building High-Quality Software Products

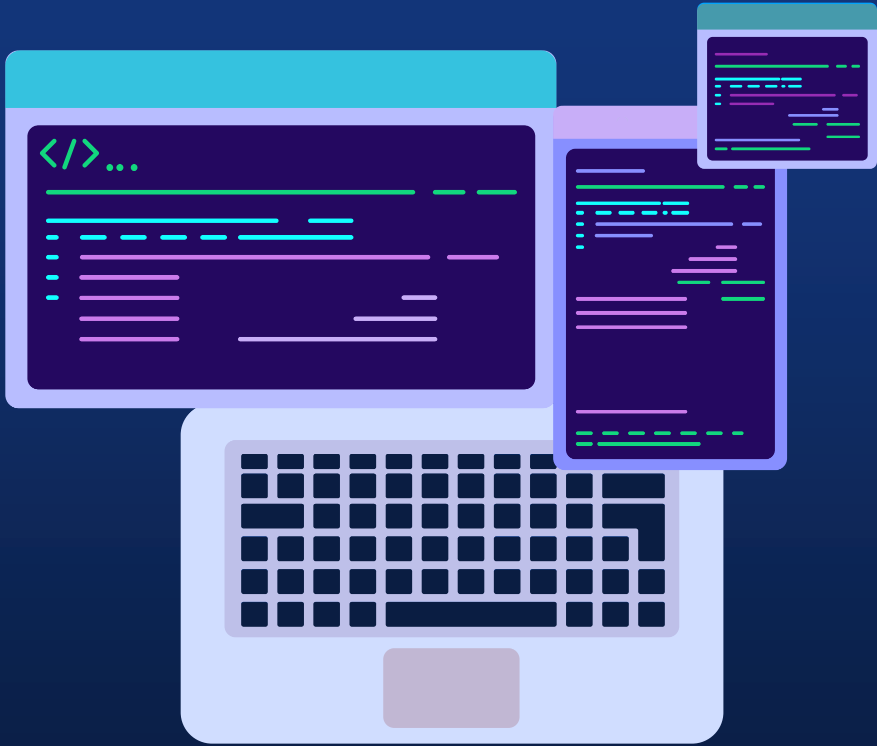


TABLE OF CONTENTS

“Everybody should learn to program a computer because it teaches you how to think.”

– Steve Jobs

Why It Matters?

- ➢ Make things work better.
- ➢ Keep things easy to fix and change.
- ➢ Stop mistakes and problems.
- ➢ Help teams work well together.
- ➢ Make customers happy.

IT’S BECOMING A BASIC NEED OF TIME!!!!

01

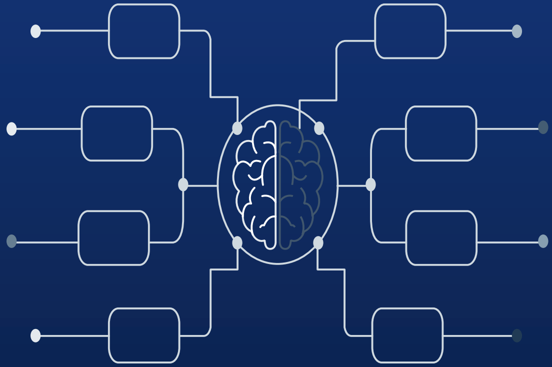
AGILE DEVELOPMENT

WHAT IS AGILE?

- ➢ Making software in small steps.
- ➢ Working closely with customers.
- ➢ Able to change things quickly

WHY AGILE?

- Adapt to customer needs.
- Deliver value sooner.
- Embrace change for better results.



AGILE METHODOLOGIES

Scrum

Short,iterative cycles called sprints,emphasizes collaboration

Kanban

Visualizes work,focuses on flow and continuous improvement

Extreme Programming

Prioritizes customer feedback, include practice like pair programming

Lean

Eliminates waste, delivers value quickly, promotes continuous learning

ROLE OF AGILE

Adapt Quickly

- Respond fast to change
- Be ready for new things

Work together

- Team up and talk a lot
- Make sure everyone is happy

Deliver Values

- Give customer useful stuff often
- Focus on what matter most

Improve Constantly

- Keep getting better
- Make things smoother over time

"Agile: Where adaptability meets excellence in software development."

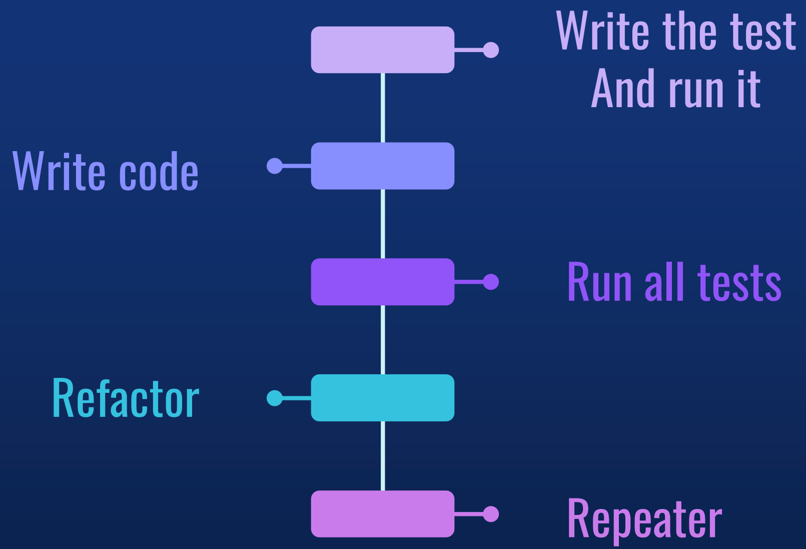
02

TEST DRIVEN DEVELOPMENT (TDD)

What is Test Driven Development

TDD (Test-Driven Development) is a software development approach where tests are written before the actual code. It promotes incremental development by breaking down the process into small, manageable steps.

PROCESS INVOLVED IN TDD:



WHY TESTING IS IMPORTANT

- Early bug detection and issue resolution
- Ensuring software compliance with requirements
- Improving software quality and reliability
- Enhancing user experience
- Building stakeholder confidence

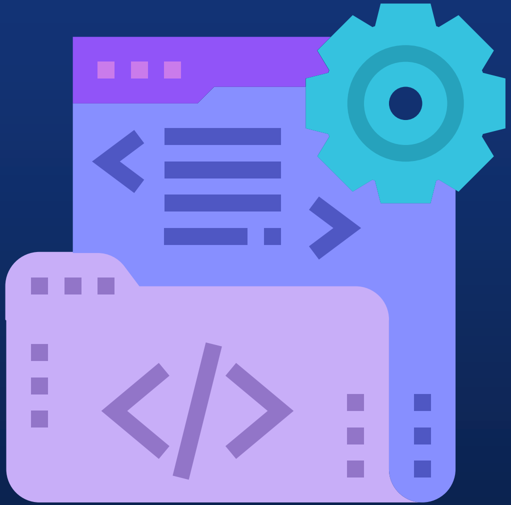
AWESOME WORDS

Examples, think of examples. Work from specific to general. But maybe not everybody is like that.
-- Kent Beck

"Write tests first, code second—Test-Driven Development, where assurance meets innovation."

03

CI & CD



Continuous Integration (CI)

Frequent Integration

Developers integrate code changes multiple times daily.

Automated Builds & Tests

Automated processes trigger builds and tests upon each integration.

Early Error Identification

Facilitates early detection and resolution of integration errors

Continuous Delivery (CD)

Automated Deployment Pipelines

Utilize automated deployment pipelines for software updates.

Potentially Releasable Code

Code changes passing through CI are potentially ready for release

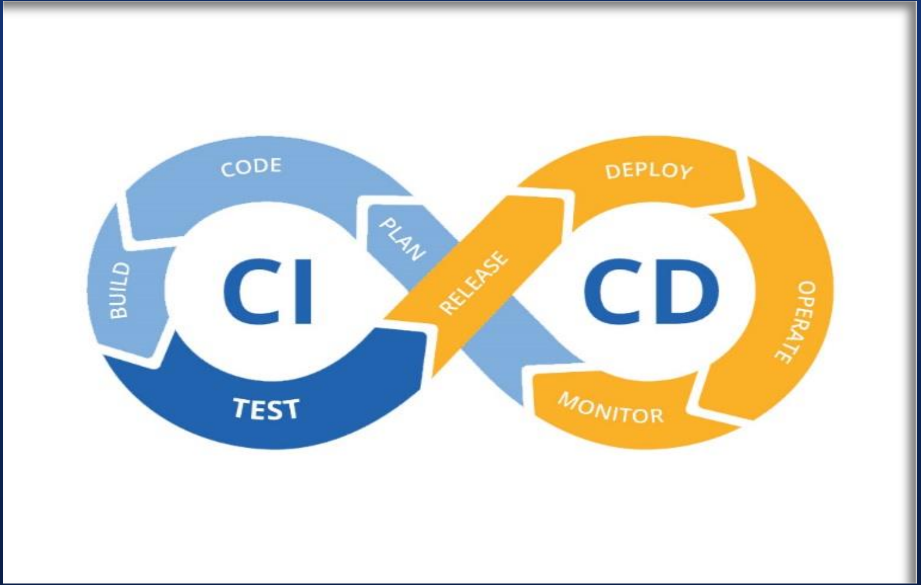
Emphasis on CD

CD prioritizes automated testing and deployment for reliable, repeatable releases

Role of CI & CD

- Early Detection of Issues
- Automated Testing
- Rapid Feedback Loop
- Consistent and Reliable Releases
- Continuous Improvement
- Enhanced Collaboration

LET’S GO THOUGH THE GRAPHIC DIAGRAM



BREAK-EVEN ANALYSIS

BREAK-EVEN POINT

It's the biggest planet in the Solar System

BIG LOSS

Mercury is the closest planet to the Sun

LOSS

Despite being red, Mars is a very cold place

PROFIT

It's composed of hydrogen and of helium

HIGH PROFIT

Neptune is the farthest planet from the Sun

"CI/CD: Code's fast track to deployment."

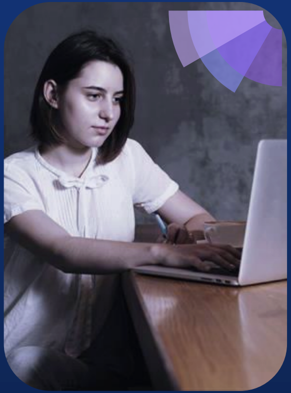
04

CODE REVIEW

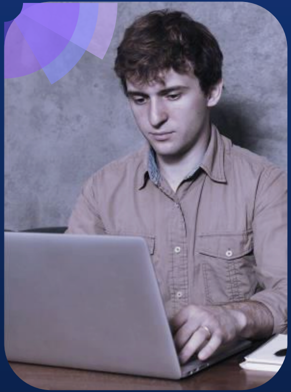
Code Reviews

Inspection of code made by one developer, by other developers.

They check for:


Buges

Logical errors


Security vulnerabilities

BENEFITS

Knowledge Sharing

Developers learn together
Fosters a collaborative learning environment

Consistency in Coding Style

Uniform style across the program
Enhances code understanding and maintenance

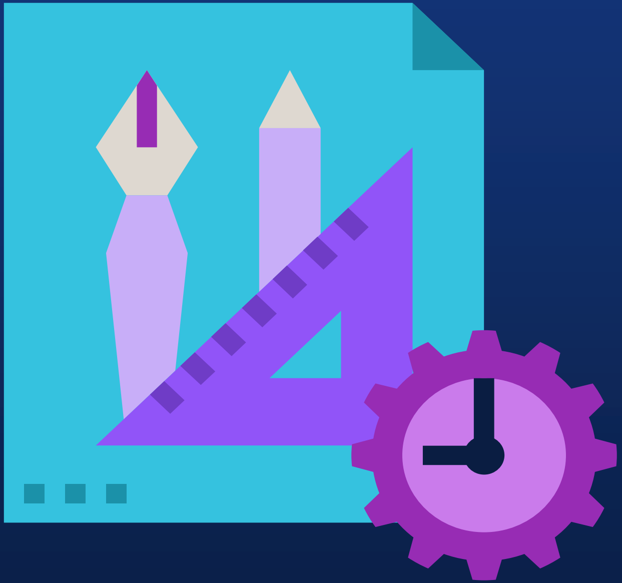
Early Issue Detection

Identify and address issues early
Reduces costs and minimizes single points of failure

" Where collaboration meets excellence, ensuring code is not just written, but crafted to perfection."

05

DESIGN PATTERNS



DESIGN PATTERNS

Reusable solutions to common problems

Blueprint for specific design issues

EXAMPLE

DESIGN PATTERN

- Singalton pattern
- Fectory Method pattern
- Observer pattern

DESIGN PRINCIPLES

- Fundamental rules that govern software design
- They ensure that software is easy to understand, well-structured, and maintainable.

EXAMPLE

DESIGN PRINCIPLES

Single Responsibility

Liskov substituotion

Don’t repeat yourself

BENEFITS OF DESIGN PATTERNS

REUSEABILITY

Save time with reusable solutions for similar challenges.

MAINTAINABILITY

Enhance readability and organization for easy maintenance.

PERFORMANCE OPTIMIZATION

Improve efficiency through optimized resource usage.

IMPROVED COLLABORATION

Foster consistency for seamless teamwork in software design

SCALABILITY

Enable adaptability and growth with a design that accommodates increased demands

"Design Patterns: Coding made smarter, not harder."

06

AUTOMATED TESTING

AUTOMATED TESTING

Automated testing means using computer programs to check if software works right. It helps make sure software is good quality without needing lots of people to test it by hand.

TYPES

Unit Tests

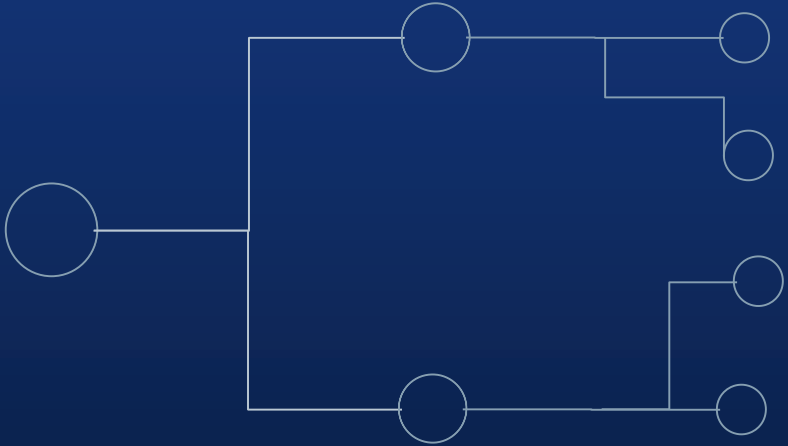
Integration Tests

Functional Tests

Regression Tests

Performance Tests

Security Tests



ADVERTISING AND PROMOTION

Quick Testing

Continuous Integration

Consistency

Continuous development

Early finding Bugs

Test Coverage

BENEFITS

Accelerated Testing Iterations

Enables faster and more frequent testing cycles

Consistent and Repeatable Testing

Provides reliable test conditions, minimizing human errors.

Early Defect Identification

Identifies defects in the development process, reducing issue-fixing costs.

BENEFITS

Enhanced Code Maintainability and Scalability

Ensures comprehensive test coverage for improved code quality and scalability.

Facilitates Continuous Integration and Delivery

Supports rapid, high-quality software delivery through continuous practices

By embracing these principles and methodologies, software development teams can build robust, reliable, and high-quality software products that meet user expectations and business objectives

"Automated Testing: Where bugs fear to hide, and quality takes the lead."

Key Takeaways

- Emphasized importance of software engineering best practices.
- Explored Agile methodologies, TDD, CI/CD, code reviews, design patterns, and automated testing.
- Highlighted their collective role in building high-quality software products.

Call to action

Implement these practices for enhanced software development.
Foster a culture of continuous improvement.

Floor is open for Questions

From,

- Syed Waleed
- Shayan Naimat
- Saniya
- Sofia
- Sanjana