CS 569
Selected Topics in Software Engineering: Program Analysis & Evaluation

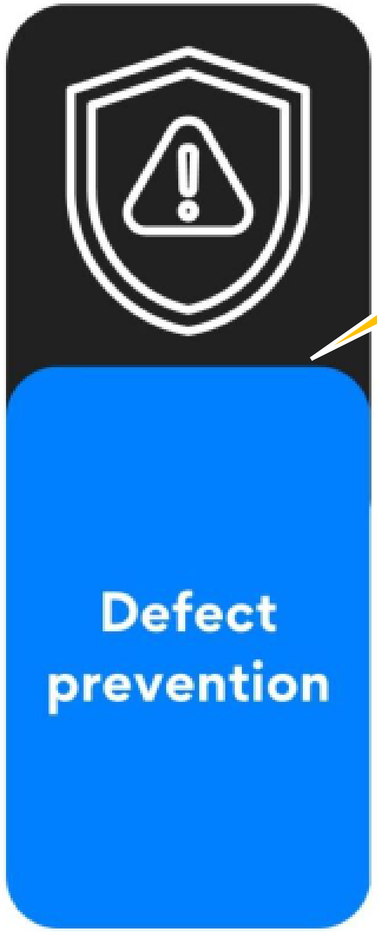
Software Quality Assurance

Oregon State University, Winter 2024

What is Software Quality Assurance ?

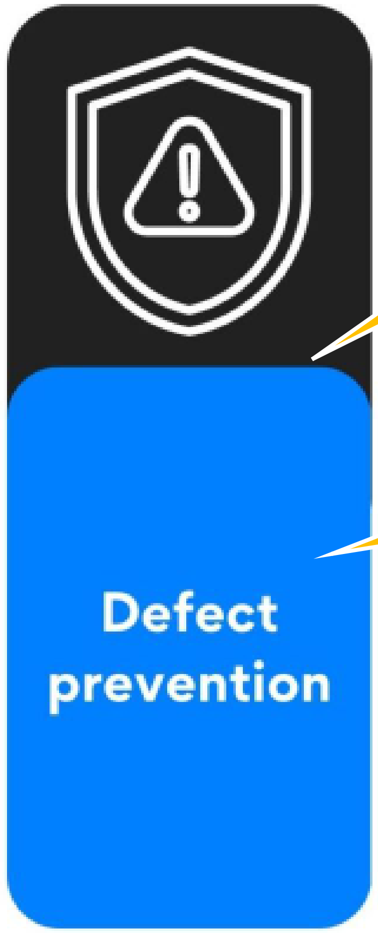
- Methodology to ensure that the quality of the software product complies with a predetermined set of standards
- Functions in parallel with the software development life cycle (SDLC)
- Tests every block of development process individually to identify issues before they become major problems
  - Externally, businesses evaluate efficiency, reliability, and cost of maintenance.
  - Internal evaluation includes structure, complexity, readability, flexibility, testability, and the coding practices developers have followed to develop the software.

Principles of Software Quality Assurance



Prevention is better than cure!

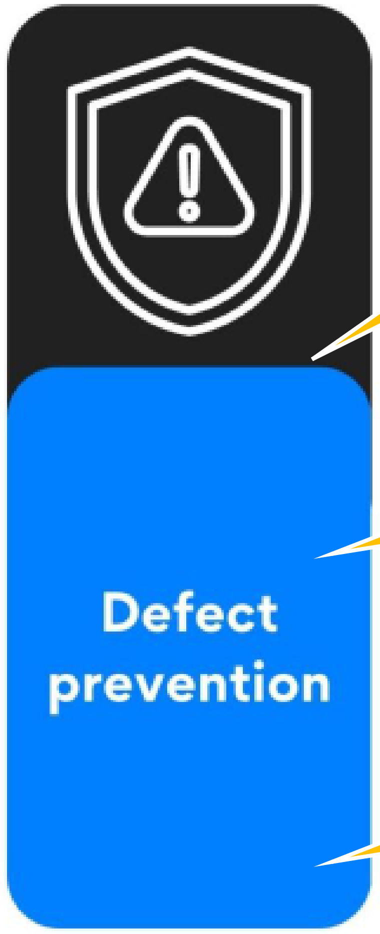
Principles of Software Quality Assurance



Prevention is better than cure!

Identify and address potential issues early in the SDLC.

Principles of Software Quality Assurance



Prevention is better than cure!

Identify and address potential issues early in the SDLC.

Focus on fixing the root cause of defects and errors, and not just the symptoms

Principles of Software Quality Assurance

Defect prevention
Continuous improvement

SQA is not a one-time thing!
Its an on-going process to be integrated in SDLC

Principles of Software Quality Assurance

Defect prevention
Continuous improvement

SQA is not a one-time thing! Its an on-going process to be integrated in SDLC

Consistently monitor and improve the quality of the software product

Principles of Software Quality Assurance

Defect prevention
Continuous improvement
Stakeholder involvement

Focus on collaboration and communication between the involved parties to ensure a smooth software development process

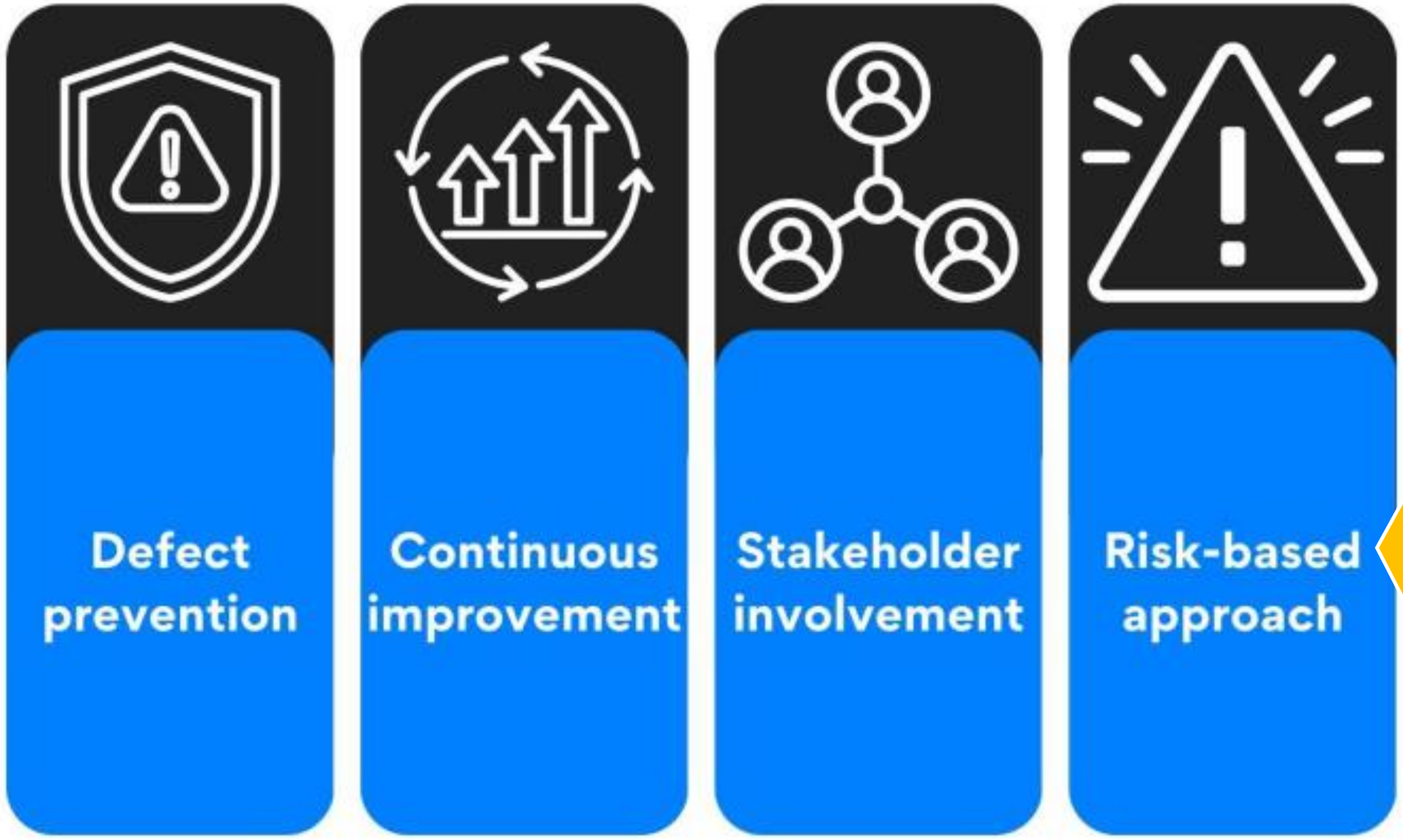
Principles of Software Quality Assurance

Defect prevention
Continuous improvement
Stakeholder involvement

Focus on collaboration and communication between the involved parties to ensure a smooth software development process

SQA must involve all stakeholders, including customers, developers, testers, QA team leads, and project managers.

Principles of Software Quality Assurance



SQA must focus on identifying and addressing the most significant risks in the software product

How to Implement Software Quality Assurance

1 Define quality standards

- Define the quality standards that software product must meet.
- Includes defining requirements, acceptance criteria, and performance metrics.
- These standards should be agreed upon by all stakeholders.

How to Implement Software Quality Assurance

1 Define quality standards

2 Plan SQA activities

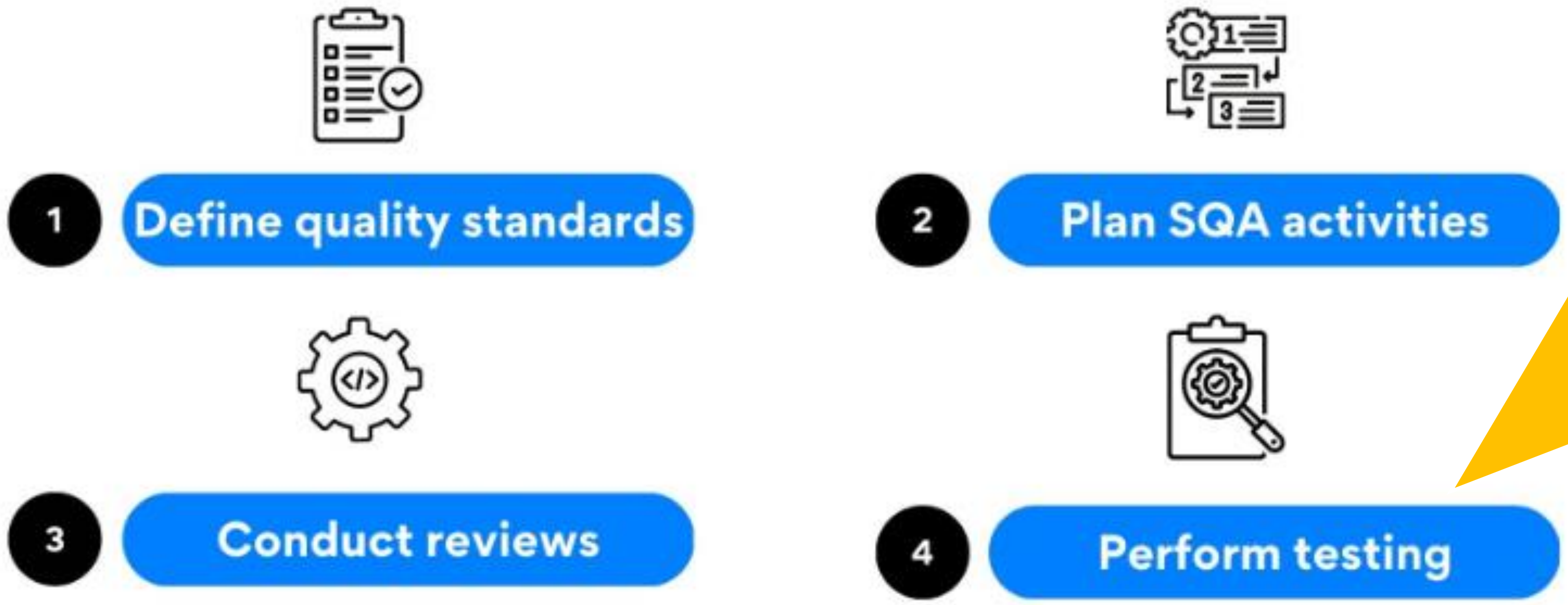
- Plan activities to be performed throughout the SDLC.
- The plan should include reviews, testing, and documentation.
- Also specify who will be responsible for each activity and when it will be performed.

How to Implement Software Quality Assurance

1. Define quality standards
2. Plan SQA activities
3. Conduct reviews

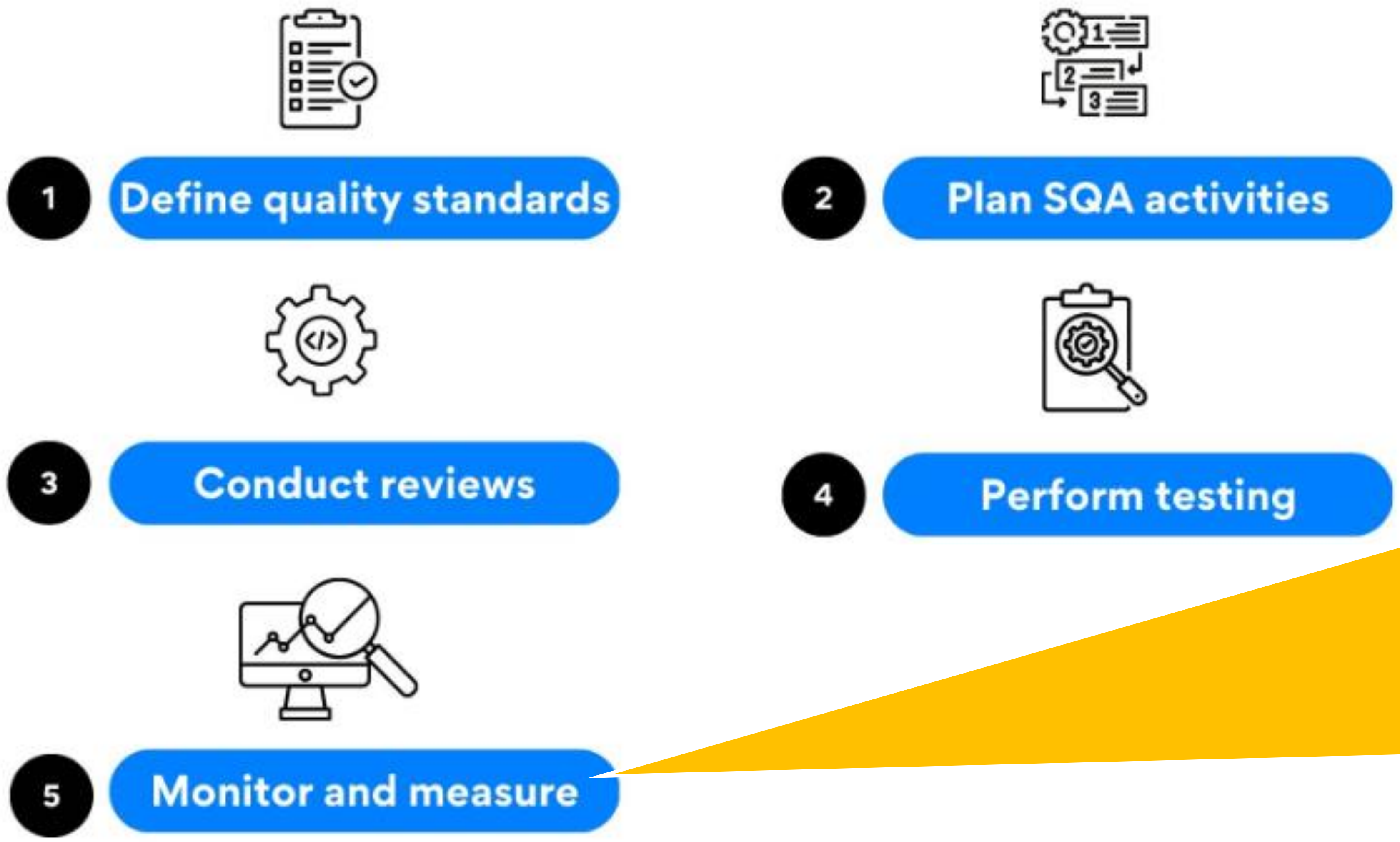
- Reviews software artifacts such as requirements, design documents, and code.
- Reviews conducted by a team of experts who are **not directly involved** in the development process.

How to Implement Software Quality Assurance



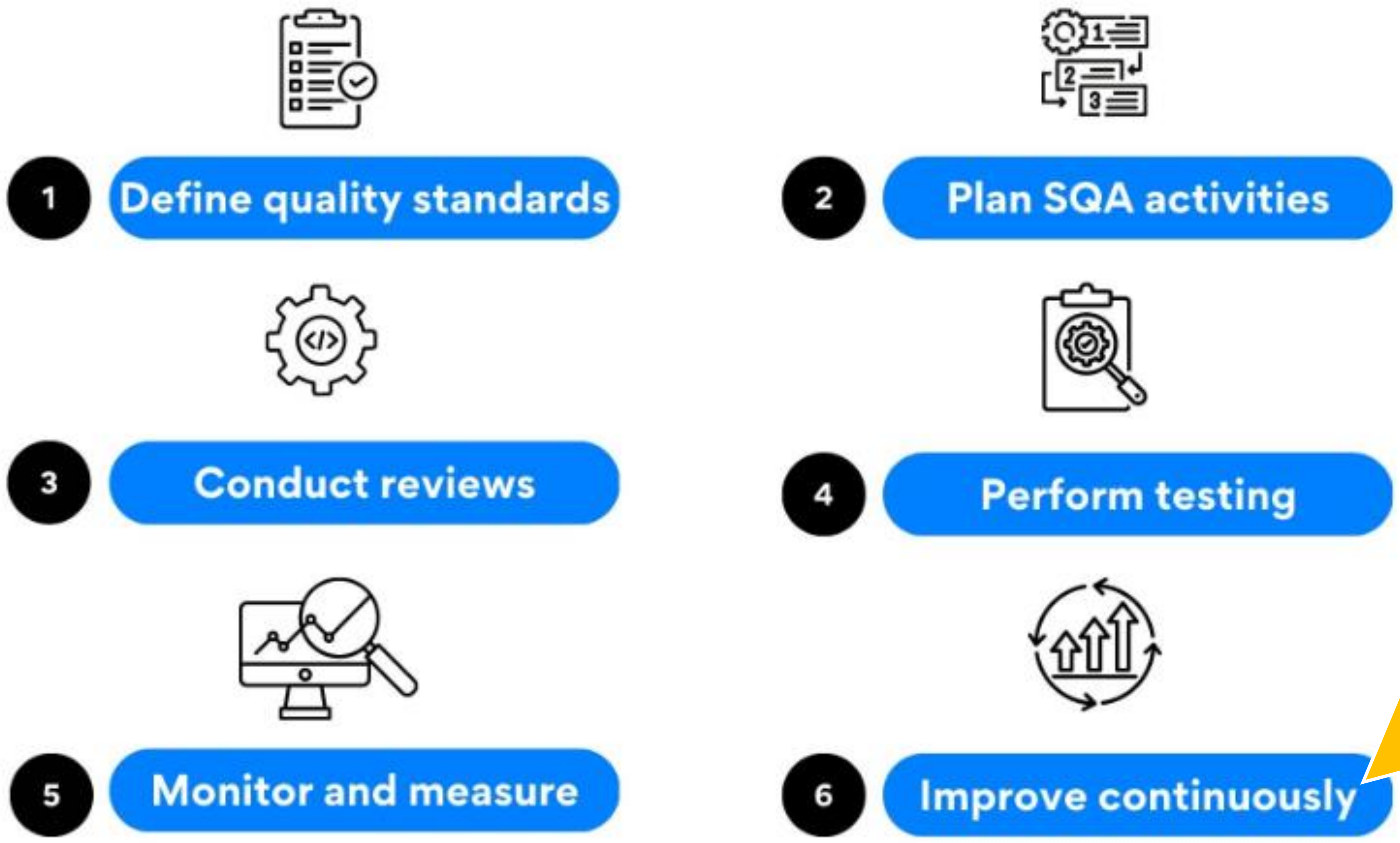
- Perform different types of testing (unit testing, integration testing, system testing, and acceptance testing).
- Use automated testing tools to increase efficiency and reduce the risk of human error.

How to Implement Software Quality Assurance



- Monitor and measure the software quality throughout the development process.
- Track defects, analyze metrics such as code coverage and defect density, and conduct root cause analysis.

How to Implement Software Quality Assurance



- Continuously improve the SQA process by analyzing the results of the monitoring and measuring activities.
- Use this data to identify areas for improvement and implement changes to the SQA process.

Software Quality Assurance Approaches

- Software quality defect management approach
  - Focuses on counting and managing defects
  - Use tools like defect leakage matrices and control charts to measure and enhance SDLC
- Software quality attributes approach
  - Focuses on directing the engineer's attention to several quality factors:
    - Reliability - system's ability to continue operating over time under different working environments and conditions
    - Usability - how easy it is to learn and navigate the application.
    - Efficiency - how well the system uses all the available resources, shown by the amount of time the system needs to finish any task
    - Maintainability - how easy it is to maintain different system versions and support changes and upgrades cost-effectively
    - Portability - system's ability to run effectively on various platforms — for example, data portability, viewing, hosting, and more

SQA Approaches in Industry

- Traditional
  - SQA is performed at the end of each phase in Waterfall SDLC model
- Agile
  - Focuses on self-organizing teams, continuous integration and testing, continuous delivery, and continuous feedback to ensure a high-quality software product.
- DevOps
  - Combination of development and IT operations
  - Emphasizes collaboration, automation, and continuous delivery to deliver the software product quickly and efficiently.
- Six Sigma
  - Data-driven approach that focuses on reducing defects and errors
  - Uses statistical tools and techniques to measure and improve the software quality

SQA Approaches in Industry (Contd.)

- Lean
  - Focuses on efficiency and waste reduction in the software development process.
  - Emphasizes continuous improvement and the elimination of non-value-added activities.
- Continuous integration and continuous deployment (CI/CD)
  - Focuses on continuous integration and deployment of software products.
  - Emphasizes automation, continuous testing, and continuous delivery of software products.
- Test-driven development (TDD)
  - Involves writing tests before writing the code to ensure that the code meets the requirements and specifications of the software product
- Risk-based approach
  - Involves risk assessment, risk mitigation, and risk monitoring to ensure that the software product meets established standards.

Why is SQA Important?

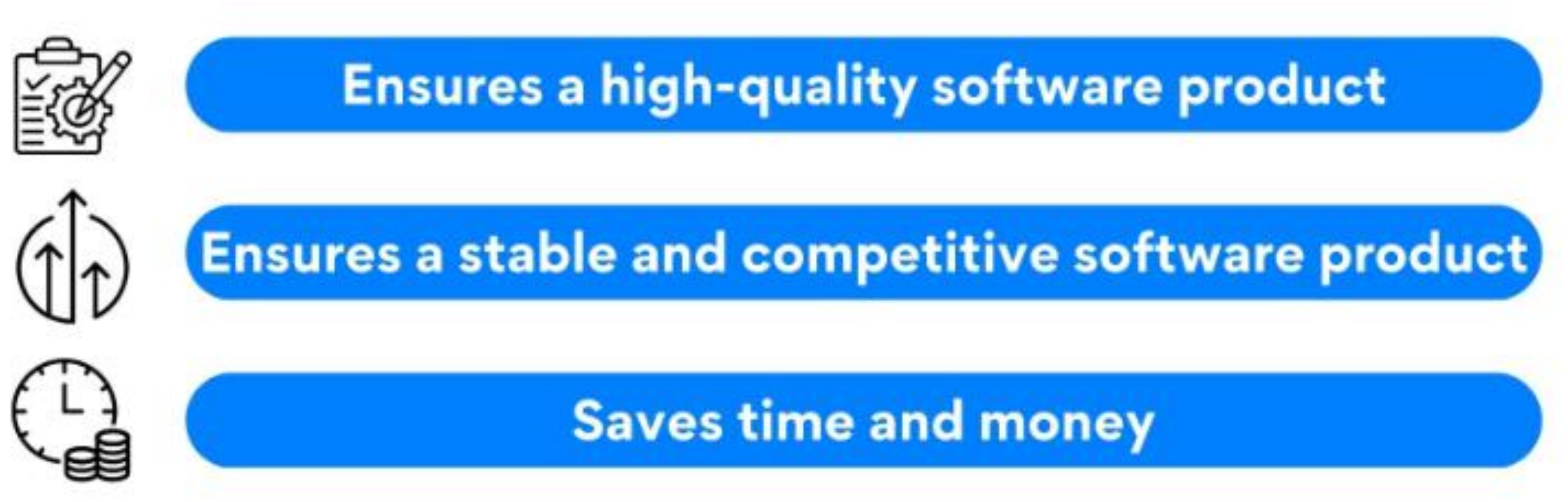
Ensures a high-quality software product

Why is SQA Important?

Ensures a high-quality software product

Ensures a stable and competitive software product

Why is SQA Important?

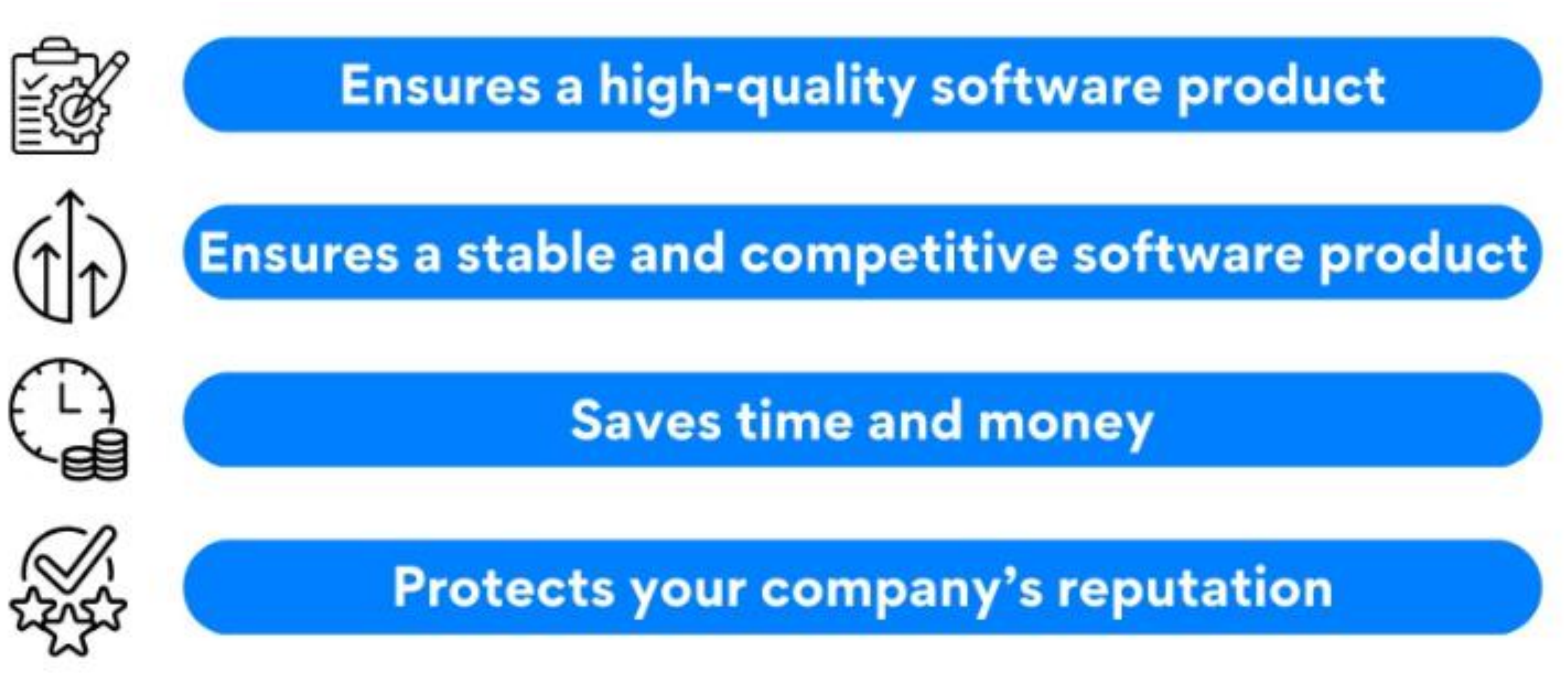


Ensures a high-quality software product

Ensures a stable and competitive software product

Saves time and money

Why is SQA Important?



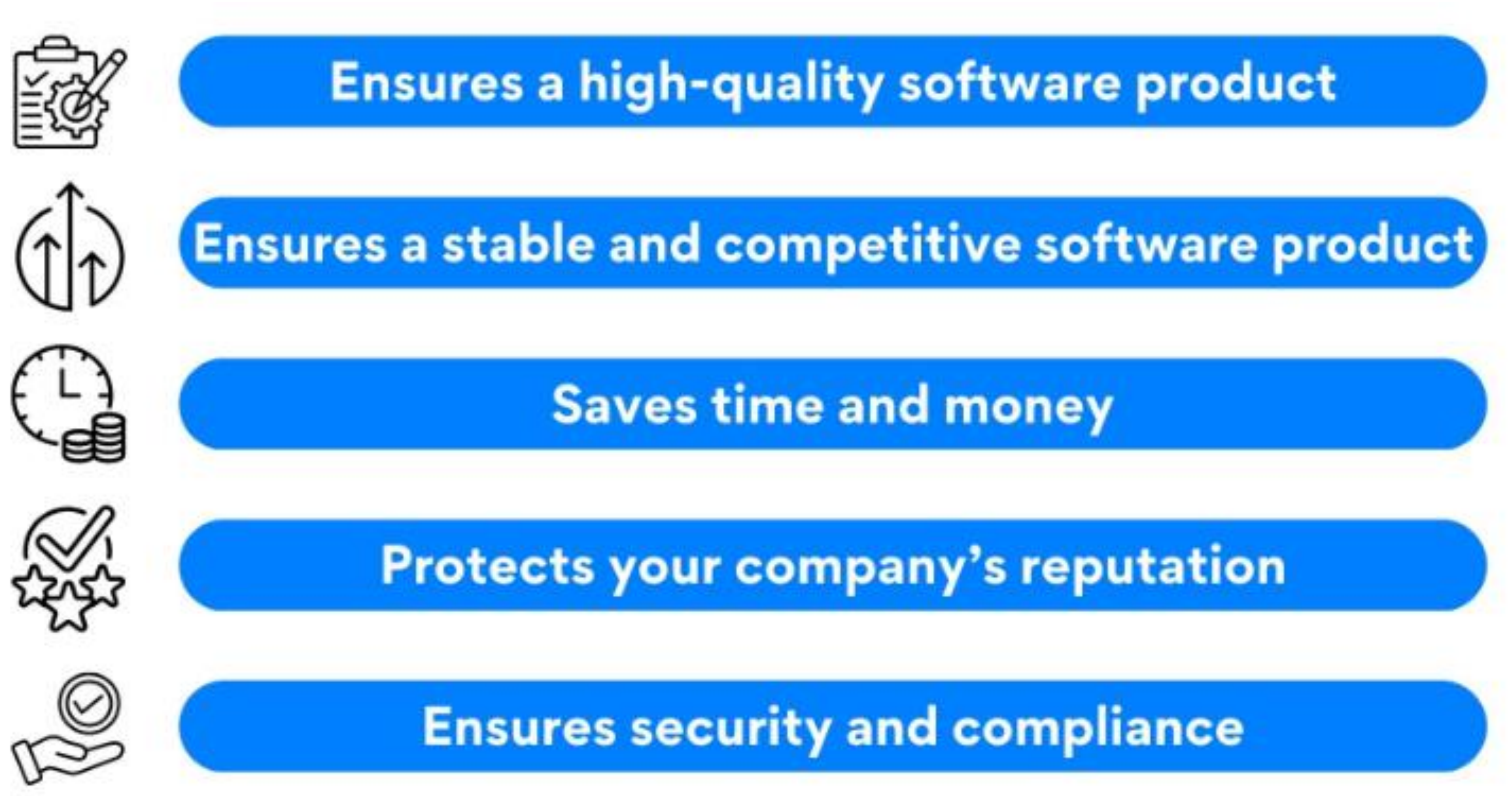
Ensures a high-quality software product

Ensures a stable and competitive software product

Saves time and money

Protects your company’s reputation

Why is SQA Important?



Ensures a high-quality software product

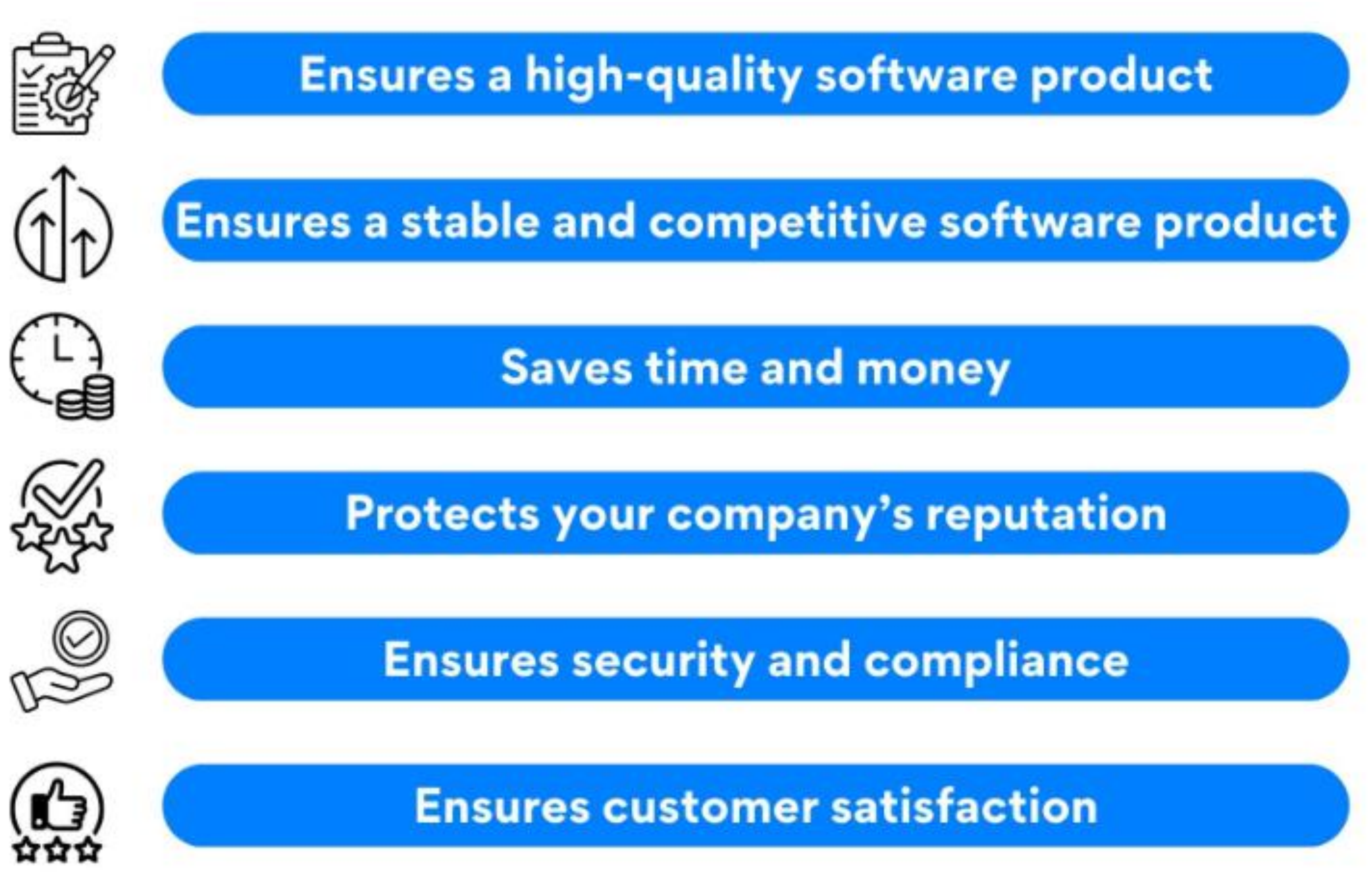
Ensures a stable and competitive software product

Saves time and money

Protects your company’s reputation

Ensures security and compliance

Why is SQA Important?

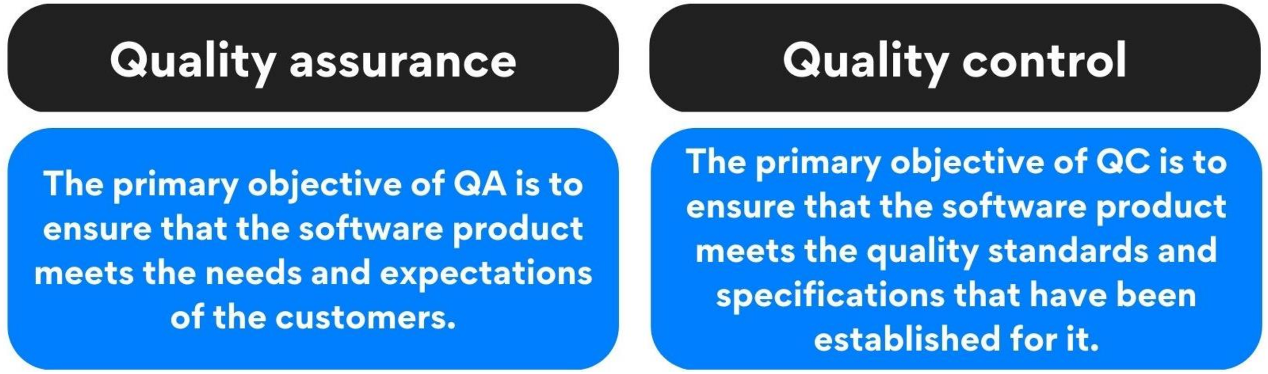


- Ensures a high-quality software product
- Ensures a stable and competitive software product
- Saves time and money
- Protects your company's reputation
- Ensures security and compliance
- Ensures customer satisfaction

Quality Assurance vs. Quality Control

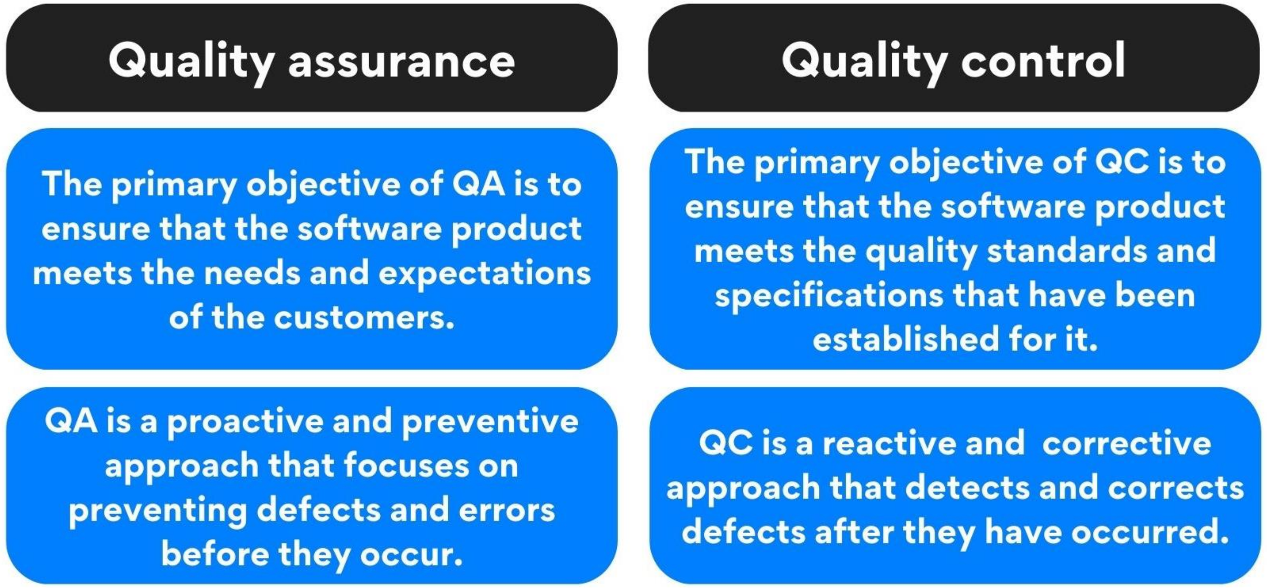
Quality assurance **Quality control**

Quality Assurance vs. Quality Control



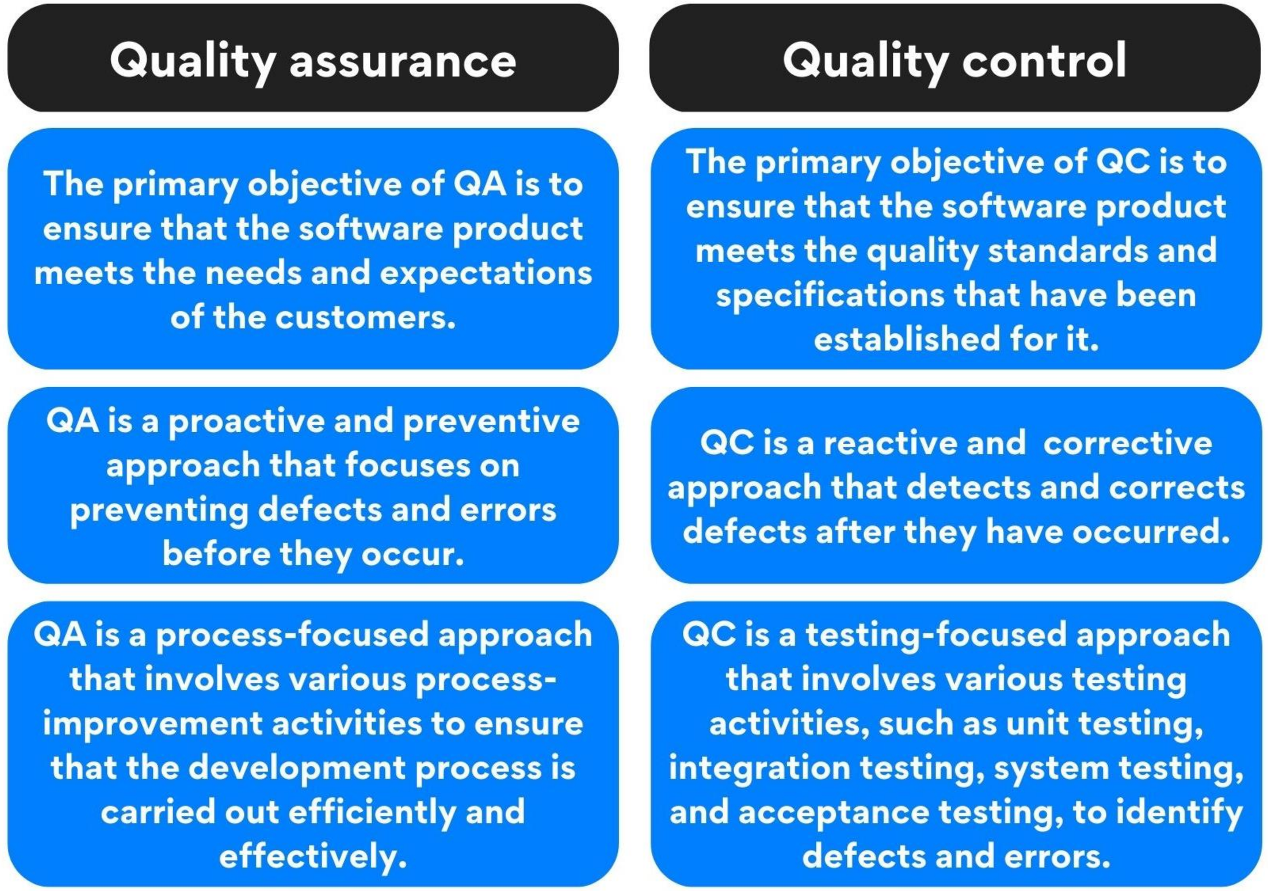
Quality assurance	Quality control
The primary objective of QA is to ensure that the software product meets the needs and expectations of the customers.	The primary objective of QC is to ensure that the software product meets the quality standards and specifications that have been established for it.

Quality Assurance vs. Quality Control



Quality assurance	Quality control
The primary objective of QA is to ensure that the software product meets the needs and expectations of the customers.	The primary objective of QC is to ensure that the software product meets the quality standards and specifications that have been established for it.
QA is a proactive and preventive approach that focuses on preventing defects and errors before they occur.	QC is a reactive and corrective approach that detects and corrects defects after they have occurred.

Quality Assurance vs. Quality Control



Quality assurance	Quality control
The primary objective of QA is to ensure that the software product meets the needs and expectations of the customers.	The primary objective of QC is to ensure that the software product meets the quality standards and specifications that have been established for it.
QA is a proactive and preventive approach that focuses on preventing defects and errors before they occur.	QC is a reactive and corrective approach that detects and corrects defects after they have occurred.
QA is a process-focused approach that involves various process-improvement activities to ensure that the development process is carried out efficiently and effectively.	QC is a testing-focused approach that involves various testing activities, such as unit testing, integration testing, system testing, and acceptance testing, to identify defects and errors.

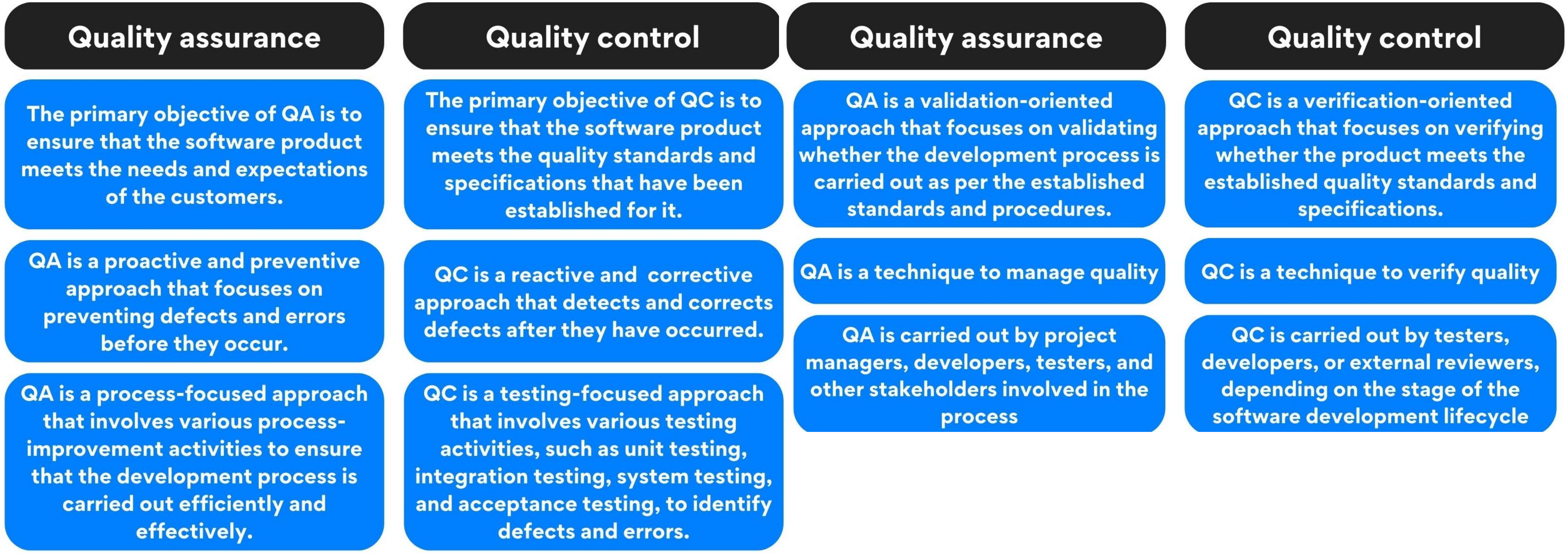
Quality Assurance vs. Quality Control

Quality assurance	Quality control	Quality assurance	Quality control
The primary objective of QA is to ensure that the software product meets the needs and expectations of the customers.	The primary objective of QC is to ensure that the software product meets the quality standards and specifications that have been established for it.	QA is a validation-oriented approach that focuses on validating whether the development process is carried out as per the established standards and procedures.	QC is a verification-oriented approach that focuses on verifying whether the product meets the established quality standards and specifications.
QA is a proactive and preventive approach that focuses on preventing defects and errors before they occur.	QC is a reactive and corrective approach that detects and corrects defects after they have occurred.		
QA is a process-focused approach that involves various process-improvement activities to ensure that the development process is carried out efficiently and effectively.	QC is a testing-focused approach that involves various testing activities, such as unit testing, integration testing, system testing, and acceptance testing, to identify defects and errors.		

Quality Assurance vs. Quality Control

Quality assurance	Quality control	Quality assurance	Quality control
The primary objective of QA is to ensure that the software product meets the needs and expectations of the customers.	The primary objective of QC is to ensure that the software product meets the quality standards and specifications that have been established for it.	QA is a validation-oriented approach that focuses on validating whether the development process is carried out as per the established standards and procedures.	QC is a verification-oriented approach that focuses on verifying whether the product meets the established quality standards and specifications.
QA is a proactive and preventive approach that focuses on preventing defects and errors before they occur.	QC is a reactive and corrective approach that detects and corrects defects after they have occurred.	QA is a technique to manage quality	QC is a technique to verify quality
QA is a process-focused approach that involves various process-improvement activities to ensure that the development process is carried out efficiently and effectively.	QC is a testing-focused approach that involves various testing activities, such as unit testing, integration testing, system testing, and acceptance testing, to identify defects and errors.		

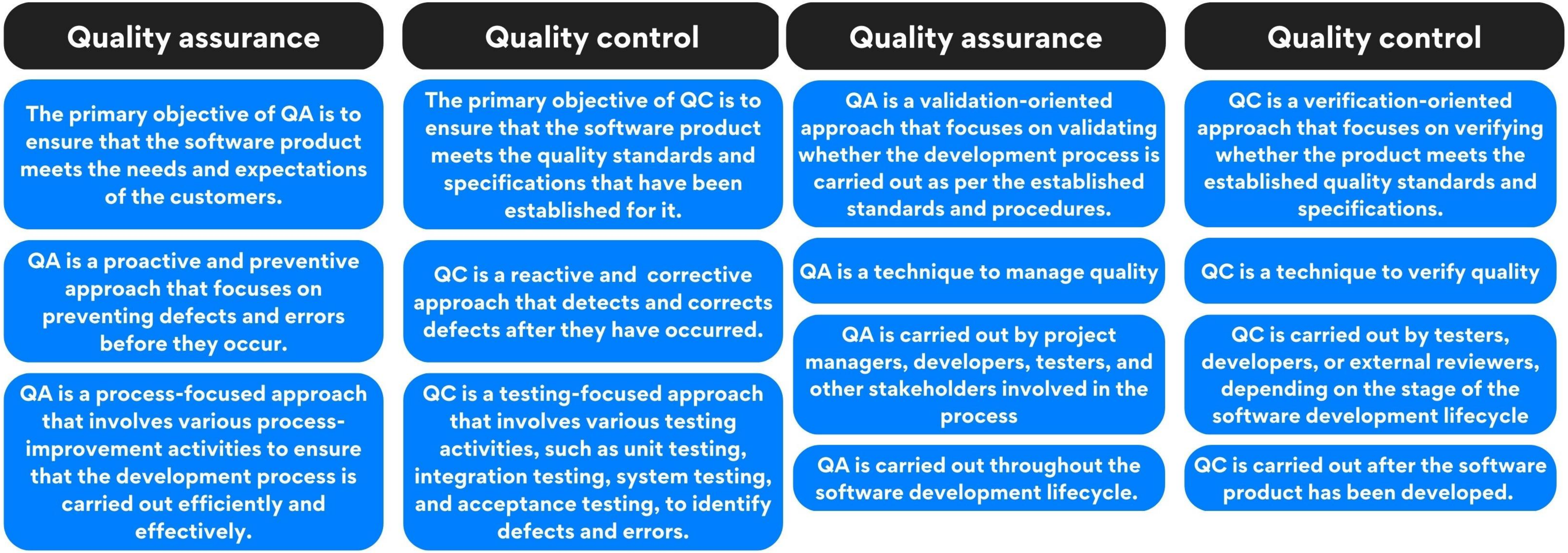
Quality Assurance vs. Quality Control



Quality assurance	Quality control
The primary objective of QA is to ensure that the software product meets the needs and expectations of the customers.	The primary objective of QC is to ensure that the software product meets the quality standards and specifications that have been established for it.
QA is a proactive and preventive approach that focuses on preventing defects and errors before they occur.	QC is a reactive and corrective approach that detects and corrects defects after they have occurred.
QA is a process-focused approach that involves various process-improvement activities to ensure that the development process is carried out efficiently and effectively.	QC is a testing-focused approach that involves various testing activities, such as unit testing, integration testing, system testing, and acceptance testing, to identify defects and errors.

Quality assurance	Quality control
QA is a validation-oriented approach that focuses on validating whether the development process is carried out as per the established standards and procedures.	QC is a verification-oriented approach that focuses on verifying whether the product meets the established quality standards and specifications.
QA is a technique to manage quality	QC is a technique to verify quality
QA is carried out by project managers, developers, testers, and other stakeholders involved in the process	QC is carried out by testers, developers, or external reviewers, depending on the stage of the software development lifecycle

Quality Assurance vs. Quality Control

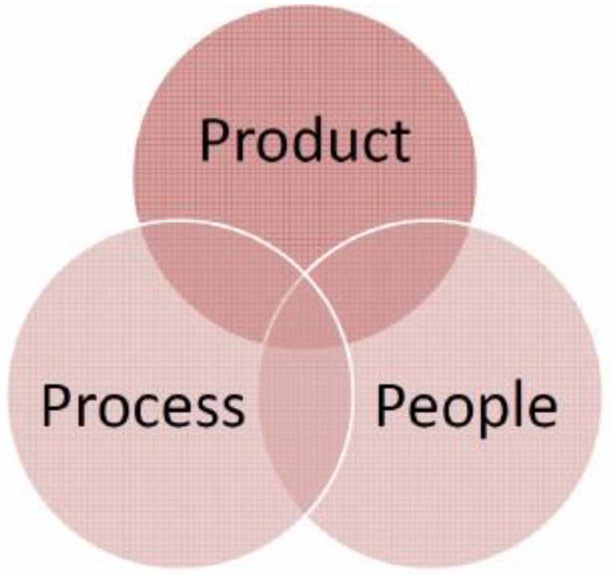


Quality assurance	Quality control
The primary objective of QA is to ensure that the software product meets the needs and expectations of the customers.	The primary objective of QC is to ensure that the software product meets the quality standards and specifications that have been established for it.
QA is a proactive and preventive approach that focuses on preventing defects and errors before they occur.	QC is a reactive and corrective approach that detects and corrects defects after they have occurred.
QA is a process-focused approach that involves various process-improvement activities to ensure that the development process is carried out efficiently and effectively.	QC is a testing-focused approach that involves various testing activities, such as unit testing, integration testing, system testing, and acceptance testing, to identify defects and errors.
QA is a validation-oriented approach that focuses on validating whether the development process is carried out as per the established standards and procedures.	QC is a verification-oriented approach that focuses on verifying whether the product meets the established quality standards and specifications.
QA is a technique to manage quality	QC is a technique to verify quality
QA is carried out by project managers, developers, testers, and other stakeholders involved in the process	QC is carried out by testers, developers, or external reviewers, depending on the stage of the software development lifecycle
QA is carried out throughout the software development lifecycle.	QC is carried out after the software product has been developed.

SQA Summary

It is difficult to assure the quality of a software product

- **What are we assuring?**
  - Building the right system? Building the system right?
  - Presence/absence of good/bad properties?
  - Identifying errors? Confidence in the absence of errors?
  - Robust? Safe? Secure? Available? Reliable?
  - Understandable? Modifiable? Cost-effective? Usable? …
- **Why are we assuring it?**
  - Business/Economic reasons, Legal, Ethical, Social, Personal satisfaction, ...
- **How do we assure it?**
  - How do we know we have assured it?



Large Code Challenges

- How to ensure
  - Maintainable code?
  - DRY code?
  - Readable code?
  - Bug-free code?
- Average defect detection rate for various testing
  - Unit testing: 25%
  - Function testing: 35%
  - Integration testing: 45%
- How can this be improved?

Code Reviews

- **Code review:** A constructive review of a fellow developer's code. A required sign-off from another team member before a developer is permitted to check-in changes or new code.

- Analogy: writing articles for a newspaper
  - What is the effectiveness of…
    - Spell-check/grammar check?
    - Author editing own article?
    - Others editing others' articles?

Code Review Mechanics

- **Who:** Original developer and reviewer, sometimes together in person, sometimes offline.
- **What:** Reviewer gives suggestions for improvement on a logical and/or structural level, to conform to previously agreed upon set of quality standards.
  - Feedback leads to refactoring, followed by a 2nd code review.
  - Eventually reviewer approves code.
- **When:** When code author has finished a coherent system change that is otherwise ready for check-in
  - change shouldn't be too large or too small
  - before committing the code to the repository or incorporating it into the new build

Why Do Code Reviews?

- > 1 person has seen every piece of code
    - Prospect of someone reviewing your code raises quality threshold.
- Forces code authors to articulate their decisions
- Hands-on learning experience for novices without hurting code quality
    - Pairing them up with experienced developers
- Team members involved in different parts of the system
    - Reduces redundancy, enhances overall understanding
- Author and reviewer both accountable for committing code

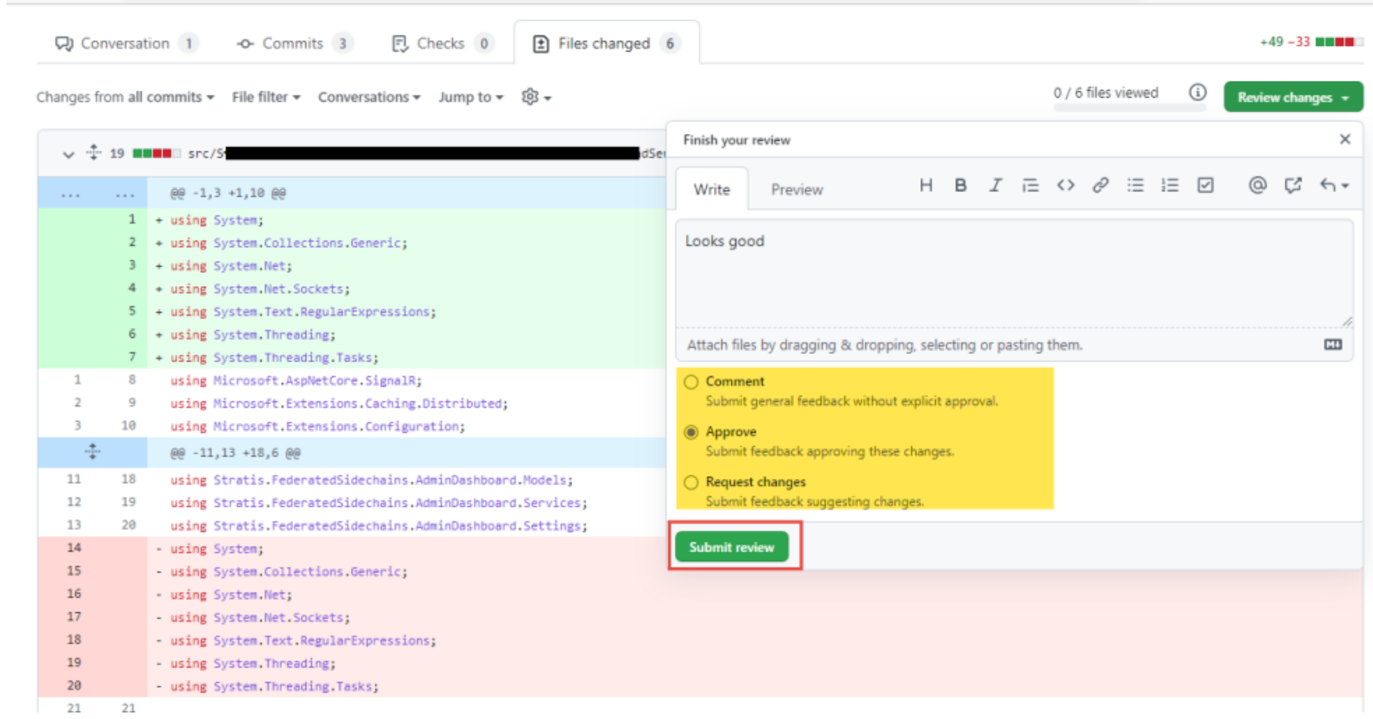
Why Do Code Reviews? Actual Studies

- Average defect detection rate for various testing
  - Unit testing: 25%
  - Function testing: 35%
  - Integration testing: 45%
  - **Design and code inspections: 55% and 60%**
- 11 programs developed by the same group of people
  - First 5 without reviews: average 4.5 errors per 100 lines of code
  - Next 6 with reviews: average 0.82 errors per 100 lines of code
  - **Errors reduced by > 80 percent**.

Source: Steve McConnell's *Code Complete*

Code Reviews in Industry

- Code reviews are a ***very common*** industry practice.
- Made easier by advanced tools that:
  - integrate with configuration management systems
  - highlight changes (i.e., diff function)
  - allow traversing back into history
  - E.g.: Eclipse, Git/SVN tools



19 src/S[redacted]dSe

```
@@ -1,3 +1,10 @@
+ using System;
+ using System.Collections.Generic;
+ using System.Net;
+ using System.Net.Sockets;
+ using System.Text.RegularExpressions;
+ using System.Threading;
+ using System.Threading.Tasks;
  using Microsoft.AspNetCore.SignalR;
  using Microsoft.Extensions.Caching.Distributed;
  using Microsoft.Extensions.Configuration;
@@ -11,13 +18,6 @@
  using Stratis.FederatedSidechains.AdminDashboard.Models;
  using Stratis.FederatedSidechains.AdminDashboard.Services;
  using Stratis.FederatedSidechains.AdminDashboard.Settings;
- using System;
- using System.Collections.Generic;
- using System.Net;
- using System.Net.Sockets;
- using System.Text.RegularExpressions;
- using System.Threading;
- using System.Threading.Tasks;
```

Finish your review

Looks good

Attach files by dragging & dropping, selecting or pasting them.

- ○ Comment — Submit general feedback without explicit approval.
- ● Approve — Submit feedback approving these changes.
- ○ Request changes — Submit feedback suggesting changes.

https://www.c-sharpcorner.com/blogs/how-to-give-code-review-feedback-in-github

Code Review Variations

- **Inspection:** A more formalized code review with:
  - roles (moderator, author, reviewer, scribe, etc.)
  - several reviewers looking at the same piece of code
  - a specific checklist of kinds of flaws to look for
  - possibly focusing on flaws that have been seen previously
  - possibly focusing on high-risk areas such as security
  - specific expected outcomes (e.g. report, list of defects)
- **Walkthrough:** informal discussion of code between author and a single reviewer
- **Code reading:** Reviewers look at code by themselves (possibly with no actual meeting)

Code Reviews at Google

- "All code that gets submitted needs to be reviewed by at least one other person, and either the code writer or the reviewer needs to have readability in that language."
- "Most people use Mondrian to do code reviews, and obviously, we spend a good chunk of our time reviewing code."

– Amanda Camp, Software Engineer, Google

Code Reviews at Yelp

- "At Yelp we use review-board. An engineer works on a branch and commits the code to their own branch. The reviewer then goes through the diff, adds inline comments on review board and sends them back."
- "The reviews are meant to be a dialogue, so typically comment threads result from the feedback. Once the reviewer's questions and concerns are all addressed, they'll click "Ship It!" and the author will merge it with the main branch for deployment the same day."

– Alan Fineberg, Software Engineer, Yelp

Code Reviews at Meta (Facebook)

- "At Facebook, we have an internally-developed web-based tool to aid the code review process. Once an engineer has prepared a change, she submits it to this tool, which will notify the person or people she has asked to review the change, along with others that may be interested in the change – such as people who have worked on a function that got changed. At this point, the reviewers can make comments, ask questions, request changes, or accept the changes. If changes are requested, the submitter must submit a new version of the change to be reviewed. All versions submitted are retained, so reviewers can compare the change to the original, or just changes from the last version they reviewed. Once a change has been submitted, the engineer can merge her change into the main source tree for deployment to the site during the next weekly push, or earlier if the change warrants quicker release."

- – Ryan McElroy, Software Engineer, Meta

Counter Argument 1

“I don't like doing a code review of every commit. It takes too long. I want to make lots of commits and it slows me down.”

- Maybe you are making **too many** commits.
- Commit **coherent, complete changes** to the code base, not incomplete work or tiny changes.
- If something was quick to write, or a small change, it will also be quick to code review.
- If you check-in buggy code, you'll lose more time than it would have taken to code review it and (hopefully) find the bug.

Counter Argument 2

"We tried doing code reviews, but it was not useful. We never find any bugs or errors; the code is always approved."

- Maybe the reviewer should be more thorough. Almost all reviews of non-trivial code uncover at least *some* issues to look into.
- Sometimes the point is not to find flaws and fix them, but also to make sure that another developer is familiar with that code.
- The fact that the original coder knew it would be reviewed increases the chance that they wrote better code to start with.

QUIZ: Code Reviews

What changes (if any) would you suggest in the following code?

```
public class Account {
  double principal,rate; int daysActive,accountType;
  public static final int STANDARD = 0, BUDGET=1,
      PREMIUM=2, PREMIUM_PLUS = 3;
  } ...

  public static double calculateFee(Account[] accounts)
  {
      double totalFee = 0.0;
      Account account;
      for (int i=0;i<accounts.length;i++) {
          account=accounts[i];
          if ( account.accountType == Account.PREMIUM ||
          account.accountType == Account.PREMIUM_PLUS )
            totalFee += .0125 * (   // 1.25% broker's fee
            account.principal * Math.pow(account.rate,
            (account.daysActive/365.25))
            - account.principal);   // interest-principal
      }
      return totalFee;
   }
}
```

QUIZ: Code Reviews

What changes (if any) would you suggest in the following code?

```
public class Account {
  double principal,rate; int daysActive,accountType;
  public static final int STANDARD = 0, BUDGET=1,
      PREMIUM=2, PREMIUM_PLUS = 3;
  } ...

  public static double calculateFee(Account[] accounts)
  {
      double totalFee = 0.0;
      Account account;
      for (int i=0;i<accounts.length;i++) {
          account=accounts[i];
          if ( account.accountType == Account.PREMIUM ||
          account.accountType == Account.PREMIUM_PLUS )
            totalFee += .0125 * (   // 1.25% broker's fee
            account.principal * Math.pow(account.rate,
            (account.daysActive/365.25))
            - account.principal);   // interest-principal
      }
      return totalFee;
   }
}
```

- Add comments
- Make fields private.
- Replace magic values (e.g., 365.25) with constants.
- Use an ***enum*** for account types.
- Use consistent whitespace, line breaks, etc.

QUIZ: Code Reviews

Improved Code

```java
/** An individual account. Also see CorporateAccount. */
public class Account {
    /** The varieties of account our bank offers. */
    public enum Type {STANDARD, BUDGET, PREMIUM, PREMIUM_PLUS}

    /** The portion of the interest that goes to the broker. */
    public static final double BROKER_FEE_PERCENT = 0.0125;

    private Type type;
    private double principal;

    /** The yearly, compounded rate (at 365.25 days per year). */
    private double rate;

    /** Days since last interest payout. */
    private int daysActive;

    /** Compute interest on this account. */
    public double interest() {
        double years = daysActive / 365.25;
        double compoundInterest = principal * Math.pow(rate, years);
        return compoundInterest - principal;
    }
    ...
```

- Add comments
- Make fields private.
- Replace magic values (e.g., 365.25) with constants.
- Use an ***enum*** for account types.
- Use consistent whitespace, line breaks, etc.

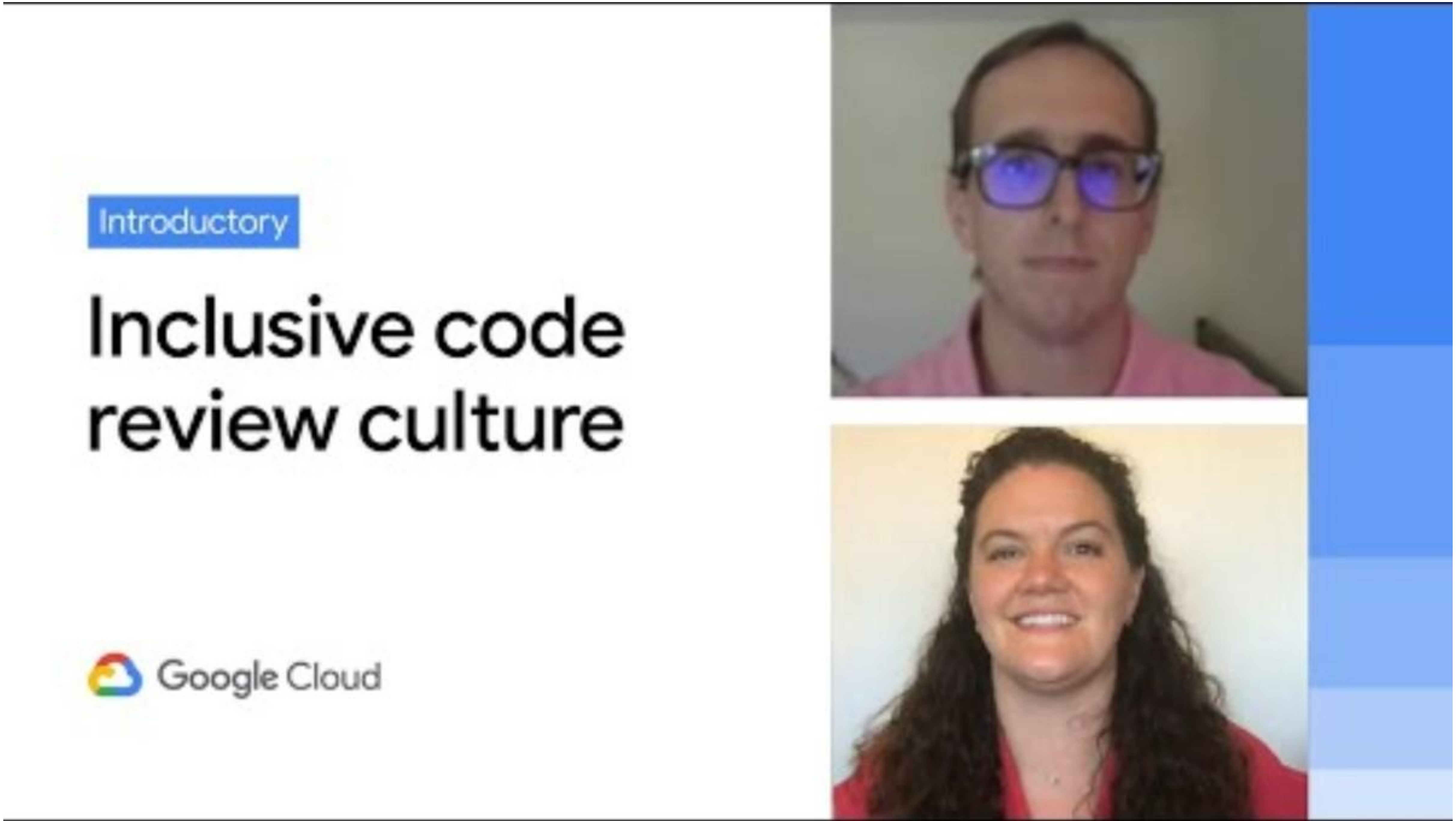
QUIZ: Code Reviews

Improved Code

```
    ...
    /** Return true if this is a premium account. */
    public boolean isPremium() {
        return accountType == Type.PREMIUM ||
               accountType == Type.PREMIUM_PLUS;
    }

    /** Return the sum of broker fees for all given accounts. */
    public static double calculateFee(Account[] accounts) {
        double totalFee = 0.0;
        for (Account account : accounts) {
            if (account.isPremium()) {
                totalFee += BROKER_FEE_PERCENT * account.interest();
            }
        }
        return totalFee;
    }
}
```

- Add comments
- Make fields private.
- Replace magic values (e.g., 365.25) with constants.
- Use an ***enum*** for account types.
- Use consistent whitespace, line breaks, etc.


Introductory

Inclusive code review culture

Google Cloud

Reminders

- **Project Plan Presentation** during **next class**. Please upload your presentations and reports on time.
  - Each group will present for **15 minutes** about **what exactly they plan to do** and **what exactly they expect to achieve**
  - Your grades are based on your submission so **seek feedback before submission deadline**
- **Paper Presentations** start next Monday.
  - **Do not just present the paper,** you need to come up with **3 discussion points** that are **not** described in the paper!
  - Class participation depends on Q/A during presentations!