**Software Quality**

Software Quality shows how good and reliable a product is. To convey an associate degree example, think about functionally correct software.

It performs all functions as laid out in the [SRS document](https://www.geeksforgeeks.org/software-requirement-specification-srs-document-checklist/). But, it has an associate degree virtually unusable program. even though it should be functionally correct, we tend not to think about it to be a high-quality product.

**Factors of Software Quality**

The modern read of high-quality associates with software many quality factors like the following:

1. **Portability:** A software is claimed to be transportable, if it may be simply created to figure in several package environments, in several machines, with alternative code merchandise, etc.
2. **Usability:** A software has smart usability if completely different classes of users (i.e. knowledgeable and novice users) will simply invoke the functions of the merchandise.
3. **Reusability:** A software has smart reusability if completely different modules of the merchandise will simply be reused to develop new merchandise.
4. **Correctness:** Software is correct if completely different needs as laid out in the SRS document are properly enforced.
5. **Maintainability:** A software is reparable, if errors may be simply corrected as and once they show up, new functions may be simply added to the merchandise, and therefore the functionalities of the merchandise may be simply changed, etc
6. **Reliability.**Software is more reliable if it has fewer failures. Since software engineers do not deliberately plan for their software to fail, reliability depends on the number and type of mistakes they make.

Designers can improve reliability by ensuring the software is easy to implement and change, by testing it thoroughly, and also by ensuring that if failures occur, the system can handle them or can recover easily.

1. **Efficiency.** The more efficient software is, the less it uses of CPU-time, memory, disk space,[network bandwidth](https://www.geeksforgeeks.org/introduction-to-bandwidth/), and other resources. This is important to customers in order to reduce their costs of running the software, although with today’s powerful computers, CPU time, memory and disk usage are less of a concern than in years gone by.

**Software Quality Management System**

Software Quality Management System contains the methods that are used by the authorities to develop products having the desired quality.

1. **Managerial Structure**

Quality System is responsible for managing the structure as a whole. Every Organization has a managerial structure.

1. **Individual Responsibilities**

Each individual present in the organization must have some responsibilities that should be reviewed by the top management and each individual present in the system must take this seriously.

**Quality System Activities**

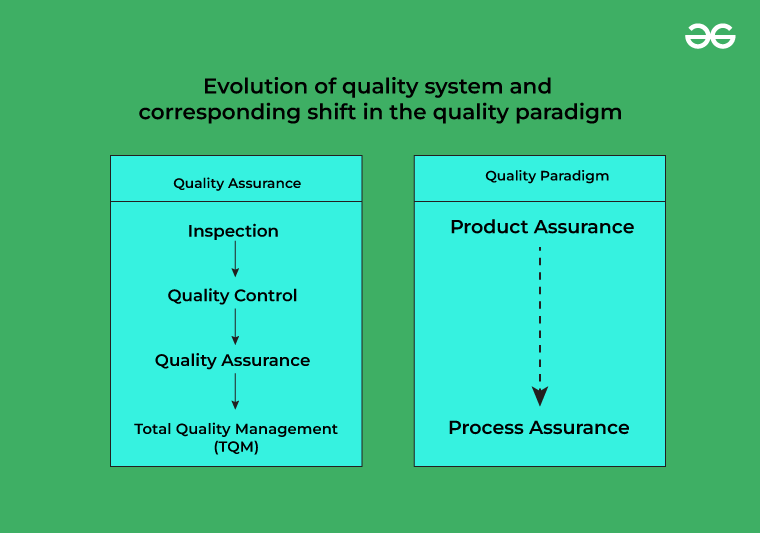
The activities which each quality system must have been

1. Project Auditing.
2. Review of the quality system.
3. It helps in the development of methods and guidelines.

**Evolution of Quality Management System**

Quality Systems are basically evolved over the past some years. The evolution of a Quality Management System is a four-step process.

1. The main task of [quality control](https://www.geeksforgeeks.org/differences-between-quality-assurance-and-quality-control/) is to **detect defective devices**, and it also helps in finding the cause that leads to the defect. It also helps in the correction of bugs.
2. [**Quality Assurance** :](https://www.geeksforgeeks.org/software-engineering-software-quality-assurance/)helps an organization in making good quality products. It also helps in improving the quality of the product by passing the products through security checks.
3. [**Total Quality Management(TQM)**](https://www.geeksforgeeks.org/what-is-total-quality-management-tqm-and-just-in-time-jit-kanban/): checks and assures that all the procedures must be continuously improved regularly through process measurements.



***Key terms used in software quality***

**Error vs Defect vs Failure**

https://tuskr.app/cassets/img/1px.png

The terms: error, [defect](https://tuskr.app/learn/defect), bug, and failure are closely related in software development. In a nutshell: An error leads to a defect, and when these defects go undetected, they lead to failure.

**Error**

An error is a mistake made by a developer in the code. Its main causes are:

* **Negligence:** Carelessness often leads to errors
* **Miscommunication:** An unclear feature specification or its improper interpretation by developers could lead to slips.
* **Inexperience:** Inexperienced developers often miss out on essential details, leading to faults down the line.
* **Complexity:** Intricate algorithms can cause developers to make mistakes in their coding logic.

**Defect & Bug**

A system is said to have a defect is when its expected and actual behavior differ. A defect, when found by a tester or end-user, is called a bug. Its main causes are:

* Missing test cases
* Test cases not executed by the tester
* Improper execution of test cases
* Code changed after testing

**Failure**

When one or more bugs cause a significant loss of functionality, a failure is said to have occurred.

**software quality assurance**

Software quality assurance (SQA) is a methodology to ensure that the quality of the software product complies with a predetermined set of standards.

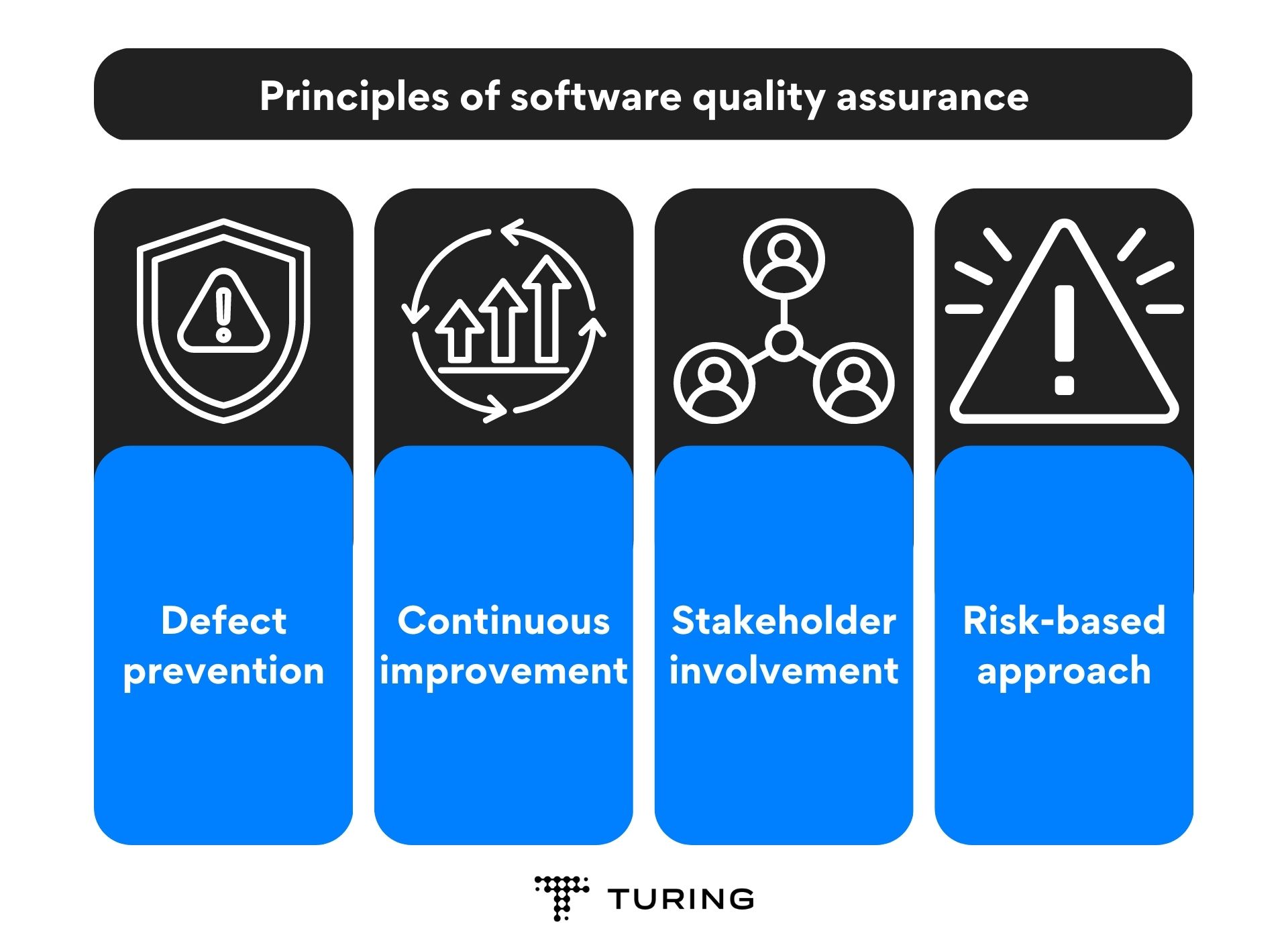
**purpose of software quality assurance**

SQA is not just a step in the development process; it functions in parallel with the software development life cycle. Businesses must ascertain that every part of the software, internal and external, is up to the predefined standard.

SQA tests every block of this process individually to identify issues before they become major problems.

* Externally, businesses evaluate efficiency, reliability, and cost of maintenance.
* Internal characteristics tested by software QA processes include structure, complexity, readability, flexibility, testability, and the coding practices developers have followed to develop the software.

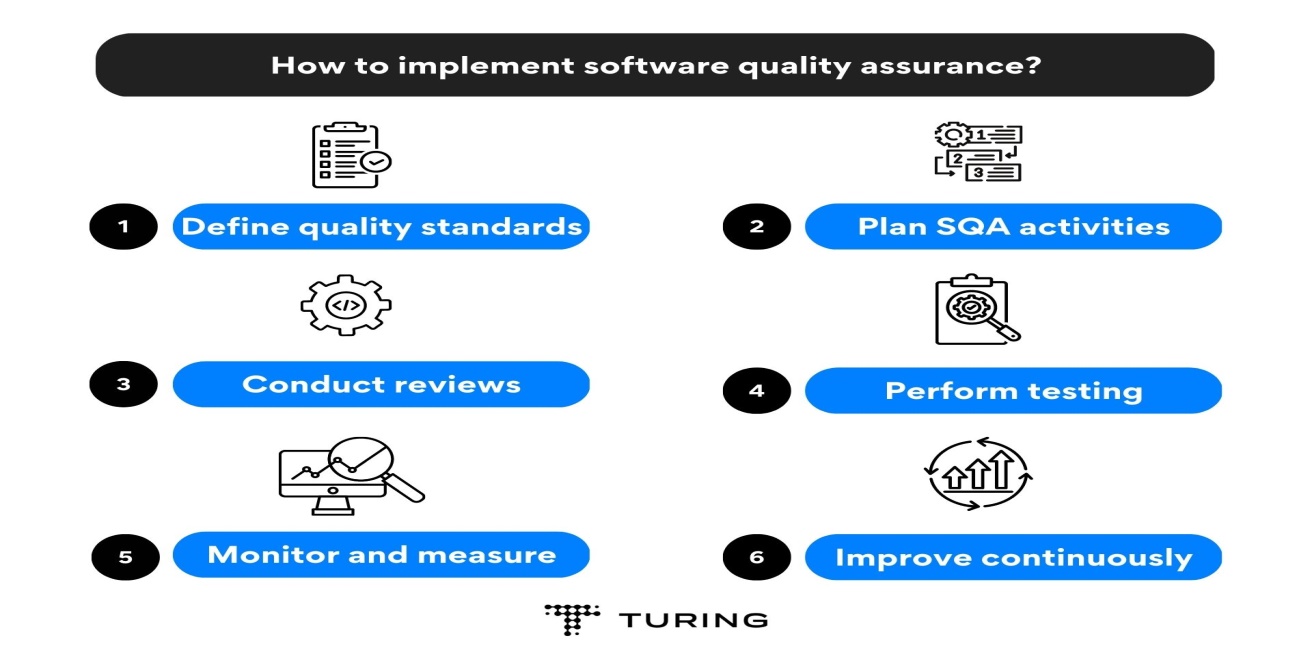
**What are the principles of software quality assurance?**



Principles of Software Quality Assurance

1. **Defect prevention:** It is always better to prevent defects and errors in the software product than to correct them later. And so, the first principle of SQA emphasizes the importance of identifying and addressing potential issues early in the software development lifecycle. Unlike quality control, SQA focuses on fixing the root cause of defects and errors, and not just the symptoms.
2. **Continuous improvement:** Here’s the thing: SQA is not a one-time thing. It is more like an ongoing process you need to integrate into your software development lifecycle. In other words, the second principle, i.e., continuous improvement underlines the need to consistently monitor and improve the quality of the software product.
3. **Stakeholder involvement:** SQA must involve all stakeholders in the software development process, including customers, developers, testers, [QA team leads](https://www.turing.com/resources/quality-assurance-team-lead), and project managers. And thus, this third principle talks about the importance of collaboration and communication between the involved parties to ensure a smooth software development process.
4. **Risk-based approach:** Last but not least, SQA must focus on identifying and addressing the most significant risks in the software product. Simply put, this principle emphasizes the importance of prioritizing risks based on their potential impact on the software product.

**How to implement software quality assurance**



**How to implement software quality assurance**

To implement SQA effectively, it is essential to follow a structured approach. You can follow the below-mentioned steps to implement SQA:

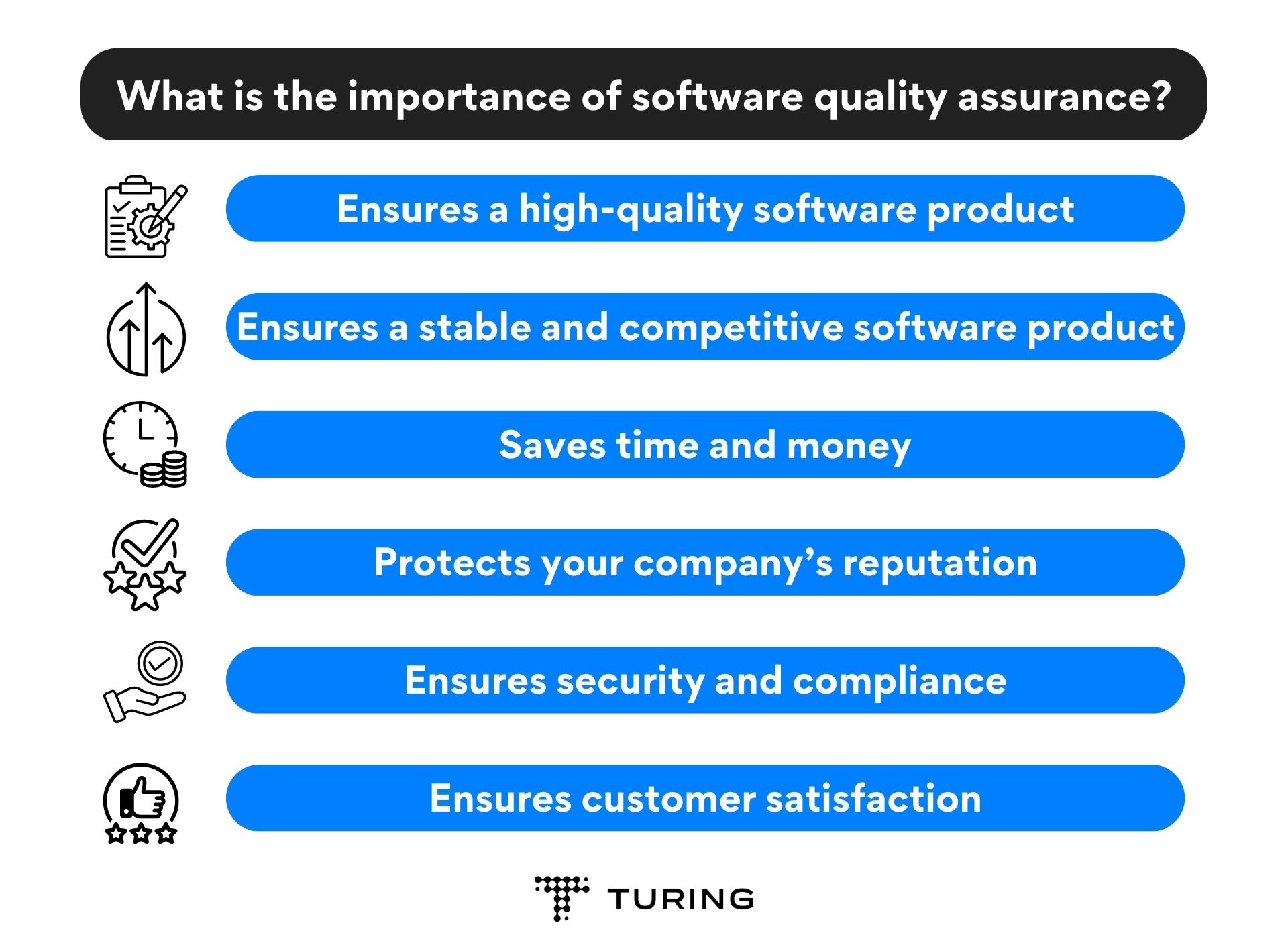
1. **Define quality standards:** Clearly define the quality standards that your software product must meet. This includes defining requirements, acceptance criteria, and performance metrics. These standards should be agreed upon by all stakeholders, including the development team, management, and customers.
2. **Plan SQA activities**: Develop a plan for the SQA activities that will be performed throughout the software development life cycle. This plan should include reviews, testing, and documentation activities. It should also specify who will be responsible for each activity and when it will be performed.
3. **Conduct reviews**: Conduct reviews of software artifacts such as requirements, design documents, and code. These reviews should be conducted by a team of experts who are not directly involved in the development process. This will help identify defects early in the development process and reduce the cost of fixing them later.
4. **Perform testing:** Perform different types of testing such as unit testing, integration testing, system testing, and acceptance testing. Use automated testing tools to increase efficiency and reduce the risk of human error.
5. **Monitor and measure**: Monitor and measure the quality of the software product throughout the development process. This includes tracking defects, analyzing metrics such as code coverage and defect density, and conducting root cause analysis.
6. **Improve continuously:** 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.

**What are the different software quality assurance approaches**

We have divided this section into parts based on the approaches to software quality assurance.

**Part 1: From a broader perspective, there are two different approaches to software QA:**

1. **Software quality defect management approach**  
   The software quality defect management approach focuses on counting and managing defects. The level of severity can generally categorize defects. Software development teams use tools like defect leakage matrices and clear and concise control charts to measure and enhance the capability of their [software development](https://www.turing.com/blog/software-development-trends/) process.
2. **Software quality attributes approach**  
   The software quality attributes approach works by helping software engineers analyze the performance of a software product. This approach focuses on directing the engineer’s attention to several quality factors. While some of these attributes may overlap or fall under another, there are five essential quality characteristics that you should consider:
3. **Reliability**. Reliability reflects the system’s ability to continue operating overtime under different working environments and conditions. The application should consistently return correct results.
4. **Usability.** Software applications should be easy to learn and navigate. This user-friendliness and effectiveness of utilizing the product are called usability.
5. **Efficiency**. This software QA attribute indicates how well the system uses all the available resources. It is shown by the amount of time the system needs to finish any task.
6. **Maintainability**. It shows how easy it is to maintain different system versions and support changes and upgrades cost-effectively.
7. **Portabilit**y. This software quality assurance attribute demonstrates the system’s ability to run effectively on various platforms — for example, data portability, viewing, hosting, and more.



**Difference between quality assurance and quality control**

**Quality Assurance**

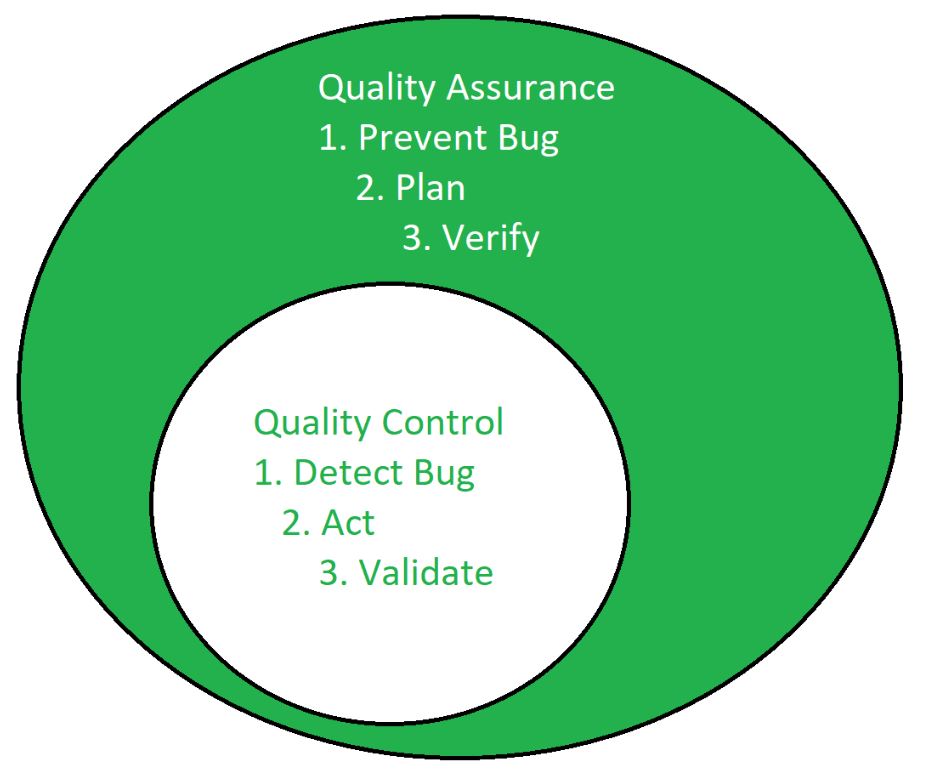
Quality assurance is a method of making the software application with fewer defects and mistakes when it is finally released to the end users. Quality Assurance is defined as an activity that ensures the approaches, techniques, methods, and processes designed for the projects are implemented correctly. It recognizes defects in the process. Quality Assurance is completed before Quality Control.

* It focuses on preventing defects.
* It is a proactive process and is preventive in nature.
* It helps to recognize flaws in the process.
* These activities monitor and verify that the processes used to manage and create deliverables have been followed.

**Quality Control**

Quality Control is a software engineering process that is used to ensure that the approaches, techniques, methods, and processes designed for the project are followed correctly. Quality control activities operate and verify that the application meet the defined quality standards.

* It focuses on an examination of the quality of the end products and the final outcome rather than focusing on the processes used to create a product.
* It is a reactive process and is detection in nature.
* These activities monitor and verify that the project deliverables meet the defined quality standards.

   
Below are the differences between Quality Assurance and Quality Control:

**l**

**What is Capability Maturity Model (CMM)?**

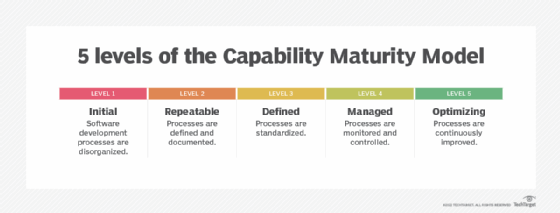
The Capability Maturity Model (CMM) is a methodology used to develop and refine an organization's [software development](https://www.techtarget.com/whatis/definition/software-development) process. The model describes a five-level evolutionary path of increasingly organized and systematically more mature processes.

CMM establishes a framework for continuous process improvement. It is more explicit than the ISO standard in defining the means to be employed to that end.

**CMM's five levels of maturity for software processes**

There are five levels to the CMM development process. They are the following:

1. **Initial.** At the initial level, processes are disorganized, ad hoc and even chaotic. Success likely depends on individual efforts and is not considered to be repeatable. This is because processes are not sufficiently defined and documented to enable them to be replicated.
2. **Repeatable.** At the repeatable level, requisite processes are established, defined and documented. As a result, basic [project management](https://www.techtarget.com/searchcio/definition/project-management) techniques are established, and successes in key process areas are able to be repeated.
3. **Defined.** At the defined level, an organization develops its own standard software development process. These defined processes enable greater attention to documentation, standardization and integration.
4. **Managed.** At the managed level, an organization monitors and controls its own processes through data collection and analysis.
5. **Optimizing.** At the optimizing level, processes are constantly improved through [monitoring feedback from processes](https://www.techtarget.com/searchsoftwarequality/tip/6-ways-to-catch-defects-in-software-tighten-feedback-loops) and introducing innovative processes and functionality.

The Capability Maturity Model takes software development processes from disorganized and chaotic to predictable and constantly improving

**Software Metrics**

A software metric is a measurable or countable measure of software characteristics. Software metrics can be used for a range of things, such as evaluating software performance, planning work items, estimating productivity, and so on.

Many indicators are intertwined throughout the software development process. The four functions of management are equivalent to software metrics: **planning**, **organization**, **control**, and **improvement**.

**Types of Software Metrics**

There are three types of metrics: product metrics, process metrics, and project metrics.

1. **Product Metrics** - Size, complexity, design features, performance, and quality level of the product are all factors to consider.
2. **Process Metric**s - To optimise software development and maintenance, process metrics can be used. Eliminating the faults rate during development, the pattern of testing defect arrival and the time it takes for a fixed operation to complete are all examples.
3. **Project Metrics** - The characteristics and execution of a project are described by project metrics. The number of software developers, staffing patterns across the software life cycle, cost, scheduling, and productivity is all examples of factors to consider.

Some measures fall into more than one category. A project's in-process quality metrics, *for example*, are both process and project metrics.

Software metrics cover a wide range of activities, including the following:

* Estimation of costs and time
* Model and metrics of productivity
* Gathering data
* Models and measures of quantity
* Models of dependability
* Models for performance and evaluation
* Metrics of structural and complexity

Software measurement encompasses a wide range of activities, including models for *projecting software project expenses* at various stages and metrics of program structure.

**How can Software Metrics be Used?**

Software development metrics can be used by project owners, project managers, developers, and quality assurance teams to:

* **Project Management and Planning**  
  Management relies heavily on measurement. Metrics for software development give a clear picture of what and how teams performed in previous project iterations. A project manager can better estimate and plan a budget, time, resources, and requirements for subsequent iterations based on the data, as well as quickly determine if an iteration or the complete project goes wrong based on the data.
* **Project Overview**  
  Metrics allow the project owner to easily comprehend and assess the project's current state, challenges, and solutions.
* **Task Prioritization**  
  Metrics are a useful technique to determine which tasks should be completed in which sequence to maximise value. If, *for example*, user satisfaction is low due to ongoing dissatisfaction with the quality of software upgrades that cause problems. Instead of delivering a huge number of new features, it might be time to start allocating more effort to regression testing with each iteration.
* **Change Management**  
  Metrics can help determine whether changing a strategy, a practice, a tool, or anything else adds value, how much benefit can be expected, and how it connects to the investments made
* **SLA Monitoring and Reporting**  
  Customers may easily define and track the value they expect from an outsourcing provider using metrics, as well as determine how productive outsourced employees are. A vendor, on the other hand, may vividly demonstrate the enhancements that have been made.

**What should be Measured?**

It's vital to remember that the metrics list should be defined on a case-by-case basis before we begin. It is an outrageous waste of time and effort to merely track whatever a project management tool gives or a software development framework recommends, or to blindly duplicate the metrics provided from another project.

* **Quality of the Delivered Solution**  
  The external characteristics of software are of relevance to business stakeholders. Modern software quality can be represented by eight basic criteria, [according to ISO/IEC 25010](https://www.iso.org/standard/35733.html): being delivered to requirements, reliability, maintainability, compatibility, portability, security, usability, and performance. Each of the features can be further broken into a collection of characteristics, requiring the tracking of a large number of metrics to provide a complete picture.
* **Code Quality**  
  The development teams, project managers, and project owners are all interested in code quality. As a result, metrics that can provide insight into the project's technical characteristics, such as algorithmic complexity, code duplication, test coverage, the number of needless dependencies, defect density, and so on, are of interest to team leads, architects, and developers.
* **Process Quality**  
  The project manager's primary focus will be on tracking costs, resources, timelines, and performance. They must also comprehend the efficiency of the current development techniques. Each programming paradigm, software development model, and framework will have its own set of success indicators: for linear (traditional) development with a fixed scope, it'll be the percentage of scope completed, whereas agile and lean processes will require measurements of general lead time, cycle time, team velocity, and so on.
* **User Satisfaction**  
  It's also critical to assess the satisfaction of the intended users. A customer satisfaction score will be used for public products, while employee feedback will be used for internal applications. In these circumstances, criteria such as interface consistency, the attractiveness of interactions, message clarity, interface element clarity, and function understandability are appropriate.