# Software Quality Factors

**Software engineering is …**

The systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, and testing of software.

**SE was introduced to …**

Address the issues of low-quality software projects.

Problems generally arise when a software exceeds timelines, budgets, and reduced levels of quality.

It ensures that the application is built within requirements, on time, and on budget.

**How do we “Assess” a Software?**

A software product is generally judged by how easily it can be used by the end-user and the features it offers.

A software product, or application, must *…*

* Work without crashing or producing errors.
* Include functionalities that fit the requirements.
* Show accurate data / output.

**Security/Integrity:** The characteristic of a system that does not reveal information to improper parties.

**Maintainability:** is the ease with which software can be modified and adapted for other purposes. The software relies on architectural rules and consistent coding to be more maintainable.

**Reliability:** the stability of a program when exposed to unexpected conditions. Reliable software has minimal downtime, good data integrity, and no errors that directly affect users.

**Usability:** Usability requirements deal with the staff resources needed to train a new employee and to operate the software system.

**Flexibility:** This factor deals with the capabilities and efforts required to support adaptive maintenance activities of the software. These include adapting the current software to additional circumstances and customers without changing the software.

**Efficiency:** The characteristic of software that do not make wasteful use of system resources such as memory and processor cycles.

**Compatibility: The degree to which the software is compatible with the intended hardware, browser, operating system, etc.**

**Portability:** The adaptation of a software system to other environments consisting of different hardware, different operating systems, and so forth.

**Testability:** Testability is the effort required to test to ensure that the system or a module performs its intended function.

**Reusability:** the extent to which parts of the software can be reused in other related applications.

**Correctness:** The extent to which a program satisfies its specifications.

**Functional Suitability:** This characteristic represents the degree to which a product or system provides functions that meet stated and implied needs.

# Software Bugs

A Software Bug is a **failure** or **flaw** in a program that produces **unexpected** or **incorrect** **results**, or **behaves** in **unintended** **ways**. It’s an **error** that prevents the application from functioning as it should.

Bugs can trigger errors that may have **ripple effects**. Bugs may have **subtle** **effects** or cause the program to **crash** or **freeze** the computer. Other bugs qualify as **security** **bugs** and might, for example, enable a **malicious user to bypass access controls** in order to obtain unauthorized privileges.

**Causes:**

* **Miscommunication:**The success of any software application depends on communication between stakeholders, development and testing teams. Unclear requirements and misinterpretation of requirements are two major factors causing defects in software.
* **Software Complexity:**   
  The complexity of current software applications can be difficult for anyone without experience in modern-day software development (Windows-type interfaces, Client-Server, and Distributed Applications, Data Communications, enormous relational databases, …)
* **Programming Errors:**Programmers can make programming mistakes (experience, domain knowledge, focus …). Lack of simple coding practices, unit testing, debugging are some of the common reasons why these issues get introduced at the development stage.
* **Changing Requirements:**The customer may not understand the effects of changes or may understand and request them anyway.
* **Overconfidence:**Saying “No problem! It’s a piece of cake.” instead of “That adds a lot of complexity and we could end up making a lot of mistakes if we are not given the adequate time.”
* **Lack of Testing Skills:**

Domain knowledge and the tester’s ability to find defects can produce high-quality software. If neglected, they can result in defective software.

**Types:**

* **Crash:**A situation when the software solution, operating system or program stops working properly and forces itself to shut down. It can cause a lot of damage because it often results in data loss.
* **Functional Error:**One of the most basic qualities of software is the fact that everything works as it was intended to do and the software solution is fully functional. But if there is something wrong and the behavior of the software varies from the expected, we call it a functional error.
* **Missing Command:**This particular type of bug occurs when there is a command missing. The user is expecting an action or to be allowed to perform some activity.
* **Hardware Usage Errors:**Occur when the software is being used on a wrong device or in an unadjusted environment. Most common hardware usage errors are caused by the wrong operating system, too low computing power or a mismatched device.
* **Control Flow Errors:**Control flow of software describes what will be done next and on what conditions. Errors connected to the control flow prevent software from proceeding to the next tasks in the correct way.

# Software Development Life Cycle

**Off the shelf** software are standardized software applications that are mass-produced, available to the general public, and fit for immediate use. They are designed for a broad range of customers, offering a comprehensive set of features to streamline operations.

**Tailor-made** or **bespoke** software is [software](https://en.wikipedia.org/wiki/Software) that is specially [developed](https://en.wikipedia.org/wiki/Software_development) for some specific organization or other user. As such, it can be contrasted with the use of software packages developed for the [mass market](https://en.wikipedia.org/wiki/Mass_marketing), such as [commercial off-the-shelf](https://en.wikipedia.org/wiki/Commercial_off-the-shelf) software, or existing [free software](https://en.wikipedia.org/wiki/Free_software).

**Software Specification:**

* The process of understanding and defining what services are required and identifying the constraints on these services.
* It ensures software will meet the user’s expectations.
* It’s a critical stage of the software process as errors at this stage will reflect later on the next stages.
* At the end of this stage, a requirements document that specifies the requirements will be produced and validated with the stakeholders.

**Software Design and Implementation:**

* A software design is a description of the structure of the software to be implemented, data models, interfaces between system components, and maybe the algorithms used.
* The implementation phase is the process of converting a system specification into an executable system.

**Software Verification and Validation:**

* Software V&V is intended to show that a system both conforms to its specification and that it meets the expectations of the customer.
* Validation may also involve checking processes, such as inspections or reviews at each stage of the software process, from defining the requirements till the software development.
* Testing, where the system is executed using simulated test data, is an important validation technique.

**Software Maintenance and Evolution:**

* Requirements are always changing, even after the system has been put into its operating environment.
* Even after deployment, software systems may need to undergo changes, to ﬁx problems or improve the system.
* Software maintenance can be adaptive, corrective, perfective, or preventive.
* It’s hard to design a maintainable system, because it’s hard to foresee all future change
* Evolution is intrinsic to the nature of software development.

**SDLC is** a process used by the software industry to design, develop and test high quality software. It is methodology for improving the quality of software and the overall development process.

**SDLC aims** for production of high quality software that meet customers’ expectations and reaches completion within times and cost estimates.

**SDLC Models:**

**Waterfall model, Iterative model, Spiral model, V-model, Agile model, RAD model.**

Each model follows a series of steps unique to its type to ensure success in the process of software development.

**SDLC phases:**

1. Analysis (Plan)
2. Design
3. Implementation
4. Testing
5. Deployment
6. Maintenance

# Ethics in SE

Ethics are broadly the **set of rules**, written and unwritten, that set out how we expect ourselves and others to behave, and why.

A **dilemma** is a situation involving choice between equally unsatisfactory alternatives.

**Ethical dilemmas** are situations involving conflicting moral claims, and give rise to such questions as: What **harm** and **benefit** result from this decision?

**Software Engineers / Ethics…**

By participating in a software development process, software engineers can influence the final product, namely the software itself, in different ways including those that may be contrary to public interest.

they could engage in an unethical behavior, inadvertently or deliberately.

This can adversely affect the acceptance of software as a useful product, question the credibility of software engineering as a profession, and lead to legal implications

**Software Developers …**

Are an integral part of protecting our critical infrastructure:

* They write the algorithms that make increasingly important decisions about people’s lives.
* They help safeguard our information from hackers.

They determine how computers, drones, banks, criminal sentencing, predictive policing, and surveillance work

**General Ethical Principles by ACM:**

**Personal Responsibilities:**

1. Contribute to human well-being
2. Avoid Harm
3. Be Honest and Trustworthy
4. Be fair and take action not to discriminate
5. Respect the work required to produce new ideas, inventions, creative works, and computing artifacts.
6. Respect privacy.
7. Honor Confidentiality

**Professional Responsibilities:**

1. Strive to achieve high quality in both the processes and products of professional work
2. Maintain high standards of professional competence, conduct, and ethical practice.
3. Know and respect existing rules pertaining to professional work
4. Accept and provide appropriate professional review
5. Give comprehensive and thorough evaluations of computer systems and their impacts, including analysis of possible risks
6. Perform work only in areas of competence.
7. Design and implement systems that are robustly and usably secure.

# Internet of Things (IoT)

It is the connection of any device to the **internet** using **embedded software** and **sensors** to collect and exchange data with one another. We can teach objects to respond to physical stimulus (motions, vocal commands, eyeball tracking) and even automatic physiological behaviors. You can lock doors, arm the alarm and reset the thermostat when the owners leave the proximity of the house. Alphanumeric passwords are not a biologically intuitive way to prove identity. A more natural and secure method would come in the form of sensors that can read personal attributes such as a fingerprint, or a heart rate.

Enabling a form of free-flowing conversation between man and machine

(anticipate, react, respond and enhance the physical world).

**Major uses of IoT:**

* Connected appliances.
* Smart home security systems.
* Autonomous farming equipment.
* Wearable health monitors.
* Smart factory equipment.
* Biometric cyber security scanners.

The increasing sophistication of the sensors embedded in technology makes it possible for devices (“things”) to read, determine and understand consumers at unprecedented levels.