***Define software metrics, explain the three types of metrics in detail, How can Software Metrics be Used***

Software metric is a quantitative measure of software properties that may be measured or counted. These metrics evaluate the quality, performance, and efficiency of a software product or development process. Software metrics are useful for several purposes, including measuring software performance, planning work items, and evaluating productivity.

**Project Metrics**

Project metrics in software engineering are quantitative statistics used to evaluate the performance and progress of software development initiatives. These metrics give critical information about several elements of project management, including schedule adherence, resource usage, cost control, and overall project quality. By measuring and evaluating project metrics, software engineering teams may efficiently manage projects, assure timely delivery, and stay on track with corporate goals.

**Process metrics**Process metrics in software engineering are quantitative measurements used to analyze and monitor the efficacy, efficiency, and quality of the software development process. These metrics examine the actions, workflows, and methods used throughout the software development lifecycle. By examining process metrics, software engineering teams can identify areas for improvement and make data-driven decisions that improve their development processes.   
The majority of businesses and IT community members prioritize speedier code delivery. However, delivering predictably is critical for setting right expectations, driving alignment across functional teams, and allowing for improved execution and, as a result, greater market penetration for the value your company has worked so hard to create. Cycle time and lead time, deployment frequency, and task resolution rate over time are all process metrics for software engineering teams.  
Cycle time refers to the time required to accomplish a certain task or process.  
Lead time: Measures the time passed from the beginning of a software development task to its conclusion.  
Deployment Frequency - Tracks how quickly and iteratively a software engineering team can deliver value.  
Task Resolution Rate Over Time- Tracks how a team is going toward completing work.

**Product metrics**

Product metrics in software engineering are quantifiable measurements used to evaluate the qualities and performance of software products throughout their development and maintenance lifespan. These metrics provide useful information on software quality, effectiveness, efficiency, and reliability. Using a complete product metrics framework, software engineering teams may obtain a better understanding of their products, make data-driven decisions, and constantly improve their software development processes.

**Uses of Software Metrics**

Metrics are used to track development stages and progress against timetables.  
Helps with quality assurance. Identifies opportunities for improvement and monitors failure rates.  
Metrics help in resource management. Assessing productivity helps to deploy resources more effectively.  
helps in decision-making. Provides data-driven insights to help project managers make better decisions.

***Discuss functional and non-functional requirements  in software products , use suitable  example guidelines***

**Functional requirements**The functional requirements define the exact behaviors and functions that the software system must do. They specify what the system should perform.   
  
Examples of functional requirements  
The system must allow users to log in with their username and password.  
The system will determine the overall cost of a shopping cart.  
The system will generate a sales report by product category.

**Nonfunctional Requirements**Non-functional requirements establish the software system's restrictions and characteristics, such as performance, security, usability, and maintainability. They define how well the system should perform its functions.   
  
Examples of non-functional requirements:  
Performance -The system must react to user requests within two seconds.  
Usability -The system must have a user-friendly interface and clear instructions.  
  
It was introduced in 1977. This model is incorporated with many attributes, termed software factors, which influence software. The model distinguishes between two levels of quality attributes:

**i) Quality Factors:** The higher-level quality attributes that can be accessed directly are called quality factors. These attributes are external. The attributes at this level are given more importance by the users and managers.

**ii) Quality Criteria:** The lower or second-level quality attributes that can be accessed either subjectively or objectively are called Quality Criteria. These attributes are internal. Each quality factor has many second-level quality attributes or quality criteria.

**Example:** The usability quality factor is divided into operability, training, communicativeness, input/output volume, and input/output rate.

This model classifies all software requirements into 11 software quality factors. The 11 factors are organized into three product quality factors: **Product Operation**, **Product Revision**, and **Product Transition.**

**i) Product Operation**

Product Operation includes five software quality factors, which are related to the requirements that directly affect the operation of the software such as operational performance, convenience, ease of usage, and correctness. These factors help in providing a better user experience.

* **Correctness:** The extent to which software meets its requirements specification.
* **Efficiency:** The number of hardware resources and code the software, needs to perform a function.
* **Integrity:** The extent to which the software can control an unauthorized person from accessing the data or software.
* **Reliability:** The extent to which software performs its intended functions without failure.
* **Usability:** The extent of effort required to learn, operate, and understand the functions of the software.

**ii) Product Revision**

Product Revision includes three software quality factors, which are required for testing and maintenance of the software. They provide ease of maintenance, flexibility, and testing efforts to support the software to be functional according to the needs and requirements of the user in the future.

* **Maintainability:** The effort required to detect and correct an error during maintenance.
* **Flexibility:** The effort needed to improve an operational software program.
* **Testability:** The effort required to verify software to ensure that it meets the specified requirements.