what is function point analysis in Software Engineering

ChatGPT

Function Point Analysis (FPA) is a software measurement technique used to estimate the size and complexity of a software system. FPA measures the functionality that the software provides from a user’s perspective, rather than measuring the lines of code or the number of programming hours.

The primary goal of FPA is to measure the amount of functionality that a software system provides, and then use this measurement to estimate the effort required to develop the system, the cost of development, and the schedule for completion. This measurement is expressed in terms of function points, which are a unit of measure used to represent the size of a software system based on its functionality.

To perform FPA, the software system is analyzed and categorized into five functional components:

1. External Inputs (EI)
2. External Outputs (EO)
3. External Inquiries (EQ)
4. Internal Logical Files (ILF)
5. External Interface Files (EIF)

Each functional component is then assigned a weight based on its complexity and the number of inputs and outputs it has. The weights are then added together to calculate the total function points for the software system.

FPA is useful for estimating the size and complexity of a software system early in the development process. It can also be used to compare the relative size and complexity of different software systems. However, it should be noted that FPA does not account for factors such as the quality of the code, the experience of the development team, or the complexity of the hardware or software environment in which the system will operate.

Jagdish Singh Charitable Trust's (Regd.)
UR COLLEGE OF ENGINEERING & TECHNOL
tcet

what are COCOMO and COCOMO 2 models in software engineering, explain in short in layman's terms

ChatGPT

COCOMO stands for Constructive Cost Model, which is a software cost estimation model developed by Dr. Barry Boehm in the late 1970s. The model helps in estimating the effort, cost, and time required for software development based on various factors such as project size, complexity, and experience of the development team.

COCOMO uses a set of equations that are based on historical data to estimate the effort required to develop software. It considers various factors such as the size of the software, the complexity of the project, the experience of the development team, and the software development environment. The model consists of three levels:

1. Basic COCOMO: This level estimates the effort required to develop the software based on the size of the project. The size is measured in lines of code.

2. Intermediate COCOMO: This level estimates the effort required to develop the software based on various factors such as complexity, development environment, and team experience.
3. Detailed COCOMO: This level estimates the effort required to develop the software based on a more detailed analysis of the project.

COCOMO 2 is an updated version of COCOMO, which was released in 1981. It includes additional factors such as software reuse, documentation requirements, and the impact of modern software development practices. COCOMO 2 has three sub-models:

1. Application Composition Model (ACM) - used for estimating the cost of software development through software reuse.
2. Early Design Model (EDM) - used for estimating the cost of software development during the early stages of the project when there is limited information available.
3. Post-Architecture Model (PAM) - used for estimating the cost of software development after the architecture has been defined.

Overall, COCOMO and COCOMO 2 are widely used software cost estimation models that help in estimating the effort, cost, and time required for software development. They are based on historical data and consider various factors to provide accurate estimations.

Give me basic point of difference between COCOMO and COCOMO 2

ChatGPT

The main difference between COCOMO and COCOMO 2 is that COCOMO 2 is an updated version of COCOMO and includes additional factors and sub-models that were not present in the original model.

Some of the key differences between COCOMO and COCOMO 2 are:

1. Factors Considered: COCOMO considers factors such as project size and team experience, while COCOMO 2 includes additional factors such as software reuse, documentation requirements, and modern software development practices.
2. Sub-models: COCOMO has only one model, while COCOMO 2 includes three sub-models: Application Composition Model (ACM), Early Design Model (EDM), and Post-Architecture Model (PAM).
3. Level of Detail: COCOMO provides estimates at a high-level, while COCOMO 2 provides more detailed estimates that can be customized based on the specific project requirements.
4. Age: COCOMO was first released in the late 1970s, while COCOMO 2 was released in 1981 and has been updated over time to reflect changes in software development practices.

Overall, while both COCOMO and COCOMO 2 are useful for estimating software development costs, COCOMO 2 is considered to be a more comprehensive and flexible model that is better suited to modern software development practices.

RMMM model in short easy terms

ChatGPT

THAKUR COLLEGE OF ENGINEERING & TECHNOLOGY
Estd. 2001
ISO 9001 : 2015 Certified
NBA and NAAC Accredited

The RMMM model is a risk management model used in software engineering to manage risks throughout the software development lifecycle. The acronym RMMM stands for Risk Mitigation, Monitoring, and Management.

Risk Mitigation refers to the process of identifying potential risks in the project and taking steps to reduce their impact or likelihood. This may involve developing contingency plans or taking steps to prevent the risk from occurring in the first place.

Risk Monitoring involves continuously monitoring the project to identify new risks as they arise and to track the effectiveness of risk mitigation strategies. This may involve regular status reports, risk assessments, and ongoing communication with stakeholders.

Risk Management involves making decisions about how to manage the risks identified in the project. This may involve accepting the risk and taking no further action, transferring the risk to another party, or taking additional steps to mitigate the risk.

Overall, the RMMM model provides a structured approach to risk management that helps to ensure that potential risks are identified, addressed, and managed effectively throughout the software development lifecycle.

Explain in short and easy terms "Reverse Engineering" concept in software Engineering

ChatGPT

Reverse engineering is the process of analyzing and understanding an existing software system in order to create a representation of its design and functionality. This is often done when the original design or documentation of the software is not available or has been lost.

Reverse engineering can involve a variety of techniques, including examining the code of the software, analyzing its behavior and performance, and using specialized tools to extract information about the system's structure and components.

The goal of reverse engineering is to create a model or representation of the software that can be used to understand its design, behavior, and functionality. This model can be used to identify potential problems or limitations in the software, to improve the design or performance of the system, or to create a new software system that is based on the existing one.

Overall, reverse engineering is an important concept in software engineering because it allows developers to understand and work with existing software systems, even when the original design or documentation is not available.

Explain in short and easy terms "Formal Technical Review" concept in software Engineering

ChatGPT

Formal Technical Review (FTR) is a software engineering process that involves a group of software developers reviewing a software artifact, such as a requirement specification or source code, in a structured and systematic way. The goal of FTR is to identify defects and areas for improvement in the artifact and to ensure that the software meets the specified requirements and quality standards.

Zagdu Singh Charitable Trust's (Regd.)
THAKUR COLLEGE OF ENGINEERING & TECHNOLOGY
tcet
THAKUR COLLEGE OF ENGINEERING & TECHNOLOGY
Estd. 2001
ISO 9001 : 2015 Certified
NBA and NAAC Accredited

The FTR process typically involves several steps, including preparation, review, and follow-up. During the preparation phase, the software developers who will participate in the review are identified, and the artifact to be reviewed is distributed to the participants.

During the review phase, the participants carefully examine the artifact, looking for errors, omissions, or other problems. The review is typically conducted in a structured and systematic way, with the participants using a checklist or other evaluation criteria to ensure that all aspects of the artifact are examined.

After the review is complete, the participants document any defects or areas for improvement that were identified during the review. This information is then used to create an action plan for addressing the identified issues.

Overall, FTR is an important concept in software engineering because it helps to ensure that software artifacts are reviewed in a structured and systematic way, which can help to identify defects and improve the quality of the software. FTR is particularly important for critical software systems, where even small defects or errors can have serious consequences.

Explain in simple terms Unit testing, Integration testing, System testing, Acceptance Testing also include what do they fall under White box or black Box

ChatGPT

Unit testing, integration testing, system testing, and acceptance testing are all types of software testing used in software engineering. Each type of testing has a specific purpose and is used to ensure that the software meets the specified requirements and quality standards.

1. Unit Testing: Unit testing is a type of testing that involves testing individual units or components of the software, such as functions or modules. Unit testing is typically performed by developers and is often automated using specialized testing frameworks. Unit testing is a form of white-box testing because it involves examining the internal structure and code of the software.
2. Integration Testing: Integration testing is a type of testing that involves testing the interaction between different components or modules of the software. The goal of integration testing is to identify defects or issues that arise when multiple components are combined. Integration testing can be performed using a variety of techniques, including top-down and bottom-up testing. Integration testing is a form of black-box testing because it focuses on the external behavior of the software.
3. System Testing: System testing is a type of testing that involves testing the software as a whole, including its interactions with external systems or hardware. The goal of system testing is to ensure that the software meets the specified requirements and quality standards in a real-world environment. System testing is a form of black-box testing because it focuses on the external behavior of the software.
4. Acceptance Testing: Acceptance testing is a type of testing that involves testing the software with the goal of determining whether it meets the specified acceptance criteria or user requirements. Acceptance testing is typically performed by end-users or stakeholders and is often the final stage of testing before the software is released. Acceptance testing can be performed using a variety of techniques, including manual and automated testing. Acceptance testing is a form of black-box testing because it focuses on the external behavior of the software.

Zagdu Singh Charitable Trust's (Regd.)
tcet
Estd. 2001
ISO 9001 : 2015 Certified

Overall, unit testing is a form of white-box testing, while integration testing, system testing, and acceptance testing are forms of black-box testing.

Zagdu Singh Charitable Trust's (Regd.)
THAKUR COLLEGE OF ENGINEERING & TECHNOLOGY
tcet
THAKUR COLLEGE OF ENGINEERING & TECHNOLOGY
Estd. 2001
ISO 9001 : 2015 Certified
NBA and NAAC Accredited