DATE / /
PAGE

(2) Behavioural Diagram - It portray a dynamic view of a system or behaviour of a system, which describes the functioning of the system. It includes use case diagrams, state diagrams, and activity diagrams. It defines the instruction within the system.

(a) State Diagrams - It is a behavioural diagram. It portrays the system's behaviour utilising finite state transactions. It is also known as the state chart diagram.

(b) Activity Diagrams - It models the flow of control from one activity to the other. With the help of an activity diagram, we can model sequentional and concurrent activities.

(c) Use Case Diagram - It represents the functionality of a system by utilising actors and use cases. It encapsulate the functional requirement of a system and its association with actors.

(d) Interaction Diagram - Interaction Diagrams are sub-class of behavioural diagrams that give emphasize to object interac-

tion and also depict the flow between various use case's element of a system. In simple words, it shows how object interact with each other and how the data flows within them. It consists of communication, interactions over view, sequence and timing diagrams.

(i) Sequence Diagram – It shows the interaction between the object in terms of messages exchange over time.

(ii) Communication Diagram – It shows the inter change of sequence messages between the object. It focus on object and their relations. It describes the static and dynamic behaviour of a system

(iii) Timing Diagram – It is a special kind of Sequence diagram used to depict the object behaviour over the specific period of time

(iv) Interaction Over View Diagram – It is a mixer of activity and sequence diagram that depict a sequence of actions to simplify the complex ~~or action~~ interaction into simple interaction.

DATE / /
PAGE

→ UML Relationship ←

(1) Cocomo Model - Also called as Constructive Cost Model. Cocomo model is a regression model based on LOC (Number of Lines of Code).

The Cocomo Model is a procedural cost estimate model for software projects and used as process of predicting the various parameters associated with making a project such as size, effort, cost, time and quality. It was proposed by Barry Boehm in 1981. It was based on the study of 63 projects, which makes it one of the best documented model.

The key parameters that defines the quality of any software product, which are also an outcome of the cocomo are primary effort and schedule.

(a) Effort - Amount of labour that will be required to complete a task. It is measured in person, months, units.

(b) Schedule - This simply means the amount of time required for the job to be completed. It is measured in the units of time such as weeks and months.

(2) Resource Management - By taking team experience project size and complexity into account, The model helps with efficient resource allocation.

(3) Project Planning - Cocomo assist in developing practical project plans that include objectives, due dates, benchmarks.

(4) Risk Management - Early in the development process, cocomo assisti in identifying potential hazards by including risk elements.

(5) Support for decisions - During project planning, the model provides a quantitive foundation for choices about scope, priority and resource allocation.

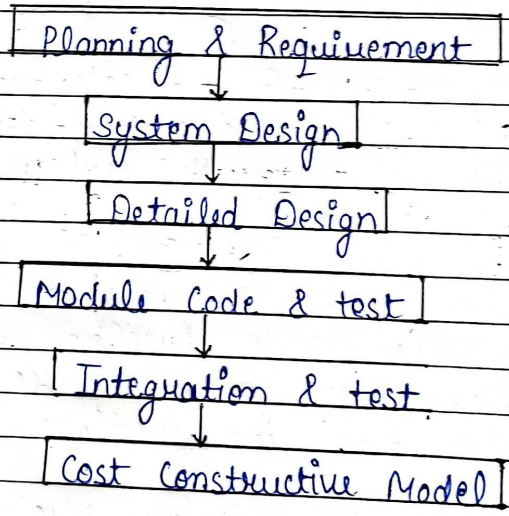
(6) Benchmarking - To compare and assest various Software development project to industry standards, cocomo offers a benchmark.

(7) Resource Optimization - The model helps to maximise the use of resources which rises productivity and lower's the cost

→ Detailed structure of Cocomo Model ←

Detailed Cocomo model incorporates ~~called~~ all-characteristics of the intermediate ~~perso~~ version with an assesment of the cost driver's impact on each step of Software engineering process. The detailed model uses different efforts, multipliers for each cost driver attribute.

There are six phases in the detailed cocomo are -



→ Importance of Cocomo Model ←

(1) Cost Estimation - To help with the resources, planning and project budgeting. Cocomo offers a methodical approach to software development cost estimation.

DATE / /
PAGE

→ Types of Software ←

(1) Organic Software
(2) Semi-Detached (~~Embedded~~)
Embedded

(1) ORGANIC - A software project is said to be organic type if the team size required is small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.

(2) SEMI-DETACHED - A software project is said to be a semi-detached type if the vital characteristics such as team size, experience and knowledge ~~of~~ of various programming environment lie in between organic and embedded. The project is less familiar and difficult to develop compared to the organic ones and require more experienced, greater guideline guidance and creativity.

(3) Embedded - A software project requiring the highest level of complexity, creativity and experience requirement falls under this category such software requires a larger team size than the other two ~~meth~~ models.

DATE / /
PAGE

* Advantages of Cocomo Model-

(1) Systematic Cost Estimation - Provides a systematic way to estimate the cost and effort of a software project.

(2) Helps to estimate cost effort - This can be used to estimate the cost & effort of a software project at different stages of the development project process.

(3) Helps in impact factors - Helps in identifying the factors that have the greatest impact on the cost and effort of a software project.

(4) Helps to evaluates the feasibility of a project - This can be used to evaluate the feasibility of a software project estimating the cost and effort required to complete it.

* Disadvantages of Cocomo Model -

(1) Assume project size as the main factor - Assumes that the size of the software is the main factor that determines the cost and effort of a software project which may not always be the case.

(2) Intermediate Model - The basic cocomo model the effort is only a function of the number of lines of code and some constant evaluated according to the different software system. However, in reality, no system's effort and schedule can be calculated based on lines of code. For that, various other factor such as reliability, experience and capability. These factors are known as cost drivers and the intermediate model utilize 15 such drivers for cost estimation. Classification of cost drivers and their attributes.

d
.38
.35
.32

(a) Product Attribute - • Required software reliability extend.
- Size of the application data base.
- The complexity of the product.
- ~~One~~ Run-time performance
- Memory constraints
- The volatile of the virtual machine environment.
- Required turn about time.
- Analyst capability.
- Software Engineering capability
- Application Experience
- Virtual Machine Experience
- Programming Lang
- Use of software tools
- Application of software engineering methods
- Required Development schedule

DATE / /
PAGE

→ Types of Cocomo Model ←

(1) Basic Model - $E = a(KLOC)^b$

$Time = c(Effort)^d$

Person required = Effort / time

KLOC → Thousand Lines of Code.

The above formula is used for the cost estimation of the basic cocomo model. The constant values 'a', 'b', 'c', 'd' for the basic model for the different categories of the system is given below -

Soft project	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-detached	3	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Note - The effort is measured in person per month, the development time is measured in months.
These formulas are used as such in the basic model calculation. This estimate is rough.

DATE / /
PAGE

adopt it to changes in hardware or software
@ Government Policy it's business rules.

(2) Perfective Maintenance - Due to pressure from

maintenance process in place, which includes testing and validation, version control and communication with stakeholders.

(3) It is important note that software maintenance can be costly, especially for large and complex system. Therefore, cost and effort of maintenance should be taken into account during the planning and development phases of the software project.

(4) It's also important to have a clear and well-defined maintenance plan that includes regular maintenance tasks testing, back-up and bug-fixing.

* Types of Software Maintenance:-

(1) Corrective Maintenance - This involves fixing errors and bugs in the software system.

(2) Patching :- It is an ~~test~~ implemented namely due to pressure from management. Patching is done for corrective maintenance but it gives rise to unforseen future errors due to lack of proper impact analysis.

(2) Adaptive Maintenance - This involves modifying the software system to

(2) Does not find development team specific characteristic. - Does not take into account the specific characteristics of the development team which can have a significant impact on the cost and effort of a software project.

(3) Not enough precise cost and effort estimate - This does not provide a precise estimate of the cost and effort of a software project, as it is based on assumptions and average

Software Maintenance

Software maintenance is a continuous process occurs throughout the entire lifecycle of the software system -

(1) The goal of the software maintenance is to keep the software system working correctly, efficiently and securly, and to ensure that continuous to too meet the needs of the user.

(2) This can include fixing bugs, adding new features, improving performance for updating the software to work with new hardware or software system.

It is also important to consider the cost effort required for software maintenance when developing a software system. It is important to have a well defined

- Older software program
- Changes a frequently left undocument which can also additionally reason greater conflict in the future.
- lack of code comments.
- lack of documentation
- legacy code.
- Complexity
- Changing requirements
- lack of test coverage
- lack of personal
- high cost
-

Reverse Engeene Engineering :—
Reverse engineering is the process of recovering the design and the requirement specification of a product from an analysis of its code

Why reverse engineering -

- providing proper system documentation
- Recovery of lost information.
- Assistant with maintenance.
- The facility of software reuse
- Descovering unexpected thought
- Implimenta innovative processes for specific use.
-

have already occured.

Software maintenance is also important part of the software development life cycle. To update the software application and to all modifications in software application so as to improve performance is the main focus of software maintenance.

Need for Software Maintenance —
Software maintenence must be perform in order to —

- Correct fault
- Improve the design.
- Impliment inhancement.
- Interface with other system
- ~~Acto~~ Accommodate programs show that different hardware, software, system features and telecommunication facilities can be used.
- Migrate legacy software
- Retire Software.
- Requirement of user changes.
- Run the code fast.

Challenges in Software Maintenence

The various challenges in software maintenence are given —

maintenance but it gives rise to unforiseen future errors due to lack of topper impact analysis.

Adaptive Maintenance — This involves modifying the software system to adapt it to changes in hardware or software, government policies and business ~~toots~~ rules.

Perfective Maintenance — ~~Due to perfective form~~ This involves improving ~~perfor~~ functionality, performance & reliability the software system. to improve changability.

~~Trib. Trivent~~

Triventive Maintenence — This involves taking measures to prevent future problems such as optimisation, updating documentation, review and texting the system and implimenting preventive measures such as backup.

Maintenance can be categorised into proactive and reactive types. proactive maintenance involves taking preventive measures to avoid problems from occuring. while reactive maintenance involves addressing problem that

Page No. :
Date :

Advantages of Software Maintenance

- Improved software Quality
- Enhance security
- Extended Software Life.
- Cost saving.
- Better Alignment with business goals.
- Competitive advantage
- Improved Collaboration
- Reduced downtime
- Improved ~~Stability~~ Scalibility

Disadvantages of Software Maintenance

- Cost
- Complex Skill
- User Resistance
- ~~Compatab~~ Compatability Issues
- Lack documentation
- [redacted] Technical Debt
- Skills [redacted] Gap
- End of life

Uses of software reverse engineering

- Software reverse engineering is used in software design.
- Reverse engineering enables the developer or programmer to at new features to the software with or without knowing the source code
- Reverse engineering is also usefull in software testing, it helps the tester to detect the virus
- Malware analysis
- Legacy system
- Intelectual property protection
- Reverse engineering of software to create a competing product.
- Reverse engineering can be complex and time consuming process and it is important to have the necessary skills, tools and knowledge to perform it.

(f) Technical Debt
(g) Skill

0

(1) Which of the following is not a non-functional requirement
(a) Security (b) portability
(c) Scalability (d) User Interaction

(2) What is the other name of DFD?

(3) If every requirement manually check by a cost effective process, then SRS is called as
(a) Verifiable (b) traceable
(c) Modifiable (d) Complete

DATE / /
PAGE

~~Virus~~

(c)
(c) Malware Analysis
(d) Legecy System
(e) It Intellectual Property Propertion.
(f) Reverse Engineering of software to create a competing product.
(g) Reverse Engineering can be complex and time consuming process and it is important to have the necessery skills, tools and knowledge to perform it.

* Advantages of Software Maintenance.-

(a) Improve Software Quality.
(b) Enhanced Security.
(c) Standard Software Life (Extended)
(d) Cost Sharing
(e) Better Allignment with business goals.
(f) Competative Advantage
(g) Improve Collabration
(h) Reduce Downtime
(i) Improve Scalability

* Disadvantages of Software Maintenance.-

(a) Cost
(b) Complexity
(c) User Resistance
(d) Compatibity Issues
(e) Lack of Documentation

* Reverse Engineering - Process and of recovering the design and the specification of a product from an analysis of its code.

- Why Use Reverse Engineering -

(a) Providing proper system documentation.
(b) Recovery of lost information.
(c) Assisting with maintenance.
(d) Facility of
(e) Discovering unexpected fault
Implements innovative process for specific used
So

- Uses of Reverse Engineering -

Software reverse engineering used in software design, reverse engineering enables the developer or programmer to add new features to the software with or without knowing the source code.

(b) Reverse Engineering is also useful in software testing, it helps the tester to detect the virus.