Unit-5

Software Maintenance And Software Reuse

Software Maintenance: Software Maintenance,Maintenance process models,Maintenance Cost,Software Configuration Management Devops
Software Reuse : What Can Be Reused? Why Almost Non Reuse So Far?Basic Issues in Reuse Approach,Reuse at Organisation Level

Software Maintenance: It Is Very Broad Activity, Which Is Performed Once The Software System Becomes Operational. After The Product Has Been Released, The Maintenance Phase Keeps The Software Up To Date With Environment Changes And Changing User Requirements. Software Maintenance Covers Correction Of Errors; *Enhancements*, Deletion And Addition Of Capabilities; Adaptation To Changes In Data Requirements And Operation Environments; Improvement Of Performance Usability Or Any Other Quality Attribute. The Quality Of Software Artifacts Such As SRS, Architecture, Design Models, Code And Documentation Plays An Important Role In Making The Maintenance Process Effective And Efficient.

“Software Maintenance Is The Totality Of Activities Required To Provide Cost-Effective Support To A Software System. Activities Are Performed During The Pre-Delivery Stage As Well As The Post-Delivery Stage. Pre Delivery Activities Include Planning For Post-Delivery Operations Supportability And Logistics Determination. Post Delivery Activities Include Software Modification Training And Operating A Helpdesk. In This Definition The Concern Of Software Maintenance Is All About Fixing Bugs Or Mistakes.

Software Maintenance Is A Set Of Engineering Activities That Occur After The Software Has Been Deliverd To The Customer And Put Into Operation. These Are Various Hurdles That Can Slow Down The Maintenance Process. Some Of These Are An Unstructured Code That Is Too Difficult To Understand; Maintenance Programmers Having Insufficient Knowledge Of The System, Documentation Being Absent, Out Of Date, Or At Best Insufficient And Software Maintenance Having A Bad Image. The Success Of The Maintenance Phase Relies On These Problems Being Fixed Earlier In The Life Cycle.

Categories Of Maintenance:

Corrective maintenance
Adaptive maintenance
Perfective maintenance
Preventive maintenance
Emergency maintenance

Maintenance Process Model:

Several Authors Have Proposed Process Models For Software Maintenance. A Process Model Is The Representation Of The Progress Or Course Taken The Model Of The Process. These Process Models Organize Maintenance Into A Sequence Of Related Activities Or Phases And Define The Order In Which These Phases Are To Be Executed. Some Of These Models Are Traditional While Others Are Evolutionary.

Quick Fix Model:
The Quick Fix Model Is And Ad-Hoc Approach To The Maintenance Process/ Its Approach Is To Work On The Code First And Then Make Necessary Changes To The Accompanying Documentation. The Idea Is To Identify The Problem And Fix It As Early As Possible. After The Code Has Been Changed, It May Affect Requirement Design Testing And Any Other Form Of Available Documents Impacted By The Modification Should Be Updated. The Main Advantage Of This Model Is That It Gets Work Done Quickly With Lower Cost. Due To Time Constraint, This Model Does Not Pay Attention To The Long Term Effects. Changes Are Often Made Without Proper Planning, Design, Impact Analysis, And Regression Testing. There Is No Enough Time Tot Update Documents. Also Repeated Changes May Demolish The Original Design Thus Making Future Modification Progressively More Expensive To Carry Out.
Osborne's Model:
The Osborne's Model Is Concerned With The Reality Of The Maintenance Environment. This Model Assumes That The Technical Problems That Aaris E During Maintenance Are Due To Poor Communication Between And Control Of The Management. According To The Osborne Strategies Maintenance Requirements Need To Be Included In The Change Specification A Quality Assurance Program Is Required To Establish Quality Assurance Requirement And A Metrics Needs To Be Developed In Order To Verify That The Maintenance Goals Have Been Met. Managers Are Required To Provide Feedback Through Performance Reviews.

iterative enhancement model: This Model Is An Iterative Process Model. This Model Considers That Making Changes In A System Throughout Its Lifetime Is And Iterative Process. The Iterative Enhancement Model Assumes That The Requirements Of A System Cannot Be Gathered And Fully Understood Initially. The System Is To Be Developed In Builds. EachBuildCompletes Corrects And Refines The Requirements Of The Previous Builds Based On The Feedback Of Users.

Full Reuse Model: In The Full Reuse Model, Maintenance Is Considered As Reuse Oriented Software Development, Where Reusable Components Are Used For Maintenance And Replacements For Faulty Components. It Begins With Requirements Analysis And Design Of A New System And Reuses The Appropriate Requirements, Design Code And Test From The Earlier Versions Of The Existing System. The Reuse Documents And Components Defining Earlier Versions Of The Current System And Other Systems In The Same Application Domain. It Also Promotes The Development Of More Reusable Components.
IEEE 1219 Model: The IEEE Standard Organizes The Maintenance Process In Seven Phases As Demonstrated. The Phases Are Classification And Identification, Analysis, Design, Implementation, System Test, Acceptance Test And Delivery. Initially, Modification Requests Are Generated By The User, Customer, Programmer, Or The Manager Of The Maintenance Team.
ISO-12207 MODEL: The ISO-12207 Standard Organizes The Maintenance Process In Six Phases As Demonstrated. The Maintenance Phases Are Process Implementation, Problem And Modification Analysis, Modification Implementation, Maintenance Review/Acceptance, Migration And Software Retirement. The Process Implementation Phase Includes The Tasks For Developing Plans And Procedures For Software Maintenance Requests And Establishing An Organizational Interface With The Configuration Management Process. It Begins Early In System Life Cycle So That A Maintenance Plan Can Be Prepared In Parallel With The Development Plans. It Describes The Scope Of Maintenance , Identification And Analysis Of Alternatives And Selection Of The Maintenance Team.

Software Configuration Management:
Software Configuration Management Is Defined As A Process To Systematically Manage, Organize, And Control The Changes In The Documents, Codes, And Other Entities During The Software Development Life Cycle. It Is Abbreviated As The SCM Process In Software Engineering. The Primary Goal Is To Increase Productivity With Minimal Mistakes.

The Primary Reasons For Implementing Software Configuration Management System Are**:**

- There Are Multiple People Working On Software Which Is Continually Updating
- It May Be A Case Where Multiple Version, Branches, Authors Are Involved In A Software Project, And The Team Is Geographically Distributed And Works Concurrently
- Changes In User Requirement, Policy, Budget, Schedule Need To Be Accommodated.
- Software Should Able To Run On Various Machines And Operating Systems
- Helps To Develop Coordination Among Stakeholders
- SCM Process Is Also Beneficial To Control The Costs Involved In Making Changes To A System

Maintenance Cost:
The Maintenance Cost Is Around 60-85% Of The Total Development Life Cycle Cost. The Maintenance Cost Can Be Reduced If The Defects Are Taken Care Of In Earlier Phases Of Development. The Maintenance Cost Varies From Application To Application. Berry Boehm Has Proposed A Formula For Estimating The Maintenance Cost As Part Of The COCOMO Estimation Model. His Maintenance Cost Model Uses A Quantity Called Annual Change Traffic(ACT). ACT Is Defined As "The Fraction Of A Software Product's Source Instructions Which Undergoes Change During A Year Through Addition, Deletion Or Modification" It Considers KLOC For Maintenance Estimation. ACT Is Related To The

Number Of Change Requests As Follows:
ACT=(Klocadd+Klocdeleted)/(Kloctotal)

Where Klocadded Is The Total Lines Of Code Added During Maintenance. Klocdeleted Is The Total Lines Of Code Deleted During Maintenance. The Maintenance Effort In Person-Months Can Be Calculated As Follows:
Maintenance Effort=ACT X Development Effort
Software Projects Have Different Characteristics. The Maintenance Effort Varies From Project To Project. Therefore Effort Adjustment Factors(EAF) Are Considered For Accurate Estimation Of The Maintenance Effort. Considering Eafs, The Maintenance Effort Is Calculated As Follows:
Maintenance Effort=ACT * Development * EAF

Software Reuse:
The Development Of Faster,Better And Cheaper Software Is A Significant Challenge For The Software Industry. A Possible Way To Produce Software Products In This Way Is Development With Software Reuse. Software Reuse Is An Efficient Way Of Implementation Or Updating Software Systems With Existing Software Components. Software Reuse Is The Process Of Producing Software Systems From Existing Software Systems Rather Than Building Reusable Components Only Once And Saves Development Effort Multiple Times.
Software Reuse Has A Great Impact On Achieving High Productivity And Quality. Also, If The Requirements Change, The Required Component Can Be Added And Software Can Be Modified. Reuse Based Development Using Components/Modules Improves The Maintainability Of Software. The Production Of Reusable Components Requires More Effort

And Resources Than The Development Of Components Customized For One Project. Reuse Requires An Extensive Upfront Planning Effort To Identify And Avail Reusable Components. Also It Requires Extra Time For Writing , Managing And Retrieving Reusable Components, Thereby Reducing The Benefits From Writing Non-Reusable Components. Although The Benefits Of Systematic Reuse Can Be High, Most Organizations Lack Success In Implementation And Getting Reuse Potential.

What is software reuse?

Software reuse is a term used for developing the software by using the existing software components. Some of the components that can be reuse are as follows; • Source code • Design and interfaces • User manuals • Software Documentation • Software requirement specifications and many more. What are stages of reuse-oriented software engineering? Requirement specification: First of all, specify the requirements. This will help to decide that we have some existing software components for the development of software or not. Component analysis Helps to decide that which component can be reused where. Requirement updations / modifications. If the requirements are changed by the customer, then still existing components are helpful for reuse or not. Requirement updations / modifications. If the requirements are changed by the customer, then still existing components are helpful for reuse or not. Reuse System design If the requirements are changed by the customer, then still existing system designs are helpful for reuse or not. Development Existing components are matching with new software or not. Integration Can we integrate the new systems with existing components? System validation To validate the system that it can be accepted by the customer or not.

Software Reuse Success Factors

1. Capturing Domain Variations 2. Easing Integration 3. Understanding Design Context 4. Effective Teamwork 5. Managing Domain Complexity What are the advantages of software reuse? 1. Less effort: Software reuse requires less effort because many components use in the system are ready made components. 2. Time-saving: Re-using the ready made components is time saving for the software team. 3. Reduce cost: Less effort, and time saving leads to the overall cost reduction. 4. Increase software productivity: when you are provided with ready made components, then you can focus on the new components that are not available just like ready made components. 5. Utilize fewer resources: Software reuse save many sources just like effort, time, money etc. 6. Leads to a better quality software: Software reuse save our time and we can consume our more time on maintaining software quality and assurance.

Basic issues in any reuse program

The following are some of the basic issues that must be clearly understood for starting any reuse program. • Component creation • Component indexing and storing • Component search • Component understanding • Component adaptation • Repository maintenance Component creation- For component creation, the reusable components have to be first identified. Selection of the right kind of components having potential for reuse is important. Domain analysis is a promising technique which can be used to create reusable components. Component indexing and storing- Indexing requires classification of the reusable components so that they can be easily searched when looking for a component for reuse. The components need to be stored in a Relational Database Management System (RDBMS) or an Object-

Oriented Database System (ODBMS) for efficient access when the number of components becomes large. Component searching- The programmers need to search for right components matching their requirements in a database of components. To be able to search components efficiently, the programmers require a proper method to describe the components that they are looking for. Component understanding- The programmers need a precise and sufficiently complete understanding of what the component does to be able to decide whether they can reuse the component. To facilitate understanding, the components should be well documented and should do something simple. Component adaptation- Often, the components may need adaptation before they can be reused, since a selected component may not exactly fit the problem at hand. However, tinkering with the code is also not a satisfactory solution because this is very likely to be a source of bugs. Repository maintenance- A component repository once is created requires continuous maintenance. New components, as and when created have to be entered into the repository. The faulty components have to be tracked. Further, when new applications emerge, the older applications become obsolete. In this case, the obsolete components might have to be removed from the repository.