B.E.

Sixth Semester Examination, May-2009

Principles of Software Engineering (CSE-302-E)

Note : Attempt any *five* questions. All questions carry equal marks.

Q. 1. (a) Describe the term software crisis. What are the causes associated with it? How it led to the development of software engg. as a discipline?

Ans. Crisis of Software : Crisis of software means that problem that lies with the software. There are so many problems that lie with software like cost, quality, time boundation etc. The main problem with the software is rechange because the technologies changes daily so the software changes according to requirement. It is the major task to change the software.

There are the following reasons due to which the software crisis arise :

(a) Software is Expensive : Over the past decades with the advancement of technology, the cost of hardware has consistently decreased. On the other hand the cost of software is increasing.

The main reasons for the high cost of software is that software development is still labour-intensive. To get an absolute idea of the costs involved. Let us consider the current state of the practice in the industry. Delivery lines of code is by for the most commonly used measure of software size in the industry.

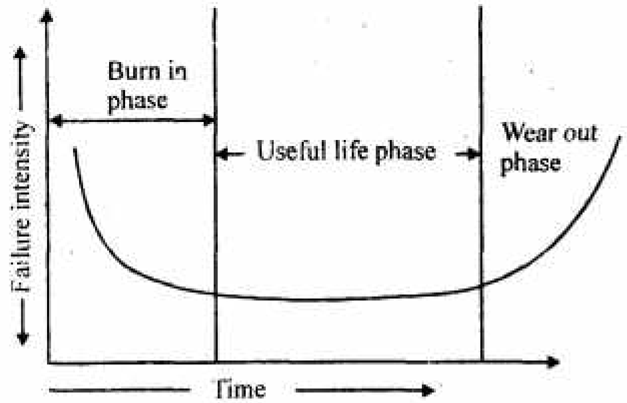
(b) Late, Costly and Unreliable : There are many instances quoted about software projects that are behind schedule and have heavy cost overruns. The software industry has gained a reputation of not being able to deliver on time and within budget.

A survey reported in stages that of the 600 firms surveyed, more than 35% reported having some computer related development project that they categorized as a runaway and a runaway is not a project that is somewhat late or somewhat over budget. It is one where the budget and schedule are out of control. The problem has become so serve that it has spawned and industry of its own.

Overall the software industry has gained a reputation of delivering software within schedule and budget and not of producing software system of poor quality.

(c) Problem of Change and Network : When the software is delivered and deployed it enters the maintenance phase. All system need maintenance but for other system it is largely due to problems that are introduced due to aging. Software needs to be maintained not because some of its components wear out and need to be replaced, but because there are often some residual errors remaining in the system that must be removed as they are recovered.

Even without bugs software frequently undergoes changes. The main reason is that software often must be upgraded and enhanced to include more features and provide more services. This also requires modification of the software. It has been argued that once a software system is deployed, the environment in which it operates changes. Hence the needs that initiated the software development also change to reflect the needs of the new environment. Hence, the software must adopt to the needs of the changed environment. The changed software than changes the environment which in turn requires further changes. This phenomenon is sometimes called the law of software evaluation.



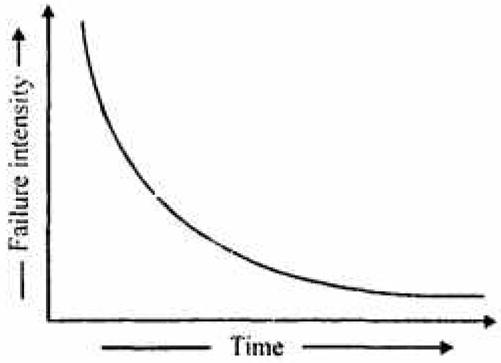
Bath Tub Curve

Maintenance due to this phenomenon is sometimes called adaptive maintenance.

Causes Associated :

(i) Software Does Not Wear Out : There is a well known 'bath tub curve' in reliability studies for hardware products. The curve is given in fig. The shape of the curve is like "bath tub curve" and is known as bath tub curve.

There are three phases for the life of a hardware product. Initial phase is burn in phase where failure intensity is high. It is expected to test the product in the industry before delivery. Due to testing and fixing faults, failure intensity will come down initially and may stabilise after certain time. The second phase is the useful life phase where failure intensity is approximately constant and is called useful life of a product. After few years again failure intensity with increase due to wearing out of components. This phase is called wear out phase. We not have this phase for the software as it does not wear out. The curve for software is given in fig.



Software Curve

Important point is software becomes reliable overtime instead of wearing out. It becomes obsolete if the environment for which it was developed, changes. Hence software may be retired due to environmental changes, new requirements, new expectations etc.

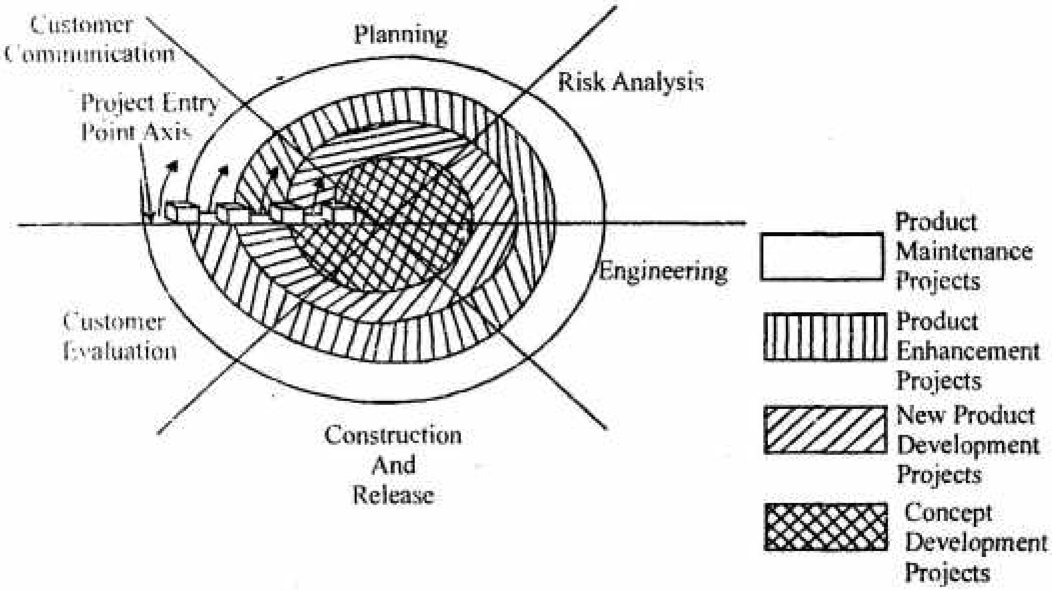
(ii) Software is not Manufactured : The life of a software is from concept exploration to the retirement of the software product. We do not have assembly line in software development. Hence, its not manufactured in the classical sense.

(iii) Reusability of Components : In software every project is a new project. We start from the scratch and design every unit of the software product. In software there is only a humble beginning like graphical user in faces are built using reusable components that enable to creation of graphics windows, pull down menus and a variety of interaction mechanisms.

(iv) Software is Flexible : We all fell that software is flexible. A program can be developed to do almost anything. Sometimes this characteristics may be the best and may help us to accommodate any kind of change. However most of the times this "almost anything" characteristics has made software development difficult to plan, monitor and control. This unpredictability to the basis of what has been referred to for the past 30 years as the "software crisis."

Q. 1. (b) What are the major phases in the spiral model of software development? Explain.

Ans.



Depicts a spiral model that contains six task regions

Spiral model is a life cycle model. In the spiral model, developers define and implement features in order of decreasing priority.

There are four phase in the "spiral model" which are : Planning, Evaluation, Risk Analysis and Engineering. These are four phases are interactively followed one after other in order to eliminate all the problems, which were faced in "The waterfall model." Interacting the phases helps in understanding the problems associated with a phase and dealing with those problems when the same phase is repeated next time, planning and developing strategies to be followed while iterating through the phases.

The Phases in "Spiral Model" are :

Plan : In this phase the objectives, alternatives and constraints of the project are determined and are documented. The objectives and other specifications are fixed in order to decide which strategies/approaches to follow during the project life cycle.

Risk Analysis : The phase is the most important part of 'spiral model'. In this phase all possible alternatives, which can help in developing a cost effective project are analyzed and strategies are decided to use them. This phase has been added specially in order to identify and resolve all the possible risks in the project development. If risks indicate any kind of uncertainly in requirements, prototyping may be used to proceed with the available data and find out possible solution in order to deal with the potential changes in the requirements.

Engineering : In this phase the actual development of the project is carried out. The output of this phase is passed through all the phases iteratively in order to obtain improvements in the same.

Customer Evaluation : In this phase developed product is carried out. The passed on to the customer in order to receive customer's comments and suggestions which can help in indentifying and resolving potential problems/errors in the software developed. The phase is very much similar to TESTING phase.

Q. 2. (a) Discuss the common sources and types of risks in software development projects and strategies to deal with them.

Ans. Definition of Software Risk : Webster's Ninth New Collegiate Dictionary defines risk as a possibility of loss. The definition of risk in the software engineering environment that we will use in exposure to harm as this includes not only the possibility of risks but their impact as well.

Software development risks could be classified into several categories. There are :

(i) Requirements Risk : Unclear requirements introduce large risks. This is the most common type of risk and is probably responsible for most failed or delayed projects. Competitive forces, business agreements with new partners force the organization and its software systems to change. Besides this users find it hard to visualize software until they see it. This makes requirements highly unclear and subject to change. The best way to handle this risk is to adopt a flexible development process.

(ii) Technology Risk : You may find that the technology being used is unable to satisfy the system requirements. For example, we could assume that the database. We use is transactional but when actually building the system. We find that it is not transactional under certain circumstances that could happen frequently.

(iii) Business Risk : There are risks introduced by business decisions. For example, a deal with a vendor does not get signed in time to use the platform that we will deploy the application on. A conflict with a partner who supplies part of the solution prevents us from proceeding with the project or the partner goes out of business.

(iv) Political Risk : These are the most difficult to overcome large organizations tend to behave like large families members are busy jockeying for power and influence over each other. Unfortunately, your project may be threatening to some groups and they will protect themselves.

(v) Resource Risk : You do not get the people, money facilities when needed. This is something that probably all of us have experienced. It usually has bad affects on the projects schedule and the project team's health. Planning for this may involve identifying alternate resource sources that may able to help. For example another team may be wiling to share some of their severs till yours come in.

(vi) Skills Risk : Your team does not have all the requisite skills to do the job. For example, they may be unfamiliar with the technology used or they may not be familiar with the business process and so on. Bringing in external consultants who can help get you quickly up to speed can mitigate this risk.

(vii) Deployment and Support Risk : The project may not be deployable when ready since the required infrastructure is not in place. The support team may not be ready for training or are over stretched. Working closely with the developing and support teams can help reduce this risk. Often this happens because these folks have no idea what you need or are doing.

(viii) Integration Risk : Most applications need to integrate with other application. Miscommunication and misunderstandings cause systems not to share accepted interfaces and they do not work together as expected. Communication is key to reducing this risk.

(ix) Schedule Risk : Schedule may contain conflicts components are not available when necessary, the delivery date is happening at an extremely busy time. Communication between all the interested parties can help reduce this risk.

(x) Maintenance and Enhancement Risk : The project cannot be maintained and enhanced properly because of inadequate documentation, the support team was not properly trained or the technical platform is obsolete. Good planning and time spent for training documentation can help reduce these risks.

(xi) Design Risk : Bad design decisions have an impact on the systems performance or it ability to satisfy the requirements. A bad design could make the software unusable for the user.

(xii) Other Risk : This is a catch all for risks that are hard to foresee and predict. Examples could be a hurricane shutting down your offices for a week, a fire in the building, a development server crashing. Virus attacks and so on. Since they are hard to foresee that can be hard to plan for.

Q. 2. (b) What are the core functions associated with project management? Explain.

Ans. The core functions associated with project management are :

(i) Scope. (ii) Quality. (iii) Time & 4 cost.

Scope and Quality : If specification of technological aspects of the project such as engineering, manufacturing or constructibility, are to be reviewed, such as investigation must clearly be conducted by those throughly conversant with the project's technology. In addition most projects today have some degree of recognizable environmental, social or safety impacts. If these have not already been analyzed & arrangements made for monitoring & mitigation, then persons with corresponding knowledge & experience must undertake such review.

Even so, certain general management questions can be formulated with regard to the technical scope & quality of the project.

For example, is the slope consistent with the projects goals & are these goals consistent with current market opportunities.

Have the major stakeholders been involved in this development process & are they required to sign off on the plan prior to implementation?

In Case of Quality : Has the project's executive given priority to building the required quality standards into the project planning & execution process right from the outset? etc.

Schedule & Cost : Similarly specific questions can be posed regarding schedule & cost.

For Example, Do project plans include a milestone schedule indicating major pieces of work to be accomplished & who will be responsible for each?

Are project schedules time estimate & logic developed using input from members of the project team, in order to build in commitment?

The cost situation should be similarly examined & the direct impacts on the cost situation of any changes in the schedule also recognized.

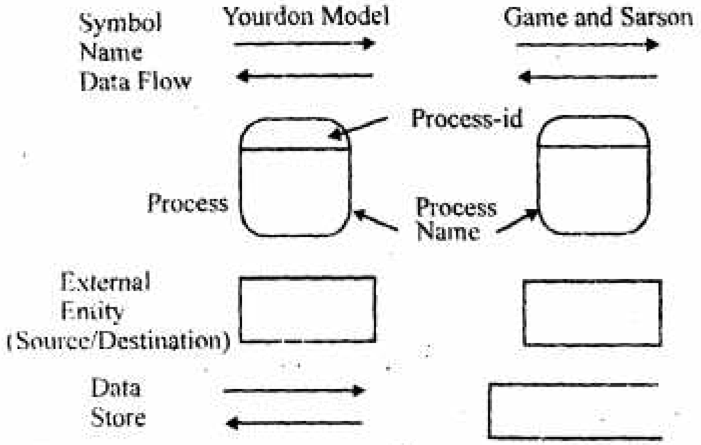
The typical cost questions should include : Is the estimate realistic, including both direct & indirect costs of all required resources. Is project financially sound based?

Q. 3. (a) Describe the basic components of a DFD (Data Flow Diagram). Make a DFD to describe library information system.

Ans. Data Flow Diagrams : During the analysis and designing of a system, it is necessary to study and investigate the way in which the data is flowing through various processes within scope of a system. It is necessary to find out how the data is originated, used or referred changed, stored and where it is shown as output.

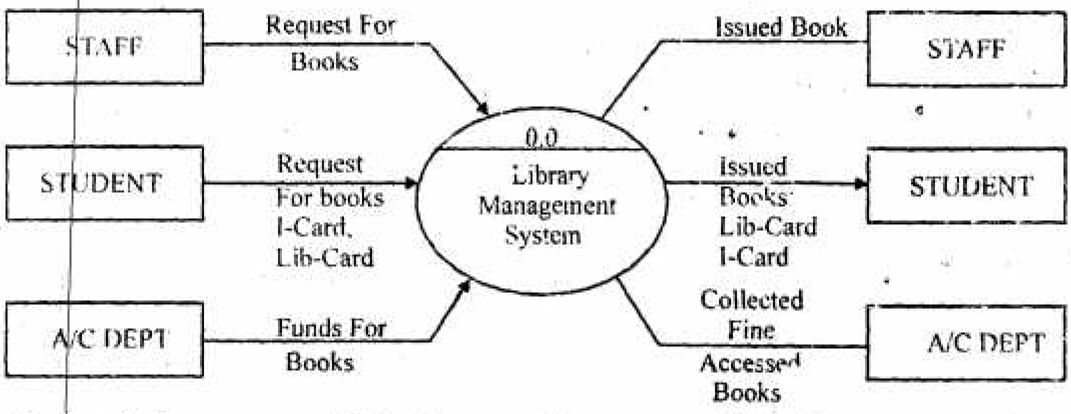
A data flow diagram is one of the popular graphical tools used to depicts the flow of data through a system. DFD shows the processes. Data stores, Data flow and the source and destination entities.

Yourdon Game and Sarson have suggested specific symbols to draw DFDs.



Symbols used to draw DFDs

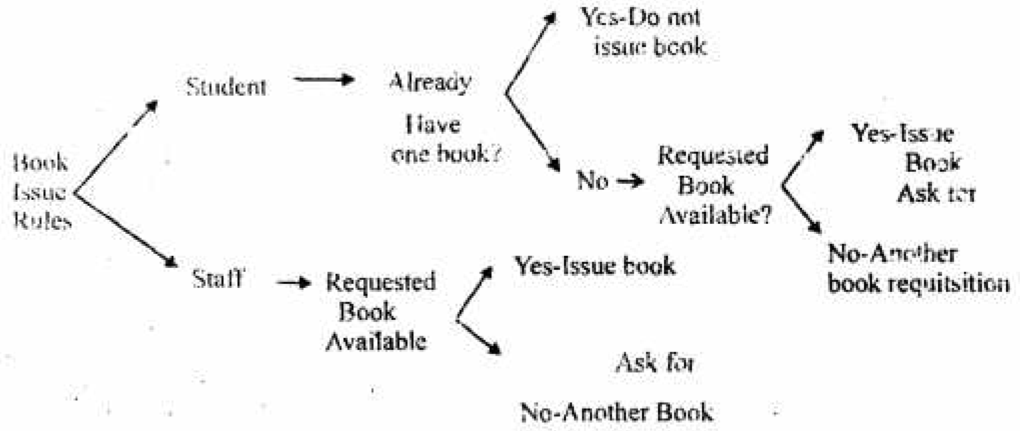
(a) Library Management System : A library management system is to be developed to assist and manage a library system efficiently. The main objective of the LMS is to provide timely information for the users. The systems keeps track of the issuing/returning, purchasing and maintenance of the books and also the amount of fine collected from the students in case of late return of the book.



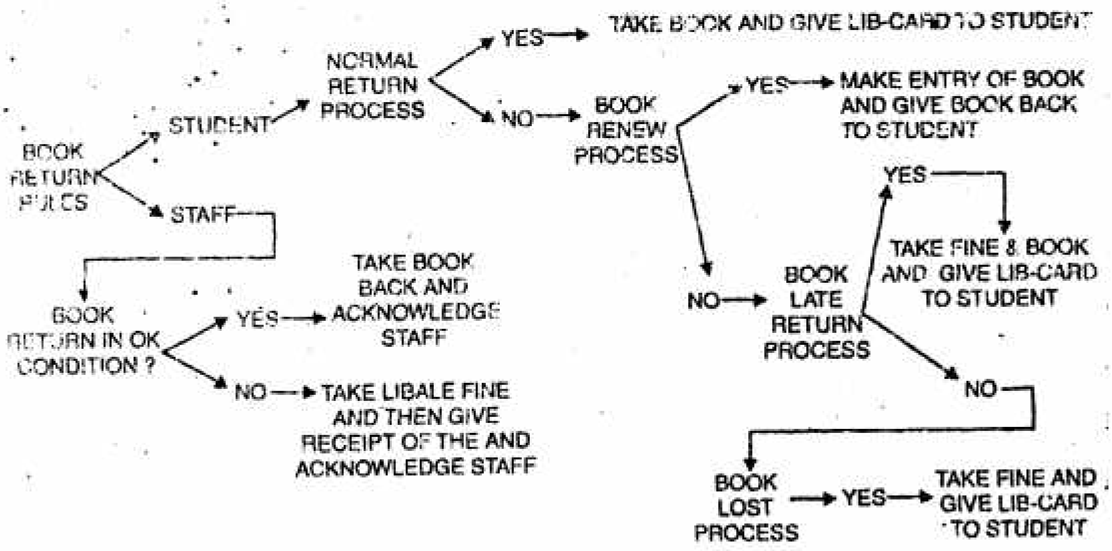
(i) Data dictionary
(ii) Student personal
(iii) Staff personal
(iv) Staff Library
(v) Book department
(vi) Book entry
(vii) Acc. Table
(viii) Finance
(ix) TSN.

(b) Decision Tree :

(i) Decision tree for book issue process :



(ii) Decisions tree for book return process :


BOOK RETURN RULES
STUDENT
NORMAL RETURN PROCESS
YES
TAKE BOOK AND GIVE LIB-CARD TO STUDENT
NO
BOOK RENEW PROCESS
YES
MAKE ENTRY OF BOOK AND GIVE BOOK BACK TO STUDENT
NO
BOOK LATE RETURN PROCESS
YES
TAKE FINE & BOOK AND GIVE LIB-CARD TO STUDENT
NO
BOOK LOST PROCESS
YES
TAKE FINE AND GIVE LIB-CARD TO STUDENT
STAFF
BOOK RETURN IN OK CONDITION ?
YES
TAKE BOOK BACK AND ACKNOWLEDGE STAFF
NO
TAKE LIBALE FINE AND THEN GIVE RECEIPT OF THE AND ACKNOWLEDGE STAFF

Decision tree for book return process

(iii) Decision table.

Q. 3. (b) Describe the term "requirement elicitation." Discuss any two techniques in detail, used for it.

Ans. Requirements Elicitation : Requirements elicitation is perhaps the most difficult, most critical, most error prone and most communication intensive aspect of software development. It is the activity that helps to understand the problems to be solved. Requirements are gathered by asking questions, writing down the answers, asking other questions etc. Hence, requirements gathering is the most communications intensive activity of software development. Developers and users and different mind set, expertise and vocabularies. Due to communication gap, there are chances of conflicts that may lead to inconsistencies, misunderstanding and omission of requirements.

Therefore requirements elicitation requires the collaboration of several groups of participants who have different background.

Two Techniques are :

(a) Brainstorming Sessions : Brainstorming is a group techniques that may be used during requirements elicitation to understand the requirements. The group discussions may lead to new ideas quickly and help to promote creative thinking.

Brainstorming has become very popular and is being used by most of the companies. It promotes creative thinking, generates new idea and provides platform to share views, apprehensions expectations and difficulties of implementation. This group techniques may be carried out with specialised group like actual users, middle level managers etc., or with total stakeholders.

Every idea will be documented in such a way that everyone can see it. While boards overhead transparencies or a computer projection system can be used to make it visible to very participant. After session, a detailed report will be prepared and facilitator will review the report.

(b) Facilitated Application Specification Technique : This approach is similar to brainstorming sessions and the objective is to bridge. The expectation gap a difference between what developers think they are supposed to build and what customers think they are going to get. In order to reduce expectation, gap, a team oriented approach is developed for requirements gathering and is called facilitated application specification technique.

This approach encourages the creation of joint team of customers and developers who together to understand the expectations and propose a set of requirements. The basis guidelines for FAST are given below :

(i) Arrange a meeting at a neutral site for developers and customers.

(ii) Establishment of rules for preparation and participation.

(iii) Prepare an informal agenda that encourages free flow of ideas.

(iv) Appoint a facilitator to control the meeting. A facilitator may be a developer, a customer or an outside expert.

(v) Prepare definition mechanism-board, flip charts, worksheets, wall strikes etc.

(vi) Participants should not criticize or debate.

Q. 4. (a) Describe the difference between cohesion and coupling. Explain various types of cohesion. Which one is the best and which one is the worst?

Ans. Cohesion : Cohesion of module represents how tightly bound the internal elements of then module is to one another. Cohesion of a module gives the designer an idea about whether the different elements of a module belong together in the same module. Cohesion and coupling are clarity related usually, the greater the cohesion of each module in the system, the lower the coupling between the modules.

There are Several Levels of Cohesion :

(i) Coincidental,

(ii) Logical,

(iii) Temporal

(iv) Procedural,

(v) Communication

(vi) Sequential

(vii) Function.

(i) Coincidental Coincidental is the lowest level and functional is the highest. Coincidental cohesion occurs when there is no meaningful relationship among the elements of a module.

(ii) Logical : A module has a logical cohesion if there is some logical relationship between the elements of a module and the elements perform the function fall same logical class.

(iii) Temporal : Temporal cohesion is the same as logical cohesion except that the elements are also related in time and are executed together. Modules that perform activities like 'initialization', clear-up and 'termination' are usually temporarily bound.

(iv) Procedural : A procedurally cohesive module contains elements that belong to a common procedural unit.

(v) Communicational : A module with the communication cohesion has elements that are related by a reference to the same input output data.

(vi) Sequential : When the element is together in a module because the output of one forms the input to another, we get sequential cohesion.

(vii) Function : In a functionally bound module, all the element of the module are related to performing single function by function, we do not mean simply mathematical functions, module accomplishing a single goal are also included. Function like "Compute square root" and "Sort an array" are clear example of functional cohesive module.

Coupling : It is an indication of the strength of interconnections between the components in a design highly coupled systems have strong interconnections with program units dependent on each other. Loosely coupled systems are made up of components which are independent.

Types of Coupling ;

(i) Data coupling.

(ii) Stamp coupling.

(iii) Control coupling.

(iv) Common coupling.

(v) Content coupling.

Q. 4. (b) Differentiate between a database and data warehouse. What is the utility of data warehouses in current scenario?

Ans. Database :

(i) It is the place where the data is taken as a base & managed to get available fast efficient access.

(ii) Database is a reservoir having operational data needed for daily business & they are ever changing. Since the new data is always over written the old one there is no way to compare the change in data this is where are data warehouse comes in picture.

(iii) Database support application data storage access & retrieval of data. It does not support decision making & business intelligence.

(iv) Database is used to running the business.

Data Warehouse :

(i) Data warehosue is the place where the application data is managed for analysis & reporting purposes.

(ii) Data warehouse has historical datas & the current data so comparisons becomes very easy for business analysis.

(iii) It support application storage access & retrieval of data & also support decision making & business intelligence.

(iv) DWH is how to run the business.

Utility : The DW environment is treated as a dynamic capability, providing the capacity for managing data resources & turning them into useful information process. These products contribute value when used for exploitative and/or explorative business processes. DW capacities are evaluated as real option investments toward the development of a framework for modeling cost utility effects of DW design decisions.

Q. 5. (a) Describe equivalent class partitioning as used in software testing.

Ans. There are essentially the following two main approaches to designing black-box tests cases :

(i) Equivalence class partitioning

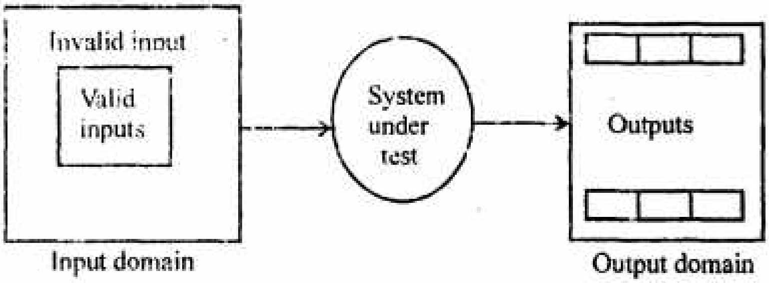
(ii) Boundary value analysis

(i) Equivalence Class Testing : In this method input domain of a program is partitioned into a finite number of equivalence classes such that one can reasonably assume, but not absolutely sure, that the test of a representative value of each class is equivalent to a test of any other value. That if one test case in a class detects an errors, all other test cases in the class would be expected to find same error. Conversely, if a test case did not detect an error we would expect that no other test cases in the class would find an error. Two steps are required in implementing this method.

(i) The equivalence classes are identified by taking each input condition and partitioning it into valid and invalid classes. For example, if an input condition specifies a range of values from-1-999 we identify one valid equivalence class [item ≤ 999] and two invalid equivalence classes [item < 1] and [item > 999].

(ii) Generate the test cases using the equivalence classes identified in the previous step. This is performed by writing test cases covering all the valid equivalence classes. Then a case is written for each invalid equivalence class so that no test contains more than one invalid class. This is to ensure that no two invalid classes mask each other.

In fig. both valid and invalid input domains are shown.



Equivalence partitioning

Q. 5. (b) Differentiate between :

(i) Alpha and Beta Testing

(ii) Verification and Validation.

Ans. (i) Alpha and Beta Testing : The terms alpha and beta testing are used when the software is developed as a product for anonymous customers. Hence, formal acceptance testing is not possible in such cases. However, some potential customers are identified to get their views about the product. The alpha tests are conducted at the developers site by a customers. These tests are conducted in a controlled environment. Alpha testing may be started when formal testing process is near completion.

The beta tests are conducted by the customers/end users at their sites. Unlike alpha testing, developers is not present here. Beta testing is conducted in a real environment that cannot be controlled by the developer. Customers are expected to report failures if any to the company.

Most of the companies are following this practice. Firstly, they send the beta release of their product for few months. Many potential customers will use the product and may send their views about the product. Some may encounter with failure situations and may report to the company. Hence, company gets the feedback of many potential customers.

(ii) Verification and Validation : These terms are often used interchangeably but have different meaning IEEE has given the definitions of both these which are being widely accepted by the software engineering community.

Verification is primarily related to manual testing because it requires looking at documents and reviewing them. However, validation usually requires the execution of program.

Verification : As per IEEE/ANSI. "It is the process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase." Hence verification activities are applied to early phases of SDLC such as requirements design, planning etc. We check the documents generated after the completion of every phase in order to ensure that what comes out of that phase is what we expected to get.

Validation : As per IEEE/ANSI. "It is the process of evaluating a system or component during or at the end of development process of determine whether it satisfies the specified requirements." Therefore validation requires actual execution of the program and is also known as computer based testing. We experience failures and identify the causes of the failures.

Hence testing includes both verification and validation.

Testing = Verification + Validation

Both are important and complementary to each other. Verification minimises the errors and their impact in the early phases of development. If we find more errors before execution, validation may be comparatively easy. Unfortunately testing is primarily validation oriented.

Q. 6. (a) Describe the role of formal technical review (FTR) as a quality assurance activity. How is it conducted?

Ans. The Formal Technical Review (FTR) :

(i) Design is important because it allows a software team to assess the quality of the software before it is implemented at a time when errors, omission or inconsistencies are easy and in expensive to correct. But how do we assess quality during design. The software can't be tested because there is no executable software to test. What to do?

During design, quality is assessed by conducting to series of formal technical reviews (FTRs).

(ii) Many different types of reviews can be conducted as part of software engineering. Each has its place. An informal meeting around the coffee machine is a form of review, if technical problems are discussed. A formal presentation of software design to an audience of customers, management and technical staff is also a form of review. In this book, however, we focus on the formal technical review, sometimes called a walkthrough or an inspection. A formal technical review is the most effective filter from a quality assurance standpoint. Conducted by software engineers for software engineers, the FTR is an effective means for uncovering errors and improving software quality.

In can be Conducted : An FTR is a meeting conducting by members of the software team. Usually two, three or four people participate depending on the scope of the design information to be reviewed. Each person plays a role, the review leader plans the meeting, sets an agenda and then runs the meeting, the recorder takers notes so that nothing is missed the producer is the person whose work product is being reviewed. Prior to the meeting each person on the review team is given a copy of the design work product and is asked to read it, looking for errors, omission or ambiguity. When the meeting commences, the intent is to note all problems with the work product so that they can be corrected before implementation beings. The FTR typically lasts between 90 minutes and two hours. At the conclusion of the FTR, the review team determines whether further actions are required on the part of producer before the design work product can be approved as part of the final design model.

***Q. 6. (b) What is CASE? How it supports software life cycle?**

Ans. Computer Aided Software Engineering Tools are similar to a crafts woman's tool set in the sense that they.

(i) The tools help in every step of the problem solving.

(ii) The tools are organized so that they (process) are easy to find and use.

(iii) The craftswoman can use the tools in an effective and efficient manner.

CASE tools can be classified by :

(i) By function or

(ii) By user type or

(iii) By stage in SE process.

The following taxonomy is classified by function :

Business Information Tools : Model business information flow. The tools represent business data object and model their flow. Network tools could be modified to yield the same functionality.

Process Management Tools : Model processes. You have to understand the process in order to model it. The tools help you capture process. This is a key component of TQI.

Project Planning Tools : Help plan and schedule projects. Example are PERT and CPM. Finding parallelism and eliminating bottlenecks assist in streamlining projects.

Risk Analysis Tools : Help build a risk table from the schedule outlining the risk each component of the production process and categorizing the risk as catastrophic, critical, marginal or negligible. A cost is associated with each risk.

Project Management Tools : Track the progress the project. They feed from the project scheduling tools and then use those tools to update plans and schedules.

Requirements Tracing Tools : Provide a systematic to isolate customer requirements and then to trace these requirements in each stage of development. In particular one can take implementation code and point to the requirement that is met by the code.

Metrics Tools : Capture specific metrics that provide an overall measure of quality. Example could be defects per function point. "LDC/person-month" and so forth.

Documentation Tools : Include word processors that give templates for the organization process documents.

System Software Tools : Includes email, bulletin boards and www access.

Quality Assurance Tools : Are actually metrics tools that audit source code to insure compliance with language standards.

Database Management Tools : Provides consistent interfaces for the project for all data, in particular the configuration objects are primary repository elements.

Software Configuration Management Tools : It is the keystone to CASE. It assists with identification version control, change control, auditing and status accounting.

Analysis and Design Tools : Create models of the system. Some create formal models. Others construct data flow models.

Pro/Sim Tools : Are prototype and simulation tools. They can help predict real time system response and allow mock-ups of such, systems to be fashioned.

Interface Design and Development Tools : Include, Motif, TCl, Java, Visual XXXX.

Programming Tools : Include compilers, debuggers, profiles, application, generators, database query languages and GUI development tools.

Static Analysis Tools : Assist the developer in driving test cases. These include code-based testing tools and requirements based testing tools.

Dynamic Testing Tools : Interact with an executing program-checking path coverage and testing assertions. Instructive tools insert code in the tested program. Non-intrusive tools use a separate hardware processor.

Test Management Tools : Manage and coordinate regression testing, perform comparisons of output and act as test drivers.

Q. 7. (a) Why CASE approach is recommended in case of large complex software solution? Comment how CASE approach affects the following :

(i) Documentation,

(ii) Programming effort.

Ans. CASE (Computer Aided Software Engineering) is the software used for the automated development of system software.

CASE is recommended for large & complex projects because it automates project management activities, manage all work products, produced throughput the process & assist engineers in their analysis, design, coding & testing work. Developing all the issues manually in such a complex cases is very difficult, time taking & even expensive.

So, CASE is recommended in case of large complex software solution.

CASE approach gives a positive affect on all issues by using various tools. For each task a separate tool is used & it automates the work & give better results :

(i) Documentation : Using the documentation tool they provide opportunities for improved productivity by reducing the amount of time needed to produce work products.

(ii) Programming Efforts : Programming efforts are also reduced I uses various kinds of programming tools & reduces the manual efforts.

Q. 7. (b) Describe Re-engineering and Reverse Engineering and differentiate between the two.

Ans. Reengineering : Cyclomatic complexity analysis provides knowledge of the structure of operational code of a system. The risk involved in reengineering a piece of code is related to its complexity. Therefore cost and risk analysis can benefit from proper application of such as analysis.

Difference between Validation and Verification : The input of verification are checklists, issue lists, walk throughs and inspection meetings, reviews and meetings. The input of validation is the actual testing of an actual product.

Software re-engineering is concerned with taking existing legacy systems and reimplementing them to make them more maintainable. As a part of this re-engineering process, the system may be redocumented. It may be translated to a more modern programming language, implemented on existing hardware technology. Thus, software re-engineering allows us to translate source code to a new language, restructure our old code, migrate to a new platform, capture and them graphically display design information and re-document poorly documented systems.

Reverse Software Engineering : Software reverse engineering involves reversing a program's machine code back into the source code that it was written in using program language statements. Software reverse engineering is done to retrieve the source code of a program because the source code was lost to study how the program performs certain operations to improve the performance of a program to fix a bag, to identify malicious content in a program such as a virus or to adapt a program a written for use with one microprocessor for use with a differently-designed microprocessor. Reverse engineering for the sole purpose of copying or duplicating programs constitutes a copyright violation and is illegal. In some cases, the licensed use of software specifically prohibits reverse engineering. The term "reverse engineering" as applied to software means different things to different people prompting chikofsky and cross to write a paper researching the various uses and defining a taxonomy. From their paper.

Reverse engineering is the process of analyzing a subject system to create representations of the system at a higher level of obstruction. It can also be seen as "going backwards through the development cycle." In this model the output of the implementation level of obstruction phase is reverse engineered back to the analysis phase, in an inversion of the traditional waterfall model.

Reverse engineering is a process of examination only : the software system under consideration is not modified.

In practice two main types of reverse engineering emerge. In the first case, source code is already available for the software but higher level aspects of the program, perhaps poorly documented but no longer valid, are discovered. In the second case, there is no source code available for the software and any efforts towards discovering one possible source code for the software are regarded as reverse engineering.

Q. 8. (a) Discuss how the reliability changes over the life time of a software product and hardware product.

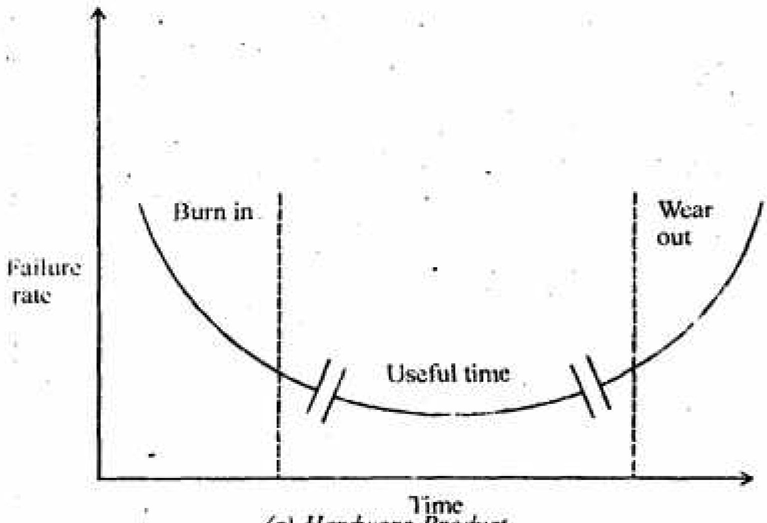
Ans. Software reliability is the probability of failure free software operation for a specified environment software reliability is also an important factor affecting system reliability. It differs from hardware reliability in that it reflects the design perfection, rather than manufacturing perfection. The high complexity of a software is the major contributing factor of software reliability problem.

Good engineering methods can largely improve software reliability. Before the deployment of software products, testing, verification & validation are necessary steps. Software testing is heavily used to trigger, locate & remove software defects. Software testing is still in its infront state; testing is crafted to suit specific needs in various software development projects in an adhoc-manner. Various analysis tools such as trend analysis fault tree analysis, orthogonal defect classification & formal methods etc. can also be used to minimize the possibility of defect occurrence after release & therefore improve software reliability.

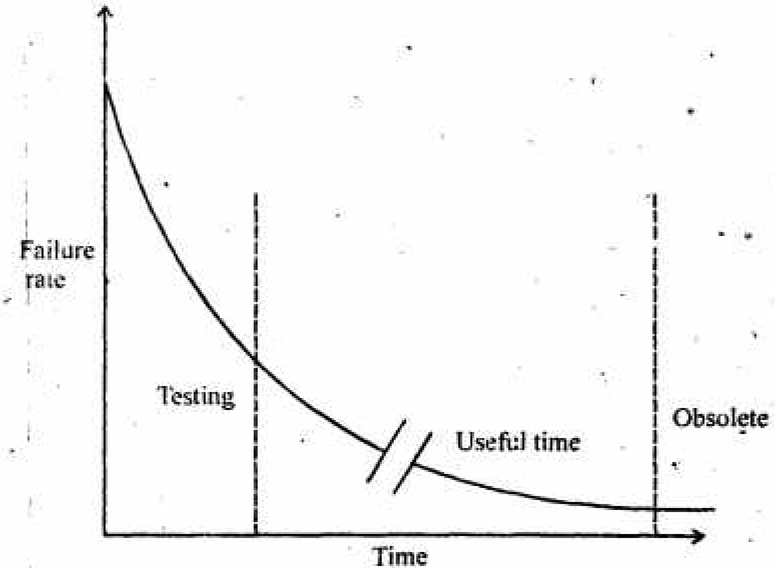
Hardware Reliability : As the time progresses, the hardware architecture gets older new hardware components arrived in the market which obsolete the oldones.

Hardware failures are inherently different from software failures. Most hardware failures are due to component wear & tear. A logic gate may be stuck at 1 or 0 or a resistor might short circuit.

The change of failure rate over the product lifetime for a typical hardware & software product are sketched in fig. For hardware products it can be observed that failure rate is high initially but decreases as the faulty components are identified and removed. The system then enters its useful life. After sometime the components wear out & the failure rate increases. This gives the hardware reliability over time its characteristics. "bath tub" shape. On the other hand, for software the failure rate is at it's highest during integration & test. As the system is tested, more & more errors are identified & removed resulting in reduced failure rate. This error removal continues at a slower pace during the useful life of the product. As the software becomes obsolete no error corrections occurs & the failure rate remains unchanged.



(a) Hardware Product



(b) Software Product

Q. 8. (b) Explain the salient features of ISO 9001 requirements.

Ans. Within the ISO-9000 series, standard ISO-9000 for quality system is the standard that is most applicable to software development.

Because of the broadness of ISO-9001. ISO has published specific guidelines to assist in applying ISO-9001 to software namely ISO-9000-3.

Salient Features are :

(i) Management Responsibility : ISO-9001 requires that the quality policy be defined, documented, understood, implemented and maintained, that responsibilities and authorities for all personnel specifying, achieving and monitoring quality be defined and that in-house verification resources be defined, trained and funded. A designed manager ensures that the quality program is implemented and maintained.

(ii) Quality System : ISO -9001 requires that a documented quality, including procedures and instructions be established. ISO-9000-3 characterizes this quality system as an integrated process throughout the entire life cycle.

(iii) Contract Review : ISO-9001 requires that constract be reviewed to determine whether the requirements are adequately defined, agreed with the bid and can be implemented.

(iv) Design Control : ISO-9001 requires that procedures to control and verify the design be established. This includes planning design activities, identifying inputs and outputs, verifying the design and controlling design changes.

(v) Document Control : ISO-9001 requires that the distribution and modification of documents be controlled.

(vi) Purchasing : ISO-9001 requires that purchased products conform to their specified requirements. This includes the assessment of potential subcontractors and verification of purchased products.

(vii) Purchaser-Supplied Product : ISO-9001 requires that any purchaser-supplied material be verified and maintained.

(viii) Product Identification and Traccability : ISO-9001 requires that the product be identified and traceable during all stages of production delivery and installation.

(ix) Processor Control : ISO-9001 requires that production processes be defined and planned. This includes carrying out production under controlled conditions according to documented instructions.

(x) Inspection and Testing : ISO-9001 requires that incoming materials be inspected before use and that in process inspection and testing be performed. Final inspection and testing are performed prior to release of finished product. Records of inspection and testing are kept.

(xi) Inspection Measuring and Test Equipment : ISO-9001 requires that equipment used to demonstrate conformance be controlled, calibrated and maintained. When test hardware or software is used it is checked before use and rechecked at prescribed intervals.

(xii) Inspection and Test Status : ISO-9001 that the requires status of inspections and tests be maintained for items as they progress through various processing steps.

(viii) Control of Non-Conforming Product : ISO-9001 requires that non-conforming product be controlled to prevent inadvertent use or installation.

(xiv) Corrective Action : ISO-9001 that the causes of non-conforming product be identified. Potential causes of non-conforming product are eliminated, procedures are changed resulting from corrective action.

(xv) Handling, Storage, Packaging and Delivery : ISO-9001 requires that procedures for handling, storage, packaging and delivery be established and maintained.

(xvi) Quality Records : ISO-9001 requires that quality records be collected, maintained are dispositioned. The practices defining the quality records to be maintained in the CMM are distributed throughout the key process areas in the various activities performed practices.

(xvii) Internal Quality Audits : ISO-9001 requires that audits be planned are performed. The results of audits are communicated to management and any deficiencies found are corrected.

(xviii) Training : ISO-9001 requires that training needs be identified and that training be provided, since selected tasks may require qualified personnel. Records of training are maintained.

(xix) Servicing : ISO-9001 requires that servicing activities be performed as specified.

(xx) Statistical Techniques : ISO-9001 status that appropriate adequate statistical techniques are identified and should be used to verify the acceptability of process capability and product characteristics.