Unit – IV

Software Reliability And Quality Management

Software Reliability And Quality Management: Software Realiability,Statistical testing,Software Quality,Software Quality Management System,ISO 9000,SEI Capability Maturity Model
Computer Adied Software Engineering: Case and its Scope,Case Environment,Case Support in Software Life Cycle,Other Characteristics of Case Tools,Towards Second Generation CASE Tool,Architecture of a case Environment

Software Reliability

Reliability is one of the important aspects of software operation . The primary concern of customers and the responsibility of development organization is to have a reliable system . Software testing methods are performed to detect and debug software defects so that a program can work without failures . Also , quality standards like ISO and the SEI _CMM provide guidelines and procedures to software development organization for systematic software development to produce quality software products . Reliability is an important factor in all kinds of software application . The criticality of reliability can be observed in the most serious areas such as life safety critical systems , where the human life depends on it . If such system fails in performing the specific tasks , then there happens a lot of losses and damages .

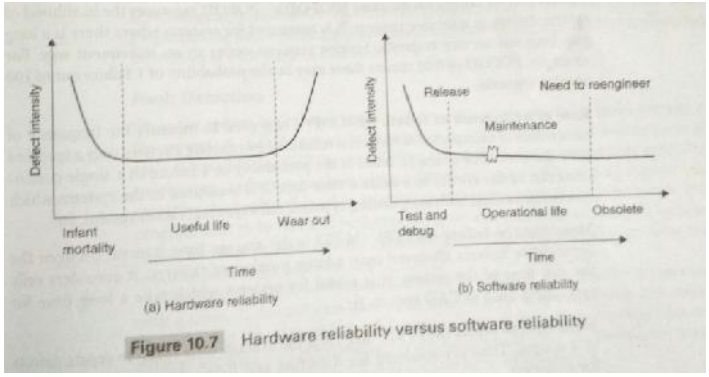

Figure 10.7 Hardware reliability versus software reliability

Reliability Metrics – An unreliable system is caused by the occurrence of failures. Some failures are more dangerous than others i.e. some failures have little consequences while other failures have very serious affects. The impact of the occurrence of failure is observed through reliability specifications. Reliability specification is in the range of data input, conditions and constraints, location in the program, and certain test criteria. Based on reliability specification, the following categories of failures are observed in the system.

Transient:- such failures occur for certain inputs.

Permanent:- Permanent failures occur for all inputs.

Recoverable:- When a recoverable failure occurs, the system can recover with or without operator intervention (i.e., without shutdown or restart, etc.).

Unrecoverable:- Unrecoverable failures need operator intervention to recover data from them.

Computing:- Such failures corrupt the system state or data.

Non-computing:- Non-computing failures occur but do not corrupt data or system state.

Reliability is used to address defect finding and fixing. The reliability of a system is unpredictable in the interval of the specified time. Due to uncertainty of defect occurrence, reliability is measured through probabilistic theory. In the following paragraphs, we will discuss various reliability metrics.

Probability Of Failure On Demand (POFOD) :- POFOD measures the likelihood of system failure in a service request. It is measured for systems where there is a long gap between service requests. Service requests occur in an infrequent way. For example, POFOD of 0.01 means three may be the probability of 1 failure out of 100 service requests.

Rate Of Occurrence Of Failure (ROCOF):- It is used to measure the frequency of occurrence of unexpected failures. It is measured by running a system over a specified time duration. For example, 0.001 is the probability of 1 failure in a single continuous run of the system in a defined time unit. It is measured in the system which runs continuously, such as operating systems, railways reservation system, etc.

Mean Time To Failure (MTTF):- MTTF is the average time interval between the consecutive failures observed over a large number of failures. It considers only the run time of the system. It is useful for systems which take a long time for processing, such as CAD system, etc.

Mean Time To Repair (MTTR):- MTTR is the average time taken to repair defects ain a system. Time is considered for detecting and fixing defects over a specified time interval.

Mean Time Between Failures (MTBF):-MTBF is measured by combining MTTR and MTTE. That is,

$$MTBF=MTTR+MTTF$$

If a failure occurs over a specified time duration, then it measures the occurrence of the next failure over the next duration. For example, if MTBF is 24 hours, it means that the next failure may occur in the next 24 hours.

System availability:- It is the likelihood that the system will available for use over a given time duration. System availability measurement excludes MTTR and MTTF. The measurement of important for systems like server machines, telecommunication system, etc.

Statistical Testing

Statistical Testing makes use of statistical methods to determine the reliability of the program. Statistical testing focuses on how faulty programs can affect its operating conditions.

How to perform ST?

- Software is tested with the test data that statistically models the working environment.
- Failures are collated and analyzed.
- From the computed data, an estimate of program's failure rate is calculated.
- A Statistical method for testing the possible paths is computed by building an algebraic function.
- Statistical testing is a bootless activity as the intent is NOT to find defects.

Software Quality

-A good quality software product satisfies the customer needs, is constructed as per standard and norms, has good internal design and developed within optimized cost and schedule.

-The quality of a software product is defined in terms of its characteristics or attributes.

- A product can be of good quality or bad quality. Weaker values of attributes define a bad quality product whereas higher values of attributes define a good quality product.

- The aim of software development organization is to produce a high quality product .The quality of a software product is defined in terms of the following points –

1. Satisfies customer requirements
2. Possesses higher values of its characteristics
3. It has sound internal and external design
4. Developed within budget, cost
5. Follows development standards.

Software Quality Factors

The quality of software is measured through well defined quality factors. There are two ways to measure software quality. One way is direct measurement, which is performed through software testing for example the number of defects detected and corrected in a program, number of faults observed in source codes and design etc. Direct measurement mainly focuses on satisfying the functional requirements. Indirect measurement is specially the quantification of non functional requirements.

McCall , Richards and Walters have proposed certain factors that affect the quality of software . These factors are the quality characteristics that measure the quality of software. These factors are classified into the following categories..

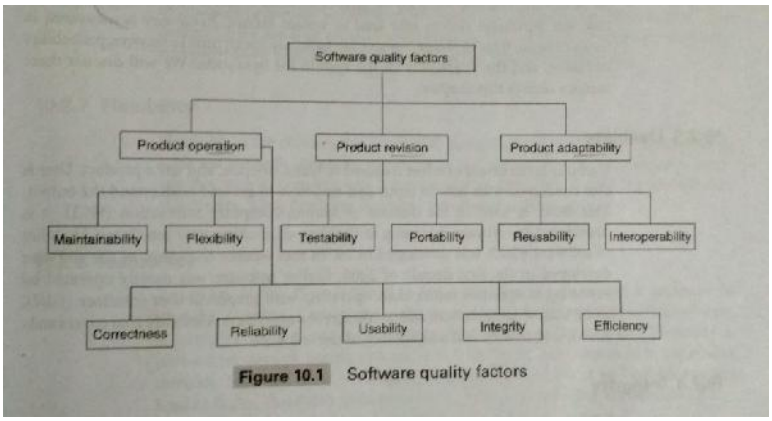

Figure 10.1 Software quality factors

Product Operational Factors – Correctness, reliability, usability, integrity, and efficiency

Product Revision Factors- Maintainability, flexibility and testability

Product Adaptability Factors – Portability, reusability and interoperability .Product operational factors focus on the use and working of a software product. Product revision factors measure the changeability characteristics of a software specification. Adaptability factors concentrate on the adaptation of a software product on different platforms and environments .The factors of these paragraphs are described in these paragraphs.

Correctness:
A program is correct if it performs according to the specifications of functions it should provide. Also the program should meet the customer objectives. Correctness is the primary quality of software without which other factors such as efficiency and usability are trivial. Achieving 100% correctness in a program is rare because it consumes a significant amount of budget and time. Correctness of software depends on the correct specification of requirements, design and implementation.

Reliability:
Reliability is the extent to which a program performs its intended functions satisfactorily with required precision without failure in a specified duration. It is measured for both hardware and software because both work together to perform the defined task. An unreliable system may lead to system failure. Reliability is measured in terms of mean time to failure, mean time to repair, mean time to recover, probability of failure and the availability of the system for operation.

Usability:
Usability is the extent of effort required to learn, operate and use a product. User is also concerned with how to input data and how to get and understand the output. This factor is used in the domain of human computer interaction (HCI). It is considered in software designing at the time of creating user interfaces. Usability of software which was developed in 1970’s was weaker than that of the software developed in the first decade of 2000. Earlier software was mostly operated by executing commands rather than operating with graphical user interface (GUI), along with sensing devices and touch screens nowadays. Usability also covers suitability, learnability and adaptability of the software.

Integrity:
Integrity is the extent of effort to control illegal access to data and program by unauthorized people .Control means to preserve the integrity of managed data to prevent and detect frauds, to process all data in a safe manner, and to detect correct and reprocess any bug in the software. Integrity is very important in the area of database and communication environment where data and programs are used from machine to machine. Therefore validation checks are used in programs to satisfy the integrity rules.

Efficiency:
Efficiency is the volume of computing resources. (Eg: processor time , memory space , bandwidth in communication devices etc.) and the code required to perform software functions . The efficiency of a software depends on various aspects such as the use of programming paradigm, OS, design and the implementation of algorithms. For eg: todays OS ‘s windows 2008 supporting multitasking multiprogramming , threading etc. , which its earlier versions like windows 3.1 did not similarly , dual core processors are more efficient than Pentium 4 processors .

Maintainability:
Maintainability is the ease to locate and correct errors. Maintainability of software is measured through mean time to change. That is when an error is found how much time it takes to analyze the change, design and modification and implement ad test the modified code. Maintainability can be increased with good design techniques, proper documentation of artifacts, good coding practices and modular design. There are four types of maintenance – corrective, adoptive, perfective and preventive.

Flexibility:
Flexibility is the cost required is modify on operational program. Flexibility can be achieved through simplicity of programs and their interfaces. Modularity characteristics such as coupling and cohesion play an important role in the modification of software. Also, flexibility is related to software maintenance.

Testability:
Testability is the effort required to test a program to ensure that it performs its intended function. Complexity of the source code affects testability of the software. A complex code needs more testing efforts than a simple code. Testability is performed through testing process, which facilitates the creation of better quality software .Software must be designed keeping a view that it will not tolerate any kind of faults. Testability aims to detect defects as soon as they are introduced.

Portability:
Portability is the effort required for transferring software products to various hardware and software environments. Objects-oriented programs support better portability than machine and assembly language programs. Portability has several dimensions, such as program portability, data portability, documentation portability, and people portability .Program portability allows running a program on different platforms. Data portability means that the same data structure is available on all data files of different platforms. People portability

includes the user in developer that can work on a several platforms .Documentation portability is the help provided on different platforms.

Reusability:

Reusability is the use of existing software or its parts. Reusability is the extent to which software or its parts can be reused in the development of some other software. Reusability increases productivity, reliability and predictability of the software and reduces product cost and development time. There are various components based technologies that support reuse such as CORBA, COM/DCOM, EJB and Active X. The most important aspect of reuse is writing of generic components so that these can be reused either without modification or with little modification.

Interoperability:

Interoperability is the effort required to couple one system to another. Strong coupling and loose coupling are the approaches used in operability. Strong coupling cause components to be integrated together in a single executable file and run as a single process. Loose coupling allows different components to execute as separate processes. Most of the component based technologies allow designing components that are interoperable.

Verification & Validation

Verification is the process of evaluating work products in the software development phases to assess whether the work product meet the specifications as intended for the purpose. The objective of verification is to ensure that the software is being built according to the requirements specifications and design specifications. Also, verification is performed to ensure that the output produced in a phase confirms to its input specifications. Verification is the process internal to the development or maintenance process. As per IEEE, “verification is the act of reviewing, inspecting, testing, checking, auditing or otherwise establishing and documenting whether product, process, service or documents confirm to the specified requirements.

Validation is the process of evaluating software during or at the end of the development process to determine whether it satisfies the specified stated requirements. It is performed to ensure that the product is actually meets the users needs and that the specifications are correct in the first place. Validation is done to assure the acceptance and the suitability of the product for the customer. It is concerned with assessing the quality of the software in its actual operating environment. As per IEEE, “validation is the process of evaluating software at the end of the software development process to determine compliance with the requirement”. Validation is the end to end verification. There are many more definitions of verification and validation in the software engineering literature. BERRY BOEHM defines and differentiates verification and validation is follows:

Verification:- Are we building the right product?

Validation:-Are we building the product right?

Software Quality Management System

Producing high quality product is the primary goal of an organization .Organization applies high quality policies to produce good products. Each organization has its own quality management system that focuses on ensuring optimum quality.

QMS is a set of procedures/processes which are carried out to ensure that the product delivered by the organization has the desired quality .Sometimes quality system is also used in place of QMS.

The success of QMS depends on employees' skills and proper support from the top management of the organization.

The quality system includes various things such as quality policy, quality manual, departmental responsibilities, roles, procedures, standards and guidelines, reports, forms, templates and checklists.

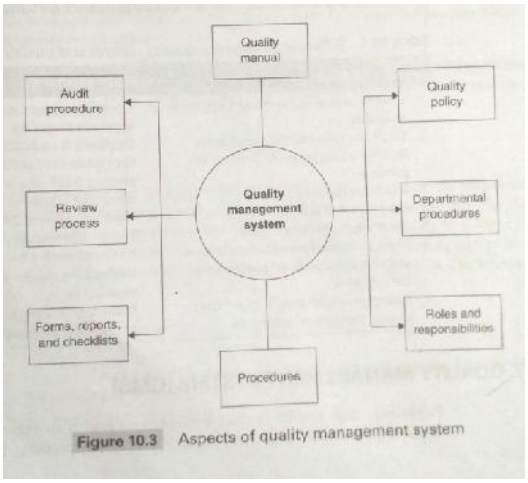

Figure 10.3 Aspects of quality management system

Auduting Procedures And Review Of The Quality System –

The quality manual states the vision and mission of the organization committing itself to quality.
Quality Policy document states the intentions and directions of work flow towards quality.
Quality Group has several sub teams for performing specific tasks of their departments.
In QMS each individual has specific role and responsibilities .Individuals are well trained in performing their roles .There are certain procedures followed by organizations like CMM,ISO .Inspection and testing methods are applied at each phase of software development .

ISO 9000

It is a set of international standards for quality management and quality assurance .Standards were developed to help companies effectively document the elements they need to maintain an efficient quality system.
Initial ISO 9000 standards were published in 1987 as ISO 9000:1987
Revised in 1994 as ISO 9000:1994

It had three standards.
ISO 9001:1994
ISO 9002:1994
ISO 9003:1994
Later ISO 9002:1994 &ISO 9003:1994 was merged into ISO 9001:1994 .Some clauses were eliminated from 9002,9003.
In the year 2000, ISO: 1994 was revised and rewritten as ISO 9000:2000.
There were several minor revisions made in the ISO 9000 series. ISO 9001:2008 is the latest ISO standard which is most widely used by the software companies. Revised 9001:2008 series of standards forms a coherent set of quality management system. Standards facilitating mutual understanding in national and international trades .It is based on 8 quality management principles:
1.Customer Focus –As organizations depend on their customers, they should understand current and future customers .
2.Leadership- It establishes unity of purpose and direction of the organization .Organizations should create and maintain an internal environment in which people can become fully involved in achieving the organizations objectives .
3.Involvement of People – People at all levels are the assets of an organization and their full involvement enables their abilities to be used for organization ‘s objectives .
4.Process Approach –The desired result is achieved more efficiently when activities and related resources are managed as a process .
5. System Approach To Management –Identifying, understanding and managing interrelated processes as a system contributes to the organization‘s effectiveness and efficiency in achieving its objectives.
6. Continual Improvement –It is the overall performance. It should be a permanent objective of the organization.
7.Factual Approach to Decision Making -Effective decisions are based on the analysis of data and information .
8.Mutually Beneficial Supplier Relationships-Organizations and suppliers are independent and mutually beneficial relationship enhances the ability of both to create value .

ISO 9000 CERTIFICATION PROCESS

Initially an organization has to decide about getting the ISO certification because it will involve in the formal process of development .Organizations willing to achieve ISO certification go through the following ISO 9000 certification steps and apply to the ISO 9000 registrar .
Proposal Stage- The organization prepares a proposal in the format and according to the guidelines provided by the ISO to apply to the registrar for ISO certification.
Pre assessment Stage – At this stage registrar and the people responsible make a rough assessment of the organization.
Document Review and Adequacy Audit –The registrar reviews the proposal and its related documents submitted by the organization and suggest improving the documents if there is any flow in the proposal. An audit confirms that the organization has made the planned

arrangement for the ISO certification and it will make follow the process based quality management system approach.

Compliance Audit – This is the verification process to check whether the organization has realized the suggestions provided by the registrar during the document review stage.

Registration –The organization receives the ISO certification from the ISO registrar after completing all the previous stages.

Continued Surveillance : The ISO registrar and his team regularly monitor the organization to ensure that the organization is following the ISO standards for software development .

ADVANTAGES OF ISO 9000:

1. Increased Marketability
2. Reduced operational expenditures
3. Improved internal communication
4. Improvd customer service
5. Reduction of product liability risks
6. Attractiveness to investors

DISADVANTAGES OF ISO 9000:

1. Owners and managers do not have an adequate understanding of ISO 9000.
2. Most of companies have less funding available, therefore companies are finding difficulties to adopt ISO system.
3. ISO 9000 registration need heavy document workload.
4. ISO 9000 registration process require long time.

SEI Capability Maturity Model :

-It is an industry standard model for defining and measuring the maturity of the development process and for providing strategy for improving the software process towards achieving high quality products.

-It was established by Software Engineering Institute (SEI) in 1986 at Carnegie Mellon University (CMU) at California, USA under the direction of US Department of Defence.

-The purpose of the SEI is to help organizations improve their software engineering capabilities and to develop or acquire the right software, defect free, within budget and on-time, every time.

-CMM model is different from software development life cycle models .The CMM is involved in the process management process to improve the software process whereas life cycle models are used for the software development process.

Initial Level-At initial level software process is characterized as adhoc, inconsistent, and occasionally even chaotic. There are no defined processes and standard practices. Processes are unpredictable.

Repeatable Level-In this software development successes are repeatable i.e. process discipline is in place to repeat earlier success on projects with similar applications. Here basic and consistent project management processes are established to track cost, schedule and

development. However the discipline and process while established vary from project to project.

Defined Level-Now the organization has standardized process for both management and developed activities. These processes are well documented, standardized and integrated into a standard software process for entire organization.

Managed Level-Here precise measurements are used to effectively control the software development effort. Both the software process and products are quantitatively understood and controlled. Here processes are predictable to determine the performance.

Optimized Level-The key characteristic of optimized level is continuous and proactive process improvement. That is continuous improvement is institutionalized in the development process by enabling quantitative analysis of the process and by piloting innovative ideas and technologies. Here processes are concerned with addressing the common causes of process variation and changing the process to improve the process performance to achieve the established quantitative process improvement objectives.

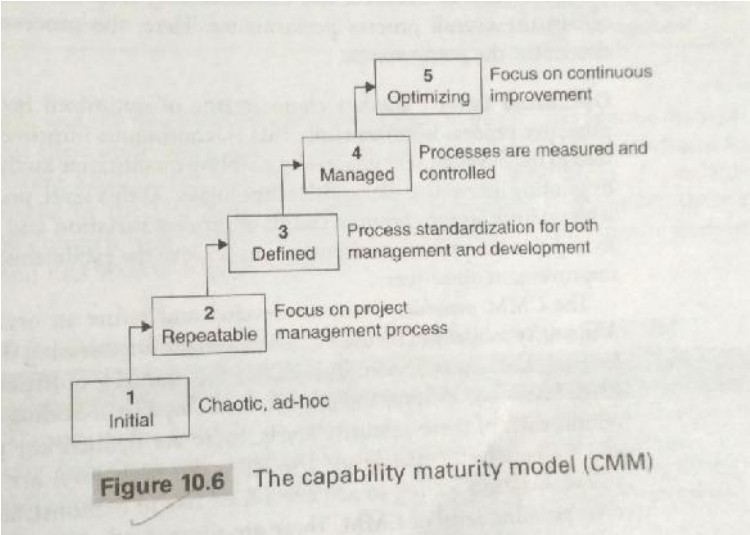

Figure 10.6 The capability maturity model (CMM)

The CMM provides a way to develop and refine an organization's process. A maturity model can be used a benchmark for assessing different organizations for equivalent comparison. It describes the maturity of the company in the above stated levels based upon the project the company is dealing with and the clients. Within each of these maturity levels, there are different key process areas(KPAs) which characterize that level. The KPAs for each level are listed in table 10.2.

An organization willing to achieve a level has to demonstrate all the KPAs in the corresponding level of CMM. There are some overlapping KPAs, such as software product engineering is addressed at defined as well as managed level

Table 10.2 Focus and KPAs for each CMM level

CMM level	Focus	KPAs
1. Initial	Competent people and heroics	Not applicable
2. Repeatable	Disciplined process	Requirement management Project planning and tracking Subcontractor management Software quality assurance Configuration management
3. Defined	Process standardization	Organization process focus Organization process definition Training program Software product engineering Integrated software development Inter-group coordination Peer reviews
4. Managed	Measurable and controlled processes for quality	Quantitative process management Quality management
5. Optimized	Continuous process improvement	Defect prevention Technology change management Process change management

COMPUTER-AIDED SOFTWARE ENGINEERING (CASE)

Computer aided software engineering (CASE) is a took support for software development, management and maintenance activities. The use of CASE tools cases and reduces the development, management, and maintenance effort. Automated tools accelerate the development activities. The CASE tools help the developers, managers and other key personnel their productivity in the development team.

Here, we will discuss various CASE tools and their benefits in software development and other activities.

CASE Environment

Software development life cycle activities can be assisted by the use of software tools. Programmers can use automated tools for development, maintenance, management, and planning purposes. The CASE environment of various automated tools is shown in figure. Some of the widely used tools, documentation tools, system software tools, quality assurance tools, database tools, software configuration management tools, prototyping tools, coding tools, testing tools, reengineering tools etc.

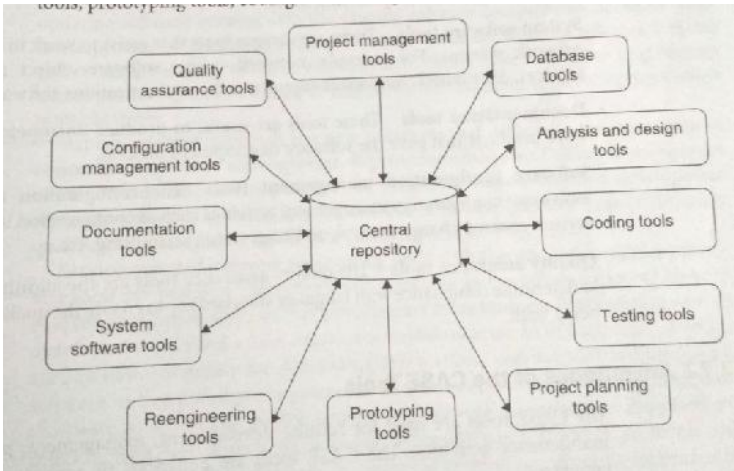

Figure 12.9 CASE tools

Project management tools-Management tools project management tools can be used to design project schedule, activity plan, etc.

Database management tools- The database for relational and object-oriented database management system is designed to provide support for the CASE repository.

Analysis and design the tools –These tools help to produce analysis and design models of the system to be built. These models describe the problem and the solutions of the system in the context.

Programming tools- Compilers, editors debuggers, programming environments, fourth generation languages, graphical programming environments, applications generators, and database query generators are the programming tools used for programming purposes.

Integration and testing tools- These tools are basically used for data preparation, static and dynamic measurements, simulation, and test planning.

Project planning tools- These tools are useful for cost and effort estimation and project scheduling activities.

Prototyping tools- These tools help to design screen layouts, data design, and report generation.

Reengineering tools- Reengineering tools are used during system migration from an old platform to a new platform. There are various reengineering tools such as reverse engineering to specification tools, code restricting and analysis tools, and online system reengineering tools.

System software tools- There are various tools that assist to work in desktop and network systems. For example, network system software, object management services, distributed component support, and communications software.

Documentation tools- These tools are useful to produce documentation of the work products that pave the software development process.

Software configuration management tools- The configuration management tools assist configuration management activities such as configuration identification, version control, change control, auditing, status accounting, etc.

Quality assurance tools- The quality assurance tools are the auditing tools used to determine compliance with language standards or to ensure the quality of software being built.

Advantages of the CASE Tools

The CASE tools are used for routine development, management, planning, and maintenance activities. The CASE tools are beneficial to achieve the following benefits;

- These are helpful in automating manual activities. This reduces the effort of development and human mistakes.
- The use of CASE tools saves development cost of all development phases.
- The CASE tools improve the overall quality of the product due to the fact that tasks are carried out in an automated manner instead of by human efforts.
- They are highly useful in software maintenance tasks.
- Requirement and configuration management with the CASE tools is more effective. Systematic management of requirements and change requests improves communication among team members and speeds up the development process.
- Also, they increase the productivity of the software development process.
- The CASE tools save manual documentation of huge documents Automated tools help to generate documents of software products.
- Modeling tools are helpful in designing the prototypes of software.