UNIT 3 CASE TOOLS

Structure **Page Nos.**

3.0 INTRODUCTION

Software Engineering is broadly associated with the development of quality software with increasing use of software preparation standards and guidelines. Software is the single most expensive item in a computer system as the cost of software during the life time of a machine is equivalent to more than 95% of the total cost (including hardware). Software Engineering requires a lot of data collection and information generation. Since the computer itself is a very useful device as the information processor, it may be a good idea to automate software engineering tasks. Computer Aided Software Engineering (CASE) tools instill many software engineering tasks with the help of information created using computer. CASE tools support software engineering tasks and are available for different tasks of the Software Development Life Cycle (SDLC). You have been introduced to CASE tools in Unit 10 of MCS-014. This unit covers various aspects of these CASE tools and their functionality for various phases of software development.

3.1 OBJECTIVES

After going through this unit, you should be able to:

- define different kinds of CASE tools and the needs of the CASE tools;
- describe the features of analysis, design, tools use in Software Engineering; and
- identify software configuration management, and project management tool.

3.2 WHAT ARE CASE TOOLS?

Computer Aided Software Engineering (CASE) tools are gradually becoming popular for the development of software as they are improving in the capabilities and functionalities and are proving to be beneficial for the development of quality software. But, what are the CASE tools? And how do they support the process of development of software?

CASE tools are the software engineering tools that permit collaborative software development and maintenance. CASE tools support almost all the phases of the

software development life cycle such as analysis, design, etc., including umbrella activities such as project management, configuration management etc. The CASE tools in general, support standard software development methods such as Jackson Structure programming or structured system analysis and design method. The CASE tools follow a typical process for the development of the system, for example, for developing data base application, CASE tools may support the following development steps:

- Creation of data flow and entity models
- Establishing a relationship between requirements and models
- Development of top-level design
- Development of functional and process description
- Development of test cases.

The CASE tools on the basis of the above specifications can help in automatically generating data base tables, forms and reports, and user documentation.

Thus, the CASE tools –

- support contemporary development of software systems, they may improve the quality of the software
- help in automating the software development life cycles by use of certain standard methods
- create an organsiation wide environment that minimizes repetitive work
- help developers to concentrate more on top level and more creative problem-solving tasks
- support and improve the quality of documentation (Completeness and non-ambiguity), testing process (provides automated checking), project management and software maintenance.

Most of the CASE tools include one or more of the following types of tools:

- Analysis tools
- Repository to store all diagrams, forms, models and report definitions etc.
- Diagramming tools
- Screen and report generators
- Code generators
- Documentation generators
- Reverse Engineering tools (that take source code as input and produce graphical and textual representations of program design-level information)
- Re-engineering tools (that take source code as the input analyse it and interactively alters an existing system to improve quality and/or performance).

Some necessary features that must be supported by CASE tools in addition to the above are:

- It should have Security of information. The information may be visible/ changeable by authorised users only.
- Version Control for various products
- A utility to Import/Export information from various external resources in a compatible fashion
- The process of Backup and Recovery as it contains very precious data.

Figure 3.1 shows an environment having CASE.

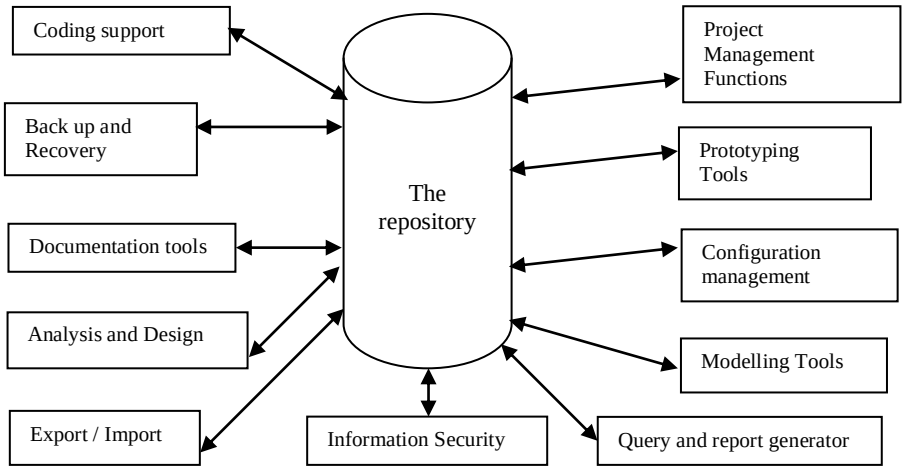

Figure 3.1: CASE Tools

3.2.1 Categories of CASE Tools

On the basis of their activities, sometimes CASE tools are classified into the following categories:

1. Upper CASE tools
2. Lower CASE tools
3. Integrated CASE tools.

Upper CASE: Upper CASE tools mainly focus on the analysis and design phases of software development. They include tools for analysis modeling, reports and forms generation.

Lower CASE: Lower CASE tools support implementation of system development. They include tools for coding, configuration management, etc.

Integrated CASE Tools: Integrated CASE tools help in providing linkages between the lower and upper CASE tools. Thus creating a cohesive environment for software development where programming by lower CASE tools may automatically be generated for the design that has been developed in an upper CASE tool.

Figure 3.2 shows the positioning of CASE tools in a Software Application development.

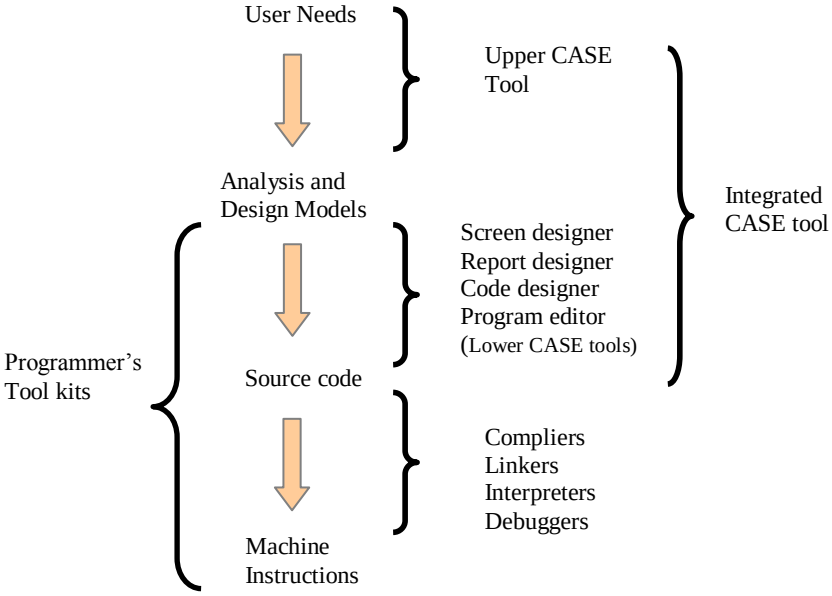

Figure 3.2: The CASE tools and Application Development

3.2.2 Need of CASE Tools

The software development process is expensive and as the projects become more complex in nature, the project implementations become more demanding and expensive. The CASE tools provide the integrated homogenous environment for the development of complex projects. They allow creating a shared repository of information that can be utilised to minimise the software development time. The CASE tools also provide the environment for monitoring and controlling projects such that team leaders are able to manage the complex projects. Specifically, the CASE tools are normally deployed to –

- Reduce the cost as they automate many repetitive manual tasks.
- Reduce development time of the project as they support standardisation and avoid repetition and reuse.
- Develop better quality complex projects as they provide greater consistency and coordination.
- Create good quality documentation
- Create systems that are maintainable because of proper control of configuration item that support traceability requirements.

But please note that CASE tools cannot do the following:

- Automatic development of functionally relevant system
- Force system analysts to follow a prescribed methodology
- Change the system analysis and design process.

There are certain disadvantages of CASE tools. These are:

- Complex functionality
- Many project management problems are not amenable to automation. Hence, CASE tools cannot be used in such cases.

3.2.3 Factors that affect deployment of CASE Tools in an organisation

A successful CASE implementation requires the following considerations in an organisation:

1. Training all the users in typical CASE environment that is being deployed, also giving benefits of CASE tools.
2. Compulsory use of CASE initially by the developers.
3. Closeness of CASE tools methodology to the Software Development Life Cycle
4. Compatibility of CASE tools with other development platforms that are being used in an organisation.
5. Timely support of vendor with respect to various issues relating to CASE tools:
    - Low cost support.
    - Easy to use and learn CASE tools having low complexity and online help.
    - Good graphic support and multiple users support.
6. Reverse Engineering support by the CASE tools: It is important that a CASE tool supports complicated nature of reverse engineering.

3.2.4 Characteristics of a successful CASE Tools

A CASE tool must have the following characteristics in order to be used efficiently:

- **A standard methodology:** A CASE tool must support a standard software development methodology and standard modeling techniques. In the present scenario most of the CASE tools are moving towards UML.

- **Flexibility**: Flexibility in use of editors and other tools. The CASE tool must offer flexibility and the choice for the user of editors' development environments.
- **Strong Integration**: The CASE tools should be integrated to support all the stages. This implies that if a change is made at any stage, for example, in the model, it should get reflected in the code documentation and all related design and other documents, thus providing a cohesive environment for software development.
- **Integration with testing software**: The CASE tools must provide interfaces for automatic testing tools that take care of regression and other kinds of testing software under the changing requirements.
- **Support for reverse engineering**: A CASE tools must be able to generate complex models from already generated code.
- **On-line help**: The CASE tools provide an online tutorial.

Now let us discuss some of the important characteristics for various types of CASE tools in the subsequent sections.

☞ Check Your Progress 1

1) What is the need of CASE tools?

……..

……..

……

2) What are the important characteristics of CASE tools?

……..

……..

……

3.3 CASE SOFTWARE DEVELOPMENT ENVIRONMENT

CASE tools support extensive activities during the software development process. Some of the functional features that are provided by CASE tools for the software development process are:

1. Creating software requirements specifications
2. Creation of design specifications
3. Creation of cross references.
4. Verifying/Analysing the relationship between requirement and design
5. Performing project and configuration management
6. Building system prototypes
7. Containing code and accompanying documents.
8. Validation and verification, interfacing with external environment.

Some of the major features that should be supported by CASE development environment are:

- a strong visual support
- prediction and reporting of errors
- generation of content repository
- support for structured methodology
- integration of various life cycle stages
- consistent information transfer across SDLC stages
- automating coding/prototype generation.

Present CASE tools support Unified Model Language (UML).

We will elaborate on the features of CASE tools for various stages of software development process in coming sub-sections.

CASE and Web Engineering

CASE Tools are also very useful in the design, development and implementation of web site development.

Web Engineering requires tools in many categories. They are:

- Site content management tools
- Site version control tools
- Server management tool
- Site optimisation tools
- Web authoring and deployment tools
- Site testing tools that include load and performance testing
- Link checkers
- Program checkers
- Web security test tools.

A detailed discussion on these tools is beyond the scope of this unit. However, various stages of development of a web project also follows the normal SDLC. These are discussed in the subsequent sections.

3.4 CASE TOOLS AND REQUIREMENT ENGINEERING

Let us first answer the question:

Which is the most Common Source of Risk during the process of software development?

One of the major risk factors that affect project schedule, budget and quality can be defined as the ability to successfully elicit requirements to get a solution.

Statistically it has been seen that about 80% of rework in projects is due to requirement defects.

How can a CASE tools help in effective Requirements Engineering (RE)

A good and effective requirements engineering tool needs to incorporate the best practices of requirements definition and management.

The requirements Engineering approach should be highly iterative with the goal of establishing managed and effective communication and collaboration.

Thus, a CASE tool must have the following features from the requirements engineering viewpoint:

- a dynamic, rich editing environment for team members to capture and manage requirements
- to create a centralised repository
- to create task-driven workflow to do change management, and defect tracking.

But, what is a good process of Requirements Engineering for the CASE?

A simple four-step process for requirements engineering is given in *Figure 3.3*.

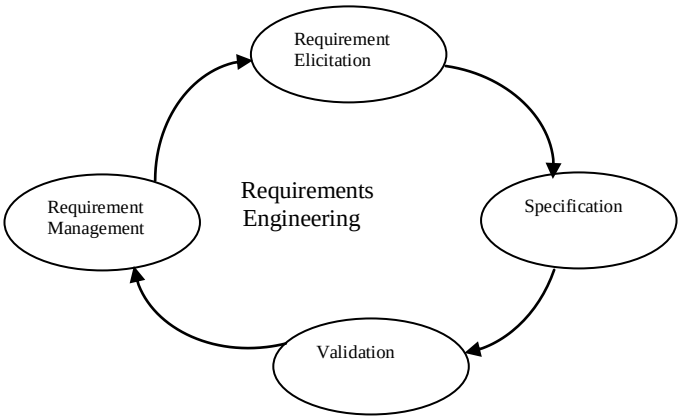

Figure 3.3: The four-step requirement engineering process

Requirement Elicitation

A simple technique for requirements elicitation is to ask "WHY".

CASE tools support a dynamic, yet intuitive, requirements capture and management environment that supports content and its editing. Some of the features available for requirement elicitation are:

- Reusable requirements and design templates for various types of system
- Keeping track of important system attributes like performance and security
- It may also support a common vocabulary of user-defined terms that can be automatically highlighted to be part of glossary.
- Provide feature for the assessment of the quality of requirements
- Separate glossary for ambiguous terms that can be flagged for additional clarification.

What do we expect from the tool?

- It should have rich support for documentation that is easily understandable to stakeholders and development teams.
- It should have the capability of tracking of requirements during various SDLC systems.
- It should help in the development of standard technical and business terminology for end clients.

Software Analysis and Specification

One of the major reasons of documenting requirements is to remove the ambiguity of information. A good requirement specification is testable for each requirement.

One of the major features supported by CASE tools for specification is that the design and implementation should be traceable to requirements. A good way to do so is to support a label or a tag to the requirements. In addition it should have the following features:

- Must have features for storing and documenting of requirements.
- Enable creation of models that are critical to the development of functional requirements.
- Allow development of test cases that enable the verification of requirements and their associated dependencies.
- Test cases help in troubleshooting the correlation between business requirements and existing system constraints.

What do we expect from the tool?

- It should allow development of a labeled requirements document that helps in traceability of requirements.
- It should allow both functional and non-functional requirements with related quality attributes to be made.
- We should be able to develop the associated models.

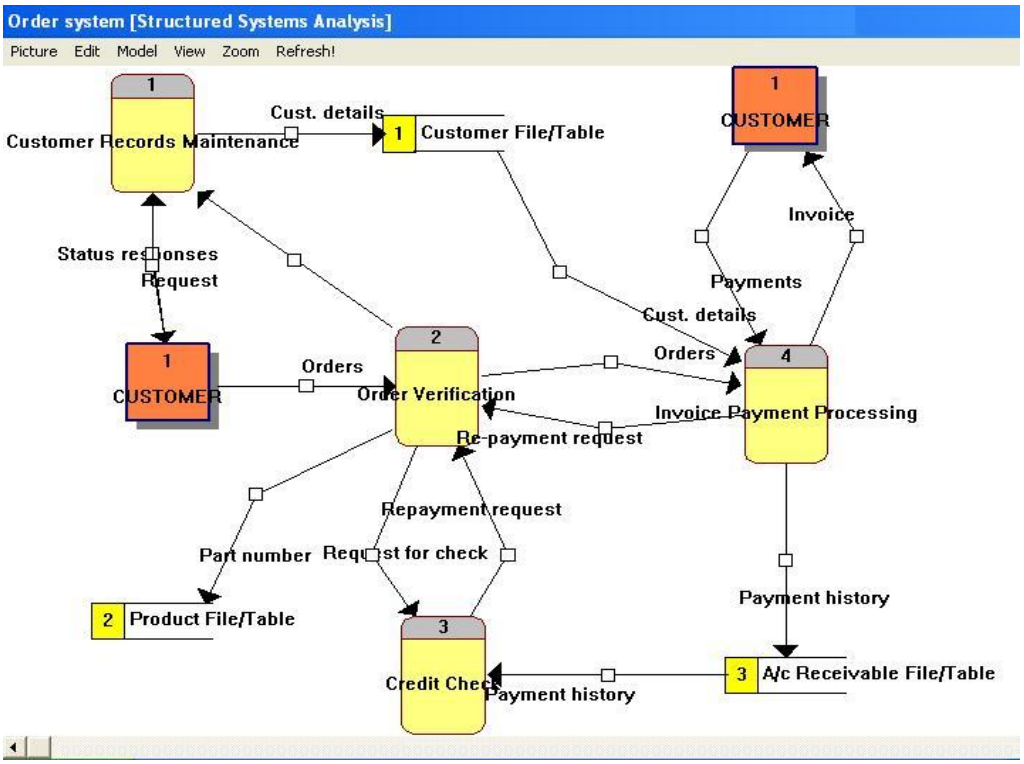

Figure 3.4: A sample DFD using CASE Tools

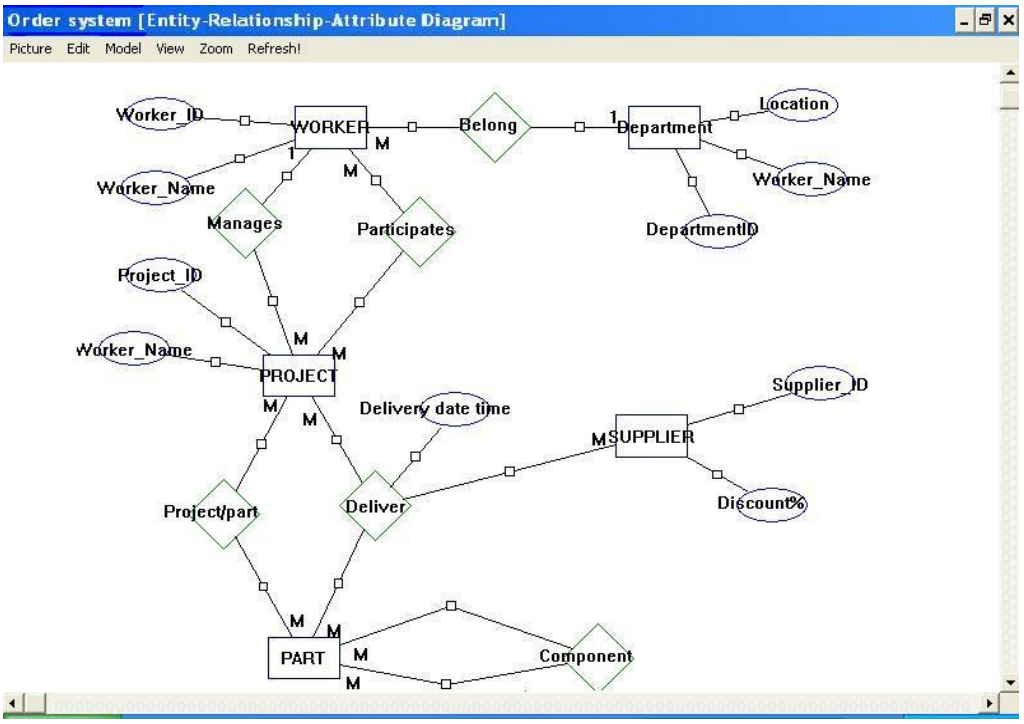

Figure 3.5: A Sample ER Diagram using CASE Tools

Figures 3.4 and 3.5 show some of the models developed with the help of a sample CASE Tool.

"The automated approach of software development needs to have a spirit of collaboration, bringing together world-renowned principal investigators, business analysts, and software development teams."

Validation of Requirements

A very important feature in this regard is to allow collaboration yet customizable workflows for the software development team members. Also facilitating approvals and electronic signatures to facilitate audit trails. Assigning owner of requirement may be helpful if any quality attributes may need changes. Thus, a prioritised validated documented and approved requirements can be obtained.

Managing Requirements

The requirements document should have visibility and help in controlling the software delivery process. Some such features available in CASE tools are:

- estimation of efforts and cost
- specification of project schedule such as deadline, staff requirements and other constraints
- specification of quality parameters.

Software Change Management

An incomplete or ambiguous requirement if detected early in the analysis phase can be changed easily with minimum cost. However, once they are converted to baselines after requirements validations the change should be controlled.

One of the major requirements of change management is:

Any change to these requirements should follow a process which is defined as the ability to track software requirements and their associated models/documents, etc. (e.g., designs, DFDs and ERDs, etc.). This helps in determining the components that will be impacted due to change. This also helps tracking and managing change. The whole process starts with labeling the requirements properly.

Most CASE Tools store requirement baselines, including type, status, priority and change history of a software item. Such traceability may be bi-directional in nature.

Static word processing documents or spreadsheet recedes communication issues because those tools do not allow sharing and collaboration on up-to-date requirements. To overcome this a CASE enabled requirements management infrastructure offers the familiarity of environments such as Microsoft Word, with added communication methods, such as email. Thus, CASE is a very useful tool for requirement engineering.

What are the features of high quality requirements?

- Correctness
- Unambiguous
- Must
- Consistency
- Known constraints
- Modifiable
- Priority Assigned
- Traceable
- Verifiable

3.5 CASE TOOLS AND DESIGN AND IMPLEMENTATION

In general, some CASE tools support the analysis and design phases of software development. Some of the tools supported by the design tools are:

- Structured Chart.
- Pogram Document Language (PDL).
- Optimisation of ER and other models.
- Flow charts.
- Database design tools.
- File design tools.

Some of functions that these diagrams tool support are simple but are very communicative as far as representations of the information of the analysis and design phases are concerned. CASE Tools also support standard representation of program architecture; they also contain testing items related to the design and debugging. Automatic support for maintenance is provided in case any of the requirements or design items is modified using these diagrams. These CASE tools also support

error-checking stages. They allow checks for completeness and consistency of the required functional design types and consistency at various levels of cross referencing among the documents consistency checking for these documents during the requirements analysis and design phases.

Proper modeling helps in proper design architecture. All the CASE tools have strong support for models. They also help in resolving conflicts and ambiguity among the models and help in optimising them to create excellent design architecture and process implementation.

But why do we need to model?

Can you understand code? If you understand it, then why would you call it code?

The major advantages for modeling are:
A model enhances communication, as it is more pictorial and less code.

Even early humans have used pictures to express ideas in a better way.

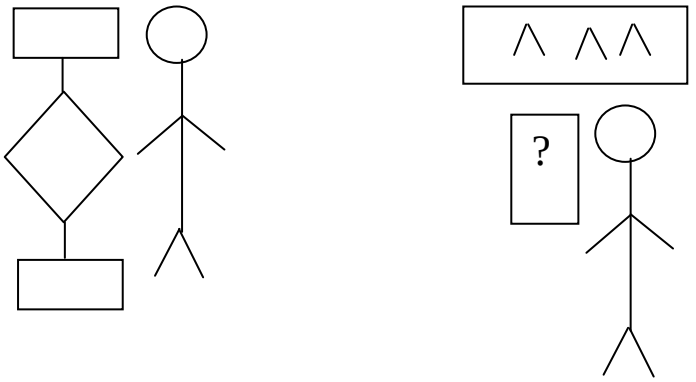

Figure 3.6: Model versus code

A model

- helps the users
- may reduce the cost of project
- can convey a broad spectrum of information.

Some standard models may need to represent:

- Overall system architecture of the software
- Software dependencies
- Flow of information through a software system
- Database organisation and structure.
- Deployment configuration, etc.

Models also help in better planning and reduction of risk as one can make top down models, thus controlling complexity.

So what is a good modeling tool?

The following are some of the characteristics of a good modeling tool:

CASE tools provide continuously synchronized models and code, thus also help in consistent understanding of information and documentation. It also helps other software developers to understand the portions and relationships to portions other team members are working on.

Help in management of source code through a more visual model.

Refactoring of the code allows a re-look and improvement of code, as modeling tools contains, thus are most continuously sychronised code and models suitable for refactoring.

Modeling can help in creating good patterns including modeling patterns, language specific patterns and basic patterns for common operations such as implementing interfaces, etc.

Models also facilitate reverse engineering.

Given the task of maintaining software that was built by a team that has moved on to a new project, there may be very little documentation available for the project.

The value of a modeling tool can be tremendous in such cases. The developer can take the existing code, create models for it to understand the software. This can save a lot of time for the developer.

A modeling tool should have been integrated with requirements tools, so that architects see consistently unambiguous information.

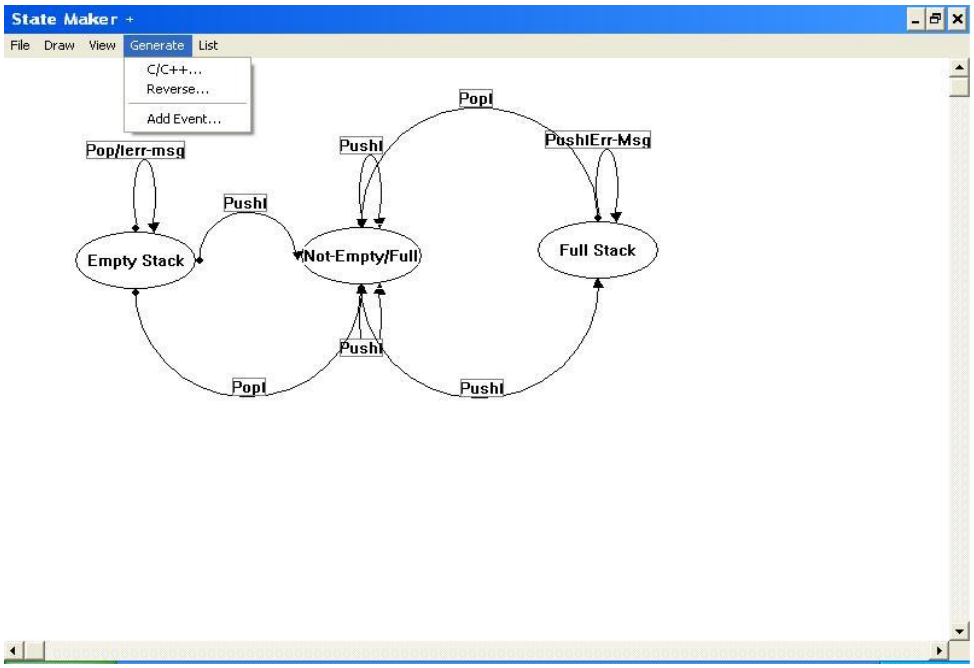

Figure 3.7: A State Model

CASE Repository

CASE Repository stores software system information. It includes analysis and design specifications and helps in analysing these requirements and converting them into program code and documentation. The following is the content of CASE repository:

- Data
- Process
- Models
- Rules/ Constraints.

Data: Information and entities/object relationship attributes, etc.

Process: Support structured methodology, link to external entities, document structured software development process standards.

Models: Flow models, state models, data models, UML document etc.

Rules/Constraints: Business rules, consistency constraints, legal constraints.

The CASE repository has two primary segments.

1. Information repository
2. Data dictionary.

Information Repository includes information about an organisation's business information and its applications.

The CASE tools manage and control access to repository. Such information data can also be stored in corporate database.

Data dictionary contains all the data definitions for all organisational applications, along with cross-referencing if any.

Its entries have a standard definition, viz., element name and alias; textual description of the element; related elements; element type and format; range of acceptable values and other information unique to the proper processing of the element.

CASE Repository has additional advantages such that it assists the project management tasks; aids in software reusability by enabling software modules in a manner so that they can be used again.

Implementation tools and CASE

CASE tools provide the following features for implementation:

- Diagramming tools enable visual representation of a system and its components
- Allow representing process flows.
- Help in implementing the data structures and program structures
- Support automatic creation of system forms and reports.
- Ready prototype generation.
- Create of both technical and user documentation.
- Create master templates used to verify documentation and its conformance to all stages of the Software Development Life Cycle (SDLC).
- Enable the automatic generation of program and database from the design documents, diagrams, forms and reports stored in the repository.

3.6 SOFTWARE TESTING

CASE tools may also support software testing. One of the basic issues of testing is management, execution and reporting. Testing tools can help in automated unit testing, functional regression testing, and performance testing. A testing tool ensures the high visibility of test information, types of common defects, and application readiness.

Features Needed for Testing

The tool must support all testing phases, viz., plan, manage and execute all types of tests viz., functional, performance, integration, regression, testing from the same interface.

It should integrate with other third party testing tools.

It should support local and remote test execution.

It should help and establish and manage traceability by linking requirements to test cases. This also helps when requirements change; in such as case the test cases traced to the requirement are automatically flagged as possible candidates for change.

It should create a log of test run.

It should output meaningful reports on test completeness, test analysis.

It may provide automated testing.

☞ Check Your Progress 2

1) List any four tools that are used in web software engineering but are not used in general software projects.

……

……

2) Which of the requirements engineering processes is not supported by CASE tools?

……

……

3) List any four major requirements of a design tool.

……

……

4) List four important features of a testing tool.

……

……..

3.7 SOFTWARE QUALITY AND CASE TOOLS

Software quality is sacrificed by many developers for more functionality, faster development and lower cost. However, one must realise that a good quality product actually enhances the speed of software development. It reduces the cost and allows enhancement and functionality with ease as it is a better structured product. You need to pay for a poor quality in terms of more maintenance time and cost. Can the good quality be attributed to a software by enhancing the testing? The answer is NO. The high quality software development process is most important for the development of quality software product. Software quality involves functionality for software usability, reliability, performance, scalability, support and security.

Integrated CASE tools:

- help the development of quality product as they support standard methodology and process of software development
- supports an exhaustive change management process
- contains easy to use visual modeling tools incorporating continuous quality assurance.

Quality in such tools is represented in all life cycle phases viz., Analysis/ Design development, test and deployment. Quality is essential in all the life cycle phases.

Analysis: A poor understanding of analysis requirements may lead to a poor product. CASE tools help in reflecting the system requirements clearly, accurately and in a simple way. CASE tools support the requirements analysis and coverage also as we have modeling. CASE also helps in ambiguity resolution of the requirements, thus making high quality requirements.

Design: In design the prime focus of the quality starts with the testing of the architecture of the software. CASE tools help in detecting, isolating and resolving structure deficiency during the design process. On an average, a developer makes 100 to 150 errors for every thousand lines of code. Assuming only 5% of these errors are

serious, if software has ten thousand lines of code you may still have around 50 serious coding errors in your system. One of the newer software development processes called the Agile process helps in reducing such problems by asking the developer to design their test items first before the coding.

A very good approach that is supported by CASE tools specially running time development of C, C++, JAVA or .NET code is to provide a set of automatic run time Language tools for development of reliable and high performance applications.

Testing: Functionality and performance testing is an integrated part of ensuring high quality product. CASE support automated testing tools that help in testing the software, thus, helping in improving the quality of testing. CASE tools enhance the speed breadth and reliability of these design procedures. The design tools are very important specifically in case of a web based system where scalability and reliability are two major issues of design.

Deployment: After proper testing a software goes through the phase of deployment where a system is made operational. A system failure should not result in a complete failure of the software on restart. CASE tools also help in this particular place. In addition, they support configuration management to help any kind of change thus to be made in the software.

Quality is teamwork: It involves integration of workflow of various individuals. It establishes a traceability and communication of information, all that can be achieved by sharing workload documents keeping their configuration items.

3.8 SOFTWARE CONFIGURATION MANAGEMENT

Software Configuration Management (SCM) is extremely important from the view of deployment of software applications. SCM controls deployment of new software versions. Software configuration management can be integrated with an automated solution that manages distributed deployment. This helps companies to bring out new releases much more efficiently and effectively. It also reduces cost, risk and accelerates time.

A current IT department of an organisation has complex applications to manage. These applications may be deployed on many locations and are critical systems. Thus, these systems must be maintained with very high efficiency and low cost and time. The problem for IT organisations has increased tremendously as some of the organisations may need to rebuild and redeploy new software versions a week over multi-platform global networks.

In such a situation, if rebuilding of application versions is built or deployed manually using a spreadsheet, then it requires copying of rebuilt software to many software locations, which will be very time consuming and error prone. What about an approach where newer software versions are automatically built and deployed into several distributed locations through a centralised control. For example, assume that IGNOU has a data entry software version 1.0 for entering assignment marks which is deployed all the RCs. In case its version 1.1 is to be deployed, if the re-built software need to be sent and deployed manually, it would be quite troublesome. Thus, an automatic deployment tool will be of great use under the control of SCM.

What do we need?

We need an effective SCM with facilities of automatic version control, access control, automatic re-building of software, build audit, maintenance and deployment. Thus, SCM should have the following facilities:

- Creation of configuration

- This documents a software build and enables versions to be reproduced on demand
- Configuration lookup scheme that enables only the changed files to be rebuilt. Thus, entire application need not be rebuilt.
- Dependency detection features even hidden dependencies, thus ensuring correct behaviour of the software in partial rebuilding.
- Ability for team members to share existing objects, thus saving time of the team members.

Figure 3.8 shows a simple Configuration Management based rebuilding and deployment process.

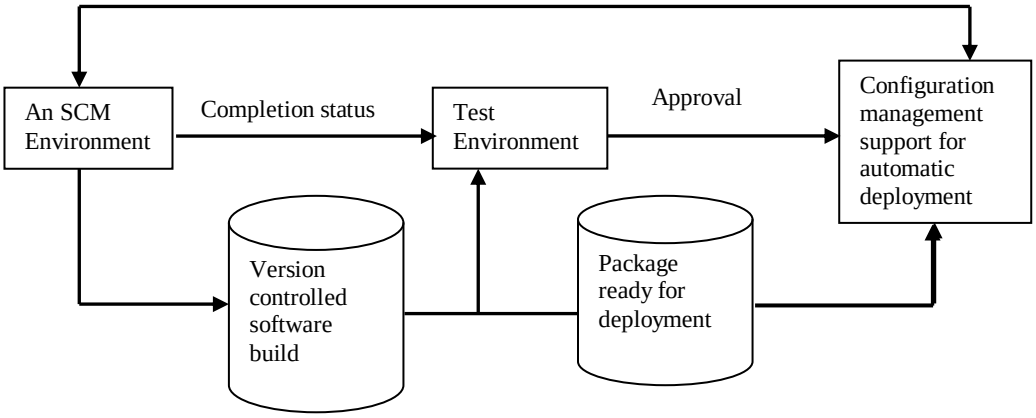

Figure 3.8: Simple configuration management environment

3.9 SOFTWARE PROJECT MANAGEMENT AND CASE TOOLS

Developing commercial software is not a single-player activity. It is invariably a collaborative team activity. The development of business-critical systems are also too risky, or are too large for a single developer to deliver the product in a stipulated time and quality frame.

A software team involves designers, developers, and testers who work together for delivering the best solution in the shortest time. Sometimes, these teams can be geographically dispersed. Managing such a team may be a major change requests, and task management.

The CASE tools help in effective management of teams and projects. Let us look into some of the features of CASE in this respect:

- Sharing and securing the project using the user name and passwords.
- Allowing reading of project related documents
- Allowing exclusive editing of documents
- Linking the documents for sharing
- Automatically communicative change requests to approver and all the persons who are sharing the document.
- You can read change requests for yourself and act on them accordingly.
- Setting of revision labels so that versioning can be done.
- Any addition or deletion of files from the repository is indicated.
- Any updating of files in repository is automatically made available to users.
- Conflicts among versions are reported and avoided
- Differences among versions can be visualized.

- The linked folder, topics, and change requests to an item can be created and these items if needed can be accessed.
- It should have reporting capabilities of information

The project management tools provide the following benefits:

- They allow control of projects through tasks so control complexity.
- They allow tracking of project events and milestones.
- The progress can be monitored using Gantt chart.
- Web based project related information can be provided.
- Automatic notifications and mails can be generated.

Some of the features that you should look into project management software are:

- It should support drawing of schedules using PERT and Gantt chart.
- It should be easy to use such that tasks can be entered easily, the links among the tasks should be easily desirable.
- Milestones and critical path should be highlighted.
- It should support editing capabilities for adding/deleting/moving tasks.
- Should map timeline information against a calendar.
- Should allow marking of durations for each task graphically.
- It should provide views tasks, resources, or resource usage by task.
- Should be useable on network and should be able to share information through network.

☞ **Check Your Progress 3**

1) How do CASE tools support quality?

……

……

2) What is the role of CASE in software configuration management?

……

……

3) Can CASE tools result in perfect project management?

……

……

3.10 SUMMARY

This unit provides an introduction to CASE tools in action. The CASE tools are primarily used to automate the stages of the Software Development Life Cycle. It automates most of the tasks of software engineering such as requirements analysis, modeling, designing, support for testing, and implementation, project management, software configuration management, software quality management, etc. CASE tools also help in traceability of requirements. They are useful tools for reverse engineering. Presently, many CASE tools which integrate the complete Software Development Life Cycle are available and can be deployed successfully in an organisation. These CASE tools support standard methodology, provide flexibility of environment, strong integration of various software development stages, reverse engineering, project management, and configuration management under one single environment. However, it is to be noted that all deployments of CASE have not succeeded.

Successful CASE tools are those which are simple to use support standard methodology, and reverse engineering and have a strong vendor support including training.

3.11 SOLUTIONS/ANSWERS

Check Your Progress 1

1) CASE tools are needed for
   - Development of cost effective software
   - Minimisation of development time
   - Standardising the software development process
   - Avoiding repetition and maximising reuse
   - Development of better quality product
   - Collaborative developments etc.

2) CASE tools
   - Follow standard methodology
   - Allow integration of information
   - Allow traceability
   - Help improving of quality
   - Reduce cost
   - Support reverse engineering
   - Provide on-line help
   - Check for inconsistency of information
   - Provide tools for configuration management and project management.

Check Your Progress 2

1) Web security test tools, link checkers, site optimization tools, server management tools

2) Although requirements engineering phases viz., requirement elicitation, specification, validation, requirements management, have some part being played by CASE tools, yet it is the stage of requirement elicitation where CASE tools are least helpful as this stage requires interaction with the users.

3) The design tools support the following:
   a. A strong consistent data dictionary
   b. Support for presenting design architecture and data design.
   c. Support for traceability and integration with test scenarios
   d. Optimisation of models and process flow charts.

4) Test planning, Integration with automatic testing tool, Traceability, Flagging the possible candidates for change.

Check Your Progress 3

1) The quality of the software is supported at all the Life Cycle stages of software development with the help of CASE tools. For example, in the analysis phase CASE tools may automatically flag requirement ambiguity in

the design phase they may point out any inconsistency, in the test phase they may point out the reliability of software, etc.

2) CASE tools control the baseline configuration items thus support a strict change control such that any changes made during any life cycle phase results in automatic flagging of the various documents of various stages of SDLC affected by that change. They also help in version control.

3) No. They are only predictors of various failures to meet the schedule of the project.

3.12 FURTHER READINGS

1) *Software Engineering, Sixth Edition, 2001*, Ian Sommerville; Pearson Education.
2) *Software Engineering – A Practitioner's Approach*, Roger S. Pressman; McGraw-Hill International Edition.

Reference Websites

- **http://www.rspa.com**
- **http://www.ieee.org**