Figure: PERT chart representation of the MIS problem

15. CASE

CASE stands for **C**omputer **A**ided **S**oftware **E**ngineering. It means, development and maintenance of software projects with help of various automated software tools.

Benefits of CASE tools

- Reduces the software development time and cost by automating many repetitive manual tasks.
- Helps to create good quality documentation and thus provides a better quality product.
- Helps to create more maintainable software systems.
- Reduces the burden of software engineer.
- Provides more structured and ordered development methodology.

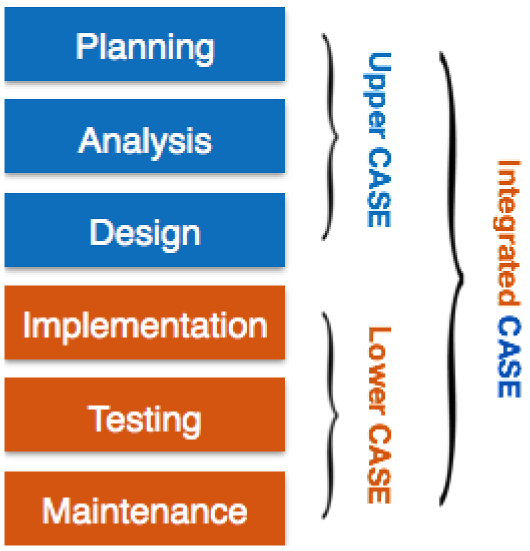
Characteristics of a successful CASE tool

- It should support standard software development methodology and modelling techniques.
- It must provide an integrated environment for software development.
- It must be flexible, so that user can make necessary changes.
- It should support reverse engineering process.
- It must support integration with automated testing tools.
- It must provide online help.

CASE CLASSIFICATIONS

- **Upper Case Tools** – Tools that mainly concentrate on high level activities of SDLC. Upper CASE tools are used in planning, analysis and design stages of SDLC.
- **Lower Case Tools** – tools mainly focuses on the implementation of the system. Lower CASE tools are used in implementation, testing and maintenance.
- **Integrated Case Tools** - Integrated CASE tools are helpful in all the stages of SDLC, from Requirement gathering to Testing and documentation.

CASE tools can be grouped together if they have similar functionality, process activities and capability of getting integrated with other tools.



CASE ENVIRONMENT

- A schematic representation of a CASE environment is shown below

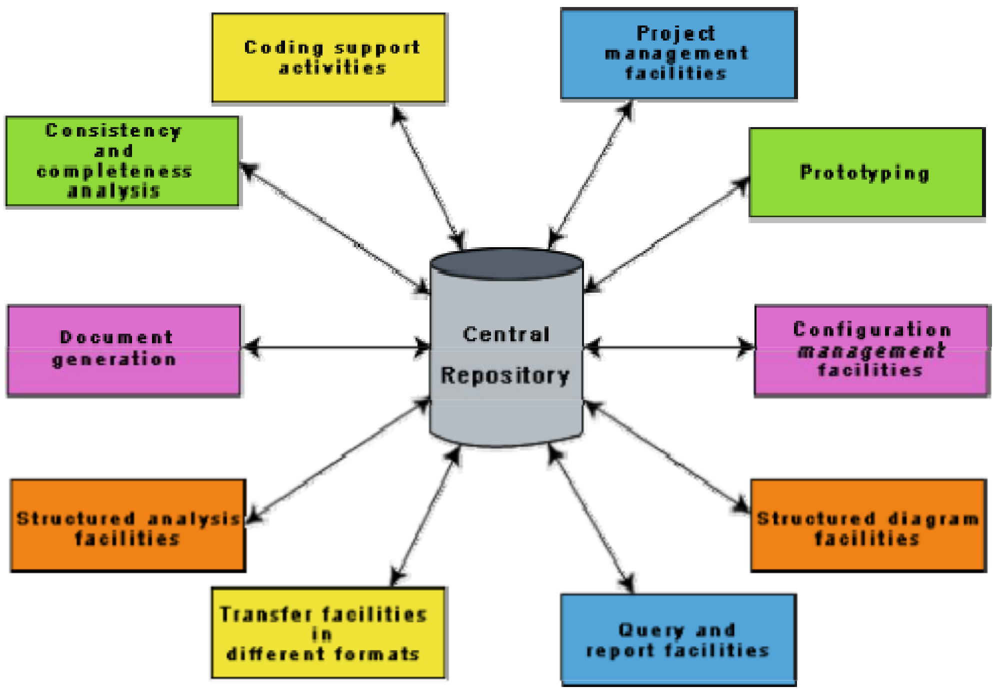

Coding support activities
Project management facilities
Consistency and completeness analysis
Prototyping
Document generation
Central Repository
Configuration management facilities
Structured analysis facilities
Structured diagram facilities
Transfer facilities in different formats
Query and report facilities

Figure A CASE Environment

- Although individual CASE tools are useful, the true power of a tool set can be realized only when these set of tools are integrated into a common framework or environment. CASE tools are characterized by the stage or stages of software development life cycle on which they focus. Since different tools covering different stages share common information, it is required that they integrate through some central repository to have a consistent view of information associated with the software development artifacts.
- This central repository is usually a data dictionary containing the definition of all composite and elementary data items. Through the central repository all the CASE tools in a CASE environment

share common information among themselves. Thus a CASE environment facilities the automation of the step-by-step methodologies for software development.

- **Central Repository** - CASE tools require a central repository, which can serve as a source of common, integrated and consistent information. Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as data dictionary.

CASE TOOLS

- CASE tools are set of software application programs, which are used to automate SDLC activities. CASE tools are used by software project managers, analysts and engineers to develop software system.
- There are number of CASE tools available to simplify various stages of Software Development Life Cycle such as Analysis tools, Design tools, Project management tools, Database Management tools, Documentation tools are to name a few.
- Use of CASE tools accelerates the development of project to produce desired result and helps to uncover flaws before moving ahead with next stage in software development.
- CASE tools can be broadly divided into the following parts based on their use at a particular SDLC stage:
- Various CASE tools are listed below:

1. **Diagram tools**
   These tools are used to represent system components, data and control flow among various software components and system structure in a graphical form. For example, Flow Chart Maker tool for creating state-of-the-art flowcharts.
2. **Process Modelling Tools**

   Process modelling is method to create software process model, which is used to develop the software. Process modelling tools help the managers to choose a process model or modify it as per the requirement of software product. For example, EPF Composer

3. **Project Management Tools**

   These tools are used for project planning, cost and effort estimation, project scheduling and resource planning. Managers have to strictly comply project execution with every mentioned step in software project management. Project management tools help in storing and sharing project information in real-time throughout the organization. For example, Creative Pro Office, Trac Project, Basecamp.

4. **Documentation Tools**

   Documentation in a software project starts prior to the software process, goes throughout all phases of SDLC and after the completion of the project.

   n tools generate documents for technical users and end users. Technical users are mostly in-house professionals of the development team who refer to system manual, reference manual, training manual, installation manuals etc. The end user documents describe the functioning and how-to of the system such as user manual. For example, Doxygen, DrExplain, Adobe RoboHelp for documentation.

5. **Analysis Tools**

   These tools help to gather requirements, automatically check for any inconsistency, inaccuracy in the diagrams, data redundancies or erroneous omissions. For example, Accept 360, Accompa, Case-Complete for requirement analysis, Visible Analyst for total analysis.

6. **Design Tools**

These tools help software designers to design the block structure of the software, which may further be broken down in smaller modules using refinement techniques. These tools provides detailing of each module and interconnections among modules. For example, Animated Software Design

7. **Configuration Management Tools**

   An instance of software is released under one version. Configuration Management tools deal with –

   Version and revision management

   Baseline configuration management

   Change control management

   CASE tools help in this by automatic tracking, version management and release management. For example, Fossil, Git, Accu REV.

8. **Change Control Tools**

These tools are considered as a part of configuration management tools. They deal with changes made to the software after its baseline is fixed or when the software is first released. CASE tools automate change tracking, file management, code management and more. It also helps in enforcing change policy of the organization.

9. **Programming Tools**

These tools consist of programming environments like IDE (Integrated Development Environment), in-built modules library and simulation tools. These tools provide comprehensive aid in building software product and include features for simulation and testing. For example, Cscope to search code in C, Eclipse.

10. **Prototyping Tools**

Software prototype is simulated version of the intended software product. Prototype provides initial look and feel of the product and simulates few aspect of actual product.

Prototyping CASE tools essentially come with graphical libraries. They can create hardware independent user interfaces and design. These tools help us to build rapid prototypes based on existing information. In addition, they provide simulation of software prototype. For example, Serena prototype composer, Mockup Builder.

11. **Web Development Tools**

These tools assist in designing web pages with all allied elements like forms, text, script, graphic and so on. Web tools also provide live preview of what is being developed and how will it look after completion. For example, Fontello, Adobe Edge Inspect, Foundation 3, Brackets.

12. Quality Assurance Tools

Quality assurance in a software organization is monitoring the engineering process and methods adopted to develop the software product in order to ensure conformance of quality as per organization standards. QA tools consist of configuration and change control tools and software testing tools. For example, SoapTest, AppsWatch, JMeter.

13. Maintenance Tools

Software maintenance includes modifications in the software product after it is delivered. Automatic logging and error reporting techniques, automatic error ticket generation and root cause Analysis are few CASE tools, which help software organization in maintenance phase of SDLC. For example, Bugzilla for defect tracking, HP Quality Center.

Architecture of a CASE environment

- The architecture of a typical modern CASE environment is shown below. The important components of a modern CASE environment are user interface, tool set, object management system (OMS), and a repository. Characteristics of a tool set have been discussed earlier.

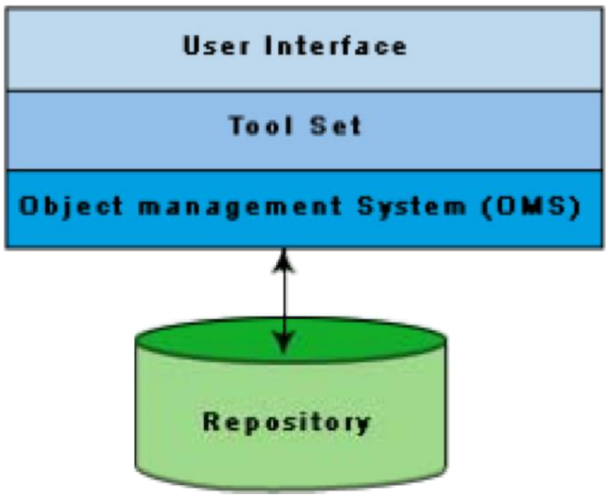


Figure: Architecture of a Modern CASE Environment

User Interface

The user interface provides a consistent framework for accessing the different tools thus making it easier for the users to interact with the different tools and reducing the overhead of learning how the different tools are used.

Object Management System (OMS) and Repository

Different case tools represent the software product as a set of entities such as specification, design, text data, project plan, etc. The object management system maps these logical entities such into the underlying storage management system (repository). The commercial relational database management systems are geared towards supporting large volumes of information structured as simple relatively short records. There are a few types of entities but large number of instances. By contrast, CASE tools create a large number of entity and relation types with perhaps a few instances of each. Thus the object management system takes care of appropriately mapping into the underlying storage management system.

WORKBENCHES

A coherent set of tools that is designed to support related software process activities such as analysis, design or testing

- Workbenches integrate several CASE tools into one application to support specific software-process activities. Hence they achieve

- ➢ A homogeneous and consistent interface (presentation integration).
- ➢ Easy invocation of tools and tool chains (control integration).
- ➢ Access to a common data set managed in a centralized way (data integration).

- CASE workbenches can be further classified into following 8 classes:

1. Business planning and modelling
2. Analysis and design
3. User-interface development
4. Programming
5. Verification and validation (Testing Workbench)
6. Maintenance and reverse engineering
7. Configuration management
8. Project management

ANALYSIS & DESIGN WORKBENCH

- Analysis and design workbenches support system modelling during both requirements engineering and system design.
- These workbenches may support a specific design method or may provide support for a creating several different types of system models.
- Analysis & Design workbench performs various analysis required during the development phase. It will automatically generate the various software design forms like DFDs, UML diagrams, Algorithms etc.
- It will automatically generate suitable documents associated with analysis and design phases of the software.
- The various components of Analysis & Design workbench is shown below:

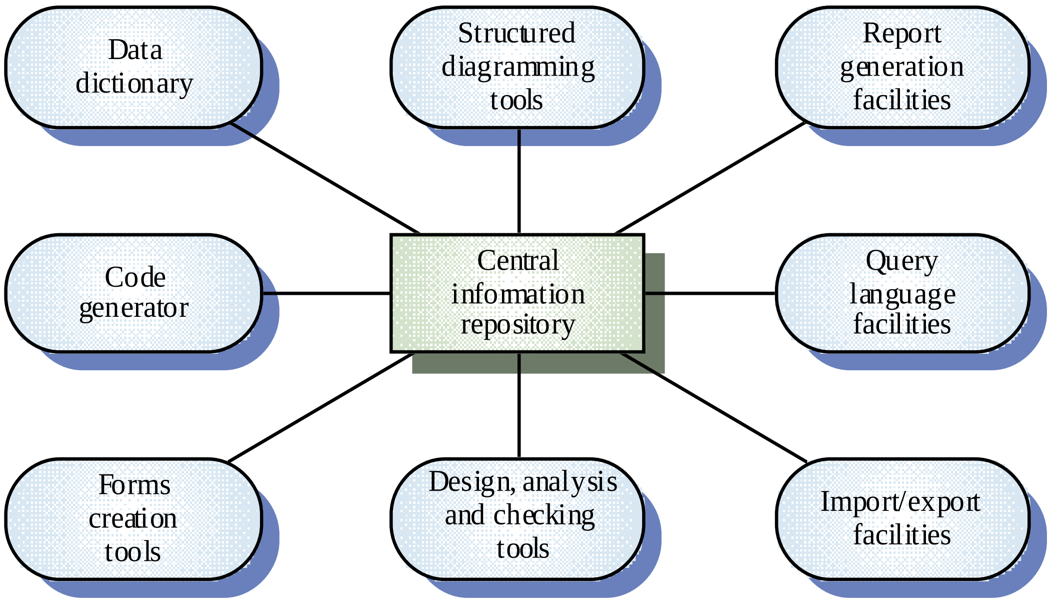
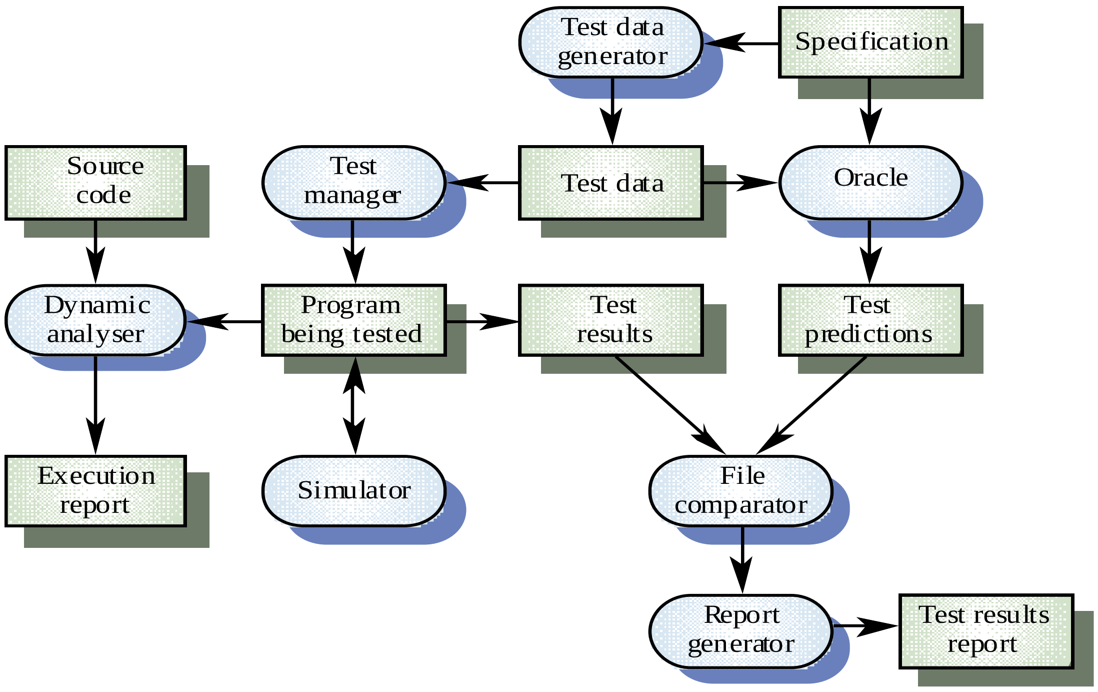

Data dictionary
Structured diagramming tools
Report generation facilities
Code generator
Central information repository
Query language facilities
Forms creation tools
Design, analysis and checking tools
Import/export facilities

Figure: Components of Analysis & Design Workbench

TESTING WORKBENCH

- Testing is an expensive process phase.
- Testing workbenches provide a range of tools to reduce the time required and total testing costs
- Most testing workbenches are open systems because testing needs are organization-specific
- Difficult to integrate testing with closed design and analysis workbenches.
- Scripts may be developed for user interface simulators and patterns for test data generators
- Test outputs may have to be prepared manually for comparison
- The various components of testing workbenches are shown below:



REFERENCES

Text books

1. Fundamentals of Software Engineering – Rajib Mall
2. Software Engineering Principles & Practices – Rohit khurana
3. Software Engineering A Practioner's approach – Roger S Pressman
4. Software Engineering –Sommerville
5. An integrated Approach to Software Engineering – Pankaj Jalote

Websites

1. **http://istqbexamcertification.com/**