SEPM
ASSIGNMENT 1

Done By

Harikrishnan V

CS6A

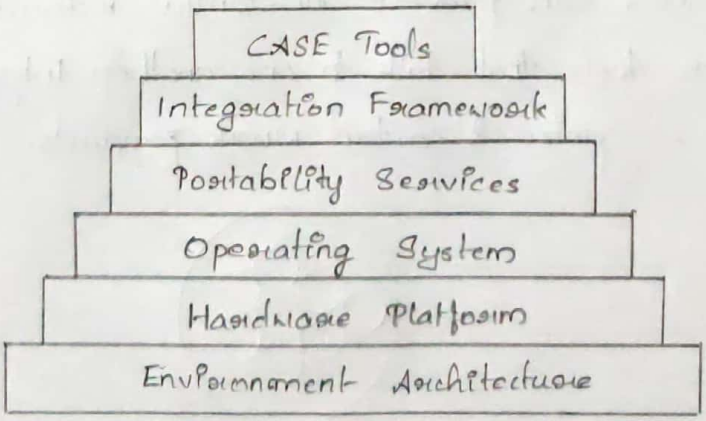
Roll No : 27

CASE BUILDING BLOCKS

Computer-aided software engineering (CASE) tools assist software engineering managers & practitioners in every activity associated with the software process.

CASE Building Blocks

- CASE tools.
- Integration framework (specialized programs allowing CASE tools to communicate with one another).
- Portability services (allow CASE tools & their integration framework to migrate across different operating systems & hardware platforms without significant adaptive maintenance).
- Operating system (database & object management services).
- Hardware platform.
- Environmental architecture (hardware & system support).



CASE Tool Components

- Integration framework
- Specialized programs allowing CASE tools to communicate

- Portability services
- Operating system
- Database & object management services
- Hardware platform

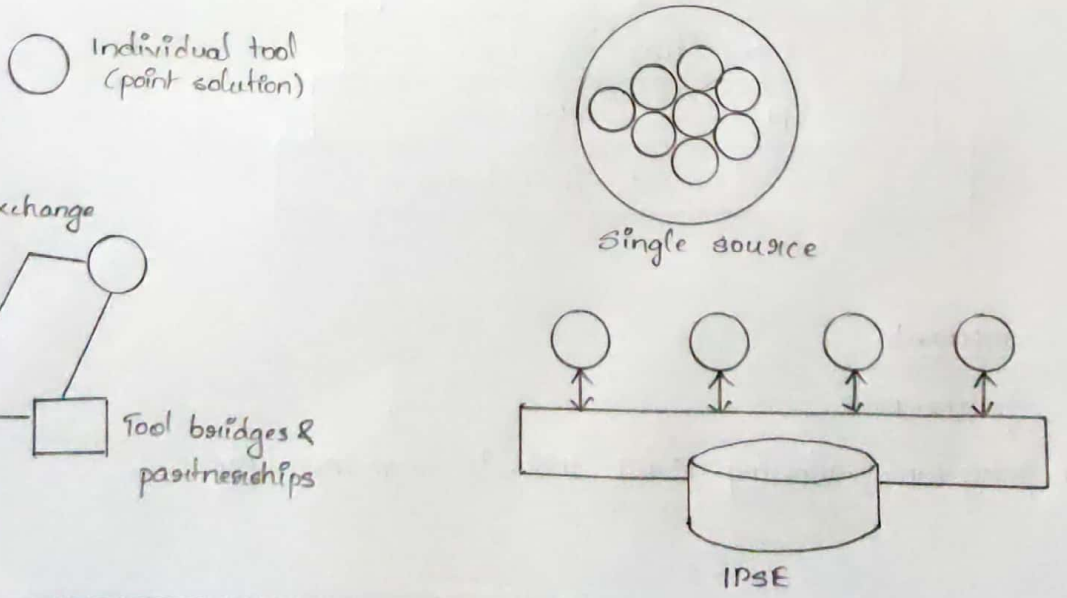
The environment architecture, composed of the hardware platform & system support, lays the ground work for CASE. But the CASE environment itself demands other building blocks.

A set of portability services provides a bridge between CASE tools & their integration framework & the environment architecture.

The integration framework is a collection of specialized programs that enables individual CASE tools to communicate with one another, to create a project database, & to exhibit the same look & feel to the end-user.

Portability services allow CASE tools & their integration framework to migrate across different hardware platforms & operating systems without significant adaptive maintenance.

Point-solution CASE tools can provide substantial individual benefit, but a software team needs tools that talk to one another. Integrated tools help the team develop, organize, & control work products.



At the low end of the integration spectrum is the individual (point solution) tool. When individual tools provide facilities for data exchange, the integration level is improved slightly. Such tools produce output in a standard format that should be compatible with other tools that can read the format. In some cases, the builders of complementary CASE tools work together to form a bridge between the tools.

Using this approach, the synergy between the tools can produce end products that would be difficult to create using either tool separately. Single-source integration occurs when a single CASE tools vendor integrates a number of different tools & sells them as a package. Although this approach is quite effective, the closed architecture of most single-source environments precludes easy addition of tools from other vendors.

At the high end of the integration spectrum is the integrated project support environment (IPSE). CASE tool vendors use IPSE standards to build tools that will be compatible with the IPSE & therefore compatible with one another.

TAXONOMY OF CASE TOOLS

CASE tools can be classified by function, by their role as instruments for managers or technical people, by their use in the various steps of the software engineering process, by the environment architecture that supports them, or even by their origin or cost. The taxonomy presented here uses functions as a primary criterion.

- Business process engineering tools - represent business data objects, their relationships, & flow of the data objects between company business areas.
- Process modeling & management tools - represent key elements of processes & provide links to other tools that provide support to defined process activities.
- Project planning tools - used for cost & effort estimation, & project scheduling.
- Risk analysis tools - help project managers build risk tables by providing detailed guidance in the identification & analysis of risks.
- Requirements tracing tools - provide systematic database-like approach to tracking requirement status beginning with specification.
- Metrics & management tools - management oriented tools capture project specific metrics that provide an overall indication of productivity or quality, technically oriented metrics determine metrics that provide greater insight into the quality of design or code.
- Documentation tools - provide opportunities for improved productivity by reducing the amount of time needed to produce work products.
- System software tools - network system software, object management services, distributed component support, & communications software.

- Quality assurance tools - metrics tools that audit source code to determine compliance with language standards and tools that extract metrics to project the quality of software being built.
- Database management tools - RDMS & OODMS serve as the foundation for the establishment of the CASE repository.
- Software configuration management tools - uses the CASE repository to assist with all SCM tasks.
- Analysis & design tools - enable the software engineer to create analysis & design models of the system to be built, perform consistency checking between models.
- PRO/SIM tools - prototyping & simulation tools provide software engineers with ability to predict the behavior of real-time systems before they are built & the creation of interface mockups for customer review.
- Interface design & development tools - toolkits of interface components, often part environment with a GUI to allow rapid prototyping of user interface designs.
- Prototyping tools - enable rapid definition of screen layouts, data design & report generation.
- Programming tools - compilers, editors, debuggers, OO programming environments, 4th generation languages, graphical programming environments, applications generators, & database query generators.
- Web development tools - assist with the generation of web page text, graphics, forms, scripts, applets, etc.
- Integration & testing tools - In their directory of software testing tools, Software Quality Engineering [SQE95] defines the following testing tools categories: Data acquisition, Static measurement, Dynamic measurement,

Simulation, Test management, Cross-functional tools.

- Static analysis tools - code-based testing tools, specialized testing languages, requirements-based testing tools.
- Dynamic analysis tools - intrusive tools modify source code by inserting probes to check path coverage, assertions, or execution flow, non-intrusive tools use a separate hardware processor running in parallel with processor containing the program being tested.
- Test management tools - coordinate regression testing, compare actual & expected output, conduct batch testing, & serve as generic test drivers
- Client/server testing tools - exercise the GUI & network communications requirements for the client & server
- Reengineering tools :

→ Reverse engineering to specification tools - generate analysis & design models from source code.

→ Code restructuring & analysis tools - analyze program syntax, generate control flow graph, & automatically generates a structured program.

→ On-line system reengineering tools - used to modify on-line DBMS.

INTEGRATED CASE ENVIRONMENT

Integration of a variety of tools & information that enables closure of communication among tools, between people & across the software process.

The benefits of integrated CASE (I-CASE) include

(1) Smooth transfer of information from one tool to another & one software engineering step to the next;

(2) A reduction in the effort required to perform umbrella activities such as software configuration management, quality assurance, & document production;

(3) An increase in project control that is achieved through better planning, monitoring, & communication; and

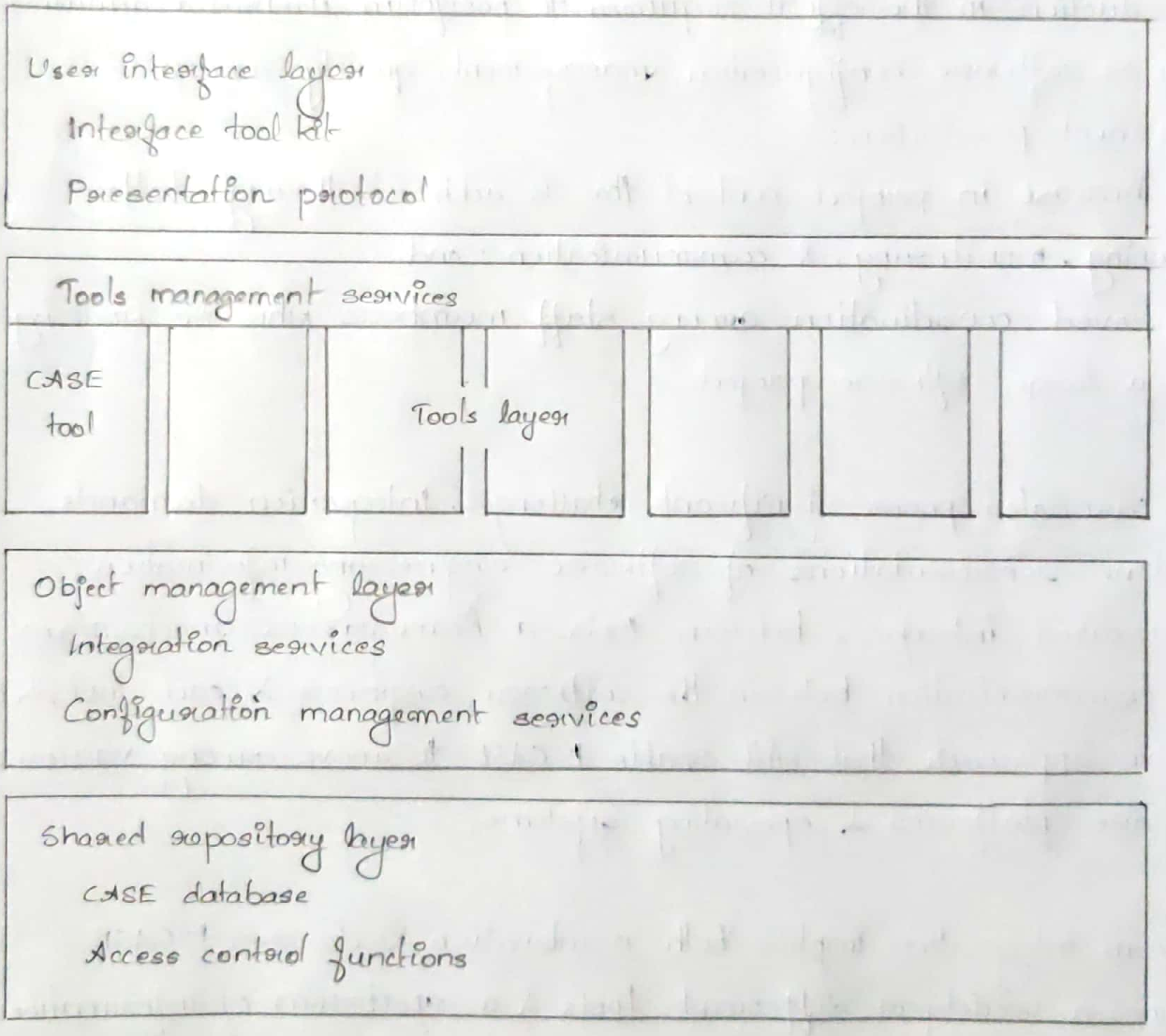
(4) Improved coordination among staff members who are working on a large software project.

But I-CASE also poses significant challenges. Integration demands consistent representations of software engineering information, standardized interfaces between tools, a homogeneous mechanism for communication between the software engineer & each tool, & an effective approach that will enable I-CASE to move among various hardware platforms & operating systems.

The term integration implies both combination & closure. I-CASE combines a variety of different tools & a spectrum of information in a way that enables closure of communication among tools, between people, & across the software process. Tools are integrated so that software engineering information is available to each tool

that needs it; usage is integrated so that a common look & feel is provided for all tools; a development philosophy is integrated, implying a standardized software engineering approach that applies modern practice & proven methods.

Integration Framework Diagram



A software engineering team uses CASE tools, corresponding methods, & a process framework to create a pool of software engineering information. The integration framework facilitates transfer of

information into & out of the pool.

To accomplish this, the following architectural components must exist: a database must be created; an object management system must be built; a tools control mechanism must be constructed; a user interface must provide a consistent pathway between actions made by the user & the tools contained in the environment. Most models of the integration framework represent these components as layers.

- The user interface layer consists of interface tool kit & presentation protocols.
- The presentation protocol is the set of guidelines that gives all CASE tools the same look & feel.
- The tools layer incorporates a set of tools management services with the case tools themselves.
- The object management layer (OML) performs the configuration management functions.
- The shared repository layer is the CASE database & the access control functions that enable the object management layer to interact with the database.

These provide

- Mechanism for sharing information among all tools contained in the environment.
- Enable changes to items to be tracked to other information items.
- Provide version control & overall configuration management.
- Allow direct access to any tool contained in the environment.
- Establish automated support for the chosen software process model.

integrating CASE tools & SCI's into a standard work break down structure.

- Enable users of each tool to experience a consistent look & feel at the human-computer interface.
- Support communication among software engineers.
- Collect both management & technical metrics to improve the process & the product.

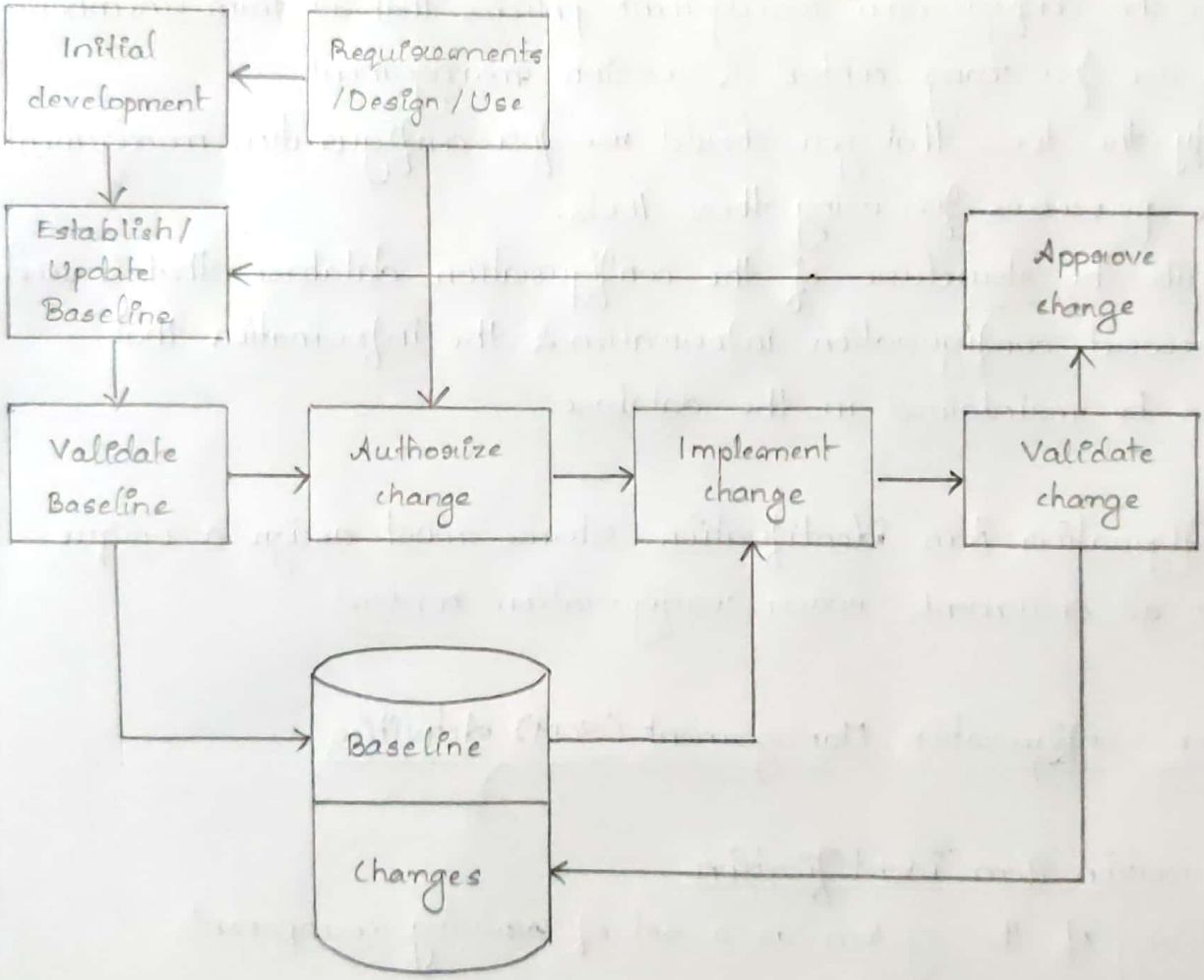
The repository for an I-CASE environment is the set of mechanisms & data structures that achieve data/tool & data/data integration. It provides the obvious functions of a database management system, but in addition, the repository performs the following functions:

- Data integrity - includes functions to validate entries to the repository & ensure consistency among related objects.
- Information sharing - provides mechanism for sharing information among multiple developers & multiple tools.

SOFTWARE CONFIGURATION MANAGEMENT

Software Configuration Management (SCM) is a set of management disciplines within the software engineering process to develop a baseline.

Software Configuration Management encompasses the disciplines & techniques of initiating, evaluating & controlling change to software products during & after the software engineering process.



Configuration managers are responsible for keeping track of the differences between software versions, for ensuring that new versions are derived in a controlled way & for releasing new versions to the right customers at the right time.

The CM plan should be organised into a number of sections that:

1. Define what is to be managed & the scheme that you should use to identify these entities.
2. Set out who is responsible for the configuration management procedures & for submitting controlled entities to the configuration management team.
3. Define the configuration management policies that all team members must use for change control & version management.
4. Specify the tools that you should use for configuration management & the processes for using these tools.
5. Describe the structure of the configuration database that is used to record configuration information & the information that should be maintained in the database.

The configuration item identification scheme must assign a unique name to all documents under configuration control.

Software Configuration Management (SCM) Activities

- Configuration item identification
  → Modeling of the system as a set of evolving components.
- Promotion management
  → Creation of versions for other developers.
- Release management
  → Creation of versions for the clients & users.

- **Branch management**
  → Management of concurrent development.
- **Variant management**
  → Management of versions intended to coexist.
- **Change management**
  → Handling, approval & tracking of change requests.

SCM Roles

- **Configuration Manager**
  → Responsible for identifying configuration items.
- **Change control board member**
  → Responsible for approving or rejecting change requests.
- **Developer**
  → Creates promotions triggered by change requests or the normal activities of development.
- **Auditor**
  → Responsible for the selection & evaluation of promotions for release & for ensuring the consistency & completeness of this release.

Configuration Item

An aggregation of hardware, software, or both, that is designated for configuration management & treated as a single entity in the configuration management process.

Baseline

A specification or product that has been formally reviewed & agreed to

by responsible management, that thereafter serves as the basis for further development, & can be changed only through formal change control procedures.

SCM Directories

- **Programmer's Directory**
  → Library for holding newly created or modified software entities.
- **Master Directory**
  → Manages the current baseline(s) & for controlling changes made to them.
- **Software Repository**
  → Archive for the various baselines released for general use.

Version

An initial release or re-release of a configuration item associated with a complete compilation or recompilation of the item.

Revision

Change to a version that corrects only errors in the design/code, but does not affect the documented functionality.

Release

The formal distribution of an approved version.