SOFTWARE ENGINEERING

LAB ASSIGNMENT-1

GROUP MEMBERS:
AVANI KHANDELWAL- EN18CS301061
BURHAUDDIN- EN18CS301070
CHIRAYU YADAV- EN18CS301072
DHRUV BOTHRA- EN18CS301079

INDEX

S.no	title	Page no.
1.	Differentiate between API, plug-in, component, pattern, design pattern and framework using an example program. Explain how these terms are related with software engineering.	2-4
2.	What are CASE tools? List various CASE tools available for different stages of software development or for overall development.	4-5
3.	List the types of CASE tools and its importance.	5-9
4.	Detailed case study of any of the CASE tools.	9-13
5.	What is the impact of recent developments in industry on such CASE tools.	13-14

LAB ASSIGNMENT-I

Q1. Differentiate between API, plug-in, component, pattern, design pattern and framework using an example program. Explain how these terms are related with software engineering.

Solution:

- **API:** API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.

  Code: API

  ```
  import requests
  url = "https://inventory-fac4.restdb.io/rest/motorbikes"
  headers = {
  'content-type': "application/json",
  'x-apikey': "560bd47058e7ab1b2648f4e7",
  'cache-control': "no-cache
  }
  response = requests.request("GET", url, headers=headers)
  print(response.text)
  ```

- **PLUG-IN:** In computing, a plug-in (or plugin, add-in, addin, add-on, or addon) is a software component that adds a specific feature to an existing computer program. When a program supports plug-ins, it enables customization. A theme or skin is a preset package containing additional or changed graphical appearance details, achieved by the use of a graphical user interface (GUI) that can be applied to specific software and websites to suit the purpose, topic, or tastes of different users to customize the look and feel of a piece of computer software or an operating system front-end GUI (and window managers).

  code: plug-in

  ```
   <?php
  /*Use this function to replace a single word*/
  function renym_wordpress_typo_fix( $text ) {
  ```

```
		return str_replace( 'wordpress', 'WordPress', $text );
	}
	add_filter( 'the_content', 'renym_wordpress_typo_fix' );

	/*Or use this function to replace multiple words or phrases at once*/
	function renym_content_replace( $content ) {
		$search  = array( 'wordpress', 'goat', 'Easter', '70', 'sensational' );
		$replace = array( 'WordPress', 'coffee', 'Easter holidays', 'seventy', 
	'extraordinary' );
		return str_replace( $search, $replace, $content );
	}
	add_filter( 'the_content', 'renym_content_replace' );

	/*Use this function to add a note at the end of your content*/
	function renym_content_footer_note( $content ) {
		$content .= '<footer class="renym-content-footer">Thank you for 
	reading this tutorial. Maybe next time I will let you buy me a coffee! For more 
	WordPress tutorials visit our <a href="http://wpexplorer.com/blog" 
	title="WPExplorer Blog">Blog</a></footer>';
		return $content;
	}
	add_filter( 'the_content', 'renym_content_footer_note' );
	?>
```

- **COMPONENT**: A component is a modular, portable, replaceable, and reusable set of well-defined functionality that encapsulates its implementation and exporting it as a higher-level interface. A component is a software object, intended to interact with other components, encapsulating certain functionality or a set of functionalities. It has an obviously defined interface and conforms to a recommended behaviour common to all components within an architecture.
- **DESIGN PATTERN**: Design patterns are used to represent some of the best practices adapted by experienced object-oriented software developers. A

design pattern systematically names, motivates, and explains a general design that addresses a recurring design problem in object-oriented systems. It describes the problem, the solution, when to apply the solution, and its consequences. It also gives implementation hints and examples.

- **FRAMEWORK**: A framework, or software framework, is a platform for developing software applications. It provides a foundation on which software developers can build programs for a specific platform. It is a Standard way to build and deploy applications. Software Process Framework is a foundation of complete software engineering process. Software process framework includes all set of umbrella activities. It also includes number of framework activities that are applicable to all software projects.

Q2. What are CASE tools? List various CASE tools available for different stages of software development or for overall development.

Solution: The workshop for software engineering has been called an integrated project support environment (discussed later in this chapter) and the tools that fill the workshop are collectively called computer-aided software engineering.

CASE provides the software engineer with the ability to automate manual activities and to improve engineering insight. Like computer-aided engineering and design tools that are used by engineers in other disciplines, CASE tools help to ensure that quality is designed in before the product is built.

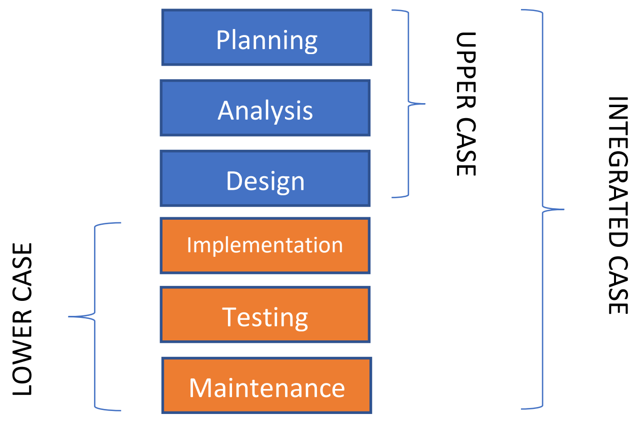
The essential idea of CASE tools is that in-built programs can help to analyze developing systems in order to enhance quality and provide better outcomes. Throughout the 1990, CASE tool became part of the software lexicon, and big companies like IBM were using these kinds of tools to help create software.

Various tools are incorporated in CASE and are called CASE tools, which are used to support different stages and milestones in a software development life cycle.

Case tools that are available at various stage for software development:

- **Central Repository**: A central repository is required by the tools to serve as a common source of integrated and consistent information. The central place of storage consisting of specifications of product, documents requirement, diagrams and reports and information about the management is a central repository. The central repository also acts as a data dictionary.

- **Upper**: Planning, analysis, and designing of different stages of the software development life cycle can be performed using upper case.
- **Lower**: Implementation, testing, and maintenance can be performed using lower case.
- **Integrated**: All the stages of the software development life cycle right from the gathering of requirements for testing and documentation can be performed using integrated tools.



Q3. List the types of CASE tools and its importance.

Solution: There are number of CASE tools available to simplify various stages of Software Development Life Cycle :-

Business process engineering tools. By modeling the strategic information requirements of an organization, business process engineering tools provide a "meta-model" from which specific information systems are derived. Rather than focusing on the requirements of a specific application, business information is modeled as it moves between various organizational entities within a company. The primary objective for tools in this category is to represent business data objects, their relationships, and how these data objects flow between different business areas within a company.

Process modeling and management tools. If an organization works to improve a business (or software) process, it must first understand it. Process modeling tools (also called *process technology* tools) are used to represent the key elements of a process so that it can be better understood. Such tools can also provide links to process descriptions that help those involved in the process to understand the work tasks that are required to perform it. Process management tools provide links to other tools that provide support to defined process activities.

Project planning tools. Tools in this category focus on two primary areas: software project effort and cost estimation and project scheduling. Estimation tools compute estimated effort, project duration, and recommended number of people for a project. Project scheduling tools enable the manager to define all project tasks create a task network (usually using graphical input), represent task interdependencies, and model the amount of parallelism possible for the project.

Risk analysis tools. Identifying potential risks and developing a plan to mitigate, monitor, and manage them is of paramount importance in large projects. Risk analysis tools enable a project manager to build a risk table by providing detailed guidance in the identification and analysis of risks.

Project management tools. The project schedule and project plan must be tracked and monitored on a continuing basis. In addition, a manager should use tools to collect metrics that will ultimately provide an indication of software product quality. Tools in the category are often extensions to project planning tools.

Requirements tracing tools. When large systems are developed, things "fall into the cracks." That is, the delivered system does not fully meet customer specified requirements. The objective of requirements tracing tools is to provide a systematic approach to the isolation of requirements, beginning with the customer request for proposal or specification. The typical requirements tracing tool combines human interactive text evaluation with a database management system that stores and categorizes each system requirement that is "parsed" from the original RFP or specification.

Metrics and management tools. Software metrics improve a manager's ability to control and coordinate the software engineering process and a practitioner's ability to improve the quality of the software that is produced. Today's metrics or measurement tools focus on process and product characteristics. Management-oriented tools capture project specific metrics that provide an overall indication of productivity or quality. Technically oriented tools determine technical metrics that provide greater insight into the quality of design or code.

Documentation tools. Document production and desktop publishing tools support nearly every aspect of software engineering and represent a substantial "leverage" opportunity for all software developers. Most software development organizations spend a substantial amount of time developing documents, and in many cases the documentation process itself is quite inefficient. It is not unusual for a software development organization to spend as much as 20 or 30 percent of all software development effort on documentation. For this reason, documentation tools provide an important opportunity to improve productivity.

System software tools. CASE is a workstation technology. Therefore, the CASE environment must accommodate high-quality network system software, object management services, distributed component support, electronic mail, bulletin boards, and other communication capabilities.

Quality assurance tools. The majority of CASE tools that claim to focus on quality assurance are actually metrics tools that audit source code to determine compliance with language standards. Database management software serves as a foundation for the establishment of a CASE database (repository) that we have called the project database. Given the emphasis on configuration objects, database management tools for CASE are evolving from relational database management systems to object oriented database management systems.

Software configuration management tools. Software configuration management lies at the kernel of every CASE environment. Tools can assist in all five majors SCM tasks—identification, version control, change control, auditing, and status accounting. The CASE database provides a mechanism for identifying each configuration item and relating it to other items; the change control process can be implemented with the aid of specialized tools; easy access to individual configuration items facilitate the auditing process; and CASE communication tools can greatly improve status accounting.

Analysis and design tools. Analysis and design tools enable a software engineer to create models of the system to be built. The models contain a representation of data, function, and behavior (at the analysis level) and characterizations of data, architectural, component-level, and interface design.1 By performing consistency and validity checking on the models, analysis and design tools provide a software engineer with some degree of insight into the analysis representation and help to eliminate errors before they propagate into the design, or worse, into implementation itself.

PRO/SIM tools. PRO/SIM (prototyping and simulation) tools provide the software engineer with the ability to predict the behavior of a real-time system prior to the time that it is built. In addition, these tools enable the software engineer to develop mock-ups of the real-time system, allowing the customer to gain insight into the function, operation and response prior to actual implementation.
Interface design and development tools. Interface design and development tools are actually a tool kit of software components (classes) such as menus, buttons, window structures, icons, scrolling mechanisms, device drivers, and so forth.

Prototyping tools. A variety of different prototyping tools can be used. Screen painters enable a software engineer to define screen layout rapidly for interactive applications. More sophisticated CASE prototyping tools enable the creation of a data design, coupled with both screen and report layouts. Many analysis and design tools have extensions that provide a prototyping option. PRO/SIM tools generate skeleton Ada and C source code for engineering (real-time) applications. Finally, a variety of fourth generation tools have prototyping features.

Programming tools. The programming tools category encompasses the compilers, editors, and debuggers that are available to support most conventional programming languages. In addition, object-oriented programming environments, fourth generation languages, graphical programming environments, application generators, and database query languages also reside within this category.

Web development tools. The activities associated with Web engineering are supported by a variety of tools for WebApp development. These include tools that assist in the generation of text, graphics, forms, scripts, applets, and other elements of a Web page.

Dynamic analysis tools. Dynamic testing tools interact with an executing program, checking path coverage, testing assertions about the value of specific variables, and otherwise instrumenting the execution flow of the program. Dynamic tools can be either intrusive or nonintrusive. An intrusive tool changes the software to be tested by inserting probes (extra instructions) that perform the activities just mentioned. Nonintrusive testing tools use a separate hardware processor that runs in parallel with the processor containing the program that is being tested.

Test management tools. Test management tools are used to control and coordinate software testing for each of the major testing steps. Tools in this category manage and coordinate regression testing, perform comparisons that ascertain differences between actual and expected output, and conduct batch testing of programs with interactive human/computer interfaces. In addition to the functions noted, many test management tools also serve as generic test drivers.

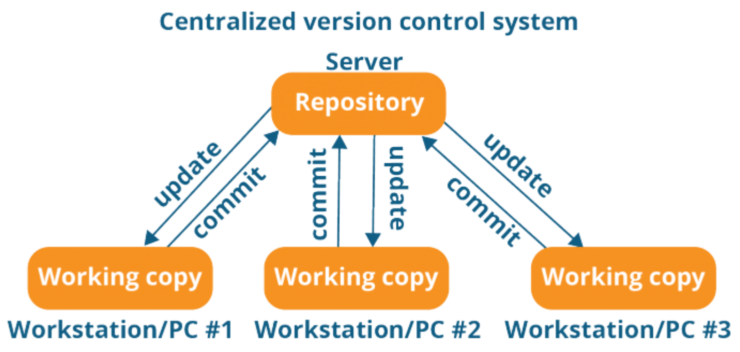
Client/server testing tools. The c/s environment demands specialized testing tools that exercise the graphical user interface and the network communications requirements for client and server.

Q4. Detailed case study of any of the CASE tools.
Solution: Git is a free, open source distributed version control system tool designed to handle everything from small to very large projects with speed and efficiency. It was created by Linus Torvalds in 2005 to develop Linux Kernel. Git has the functionality, performance, security and flexibility that most teams and individual developers need. It also serves as an important distributed version-control DevOps tool.
Git is primarily used to manage your project, comprising a set of code/text files that may change.
Version Control is the management of changes to documents, computer programs, large websites and other collection of information.
There are two types of VCS:

- Centralized Version Control System (CVCS)
- Distributed Version Control System (DVCS)

Centralized VCS

Centralized version control system (CVCS) uses a central server to store all files and enables team collaboration. It works on a single repository to which users can directly access a central server.

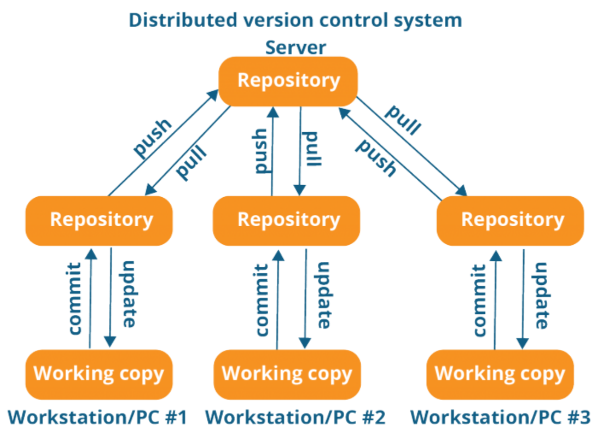


The repository in the above diagram indicates a central server that could be local or remote which is directly connected to each of the programmer's workstation.

Every programmer can extract or **update** their workstations with the data present in the repository or can make changes to the data or **commit** in the repository. Every operation is performed directly on the repository.

Distributed VCS

These systems do not necessarily rely on a central server to store all the versions of a project file. In Distributed VCS, every contributor has a local copy or "clone" of the main repository i.e. everyone maintains a local repository of their own which contains all the files and metadata present in the main repository.



every programmer maintains a local repository on its own, which is actually the copy or clone of the central repository on their hard drive. They can commit and update their local repository without any interference.
They can update their local repositories with new data from the central server by an operation called "**pull**" and affect changes to the main repository by an operation called "**push**" from their local repository.

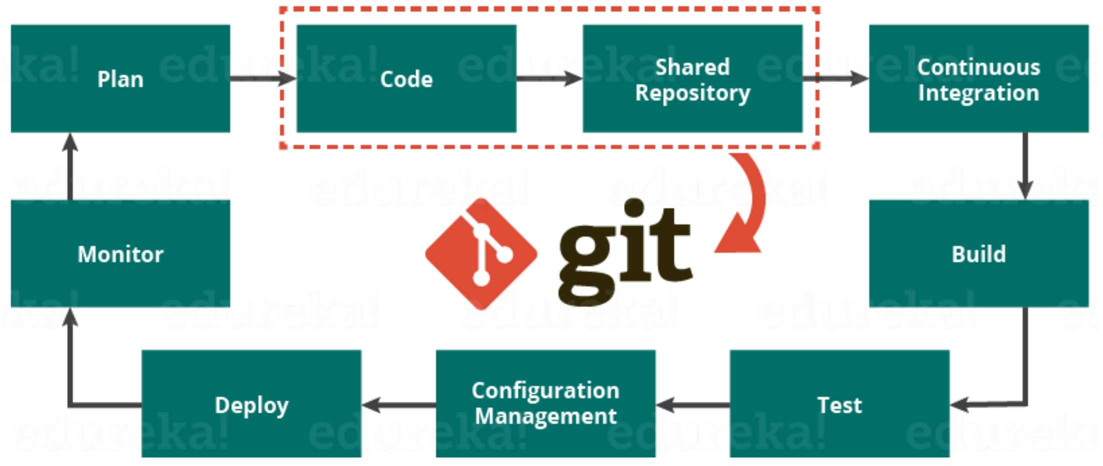
Features of Git:

- Free and open source:
  Git is released under GPL's (General Public License) open source license.
  You don't need to purchase Git. It is absolutely free. And since it is open source, you can modify the source code as per your requirement.
- Speed:
  Since you do not have to connect to any network for performing all operations, it completes all the tasks really fast. Performance tests done by
  Mozilla showed it was an order of magnitude faster than other version control systems. Fetching version history from a locally stored repository can be one hundred times faster than fetching it from the remote server. The core part of Git is written in C, which avoids runtime overheads associated with other high-level languages.
- Scalable:
  Git is very scalable. So, if in future, the number of collaborators increase Git can easily handle this change. Though Git represents an entire repository, the data stored on the client's side is very small as Git compresses all the huge data through a lossless compression technique.
- Reliable:
  Since every contributor has its own local repository, on the events of a system crash, the lost data can be recovered from any of the local repositories. You will always have a backup of all your files.
- Secure:
  Git uses the *SHA1* (Secure Hash Function) to name and identify objects within its repository. Every file and commit are check-summed and retrieved by its checksum at the time of checkout. The Git history is stored in such a way that

the ID of a particular version (a *commit* in Git terms) depends upon the complete development history leading up to that commit. Once it is published, it is not possible to change the old versions without it being noticed.

- Economical:
  In case of CVCS, the central server needs to be powerful enough to serve requests of the entire team. For smaller teams, it is not an issue, but as the team size grows, the hardware limitations of the server can be a performance bottleneck. In case of DVCS, developers don't interact with the server unless they need to push or pull changes. All the heavy lifting happens on the client side, so the server hardware can be very simple indeed.
- Supports non-linear development:
  Git supports rapid branching and merging, and includes specific tools for visualizing and navigating a non-linear development history. A core assumption in Git is that a change will be merged more often than it is written, as it is passed around various reviewers. Branches in Git are very lightweight. A branch in Git is only a reference to a single commit. With its parental commits, the full branch structure can be constructed.
- Easy Branching:
  Branch management with Git is very simple. It takes only few seconds to create, delete, and merge branches. Feature branches provide an isolated environment for every change to your codebase. When a developer wants to start working on something, no matter how big or small, they create a new branch. This ensures that the master branch always contains production-quality code.
- Distributed development:
  Git gives each developer a local copy of the entire development history, and changes are copied from one such repository to another. These changes are imported as additional development branches, and can be merged in the same way as a locally developed branch.

DevOps is the practice of bringing agility to the process of development and operations. It's an entirely new ideology that has swept IT organizations worldwide, boosting project life-cycles and in turn increasing profits. DevOps promotes communication between development engineers and operations, participating together in the entire service life-cycle, from design through the development process to production support.



DevOps is the practice of bringing agility to the process of development and operations. It’s an entirely new ideology that has swept IT organizations worldwide, boosting project life-cycles and in turn increasing profits. DevOps promotes communication between development engineers and operations, participating together in the entire service life-cycle, from design through the development process to production support.

Q5. What is the impact of recent developments in industry on such CASE tools?

Solution:

The growing complexity of the software systems being developed and the use of different methodologies indicate the need for more computer support for automating software development process and evolution activity. Currently, Computer-Aided Software Engineering (CASE), which is a set of software systems aimed to support set of software process activities, does this automation. While CASE tools prove its importance to develop high quality software, unfortunately CASE tools don’t cover all software development activities.

As the software products become large-scale, their maintenance becomes complex, and the growth in the software industry is increased, the demanding for fully automating of software development activities becomes necessary. Poor defined application domains, noisy, changing and conflicting objectives of the developed software are some of the problems' properties of the software development process. These properties would force the software developers to change the development and deployment methods and to get the support of intelligent software development tools for speeding up the development process and decreasing the cost of the development. AI, ML, deep-learning techniques prove to be the best answer to these types of problem, since they based on imitating human intellectual skills. I-CASE tools that based on AI techniques. The ultimate goal of I-CASE is to speed up and facilitate the efforts of the software development. The survey, analysing, and definition of I-CASE tools cover the development activities of requirements engineering, design, coding, testing, documentation, and other processes; highlighting the problems facing the automation of these activities and the required AI technique