Unit 11 Computer Aided Software Engineering (CASE) Tools

Structure:

11.1 Introduction

In the previous unit, we have discussed the importance of quality in software development. In this unit, let's discuss various standard tools available for software development.

Nowadays everything has to go faster. Because of the increasing speed of changing market-demands new products replace old ones much earlier than before, so the development of new products has to go faster. Thus the production lines have to be developed faster, too. A very important role in this development is software engineering because many production processes are 'computer aided', so software has to be designed for this production system. It seems very important to do the software engineering right and fast. In the past, software systems were built using traditional development techniques, which relied on hand-coding applications. Software engineers had to design software without help of computers, by programming each step at one time. This way of developing software is too much costly and time-consuming. In order to speed up the process the bottlenecks in building software systems are to be found. This is hard to do because of the increasing role of computers in our society. Technology is developing further day by day, so that faster and bigger computers enter the scene. The software running on these computers can be more extensive because they can handle more information in the same time, so there is an increasing amount of data to go with it. Finding the right data out of this increasing amount of information is getting harder and harder, so finding the bottleneck is harder to do. To speed up the software system building process, a new concept of designing software is introduced in the '70s, called ***Computer Aided Software Engineering (CASE).*** We will be discussing CASE in detail in this unit.

Objectives:

After studying this unit, you should be able to:

- explain the basic concepts of CASE
- discuss how CASE supports software life cycle
- classify CASE tools
- describe integrated CASE environments
- explain the architecture of CASE environment
- discuss CASE repository

11.2 CASE Concepts

The term CASE is used for a new generation of tools that applies rigorous engineering principles to the development and analysis of software specifications. Simply, computers develop software for other computers in a fast way by using specific tools. When implanted in a concurrent engineering environment, this process is taking place while some parts of the developed software are running already. It's a sort of on-line software engineering. There are a lot of problems with these kinds of systems because they are very complex, not easily maintainable and fragile. Some of the tools which were very progressive back then, are obsolete right now, so they have to be updated, which can be a problem because the software engineering process is fit around these tools.

The tools developed right now are evolutional products out of earlier tools. The first tools, developed in the mid '70s, were mainly introduced to automate the production and to maintain structured diagrams. When this automation covers the complete life-cycle process, we speak of *Integrated Computer Aided Software Engineering (I-CASE).* When only one specific part of the life-cycle process is covered we speak of -just- Computer Aided Software Engineering (CASE).

Later on, integrated CASE (I-CASE) products were introduced. This was an important step because I-CASE products are capable of being used to generate entire applications from design specifications.

Recently, CASE tools have entered a third phase: the introduction of new methodologies based on capabilities of I-CASE tools. These new methodologies utilize Rapid Prototyping techniques to develop applications faster, at lower cost and higher quality. By using Rapid Prototyping a prototype can be made fast, so the developed system can be tested more often in between the development-phases because it doesn't cost much time to create a prototype. Mistakes can be detected and corrected earlier this way. The earlier this can be done, the better because correcting these mistakes gets harder and more expensive when the system is developed further. So a lot of time and money can be saved using Rapid Prototyping.

As said above, a new set of tools is necessary. These tools should automate each phase of the life-cycle process and tie the application development more closely to the strategic operations of the business. A lot of different tools have been developed over the years and are being developed right now. There is so many tools, that we could easily get confused.

The heart of a well-designed I-CASE system is a repository, which is used as a knowledge base to store information about the organization, its structure, enterprise model, functions, procedures, data models etc. The meaning represented by diagrams and their detail windows is stored in the repository. The repository steadily accumulates information relating to the planning, analysis, design, construction and maintenance of systems. In other words: *The repository is the heart of a CASE system.*

Two types of mechanisms have been used in CASE software to store design information:

1) A *dictionary*, which contains names and descriptions of data items, processes, etc.
2) A *repository*, which contains this dictionary information and a complete coded representation of plans, models and designs, with tools for cross-checking,

To survey all these CASE tools we divide them in the following categories:

1) *Information engineering-supporting products.* These are life-cycle processes, derived from the strategic plans of the enterprise and which provide a repository to create and maintain enterprise models, data models and process models.
2) *Structured diagramming-supporting products.* These are derived from several development methodologies like Gane-Sarson or Jackson. These products at least support data flow, control flow and entity flow, which are the three basic structured software engineering diagram types.
3) *Structured development aids-providing products.* These products are providing aids for a structured development of the process. These products are very suitable to be used by the system analysts, because they are helped very much by a structured process, because those can be analyzed faster and more accurately.
4) *Application-code-generating products: These are products that* generate application-code for a specific goal, set by the designer. Most of the products in this area are using a COBOL-generator, which is a tool that generates programming code in a specific language out of specifications set by the system-designer.

Computer Aided Software Engineering (CASE) tools assist software engineering managers and practitioners in every activity associated with the software process. They automate project management activities; manage all work products produced throughout the process, and assist engineers in their analysis, design, coding and testing work.

Computer Aided Software Engineering (**CASE**) is the use of software tools to assist in the development and maintenance of software. Tools used to assist in this way are known as **CASE Tools**

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 that product is built.

Some typical CASE tools are:

- Code generation tools
- Data modeling tools
- UML
- Refactoring tools
- QVT or Model transformation Tools
- Configuration management tools including revision control

All aspects of the software development lifecycle can be supported by software tools, and so the use of tools from across the spectrum can, arguably, be described as CASE from project management software through tools for business and functional analysis, system design, code storage, compilers, translation tools, test software, and so on.

However, it is the tools that are concerned with analysis and design, and with using design information to create parts (or all) of the software product, that are most frequently thought of as CASE tools. Such tools arose out of developments such as Jackson Structured Programming and the software modelling techniques promoted by researchers such as Ed Yourdon, Chris Gane and Trish Sarson. Applying CASE to a database software product, might normally involve:

- Modelling business / real world processes and data flow
- Development of data models in the form of entity-relationship diagrams
- Development of process and function descriptions
- Production of database creation SQL and stored procedures
- Program specifications
- User documentation

A number of risks are inherent whenever we attempt to categorize CASE tools. There is a subtle implication that to create an effective CASE environment, one must implement all categories of tools – this is simply not true. Confusion (or antagonism) can be created by placing a specific tool within one category when others might believe it belongs in another category. Some readers may feel that an entire category has been omitted – thereby eliminating an entire set of tools for inclusion in the overall CASE environment. In addition, simple categorization tends to be flat – that is, we do not show the hierarchical interaction of tools or the relationships among them. But even with these risks, it is necessary to create taxonomy of CASE tools – to better understand the breadth of CASE and to better appreciate where such tools can be applied in the software engineering process.

Self Assessment Questions

1. The term CASE refers to new generation of tools that applies rigorous engineering principles to the development and analysis of software specifications. (True / False)
2. CASE stands for _______________.
3. The _______________ is the heart of a CASE system. (Pick right option)
   a) Repository
   b) Dictionary
   c) Design
   d) Coding

11.3 Classification 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 (hardware and software) that supports them, or even by their origin or cost. The taxonomy presented here uses function as a primary criterion.

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 (the work breakdown structure), 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 continuous basis. In addition, a manager should use tools to collect metrics that will ultimately provide an indication of software product quality.

The following tools are 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 (e.g., LOC (Lines of Code)/person-month, defects per function point) 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. Other tools extract technical metrics and in an effort to project the quality of the software that is being built.

Database Management Tools: 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 major 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 facilitates the auditing process; and CASE communication tools can greatly improve status accounting (reporting information about changes to all who need to know).

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. 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. However, these tool kits are being replaced by interface prototyping tools that enable rapid onscreen creation of sophisticated user interfaces that conform to the interfacing standard that has been adopted for the software.

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.

Self Assessment Questions

4. Business process engineering tools provide a meta-model from which specific information systems are derived. (True / False)
5. _______________ are used to represent the key elements of a process so that it can be better understood.
6. Screen painter is an example for _______________ tools. (Pick right option)
   a) Programming
   b) Prototyping
   c) Design
   d) Analysis

11.4 Steps for CASE Tool Implementation

Now let's see various steps for implementing CASE tools.

1) Conduct a technology-impact study to determine how the basic business of the organization should change to maximize the opportunities presented by rapid technological change.
2) Evaluate how the organization should be re-engineered to take advantage of new technology.
3) Establish a program for replacing the old systems with the most effective new technology.
4) Commit to an overall integrated architecture.
5) Select a development methodology.
6) Select a CASE tool.
7) Establish a culture of reusability.
8) Strive for an environment of open interconnectivity and software portability across the entire enterprise.
9) Establish inter-corporate network links to most trading partners.

10) Determine how to provide all knowledge to workers with a high level of computerized knowledge and processing power.
11) Determine the changes in management-structure required to take full advantage of innovative systems, architectures, methodologies and tools.

11.5 Integrated CASE Environments

An I-CASE environment is a collection of CASE tools and other components together with an integration approach that supports most or all of the interactions that occur among the environment components, and between the users of the environment and the environment itself.

The critical part of the above definition is that the interactions among environment components are supported within the environment. What distinguishes a CASE environment from a random amalgamation of CASE tools is that there is some thing that is provided in the environment that facilitates interaction of those tools. This 'something' may be a physical mechanism such as a shared database or a message broadcast system, a conceptual notion such as a shared philosophy on tool architectures or common semantics about the objects the tools manipulate, or some combination of these things.

The integrated CASE (I-CASE) include (1) smooth transfer of information (models, programs, documents, data) from one tool to another and 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, and document production; (3) an increase in project control. That is achieved through better planning, monitoring, and communication; and (4) Improved coordination among staff members who are working on 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 and each tool, and an effective approach that will enable I-CASE to move among various hardware platforms and operating systems. Comprehensive I-CASE environments have emerged more slowly than originally expected. However, integrated environments do exist and are becoming more powerful as the years pass.

The term *integration* implies both *combination* and *closure*. I-CASE combines a variety of different tools and a spectrum of information in a way that enables closure of communication among tools, between people, and 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 and feel is provided for all tools; a development philosophy is integrated, implying a standardized software engineering approach that applies modern practice and proven methods.

To define integration in the context of the software engineering process, it is necessary to establish a set of requirements for I-CASE: An integrated CASE environment should provide the following,

- Provide a mechanism for sharing software engineering information among all tools contained in the environment.
- Enable a change to one item of information to be tracked to other related information items.
- Provide version control and overall configuration management for all software engineering information.
- Allow direct, non-sequential access to any tool contained in the environment.
- Establish automated support for the software process model that has been chosen, integrating CASE tools and software configuration items (SCIs) into a standard work breakdown structure.
- Enable the users of each tool to experience a consistent look and feel at the human/computer interface.
- Support communication among software engineers.
- Collect both management and technical metrics that can be used to improve the process and the product.

The range of possible ways of providing the `glue' that links CASE tools together inevitably leads to a spectrum of approaches to implementing a CASE environment. There are many ways to build a CASE environment. While many people concentrate on the selection of CASE tools and components when assembling a CASE environment, they largely ignore the need to support the interactions among those components. We concentrate less on which components should be chosen, and much more on how the selected components can be made to work together effectively. Whether a chosen approach to component interaction is appropriate in a given context will depend on many overlapping factors: the needs of the organization in question, the available resources, and so forth. A detailed assessment of these related factors and constraints is necessary to determine the CASE environment most suited to the problem at hand.

Self Assessment Questions

7. The term integration implies both combination and closure. (True / False)
8. SCI stands for _______________.

11.6 Architecture of CASE Environment

Architecture refers to the way a system is designed and how the components are connected with each other. There are computer architectures, network architectures and software architectures. IT architecture is a design for the arrangement and interoperation of technical components that together provide an organization its information and communication infrastructure.

Architecture is the basis for building any software or hardware systems. Here we are going to discuss the ICASE architecture. A software engineering team uses CASE tools, corresponding methods, and a process framework to create a pool of software engineering information. The integration framework facilitates transfer of information into and out of the pool. To accomplish this, the following architectural components must exist: a database must be created (to store the information); an object management system must be built (to manage changes to the information); a tools control mechanism must be constructed (to coordinate the use of CASE tools); a user interface must provide a consistent pathway between actions made by the user and the tools contained in the environment. Most models of the integration framework represent these components as layers. A simple model of the framework, depicting only the components just noted is shown in figure 11.1.

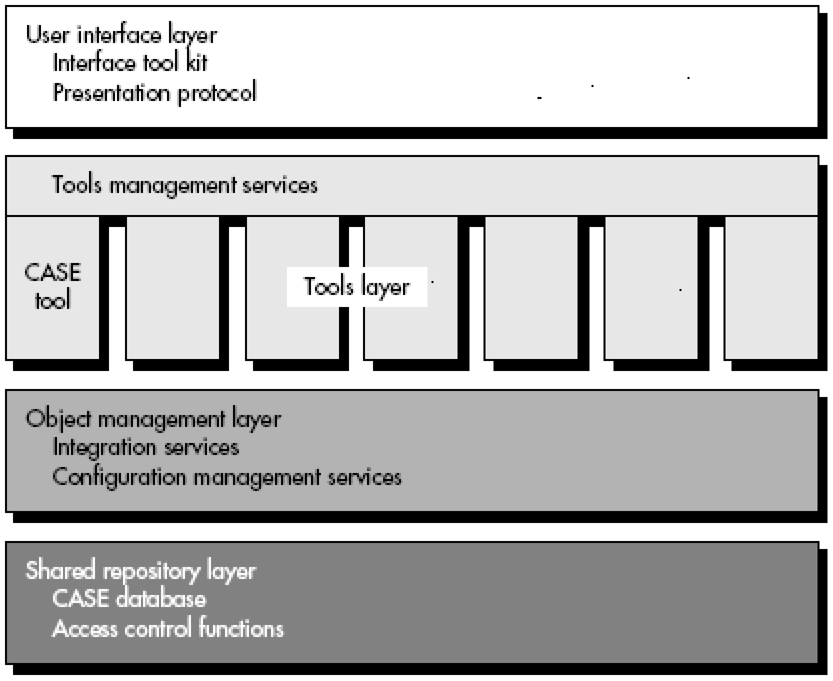

Fig. 11.1: Integrated CASE environment

The *user interface layer* (Figure 11.1) incorporates a standardized interface tool kit with a common presentation protocol. The interface tool kit contains software for human/computer interface management and a library of display objects. Both provide a consistent mechanism for communication between the interface and individual CASE tools. The *presentation protocol* is the set of guidelines that gives all CASE tools the same look and feel. Screen layout conventions, menu names and organization, icons, object names, the use of the keyboard and mouse, and the mechanism for tools access are all defined as part of the presentation protocol.

The *tools layer* incorporates a set of tools management services with the CASE tools themselves. *Tools management services* (TMS) control the behavior of tools within the environment. If multitasking is used during the execution of one or more tools, TMS performs multitask synchronization and communication, coordinates the flow of information from the repository and object management system into the tools, accomplishes security and auditing functions, and collects metrics on tool usage. In essence, software in this layer of the framework architecture provides the mechanism for tools integration. Every CASE tool is "plugged into" the object management layer. Working in conjunction with the CASE repository, the Object Management Layer (OML) provides integration services – a set of standard modules that couple tools with the repository. In addition, the OML provides configuration management services by enabling the identification of all configuration objects, performing version control, and providing support for change control, audits, and status accounting. The *shared repository layer* is the CASE database and the access control functions that enable the object management layer to interact with the database.

CASE Repository

CASE Repository is a database that acts as the center for both accumulation and storage of software engineering information. The role of the person (the software engineer) is to interact with the repository using CASE tools that are integrated with it. The repository is a relational or object-oriented database that is "the center of accumulation and storage" for software engineering.

Self Assessment Questions

9. _______________ control the behavior of tools within the environment.
10. _______________ is a database that acts as the center for both accumulation and storage of software engineering information.

11.7 Summary

Let's recapitulate the important points discussed in this unit.

- Computer-aided software engineering tools span every activity in the software process and those umbrella activities that are applied throughout the process.
- CASE combines a set of building blocks that begins at the hardware and operating system software level and end with individual tools.
- CASE categories encompass both management and technical activities that span most software application areas. Each category of tool is considered a "point solution."
- The I-CASE environment combines integration mechanisms for data, tools, and human/computer interaction.

- Data integration can be achieved through direct exchange of information, through common file structures, by data sharing or interoperability, or through the use of a full I-CASE repository.
- Tools integration can be custom designed by vendors who work together or achieved through management software provided as part of the repository. Human/computer integration is achieved through interface standards that have become commonplace throughout the industry. Integration architecture is designed to facilitate the integration of users with tools, tools with tools, tools with data, and data with data.
- The CASE repository has been referred to as a "software bus." Information moves through it, passing from tool to tool as software engineering progresses. But the repository is much more than a "bus." It is also a storage place that combines sophisticated mechanisms for integrating CASE tools and thereby improving the process through which software is developed. The repository is a relational or object-oriented database that is "the center of accumulation and storage" for software engineering.

11.8 Terminal Questions

1. Explain the classifications of CASE tools.
2. List out the necessary steps for implementing CASE tools.
3. What do you mean by I-CASE? Explain.
4. Give the architecture of CASE environment.

11.9 Answers

Self-Assessment Questions

1. True
2. Computer Aided Software Engineering
3. a) Repository
4. True
5. Process modeling tools
6. b) Prototyping
7. True
8. Software Configuration Items
9. Tools management services
10. CASE Repository

Terminal Questions

1. Business Process Engineering Tools, Process Modeling and Management Tools, Project Planning Tools, Risk Analysis Tools, Project Management Tools, etc. (Refer Section 11.3 for detail)
2. Conduct a technology-impact study to determine how the basic business of the organization should change to maximize the opportunities presented by rapid technological change. Evaluate how the organization should be re-engineered to take advantage of new technology. (Refer Section 11.4)
3. An I-CASE environment is a collection of CASE tools and other components together with an integration approach that supports most or all of the interactions that occur among the environment components, and between the users of the environment and the environment itself. (Refer Section 11.5)
4. A software engineering team uses CASE tools, corresponding methods, and a process framework to create a pool of software engineering information. The integration framework facilitates transfer of information into and out of the pool. (Refer Section 11.6)