CASE

Computer Aided Software Engineering – sometimes called computer aided systems engineering - is the automation of step-by-step methodologies for software and systems development to reduce the amount of repetitive work the developer needs to do. Its adoption can free the developer for more creative problem solving tasks. CASE tools also facilitate the creation of clear documentation and the coordination of team development efforts. Team members can share their work easily by accessing each other's files to review or modify what has been done. Some studies have found that systems developed with CASE and the newer methodologies are more reliable, and they require repairs less often (Dekleva, 1992). Modest productivity benefits can be achieved if the tools are used correctly. Many CASE tools are PC-based with powerful graphical capabilities.

CASE tools provide automated graphics facilities for producing charts and diagrams, screen and report generators, data dictionaries, extensive reporting facilities, analysis and checking tools, code generators and document generators. Most CASE tools are based on one or more of the popular structured methodologies. Some support object oriented development. In general CASE tools try to increase productivity and quality by doing the following

- Enforce a standard development methodology and design discipline.
- Improve communication between users and technical specialists.
- Organise and correlate design components and provide rapid access to them via a design repository.
- Automate tedious and error prone portions of analysis and design.
- Automate code generation, testing and control rollout.

Many CASE tools have been classified in terms of whether they support activities at the front end or the back end of the systems development process. Front-end CASE tools focus on capturing analysis and design information in the early stages of systems development, whereas back end CASE tools address coding, testing and maintenance activities. Back end tools help convert specifications automatically into program code.

CASE tools automatically tie data elements to the process where they are used. If a data flow diagram is changed from one process to another, the elements in the data dictionary would be altered automatically to reflect the change in the diagram. CASE tools thus support iterative design by automating revisions and changes and providing prototyping facilities.

A CASE information repository stores all the information defined by the analysts during the project. The repository includes data flow diagrams, structure charts, entity relationship diagrams, data definitions, process specifications, screen and report formats, notes and comments and test results. CASE tools now have features to support client-server applications; object oriented programming and business process redesign. Methodologies and tools sets are being created to leverage organisational knowledge of business process reengineering (Nissen, 1998).

To be used effectively CASE tools require organisational discipline. Every member of a development project must adhere to a common set of naming conventions and

standards as well as a development methodology. The best CASE tools enforce common methods and standards, which may discourage their use in situations where organisational discipline is lacking.

Select SSADM

When To Use Select OMT (Object Modelling Technique) and the *Select Perspective* systems development method.

The **File** menu is for creating and updating diagrams; you can use it to change projects.
The **Project** menu is used for project maintenance, off-lining, restoring, copying, backup and so on.
The **Dictionary** menu is for populating the dictionary or for generating reports.
The **Tools** menu gives you access to the add-on tools. There is a document generator.
The **Maintenance** menu can be used for unlocking files.
The **Help** menu uses the standard Windows help format.

What is Project?
At this stage you will need to understand what *Select* means by a **project**. A project is a collection of diagrams and associated flies, which is given a name and stored together in a directory. You have to have a project loaded to open the diagrams it contains. *Select* stores each diagram in a separate file with a .DAT extension, and it groups the diagrams that relate to a project in a project directory. Along with the diagram files there are some hidden files.

So creating a project is equivalent to creating a new directory and starting a new diagram is equivalent to opening a new file. Each time a diagram is altered the alterations are automatically saved, so you don't have to remember to save your diagrams, however if you need to keep an earlier version, you will need to use the Version option from Project on the menu bar. As with all software, you should quit in the proper way by choosing File then Exit.

To see what projects are available to you choose File from the menu bar and then Open. You will then get a screen that shows you what project is currently loaded. It also shows you what files or diagrams there are for that project. You can change the project by hitting the "***Project...***" button. A list box tells you what project *Select* knows about and the "*Switch to*" button allows you to select one.

Managing the Project
A word of warning. *Select* maintains this catalogue of the projects it knows about on the C: drive of your current machine. If you have off-lined your project to the A: drive or if you work from the H: drive and return to your project using another machine, or come back using the same machine but on another day after its hard drive has been cleaned then *Select* will have forgotten about your project and it will not appear in this list box. Don't despair all is not lost. There are at least three ways to work: -

Work on the C: Drive

This is the easy way. You open projects or create new ones, and *Select* updates and saves them for you. The catalogue it maintains is always accurate. However you can only do this if you have your own machine and you are the only one using it. You will have to handle the Backup of your C: drive yourself.

Work on the C: Drive and Off-line to A: Drive

Here you work on the C: drive but when you have finished your session you must allow a little time to Off-line or move your files onto a floppy in the A: drive before you leave the machine. Don't move the files with the File Manager or using DOS, but whilst still in *Select*, close all files, then from the Project Menu select Offline. You will need to use the Restore option then next time you want to work on this project. If you want to make a backup of your project you can do so using the File Manager but remember to copy the whole directory. Do not copy the contents file by file; *Select* has some hidden files in there.

Work on the H: Drive

If you have your own personal space on the H: drive, or a space reserved for a Group project then there are advantages to working on the H: drive. Backups are handled automatically by the technical staff and you don't have to carry floppy disk around with you. However when you come back to a project using a different machine *Select* will not know about it so go through the procedure (below) for creating a new project but specify the same name and directory as you used previously, you will find all the diagrams there intact.

Creating a new project

From the top-level menu select Project and not File. From this Project menu select New. Into the blank Project Details screen enter the details of your project. Press Create on this screen and tell *Select* to create the directory.

Before adding too much vital information to your project, it is always worth running through your chosen saving, quitting and reloading routines. To make sure that you have understood how to manage your project, follow one of the procedures are outlined above.

Restoring a project after a crash

If the machine you are using crashes or *Select* crashes (I've not known this to happen) or you leave *Select* in a hurry and forget to Off-line your project, then you can usually get the work back especially if you can see the files with the file manager.

Don't try and Restore the project. This may end up doing just what you don't want i.e. deleting the files from the disk.

Instead go into *Select* and choose Project followed by New. Now put in the title and the directory where the files are to be found. It doesn't matter if you get the title wrong, but it is obviously important to get the right directory.

Select will create a new project, that means it will put a new project into the list of projects it knows about, but it will use the files that are already there. It won't overwrite those files.

When you have opened Select SSADM the first thing to do is to create a project folder. This is where all your diagrams for the project will be kept. It is also the repository for all the data about the diagrams - the metadata - and every object on them.

Select Project\New from the menu bar. The Project Details dialogue box will pop up.

Click the Browse button to open the Directory Selector dialogue box. Specify the h:\select\system directory and click OK. In the Project Details dialogue box give your project a title - no more than eight characters long - and add this to the Directory path. For example you might give your project the title "xercises" and the Directory path will read h:\Select\SYSTEM\xercises

Click Create. This will open a dialogue box asking you if you want to create the directory

Click Yes. A message will pop up to tell you that the project directory has been Created

Creating a new diagram

To create a new diagram on SSADM Select go through the following steps -

- Select File\New from the menu
  The Create New Diagram dialogue box will pop up. The box will show the project you have been working on most recently. If you have more than one project folder in your directory you can use the Project. button to specify the one you want to work with
- The dialogue box shows a list of all the SSADM models that Select supports. Select the Type of diagram you want to create from the list box. The tool gives you a unique default file ID (ggd00001), which you can change if you wish. This is a identification key which uniquely identifies the diagram in the project data dictionary (you'll learning all about keys and their uses in your systems analysis modules)
- Click the Create button
  A window will open dedicated to creating the type of diagram that you have Chosen

Developing the model

The window you have opened is a toolbox for creating whatever kind of model you have selected. It will enable you to use a notation and selection of objects that are specific to that model. For example, if you had chosen to create a DFD you would be able to draw process boxes, data stores, external entities and data flows.

Programming Languages

The **First Generation** of computer languages consisted of **machine language** – required programmer to write all program instructions in the 0s and 1s of binary code and to specify storage locations for every instruction and item of data used. It was very slow and labour intensive. As computer hardware improved and processing

speed and memory size increased, programming languages became progressively easier for people to understand and use. From 1950s to mid 70s, high-level programming languages emerged, allowing programs to be written with regular words using sentence like statements.

Assembly language

Second-generation language, it is designed for a specific machine and specific microprocessors. It makes use of certain mnemonics (e.g. load, sum) to represent machine language instructions and storage instructions. It gives programmers great control but it is difficult and costly to write and learn. It is used today primarily in system software.

Third generation languages

Specify instructions as brief statements that are more like natural languages than assembly languages. All are less efficient in the use of computer resources than earlier languages, they are easier to write and understand and have made it possible to create software for business and scientific problems

- FORTRAN
  FORmula TRANslator was developed in 1956 to provide an easy way of writing scientific and engineering applications. It is especially useful in processing numeric data.
- COBOL
  COmmon Business Orientated Language was developed in the 1960s by a committee representing both government and industry. Rear Admiral Grace M. Hopper was a key committee member who played a major role in COBOL development. It was designed with business administration in mind, for processing large data files with alphanumeric characters and for performing repetitive tasks such as payroll. It is poor at complex mathematical calculations. Also there are many versions of COBOL and not all are compatible with each other.
- BASIC and Pascal
  Used primarily in education to teach programming. BASIC (Beginners All purpose Symbolic Instruction Code) was developed in 1964 by John Kemeny and Thomas Kurtz to teach students at Dartmouth College how to use computers. It is easy to use but does few computer-processing tasks well even though it does them all. Different versions of BASIC exist.
  Named after Blaise Pascal, 17^{th} Century mathematician and philosopher. It was developed by Swiss computer scientist Niklaus Wirth in the late 60s. Primarily used in Computer Science courses to teach sound programming practices.
- C and C++
  C is a powerful and efficient language developed at AT&T's Bell labs in the early 70s. It combines machine portability with tight control and efficient use of computer resources, and it can work on a variety of different computers. It is used primarily by professional programmers to create operating systems and application software especially in PCs.

C++ is a newer version of C that is object oriented. It has all the capabilities of C plus additional features for working with software objects. C++ is used for the developing application software.

4GL

Fourth Generation Languages consist of a variety of software tools that enable end-users to develop software applications with minimal or no technical assistance or that enhance professional programmer's productivity. Fourth generation languages tend to be nonprocedural or less procedural than conventional programming languages. Procedural languages require specification of the sequence of steps, or procedures, that tell the computer what to do and how to do it. Nonprocedural languages only specify what has to be accomplished rather than provide details about how to carry out the task. Some of these nonprocedural languages are **natural languages** that enable users to communicate with the computer using conversational commands resembling human speech. There are 7 categories of 4GLs, listed in terms of ease of use by nonprogramming end users.

- PC software tools
  General-purpose application software packages for PCs
  e.g. WordPerfect, Internet Explorer, Access
- Query languages
  Languages for retrieving data stored in databases or files. Capable of supporting requests for information that are not predefined.
  e.g. SQL
- Report generators
  Extract data from files or databases to create customised reports in a wide range of formats not routinely produced by an information system. Generally provide more control over the way data are formatted, organised and displayed than query languages.
  e.g. RPG III
- Graphics languages
  Retrieve data from files or databases and display them in graphic format. Some graphics software can perform arithmetic or logical operations on data as well.
  e.g. SAS graph, Systat.
- Application generators
  Contain pre-programmed modules that can generate entire applications, including Web sites, greatly speeding development. A user can specify what needs to be done, and the application generator will create the appropriate program code for input, validation, update, processing and reporting.
  e.g. FOCUS, PowerBuilder, Microsoft FrontPage.
- Application software packages
  Software programs sold or leased by commercial vendors that eliminate the need for custom written, in house software.
  e.g. PeopleSoft HRMS, SAP R/3
- Very high level programming languages.
  Generate program code with fewer instructions that conventional languages, such as COBOL or FORTRAN. Designed primarily as productivity tools for professional programmers.
  e.g. APL, Nomad2.

End users are most likely to work with PC software tools and query languages. **Query languages** are software tools that provide immediate on-line answers to requests for information that are not predefined such as "Who are the highest performing sales representatives". Query languages are often tied to data management software and to database management systems.