Journal of Information Systems Education
Volume 4, Number 4

INCORPORATING CASE IN THE SYSTEMS ANALYSIS AND DESIGN COURSE

Dr. T. M. Rajkumar
Department of Decision Sciences, SBA
Miami University
Oxford, OH 45056

ABSTRACT: This paper describes CASE tools and argues for their incorporation in the class room in all the phases of the system development life cycle. The paper then describes trends that are taking place in the software engineering field. The use of CASE in teaching and supporting these new concepts is described.

KEYWORDS: CASE, Systems Analysis and Design, Cooperative Processing, Reengineering, Object Oriented Analysis

INTRODUCTION

Computer Aided Software Engineering (CASE) has been around since the early 1970's but has only emerged recently as a factor in the MIS environment. A study conducted in 1988 showed that less than 10% of the industry was actively using CASE. In contrast, by 1991 30% of the industry have adopted CASE and another 54% are studying or planning use. This is largely due to two factors. First, the costs of CASE products have dropped considerably along with other software and hardware products. Second, CASE has begun to gain momentum as more users are demanding CASE products resulting in improved vendor support.

CASE tools have been gradually introduced in the systems analysis and design courses for MIS students in the past two to three ears. The availability of free software for academic use from InterSolv has helped accelerate this trend. The popular books that are being used in the classes now include CASE in their discussions and treatment of material.

Most instructors or books do not provide a comprehensive treatment of CASE in their class rooms. This paper tries to develop an integrated approach in which CASE forms the underlying theme of the course. The paper introduces the concept of CASE in section 1. Section 2 discusses the issues and topics to be covered in the classes. Section 3 touches upon emerging issues that need to be addressed in the years to come.

CASE

CASE is a mechanism for automating structured methodologies , and an environment that supports the software engineering process . The environment is usually comprised of seven distinct components that are listed as follows: diagramming tools, syntax verifiers, prototyping tools, reengineering tools, central repository, code generators, and project management and methodology support . The diagramming tools allow analysts and users alike to quickly grasp the flow of data through the graphical representation of the processes. The automated aspect of the tools allows for great flexibility without extensive time spent redrawing the diagrams. The syntax verifier has the capacity to insure that inputs and outputs of processes are balanced. The central repository is the foundation of the CASE environment.

This repository acts as the bridge between the graphics of Data Flow Diagrams (DFD), report or screen definitions and the data definition and the generation of various code depending on the circumstances. The prototyping tools are used to develop both report outlines and screen outlines quickly and easily. These reports and screen outlines can be demonstrated to the users and modified continually until all parties are satisfied. Any of the fields used in the input or output of the screen or reports can be obtained from the central repository.

The code generators allow for specific modularized code to be generated by the analyst. The project management and methodology support tools are used to track the progress and manage the resources used for that progress. Finally the reengineering tools can take code and give it some structure or create structure charts for systems that already exist.

In this article, CASE is interpreted in a broad sense, encompassing a range of activities that support the entire systems life cycle. This interpretation is consistent with the emerging view of CASE that emphasizes attention on the entire systems environment . The approach taken here is not to teach the students just new technology but to help them use technology to learn analysis and design better.

TOPICS AND ISSUES

The topics and issues addressed here are focused on introducing the student to CASE as a broadly useful tool in the many facets of systems analysis and development. Hence, this section discusses the various features of case tools and how they can help in system analysis, design and development.

The benefits that are achieved with CASE tools include:

- increased productivity
- better quality
- reduced maintenance costs

These benefits are not always achieved unless the CASE tools are implemented correctly, and managed carefully. To successfully implement CASE tools the business must use a structured systems methodology [5]. The CASE tool is used to facilitate the chosen business methodology as opposed to replacing it. In order for a student to successfully use CASE in the work place the student needs to understand the usage of CASE in the system development life cycle.

All too often in a systems analysis and design course, CASE is introduced in the logical analysis and design phases. The student is exposed and shown how to draw DFD's, construct data dictionaries, draw structured charts etc., without showing how it can affect the rest of the life cycle.

The areas CASE can help in should be clearly specified to the student. They include the following:

Project Estimation and Planning

Developers and planners have expressed concern over their inability to accurately predict costs and estimate efforts , human resource requirements, and duration of projects. These are critical to ensure success and to reach go/no go decisions at various phases of the life cycle. CASE can help in project estimation and planning, and could help control the development costs of the project. The teaching approach used here is to have the student research how a company uses CASE and estimating tools such as Excelerator and Estimacs together in their planning and project estimation.

Project Management

Project management tools are sometimes part of CASE tools. At other times they can be easily integrated into the CASE tool (for example, Microsoft Project can be linked to Excelerator). The project management tools can be used to provide 1) detailed work plans, 2) task dependency relationships 3) tracking of the project and 4) help in allocating chargeback to the various projects. The teaching approach used here is to expose the students to the detailed work plans and drawing of GANTT and PERT charts using the CASE tool. Emphasis on the rest of the project management tools to help track the project as the student progresses through the semester in the project is also shown.

The approach taken here is not to teach the students just new technology but to help them use technology to learn analysis and design better.

Analysis and Specification

Most structured analysis and design course place a strong emphasis on this phase as they rightly should. Among the items that the students should be taught are 1) Requirements specification, 2) Data dictionary, 3) Report Design, 4) Screen Design, and 5) Entity-Relationship diagrams. Often the tools are taught in isolation and consistency checks among the various tools using the design repository are not shown. It is important to show to the student that the tools are integrated and work in unison to achieve the requirement specification.

In addition, it must be shown that other software tools such as word processing tools can be integrated with the CASE tools to perform the requirement specification, and any issue resolutions that occur between the user and the analyst. If the issue resolution results in a new specification, CASE tools can store both the old specification, and the new specification. The ease of storing and retrieving multiple versions must be demonstrated to the student with appropriate projects.

One method of teaching analysis and specification with CASE is to divide a class into teams that take turns serving as clients for each other. Student teams then criticize each other's DFD or analysis specifications. The criticism reported is formalized and written using the CASE tool. Issue resolution is then addressed formally by the analyst team. This type of student criticism has worked well. Alternately the instructor can play the role of the client and issue resolution using a CASE tool can be demonstrated to the student.

Design

The design function of normalization is generally taught in a database design course, and is reinforced during the systems analysis course. Rarely is it shown how normalization can be performed with CASE tools. Some CASE tools such as KnowledgeWare can perform normalization from E-R diagrams. Others such as Excelerator provide only consistency checking and inform the user of problems with the data model developed. In either case the student should be exposed to this concept. Similarly structure chart conversion using Yourdon's techniques of data transformation analysis and transaction analysis are taught.

Support for these functions within CASE tools is not emphasized; structure charts are invariably drawn from scratch.

Coding

The student should be exposed to how the automated coding generated by the CASE tools from screen and report generators look. They should be taught how to include such topics within the specifications of the logic. The word processing component of the CASE tool is helpful in going through this logic and identifying the program specifications of the system. The power of CASE tools is enhanced by their ability to work with language compilers such as COBOL. For example, MicroFocus workbench can take Excelerator's output and translate it into skeletal programs including procedure division code. This code can be later enhanced into compilable code.

Evaluation of the structure chart for coupling, cohesion, fan-in and fan-out etc., can be automated to some extent. Reports on such evaluation and what they imply should be made clear to the student by showing them some examples of CASE generated reports. This will help evaluate the design, and syntax check the program logical flow. Tools are currently being developed which provide context sensitive help to the user on the language and provide examples; allowing them to be cut and pasted into the logical code. Just as version control is useful in the requirement process, it is also useful in the coding process, so that programmers and analyst can go back to a previous version if they are unable to a debug an error in the current version.

Testing

A CASE tool can help in this aspect of the life cycle by enabling the analyst to develop and write test specifications, using the requirements specified earlier. CASE tools can also help generate test cases for the code they generate.

CASE tools can also help run statistics, on the data gathered from the delivery platform on actual test runs. It can provide regressions to find causes and effects of certain parameters that are tweaked, or just show simple comparisons between data that is gathered. Both these uses of CASE should be shown to the student.

Support

A great benefit of CASE tools is that they make up a new form of corporate memory that survives staff changes and the limits of paper documentation. CASE tools can help in documenting the problem management in the maintenance phase of the life cycle. It can help in documenting the situation, the action taken, the effects and response on the system. It can also help in change management by identifying the modules which will be affected by changes to the data structure or data element. Though not a primary purpose, CASE tools can help management to monitor the availability, response times, and responsiveness of the support team in mission critical applications.

Communication

CASE tools can help communication among various analysts on the team regarding the project documents through the repository. In addition, it is easy to add additional software that can be called to set up meeting schedules on CASE systems running on local area networks. Many CASE vendors are working on integrating these communication aspects into their systems [9].

As can be seen, CASE permeates every aspect of the system development life cycle. It is important for the student to understand the role CASE plays through out and how it can be used to improve the productivity of the analysts without spending much money.

EMERGING ISSUES

Some current trends are affecting how CASE is going to be used in the future. An exposure to these concepts and how they affect CASE tools and systems development is vital to the students learning process. This section discusses some trends that include among others i) object-oriented analysis, ii) reengineering, iii) cooperative processing and iv) software components.

Object Oriented Analysis

With the advent of object-oriented programming and object-oriented databases, object-oriented analysis (OOA) has taken on an important role in the systems analysis course. Courses still emphasize the traditional DFD, data dictionary and structured analysis concepts. Rarely do they teach object-orientation through out the entire course.

At the core of object-oriented analysis is the object. An object is an entity defined by a set of common attributes and the services or operations associated with it. Objects are the main actors, agents and servers in the system and OOA is concerned with identifying the objects in the problem domain. In OOA the objects are organized in a non hierarchic manner, interact and communicate with other objects based on the interfaces that are specified. Hence, in OOA both objects and their interfaces are of primary interest.

Processing takes place inside the objects and the methods for the various messages that are used to interact are encapsulated inside the object. This encapsulation provides it maintainability. The objects can inherit methods from other objects belonging to the same hierarchy. This enables objects to share common attributes as needed.

CASE tools can be used for support in this approach to identify the objects, their attributes, and class structure. The behavior of the objects is then described and documented within the object. The informational view of the system can be presented using an object notation introduced by Yourdon and Coad. From this perspective one can identify the data objects within the system and how they are affected by other objects. Presentation of object-oriented analysis can be performed using the processing oriented approaches specified by using object-oriented data flow diagrams (ODFD).

ODFDs are similar to conventional DFDs in which objects and functions appear as processes. An object appearing on the ODFD encapsulates related data elements and functions [12]. The behavioral views of the system can be specified using state transition diagrams. The state transition diagrams are used to tie the events that occur on the ODFD to a behavior. Example ODFD's are developed in [11,12].

The approach used in teaching this method involves having the student analyze and design a data entry form using the object oriented approach. This forces the student to think about the functions, methods, inheritance and hierarchical structure of the objects. The drawing of sample ODFD's, and state transition diagrams for getting input, validation and updating of the underlying object for this exercise are also required. It is important for the student to be able to perform object oriented analysis as OOA is to the 90's what structured analysis was to the 80's.

Object oriented analysis also has repercussions for design because a well specified analysis results in normalized objects. Also traditional evaluation techniques such as coupling, cohesion etc. gets modified in the object oriented environment. Metrics such as those proposed by Chidamber and Kemerer has to be taught to the students. This includes weighted methods per class, depth of inheritance tree, number of children, coupling between objects and lack of cohesion in methods. CASE tools do not support much of this activity at this time. Support, however, is forthcoming.

Reengineering

Many systems analysis and design courses treat the subject of system development purely from the view of a new system. Rarely is importance given to the fact that 50% of a traditional systems analyst job is really maintenance and keeping old systems alive . With the trend toward downsizing and converting applications to relational database systems, reengineering has taken on a new and important role in modern information systems. The student should be exposed to this concept in the systems analysis course.

Reengineering is the process by which code written for applications are revised through a CASE tool as the original code may lack structure and maintainability. CASE tools are useful and helpful in this regard. They provide tools such as documenters, analyzers, restructurers and diagrammers to aid in understanding the physical and logical design from the code itself. Documenters read the program code and provide high level information about what the system is doing such as cross referencing information. Restructurers change the unstructured code into something more structured. Analyzers evaluate the strength and weaknesses of the system. Diagrammers such as KnowledgeWare's IEW can read database code and graphically represent it as an IMS hierarchy diagram.

It is important to point out to the student that most of these CASE tools aid only in the physical development of the actual system. CASE systems currently do not support planning and analysis for reengineering. For example, most CASE systems cannot derive a DFD of the application from the program code.

With the advent of object-oriented programming and object-oriented databases, object-oriented analysis (OOA) has taken on an important role in the systems analysis course.

To use these CASE tools with this process the designers must download the information that defined the mainframe database to the PC workstation. The downloaded information typically includes aspects such as database definitions and file descriptions [9]. The database definitions are automatically reverse engineered into graphical formats supported by the CASE tools. CASE systems can produce traces and object mappings from the original definitions to the new system. This enables the analyst to understand and approve the automatic transformations.

The analyst uses the CASE tools to analyze the system and then forward engineer the new system. Forward engineering enables the system and database definitions to be redesigned completely on the PC and uploaded to the mainframe environment. Dependent applications may now need to be rewritten or new applications written to exploit the new database design. It is also important to include a good methodology in the forward engineering process, to delineate and identify the scope of the project.

To introduce these concepts to the student, projects done in the previous year by other students are given without any documentation. The students are just given a working system and the code for the systems. Improvements and incremental modifications of the system are required. The students are then required to use the CASE tool to develop the diagrams, analyze the system, generate documentation, and redevelop the system. The weakness with this approach has been the fact that the particular CASE system employed does not directly convert the code developed in either FOCUS or dBASE that has been used in previous years for the projects in this course. This project however gives the students a strong sense of the maintenance and reengineering functions of systems analyst.

Cooperative Processing

A major trend in information systems today is client server computing and cooperative processing. Cooperative processing occurs whenever a transaction requires two or more processes to execute independently of each other . The typical configuration has an intelligent workstation, connected to a LAN and a mainframe. What makes the processing cooperative is that significant parts of the application execute on the workstation.

Cooperative processing is also the backbone of IBM'S AD/Cycle strategy and is a major component in its AD/cycle architecture . A requirement of AD/cycle architecture is to merge the workstation characteristics of interactive graphical presentation with the ability of mainframe facilities to centrally control and share development resources. Known as cooperative processing this feature tries to exploit the full power of the different platforms and takes advantage of the inherent strength of each computer.

Cooperative processing introduces new concepts for systems development that students must understand. It has a major impact on methodologies in three main areas that include graphical interfaces, packaging of code and data, and the use of object-oriented analysis and design for the client server protocol design [16]. When developing a cooperative application all processing locations within the network will have to be defined, including user privileges and security policies. Real-time communication and data translation services for different platforms must be designed and plans for network version control and software distribution must be created.

CASE tools are emerging to support application development using cooperative processing. What allows cooperative processing to be developed on CASE tools is that the two cooperating processes need not be on two separate processors [16]. A major attraction of a good cooperative architecture is that the system can be coded, developed, and tested entirely on the workstation. Only when the system is ready to be implemented is it ported to the server. It is essential for the student to understand the development of these types of application as they will be the dominant type of application in the future.

To introduce these concepts to the students, students are required to develop reports in the PC environment for an application running on the mainframe. SQL/Windows from Gupta Technologies was used as the software providing the reports for an application developed using FOCUS in the mainframe. The approach used was not a typical client server approach as the mainframe was not connected into the business schools' local area network. The developed system used a snapshot of the data by downloading the information from the mainframe. Reports were then run on this downloaded data.

The real time nature of demand for data and the need for designing the communication aspects of the system was well demonstrated with this approach as the response from the mainframe was sluggish during busy hours. The analysis and design aspects can be introduced to the students by allowing them privileges of Project Manager on the CASE tool. With this privilege, the student can decide which users can get update and retrieval access and learn how to define these in the analysis phase. Students can then use the prototype screens that are developed on the CASE tool, to show the access limitations for various users.

Software Components

Reusable software and the software component industry try to identify software similar to that of a hardware component, so that software systems can be developed with a plug and work approach. Typically based on the 3C model; the component approach distinguishes between three main ideas.

Concept: a statement of what a piece of software does, eliminating how it does it. It is the specification of an abstract behavior. This notion is similar to that of object encapsulation and information hiding.

Content: a statement of how the piece of software achieves the behavior given its concepts. This notion is related to methods in object oriented design. There can be more than one method to the same concept.

Context: aspects of the software environment that are external to the concept but are relevant to the definition of the software. This includes items such as additional information and concepts needed to write the implementation. This is an important distinction from object oriented concepts since the context is separated from the content per se.

The reuse of program material is fundamental to improving the productivity of programmers. The importance of reuse has been promoted with object-oriented environments, that support inheritance, and generic procedures through the class concept. They have proved so popular that commercial vendors are now providing base classes beyond those available in the programming language. Reuse librarians retrieve software components for possible reuse. These systems usually comprise a library of components and query retrieval mechanisms. The reuse of preplanned units is only the tip of the iceberg; since greater benefits can be achieved by reworking units to fit other similar needs.

Systems such as the one developed for CONTEL exist that provide such software reuse. The approach in this system has been tc develop a generic design that can be instantiated in different environments. This reuse environment supports both top down and bottom up developments. CASE tools can help the librarian visualize and understand the abstraction and specialization process during class definition and classification. Perhaps they can aid in high level structuring and identifying and directing the assembly of reusable components.

Students taking the systems analysis courses should be exposed to the concepts of reusability and how it affects the development process. The use of CASE tools for reusable libraries should be identified and emphasized to the students. An effective approach is to have students research and present this material in class.

CONCLUSION

In teaching this course, only the current topics and issues were emphasized for undergraduates. Graduate students taking the course were introduced to the emerging issues. In addition, features of CASE were taught when the techniques such as DFD was taught. Projects where the student developed these materials manually were not assigned. Thus CASE was tightly integrated into the entire course.

Graduate students had a difficult time understanding how to perform analysis for client-server application as they had not seen many client server programs before this course. They had a smoother transition to O-O analysis and design and could see how CASE fit into the whole process. When reengineering was given as a group project, it led to many complaints from the students. Rather than keep the original code, some groups redesigned and rewrote the entire application from scratch. The consensus of the students was that CASE was a very useful technology for systems analysis and design.

CASE tools on the PC such as Excelerator, and KnowledgeWare must be available in the labs for students to study, experiment and learn the multifacets of CASE and systems analysis and development. CASE is the tool that would help achieve productivity gains in the 90's for the systems analyst, designer and developer. Areas where CASE can help include project estimation and planning, project management, analysis and specification, design, coding, testing, support and communication. In addition, it is at the forefront of development in systems analysis and design; in areas such as object-oriented design, cooperative processing, and reengineering. If a strong foundation covering all these topics and how CASE can contribute to improving productivity is provided in the classes, the student will be better prepared to face the challenges of systems development in the real world.

REFERENCES

1. Graham C., "CASE Cracks Application Backlog", Datamation, March 15, 1991.
2. Crozier M., Glass D., Hughes J. G., Johnston W., and McChesney I., "Critical Analysis of Tools For Computer-Aided Software Engineering", Information and Software Technology, November 1989, pp. 486-496.
3. Norman R. J. and Nunamaker, J.F., "CASE Productivity Perceptions of Software Engineering Professionals", Communications of the ACM, September 1989, pp. 1102-1108.
4. Burkhard, D.L., "Implementing CASE Tools", Journal of Systems Management, May 1989, pp. 20-23.
5. Henderson, J.C., Cooprider, J.G., "Dimensions of I/S Planning and Design Aids: A Functional Model of CASE Technology", Information Systems Research, September 1990, pp. 227-254.
6. Boudin B.M., "Automate Your Software Development With Minimum Pain", EDN, June 7, 1990, pp. 107-109.
7. Kemerer, C.F., "An Empirical Validation of Software Cost Estimation Models", Communications of the ACM, May 1987, pp. 416-429.
8. Fosdick H., "Re-Engineering Mainframe Databases", Enterprise Systems Journal, November 1991, pp. 10-14.
9. Loy, P.H., "A Comparison of Object-Oriented and Structured Development Methods", Pacific Northwest Software Quality Conference 1989.
10. Coad, P. and Yourdon E., "Object-Oriented Analysis", Yourdon Press, Englewood Cliffs, NJ, 1989.
11. Bailin S., "Object-Oriented Requirements Specification Method", Communications of the ACM, May 1989, pp. 608-623.
12. Wilkinson R. A., "Object Oriented Requirements Specification for the Commodity Paging System", in Standards, Guidelines and Examples in System Software Engineering, IEEE Press, 1990, pp. 478-499
13. Chidamber, S. and Kemerer, C., "Towards a Metrics Suite for Object-Oriented Design", Proc. ACM Object-Oriented Programming, Systems, Languages, and Applications Conference, October 1991.
14. Jordan, E.W., and Machesky, J.J., Systems Development, Boston: PWS-Kent, 1990, pp. 532-533.
15. Goldberg, C., "Shifting CASE Target", Software Magazine, May 1992, pp. 25-27.
16. Flaatten P., "Cooperative Processing -- What and Why?", CASE TRENDS, June 1991. pp. 26-27,47.
17. Mercurio V. J., Meyers B. F., Nisbet A. M., Radin G., "AD/Cycle Strategy and Architecture", IBM Systems Journal, 1990, pp 170-189.
18. Weide B. W., Ogden W. F., Zweben S. H., "Reusable Software Components", in Advances in Computing Vol 33., editor Yovits, M.C., Academic Press. Boston, 1991.
19. Devanbu P., Brachman R., Selfridge P.G., Ballard B.W., "LaSSIE: A Knowledge Based Software Information System", Communications of the ACM, May 1991, pp. 34-49.
20. Harrison W., "Trends in the Development of Development Environments", Proc. of the First International Workshop on CASE, Cambridge, MA, 1987, pp. 227-232.
21. Prieto-Diaz, "Implementing Faceted Classification for Software Reuse", Communications of the ACM, May 1991, pp.88-97.

AUTHOR'S BIOGRAPHY

T.M. Rajkumar is an assistant professor of MIS at Miami University, Ohio. He received his bachelors from Indian Institute of Technology, Madras and a Ph.D. from Texas Tech University. His research and teaching interests include graphics, CASE, database and data communications.

ISSN 1055-3096