CASE TOOLS-SOFTWARE DEVELOPMENT AND MAINTENANCE AID: A SURVEY

INFORMATICA 1/91

Keywords: CASE methodology, CASE tools, life cycle, maintenance, reverse engineering

Saša Bošnjak
Laslo Šereš
Faculty of Economics,
Subotica, Moše Pijade 9

SADRŽAJ. U radu je data osnovna podela CASE proizvoda s posebnim osvrtom na njihovu upotrebu u pojedinim fazama životnog ciklusa softverskih proizvoda. Za razliku od uobičajenog pristupa kojim se ističe značaj CASE proizvoda u fazi razvoja softvera, ovde je posebna pažnja posvećena korišćenju CASE prizvoda u završnoj fazi, tj. pri održavanju softverskih proizvoda. S obzirom na probleme koji se javljaju u održavanju softvera namera je da ukažemo na neke nove tehnike koje treba da doprinesu smanjenju angažovanih ljudskih i materijalnih resursa u ovim poslovima.

ABSTRACT. This article deals with the basic classification of CASE products, with a special review on their usage in various phases of software products life cycle. A particular attention has been paid to software maintenance and the help of CASE products in these activities, being a quite opposite approach to the usual ones. Usually, one can find informations about the usage of CASE in the development of software, but very rarely anything about its usage in software maintenance. Our intention was to present some of the new techniques which have been developed to decrease the costs and increase the efficiency of maintenance in IS.

1. INTRODUCTION

CASE, or computer-aided software engineering, is a tool for programmers, analysts, and system engineers, as well as for bussiness planners and executives at all levels. It provides software tools to help planners to plan and document their work; to support systems analysts in analyzing and designing systems, and in documenting those tasks; and to take some of the drudgery out of programming while documenting it.

As the notion of CASE evolved over the last five years, the definition of a CASE tool broadened from meaning simple systems analysis and documentation tools to include fullfunction tools providing automated support for the entire software life cycle process. A CASE toolkit specializes in automating particular software tasks. CASE tool users frequently use several different toolkits as well as outside tools, such as fourth-generation languages, dictionaries, and DBMSes, during the software process. Selecting a CASE workbench involves matching hardware, development methodologies, and target systems that the workbench supports to the CASE user's development style and target system requirements.

Most companies spend 60 to 80% of their software budgets on maintaining existing applications. It's expected that fixing and reengineering existing applications will consume 90% of IS software resources by 1995. It's no wonder that companies are taking a hard look at their

software maintenance operations. They are specially interested in reverse-engineering products that assist in analyzing existing code to reconstruct original source information and specifications that have been lost or forgotten. Once reconstructed, those specifications can be fed into a code generator, which would reproduce the application in a new and improved structured form. Although reverse engineering sounds easy, it represents a major technical challenge.

2. CASE - SOFTWARE AUTOMATION

The crise in the development of software (SW) has made many people think about the ways of increasing productivity in this field. Current demands in dealing with the design of a new SW greatly surpasses the possibilities of the traditional approach in management, SW creation and documentation, and the cost of such development is not acceptable. The increase in the number of designers is not a good solution, since there are not enough designers. Even if one provides enough designers, the work on a large projects by using manual techniques would show that the managing of such a project is difficult, documentation incomplete, and it would be impossible to achieve that all the parts of the project fit. All these things become even worse if the number of designers working on the project is increased. Until recently, the main method used in the development of SW systems was the method of trying and making mistakes. The disadvantages of the traditional approach showed that well known engineering techniques should be used in the development of a SW product. SW engineering includes a systematic work in all stages: dealing with development, usage, maintenance, and withdrawal from usage of a SW product. Such product is characterised by the idea of life cycle and structural approach to the development. The idea of life cycle as the basic philosophy of SW engineering derives from the fact that every SW product goes through certain life phases. There are many opinions about the number and content of certain phases of life cycle, but disregarding the selected division, the quantity of required work on a particular project and the quantity of required documentation is constant. Among the others we have choosen the following division:

1. Project planning
2. Analyzing
3. Design
4. Programming
5. Testing
6. Application
7. Maintenance

The result of the computer aided SW engineering is the automation of SW development. Simply defined, CASE is SW automation. Most often, CASE means all the SW tools used in the development of SW. CASE may also be regarded as a combination of SW tools and structured methodology of SW development in which the tools automate SW processes, while the methodology defines the processes which are to be automated. The aim of CASE systems is not to reach the complete automatic development of SW, but only to automate this process in order to help people who work with them. There are many criteria for division of CASE systems, but we will classify them according to:

— universality and

— function of CASE system.

3.CLASSIFICATION OF CASE SYSTEMS ACCORDING TO THEIR UNIVERSALITY CASE

CASE Tools

CASE tools are a kind of graphic-oriented SW tools whose hardware (HW) environments are mainly microcomputers. In a broad sence, these are SW tools that provide automatic aid for development activities. CASE tools use very powerful graphics in order to describe and document SW systems as well as to promote user interfaces. CASE tools are so integrated that the data transmission between several tools is made possible without difficulties. Yet, there is no CASE tool that can provide complete automatic aid to development and maintenance of various SW systems.

CASE Toolkit

CASE Toolkit is a collection of integrated SW tools that provide complete automation of some phases of life cycle or automation of one function through several phases (as for example system analyses, logical database design, program design and implementation). Tools that constitute CASE Toolkit, use common information that are necessary for building and maintenance of SW systems. They also share the same user interface and the same interface between certain tools. According to the phase of development and the function which they automate, CASE Toolkits can be divided to those dealing with:

— analyses and design
These tools are used for the creation of SW systems specification. They include screen generating and report pointing, simulation and prototyping. Tools for analyses and design that have been written for IBM PC compatible computers are based on the following structural methodologies: De Marco's or Gane/Sarson's structured analyses, Martin's information engineering, etc.

— database design
This toolkit provides the automation of database design, as for example, making the logical database model,

generation of database scheme and description of database relation scheme.

— programming
These tools automate the steps in programming and testing. Some CASE Toolkits provide code generation based on informations from program specifications. Such CASE Toolkits are called code generators and they automatically produce completely documented programs ready to be used.

— maintenance and reengineering
These CASE Toolkits are directed to the existing SW systems. Such toolkits are made from tools for program analyses, reverse engineering, reconstruction, restructuring programs in use, and documentation. This toolkits are capable of maintaining a large number of program lines.

— project management
This type of CASE Toolkits support the planning function, controlling, managing and reporting, which are the characteristics of SW development and maintenance.

CASE Workbenches

CASE Workbench is an integrated collection of tools that support SW development through all development phases. When CASE Workbenches are combined with hardware (HW), the result is a workstation for SW development. While choosing CASE Workbenches, attention should be paid to system for which CASE Workbench is for. Also, it is important to take into account the demands of CASE Workbench users in the respect of development methodology and demands of the aim system.

4.CLASSIFICATION OF CASE SYSTEMS ACCORDING TO THEIR FUNCTIONS

In such a classification, serious problems arise due to lack of standards for CASE Systems division. Under these circumstances, different producers of the same type of CASE Systems put them in different categories or they put different types of CASE Systems in the same category according to their own criteria. Each producer tend to give his criteria as a standard. Despite the discussions about the reliability of functional division of CASE Systems, the following division can be accepted:

1. UPPER CASE
2. MIDDLE CASE
3. LOWER CASE

UPPER CASE

We refer to Upper CASE (UC) as a computer aided planning. Managers spend much of their time trying to understand the company, create plans for its activities, generate strategies for achieving the prescribed company's goals. They also establish standards for acceptable level of performance of the activities, which should result in fullfilment of the goals. Graphic diagrams are used to describe the company and its plans. According to these diagrams, the company's important aspects are decomposed and strategies planned. The planning methodology provides the structure into which the planning attributes have to be entered. For different plans, the structure of the company and its requirements remain the same, only the planning attributes values change. Every plan depends on timely informations to ensure its success. The use of UC products in planning gives a wide range of opportunities in reusing the earlier description and shortens the time required for designing a new or revising the existing plan.

MIDDLE CASE

A Middle CASE (MC) products are very helpfull in system analyses and design and belong to the first generation of CASE products as well as the UC. MC analyzes the problems and designs the solution for them. Most MC systems consist of diagraming and dictionary components, which are based on different methodologies. These systems are also used for symbolic description of the real world objects, using graphic diagrams. MC stores the type of knowledge that usually resides only in the minds of system analysts, e.g. the understanding how the company functions and what are its information needs. Thus the experience gained through years and the knowledge about companies functioning become more accessible. One can get these informations simply by retrieval of design specifications. Since MC specifications predominantly involve documenting a company's activities and the ways in which information serves it, only a small percentage of these specifications is directly mapped into Lower CASE systems (25-30%).

LOWER CASE

Lower CASE (LS) products are used in later phases of system development, to create a program specifications, the programming codes and the documentation related to the program specifications. LC products rarely provide a graphic component, since they are oriented on programming. However, it contains a data dictionary which enables one to enter specifications of the modelled real world objects. An active dictionary comprises three major components:

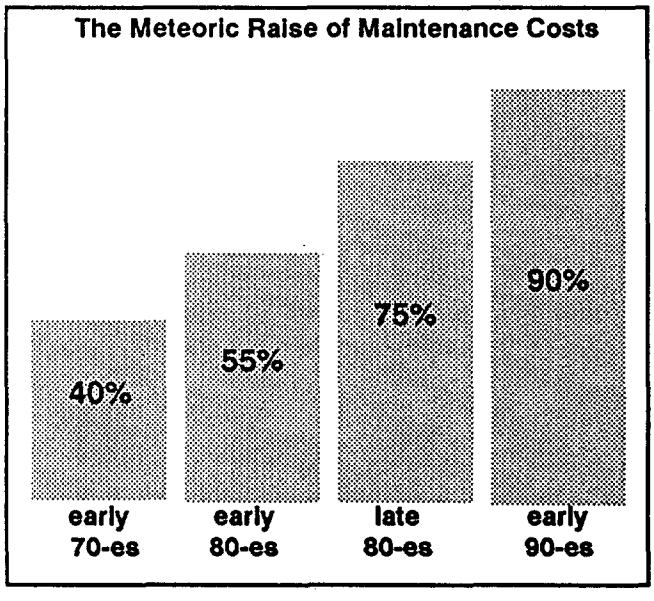
—a database in which to store the characteristics of the computing environment and application systems;
—frameworks for procedural logic and specific types

of procedural commands and modules contained within typical programs in the application systems; and — an activator capable of combining environmental and application characteristics with selected procedural - command frameworks and models to produce application programs.
LC products can generate both development and user documentation and make it available in various formats.

The usage of any category of CASE products demands a great deal of intellectual work, but the additional efforts and time, needed for completion of the SW design, are reversely proportional to the length and intensity of the CASE product implementation.

5.CASE SYSTEMS IN THE MAINTENANCE OF SOFTWARE

Majority of the SW development cost goes for its maintenance (up to 80 %). In USA this percentage is equal to $ 30 billions.



From this point of view, the efforts being made for reducing the maintenance costs are understandable, (especially in some branches as insurance, finance, telecommunications, etc., which are information-packed). We must not forget the fact that the maintenance quality is equally important for effectiveness and competitiveness of SW product as the development itself. Once this attitude is accepted, maintenance can't be regarded as a task of uninventive programmers or unsuccesful developing personell. New relations toward maintenance often can be seen through the following:

1. employment of the best team in maintenance jobs (since the savings made in this way are much bigger than the additional costs that could appear in the development field because of the migration of good teams)

2. the improvement of work organization related to maintenance, and

3. development of CASE systems that include tools for maintenance and reverse engineering.

Most of the existing CASE systems can not satisfy the needs of the designers and programmers working on maintenance: to provide the products that will enable reverse engineering of the existing application, using the advantages of CASE design, analysis and code generation. Even the producers of CASE systems themselves (IBM, TI, ORACLE Corp., etc.) admit that they still do not have the way to help the maintainers of the existing applications, although they have been working on such CASE tools designs.

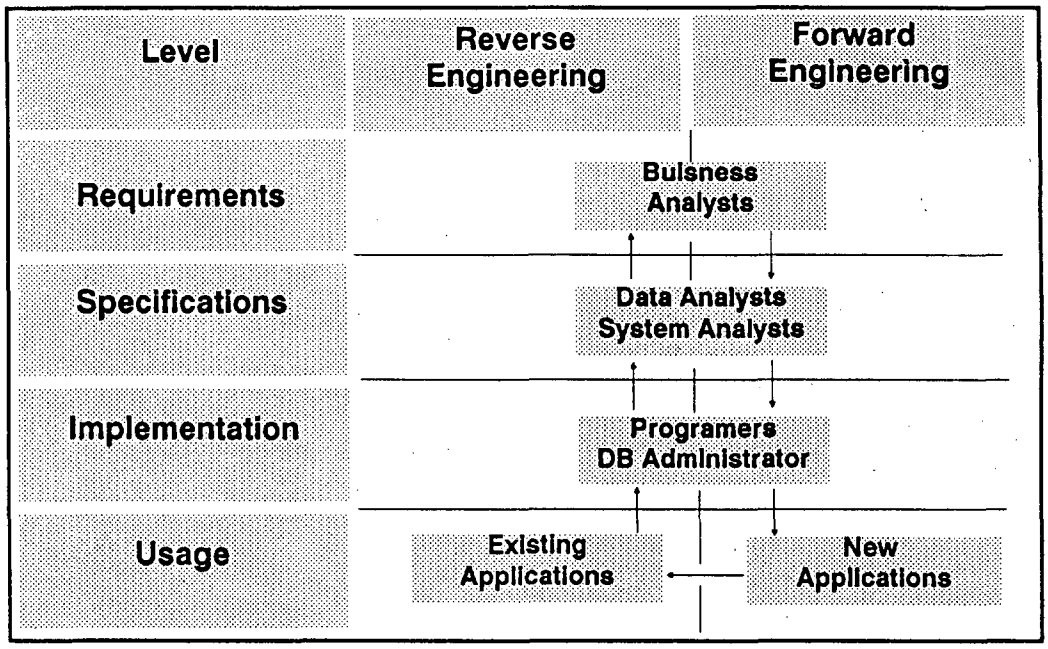
In these efforts their resources are, first of all, aimed to the design of tools for reverse engineering, that help in analyzing the existing source code. All this is aimed towards the reconstruction of original program specifications and the basic informations that have been lóst or forgotten long time ago. In this way reconstructed program specification can be put into code generator that will make the application in its new and better form, in the optional language of third or fourth generation.

Basic CASE concepts and products are directed towards the development of new applications and they are applied in scientific institutions and in a number of large systems. On the contrary, the new generation of CASE tools should include the possibility of fast change of existing applications, in accordance to the demands of market changes and new computer technologies. Such demands impose the spread of CASE concepts in respect of maintenance to include both improvement and upgrading the existing applications. IBM applications themselves, written in COBOL, include 77 million lines of source code, which speaks well enough about the dimension of maintenance and improvement of existing applications.

Reverse engineering enables separation of rules from all applications and uses them as a base for maintenance and improvement of these applications. Very often source code of an old application is not enough for the reverse engineering and additional informations about conditions of object system is also needed. It is impossible to expect from an individual or even a group to find out all the incomplete or missing components of the IS in use, so an Expert System as a part of a CASE tools could be of some help in solving this problem.

Classical CASE approach is based on top-down methodology of SW engineering, in which the information demands of a new IS have been worked out and documented. First step in this methodology includes carefull study and documentation of information demands, which are to satisfy information needs in a new IS. In the second step, the specifications are designed on the basis of study from previous phase. In the next step, programming and modelling of data are accomplished. Designer of database creates a database scheme that cor-

Re- Engineering Cycle



responds to the real system limits. Creation of program includes writing or generating a source code, compiling and testing. New CASE approach should enable an universal, simplier, and more effective way of IS development. Forward as well as reverse engineering, with the possibility of change of all informations in every common point would thus be included in CASE. Both of these engineerings together make a cycle of reengineering making possible reprojecting and improving the previously developed applications in an effective way. We expect these concepts to decrease the time needed for SW design.

6.WHY TO USE CASE SYSTEMS?

The benefits of CASE system's usage are great and they are the result of the CASE philosophy compounds. Models created by this systems allow better understanding of its functioning. The plans are more reliable, the decisions are more valid. The automation with CASE systems enables fast "what-if" analyses and choosing the scenario for better or worse case. During the system analysis, there is an opinion exchange on the relation designer end-users, so the diagrams and data dictionaries suit the end-user needs. The Middle CASE products give the possibility of prototyping by creating screens and reports which simulate the inputs and outputs of new application. Making prototypes in the phase of analysis and design, enables a simulation of searching and updating future database. It means that one can perceive the needs for information that some components of a system will satisfy, already in the phase of design. The Lower CASE products generate 60-80% of source code and considerably decrease the time required for SW development. Also these products make the program modification easier. The greatest benefit from Lower CASE usage, after all, is the possibility of prototyping during the development of complete application which is functioning as an independent system.

7. REFERENCES

MOGIN P. "*Karakteristike i uloga CASE alata u razvoju savremenih IS*", IIS - prolećni seminari informaciono-upravljačkih sistema '89

LAZAREVIĆ B. "*Razvoj i upotreba CASE alata*", *Metode i alati za projektiranje IS*, Opatija 1989

BACHMAN Ch. "*A CASE for Reverse engineering*", Datamation, July 1988

MCCLURE C. "*The CASE Experience*", Byte, April 1989

MCCLURE C. "*CASE is Software Automation*", Prentice Hall 1988

GIBSON M. L. "*The CASE Philosophy*", Byte, April 1989

GANE C., SARSON T. "*Structured Systems Analisys: Tools and Techniques*", Englewood Cliffs, Prentice Hall 1979

BEST L. J. "*Building Software Skyscrapers*", Datamation, March 1990

MOAD J. "*The Software Revolution*", Datamation, February 1990

MOAD J. "*Maintaining the Competitive Edge*", Datamation, February 1990