Journal of International Information Management

Volume 3 | Issue 1 Article 7

1994

The impacts and benefits of using CASE tools in the system development life cycle

Charles Necco
California State University, Sacramento

Nancy Tsai
California State University, Sacramento

Shyh-Jen Chang
California State University, Sacramento

The impacts and benefits of using CASE tools in the system development life cycle

Charles Necco
Nancy Tsai
Shyh-Jen Chang
California State University, Sacramento

ABSTRACT

The basic idea underlying computer-aided software engineering (CASE) technology is to increase a systems developer's productivity by providing a set of well-integrated, labor-saving tools that transform computer-based information systems (CBIS) development into an automated process. Two major issues of interest are addressed by this article. The first issue includes subjects related to the corporate data processing environment when adopting CASE technology for developing CBIS applications. The second issue includes subjects related to examining the impacts and benefits of using CASE technology as an automated development CBIS application tool within the system development life cycle.

INTRODUCTION

The electronic computer is probably one of the most important inventions in human history. Because of the computer's enormous speed and accuracy in calculation, computer-based information systems (CBIS) have been developed and used in almost every industry as a strategic weapon to gain a competitive advantage since the 1950s. Unfortunately, the developments of CBIS have faced many common problems such as cost overrun, late delivery, inadequate performance, impossible or cost-prohibitive maintenance, and unreliability. These problems are the major reasons for allocating 80% of the current management information systems (MIS) resources in an organization to maintaining the existing CBIS (Moad, 1990). Furthermore, the fast growing end user demand for new development and the shortage of skillful development personnel have increased the backlog for new application to an average of 5 years (Stamps, 1987).

Therefore, the greatest challenge facing the MIS practitioners and researchers is constantly exploring new techniques for improving the quality, reliability, and productivity of CBIS development projects. Several techniques which have emerged along the evolutionary path such as the system development life cycle (SDLC), structured software development

methodologies, fourth generaltion languages, prototyping, end-user computing, and information centers have been developed to address the high maintenance cost and huge backlog problems with some success in the past two decades. Currently, a new revolution technology named computer-aided software engineering (CASE) is being gradually adopted by the MIS field to automate every process of CBIS development in an attempt to improve the productivity and quality of the work at the same time.

The basic idea underlying CASE technology is to combine CASE tools and structured development methodologies with the SDLC. The CASE tools can automate the CBIS development process; and the methodologies define the process to be automated. The front-end or upper CASE tools can computerize the structured diagramming techniques needed to prepare documents for facilitating the understanding of the current and proposed CBIS during the analysis and design phases of the SDLC. The back-end or lower CASE tools can create a central repository to store every piece of system data related to CBIS development in order to not only computerize the programming and testing tasks required in the implementation and maintenance phases of the SDLC, but also to support the management function for planning and controlling the project.

Thus, the focus of CASE technology is to increase a systems developer's productivity by providing a set of well-integrated, labor-saving tools that transform CBIS development into an automated process. The other potential benefits of using CASE technology include: improving the quality and accuracy of the system documentation, reducing the system development backlog, decreasing the development and maintenance costs, cutting down development time, and enhancing management planning and controlling.

Two major issues of interest are addressed by this article. The first issue includes subjects related to the corporate data processing environment when adopting CASE technology for developing CBIS applications. The second issue includes subjects related to examining the impacts and benefits of using CASE technology as an automated development CBIS application tool within the SDLC.

THE RESEARCH

A questionnaire designed to address these issues was sent to 1000 organizations listed in the *Directory of Top Computer Executives* (1989 Edition) who had at least one mini-computer and/or mainframe computer in order to increase the response rate. It was believed that organizations with these types of computers would have larger data processing budgets and would be more inclined to adopt CASE technology. In an attempt to provide a comprehensive view of the usage of CASE technology in the U.S., sample organizations were selected according to the ratios for the thirteen different industries listed in the East and West editions of the directory. Although a stratified and systematic sampling method was used to select the sample, it was believed that the diverse range of organizations should represent a fair sample of the CASE technology users' implementation experiences. It is hoped that future CASE technology users might have an opportunity to share some experiences from the collective successes and failures of current CASE technology users.

Although the questionnaire was relatively long and complex, 117 usable surveys were returned. Of these, 42% indicated that their organizations were currently using a CASE tool on some kind of hardware platform, i.e, mainframe, mini, or microcomputer, to support at least one SDLC phase. These organizations were asked to evaluate their experience with CASE technology. The non-users of CASE technology were asked to identify their reasons for not using the technology.

DESCRIPTIVE RESULTS

Table 1 presents a profile of the CASE technology user organizations investigated in this survey. In general, CASE technology user organizations tend to be corporations with strong financial strength to support data processing (DP) personnel who practice structured CBIS development methodologies and use CASE technology in order to increase implementation productivity. Furthermore, these user organizations have an average of more than a quarter of a century of experience in developing CBIS which could indicate they have many old systems with high maintenance costs and thus, need the use of CASE technology to reduce these costs in a reasonable manner. A brief discussion of some other major conclusions based on Table 1 follows.

Table 1. Organizational Profile

Organization Profile	Average Statistic
Annual Revenue (millions)	$15,511
Annual DP Budget (thousands)	$903,519
Total Software Budget (thousands)	$17,134
Total Number of Employees	11,375
Total DP/MIS Employees	236
Number of Years Using Computers	26
Hardware	
Mainframe	74%
Mini	19%
Microcomputer	15%
Development Methods:	
In-House	74%
Acquiring Packages	19%
Contracting Out	14.9%
Other	12%
Language:	
ADA	2%
Assembly	--
BASIC	--
C	--
COBOL	71.4%
PL/1	6.1%
4GLs	14.3%
Other	6.1%

The finding shows that the majority of the user organizations used mainframe computers (74%) was expected since large and complex CBIS generally require mainframe computers and are more likely to justify the use of automated CASE tools for their development and maintenance. In terms of software development methods, the high percentage of in-house application development (74%) among user organizations may be the major cause for adopting CASE tools to gain the advantages of faster and cheaper development and maintenance.

COBOL, which is obviously not one of the user friendly fourth generation languages, is overwhelmingly (71.4%) used among the user organizations as the production programming language. This result could explain why most of the current market's code generators for back-end CASE tools are developed to produce COBOL code.

Table 2 provides a listing of some possible reasons for implementing CASE technology which are presented in the research literature, and indicates how frequently the user organizations viewed these reasons as appropriate. Interestingly, there is no one reason agreed upon by the majority of the user organizations to use CASE technology. The user organizations appear to have a balanced view between higher productivity (47.9%) and better systems quality (31.3%) as the most important reason to use CASE technology. This result supports the purpose of the current CASE tools usage as stated in the MIS literature. None of the user organizations indicated that reducing DP personnel or improving team communication were reasons for adopting the use of CASE tools.

This could imply that CASE technology is being used only as a set of automated tools to assist individual DP personnel to do their job better or faster and is not being used to replace DP professionals or being used as a communication tool as in the responding organizations.

Table 2. Reasons for Using CASE Technology

REASONS	Total Users	Percentage
Improve Productivity	23	47.9%
Improve System Quality	15	31.3%
Ease Future System Maintenance	3	6.3%
Integrate Corporate Information Systems	2	4.2%
Automate System Development	1	2.1%
Reduce DP Personnel	0	0%
Improve System Documentation	1	2.1%
Improve Team Communication	0	0%
Other Reasons	3	6.3%
TOTAL	48	100%

Table 3 presents the current CASE tools used by organizations to support each of the four SDLC phases, i.e., analysis, design, implementation, and maintenance on the different hardware platforms. This table also indicates how many months these organizations have been using CASE tools. It was interesting to find that more organizations were using front-end CASE tools to support the analysis (39 cases) and design (37 cases) phases than back-end CASE tools to support the implementation (24 cases) and maintenance (21 cases) phases. More

front-end CASE tools were being used on microcomputers—analysis phase (85%) and design phase (76%)—than on the mainframe or minicomputers suggesting that the use of front-end CASE tools may be more appropriate for the newer hardware technology.

However, the above usage figures were reversed for the back-end CASE tools that support the implementation and maintenance phases. More back-end CASE tools were used in the mainframe computer environment for the implementation phase (50%) and the maintenance phase (52%), whereas the usage of back-end CASE tools on microcomputers (38%) and minicomputers (12%) was much less in the implementation phase. CASE tools usage for the maintenance phase is similar to that for the implementation phase suggesting that the use of back-end CASE tools may be more appropriate for the older hardware technology.

A similar pattern was also found for the average time CASE tools have been used in these organizations. In general, CASE tools have been used significantly longer on mainframe computers than on microcomputers and mini computers. This is true in all phases but the analysis phase. Microcomputer analysis type CASE tools have been used slightly longer (24 months) than their mainframe counterparts (21 months).

This survey data on the current CASE tools usage seems to suggest the following. First, the use of CASE technology is indeed in its infancy. Most CASE tools have been implemented by these user organizations for only about two years. Second, microcomputer-based CASE tools currently dominate the front-end CASE market, while back-end CASE tools are more prevalent in the mainframe market. This situation may change in several years when back-end CASE tools are also available on microcomputers which can support mainframe computer application development.

Table 3. CASE Tools Usage by Phase and Hardware

Phases	Hardware	Count	Phase Percent	Average Months Used
Analysis	Mainframe	4	10%	23
	Mini	2	5%	6
	PC	33	85%	24
	Subtotal	39	100%	AVG 23
Design	Mainframe	7	19%	24
	Mini	2	5%	6
	PC	28	76%	18
	Subtotal	37	100%	AVG 19
Implementation	Mainframe	12	50%	33
	Mini	3	12%	20
	PC	9	38%	13
	Subtotal	24	100%	AVG 24
Maintenance	Mainframe	11	52%	34
	Mini	4	19%	15
	PC	6	29%	16
	Subtotal	21	100%	AVG 25

Additionally, the organizations were asked to evaluate the impact of the use of CASE tools on the different phases of the SDLC. Table 4 presents a summary of the impact of the use of CASE technology on the overall SDLC. The user organizations have experienced changes in terms of the percentage of DP staff assigned to the front-end activities and to the back-end activities. An increase of DP staff allocation was reported for the front-end phases, i.e., analysis and design. A decrease was observed in the two back-end phases, i.e., programming and maintenance. The design phases increased the most (6.7%) and the programming phase decreased the most (-7.9%) in terms of the percentage of DP staff allocation among the four phases of the SDLC.

Table 4. CASE Impact on System Development Staff Allocations

Phases	Before CASE Usage	After CASE Usage	Percentage Change
Analysis	24.5%	18.0%	6.5%
Design	28.1%	21.4%	6.7%
Programming	36.3%	28.4%	-7.9%
Maintenance	24.3%	19.3%	-5.0%

These results clearly support the current literature which suggests that the use of CASE tools has shifted resource allocation from back-end to front-end activities in the SDLC by placing more emphasis on the analysis and design phases.

Furthermore, the user organizations were asked to evaluate a listing of some of the potential impacts of the use of CASE tools which have been presented in the MIS literature. All the impacts were measured by using a 1-7 Likert scale where 7 is strongly agree, 4 is neutral, and 1 is strongly disagree. Table 5 presents a summary of how the CASE user organizations viewed these impacts on the different phases in the SDLC. A brief discussion of the major findings follows.

The user organizations indicated that the usage of CASE tools has indeed improved system design accuracy. This result is not surprising since CASE tools can help to reduce system design errors by providing developers with several automated error checking and reporting functions. This can relieve the developer from the tedious and time consuming tasks of error checking and correcting design specifications. The developer then could use the available time more efficiently to concentrate on a more thorough system analysis and design so that the user requirements are correctly and initially incorporated into the system. This can lead to a more complete, accurate, and consistent system design.

Not surprisingly, the responding user organizations found that the use of CASE tools has positively increased user involvement, and the related communication between developers and users in the system analysis and design phases. This is probably due to the fact that CASE tools have the capability to quickly generate graphic models for analyzed systems and can instantly accommodate changes to designed systems. Moreover, this higher quality of communication between user and developer about a target system through the use of CASE tools can produce a system which better meets users' needs.

The overall responses for the implementation phase show that most user organizations agree that CASE tools have a positive impact on this phase. However, they did express a negative opinion about the statement that the use of CASE tools can assure faster system development (3.5 average score). An explanation for this negative response may be that current CASE tools are not being integrated as one software package. Some upper CASE tools offer front-end analysis and design capabilities only; some lower CASE tools offer back-end code generation and testing only. Thus, the output of a front-end analysis and design CASE tool cannot be used as an input for a back-end CASE tool for automated code generation and programming testing. The time and effort needed to build an interface to connect these two types of CASE tools might offset the faster system development provided by the individual CASE tools. Hopefully, future integrated CASE software packages will offer both upper and lower CASE functions to facilitate faster system development.

As expected, CASE tools did offer the user organizations the potential to increase programmer productivity and reusability of programs. This is not surprising, since the lower CASE tools contain code generators which allow developers to generate modular code from a design specification compatible with a high-level programming language, such as COBOL. Furthermore, the program source code can then be stored in the CASE tool's central repository for future usage. Subsequently, a code generator can be used to interface with the central repository to retrieve the program source code and regenerate it for a variety of appropriate programming languages.

Moreover, the accuracy of computer generated code can be checked by using a test data set. If there is no human error in the design input, the system should produce correct results the first time. Incorrect testing results can be used to pinpoint design errors. Thus, the code generators and central repository of a CASE tool can provide user organizations with some positive impacts on the tasks performed during the system implementation phase.

Table 5. CASE Impact of Different Phases in SDLC

IMPACT ITEM	MEAN
Analysis & Design Phases:	
1. Improve Design Accuracy	5.0
2. Increase Analysis and Design Productivity	4.9
3. Increase User Involvement	4.8
4. Improve Communication between Developer and User	4.8
Implementation Phase:	
1. Increase Programmer Productivity	4.8
2. Increase Reusability of Programs	4.7
3. Decrease System Testing Time	4.4
4. Assure Faster System Development	3.5
Maintenance Phase:	
1. Improve System Documentation	5.7
2. Improve System Maintainability	5.6
3. Improve Maintenance Productivity	5.3

With regard to the maintenance phase of the SDLC, one major impact of the use of CASE tools is the ease of future system maintenance. Indeed, the greatest positive impacts attributed to the different phases in the SDLC by the user organizations are in the maintenance phase. Historically, the major causes for system maintenance is the need to meet new user requirements by adding new system enhancements. Here again, the capabilities of CASE tools can be used to help a developer do a more thorough system analysis and design so that the new user requirements are accurately and completely incorporated into the system. This in turn not only helps to reduce the subsequent system maintenance caused by ill-defined user requirements, but also gives the developer more opportunities to focus on adding new enhancements to the system.

System documentation has always been a problem in a non-CASE environment since much of the documentation has been incomplete, error-prone, and labor-intensive. The use of CASE tools has totally changed the process of generating documentation from a manual operation into an automated one. The documentation of the detailed system design specification becomes the central and the most important element in a CASE environment. This is because the design specificaiton is the major input for the subsequent implementation and maintenance phases. Furthermore, all system documentation is stored in a central repository to enable the developer to perform version control and to track all changes. The traceability capability of the central repository enforces documentation standards and quality. The use of CASE tools has made documentation a by-product of the development process rather than a separate time-consuming effort. For the above reasons, it is believed that the advantage of improving system documentation had the highest score among all the impact items on the SDLC.

Table 6 provides a listing of some potential benefits from the use of CASE tools as expressed in the MIS literature, and indicates how user organizations viewed the realization of these benefits. The same 1-7 Likert scale was used as described for Table 5. Interestingly, the greatest benefit attributed to the use of CASE tools was better usage of DP resources. An explanation for this finding may be that historically most organizations used only about 20% of their DP resources to develop new systems and 60% to 80% of their DP resources to maintain old systems in a non-CASE environment (Moad, 1990). On the other hand, the reverse engineering feature of CASE tools can enable developers to extract the design specification of an existing system from the central repository and revise it to meet the new requirements and desired enhancements. Individual programs can then be generated or regenerated by code generators. Therefore, organizations can effectively allocate more resources to creating new systems as the environment evolves, rather than fixing problems constantly for the existing systems.

Additionally, the user organizations gained improvements in system reliability, productivity, and quality. The main reasons for these improvements are probably due to different capabilities provided for the use of CASE tools, such as: (1) methodology training and enforcement; (2) system analysis diagrams support; (3) errors and consistency checking; (4) system testing and enhancement modificaiton; (5) code generation and reverse engineering; and (6) automated system data and documentation storing and retrieving.

Table 6. CASE Benefits

BENEFIT	MEAN
1. Better Use of DP Resources	5.3
2. Increase System Reliability	5.1
3. Increase System Productivity	5.0
4. Assure Higher System Quality	4.9
5. Integrate Software Development & Data Administration	4.8
6. Assure Greater User Satisfaction	4.4
7. Lengthen System/Program Life	4.4
8. Increase System Efficiency	4.2
9. Better Project Management	4.2
10. Improve System Portability	3.9
11. Reduce Backlog	3.9
12. Enable End User to Design System	2.8

The CASE benefit related to data administration is probably one of the most important but least discussed in the MIS literature. The user organizations in this survey perceived some positive gain in integrating the activities of software development and data administration. This is probably because most system components, i.e., design specification, data, and programs, are stored in the CASE central repository and shared by all the applications. Actually, the CASE central repository can be considered as a single point of control for all the system and software related data.

If the use of CASE tools does help in improving systems quality and reliability, it should logically result in greater user satisfaction. Surprisingly, the user organizations did not experience an impressive improvement in user satisfaction after using CASE tools. It is also worth noting that the user organizations had the same somewhat lower opinion toward the benefit of improving system efficiency. This finding may imply that most developers in these user organizations believe that machine-generated code is less efficient than programs written by human programmers. This suggests that improving the efficiency of code generators can be an important area for CASE tool manufacturers to investigate.

The forward engineering features of CASE tools can help a system developer do a thorough analysis and develop a more accurate system to meet a user's needs. The reverse engineering capability of CASE tools can assist a system developer with incorporating changes in an existing system. Thus, it is reasonable that an extended system life is being experienced by the user organizations.

The use of CASE tools should provide some positive benefits in the area of project management since these tools provide better project control and enable better use of resources. Surprisingly, the survey results do not support this statement. It is believed that when more integrated CASE tools are used for a longer period of time, CASE user organizations will experience better project management.

In theory, it would seem that the use of CASE tools should reduce the system development backlog because of the improved development productivity and system quality. To the contrary, the user organizations did not believe that the use of CASE tools has reduced the system development backlog. This may be due to the fact that most of the current user

organizations had used CASE tools for only a short period of time. Therefore, most of them may still be in the learning stage. Their current use of CASE tools may be too short for the user organizations to have experienced or even recognized the impact of CASE on the systems backlog. Long-term follow-up studies may be needed to understand the true impacts of CASE tools on the system backlog.

User organizations also have a negative opinion about the ability of CASE tools to improve system portability. The lack of code generators for different languages may be one possible explanation for their belief that the use of CASE tools has not improved system portability. Finally, the user organizations did not believe that end users can be trained to use CASE tools to develop systems. This finding may be based upon the following: it is difficult to use the current CASE tools; a CASE user should have a formal structured methodology background; and there may be resistance from the current system developers who wish to protect their jobs.

CONCLUSION

This paper surveyed the attitudes of organizations which use CASE tools to determine the degree of success being achieved in the use of this technology. The results indicate that microcomputer-based CASE tools dominate the front-end CASE market. However, back-end CASE tools are more often used in the mainframe market. The two most important reasons given for implementing CASE tools were to improve systems development productivity and system quality. In the evaluations provided, the analysis and design phases gained the most positive impact from using CASE tools, followed by the maintenance phase and implementation phase. Significant CASE benefits were reported in terms of improving the use of DP resources, system reliability, productivity, quality, and integrating software development and system data administration. User organizations did not experience a gain in improvement in terms of system portability and reduced backlog. They also did not believe that end users can be trained to use CASE tools to design and develop systems.

Based on these findings, the following recommendations are offered. CASE tools have received a great deal of attention as an automated means to develop high quality and reliable systems for users. It is believed that the future usage of CASE tools will dramatically expand and improve a developer's productivity in every phase of the SDLC. Therefore, MIS educators should change the focus of their academic programs in order to prepare students to meet the job requirements in a CASE environment. The emphasis in system courses should be on the structured system development methodologies which are the necessary foundation for the use of CASE tools. Programming courses should be concentrated more on the basic structured concepts rather than coding since most of the programs will be generated by code generators in the future.

There are also some recommendations based upon the somewhat negative findings which should be of interest to CASE researchers and manufacturers. They should enrich their future CASE products in terms of (1) increasing the efficiency and variety of code generators; (2) integrating the front-end and back-end CASE tools into one software package; (3) developing a more user friendly interface; (4) escalating the capabilities provided by reverse engineering; and (5) expanding system portability to provide compatibility with hardware and software platform.

REFERENCES

Bachman, C. (1988, July 1). A CASE for reverse engineering. *Datamation*, 49-56.

Burkhard, D. (1989, May). Implementing CASE tools. *Journal of System Management*, 20-25.

Davis, J. M. (1988, May). CASE deciphered. *Database Programming and Design*, 42-53.

Directory of Top Computer Executives, 1989 Edition.

Gibson, M. L., Snyder, C. A. & Rainer, R. K. (1989, May). CASE: Clarifying common misconceptions. *Journal of System Management*, 12-19.

Martin, J. & McClure, C. (1985). *Diagramming techniques for analysts and programmers*, Prentice-Hall.

Martin, J. & McClure, C. (1988). *Structured techniques: The bases for CASE*, Prentice-Hall.

Martin, J. (1990). *Information engineering: Design and construction*, Prentice-Hall.

McClure, C. (1987, September). Software automation. *Business Software Review*, 28-34.

McClure, C. (1988, August). The CASE for structured development. *PC Tech Journal*, 51-67.

Moad, J. (1990, February 15). Maintaining the competitive edge. *Datamation*, 61-66.

Stamps, D. (1987, July). CASE: Cranking out productivity. *Datamation*, 55-58.

Statland, N. (1989, April 1). Payoffs down the pike: A CASE study. *Datamation*, 32-33.

Rubenstein, B. & Chilkofsky, E. J. (1988, March). CASE: Reliability engineering for information systems. *IEEE*, 11-16.