The effective use of automated application development tools

by P. J. Guinan
J. G. Cooprider
S. Sawyer

In this paper we report on the results of a four-year study of how automated tools are used in application development (AD). Drawing on data collected from over 100 projects at 22 sites in 15 Fortune 500 companies, we focus on understanding the relationship between using such automated AD tools and various measures of AD performance—including user satisfaction, labor cost per function point, schedule slippage, and stakeholder-rated effectiveness. Using extensive data from numerous surveys, on-site observations, and field interviews, we found that the direct effects of automated tool use on AD performance were mixed, and that the use of such tools by themselves makes little difference in the results. Further analysis of key intervening factors finds that training, structured methods use, project size, design quality, and focusing on the combined use of AD tools adds a great deal of insight into what contributes to the successful use of automated tools in AD. Despite the many grand predictions of the trade press over the past decade, computer-assisted software engineering (CASE) tools failed to emerge as the promised "silver bullet." The mixed effects of CASE tools use on AD performance that we found, coupled with the complex impact of other key factors such as training, methods, and group interaction, suggest that a cautious approach is appropriate for predicting the impact of similar AD tools (e.g., object-oriented, visual environments, etc.) in the future, and highlight the importance of carefully managing the introduction and use of such tools if they are to be used successfully in the modern enterprise.

The software industry has searched for the "silver bullet" in application development (AD) productivity for over four decades,[1] yet the field continues to have many highly visible examples of software development failures. The opening of the new Denver, Colorado, international airport was delayed for more than a year—at a cost of more than $1 million per day—due to a software problem in the automated baggage-handling system. In developing a new air traffic control system, the U.S. Federal Aviation Administration is currently five years behind schedule and more than $1 billion over budget.[2] One study found that almost 75 percent of all AD projects are never completed,[3] while other studies have estimated that between one-third and one-half of all systems projects never reach the implementation stage.[4] This vexing problem has led practitioners and researchers alike to look for an answer—a silver bullet—in many areas. Despite the huge amounts of attention and effort focused on improving AD tools and methods, however, few clear-cut solutions to improving the results of using AD tools have emerged. Over the past 40 years, the procession of new AD tools and methods has led to only incremental improvements in overall performance. When new tools or methods are introduced, many in the field seem to rush to embrace them, only to realize later that unrealistic expectations cannot be met.

In our study we find this to have been particularly true for computer-assisted software engineering (CASE) technologies. Moreover, and perhaps more importantly, this may also be the case for other new

0018-8670/97/$5.00 © 1997 IBM

approaches to software development such as object-oriented (OO) approaches[5] and other proposed AD solutions. Drawing on data collected in our four years of studying AD projects in 15 of the most highly regarded companies in the United States and Canada, we feel that we can appropriately point to the benefits and drawbacks of using automated tools in organizations today in the hope that what we have learned from the past will contribute to the successful use of the next generation of tools and methods.

In this paper we begin our discussion by looking back at a major development in software practices—the development and delivery of automated design and development tools that were positioned to revolutionize the way in which systems were developed. The results of our four-year longitudinal study show that tools are helpful only under the proper set of conditions—and in some situations the tools may actually hinder AD performance. We have also included a general reference section that contains other relevant related work.

One finding from our study indicates that the use of automated development tools may have a very positive impact from one point of view (e.g., the tools may enable developers to create systems with which users are more satisfied), while having little or even a negative impact on other AD performance measures (such as adherence to schedule). The relationship between tool use and performance is complex and affected by several key mitigating factors that we found, including: training both in specific tool operations and general AD processes, use of structured methods, project size, team interactions, and the quality of the initial application design. At the same time, however, some of our findings were counterintuitive and illustrate the need to better examine and understand how tools and methods actually affect AD performance.

As an example, we found that application development teams with more operational training in specific tool use receive higher satisfaction ratings from their end users—but the development teams are also more likely to miss their planned schedules. By conducting additional analysis, however, we developed a much more complete picture of the role of training in tool use. When teams receive *both* tool-specific operational training *and* more general AD training, there is *less* schedule slippage and *greater* user satisfaction. If organizations do not provide both types of training for their developers, they may be unhappy with the tool impacts that they see but not know what they could have done differently to make the tools more effective in their organizations.

Another important finding of our study is the need to better understand what can be characterized as the "backlash" against CASE—and what it may mean to the future of our industry. For example, we expected tool use to be higher and related to increased AD performance when it is accompanied with a design document of high quality. Surprisingly, higher quality design documents were related to *lower* levels of tool use. In *post hoc* interviews with the development teams, we learned that many developers felt that there was too much excitement and expectation with using CASE, and they were therefore less willing to use the tools—even if they expected an increase in the quality of their work. Finally, perhaps the most important and encouraging result of our study is the 50 percent performance improvement achieved by teams that extensively used *both* CASE tools *and* formal structured methods. Conversely, for teams characterized by low use of both structured development processes and CASE tools, their systems cost more and were less acceptable to the project stakeholders. We found that automated development tools may be a magnifier. That is, for teams with well-structured processes, use of such tools enhanced the process and improved performance. For teams with more informal or *ad hoc* processes, tool use abetted chaos.

It is clearly important for information systems managers and developers to have realistic expectations for the use of automated application development tools. The potential benefits from these tools are not universal, and they can lead to important performance trade-offs. As an industry, the information systems groups continue to experience serious failures and limited success in using these tools. "Those who cannot remember the past are condemned to repeat it."

To elaborate on these findings, we begin with a brief review of CASE research to set the stage for the study we discuss. We then describe our research approach, a longitudinal study in which we collected data at specific points over the life cycle of 57 development projects. Data collection is organized using the IBM AD/Cycle* model.[6] Analysis focuses on analysis and design activities, and findings do not pertain to the maintenance aspects of the model, since we looked at new development. This section is followed by a presentation of the major results of the study. We discuss the use of application development tools and

examine the effect of this use to AD performance. We conclude the paper by focusing on implications for managers and system development teams.

CASE history

The rise in interest in CASE tools is best understood by using the growth of the software industry as a backdrop. Recent figures put the present value of software at $2.7 trillion.[7] Estimates of the software industry calculated it to be a $1 trillion business in 1995, growing by nearly 8 percent annually.[8] This is particularly notable since the investment has all occurred within the past 40 years, with most of it in the last 20 years. Software systems and their development are central to the modern enterprise.

Drawing on data collected, we can appropriately point to the benefits and drawbacks of using automated tools.

With this unparalleled growth come a number of productivity challenges. Software development is both complex and problematic.[1,9–12] The need to better support developers has led to several evolutions in the tools and methods for software development. For example, structured methods and structured programming are techniques to assist developers to design and build software more systematically.[6,13,14] Improvements in programming languages and their compilers and debuggers increase developers' productivity.[15,16]

In the 1980s this attention to tools rapidly expanded as CASE began to appear.[14] As the interest in CASE tools rose, research reporting on the use and productivity of CASE tools remained scarce and mixed. While the empirically based research on CASE tool use was equivocal at best, trade journals presented powerful claims of CASE tool use success.

For example, drawing on empirical data, Lempp and Lauber[17] found the quality of CASE-abetted systems is higher, although costs are higher. This study also reported that the use of CASE tools encourages developers to spend more time and effort documenting their work. More recent research indicates that tool use has no impact on the final quality of a system.[18] These contradictory results also occur in other research efforts. Surveys and observations across multiple organizations find more questions than answers about the impacts of CASE tool use on productivity, quality, and cost.[19,20] Scientific research also highlights limitations within the CASE tool suites and reveals significant social dilemmas that organization members experience after purchasing and using the tools.[21]

Concurrently, the trade literature frequently touted the benefits of using CASE tools. While the research literature reported mixed findings, Burkhard[22] reported " . . . there were strong indications that CASE actually improves productivity." Other articles reported tremendous success stories of CASE use,[23,24] but these claims were based on anecdotal information. For instance, a report circulated by one consulting group indicated that CASE use was creeping into most major companies, barriers to CASE use were falling, and pilot CASE projects had been very successful.[25]

Given the mixed views of CASE tool use, we recognized the need to approach the problem from a solid theoretical and empirical perspective. To understand how CASE tools are used, we felt it was imperative to study development teams across the systems development life cycle, across multiple development efforts, and across many organizations, with both objective and subjective performance measures. To our knowledge, no other study has attempted to collect data about CASE use and impacts in such a comprehensive way.

Study approach

Our interest in automated application development tools—their use and their effects on AD performance—derives from our ongoing research on how groups of people work together, how they use information technology (such as CASE tools), and how this impacts team performance. Application development is a particularly interesting domain to examine because the work demands a team orientation. CASE tools are an excellent example of cooperative work methods and tools, and improving AD performance is a critical issue for our industry. In order to accomplish the research objectives, the study had to be rigorous and based on theory-driven research models. Specifically, the theoretical and conceptual

support for this study includes related work from information systems research, organizational theory, small group theory, and communications research.[26–32] In addition, to make the results of this research useful to application development organizations and personnel, the study had to focus on relevant issues and concepts. Hence, we based our measures and present our results in ways that reflect current industry standards.

In order to compare results across organizations, we focused on large companies that had extensive in-house information system departments. To control the project size, projects had to be 12 to 18 months in planned duration. The selected projects were business applications with some strategic relevance to the company. To accommodate the changing nature of AD projects across the life of the project, we chose to follow each project from inception through delivery and system use. We developed several data collection instruments for use in gathering data from developers, managers (information systems and line-of-business managers), and users. Software metrics on each project (e.g., function points, labor costs, schedule adherence) were also collected.

Table 1 summarizes our sample population. Data were collected on more than 100 projects (not all completed) at 22 sites of 15 organizations in the United States and Canada. Contributing organizations represent financial services, manufacturing, and high-technology industries from the *Fortune* magazine Fortune 500 list. For each project, we surveyed the development team at three stages of project development: at the end of requirements, at the end of design, and at project implementation. At the end of requirements and implementation we also surveyed the key managers who were invested in the outcomes of the projects. These "stakeholders" provided a critical perspective on AD team performance, because they were heavily invested in the outcomes of the projects. After the systems were in operation for approximately six months, we surveyed the end users and user managers on their satisfaction with the system. Finally, we gathered data on the technical environment of each organization, including descriptions of the platforms and hardware and software configurations for each project. Of the more than 100 projects we began tracking, 57 were completed, implemented, and placed in production. From these 57 projects come the data for this paper.

Table 1 Description of the four-year study

Environment	How Many
Contributing organizations	15
Different CASE tools used	20
Projects completed	57
Participants	More than 2000
Observations per team	Approximately 4000
Data points	Nearly 500000

Through the course of the four-year study, we visited the project sites on a quarterly basis. During these visits we interviewed project team leaders, key technical people, key managers in the information systems department, as well as senior managers in many of the companies. We gathered project documents, observed daily work, and spent time with individual members of many of the teams. We maintained phone and mail contact with all projects on a regularly scheduled basis. Each project contact was arranged directly with the project team and the site sponsor. At the completion of the data collection, we had collected data from more than 2000 people. These data include more than 4000 observations per team, for a total of nearly 500000 data points.

This data set provides an exceptionally rich picture of automated development tool use in organizations. The study is contextual, drawing from data about the organizational environment and the department environment as well as the individual project. The data include a variety of computing infrastructures, including mainframe, local networks, uncoupled workstations, mixed vendor shops, and rudimentary client/server systems. CASE tool use varies widely across the projects in our sample: both by project and by phase. Studied projects made use of more than 20 different CASE tools.

In the following sections we highlight our findings on CASE tool usage and application development team performance. To present this analysis, we begin by describing how we analyzed the CASE usage data. We also describe the key performance measures used. Finally, we explain how various characteristics of the software teams, the development projects, and the training and use of the tool relate to technology usage and, ultimately, to AD team per-

Figure 1 A life-cycle model of CASE tool use

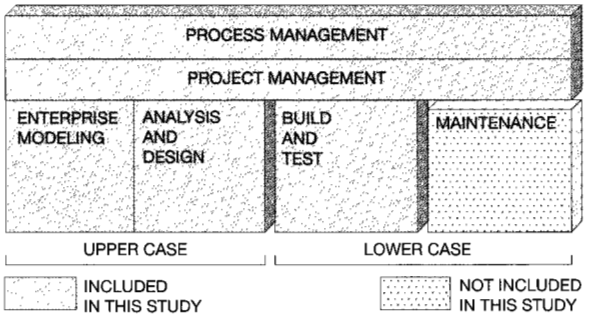


Table 2 CASE tool use at different project stages

	Aggregate Values for Analysis/Design Phase	Build/Test Phase
Upper CASE	3.65	2.44
Lower CASE	1.85	1.51
Process management	2.14	1.67
Project management	1.85	1.44
Overall usage	2.38	1.77

Scale: 1 no usage; 2 3 4 weekly usage; 5 6 7 daily usage

formance. The following research questions were addressed by this study:

- How are automated tools used in AD projects?
- What are the impacts of automated tool use on AD performance?
- Which key factors influence the relationship between tool use and AD performance?

A framework to view CASE tool use

To better understand how CASE tools are used in organizations today, we use a life-cycle-based model of tool usage as a basis for data collection.[33] Figure 1 presents the life-cycle model used. This model represents a vision of CASE embodied as a set of solutions packaged in a common environment working from a central repository and is similar to the IBM AD/Cycle model.[33,34] This comprehensive view serves as a way to characterize the use of CASE tools by their applicability to a specific task in the AD life cycle. For instance, tools developed to assist in the early stages of AD span planning and analysis. One example of this are the automated data flow modeling tools found in most major tool vendor product suites. These have come to be known as upper CASE tools. Lower CASE tools support the production and maintenance aspects of the AD life cycle. Additional tools such as process management and project management support the entire range of AD tasks and have come to be known as cross-life-cycle tools.

Other available frameworks depict CASE tool use from a more behaviorally oriented functional perspective.[26,35,36] A functional perspective focuses on the behavior of the *developer* rather than the features and functions of the *tool*. For this analysis we used a life-cycle model because of its easy mapping to vendor products and the familiarity of its components (e.g., enterprise modeling, analysis, testing) to information system practitioners.

To measure CASE use, we use six aggregates collected at two stages in the development of each project. The six CASE tool measures for which data have been collected are grouped into the three categories: upper CASE tools, lower CASE tools, and cross-life-cycle tools. Drawing on our life-cycle model, the enterprise modeling tools and the analysis and design tools are the key components of upper CASE tools. The build and test tools and the maintenance tools are the key components of lower CASE tools. However, maintenance tool use in this context refers to those functions used to support the current system as it is being developed. The process management and the project management tools each represent cross-life-cycle tools.

CASE tool usage

We collected data on CASE tool use at the end of analysis and design, and again at the completion of the project—at implementation. In addition, we also gathered data on the level of adherence to structured methods. Aggregated values across all projects for the CASE variables, at the two stages, are presented in Table 2. These data encompass the 57 teams and more than 500 respondents who completed and installed their systems. These data were collected using a 7-point scale. For these CASE tool usage scales, 7 represents daily use and 1 represents no use. Thus, a response of 4 represents a general average of weekly use. We asked the respondents to provide

their average level of use at the particular project stage and did not focus on total usage as a function of time. Questions were asked about the team's use of CASE tools, and the data were aggregated to team levels for this analysis.

From Table 2 it can be seen that overall usage, and usage in each category, drops from analysis and design through build and test. Second, process management and project management (cross-life-cycle activities) are relatively low at both points. Finally, the largest change between the two phases is in the use of upper CASE tools.

CASE tool usage is much lower than would be expected based on the popular literature. For instance, there occurred a relatively large use of upper CASE tools during analysis and design that dropped during build and test. The *a priori* expectation would be to see a rise in lower CASE tool use during the build and test phases of the projects.[37] This expectation is driven by the belief that CASE tools provide a way to move from automated design to automated development.[38] The low use of CASE tools and drop in use between upper and lower CASE tools may have been because the tools are perceived as inadequate for their intended purpose.[19,39] That is, CASE tool use may have been low because the teams did not feel that using the tools really helped their efforts as much as they had hoped.

CASE tool use and AD performance

As we stated above, one of the goals of the study was to determine the relationship between CASE tool use and AD performance. As others have noted, however, the relationship between AD performance measures varies considerably and is not well understood.[40–42] We do know that AD performance is a multidimensional concept with many facets, any of which may or may not be in concert with any other. In order to get a reasonably broad assessment of AD performance, we collected a variety of measures. This suite of measures provides a gauge of the effects of CASE tool use on both subjective and objective AD performance measures.

We measured AD performance both objectively and subjectively. The subjective measures include impressions of system *effectiveness* from project stakeholders (e.g., user managers and information system managers) and *satisfaction* ratings from the actual users of the systems. The two objective measures of performance are *labor cost per function point* delivered and *schedule slippage* (in percent of slip from the scheduled development time).

We collected the evaluations of key stakeholders at the time the project was implemented. Stakeholders were interviewed by the researchers about the effectiveness of the project in regard to quality, productivity, and time-to-market. Four to six months after each system became operational, system users were asked about their satisfaction with the implemented system. Schedule adherence was collected as the percent slip from the baseline schedule estimate (for example, a project that took six months to complete but was originally scheduled for four months has a 50 percent slippage). Function points were counted by the members of the research team using a common standard[43] that ensured a common basis for comparison. A function point means any collection of code that stands as a functional implementation of a requirement.[43] The total labor cost for each project was divided by the total function points sum of the project to get labor cost per function point delivered.

The first stage of our analysis explores the direct relationships between CASE tool use and AD performance. Specifically, we ask if more use of CASE tools leads to higher AD performance as measured by: stakeholders' rated effectiveness, user satisfaction ratings, labor cost per function points, and schedule slippage. In the following set of analyses we make use of various forms of ordinary least squares (OLS) regression.[44,45] We present these results in terms of which factors best predict variations in the performance measures. Table 3 summarizes our analysis of the direct relationship between using CASE tools and how much of the variance in AD performance is explained.

Table 3 shows that, of the four performance measures, CASE tool usage explained a significant amount for only two. Approximately one-fifth of the variance in schedule slippage is attributed to using lower CASE tools. This implies that using lower CASE tools increases the likelihood that the team would exceed their scheduled timetables. Further, more than half of the variance between the best and worst stakeholder-rated projects was accounted for by using both lower CASE and process management tools. No direct effects of CASE tool use to AD performance were observed for user satisfaction or labor cost per function point delivered. Thus, CASE tool use did not directly impact user satisfaction or production efficiency. However, it is reasonable to expect that the

Table 3 CASE tool use and application development performance

Application Development Performance Measure	Significant CASE Tool	Percent of Variance in Performance Explained
Stakeholder-rated effectiveness	Lower CASE tools and process management tools	55
User satisfaction	No significant tools	
Labor cost per function point	No significant tools	
Schedule slippage	Lower CASE tools	19

Table 4 Expected performance effects using CASE tools

Factor	Definition	Application Development Performance Expectation
Tool training	Amount of specific training on tools	Expected to be related to higher levels of CASE use and higher levels of performance
Structured methods	Amount of use	More use expected to lead to higher levels of performance
Group coordination	Amount of reliance on other team members	More coordination expected to lead to higher levels of tool use and higher levels of performance
Design quality	Stakeholder evaluation of design quality	Expected to lead to higher levels of tool use and higher levels of performance
Project size	Number of function points	Expected to lead to higher levels of tool use and higher levels of performance

overall impact of CASE tool use on AD performance may be influenced by other factors such as the use of structured methods, training, and group or project characteristics. We explore the effect of these factors in the next section.

Key factors affecting the relationship between CASE tool use and performance

We believe, as do others, that there are certain factors that may affect the impact of CASE tool use on AD performance.[14,17,26,35,46,47] Based on our research models, these include factors about the project (such as project size and design quality), the software team (such as amount of training and level of coordination), and the management of the project (such as the use of structured methods). We investigated the impacts of five of these factors. Table 4 describes and defines these factors and the rationale for their inclusion in the analysis.

In this stage of analysis, for each of these potentially mitigating factors, we divided the sample by the mean value of that factor. We then compared the relationship between CASE tool use and performance for both high-value and low-value groups. The results are presented with respect to these key factors in Table 5, and we discuss them in the following subsections.

Tool training. CASE tool training refers to the amount and type of specific training the team members received. The expectation is that more tool training leads to higher levels of CASE tool use and improved AD performance.[48] The data from our sample indicate that most training is tool-specific. The general trend of tool-based CASE training is to have tool vendors or contractors perform the work. Vendor classes focus on the features of the product; thus, typically, no formal integration of the product into existing methods, for a site or an overview of how the tools are to be used at the site, are part of this training.

Table 5 Effects of key factors

Key Factor	Case Tool Use	Stakeholder-rated Effectiveness	User Satisfaction	Labor Cost Per Function Point	Schedule Slippage
High levels of tool and general training	Higher				Less slip
High levels of structured methods use		Higher		Lower cost	
High levels of group coordination	Lower		Higher	Higher cost	
High levels of design document quality	Lower	Higher			
Larger project size (measured in function points)			Higher	Higher cost	

[ ] Means no observed effect

Further, the average amount of tool-specific training is less than two days. CASE tool training is often done months before actual tool use begins. Still, team members reported an average satisfaction level with CASE tool-specific training. This suggests that CASE training is relatively similar to most of the other methodology or tool training that developers have received in the past.[49]

The results indicated that teams with higher levels of CASE tool training use the CASE tools significantly more than the less-trained teams. Furthermore, teams with more CASE-tool training receive higher ratings from the user population. However, for these highly trained teams stakeholder-rated effectiveness is significantly lower and schedule slippage is significantly greater. This confusing mixture of effects can be made more understandable if overall AD training (for methods and for project management) is included as well. Table 5 shows that for high levels of training that include both CASE tool-specific training and general AD training, *higher* levels of CASE tool use are related to *less* schedule slippage and improved user satisfaction. There is no negative effect on stakeholder-rated effectiveness. This is a strong message to support general AD training in conjunction with CASE tool-specific training.

Structured methods use. Structured methods use is defined as the amount of use of key structured methods for each team. On the average, structured methods use is considered moderately important. Using structured methods has been advocated as a key factor in improving AD performance,[47] and CASE tools are one way of implementing structured methods.[38] By splitting the sample into a set of two groups (one of teams with high levels of structured methods use, the other of teams with low levels of structured methods use) a more detailed picture of the impact of structured methods use is possible. From this analysis, for teams with higher than the average use of structured methods, use of CASE accounts for 16 percent of the variance in the labor cost per function point. In this same sample CASE use accounts for 55 percent of the variance in stakeholder ratings of project effectiveness. There are no significant relationships between CASE tool use and user satisfaction or schedule slippage for high levels of structured methods use. However, lower levels of structured methods use result in lower levels of user satisfaction and a higher labor cost per function point. What is clear from the data is that using structured methods influences the amount of CASE use and enhances aspects of AD performance.

Group coordination. Group coordination is the extent to which team members share information and make efforts to work together. Research indicates that higher levels of group coordination should be

related to higher levels of performance.[28,31] Overall, teams in this sample exhibit high levels of group coordination. These groups were also relatively similar in this behavior, as the variance in the level of group coordination across these teams was small.

However, splitting the sample on the mean level of coordination shows sizable differences in both CASE use and the relationship between CASE tool use and some of the AD performance measures. Teams with higher levels of group coordination use CASE tools less and the labor cost per function point is higher. However, user satisfaction is higher for groups with higher levels of coordination. Teams with less group coordination use CASE more, have lower labor cost per function point and lower user satisfaction. For teams with low levels of group coordination, using project management tools helps to explain the variance in the labor cost per function point. Higher levels of project management tool use, especially early in the project, help to explain nearly 25 percent of the variance in labor costs per function point. Finally, CASE tool use is related to schedule slippage for both levels of group coordination. This finding suggests that CASE tool use does not provide the time savings that the trade press reported.[50]

Design document quality. Design document quality is defined as the level of design quality as reflected in the design document. We focused on the design document because it is the primary artifact of the requirements and design stage. This document typically represents both the requirements as agreed upon and the design developed to meet those requirements. Thus, the quality of the design document reflects both the user needs and the developer's plan to meet those needs. We expected that for higher levels of design document quality, CASE tool use would be up and related to AD performance.[12]

Splitting the sample into two groups—high and low levels of design document quality—indicates that higher levels of design document quality are related to lower levels of CASE tool use. This may be an instance where the backlash against CASE is apparent. It may be that the teams themselves are not convinced of the necessity to use the tools.[51] Yet, stakeholders are happier with the projects that used the tools—even though they have no knowledge of CASE tools or CASE tool usage.

Project size. Project size is calculated as the total number of adjusted function points. Function points for each project were counted by a trained group of researchers using the International Function Point Users Group (IFPUG) 3.2 standard. This was done to provide a common basis for comparison across projects. Projects ranged in size, with the mean project having approximately 2600 function points. However, two distinct groups of projects existed, those smaller than 675 function points, and all others.

Using the 675 function point level as a way to split the sample, a number of interesting differences emerge. Smaller projects use less CASE, have a lower labor cost per function point, and have lower levels of user satisfaction. This result may indicate that using CASE tools for small projects is not warranted. For larger projects, increased use of CASE, specifically lower CASE and project management tools, are related to higher levels of user satisfaction. However, larger projects have higher labor costs per function point. Larger projects demand more coordination and management, which increases costs. Large software development projects are still very difficult to manage and to successfully complete. However, CASE tool use in larger projects relates to improved user satisfaction.

Performance, CASE tool use, and structured methods

Finally, just as certainly as the silver bullet does not exist, we find that the adage " . . . a fool with a tool is still a fool" is meaningful in the AD context. What this means to us is that most CASE tools automate the techniques that are already in existence in these companies. Further, many CASE tools enforce standards and conventions that are not in place at these same organizations. It is these issues (enforcing unused structure and using automation to force a process that is otherwise done in a different way) that may be problematic. For instance, a data flow diagram done by hand at one site may use different conventions and make different assumptions than its CASE-based and automated replacement. These issues are often not well documented in the existing AD method, or explicitly mentioned in the CASE tools being used to enhance the present AD method.[35] Thus, CASE tools may not be the change agent they were slated to be.

The next set of analyses focuses on AD teams that had high levels of structured methods use and high levels of CASE use. Our observations, and existing research, suggest that a combination of following structured methods and using CASE tools should lead to improved AD performance. In the overall set of

Table 6 Application development performance

Application Development Performance Measure	High Levels of Use (Important Predictors)	Percent of Variance in Performance Explained
Stakeholder-rated effectiveness	Upper CASE and project management tools and structured methods	51
User satisfaction	CASE tools (all elements) and structured methods	Not significant
Labor cost per function point	CASE tools (all elements) and structured methods	5
Schedule slippage	Upper CASE and project management tools and structured methods	58

AD teams, structured methods use overall averaged moderately high. Measurement of structured methods use is based on measuring the number of AD techniques used in developing the present project. For the respondents in our sample, structured methods use seemed to be both valuable and important. The level of CASE use also tended to reflect the split of structured methods use. Higher usage of structured methods is related to higher levels of CASE tool use.

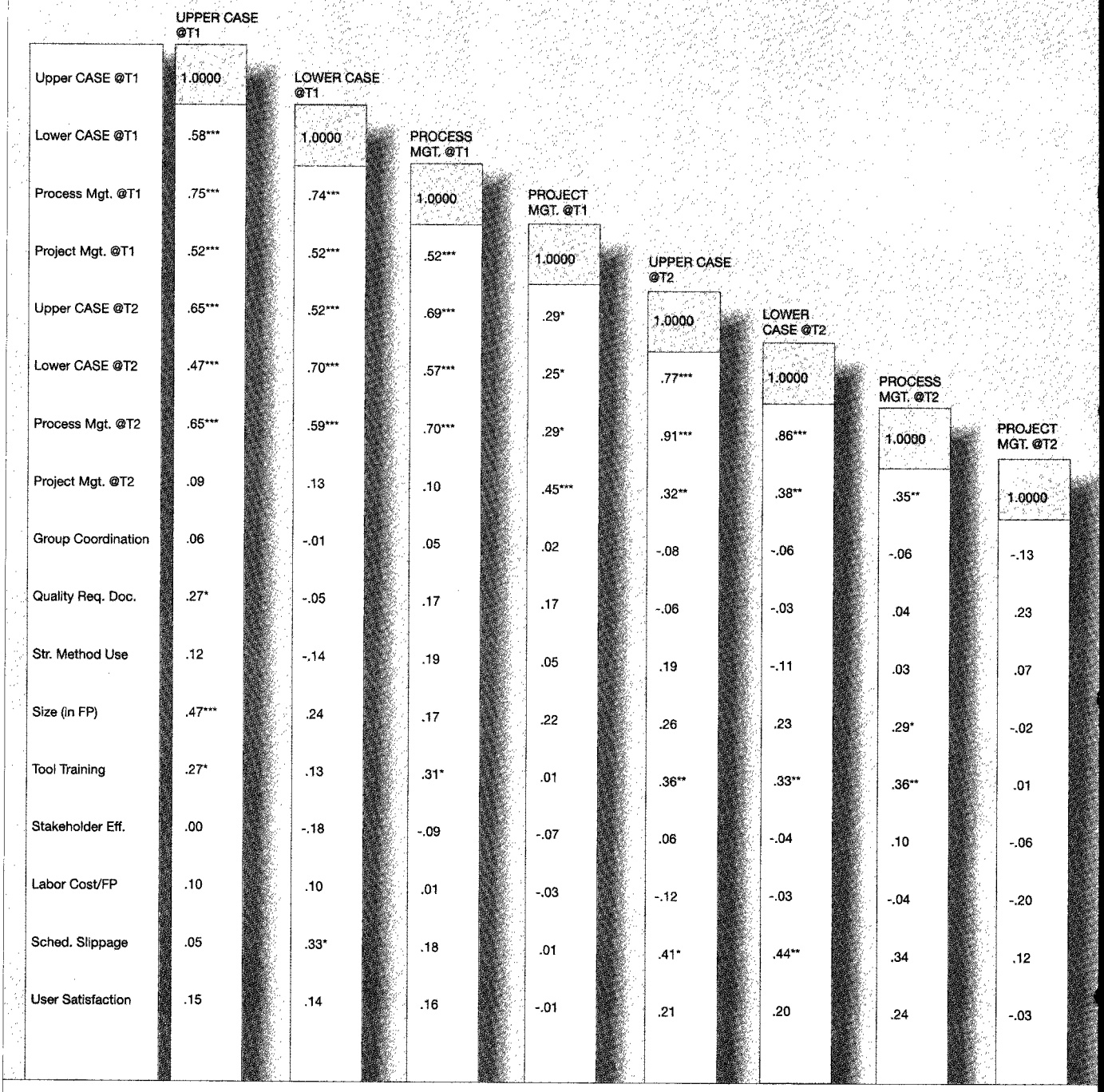
To conduct this analysis we related the variation in performance (for the four performance measures) to the CASE usage variables for the teams that were both high CASE users and high structured methods adherents. Table 6 summarizes these results. For each model, eight measures (the four CASE use measures at the two times that we collected these data for each project) are used to predict performance. Three measures (upper CASE tool use, project management tool use, and structured methods use) account for 51 percent of the variance between high and low levels for stakeholder-rated effectiveness. However, these same three elements account for 58 percent of the variance in schedule slippage. In other words, higher levels of CASE tool usage in combination with increased use of structured methods increases the likelihood that the project stakeholder will be pleased with the system. However, it will take longer to finish the project than originally estimated. This supports the common wisdom that initially project teams with CASE tools might run over schedule, but the end result will better meet the needs of the customer. It also points to the difficulty in the developers' ability to judge project time-to-market when CASE tools are involved. Perhaps due to the unrealistic expectation that the tools would drastically improve productivity, developers underestimated schedule completion time in projects using CASE extensively.

The other two models are not significant. This suggests that the combination of CASE tools and structured methods use do not drive the labor cost per function point or user satisfaction. However, the use of CASE tools later (during build and test) seems more important as a predictor of these outcomes: significant but accounting for less than 5 percent of variance. The relative lack of power of existing lower CASE tools to support the latter stages of the AD process may be one contributing reason to this lack of CASE effect. It may also be that other intervening factors have not been accounted for.

The analyses indicate that for the teams who employed structured methods and used CASE to the greatest levels, these measures predicted more than 50 percent of the variance in two key measures of performance. That is, for those teams that were taking advantage of structured methods for development, CASE tools enhanced their productivity. Conversely, in analyzing the teams that used few structured methods and made use of CASE, the data are not statistically significant. However, several trends emerge. First, CASE use in these teams falls dramatically between the analysis and design phase and the build and test phase. Second, stakeholder ratings of effectiveness are much different, as are labor costs per function point. Teams using fewer structured methods are rated lower and cost more. This suggests that CASE tool use may be a magnifier: for teams with well-structured processes, CASE use enhances the process and improves performance. For those teams with *ad hoc* processes, CASE tool use apparently abets chaos.

In comparing the findings presented in Table 4 to those presented in Table 2, additional insights can be mentioned. For example, the direct relationship between CASE tool use and AD performance may be misleading. More detailed analysis of the interven-

Major Variable Relationships as Pearson Zero-Order Correlations (two-tailed significance)



	UPPER CASE @T1	LOWER CASE @T1	PROCESS MGT. @T1	PROJECT MGT. @T1	UPPER CASE @T2	LOWER CASE @T2	PROCESS MGT. @T2	PROJECT MGT. @T2
Upper CASE @T1	1.0000							
Lower CASE @T1	.58***	1.0000						
Process Mgt. @T1	.75***	.74***	1.0000					
Project Mgt. @T1	.52***	.52***	.52***	1.0000				
Upper CASE @T2	.65***	.52***	.69***	.29*	1.0000			
Lower CASE @T2	.47***	.70***	.57***	.25*	.77***	1.0000		
Process Mgt. @T2	.65***	.59***	.70***	.29*	.91***	.86***	1.0000	
Project Mgt. @T2	.09	.13	.10	.45***	.32**	.38**	.35**	1.0000
Group Coordination	.06	-.01	.05	.02	-.08	-.06	-.06	-.13
Quality Req. Doc.	.27*	-.05	.17	.17	-.06	-.03	.04	.23
Str. Method Use	.12	-.14	.19	.05	.19	-.11	.03	.07
Size (in FP)	.47***	.24	.17	.22	.26	.23	.29*	-.02
Tool Training	.27*	.13	.31*	.01	.36**	.33**	.36**	.01
Stakeholder Eff.	.00	-.18	-.09	-.07	.06	-.04	.10	-.06
Labor Cost/FP	.10	.10	.01	-.03	-.12	-.03	-.04	-.20
Sched. Slippage	.05	.33*	.18	.01	.41*	.44**	.34	.12
User Satisfaction	.15	.14	.16	-.01	.21	.20	.24	-.03

* $p \leq .05$
** $p \leq .01$
*** $p \leq .001$

This table provides the Pearson product moment correlation coefficients and two-tailed significance levels for relationships between the factors used in this study. Since the matrix is identical across the diagonal, only the lower part is produced. Thus, the order of the columns is identical to the order of the rows. The intersection of columns and rows is the correlation of the two factors. The first correlation is between upper CASE@T1 (the first column entry) and lower CASE @T1, the second row entry. The correlation is: .58. The positive number indicates that as one variable increases, so does the other. The value suggests that the correlation between the two variables is strong. For the sample size *n* here, correlations about 0.25 are considered very good, with higher being better. If the correlation approaches 1.0, the two variables are nearly identical. A value lower than 0.25 indicates that there is relatively little correlation between the variables. The three asterisks mean that the probability of a correlation this large occurring by accident is less than one in a thousand. Two asterisks mean a probability of 1 in 100, while one asterisk means a probability of 1 in 20. Finally, a correlation implies no causality. That is, if A and B are correlated, it does not mean that A causes B or B causes A. In situations where the factors may have many common causes, this is poignant. For example, the large, positive, and significant relationship between the use of lower CASE tools at T2 and increased schedule slippage implies no causality. The implication that tool use leads to schedule slip is based solely on our expectations of the two variables.

	GROUP COORDINATION	QUALITY REQ. DOC.	STR. METHOD USE	SIZE (IN FP)	TOOL TRAINING	STAKEHOLDER EFF.	LABOR COST/FP	SCHED. SLIPPAGE	USER SATISFACTION
GROUP COORDINATION	1.0000								
QUALITY REQ. DOC.	.23	1.0000							
STR. METHOD USE	.06	.24	1.0000						
SIZE (IN FP)	-.25	-.09	-.10	1.0000					
TOOL TRAINING	.10	.11	.14	.15	1.0000				
STAKEHOLDER EFF.	.22	-.12	.06	.15	-.03	1.0000			
LABOR COST/FP	.05	-.19	-.15	.25	-.04	.06	1.0000		
SCHED. SLIPPAGE	-.41*	-.34	-.29	.47**	-.22	-.22	.02	1.0000	
USER SATISFACTION	.45**	-.34*	-.00	.09	-.09	.18	.31*	.12	1.0000

SAMPLE SIZE *n* = 57
(FOR SCHEDULE SLIPPAGE, *n* = 29.
FOR LABOR COST/FP AND SIZE, *n* = 49.)

T1 = TIME AFTER PHYSICAL DESIGN AND CODING COMPLETED
T2 = TIME AFTER IMPLEMENTATION COMPLETED

ing factors, as presented in the past two sections, indicates that a better way to examine CASE is to account for the impact of certain key factors that influence the relationship between CASE use and AD performance. Hence, Table 2 can be misleading. For instance, the key relationships between schedule slippage and CASE tool use may not be early use of lower CASE tools (as predicated by the direct relationship). A more detailed analysis indicates that both structured methods use and CASE tool use (upper CASE and project management) are better predictors of performance.

Conclusion

We began this paper by wondering aloud about our industry's search for the "silver bullet that would slay the hideous AD productivity monster." We focused our attention on the most recent pretender to the throne, CASE tools—contrasting the promises of vendors with the reality, both positive and negative, from current AD research. Drawing on a four-year study on CASE use and impact on AD productivity, we complete this paper by sharing a number of insights regarding CASE tool use and its impact on software development. We use these as a basis for presenting our recommendations to managers and to software development teams.

To begin, there are two general findings from the study that are noteworthy to reiterate: first, there are a number of different ways to view the impact of CASE tools on AD performance. Other studies have not typically taken this notion into account. For example, we found that although the overall impact of CASE was positive from one point of view (for example, from the tool's ability to create systems that users are satisfied with), at the same time it had a negative impact on other performance indicators (such as schedule slippage). If the positive impacts of an AD tool are felt in areas that are less visible to application developers (who are, after all, the primary decision makers about whether a tool is successful in its use), then the developers may form negative views on the impact of the tools, even when there are other more important positive impacts that may not be immediately apparent. Unless information technology organizations find a way to address this issue in the future, it will continue to hamper the adoption of new technologies.

Second, we must have realistic expectations for the impact of AD tools. Management cannot assume that employing a tool or sets of tools will automatically decrease software time-to-market. In our study we found just the opposite to be true—systems developed with CASE tools were more likely to be delivered behind schedule. Although this may be due to the learning curve for developers or to overly high developer expectations, it is clear that expectation management is critical to finding better ways to manage and execute software development projects.

The more detailed results of the study point out that mitigating factors influence the relationship between CASE use and AD performance. For example, CASE tools and methods go hand-in-hand. While we are clearly not the first to make this claim, our study reiterates that developers using both CASE tools *and* a well-defined structured methodology were significantly more efficient in generating systems than developers using either CASE tools *or* a structured methodology independently. Similarly, future AD innovations need both parts—tools and methods—working in concert in order to see improvements in AD performance.

It may not be as necessary to use CASE tools for small projects. Larger, more complex projects may better warrant the investment in these types of technologies. Although earlier work in CASE suggested that small projects were an appropriate place to begin when using the tools (for example, pilot projects) it may now be time to make the investment in larger projects that can better support the infrastructure that is necessary when purchasing these tools.

Interestingly, teams that were highly coordinated did not feel the need to use the tools as much as did the less coordinated teams. This may be because highly coordinated teams feel that the tools are inadequate or do not support them as much as they would like. Alternatively, it might be a manifestation of the implicit design model inherent in many CASE tools: a productivity aid for the solitary developer. Most CASE tools have great limitations in their aid to teamwork and cooperation.[26,36]

We, like others, found that training is a major factor for effective tool use. The steep learning curve of new tools and methods like CASE has been conjectured to be important. The results of this study support this conjecture with an interesting caveat. When project teams were trained specifically in the use of CASE tools by the product vendors, the resulting systems were perceived of as higher quality by both project stakeholders and by the user population. However, using CASE tools actually increased the likelihood

that these well-trained project teams would exceed their projected schedules. Upon further analysis, however, if the teams received both tool-specific and more general AD training (e.g., methods and project management), then users were more satisfied and there was less slippage. Managers must make it a priority to ensure that development teams receive adequate training that encompasses both domains.

In regard to design quality (which is reflected in the design document in our study), we found that, surprisingly, higher levels of design quality are related to lower levels of CASE tool use, yet the stakeholders are happier with the systems. As was stated previously, the backlash against CASE may be affecting developers' perceptions of the tools. Management needs to make it clear to development teams, and teams need to make it clear to management, that the tools can have positive impacts if they are given the appropriate support (training, structured methods) to do their jobs.

Finally, a question can be raised about the future of AD tools. It is not clear whether any of the current AD tools on the horizon will become the silver bullet that we have waited for. It is unlikely, but only the passage of time will tell. The factors that we have found to be important will, however, continue to be important. AD management needs to take the appropriate steps now if they want to see new advances in tools and methods successfully adopted in their AD organizations. The good news is that with the appropriate steps being taken, they can greatly increase their chances of success.

Appendix: Measure definitions

CASE tool use. To measure CASE use we used four aggregates collected at two stages in the development of each project. The four CASE tool measures for which data have been collected are grouped into the three categories: (1) upper CASE, any enterprise modeling and analysis/design tools, (2) lower CASE, any build/test and maintenance tools, and (3) cross-life cycle, process and project management tools.

Data were collected at two different stages using the same 7-point scale. For these CASE tool usage scales, 7 represents daily use and 1 represents no use. Thus, a response of 4 represents an average of weekly use. We asked the respondents to provide us with their average level of use and did not focus on total usage as a function of time per day. These questions were asked about the team's use of CASE tools. Data were aggregated to team levels for analysis.

CASE tool training. CASE tool training refers to the amount and type of specific training the team members received. This is based on the mean number of days of training each team reported having.

Structured methods use. Structured methods use was defined as the amount of and importance of using structured methods for each team. We asked whether the team used (and how important this use was) 20 key techniques embodied in structured methods.

Group coordination. Group coordination is the extent to which the team shares information and makes efforts to work together.

Design document quality. Design document quality is defined as the level of design quality as captured in the design document. This is asked of the developers after they complete design.

Project size. Project size is calculated as the total number of adjusted function points. Function points for each project were counted by a trained group of the research team using a common counting standard.

Stakeholder-rated effectiveness. We collected the evaluations of key stakeholders at the time the project was implemented. Stakeholders were interviewed by the researchers about the effectiveness of the project in regard to: quality, productivity, and time-to-market.

User satisfaction. Four to six months after each system became operational, system users were asked about their satisfaction with the system.

Schedule adherence. Data were collected as the percent slippage from the baseline estimated duration.

Labor cost/function point. Cost was the total labor cost of the project and gathered from the project team leader. Functionality was determined using function points. Function points were counted by the members of the research team using IFPUG 3.2 as a common standard, which ensured a common basis for comparison. The total function points sum was divided by labor cost for each project to get a labor cost per function point delivered.

Acknowledgments

The authors would like to thank J. Nami Kaur of the IBM Corporation and David W. Stetson and Dr. Mehdi Ghods of the Boeing Company for their continued support and guidance of the research effort. This research was supported by a grant from the IBM and Boeing Corporations (92-465).

*Trademark or registered trademark of International Business Machines Corporation.

Cited references

1. F. Brooks, *The Mythical Man-Month*, Addison-Wesley Publishing Co., Reading, MA (1975).
2. W. W. Gibbs, "Software's Chronic Crisis," *Scientific American* **271**, No. 3, 72–81 (September 1994).
3. G. Gladden, "Stop the Lifecycle I Want to Get Off," *Software Engineering Notes* **7**, No. 2, 35–39 (April 1982).
4. J. Turner, "Observations on the Use of Behavioral Models in Information Systems Research and Practice," *Information and Management* **5** (1982).
5. R. Fichman and C. Kemerer, "Adoption of Software Engineering Process Innovation: The Case of Object Orientation," *Sloan Management Review* **35**, No. 2, 7–22 (Winter 1993).
6. V. J. Mercurio, B. F. Meyers, A. M. Nisbet, and G. Radin, "AD/Cycle Strategy and Architecture," *IBM Systems Journal* **29**, No. 2, 170–188 (1990).
7. V. Merlyn and J. Parkinson, *Development Effectiveness*, John Wiley & Sons, Inc., New York (1994).
8. National Research Council, "The National Research Council of Canada: Research, Programs, and Industrial Partnership," *The CASE Center Spring Conference*, Syracuse University, NY (April, 1995).
9. W. S. Humphrey, *Managing the Software Process*, Addison-Wesley Publishing Co., Reading, MA (1988).
10. R. Kraut and L. Streeter, "Coordination in Software Development," *Communications of the ACM* **38**, No. 3, 69–81 (March 1995).
11. D. Reifer, *Software Management*, IEEE Press, Los Alamitos, CA (1994).
12. D. Halloran, S. Manchester, and J. Moriarity, "System Development Quality Control," *MIS Quarterly* **22**, No. 4, 1–13 (December 1987).
13. L. Constantine, "Control of Sequence and Parallelism in Modular Programs," *AFIPS Conference Proceedings*, Spring Joint Computer Conference **32**, Silver Spring, MD (1968), pp. 409–430.
14. L. Beck and T. Perkins, "A Survey of Software Engineering Practice: Tools, Methods, and Results," *IEEE Transactions on Software Engineering* **SE-9**, No. 5, 541–561 (September 1983).
15. S. Hanson and R. Rosinsti, "Programmer Perceptions of Productivity and Programming Tools," *Communications of the ACM* **28**, No. 2, 180–189 (February 1985).
16. A. Wasserman, *Software Development Environments*, IEEE Press, New York (1982).
17. P. Lempp and R. Lauber, "What Productivity Increases to Expect from a CASE Environment: Results of a User Survey," *Proceedings of the 27th Annual Technical Symposium on Productivity: Progress, Prospects and Payoffs*, Gaithersburg, MD (June 9, 1988), pp. 13–19.
18. K. Hayley, H. Lyman, and M. Thaine, "The Realities of CASE," *Journal of Information Systems Management* **7**, No. 3, 18–23 (Summer 1990).
19. C. Bird, "CASE Crop a Flop," *Software Magazine* **11**, No. 14, 8 (November 1991).
20. J. Carey and J. Currey, "The Prototyping Conundrum," *Datamation* **35**, No. 11, 29–33 (June 1, 1989).
21. W. Orlikowski, "Division Among the Ranks: The Social Implications of CASE Tools for Systems Developers," *Proceedings of the Tenth International Conference on Information Systems*, Boston, MA (1990), pp. 199–210.
22. D. Burkhard, "The Role of Structured Methodology in the Implementation of Computer-Aided Software Engineering Technology," unpublished manuscript, McIntire School of Commerce, University of Virginia, 1988.
23. J. Ambrosio, "DuPont Brings CASE Solutions Inside—And Out," *Digital Review* **6**, No. 14, 40–43 (June 26, 1989).
24. S. Ball, "Successful Implementation of CASE," E. Chikofsky, Editor, *Advanced Papers for the First Engineering Workshop on CASE* **1**, IEEE-CS, Cambridge, MA (1987), pp. 128–138.
25. Schubert Associates, "Comprehensive Data Summary: Computer-Assisted Software Engineering: Vendor and User Issues in an Emerging Market," Schubert Associates Publication, Boston, MA (1987).
26. J. Henderson and J. Cooprider, "Dimensions of I/S Planning and Design Aids: A Functional Model of CASE Technology," *Information Systems Research* **1**, No. 3, 227–254 (January 1990).
27. M. Withey, R. Daft, and W. Cooper, "Measures of Perrow's Work Unit Technology: An Empirical Assessment and a New Scale," *Academy of Management Journal* **26**, No. 1, 45–63 (1983).
28. J. Hackman, "A Set of Methods for the Research on Work Teams," Research Program on Group Effectiveness, Yale School of Organization and Management, Technical Report #1 (December 1982).
29. J. Hackman and J. Oldham, *Work Redesign*, Addison-Wesley Publishing Co., Reading, MA (1976).
30. J. Henderson and S. Lee, "Managing I/S Design Teams: A Control Theories Perspective," *Management Science* **38**, No. 6, 757–777 (June 1992).
31. D. Gladstein, "A Normative Model of Task Group Effectiveness," *Administrative Science Quarterly* **29**, No. 10, 499–517 (1984).
32. P. Guinan, J. Cooprider, and N. Hopkins, "CASE and the Applications Development Cycle: Measuring the Value Added," *Proceedings of the International Conference on Information Systems*, Dallas, TX (December 1992).
33. R. Radice, N. Roth, A. O'Hara, Jr., and W. Ciarfella, "A Programming Process Architecture," *IBM Systems Journal* **24**, No. 2, 79–90 (1985).
34. *AD/Cycle Overview*, G320-9842, IBM Corporation (November, 1989), available through IBM branch offices.
35. I. Vessey, S. Jarvanpaa, and N. Tractinsky, "Evaluation of Vendor Products: CASE Tools as Methodology Companions," *Communications of the ACM* **35**, No. 4, 90–105 (April 1992).
36. I. Vessey and P. Sravanapudi, "CASE Tools as Collaborative Support Technologies," *Communications of the ACM* **38**, No. 1, 83–95 (January 1995).
37. J. Norman and G. Forte, "Automating the Software Development Process: CASE in the '90s," *Communications of the ACM* **35**, No. 4, 27–32 (April 1992).
38. K. Frankel, "Toward Automating the Software-Development

Cycle," *Communications of the ACM* **28**, No. 6, 578–589 (June 1985).
39. J. Bryant, "The Problems with CASE," *Systems International* **18**, No. 4, 75–76 (April 1990).
40. P. Keen, "MIS Research: Reference Disciplines and Cumulative Traditions," *Proceedings of the International Conference on Information Systems, Vol. 1*, Philadelphia, PA, ACM Press (December 1980), pp. 8–18.
41. B. Boehm, "Improving Software Productivity," *IEEE Computer* **18**, No. 9, 43–57 (September 1987).
42. W. Delone and E. McLean, "Information Systems Success: The Quest for the Dependent Variable," *Information Systems Research* **3**, No. 1, 60–95 (March 1992).
43. D. Garmus, "IFPUG Function Point Counting Practices Manual Release 3.2," Interim Revision, International Function Point Users Group, Westerville, OH (August 1991).
44. J. Cohen and P. Cohen, *Applied Multiple Regression/Correlation for the Behavioral Sciences*, Lawrence Earlbaum Associates, Inc., 365 Broadway, Hillsdale, NJ 07642 (1983).
45. E. Pedhauzer and L. Schmelkin, *Measurement, Design and Analysis*, Lawrence Earlbaum Associates, Inc., 365 Broadway, Hillsdale, NJ 07642 (1991).
46. B. Boehm, *Software Engineering Economics*, Prentice-Hall, Inc., Englewood Cliffs, NJ (1981).
47. D. Card, F. McGarry, and G. Page, "Evaluating Software Engineering Technologies," *IEEE Transactions on Software Engineering* **SE-13**, No. 7, 845–851 (July 1987).
48. R. Norman and J. Nunamaker, "CASE Productivity Perceptions of Software Engineering Professionals," *Communications of the ACM* **32**, No. 9, 1102–1108 (September 1989).
49. J. Dutton and A. Thomas, "Relating Technological Change and Learning by Doing," *Research on Technological Innovation, Management and Policy* **2**, 187–224 (1985).
50. A. F. Case, Jr., "Computer-Aided Software Engineering: Technology for Improving Software Development Productivity," *Data Base* **17**, No. 1, 35–43 (Fall 1985).
51. T. Percy, "What CASE Can't Do Yet," *Computer World* **22**, No. 25, 59–60 (June 20, 1988).

General references

W. Chung and P. J. Guinan, "Effects of Participative Management on the Performance of Software Development Teams," *Proceedings of the ACM Special Interest Group on Computer Personnel Research Conference*, Alexandria, VA (March 1994).

P. J. Guinan, "Systems Development Surprise," *Computer World* (February 12, 1996).

P. J. Guinan, J. Cooprider, and S. Faraj, "Enabling Software Development Team Performance: A Behavioral Versus Technical Approach," *Information Systems Research* (to be published).

P. J. Guinan, G. N. Hopkins, and J. G. Cooprider, "Automation of the Applications Development Life Cycle: Measuring the Value Added," *Proceedings of the International Conference on Information Systems*, Dallas, TX (1992).

S. Watts and P. J. Guinan, "Software Development Under Conditions of High Task Complexity and Ambiguity," *Proceedings of the 31st Hawaii International Conference on Systems Sciences*, Hawaii (January, 1997).

Accepted for publication December 11, 1995.

Patricia J. Guinan *Babson College, Babson Park, Massachusetts 02157 (electronic mail: guinan@hcc01.babson.edu).* Dr. Guinan is associate professor of information systems in the mathematics and science department at Babson College. She is the McDermont Term Chair recipient at Babson and conducts both applied and theoretical research in the areas of technology transfer and communication-related issues in information system design. She received her Ph.D. from Indiana University. Her research has been published in a number of journals, including *Human Communication Research*, *Business and Communication Journal*, and *Group and Organizations*. An upcoming article is planned in *Information Systems Research*. Dr. Guinan has also published an award-winning book entitled *Patterns of Excellence for IS Professionals.*

Jay G. Cooprider *Bentley College, CIS Department, 175 Forest Street, Waltham, Massachusetts 02154-4705 (electronic mail: jcooprider@bentley.edu).* Dr. Cooprider is associate professor of computer information systems at Bentley College in Waltham, Massachusetts. He received his S.B. in computer science from the Massachusetts Institute of Technology and his Ph.D. in management with an information technology specialization from the Sloan School of Management at M.I.T. He was formerly a faculty member and associate director of the information systems management program at the Graduate School of Business, University of Texas at Austin. Dr. Cooprider is the author of numerous articles dealing with management and information technology issues in such journals as *Information Systems Research*, *MIS Quarterly*, and the *Journal of Information Systems Management*. He consults widely with firms on application development technology and strategic technology applications.

Steve Sawyer *Syracuse University School of Information Studies, 4-206 Center for Science and Technology, Syracuse, New York 13244-4100 (electronic mail: ssawyer@cat.syr.edu).* Dr. Sawyer is an assistant professor at the Syracuse University school of information studies. His research focuses on how people work together and how they use information technology. Present research includes investigating how software development can be improved through attending to the social aspects of working together, and studying how people adapt to working with distributed computing applications (e.g., organizational effects of client/server computing). Dr. Sawyer received his doctorate at Boston University. He is a member of INFORMS, IEEE, ACM, and US Rowing.

Reprint Order No. G321-5638.