The Relationship between Software Project Characteristics, Case Technology, and Software Development Productivity

RONALD H. RASCH
AUBURN UNIVERSITY

ANDREW D. CUCCIA
LOUISIANA STATE UNIVERSITY

TAREK AMER
NORTHERN ARIZONA UNIVERSITY

ABSTRACT

In order to meet the increasing demands of the software development process many software development firms are either using, or considering using, Computer Aided Software Engineering (CASE) technology. CASE technology is intended to increase productivity, enhance effective communication and integrate work done during various phases of systems design. The objective of this paper is to provide information regarding the relationship between software project characteristics, CASE technology, and software development productivity. Eighty-nine software developers from three major software development organizations completed an experimental instrument designed to gather their judgments regarding CASE technology effectiveness under several software development conditions.

The results of this study identify the factors that may be relevant to firms when evaluating current and/or emerging CASE technologies for possible adoption. The software project characteristic of program complexity is especially influential on the effectiveness of CASE technology attributes. The research results could offer guidance in selecting preferred technologies as a function of characteristics of the software to be developed.

INTRODUCTION AND BACKGROUND

As the demands for software development become more complex, the limits of human cognitive processing capability may eventually be exceeded. The software development process, which has traditionally relied on the cognitive and artistic capabilities of individuals, will need to incorporate an engineering approach by introducing more formalism and standardization [9]. In addition, as the magnitude and scope of software development projects increase, a need arises to automate as much of the process as possible. Using worldwide projected software cost trends, Boehm [2] predicts that a 20 percent improvement in software productivity would be worth $90 billion worldwide in 1995. Although the need to automate the software development process exists, it is not clear that current technology can improve the productivity of software developers nor eventually affect the software development firm's bottom line profitability [14, 15].

To address these issues there is considerable effort ongoing at most software development firms to decrease the time required to develop software, while maintaining acceptable quality. A major portion of this effort involves the use of new technology and techniques. Many software development firms are either using, or considering using, Computer Aided Software Engineering (CASE) technology as part of their systems development process. CASE technology is intended to increase productivity, enhance effective communication and integrate work done during various phases of systems design [13,20]. If CASE technology can provide these capabilities, it could enhance, and possibly even change the software development process itself. Ball [1], however, noted that enterprises that had purchased the leading CASE products found that the products were primarily being used to facilitate graphical presentations for users and managers and were not being utilized to their purported full capabilities. In addition, the current use of CASE technology has met with mixed results on performance and productivity [see, for example, 11, 16, 18].

A software development firm's decision to invest in CASE technology may involve large dollar commitments

and, possibly, changes to the way software is developed. Thus there is a need to better understand the use of this technology. Software development projects vary widely with respect to complexity, schedule constraints, and end user requirements. Highly complex development projects, for example, may benefit more from the use of certain features of CASE technology than less complex projects. It may be of value to a firm to "pick and choose" specific features of CASE based on the unique aspects of each individual development project. The firm would then be in a position to "tailor" the CASE software suite of tools to be used for each specific software development project. The objective of this paper is to provide information regarding the relationship between software project characteristics, CASE technology and software development productivity. The results of this research could provide insights regarding the selection of the appropriate set of CASE tools to be used for a project and result in improved software development productivity.

RESEARCH APPROACH

A basic problem encountered when examining the effect of CASE technology on software development productivity relates to the practical resource limitations related to methodology. Industry firms cannot afford the resources (manpower, money, and material) required to perform identical, parallel software development efforts in order to evaluate the effect of alternative technologies on productivity. Likewise, academic institutions performing controlled laboratory experiments do not have the time or manpower to emulate realistic industry projects [7]. This study attempts to bridge the gap between these two extreme research methodologies by employing a behavioral decision making exercise to model software developers' judgments concerning the relationship between software project characteristics, CASE technology and productivity [12, p. 31].

The basic psychological discipline which underlies this type of behavioral decision making process builds on Brunswik's [5] lens model which portrays the decision maker as (1) being separated by the outcome of interest by space or time, (2) faced with multiple overlapping cues which are imperfect predictors of the future state, and (3) combining these cues to form a judgment. Einhorn [8] has shown that individual judgments provide reasonable predictions under the following conditions: (1) the individual must clearly understand the cues and be able to identify information from them, (2) the cues must contain information about the final judgment, or prediction, and (3) the individual must be considered an expert. The experimental conditions and research variables developed and used in this study were structured to meet these conditions. Since the attitudes of users will moderate any potential benefits to be derived from CASE technology, software development professionals' beliefs are not only proxies for the variables of interest but are important by themselves.

RESEARCH MODEL AND VARIABLES

Research Model

The research model to be examined is based on the hypothesis that software development productivity is a function of (1) the specific attributes of the available technology (CASE), and (2) the characteristics of the software development project. This model is conceptualized as:

Productivity = f (technology, project characteristics)

where the dependent variable, productivity, is defined by the measures of Quality and Efficiency. The independent technology variable is defined by specific attributes commonly associated with CASE technology. The potential effects of CASE technology on productivity are believed to vary with certain software development project characteristics. For example, some projects may be relatively simple and benefit little from CASE technology, while more sophisticated applications may readily lend themselves to the advantages inherent in new technologies. Using the software project characteristics identified by Boehm [2, 3] as product cost drivers, this paper examines the effects of the following software project characteristics on productivity: (1) program reliability, (2) program complexity and (3) scheduling constraints. The contribution of CASE technology is expected to vary across different levels of these software project characteristics. Disregard of these factors, and the potential variation of productivity enhancement across them, could lead to inappropriate conclusions as to the benefits of CASE technology.

Due to the preliminary nature of this research, there is little empirical, or theoretical, evidence available on which to state hypotheses regarding expected effects to be found by this study. Since CASE technology is being developed to aid in complex software development projects, one would expect to see the value of this technology increase as product reliability, program complexity and schedule constraints increase. There may also be some combination of these project characteristics which make CASE technology more, or less, valuable.

Dependent Variable — Productivity

Productivity can be measured in a number of different ways. In this study, it is defined to include two constructs identified by Case [6]: (1) an individual software developer's efficiency and (2) overall software product quality. It is possible that CASE technology may have the potential to allow software developers to work more efficiently (i.e., faster, more independently, and/or with fewer mistakes) yet

have no effect on the quality of the final functioning product. Likewise, programmers may not see any effects on their individual efficiency, yet task integration, improved documentation, etc. may improve the overall quality of the software product.

INDEPENDENT VARIABLE — CASE TECHNOLOGY ATTRIBUTES

One problem in examining CASE technology is reaching an agreement on what the technology is. The CASE technology umbrella covers a number of attributes, all of which cannot be expected to contribute to productivity equally [10, 14]. An examination of the effects of CASE technology without recognition of the considerable differences across products and their specific functionalities would make little practical contribution.

To address this ambiguity, this study defines specific attributes commonly associated with CASE products. This approach builds on the prior work of Hanson and Rosinski [10] and Norman and Nunamaker [14, 15] and provides the ability to address specific functions or attributes available in various CASE products. Since this approach does not focus on specific CASE products, the study results can be generalized based on CASE technology attributes as opposed to being constrained to a particular CASE software product.

INDEPENDENT VARIABLE — SOFTWARE PROJECT CHARACTERISTICS

Using the characteristics identified by Boehm [2, 3] as software product cost drivers, this research examines the effects of (1) product reliability, (2) program complexity and (3) scheduling constraints on software development productivity.

Product reliability relates to the inherent reliability required in the delivered software product. This reliability is normally associated with the "seriousness" of the impact of a software failure. The reliability of software developed to be used in an imbedded system of a United States Air Force tactical fighter aircraft, for example, would be very high since software failure could result in loss of human life and a multi-million dollar aircraft.

Program complexity relates to the technical aspects involved in the actual software program logic, structure, and coding. Program complexity could range from simple code involving a few non-nested decision and iteration operations (DO, CASE, IF, etc.) to highly complex code which requires re-entrant and recursive coding and multiple resource scheduling with dynamic, changing priorities.

Scheduling constraints relate to the time frame allowed to complete the software project. This could range from a non-rushed forty-hour work-week environment to a highly intense "crash" project with maximum overtime and heavy time pressure on software developers to complete the project as soon as possible. The contribution of CASE technology is expected to vary across different levels of these three project characteristics.

RESEARCH METHODOLOGY

A judgment-based research instrument was developed incorporating the three variables discussed above: (1) productivity, (2) CASE technology attributes, and (3) software project characteristics. This section describes the operationalization of each of these variables. A preliminary version of the instrument was pilot-tested with twelve software developers for ease of understanding, realism, time requirements, fatigue, appropriateness of attributes included, etc.[1]

None of these responses were used in the data analyzed in this paper.

Productivity

To capture the dependent variable of productivity the study participants were asked to assess the potential contribution of CASE technology to both the Quality of the final software product and to the Efficiency with which it could be developed. They were also asked to make the same evaluations for potential contributions of six different attributes associated with CASE technology (defined below). Judgments were made on a Lickert scale ranging from 1 (no contribution over related manual methods) to 7 (significant contribution over related manual methods).

CASE Technology Attributes

Table 1 lists specific attributes commonly associated with CASE products which have been identified in previous studies. These studies ranked the usefulness of certain attributes based on professionals' perceptions and grouped them based on their productivity and complexity [15], and their sophistication and their necessity [10]. Fifteen attributes based on Norman and Nunamaker's [14, 15] research were consolidated with twenty attributes identified by Hanson and Rosinski [10]. This list of attributes was circulated to twelve practitioners and academics involved with software development. Based on their responses, the original list was consolidated into six attributes that were chosen to cover as broadly as possible the general rankings and classifications previously identified by Hanson and Rosinski [10] and Norman and Nunamaker [14, 15].

[1]The software developers used for pilot testing were taken from the three software development firms that participated in the study. This iterative pilot testing process resulted in an experimental instrument that incorporated terminology and descriptions that were familiar to all participants used in this study.

Table 1

CASE Attributes Considered

Hanson and Rosinski [10]			Norman and Nunamaker [14, 15]	
1.	Interactive Debugger	a	Data Flow Diagram	1
2.	Screen Editor	a	Data Dictionary	1
3.	Subnetwork Checker	b	Analysis-Entity List	1
4.	Process Meter	b	Import/Export Faciltiy	2
5.	Configuration Manager	b	PC/Mainframe Transportable	2
6.	Process Monitor	b	LAN Support	2
7.	Storage Monitor	b	Screen/Report Design	3
8.	Stream Editor	c	Presentation Graphics	3
9.	Big File Splitter	c	Analysis-Report Writer	3
10.	Big File Scanner	c	Structure Charts	3
11.	Source Beautifier	c	Record Layout Generator	3
12.	Test Coverage Analyzer	d	Entity Relationship Model	4
13.	Auto Test Generator	d	Logical Data Model	4
14.	Print File	e	Analysis-Graph Analysis	4
15.	Data Dictionary	e	Structure Diagrams	4
16.	Source Code Control	e		
17.	Private Library	e		
18.	File Comparator	e		
19.	Program Cross Referencer	e		
20.	Display	e		

Key to groupings

Hanson and Rosinski:
a. Minimally necessary, not sophisticated
b. Necessary but not necessarily improving productivity; sophisticated; used for testing rather than fine-tuning
c. Used for fine-tuning; sophisticated
d. Used for fine-tuning; not necessary nor sophisticated
e. Neither overly sophisticated nor productivity enhancing

Norman and Nunamaker:
1. Most productive; complex modelling tool
2. Minimal productivity improvement
3. Increased productivity; simple tool
4. Increased productivity

The six attributes are (1) Reference Dictionary, (2) Resource Monitor, (3) LAN Support, (4) Interactive Debugger, (5) Project Management, and (6) Model Generation. A description of these attributes is provided in Table 2. Model Generation, for example, includes the attributes of data flow diagram, entity/relationship model, structure charts, graph analysis, and logical data model structure diagrams identified by Norman and Nunamaker. The six attributes identified were then circulated to three other senior software developers (one from each of the three firms participating in this study) for further evaluation regarding recognizability, understanding, agreement as to their uses, etc. The finalized description of these attributes was incorporated into the research instrument.

Software Project Characteristics

Scenarios were developed which described specific software products to be developed, their applications and the environment in which they were to be developed. The development of the wording for the scenarios evolved through iterative discussions with the three senior software developers (one from each firm). These scenarios incorporated the three software project characteristics of (1) product reliability, (2) program complexity, and (3) scheduling constraints. Each of these characteristics was manipulated at two levels (high and low) in an experimental design constructed to determine their moderating effects for the six CASE attributes with respect to the productivity measures of Quality and Efficiency. In this manner the study participants were able to envision a realistic design situation and had to assess the impact of CASE technology for a specific set of circumstances that captured reliability, complexity and schedule considerations.

The HIGH RELIABILITY scenario, for example, presented the following system description:

Table 2

Description of Six Attributes Used in Study

Reference dictionary — Automatic generation of a central respository for all definitions of data, and also the clearinghouse for all of the information that is associated with a given product.

Resource monitor — Inserts probes into a program to measure the time and resources spent executing particular parts of a program (e.g., how much time is spent in a particular routine).

LAN Support — The ability to be supported by Local Area Networks.

Interactive debugger — Allows incremental execution of a program, stopping it to examine parameters, variables, and data elements.

Project management — The ability to estimate, plan and track project milestones using computer-based technology and also generate graphical output (either screen or hard copy).

Model generation — The use of on-line software technology to create and maintain graphic models useful for software development (e.g., box structures, data flow diagrams, entity-relationship diagrams, finite-state machine diagrams, structure charts (Constantine), structure diagrams (Jackson), etc.).

> Your firm has recently been selected to develop the software component of a civilian air traffic controller system. This system will be installed at a large airport in a major metropolitan area. The air traffic will be very heavy with peak periods having approximately 150 aircraft in various stages of landing and takeoff patterns using dual runways. Software failure could cause collisions that result in loss of human lives and multi-million dollar aircraft.

The corresponding LOW RELIABILITY scenario was presented by the following description:

> Your firm has recently received a contract to develop the software component for the management information system of a major corporation. The system must be capable of handling a high volume of accounting-based transactions and provide on-line access for management personnel. Software failure would result only in a temporary slowdown which could be tolerated in the daily operations of the corporation.

Based on the above descriptions regarding inherent system reliability considerations, it is apparent that HIGH and LOW reliability are relative terms in this study. This was an intentional objective so that the scenarios contain as much realism as possible. The LOW reliability scenario (the development of a transaction based accounting system), however, might be thought of as requiring high reliability in other contexts.

The significant concept in study, however, is that the subjects (through their own interpretation) view the failure of an air traffic controller system as more catastrophic (in some sense) than the failure of a transaction based accounting system. This perception leads to a higher concern for software product reliability. A sample of the research instrument is presented in Attachment A where Scenario 1 contains a software design task with LOW reliability, LOW complexity, and LOW schedule constraints depicted in paragraphs one, two and three, respectively. Scenario 2 (in Attachment A) contains a design task with HIGH reliability, HIGH complexity, and HIGH schedule constraints depicted in paragraphs one, two and three, respectively. The other versions of the instrument contained combinations of these paragraphs to develop scenarios which represented all possible combinations of the three factors. (See Technical Appendix for discussion of the research design.)

Procedure

The data were gathered at three major software development organizations located in the northeast, southwest and southeast United States. These firms are involved in a wide spectrum of software development projects ranging from embedded systems on classified government contracts to telecommunications for public telephone systems and payroll systems for business organizations. Usable data from a total of 89 subjects was collected. All respondents were informed of the purpose of the study and were guaranteed anonymity. The subjects had an average of 10 years experience in software development. All three of the firms that participated in this study advocate an integrated approach to the development of software and the participants in this study were exposed to all phases of development from initial requirements definition to final system implementation. This involvement in the total software development process provided the participants with a high level of experience regarding the six technology attributes examined in this study.

ANALYSIS AND RESULTS

The experimental research design used in this study facilitated a completely-crossed three-factor ANOVA (ANalysis Of VAriance) approach to analyze the empirical data. Each of the three independent variables (product reliability, program complexity and scheduling constraints) were manipulated across scenarios at two treatment levels each (HIGH and LOW), resulting in eight different scenarios. The dependent variables were software developers' judgments of the contribution of CASE attributes to product Quality and to software developers' Efficiency. Fourteen different analyses were performed; two examining the contribution of CASE technology (overall) to program Quality and programmer Efficiency, and twelve examining the contributions of each of the six specific CASE attributes.

The results of this study provide information regarding the relationship between software project characteristics, CASE technology and productivity. Before discussing these results, however, certain limitations of this study must be noted. First, the study examines software development professionals' judgments of the effect of CASE technology on productivity rather than actual productivity changes themselves. However, since the subjects were experienced software development professionals and were given meaningful cues (consistent with the lens model paradigm), their judgments are considered to be valid proxies for what would be observed in real life. Additionally, the subjects' experience across the entire software development process adds much to the study's external validity.

Second, only six separate CASE attributes were examined directly. The results with respect to these attributes cannot be extended to all dimensions of CASE technology. These six attributes, however, were chosen so as to be as representative of CASE technology as possible. Also, subjects were asked to make an overall assessment of CASE without regard to any specific CASE attributes. In this way, their judgments regarding CASE technology as a whole, in addition to specific CASE attributes were captured. It is felt, therefore, that the results presented here represent software development professionals' judgments of the value of the technology in general in addition to the specific attributes included in the study.

Overall Influence of CASE Technology

The three software project characteristics of reliability, schedule, and complexity have significant implications regarding the overall value of CASE technology. These effects are in the expected direction, indicating that CASE technology becomes more useful as the software development task becomes more complex, requires higher reliability specifications, or is performed according to an accelerated schedule. The statistical results are presented in Table 3. The one exception indicates that the potential contribution of CASE technology to programming Efficiency is not affected by reliability considerations. Although software developers felt that the Quality of software projects with HIGH reliability requirements will be improved by CASE technology, they indicated that their individual Efficiency will not affected by the use of CASE technology. Since reliability considerations relate to the extent to which the software program will actually perform in its operating environment, the use of CASE technology (i.e., the ability to automate and integrate programming tasks that would otherwise be performed manually) would intuitively seem to be of value regarding improved programming efficiency.

Table 3

ANOVA Results for Quality and Efficiency**

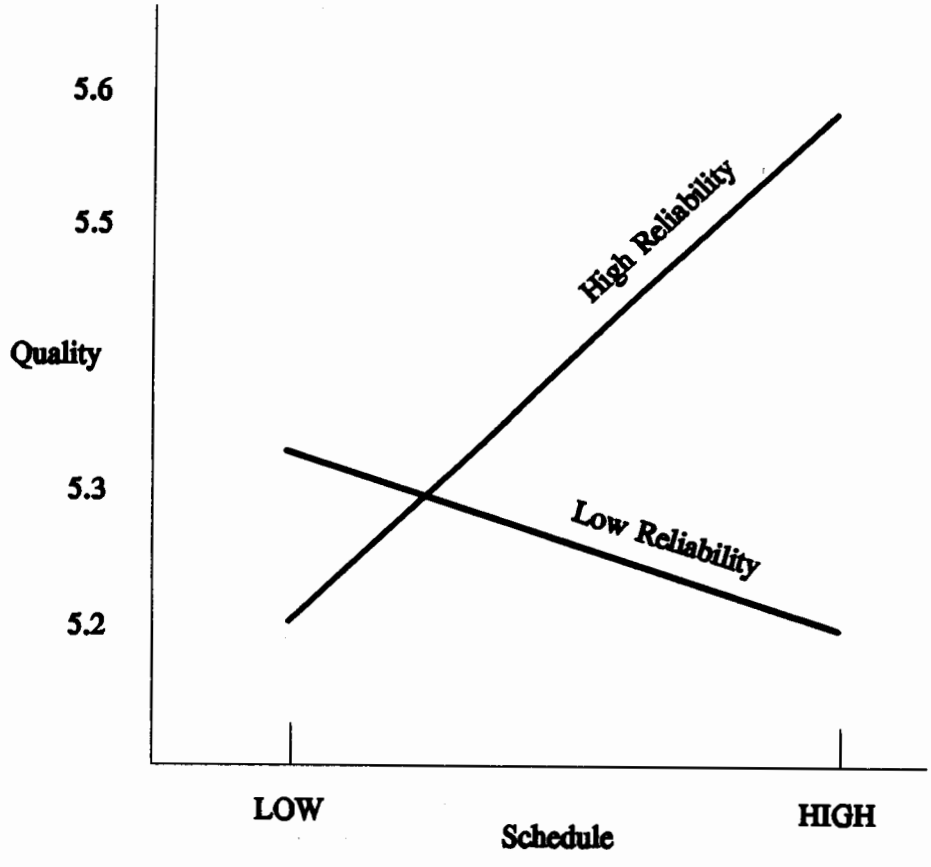
	CASE Technology Overall		Interactive Debuggers		LAN Support		Model Generation		Project Management		Resource Monitor		Reference Dictionary	
Source	**Q***	**E***	**Q**	**E**	**Q**	**E**	**Q**	**E**	**Q**	**E**	**Q**	**E**	**Q**	**E**
Reliability	.0041						.0001	.0002			.0042	.0001		.0050
Complexity	.0001	.0001	.0062	.0001	.0149	.0008	.0001	.0002	.0372	.0042	.0033	.0001	.0004	.0001
Scheduling	.0356	.0273			.0275	.0008	.0171		.0018	.0002	.0099		.0033	
RelxComp														
RelxSched			.0109											
CompxSched														

* Q and E represent the dependent variables of Quality and Efficiency respectively.

** For clarity, only p values of .05 or less are shown in this table.

Figure 1

Interaction Effect on Software Quality Using Interactive Debuggers



Influence of Individual Attributes of CASE Technology

When CASE technology was further broken down into its individual attributes, additional findings provided greater insight into the use of CASE technology. The impact of Interactive Debuggers on Quality is relatively complex due to an interaction between product reliability and schedule constraints. This interaction effect can be decomposed as shown in Figure 1. The vertical axis in Figure 1 measures the extent to which the use of Interactive Debuggers would enhance software program Quality. High reliability projects behave as expected with the value of Interactive Debuggers increasing as schedule constraints increase. For LOW reliability requirements, however, the value of using Interactive Debuggers decreases as schedule constraints increase.

The ability to function in a LAN environment improved Quality and Efficiency for software projects that contained either complex programming or accelerated schedule requirements. Neither component of productivity, however, was enhanced by LAN support based on product reliability considerations. Similar results were obtained for the Project Management feature of CASE technology. These findings indicate that product reliability characteristics have little effect on the value of providing LAN support or Project Management capabilities. Both of these CASE attributes relate more to the management control of projects than to actual technical software development tasks. Reliability, however, relates to technical performance requirements or specifications of the completed software. This finding, therefore, is logically consistent in that software developers may feel that CASE attributes aimed at project management do not enhance the technical attributes of the software.

All three project characteristics (reliability, complexity, and schedule) were found to impact the contribution of the *Model Generation* feature of CASE technology for both productivity components (Quality and Efficiency). The one exception relates to the lack of any effect of schedule on the usefulness of Model Generators regarding Efficiency. The use of on-line software technology to create and maintain

graphic models to enhance their Efficiency was not affected by schedule considerations. Quality, however, was enhanced by the use of Model Generators.

The usefulness of a *Resource Monitor* feature was also affected by all three project characteristics. The one exception relates to the lack of any effect of schedule on the usefulness of Resource Monitors regarding Efficiency. The impact of using a Reference Dictionary feature of CASE technology was also affected by all three project characteristics. One exception relates to the lack of any effect of reliability on the usefulness of a Reference Dictionary when addressing Quality. The other exception relates to the lack of any effect of schedule on the usefulness of a Reference Dictionary when addressing Efficiency.

SUMMARY AND CONCLUSIONS

This study identifies factors that may be relevant to firms when evaluating current and/or emerging CASE technologies for possible adoption. It was found that the software project characteristics of reliability, complexity, and schedule constraints affect the impact of CASE attributes on software development productivity. Most notable is the across the board influence of software project complexity on software developers' perceptions regarding the effectiveness of CASE technology attributes on the development process. This factor was strongly significant across all CASE attributes. This strong effect is as expected given that CASE technology is being developed to aid in complex software development projects. A comparatively lesser across the board effect was found for the software project characteristics of scheduling and reliability.

The interaction, or combination, effect of reliability and schedule constraints results in reduced productivity when using *Interactive Debuggers* on low reliability projects as schedule constraints become greater. This finding indicates that the use of Interactive Debuggers has less value on projects with "tight" schedules, given low reliability requirements. Similarly, projects with low schedule pressure do not benefit from the use of Interactive Debuggers if reliability requirements are high. A possible explanation for this finding is that the incremental execution of a program, stopping it to examine parameters, variables and data elements was considered more time-consuming than alternative approaches used by the software developers. These alternative approaches were not identified in this study and further research is needed to understand this counter-intuitive finding.

The ability to function in a *LAN environment* improved Quality and Efficiency for software projects that contained either complex programming or accelerated schedule requirements. Neither component of productivity, however, was enhanced by LAN support based on product reliability considerations. Similar results were obtained for the Project Management feature of CASE technology. These findings indicate that product reliability characteristics have little effect on the value of providing LAN support or project management capabilities. Both of these CASE attributes relate more to the management control of projects than to actual technical software development tasks. Reliability, however, relates to technical performance requirements or specifications of the completed software. This finding, therefore, is logically consistent in that software developers may feel that CASE attributes aimed at project management do not enhance the technical attributes of the software.

Use of the *Model Generation* feature of CASE technology resulted in increased productivity as all three project characteristics (reliability, complexity, and schedule) were increased. This result indicates that the ability to generate model abstractions enhances productivity under all project situations. This result tends to confirm anecdotal evidence that CASE developers have been presenting in the trade journals. The usefulness of a *Resource Monitor* feature and a *Reference Dictionary* feature were also affected by all three project characteristics.

The above research results offer practical guidance in selecting preferred technologies as a function of characteristics of the software to be developed. In addition, they suggest when there are preferred technologies to use across-the-board. From a more global perspective, this research advances the state of knowledge regarding the relationships between software project characteristics, CASE technology and productivity in the software development process. The methodology employed enabled assessments of both current and projected use of CASE technologies and provides a basis for future research to gain additional insight into factors affecting software development productivity. The above findings identify a number of issues related to factors that affect the productivity of software development professionals. Extension of this work could have a major impact on understanding and improving the productivity of software development.

REFERENCES

[1] Ball, S. Successful implementation of computer-aided software engineering. In E. Chikofsky (ed), *Advance Papers for the First International Workshop on Computer-Aided Software Engineering,* Vols. 1 and 2, May 27-29, 1987, Cambridge, Mass., pp. 128-138.

[2] Boehm, B. Improving software productivity, *Computer,* (September, 1987), pp. 43-57.

[3] _____. Improving software productivity. *Proceedings on The 23rd Annual IEEE Computer Society International Conference,* September, 1981a, pp. 1-16.

[4] _____. *Software Engineering Economics,* Prentice Hall, NJ, 1981b.

[5] Brunswik, E. *The Conceptual Framework of Psychol-*

ogy, University of Chicago Press, Chicago, 1952.

[6] Case, A., Jr. *Information Systems Development: Principles of Computer-Aided Software Engineering,* Prentice Hall, NJ, 1986.

[7] Dunsmore, H., Zage, W., Zage, D., and Cabral, G., Building a case for CASE. *SERC-TR-8-P,* Software Engineering Research Center, Purdue University, West Lafayette, IN, December, 1987.

[8] Einhorn, H. Expert judgement: Some necessary conditions and an example. *Journal of Applied Psychology,* 1974, Vol. 1, No. 5, pp. 562-571.

[9] Goldberg, R. Software engineering: An emerging discipline, *Systems Journal,* 25, 314, (1986), pp. 334-353.

[10] Hanson, S. and Rosinski, R. Programmer perceptions of productivity and programming tools, *Communications of the ACM,* Feb. 1985, pp. 180-189.

[11] Henderson, J. and Cooprider, J. Dimensions of I/S planning and design aids: A functional model of CASE technology, *Information Systems Research,* Sept. 1990, pp. 227-254.

[12] Libby, R. *Accounting and Human Information Processing: Theory and Applications,* Prentice-Hall, New Jersey, 1981.

[13] Lucas, H. *The Analysis, Design, and Implementation of Information Systems,* 4th Ed., McGraw-Hill, New York, 1992.

[14] Norman, R. and Nunamaker, J. An empirical study of information systems professionals' perceptions of CASE technology. *Proceedings of the Ninth International Conference on Information Systems,* Nov. 30-Dec. 3, 1988, Minneapolis, Minnesota, pp.111-118.

[15] Norman, R. and Nunamaker, J. CASE productivity perceptions of software engineering professionals. *Communications of the ACM,* Sep, 1989, pp. 1102-1108.

[16] Orlikowski, W. CASE tools as organizational change: Investigating incremental and radical changes in systems development. *MIS Quarterly,* Sept. 1993, pp. 309-340.

[17] Rasch, R. and Tosi, H. Factors affecting software developers' performance: An integrated approach. *MIS Quarterly,* Sept. 1992, pp. 395-413.

[18] Vessey, I., Jarvenpaa, S. and Tractinsky, N. Evaluation of vendor products: CASE tools as methodology companions. *Communications of the ACM,* Apr. 1992, pp. 90-105.

[19] Wagenaar, W. Note on the construction of digram-balanced latin squares, *Psychological Bulletin,* Vol. 72, No. 6, 1969, pp. 384-386.

[20] Wilkinson, J. *Accounting Information Systems: Essential Concepts and Applications,* John Wiley & Sons, New York, 1993.

[21] Winer, B. *Statistical Principles in Experimental Design,* McGraw-Hill, 1971.

ATTACHMENT A — Samples of the Experimental Instrument

Scenario 1

Your firm has recently received a contract to develop the software component for the management information system of a major corporation. The system must be capable of handling a high volume of accounting-based transactions and provide on-line access for management personnel. Software failure would result only in a temporary slowdown which could be tolerated in the daily operations of the corporation.

Based on past work done in this area, your firm has indicated that this project's software development will be relatively simple involving code with a few non-nested decision and iteration operations (DO, CASE, IF, etc.)

This project has a well-defined scope and is expected to be completed during working routine hours (no overtime).

Scenario 2

Your firm has recently been selected to develop the software component of a civilian air traffic controller system. This system will be installed at a large airport in a major metropolitan area. The air traffic will be very heavy with peak periods having approximately 150 aircraft in various stages of landing and takeoff patterns using dual runways. Software failure could cause collisions that result in loss of human lives and multi-million dollar aircraft.

Due to the unique nature of the customer's needs (to be explained in detail at a future meeting), this project's software development will be highly complex and will require re-entrant and recursive coding and multiple resource scheduling with dynamic, changing priorities.

Due to delays in contract negotiations, the software development time schedule will be very tight. To complete the project on schedule, it is estimated that 10-20 hours of overtime per week will be required for all personnel involved with the project.

TECHNICAL APPENDIX

Research Design

In order to control for the many possible individual differences between the subjects (e.g., ability, experience, past projects etc.) and to prevent such differences from affecting the analyses, a repeated measures experimental design was used.[1] Winer [21] indicates that the primary purpose of a repeated measures design is to provide a control on differences between subjects. A fully within-subjects design, however, with each subject receiving all combinations of the three independent software project characteristics would require a subject to make assessments for eight separate reliability/complexity/scheduling combinations (i.e., three factors at two levels each). To minimize the difficulty of the experimental task and to prevent subjects from potentially losing interest and not devoting their full attention to each judgment, a one-half replication was used. Subjects were randomly assigned to two balanced groups. The subjects in each group made assessments regarding four different scenarios. These four scenarios included all levels of each variable, but not all combinations. In this way all main effects and two-factor interactions were still examined within-subjects, controlling for individual differences across subjects, while each subject need only receive four scenarios [21, p. 635].[2] It is believed that the similarity of the judgments made with respect to each of four cases prevented the task from being overly demanding and that a within subjects design made the manipulations more salient while controlling for any effects of between-subject variance.[3]

Initial Analysis

A number of statistical analyses were carried out to examine issues related to the administration and integrity of the research experiment. First, there were no statistically significant differences in the responses across the three firms represented in the sample. Subject responses were, therefore, combined across the three firms for the subsequent analyses which are reported below. This finding also provides evidence for the generalization of the results across at least the three firm environments represented in the sample.

Second, subjects responded to post experimental questions regarding their judgments of the two treatment levels for each of the software project characteristic factors: product reliability, program complexity, and scheduling constraints. The software developers judged all the intended HIGH treatment levels to be significantly higher than the intended LOW treatment levels for each factor ($p<.0001$). This treatment level manipulation check provides assurance that the treatment manipulations were successful in the experimental materials used by the software developers.

Overall Influence of CASE Technology

All three software project characteristics examined were found to moderate the contribution of CASE Technology Overall for both productivity components (Quality and Efficiency). The statistical results are presented in Table 3. This supports the prediction of an influence of software project characteristics on productivity and also supports anecdotal claims made by CASE manufacturers regarding its positive impact on productivity. The one exception relates to the lack of any moderating effect of software project reliability on the use of CASE Technology Overall regarding Efficiency. There were no significant interaction effects regarding the overall effect of using CASE technology.

Interactive Debuggers

Interactive debuggers were judged to improve Quality more for projects that had HIGH complexity ($p=.0062$) characteristics. There were also effects on Quality due to a combination effect when reliability and schedule characteristics were considered together ($p=.0109$). This interaction effect indicates a more complex phenomenon regarding the impact of Interactive Debuggers on Quality. Interactive Debuggers, further, result in improved Efficiency when used on projects with HIGH complexity ($p=.0001$) specifications.

LAN Environment and Project Management

The ability to function in a LAN environment improved Quality and Efficiency more in both complex ($p=.0149$ and $p=.0008$) and accelerated schedule ($p=.0275$ and $p=.0008$) projects. Product reliability, however, did not moderate the contribution of LAN support for either component of productivity. Similar results were obtained for the Project Man-

[1]It has been shown that individual personality characteristics may affect software professionals' perceptions of productivity [17].

[2]In this design, the three-factor interaction is confounded with any differences which might exist across the groups. This is not considered to be a problem, however, since subjects were randomly assigned to groups. In addition, there is no expectation of a significant three-factor interaction.

[3]The order in which the scenarios were presented to subjects was controlled to prevent confounding the manipulations with any order effects. Such might be the case if, for example, the high complexity scenario was perceived as more complex after receiving the low complexity manipulation than when received first. This was done by using a digram-balanced latin square [19] within each block, resulting in eight different sequences. Subsequent tests revealed that while the order of presentation did moderate subjects' perceptions of the strength of manipulations, it had no main effect on subjects' productivity ratings.

agement feature of CASE technology with p values ranging from .0002 to .0372. There were no significant interaction effects present for either LAN Support or Project Management attributes.

Model Generation

All three project characteristics (reliability, complexity, and schedule), however, were found to moderate the contribution of the Model Generation feature of CASE technology for both productivity components (Quality and Efficiency). The one exception relates to the lack of any moderating effect of schedule on Efficiency. Levels of statistical significance ranged from p=.0001 to p=.0171 and there were no statistically significant ($p < 0.05$) interaction effects between any of these project characteristics for the Model Generation feature.

Resource Monitor

The benefits of a Resource Monitor feature were also moderated by all three project characteristics. The one exception relates to the lack of any moderating effect of schedule on Efficiency. The existence of a statistically significant interaction effect between product reliability and complexity (p=.0137) implies a more complex phenomenon regarding the impact of Resource Monitors on Efficiency.

Reference Dictionary

The impact of using a Reference Dictionary feature of CASE technology was also moderated by all three project characteristics with two exceptions. One exception relates to the lack of any moderating effect of reliability on Quality. The other exception relates to the lack of any moderating effect of schedule on Efficiency. The existence of an interaction effect on Quality, however, indicates that the relative effect of using a Reference Dictionary may be moderated by the interaction of reliability and complexity (p=.0433).

AUTHORS' BIOGRAPHIES

Ronald H. Rasch *is an Associate Professor at Auburn University. Dr. Rasch received his Ph.D. from The University of Texas at Austin and was a Management Information Systems Consultant and Designer for the United States Air Force. His current research interests focus on increasing productivity in the Information System development process and determining information system requirements during the system analysis and design process. His research has been published in a wide variety of journals including* Auditing: A Journal of Practice and Theory, IEEE Transactions on Systems, Man and Cybernetics, The Journal of Information Systems, *and* MIS Quarterly.

Tarek Amer *is an Assistant Professor at Northern Arizona University. Dr. Amer received his Ph.D. from The Ohio State University and his current research focuses on the behavioral aspects that affect information system design. His research has been published in a wide variety of journals including* Auditing: A Journal of Practice and Theory *and* The Journal of Information Systems.

Andrew D. Cuccia *is an Assistant Professor at Louisiana State University. Dr. Cuccia recently received his Ph.D. from the University of Florida where he investigated the behavioral models inherent in developing and using accounting information systems.*