CASE TECHNOLOGY AS A MEDIATING FACTOR IN ANALYST AND PROGRAMMER JOB OUTCOMES

by

Gregory E. Truman
Information Systems Department
Leonard N. Stern School of Business
New York University
New York, New York 10003

January 1992

Center for Research on Information Systems
Information Systems Department
Leonard N. Stern School of Business
New York University

Working Paper Series

STERN IS-92-6

Table of Contents

I. Introduction

The performance of software engineering groups has been censured by individuals, from both within the ranks of data processing personnel and those they support. End-users are unsettled as they await substantial time periods for delivery of their application systems. As a specific example consider the following: several years ago a manufacturer of paperboard products reported a three year turnaround on large development projects--two years just waiting in the queue [Gremillion et al 1983]. More generally Alloway et al (1983) found demand among four classes of application systems exceeded supply by 100 to 500 percent, as they stressed the need to refocus attention from the backlog to demand in order to gain a greater appreciation of the problem from the users' perspective. Indeed the problem of unfulfilled user demand, accompanied by decreasing hardware cost and more sophisticated user interfaces, is consistently cited in the literature as providing impetus for end-user computing proliferation [Cotterman et al 1989, Benson 1983, Rockart et al 1983]. And literature describing vital concerns of IS experts or management consistently identify end-user computing growth as one leading management concern: Straub et al (1990) as evinced through identifying the importance of Human Interface Technologies; Brancheau et al (1987) and Dickson et al (1984) as stressed explicitly by stating the importance of end-user computing.

Data processing managers are even admitting and critical of their own departments' performance. According to a survey of data processing managers, the average backlog of development projects ranged from 18 to 36 months [Plaskett et al 1983]. As further evidence an empirical study

using the Delphi technique found data processing executives ranking adequate systems development response as the number one critical success factor, as they claimed frequent and sizable cost and time overruns [Martin 1982]. This testimony is supported by continuing indications of problems in systems development as an influential concern among data processing experts and managers [Straub et al 1990, Brancheau et al 1987, Dickson et al 1984].

System analysts and programmers themselves admit the excruciating slow progress on development work. The reason is palpable--their ever increasing preoccupation with software maintenance. As recently cited, consider that maintenance demands require one-half of any typical analyst/programmer's time schedule, that maintenance consumes two-thirds of the total life-cycle resource, and that maintenance may cost as much as 200 percent of the original development cost [Gibson et al 1989]. Other large-scale studies investigating maintenance burdens report similar magnitudes of disproportionate resource allocation between maintenance and development work [Jones 1986, Lientz et al 1980]. And dollar figures have been reported. Though the validity may be debated, the figures' magnitude provides an impression--over 200 billion is spent on software engineering annually [Boehm 1987], with two to three dollars contributed to maintenance for every single dollar expended on development [Gallant 1986].

The chronic demand for more and better application systems, coupled with a desire to restrain expensive data processing labor costs [Baroudi et al 1986], has forced businesses to create new methodologies and tools to improve software development productivity and quality. Research has

followed these developments as it attempts to assess the impacts these new methods and tools will have on the productivity and quality of software engineers' work [Gibson et al 1989, Harel et al 1985, Hanson et al 1985, Jones 1978, Kemerer 1987, Mahmood 1987, Srinivasan et al 1987, Vessey et al 1986], and on the job outcomes of these individuals [Baroudi et al 1986, Mahmood 1987]. Significant research endeavor has focused on the task of developing and testing adequate productivity measures as a foundation for facilitating subsequent empirical research [Jones 1978, Kemerer 1987]. Exploration into impacts of specific methods and tools on productivity, quality and job outcomes has concurrently transpired. Hanson et al (1985) found that out of 20 available tools programmers perceived interactive debuggers and screen editors as the primary contributors to improved productivity. Harel et al (1985) tested the effects of procedural and nonprocedural languages on productivity and efficiency (quality). They found procedural languages facilitate greater machine efficiency while nonprocedural languages promote greater individual productivity. Other studies examined the effect of program complexity on maintenance task performance [Gibson et al 1989], the influence of conditional logic tools (decision tables, decision trees and structured English) on programmer/analyst performance [Vessey et al 1986], and the impact of two general development methods--the traditional structured approach and the innovative prototype approach, on the outcomes of development projects [Mahmood 1987]. Approaching the research issue from an organizational level of analysis, Srinivasan et al (1987) found that organization resources, external influences on the development process and the project teams' experience levels can

influence software development quality and effectiveness. Meanwhile Baroudi et al (1986) discovered that structured design approaches increase role conflict, structured programming techniques decrease role ambiguity and fourth generation languages increase job satisfaction. And Mahmood (1987) also found that structured development and prototype approaches have varied impacts on certain affective states of the programmer and analyst. Clearly the interest among researchers in establishing relationships between software engineering productivity, quality and quality of work-life and new software engineering methods and tools is pervasive and proceeding.

II. Advent of CASE Technologies

Given the existing problems in systems development, effort to develop more powerful and sophisticated software engineering methods and tools continues. Of recent development in this domain of technological advancement is the advent of Computer Automated Software Engineering (CASE) tools. This technology has been described as the automated manifestations of previous research efforts exploring and promoting strategies for integrated software engineering processes [Normon et al 1989]. Designed to improve the productivity and quality of software engineering work by automating previously performed manual tasks, CASE provides potential to ease the current problems. Though the proliferation of CASE tools is slight as less than 10% of analysts and programmers have actually used them [Carlyle 1988], utilization is likely to expand as the anticipated benefits materialize [Gane 1988]. Further support of this trend is indicated in Necco et al (1987); they found

general consensus among data processing personnel for expanding utilization of specific automated analysis and design tools. And Straub et al (1990) solicited opinions of ten Information System experts who believed CASE will have a major, though indirect, effect on business conduct as it will facilitate faster response to application system development and maintenance requests. "CASE's value is rooted in its ability to automate the human designer and coder", one expert was quoted. In light of the collective evidence it appears that CASE will become pervasive in organizations.

Similar to the introduction of other methods and tools assisting in software engineering, CASE and its impact on productivity, quality and programmer/analyst job outcomes will likely allure ample research attention. Though there currently exists a relative paucity of CASE research in these contexts, the initial probings have begun. Normon et al (1989) has paralleled the method of [Hanson et al 1985] to ascertain programmer perceptions regarding the varied impact CASE technology components exert on productivity and quality. And Orlikowski (1989) investigated the behavioral implications of CASE technology deployment at a software consulting firm. The erection of social barriers between a technical group responsible for enhancing and supporting the CASE technology and the functional group responsible for leveraging the power of the CASE technology during application development was observed. Other behavioral repercussions resulting from CASE deployment were noticed as well. These studies exhaust, to the best of the author's knowledge, the empirical investigations of CASE technology's impact on productivity, quality and programmer/analyst job outcomes. As an effort

to expand research in this domain, and to continue in the traditional streams of research in software development, this study will assess the effect CASE tools may exert on programmers' and analysts' job outcomes.

Interest in focusing on job outcomes is borne on two rationales. First, an assessment of job outcomes contingent on CASE technology deployment is significant as turnover of data processing personnel has consistently imparted concern and attention among data processing managers [Baroudi et al 1986, Bartol 1983, Brancheau et al 1987, Ives et al 1981, Martin 1982, Rockart 1982]. As CASE mediates system development practices [Orlikowski 1989], it may possess the potential to alter the task set of programmers and analysts and the working relationships among programmers and analysts. And more generally information technology--induced changes to a job fundamentally alters the individual's relation to the task, forcing task execution into an abstract mode [Zuboff 1982]. Theoretically task alteration [Hackman et al 1980] and changes in working relationships, or role perceptions [Kahn et al 1964], may affect job outcomes such as job satisfaction. As job outcomes are related to turnover among IS personnel as shown by [Bartol 1983] and supported by Baroudi's review of relevant literature [Baroudi 1985], a link between CASE technology deployment and turnover is established.

The second rationale for interest in CASE technology influences on job outcomes is advanced by the findings of Curtis et al (1988). They cited several supportive studies and demonstrated that behavioral components generally impact software engineering productivity and quality significantly, while methods and tools have only small to moderate

influence. In light of these results dissection of the behavioral implications of CASE technology deployment may provide more efficacious extension of research, as opposed to direct assessment of CASE technology's impact on productivity and quality independent of the behavioral associations.

III. Current State of CASE Technology

The infancy of CASE technology is reflected not only in its limited exposure to programmers and analysts as indicated above, but also in its sophistication. To date there is no CASE tool that integratively supports the entire systems development process from planning through implementation [Gane 1988], which, at least theoretically, is the objective. Progress towards this goal is occurring incrementally. The vendor market currently delivers a set of tools providing fragmented support of substantial variation across systems development stages. For example Excelerator, the leading CASE tool by market share (28.9%), supports nearly every stage of the development process: analysis is aided through data flow diagram capability; design is supported through automated entity relationship diagrams, structured charts/diagrams, and state-transition diagrams among other design aids; code generation is limited to automating code akin to that of a COBOL 'Data Division', however "add-ons" can be attached for generation of rudimentary COBOL 'Procedure Division'-like code; prototyping is functional; documentation generation is highly sophisticated, evidenced by an interface to desktop publishing software; and finally project management is facilitated through an optional link to Project Management software [Gane 1988]. And

this list is not exhaustive of Excelerator functions. In contrast ER-Designer (1.5%), ANATOOL (1.6%) and Transform (0.1%) are functionally limited as they support only entity relationship diagrams (design), data flow diagrams (analysis) and code generation (programming) respectively [Gane 1988].

As further evidence regarding the varied nature of basic functionality among CASE tools, a brief inventory was taken according to the information provided by Gane (1988). Specifically prototyping, code generation, documentation generation and project management functions were considered. Of the 25 tools researched 14 contained some form of prototyping capability, 17 generated COBOL 'Data Division' code or comparable code of some other language while only 8 generated COBOL 'Procedure Division' code or something comparable, 19 supported documentation generation, and 12 yielded some type of project management assistance. And there was the tendency for a CASE tool to provide either a front-end function e.g. prototyping, or a back-end function e.g. code generation. Therefore depending on the level of granularity upon which various stages of the system development process is defined, a given CASE tool may be considered to lend support for a particular system development stage or it may not.

Limited CASE technology support may lend cause to argue against significant influence on software engineering job outcomes. However as some are relatively comprehensive e.g. Excelerator, and vendor plans are generally expansive e.g. of the 25 vendors 14 had plans for adopting other functional roles [Gane 1988], it is contended software engineers will become increasingly exposed to automated development mechanisms in

the future. Consequently consideration for focusing on CASE tools incorporating greater functionality will guide the survey cite selection decision, even though limited CASE tools may wield some effect as well. A framework for analyzing how CASE technology may impart impact on software engineers' job outcomes follows.

IV. The Bounding Effects of CASE Technology

According to Orlikowski et al (1989) CASE technologies hold the potential to exert bounding influences on systems development activity. As indicated in [Orlikowski et al 1989], three bounding effects may occur:

> Constitutional Bounding - Reflects the notion that design activity is constituted by a set of underlying assumptions, concepts, norms, interests, and values--that is, a language.
>
> Methodological Bounding - Recognizes that each CASE tool supports a different set of system design methodologies for the task it addresses.
>
> Implementation Bounding - Reflects specific constraints imposed on the design activity consequent to CASE deployment which reduces the designers' degrees of freedom with respect to the sequence of design attention, representation and manipulation of objects, interface characteristics or possible methodological "short-cuts".

These three bounding effects become manifested in the semantic and syntactic aspects of design activity a la CASE technology [Orlikowski et al 1989]. The semantic aspect will be addressed first, followed by the syntactic aspect.

The Semantic Bounding Aspect

The semantic aspect refers to the assortment of diagrams and representation symbols used to facilitate the conduct and convey the

results of development activity. The "soul" of the semantic aspect is the software development methodology to which the tool subscribes. And all CASE technologies must subscribe to a methodology as their existence necessarily embodies one. Orlikowski et al (1989) referred to the semantic aspect as the content of the CASE tool. To the extent that the design activity is constrained or influenced by the CASE tool, a semantic bounding effect transpires [Orlikowski et al 1989]. As indicated in Orlikowski et al (1989), several semantic facets may exert a bounded influence on programmers and analysts; facets centered on (1) the number of design objects offered by each tool, (2) the variety of design object types, and (3) the rarity of design objects.

As a concrete example illustrating the semantic bounding effect consider the following. A tool used during the analysis phase of a development project is the data flow diagram [Davis 1983, Gane et al 1979, Marshall 1986, Whitten et al 1989]. Indeed data flow diagrams have evolved into a highly pervasive mechanism through which systems analysis is conducted [Whitten et al 1989]. These diagrams reveal the data's origin (input), its destination (output), its interim storage area (storage), and its transformations (process) [Davis 1983, Gane et al 1979, Marshall 1986]. The data flow diagram assists in organizing masses of information, in facilitating communication with the user as meaning is embodied in concise, non-technical picture format, and in "bridging" to the design phase by conveying high-level design specifications [Davis 1983]. There are two predominate symbol sets popularly employed for data flow diagrams; each set has an exhaustive collection of symbols for conveying necessary information and each contains a symbol comparable to

one comprised in the other [Whitten et al 1989]. The two symbol sets are referred to as the Gane-Sarson Data Flow Diagram and the DeMarco-Yourdon Data Flow Diagram. As revealed in Gane (1988), of the 25 CASE products surveyed one supported the Gane-Sarson symbol set only, three supported the DeMarco-Yourdon symbol set only, six supported both sets and eight supported data flow diagrams but were unspecified as to the exact symbol set utilized. (Seven CASE tools did not support any analysis activity.) To the extent that the deployed CASE tool's faculty of data flow diagramming technique counters the developers' norm of data flow diagramming technique, a semantic bounding effect will occur. For example a CASE tool providing Gane-Sarson symbols while the developer's experience is grounded in the DeMarco-Yourdon symbol set substantiates an occurrence of semantic bounding. Clearly this is possible as four of the tools engaged only <u>one</u> of the symbol sets. As a CASE tool will likely govern the development activity of software engineering groups, these semantic bounding influences may potentially reverberate to many individuals within a single group.

As another example consider the activity of program design. Davis (1983), Gane et al (1979), Marshall (1986) and Whitten et al (1989) collectively identify the following tools or techniques for supporting program logic specification: Warnier-Orr diagrams, traditional (IBM) flowcharts, decision tables, decision trees, HIPO/IPO charts, pseudo code or structured English, and Nassi-Schneiderman diagrams. There are subtle differences among these but their general purpose--program logic specification, is the same. For example decision tables are generally more useful for illustrating <u>complex</u> decisions i.e. many alternatives

[Davis 1983, Whitten et al 1989], while decision trees are generally more useful when many levels characterize the decision [Marshall 1986]. And Vessey et al (1986) confirmed these subtle trade-offs by finding varying programmer performance levels transpiring during two psychological processes (taxonomizing and sequencing) occurring during program design; the variance resulted from the leveraging of either decision trees, decision tables or structured English. For taxonomizing decision trees provided the best performance level followed by structured English and decision tables. For sequencing decision trees and structured English facilitated comparable performance, while performance using decision tables lagged behind.

Consequently a programmer may prefer utilizing one tool or technique, depending on the nature of the program, or indeed one may be advantageous. However by operating within the jurisdiction of CASE technology, a programmer will be bounded by the program logic tool or technique made available through the CASE technology. As shown in Gane (1988) Warnier-Orr diagrams were available in three CASE tools, three employed the traditional (IBM) flowcharts, one facilitated the use of decision tables, none used decision trees, one implemented HIPO/IPO diagrams, none leveraged pseudo code or structured English, one engaged the Nassi-Schneiderman option, and finally five employed some type of customized feature for program logic specification. (Fourteen did not accommodate program logic specification or did not specify the alternative.) Most tools provided only one option, although several had two options available and one had three alternatives. The potential for constraining or influencing the programmer through activation of the

semantic bounding aspect during program logic specification exists under these conditions, as a programmer's preferred method may not be available in a given CASE environment. Further examples of the semantic bounding influences are available in Orlikowski et al (1989).

The Syntactic Bounding Aspect

The syntactic aspect of design is bounded primarily through implementation bounding effects [Orlikowski et al 1989]. Generally it refers to the formalization and standardization of development activity as necessarily imposed through tool usage [Orlikowski et al 1989]. Additionally, the ordering of work activities as prescribed by the CASE tool is another manifestation of implementation effects. As noted in [Orlikowski et al 1989]:

> The details of a specific tool implementation impose a context of use on the designer/tool user by determining the spatial and temporal conditions within which design tasks are executed.

The syntactic aspect is the <u>form</u> of the CASE tool, which contains several facets capable of exerting potential bounding effects on programmers and analysts [Orlikowski et al 1989]. These facets include the technical limitations, the extent of design assistance, the degree of integration, and the support for multiple users.

As an example to illustrate the syntactic bounding effects, consider the facet of multiple user support and the implications this has for sustaining integrity of the work activity. Each CASE tool employs a control mechanism over access to the repository containing the results of development activity. Various degrees of locking exist. Gane (1988) found three CASE tools had no locking capacity, eight employed a locking

mechanism at the entity or object level, five provided locking at the document or diagram level, two maintained locking functions at the application or database level, one locked at the release or version level, one allowed only single user capability, and for the remaining tools the locking mechanism was either not discernible or not applicable. And all locking mechanisms were instantiated either formally or informally. Formal meaning locking mechanisms were embedded in the technical design of the CASE product; informal meaning locking mechanisms were constituted through notification of developers that sharing of work objects was occurring. The latter mechanism left the burden for maintaining integrity to the developers. To the extent the locking mechanism constrains or influences the developers, a syntactic bounding effect transpires [Orlikowski et al 1989]. And the intrusiveness of the locking mechanism's effect on the development team effort will calibrate the required communication among the members during the design process [Orlikowski et al 1989]. Further examples of the syntactic bounding influences are available in [Orlikowski et al 1989].

In general the semantic and syntactic bounding effects evinced through CASE technology deployment will constrain developers by omitting any expression or process of problem resolution outside the realm of the CASE language; similarly CASE deployment will influence developers by coaxing use of contained problem solving expressions and processes [Orlikowski et al 1989]. The specific manifestations of these bounding effects is hypothesized to impact on the job characteristics and role perceptions of software engineers. Testing these effects is the essence of this study. Providing evidence to support his conjecture emerges, a

prediction regarding CASE technology's impact on job outcomes can be made as task characteristics [Hackman et al 1975] and role perceptions [Kahn et al 1964] have been shown to impact job outcomes such as job satisfaction. Before stating the hypotheses, brief descriptions of the Job Characteristics model [Hackman et al 1980] and the role perceptions model [Kahn et al 1964] are conducted, accompanied by a review of each respective models use in IS research.

V. The Hackman and Oldham Job Characteristic Model

The Hackman and Oldham Job Characteristics model specifies three psychological states that, when obtained, lead to positive influences on job outcomes. These states manifest themselves in a feeling of meaningfulness from work, a sense of responsibility for work outcomes and an obtainment of knowledge regarding work activity. All psychological states are necessary conditions to instantiate a positive influence on job outcomes. Each state is in turn influenced by one or more job characteristics as described along several dimensions. These dimensions--skill variety, task identity, task significance, autonomy and feedback, are the specific characteristics to be examined for change consequent to the impact of CASE technology's semantic and syntactic bounding influences.

Job Characteristics

Listed below are the specific job characteristics identified by Hackman and Oldham as influencing psychological states [Hackman et al 1980].

Skill Variety: The degree to which a job requires a variety of different activities in carrying out the work, involving the use of a number of different skills and talents of the person.

Task Identity: The degree to which a job requires completion of a "whole" and identifiable piece of work; that is, doing a job from beginning to end with a visible outcome.

Task Significance: The degree to which the job has a substantial impact on the lives of other people, whether those people are in the immediate organization or in the world at large.

Autonomy: The degree to which the job provides substantial freedom, independence, and discretion to the individual in scheduling the work and in determining the procedures to be used in carrying it out.

Job Feedback: The degree to which carrying out the work activities required by the job provides the individual with direct and clear information about the effectiveness of his or her performance.

The first three characteristics contribute to one psychological state in disjunctive form. Specifically the presence of either skill variety, task identity or task significance will allow for a feeling of meaningfulness to emerge from work activity. The presence of autonomy will allow for a sense of responsibility for outcomes of work effort, while the presence of job feedback will enable the accumulation of knowledge regarding the results of work activities. Together the three substantiate the motivating potential of the individual as alluded to above. Additionally, the model has some verified moderating variables which are discussed below.

Job Characteristic Model Moderators

People are different. They respond to similar situations and routines in various fashions as they experience different feelings. Several factors, intrinsic to the individual, have been recognized as moderating the resulting motivating potential and job outcomes as changes

in job characteristics occur [Hackman 1980]. The first moderating variable is knowledge and skill. Assuming a job is rated highly on all job characteristics, the individual's level of knowledge and skill will influence the job outcome. (This assumption is necessary as no influence on job outcomes transpires when job characteristics are rated low.) Substantive knowledge and skill necessary for the job will facilitate the individual's ability to perform well, thereby allowing positive job outcomes to follow. The individual's perception regarding the relevancy of his knowledge and skill in performing his job must therefore be recorded and controlled for. Given the context of the research problem posed here, this moderating variable has potential for considerable effect. CASE tools are a relatively innovative technology, therefore the potential for programmers and analysts to feel inadequate in knowledge and skill with the tool is possibly quite high.

The other two moderating variables are 'growth need strength' and 'satisfaction level with the work context'. Growth need strength refers to the intrinsic desire for personal accomplishment; satisfaction level with the work context refers to other facets of the individual's relationship with the organization. These facets include feelings toward pay, fellow workers, physical working environment and the like. Different from the first moderator, these two will influence the relationship between job characteristics and job outcomes via an interaction effect. Concurrent high growth need strength and high satisfaction with the work situation will lead to strong influence on job outcomes from high levels of the job characteristics. Concurrent low growth need strength and low satisfaction with the work situation will

lead to weak or no influence on job outcomes from high levels of the job characteristics. Concurrently mixed levels of the two moderators will cause a moderate influence on job outcomes. Though the context of the research question does not lend enhanced likelihood of these moderating effects occurring, ascertainment of them is still necessary for control purposes during subsequent interpretation of results.

The reliability and construct validity of the Job Diagnostic Survey [Hackman et al 1975]--the instrument for capturing measures of the task characteristics and moderating factors, has been confirmed by several sources. Hackman and Oldham performed exhaustive reliability and validity tests on the instrument at the time of model inception [Hackman et al 1974], and independent sources have corroborated their results [Cook et al 1981]. A review of several studies in Information Systems research employing Hackman and Oldham's model follows.

The Job Characteristics Model and IS Research

Turner (1984) focused on the task environment embodied within the amount of work demanded of workers, the degree of discretion allowed workers and the interdependence among workers as intervening factors between technology utilization and workers' attitudes and performance. The embodiment elements are, in the context of the Hackman and Oldham model, equivalent to the job characteristics identified above. The empirical results indicated both positive and negative impacts on job outcomes occurred.

Kraut et al (1989) extended on [Turner 1984]. The original model from [Turner 1984] was expanded to incorporate other elements such as

implementation strategy, organizational environment and composition of work force as moderating factors, but the "heart" of the original model was left in tact--the nature of job characteristics remained the central mediating factor between technology implementation and workers' attitudes and productivity. Again the results revealed both positive and negative impacts on job outcomes.

Cougar and Zawacki (1980) employed Hackman and Oldham's model and surveyed over 2500 data processing employees from management, development (data processing professionals) and operations. They found data processing managers possess high growth need strength and they perceived their jobs to have high motivating potential. Hence a good match exists. Data processing professionals also have high growth need strength and their jobs have high motivating potential, however variation at the organization level existed. This suggested over-specialization of jobs at some organizations may affect task characteristics and detract from the motivating potential creating a mismatch between person and job, and consequent low job satisfaction. Operations personnel were found to have growth need strength comparable to the other groups, however the motivating potential of their jobs was significantly lower. This indicated a serious mismatch between operations personnel and their jobs, suggesting a need for work redesign.

Finally Yaverbaum (1988) leveraged the Hackman and Oldham model to study the job satisfaction of end-users. She found end-users generally perceive their jobs as providing greater motivating potential than comparable workers performing their tasks without the assistance of information technology. This finding contrasted with a similar

comparison at the management level. Managers using information technology did not behold their jobs as more significant or meaningful, relative to managers not using it.

The Hackman and Oldham Job Characteristics model has been useful for assessing the impact of information technology on job outcomes, both within and outside of the ranks of data processing personnel, as evidenced by these studies. However there is a limitation associated with it. The Job Characteristics model maintains an implicit assumption of job characteristics impacting job outcomes for an individual working independently [Hackman et al 1980]. It does not capture the orthogonal dimension of working relationships that has been shown to also impact job outcomes [Kahn et al 1964]. As significant exposure of software engineers to other members of the project team and the user community occurs [Goldstein et al 1984, Goldstein 1989], the relevance of capturing the impact of working relationships on software engineers' job outcomes is established.

VI. The Role Perception Constructs

The role perception constructs capture the effects of worker interaction on job satisfaction [Kahn et al 1964]. Kahn et al (1964) discovered two specific constructs holding potential for impacting job satisfaction--role conflict and role ambiguity. Definition of each follows [Bostrom 1981].

> Role Conflict - The degree of incongruity or incompatibility in the expectations or requirements communicated to a focal person [p. 92].
>
> Role Ambiguity - The degree to which desired expectations are vague, ambiguous or unclear, thereby making it difficult for the person to fulfill the requirements (of his/her role) [p. 93].

Kahn et al (1964) identified four components of role conflict. Role ambiguity is constituted by a single component. The role conflict components follow.

(1) Person-role Conflict - The extent to which role expectations are incongruent with the orientations, standards, or values of the focal person.

(2) Intrasender Conflict - The extent to which role requirements are incompatible with the resources or capabilities of the focal person.

(3) Intersender Conflict - The extent to which role requirements or expectations from one party oppose those from one or more other parties.

(4) Role Overload - The extent to which the various role expectations communicated to the focal person exceed the amount of time available for their accomplishment.

These constructs are the specific job interaction properties to be examined for change consequent to potential impact of CASE technology's semantic and syntactic bounding influences. Several IS research studies have been conducted focusing on measurement of the role constructs; descriptions of these follow.

The Role Perception Constructs and IS Research

Bostrom (1981) found a negative correlation between software engineers' sense of role conflict and role ambiguity and their job satisfaction. (This inverse relationship was originally found by Kahn et al (1964)). More interestingly, as the software engineers' perceptions of role variable levels increased, users' satisfaction with the information system decreased. Baroudi et al (1986) tested various technological environment elements' impact on perceived role conflict and role ambiguity. They found structured design techniques correlate

positively with role conflict, while reporting to a project leader and working on innovative projects correlate negatively with it. Role ambiguity correlated negatively with only two variables--structured programming and project innovativeness. And congruent to theoretical rationale, they found role perceptions correlate negatively with job satisfaction.

Goldstein et al (1984) tested role perceptions, job characteristics and leadership variables to ascertain which set explains a greater proportion of variance in job satisfaction. They found job characteristics, role perceptions and leadership characteristics account for job satisfaction variance in decreasing order. And both role perceptions and leadership characteristics account for significant variance beyond that accounted for by job characteristics. However leadership characteristics did not account for significant variance beyond that accounted for by job characteristics and role perceptions. And finally Goldstein (1989) found that the perceived level of role ambiguity differed significantly among data processing professionals grouped into four functional areas--user support, maintenance, development-analysis, and development-programming. Using the Scheffe test he showed that development-analysis personnel experience higher role ambiguity than development-programming personnel.

The role perception constructs have been instrumental in IS research for capturing significant relationships between the worker interaction dimension and job satisfaction. This dimension, coupled with analysis of the task environment as captured by the Job Characteristics model, will more fully apprehend the dynamic factors operating on job satisfaction in

a software engineering environment with CASE technology deployment than would either alone. To this end, both dimensions will be tested for sensitivity to the semantic and syntactic bounding effects.

VII. Hypotheses

Orlikowski et al (1989) identified a semantic bounding rooted in a CASE tool's subscription to a specific systems development methodology. Methodology implies a parochial set of design tools and techniques. This may constrain or influence the programmers' and analysts' development practice as their preferred tools and techniques may be disallowed by the tool's methodology domain. Essentially CASE may narrow the set of skills and talents which they most appreciably would exercise outside a CASE environment, as indicated by quotes from CASE-users in [Orlikowski et al 1989]:

> "Tools force people to think in a certain way. We all think screens and reports. So we don't have a chance to think if things could be done a better way. Tools have definitely stopped me thinking about other ways of doing things.",

and

> "With tools we force one path, and force everyone down that path. I am not sure it's the right path, but at least it's a standardized path."

Hypothesis 1
Given the semantic bounding effects in this context, CASE will lessen skill variety as some of the individuals' skills and talents may be inhibited by the technology.

Orlikowski et al (1989) indicated automated development assistance as one manifestation of syntactic bounding. The rationale behind inclusion of automated assistance is promotion of productivity and quality; however, an undesirable consequence of this may emerge as the

tool governs work activity previously performed by the individual. As Hackman et al (1980) indicated a potential infringement on task identity by either coworkers or machines, a software engineer's ability to experience task identity leveraging CASE technology may be restrained.

Hypothesis 2
Given the syntactic bounding effects in this context, CASE will lessen a sense of task identity as the machine inherits portions of development tasks.

Orlikowski et al (1989) also disclosed CASE tools reducing the perceived level of active problem solving by forcing developers into working more abstractly. As problem solving is the substance of analytical work, a software engineer with less opportunity to solve problems may inherit a lesser sense of accomplishment and influence in his or her work; and consequently, according to Hackman et al (1980), a lower perceived task significance.

Hypothesis 3
Given the semantic bounding effects in this context, CASE will lessen a sense of task significance, as the software engineer experiences a reduced sense of accomplishment and influence.

Orlikowski et al (1989) described the semantic bounding effects as employing a prescribed systems development methodology, consequently forcing the software engineer to utilize a specific repertoire of symbols and objects provided by the CASE tool. Additionally, Orlikowski et al (1989) described a syntactic bounding feature of degree of integration. As noted in [Orlikowski et al 1989] a higher level of integration tightens control over the ordering of tasks, imposing a temporal

constraint on design activity. Consequently many developers would attempt to "trick" the tool to create the appearance of task completion so that work could proceed. Together these semantic and syntactic bounding effects may lessen autonomy as discretion in work activity and the ordering of it decreases, and as CASE imposes a reduced sense of independence.

Hypothesis 4
Given the semantic and syntactic bounding effects in these contexts, CASE will lessen a software engineer's sense of autonomy as the tool imposes a repertoire of development aids and a temporal constraint on design activity.

Some CASE technologies provide capabilities which previously evaded programmers and analysts. For example user interface prototyping was rarely performed prior to CASE for lack of a mechanism. (Here the syntactic bounding effect of CASE technology has influenced design activity by extending the developers' capabilities.) As Mahmood (1987) found increased user participation in the design process using the prototype method, this process will facilitate greater opportunity for feedback from the user community. Additionally, CASE tools are constantly performing cross-checking to enforce standards and structure in the design process [Orlikowski et al 1989]. This is a syntactic bounding feature referred to as design assistance. Any violations in data integrity for example are immediately referred to the programmer or analyst for prompt correction. Prior to CASE any threats to standards and structure remained unresolved until later stages, or perhaps never detected. The tool is essentially providing substantive feedback on a near continual basis as software engineering activity occurs, therefore

perceived levels of feedback from the development environment may rise as well.

Hypothesis 5
Given the syntactic bounding effects in these contexts, CASE will increase the level of feedback both from the user community and the technical environment.

CASE technologies are implemented in part to elicit a structured development process [Orlikowski 1988] and dictate standards to [Normon et al 1989]. Goldstein (1982) asserted that structured development processes will reduce role conflict and role ambiguity. Baroudi et al (1986) adhered to this claim by hypothesizing a reduction in both role conflict ambiguity, however they found an increase in role conflict and no impact on role ambiguity. Baroudi et al (1986) also investigated the impact of structured programming on role ambiguity. They hypothesized and found a reduction in role ambiguity as structured programming provides guidelines regarding the process of programming. Consequently the results regarding the impact of structured development processes on role perceptions are mixed. Nonetheless assuming structured techniques and standards are imposed by CASE deployment, the technology may reduce role conflict and role ambiguity given the tool's provision of technically enforceable guidelines and automated assistance.

Hypothesis 6
To the extent CASE imposes structured techniques and standards, role conflict and role ambiguity will decrease.

As support for hypotheses 1, 2, and 4 cast in the context of the semantic/syntactic framework, the hypothesized reduction for these three job characteristics concurs with Kraft's (1977) and Goldstein's (1982)

assertion that structured development methods will reduce skill variety, task identity and autonomy.

A summary of hypotheses is listed in Table 1. The overall impact on job satisfaction and motivating potential is indeterminate as there exists a competing influence of hypothesized effect among the five job characteristics and the two role constructs. The first four job characteristics are hypothesized to decrease--reduce job satisfaction, while the last one is hypothesized to increase--increase job satisfaction. The two role constructs are hypothesized to decrease--increase job satisfaction. The composite effect of these constructs on job satisfaction is consequently unknown. Accordingly no hypothesis regarding overall job satisfaction is made.

VIII. Research Methodology

To test the hypotheses, an organization with CASE technology deployment was sought out. The requirements of the organization's data processing department included (1) deployment of a CASE tool accommodating at least two stages of the development process e.g. requirements definition and analysis, analysis and design, or design and code generation, (2) at least six months of development activity leveraging CASE technology, and (3) the persistence of development activity using traditional methods.

A consulting company in the New York City area meeting the requirements was found. It has deployed a CASE tool supporting, to varying degrees, every development stage. The CASE tool incorporates prototyping capability, allowing screen and report creation as it

facilitates the designer-user dialogue. Systems analysis is supported via automated construction of data flow diagrams with explosion capability. The tool assists in the design as well, granting several mechanisms to aid the developer in conducting detailed system and program specifications. And finally a foundation for code generation is sustained through inclusion of established rules, resembling structured English and providing the "building blocks" of program construction. This brief list of functions is intended not to exhaust the complete functionality of the CASE technology, but to grant a flavor of the CASE technology's automated features at the research cite.

Research Design

The study design will contrast the subjects' opinions regarding job characteristics and role perceptions. Subjects will be placed in two groups. One group will consist of software engineers working in a CASE environment and the other group of software engineers working in a non-CASE environment. Assessment and comparison of job characteristics and role perceptions as conveyed by the subjects of these distinctive environments will allow hypotheses testing, analysis supplemented by the theoretical support provided by the Job Characteristics model and role perception constructs. Figure 1 presents a model of the research design, referred to as the Development Environment Impact model. Campbell et al (1963) refers to this design as the Posttest-Only Control Group Design; it is vulnerable to no internal sources of invalidity and only two external sources--(1) the interaction of selection and treatment and (2) the reactive arrangements.

A questionnaire will be administered to the individuals for data collection. The questionnaire administration will require approximately one hour and will be conducted according to the guidelines as indicated in [Hackman et al 1975]. All subjects will undertake the inquisition concurrently and will be guaranteed anonymity.

Subjects

Thirty software engineers constitute the pool of subjects; fifteen from each environment will be chosen at random. This will assist in removing potential confounding factors of experience levels, nature of application, and supervisor relations. More subjects were desired, however the limit of thirty was imposed by the organization.

Measures

The data collection instrument will capture data on job characteristics using an augmented Job Diagnostic Survey [Hackman et al 1975], expanded to include assessment of role perceptions utilizing scales accommodated to software engineers by Rizzo et al (1970). The number of scales for each construct follows: three scales each for skill variety, task identity, task significance and autonomy; six scales for feedback--three for feedback from the job/tool and three for feedback from agents (coworkers, users and supervisor); eight scales for all facets of role conflict[1]; and six scales for role ambiguity. Data

[1] Bostrom (1981) found role overload did not correlate significantly with programmer/analysts' job satisfaction. However as a major impetus behind CASE technology proliferation is improved productivity, this variable is maintained to test whether anticipated productivity enhancement infringes on perceptions of

covering the confounding factors as identified by Hackman et al (1980) will also be solicited to control for these effects during statistical analyses. As described above the confounding factors include possession of knowledge and skill, growth need strength and satisfaction with the work context.

Testing

T-tests will be conducted on composite scores for each variable, calculated through averaging responses across respective scales. The t-tests will compare the responses of the two groups to assess the significance. Significance will be set at $p<.05$. Two-tailed t-tests will be used to test for effects in both directions, since no empirical evidence exists indicating any unidirectional influence. Means will be used to assess the direction and magnitude of differences.

Power Analysis

A medium effect size in programmers' and analysts' perceptions is anticipated from the deployment of CASE technology. In standardized units this translates into a .50 magnitude of change [Cohen 1977]. Assuming a two-tailed t-test at the .05 alpha level and a sample size of 30, the test will provide a power level of 47 percent [Cohen 1977]. To achieve a desirable 80 percent power level a sample size of 64 would be required. Unfortunately due to the externally enjoined constraints previously mentioned, this sample size will not be possible. Table 2 reveals power levels for various sample sizes assuming a two-tailed alpha

---

role overload.

of .05 and a medium effect size [Cohen 1977].

IX. Discussion

This research study has potential to reveal specific impacts CASE technologies may impart on job characteristics and on the pattern of interactions among development team members. Assuming analyses of results indicate alterations in the perceived job characteristics and role perceptions among software engineers consequent to CASE deployment, germane suggestions for the redesign of programmers' and analysts' work routine will be forthcoming. As CASE may be construed as an "information system" to develop information systems, the introduction of new information systems (CASE technology) generally leads to changes in job design [Davis et al 1980]. For example if CASE is found to decrease skill variety as hypothesized, management may dispense expanded responsibility to counterbalance the intrusion of CASE on the software engineers' ill-utilized skill set. Or, if CASE increases the degree of perceived role ambiguity, then increased management heed to software engineer's work objectives and more guidance to their work tactics may be suitable.

The research study may render valuable information to CASE vendors as well. To the extent significant findings emerge, vendors may gain insight into specific means by which CASE tools cause dissatisfaction. Adjustment to overcome consistent impetuses toward software engineer discontent may adequately relieve the adverse consequences, conducted within the constraints of technical feasibility and without compromising the benefits a CASE technology delivers e.g. integrity control measures.

Creative thought, coupled with technical expertise, will be necessary. However resources, dedicated to leverage any forthcoming insight this study may provide, will be well expended as CASE technologies increase in sophistication and transform the landscape of software engineering.

Table 1

Hypotheses Summary

	Semantic and Syntactic influence on...	
	..respective factor.	...job satisfaction.
Skill Variety	Decrease	Decrease
Task Identity	Decrease	Decrease
Task Significance	Decrease	Decrease
Autonomy	Decrease	Decrease
Feedback	Increase	Increase
Role Conflict	Decrease	Increase
Role Ambiguity	Decrease	Increase
Overall Job Satisfaction	n/a	Indeterminate

Table 2

n	Power
25	.41
50	.70
75	.86
100	.94

REFERENCES

Alloway,R.M. and Quillard,J.A. User Managers' System Needs. MIS Quarterly 7:2 June 1983, p. 27-41.

Baroudi,J.J. The Impact of Role Variables on IS Personnel Work Attitudes and Intentions. MIS Quarterly 9:4 December 1985, p. 341-356.

Baroudi,J.J. and Ginzberg,M.J. Impact of the Technical Environment on Programmer/Analyst Job Outcomes. Communications of the ACM 29:6 June 1986, p. 546-554.

Bartol,K.M. Turnover Among Data Processing Personnel: A Causal Analysis. Communications of the ACM 26:10 October 1983, p. 807-811.

Benson,D.H. A Field Study of End User Computing: Findings and Issues. MIS Quarterly 7:4 December 1983, p. 33-45.

Boehm,B. Improving Software Productivity. Computer September 1987, p. 43-57.

Bostrom,R.P. Role Conflict and Ambiguity: Critical Variables in the User-Designer Relationship. Proceedings of the 17th Annual Computer Personnel Research Conference 1981, p. 88-112.

Brancheau,J.C. and Wetherbe,J.C. Key Issues in Information System Management. MIS Quarterly 11:1 March 1987, p. 23-36.

Campbell,D.T. and Stanley,J.C. Experimental and Quasi-Experimental Designs for Research. Houghton Mifflen Company, Boston, 1963.

Carlyle,R.E. Where Methodology Falls Short. Datamation 34:24 December 1988, p. 179-191.

Cohen,J. Statistical Power Analysis for the Behavioral Sciences. Lawrence Erlbaum Associates, London, 1977.

Cook,D. Hepworth,F. Wall,G. and Warr,W. The Experience of Work. Academic Press, London, England. 1981.

Cotterman,W.W. and Kuldeep,K. User Cube: A Taxonomy of End Users. Communications of the ACM 32:11 November 1989, p. 1313-1320.

Cougar,J.D. and Zawacki,R.A. Motivating and Managing Computer Personnel. Wiley, New York,NY. 1980.

Curtis,B. Krashner,H. and Iscoe,N. A Field Study of the Software Design Process for Large Systems. Communications of the ACM 31:11 November 1988, p. 1268-1287.

Davis,W. Tools and Techniques for Structured Systems Analysis and Design. Addison-Wesley Publishing Company, Inc. Reading, MA. 1983.

Davis,G.B. and Olson,M.H. Management Information Systems: Conceptual Foundations, Structure, and Development. Second Edition, McGraw-Hill, Inc., New York, NY. 1985. p. 354.

Dickson,G.W. Leitheiser,R.L. Wetherbe,J.C. and Nechis,M. Key Information System Issues for the 1980s. MIS Quarterly 8:3 September 1984, p. 135-154.

Gallant,J. Survey Finds Maintenance Problem Still Escalating. Computerworld, January 27, 1986.

Gane,C. and Sarson,T. Structured Systems Analysis: Tools and Techniques. Prentice-Hall, Inc. Englewood Cliffs, NJ. 1979.

Gane,C. Computer Aided Software Engineering: The Methodologies, The Products, The Future. Technical Report, Rapid System Development, Inc. 1988.

Gibson,V.R. and Senn,J.A. System Structure and Software Maintenance Performance. Communications of the ACM 32:3 March 1989, p. 347-358.

Goldstein,D.K. The Effects of Structured Development Methods on the Job Satisfaction of Programmer/Analysts: A Theoretical Model. Working Paper CISF-90. Alfred P. Sloan School of Management, MIT, Cambridge, MA. 1982.

Goldstein,D.K. and Rockart,J.F. An Examination of Work-Related

Correlates of Job Satisfaction in Programmer/Analysts. MIS Quarterly 8:2 June 1984, p. 103-115.

Goldstein,D.K. The Effects of Task Differences on the Work Satisfaction, Job Characteristics, and Role Perceptions of Programmer/Analysts. Journal of Management Information Systems 6:1 Summer 1989, p. 41-58.

Gremillion L., Pyburn P., "Breaking the Systems Development Bottleneck", Harvard Business Review, March-April 1983, p. 130-137

Hackman,J.R. and Oldham,G.R. The JDS: An instrument for the Diagnosis of Jobs and the Evaluation of Redesign Projects. Technical Report #4, Department of Administrative Sciences, Yale University, New Haven, CT. 1974.

Hackman,J.R. and Oldham,G.R. Development of the Job Diagnostic Survey. Journal of Applied Psychology 60:2 1975, p. 159-170.

Hackman,J.R. and Oldham,G.R. Work Redesign, Addison-Wesley, Reading, MA, 1980

Hanson,S.J. and Rosinski,R.R. Programmer Perceptions of Productivity and Programming Tools. Communications of the ACM 28:2 February 1985, p. 180-189.

Harel,E.C. and McLean,E.R. The Effects of Using a Nonprocedural Computer Language on Programmer Productivity. MIS Quarterly 9:2 June 1985, p. 109-120.

Ives,B. and Olson,M.H. Manager or Technician? The Nature of the IS Manager's Job. MIS Quarterly 5:4 December 1981, p. 49-63.

Jones,T.C. Measuring Programming Quality and Productivity. IBM Systems Journal 17:1 1978.

Jones,T.C. Programming Productivity: Issues for the Eighties. IEEE Computer Society Press Second Edition, Washington D.C. 1986.

Kahn,R.L. Wolfe,D. Quinn,R. Snoek,J.D. and Rosenthal,R. Organizational Stress: Studies in Role Conflict and Role Ambiguity. John Wiley and Sons, New York, NY. 1964.

Kemerer,C.F. Measurement of Software Development Productivity. Doctoral Dissertation, Carnegie-Mellon University, Pittsburgh, PA. 1987.

Kraft,P. Programmers and Managers: The Routinization of Computer Programming in the U.S. Springer-Verlag, New York, NY. 1977.

Kraut,R., Dumais,S., and Koch,S. Computerization, Productivity, and Quality of Work-life. Communications of the ACM 32:2 February 1989, p. 220-238.

Lientz,B.P. and Swanson,E.B. Software Maintenance Management. Addison-Wesley, Reading, MA 1980.

Mahmood,M.A. Systems Development Methods-A Comparative Investigation. MIS Quarterly 11:3 September 1987, p. 293-311.

Marshall,G.R. Systems Analysis and Design: Alternative Structured Approaches. Prentice-Hall, Inc. Englewood Cliffs, NJ. 1986.

Martin,E.W. Critical Success Factors of Chief MIS/DP Executives. MIS Quarterly 6:2 June 1982, p. 1-11.

Necco,C.R. Gordon,C.L. and Tsai,N.W. Systems Analysis and Design: Current Practices. MIS Quarterly 11:4 December 1987, p. 461-473.

Norman,R.J. and Nunamaker,J.F. Jr. CASE Productivity Perceptions of Software Engineering Professionals. Communications of the ACM 32:9 September 1989, p. 1102-1108.

Orlikowski,W.J. Information Technology and Post-Industrial Organizations: An Examination of the Computer-Mediation of Production Work. Ph.D Thesis, Stern School of Business, New York University, New York, NY. 1988.

Orlikowski,W.J. Division Among the Ranks: The Social Implications of CASE Tools for System Developers. Proceedings of the Tenth International Conference on Information Systems. 1989.

Orlikowski,W. and Ariav,G. The Bounding Effect of IS Design Tools: A Critical Examination of CASE Technology. Working Paper #204; Center for

Research on Information Systems; New York University. March 1989.

Plasket R., Wilneff P.,"Productivity and DP Management: Losing Control?", Journal of Systems Management, October 1983, p. 30-35

Rizzo,J. House,R. and Lirtzman,S. Role Conflict and Ambiguity in Complex Organizations. Administrative Science Quarterly 15:2 June 1970, p. 150-163.

Rockart,J.F. The Changing Role of the IS Executive: A Critical Success Factor Perspective. Sloan Management Review 24:1 Fall 1982, p. 3-13.

Rockart,J.F. and Flannery,L.S. The Management of End-User Computing. Communications of the ACM 26:10 October 1983, p. 776-784.

Srinivasan,A. and Kaiser,K.M. Relationships between Selected Organizational Factors and Systems Development. Communications of the ACM 30:6 June 1987, p. 556-562.

Straub,D.W. and Wetherbe,J.C. Information Technologies for the 1990s: An Organizational Impact Perspective. Communications of the ACM 32:11 November 1989, p. 1328-1339.

Turner,J.A. Computer Mediated Work: The Interplay Between Technology and Structured Jobs. Communications of the ACM 27:12 December 1984, p. 1210-1217.

Vessey,I. and Weber,R. Structured Tools and Conditional Logic: An Empirical Investigation. Communications of the ACM 29:1 January 1986, p. 48-57.

Whitten,J.L. Bentley,L.D. and Barlow,V.M. Systems Analysis and Design Methods. Richard D. Irwin, Inc. Boston, MA. 1989.

Yaverbaum,G.J. Critical Factors in the User Environment: An Experimental Study of Users, Organizations and Tasks. MIS Quarterly 12:1 March 1988, p. 75-88.

Zuboff,S. New Worlds of Computer Mediated Work. Harvard Business Review October-November 1982.