Fourth Generation Languages Update: Potential Unrealized

INPUT®

FOURTH GENERATION LANGUAGES UPDATE: POTENTIAL UNREALIZED

FOURTH GENERATION LANGUAGES UPDATE: POTENTIAL UNREALIZED

ABSTRACT

Controversies concerning data models, operating systems, and hardware architecture all appear relatively objective compared to the emotional arguments that have traditionally surrounded computer languages. The current claims being made for fourth generation languages are all too familiar to anyone who is aware of the history of language development, but even those who have some perspective have a self-destructive propensity to make age-old mistakes. One of the mistakes is to put a general label on an ill-defined concept; another is to promote limited solutions for universal problems.

This report positions FGLs between data base management systems and more advanced productivity improvement techniques and analyzes all current productivity improvement efforts against the projected technological environment and IBM's software strategy. The scope of this report will range from first to fourth, fifth, and future generation languages. The purpose is to bring order out of the chaos, over-simplifications, and misunderstandings which surround the subject of languages.

This report contains 54 pages, including 7 exhibits.

FOURTH GENERATION LANGUAGES UPDATE: POTENTIAL UNREALIZED

CONTENTS

FOURTH GENERATION LANGUAGES UPDATE: POTENTIAL UNREALIZED

EXHIBITS

I INTRODUCTION

A. REASONS FOR PREPARING REPORT

- Controversies concerning data models, operating systems, and hardware architectures all appear relatively objective compared to the emotional (and frequently irrational) arguments which have traditionally surrounded computer languages. The current claims being made for fourth generation languages are all too familiar to anyone who is aware of the history of language development, but even those who have some perspective have a self-destructive propensity to make age-old mistakes. One of the mistakes is to put a general label on an ill-defined concept; another is to promote limited solutions for universal problems.

- INPUT recognized the first of these mistakes in Trends and Opportunities in Fourth Generation Languages and deliberately defined fourth generation languages on a broad basis which included the potential for growth and even relabeling. Rather than use 4GL as an identifier, FGL was adopted so fourth and fifth and future could all be rolled together. The point is that languages evolve and do not lend themselves to strict classification. This is especially true for the hodge-podge of current FGLs and their obvious (and sometimes ominous) direction. This report will attempt to provide some clarity to the situation by providing a structure based on where current FGLs came from and where they are likely to flounder on some well-charted rocks of reality.

- This report will position FGLs between data base management systems and more advanced productivity improvement techniques and will analyze all current productivity improvement efforts against the projected technological environment and IBM's software strategy. This report can be viewed as both an update and extension of previous INPUT reports on FGLs.

B. SCOPE AND METHODOLOGY

- This report is the centerpiece of a set which includes Data Base Management Systems and Applications Development Techniques. The three reports are tightly integrated, and it is not recommended that any one be used without a thorough understanding of the other two; they do not stand alone. The structure and methodology employed for the series was explained in the introduction to the first report, Data Base Management Systems.

- The scope of this report remains necessarily broad because INPUT believes that languages have evolved and will continue to evolve. In the belief that "what is past is prologue," this report will range from first to fourth to fifth to future. Significant past failures and successes will be isolated with specific emphasis upon how languages have restructured and extended the market for computer products and where they have failed to do so.

- The purpose is to bring some order out of the chaos, oversimplication, and misunderstanding which surrounds the subject of languages.

II EXECUTIVE SUMMARY

- This executive summary is designed in a presentation format in order to:
  - Help the busy reader review key research findings.
  - Provide a ready-to-go executive presentation, complete with a script to facilitate group communication.
- Key points of the report are summarized in Exhibits II-1 through II-5. On the left-hand page facing each exhibit is a script explaining the contents of the exhibit.

A. TRENDS IN LANGUAGE USE

- 4GLs are breaking away from traditional uses in the years ahead.

  - Simple one-time reports will be replaced by production applications.

  - Today's "pseudo production" will become mainline transaction and batch systems.

  - Performance requirements will shift in importance from low to high.

  - Tools for the analyst/programmer will continue to be needed, but the need will expand to the entire use set.

  - Attention to programming and testing will be secondary, with companies' primary attention being placed on life cycle considerations.

- FGLs will not by constrained by 4GL definitions; rather:

  - ICONs, charts, and symbols will proliferate.

  - Procedures will be inherent in process.

  - Differentiation will be the key for expert systems.

EXHIBIT II-1

INPUT®

TRENDS IN LANGUAGE USE

- 4GLs

Ad Hoc Reporting	$\longrightarrow$ Production Applications
"Pseudo Production"	$\longrightarrow$ Transaction and Batch
Low Performance Requirements	$\longrightarrow$ High Performance Requirements
Analyst/Programmers	$\longrightarrow$ Programmers and Casual Users
Programming and Testing	$\longrightarrow$ Requirements Through Maintenance

- FGLs will not be restricted by 4GL definitions.

B. DO 4GLs CONTRIBUTE TO DSD PROBLEMS?

- INPUT has identified certain potential problems associated with Distributed Systems Development (DSD). These include:

  - Hardware/software performance.

  - Systems and data/information quality.

  - Excessive rework.

  - Conflicting results.

- The misuse of a 4GL in implementing the Motor Vehicle Registration System for New Jersey seemed to confirm INPUT's concerns. In this case:

  - Response time exceeded specifications by more than an order of magnitude with only one-fifth of the terminals up.

  - Operators could not keep up with transaction rates (even with overtime).

  - Data and information chaos resulted--police could not check registrations.

  - The result is the system must be redone at substantial expense.

EXHIBIT II-2

INPUT®

ARE 4GLs A PART OF THE PROBLEM?

4GL Misuse - Case Study

- **Response Time 5 Seconds $\rightarrow$ > 5 Minutes**
- **Unable to Process Transactions**
- **Data/Information Quality Unacceptable**
- **System Unworkable**
- **Enough Blame for Everyone - Consultant/Client/Vendor**

C. OVERRATING AND UNDERVALUING

- Credit Lady Lovelace with the profound insight that most new developments tend to be "first overrated and then undervalued."

- She made the statement about Babbage's "analytical engine" (the first computer), and it remains an especially important observation about productivity tools.

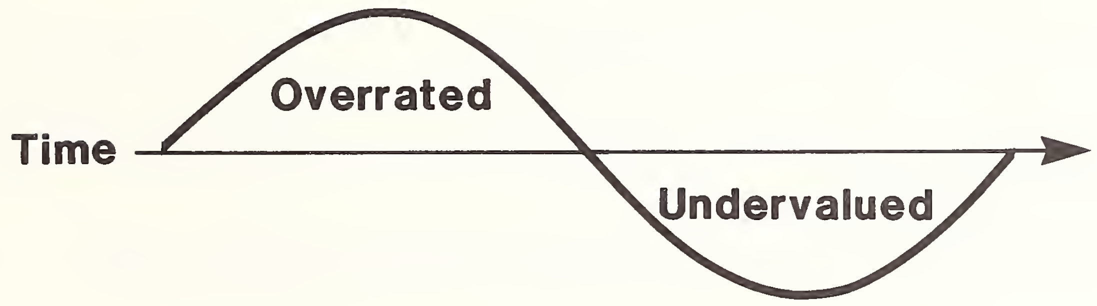
- COBOL, DBMS, and structured programming have all been promoted as the final solution to the productivity problem, and then undervalued because they failed to meet impossible objectives and expectations.

- 4GLs have peaked in the overrated phase of the cycle and now face the prospect of being undervalued.

- INPUT urges IS management to take the initiative so that this undervaluation phase is minimized and 4GLs are placed in their proper perspective.

INPUT®

OVERRATING AND UNDERVALUING



- Cobol
- DBMS
- Structured Programming
- 4GLs?

D. TAKE A BROAD VIEW OF PRODUCTIVITY

- The ability of an individual to get results at performance Level II must be weighed against possible impacts at other levels.

- A 4FGL may permit a human to generate code faster, but what is the impact on:

  - Hardware/software performance?

  - The quality of data/information available to work units?

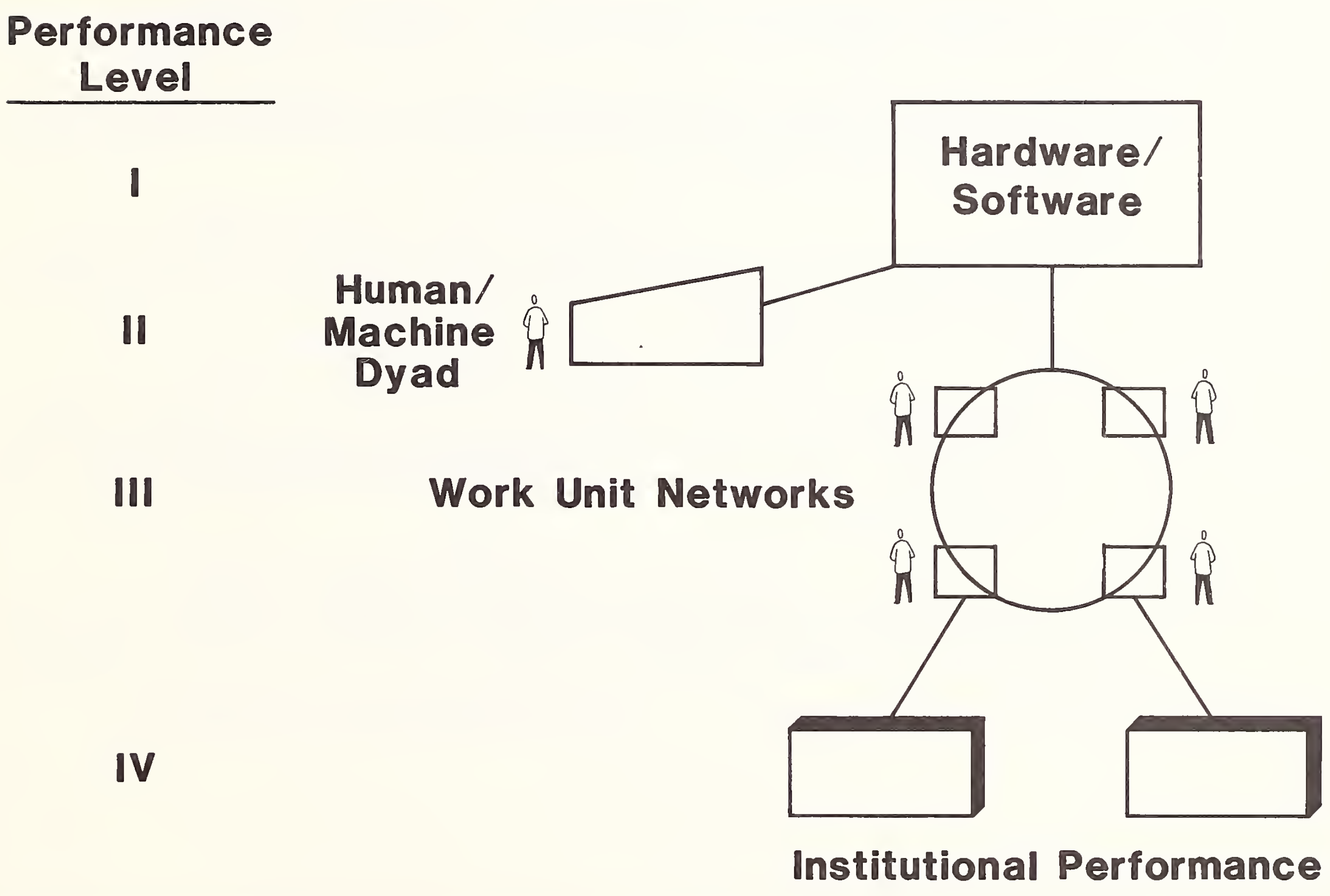
  - The contribution to institutional performance (bottom line)?

- Intelligent use of FGLs is required if adverse impacts on other performance levels are to be avoided and contribution to improved performance maximized.

- INPUT continues to believe that too much emphasis is placed on tools at the expense of sound system practices. Sound systems analysis is needed to optimize all four levels of performance.

INPUT®

TAKE A BROAD VIEW OF PRODUCTIVITY



E. ESTABLISH STANDARDS AND GUIDELINES

- INPUT has attempted to provide a general framework for evaluation of productivity tools by presenting systems categories. Establish boundaries for 4FGLs use based on the following:

  - Where the resulting application (system) will reside in the network hierarchy (mainframe, minicomputer, or intelligent workstation).

  - What phase of the development/life cycle is being addressed (requirements, analysis, programming, maintenance, etc.).

  - The type of system being developed (batch, transaction, interactive, decision support, expert).

  - The anticipated systems requirements in terms of transaction rates, processing, data base size, functionality, decision roles, and responsiveness.

- The resulting standards and guidelines need to recognize that there is no one total solution to all of the categories and that no tool can substitute for good systems analysis and design.

ESTABLISH STANDARDS AND GUIDELINES

- Establish Boundaries of Use (Standards) for 4GLs and Other Productivity Tools
  - Network Hierarchy
  - Development/Life Cycle
  - Systems Type
  - Systems Requirements
- Publish Guidelines

III THE STATE OF FOURTH GENERATION LANGUAGES

A. FGL TRENDS

1. REVIEW OF LANGUAGE EVOLUTION

- The only purpose of computer languages and systems software in general is to make computers easier to use. While this conclusion may seem somewhat simplistic, it becomes extremely complex as soon as two questions are asked:

  - Easy for whom?

  - Easy for doing what?

- The history of language development is replete with spectacular and embarrassing failures which mark times when these two questions were either not asked or the importance of human differences and problem diversity were ignored. The current enthusiasm for FGLs as the "new solution" for everybody and to everything is, at best, naive and can be dangerous for vendors and users who actually believe that a magic solution to the productivity problem has suddenly appeared.

- A general review of language evolution will provide some fundamental lessons which should have been learned from past experience. A brief analysis of these lessons will help in structuring the FGL market along the lines of "for whom" and "for what."

- Perhaps the first lesson learned about computers was the most profound: computer programs malfunctioned. This meant that computers did not always do what you "told them to do."

- The second lesson was that it was extremely tedious to communicate with computers in their own language.

- The third lesson was that even after you take substantial amounts of tedium out of the programming process, aptitudes for the craft vary significantly. (INPUT generally uses a range of 30 to 1 to define variations in individual productivity, and some experts consider that to be conservative.)

- Early experience with assembly language soon disclosed that the handling of data was much more difficult than performing simple arithmetic on those data (Lesson 4).

- The development of FORTRAN confirmed that scientific notation was easily translated to machine language (Lesson 5).

- Since scientifc notation was familiar to engineers and scientists, it was discovered they could program on a casual or "open shop" basis (Lesson 6). (It is also probable that aptitude of these early open shop programmers was substantially higher than that of the general business community, much less the general population.)

- However, former tabulating equipment supervisors soon discovered that, given a simple data structure (card image), simple reports could be generated with more ease than "wiring a board," much less programming a computer (Lesson 7).

- LISP is currently being touted as the second oldest higher level language (after FORTRAN), so Lesson 8 was learned early--human reasoning is difficult to simulate in computer systems.

- Unlike scientific notation, it soon became apparent to even the staunchiest COBOL advocate that natural English could not be easily translated by computers (Lesson 9).

- At this point, it also became apparent that executives could not read COBOL programs (written in "English" acceptable to computers) and understand what a computer was doing. In fact, it was discovered that programmers could not easily read each other's programs, and voluminous extraneous documentation was required. Thus, early claims of self-documentation of "English" as a computer language were put to rest (Lesson 10).

- However, a language which could describe precisely what a computer was doing in very concise form was developed. It was called A Programming Language, and it soon became possible to write entire COBOL and FORTRAN programs in a few "simple" statements (Lesson 11).

- Unfortunately, it was found that a language suitable for exercising the full power of computers in concise language appealed to only a small subset of those called programmers, confirming Lesson 3 (Lesson 12).

- Regardless of all subsequent disclaimers, it was IBM's intention to replace both COBOL and FORTRAN with PL/1, but it became obvious that programmers are reluctant to change languages and that established languages are virtually impossible to replace (Lesson 13).

- Designed as a multi-purpose language for both scientific/engineering and commercial data processing, PL/1 succeeded in attracting few from either user set (Lesson 14).

- The development of integrated operating environments under general purpose operating systems capable of running interactive, batch, DBMS, and a variety of languages demonstrated that a new level of language complexity (JCL) was required in order to use the various facilities (Lesson 15).

- Concepts of data reduction and information retrieval combined with the "ready availability" of processable data led to the development of query languages and ad hoc reporting languages to support management information systems. Unfortunately, it was soon discovered that file access methods did not permit easy development of such systems (Lesson 16).

- Even when systems were laboriously hand tailored to provide such systems, the data files soon required complex restructuring, and even then the data quality frequently was not capable of supporting the resulting system (Lesson 17).

- As DBMSs developed with languages (such as DL/I) to facilitate the building and maintenance of data bases, the fact remained that there was a lot of tedious and unappealing work associated with developing and maintaining high quality data bases. Considerable human and machine resource is necessary to maintain data quality. In other words, data base administration and management remains expensive and unappealing (Lesson 18).

- As timesharing extended computer power to a broader spectrum of users, it became apparent that a simple, interpretive language was necessary for casual users sitting at terminals. BASIC proved successful as a language for the fledgling programming, but is still met with disdain by most "professionals" (Lesson 19).

- The relational data model (and its associated algebra) demonstrated that flexibility and ease of use were performance penalties (Lesson 20).

- It took years for IBM to get a relational DBMS out of the laboratory where it was invented, and the performance problems associated with various prototypes have been well documented. Even after announcement of DB2, caution has been advised against use with large transaction-oriented data bases. When IBM urges caution because of performance problems, everyone should listen (Lesson 21).

- Exhibit III-1 summarizes the above history and resultant lessons.

• There are probably other lessons which could be learned from the stormy history of language development, but those listed by INPUT should be sufficient to permit some structuring of the demand for 4GLs and some barriers to unqualified accepance of their successors (FGLs).

2. WHERE DO 4GLs STAND?

• INPUT uses a complex set of systems categories for purposes of analysis, structuring, and forecasting. Each of these categories is broken down into subsets which are used as a general framework for specific product analysis. (The categories and their subsets, along with appropriate INPUT references, are listed in Appendix A.) While it is obviously beyond the scope of this study to pursue methodology in detail, it is helpful to display current 4GLs against some of these categories to illustrate their current use, potential, and limitations (see Exhibit III-2).

• For example, against the development/life cycle category, it is possible to reach certain fundamental conclusions concerning 4GLs.

- There is little argument that the primary benefit of 4GLs is to get applications up and running more rapidly than with a third generation

EXHIBIT III-1

LANGUAGE EVOLUTION

DATES	LANGUAGES	LESSONS
1950s	Machine	1 - Programs will malfunction 2 - Programming is tedious
	Assembly	3 - Aptitude varies widely 4 - Data manipulation more difficult than calculating
	Algebraic	5 - Scientific notation easily translated 6 - "Open Shop" programming possible
	RPGs	7 - Simple data/report structures easily automated
	List/symbolic	8 - Human reasoning difficult to represent
1960s	English	9 - "English" difficult to represent 10 - "English" not self-documenting
	APL	11 - Power and "simplicity" both possible 12 - Few humans appreciate power and simplicity
	PL/1	13 - Established languages not easily replaced 14 - Multi-purpose not accepted by either user set
	JCL	15 - Integrated operating environments add complexity
	Query/ad hoc	16 - Access methods not enough to support MIS 17 - Data structure and quality required to support IS
	DBMS	18 - Data bases expensive and difficult regardless of language
	BASIC	19 - Simple languages encourage casual use, but few "professionals" are impressed
1970s	Relational (algebra)	20 - Ease of use and flexibility are expensive 21 - Performance penalties will restrict use

EXHIBIT III-2

TRENDS WITHIN SELECTED SYSTEMS

CATEGORY	CURRENT	BARRIERS (Lessons Violated)	TREND
C - Network Structure	Output →	15 →	All
D - Network Hierarchy	Mainframes →	14, 15 →	Minis and Micros
G - Development/ Life Cycle	Programming and Test/Debug →	9, 10 →	All
H - Systems Type	Decision Support →	20, 21 →	Transaction and Batch
I - Systems Requirements	Low →	20, 21 →	High
J - User Set	Analyst/ Programmer →	3, 14, 19 →	Programmers and Casual Users

language such as COBOL, and that these benefits accrue primarily in the programming and testing/debugging phases of the development/life cycle.

- It is also well known that these two subsets represent approximately 15% of the life cycle costs of an applications system. Thus, if 4GLs permit development in 1/5 the time of a 3GL (as is usually quoted), then it is possible to save 12% of life cycle costs. While this may be significant, it is not sufficient to replace established languages (Lesson 13).

- In addition, it has long been INPUT's conclusion (based upon extensive research in systems development productivity) that improvement is needed in the early stages of the systems development process (in determining requirements, establishing specifications, and especially analysis and design). In fact, although tools and aids may be important in these early stages, it has been concluded that merely spending more time up front would be of significant benefit over the system's life cycle. While it can be argued that prototyping using 4GLs gets users involved in the development process early, it can also be argued with equal validity that, in the rush to get something running, all four of the early development phases suffer substantially.

- Past experience has shown that development efforts which do not emphasize the first four phases of the development cycle pay later--in the last three phases (documentation, installation, and maintenance), and everyone knows that maintenance represents approximately 60% of the life cycle costs. It is INPUT's opinion that current 4GLs do little to cut these costs and may, in fact, contribute to substantial problems if they are not used intelligently. (This conclusion is explained fully in New Opportunities for Software Productivity Improvement and Market Impact of New Software Productivity Techniques. The problem has to do with general systems quality issues and will be discussed later.)

- The term "intelligent use" implies that 4GLs are best suited for developing particular types of applications. The general network (or systems) structure category contains only five subsets: input, transmission (communications), processing, storage, and output. There is no question that 4GLs have historically addressed the output portion of this structure and can trace their genealogy directly back to RPGs, albeit incorporating inverted rather than sequential file structures (Lesson 7).

- Viewed from the perspective of the systems type category (which includes subsets of batch, transaction, interactive, realtime, decision support, and expert), it is also quite clear that 4GLs are most attractive for decision support systems. In other words, the oridignal purpose was to support query and ad hoc reporting. The two lessons learned in the 1960s concerning query/ad hoc reporting languages remain true today: access methods (by any other name) are not enough to support management information systems (MIS was the earlier name for decision support systems), and data structure (models) and quality are required to support any information systems (Lessons 16 and 17).

- In terms of the systems requirements category, which addresses performance in terms of transaction rates, etc., it is apparent that 4GLs have traditionally been used for developing relatively small systems which do not require large transaction volumes, heavy processing, or fast response against large data bases.

- In terms of the network hierarchy category (large mainframes, minicomputers, intelligent workstations, terminals, and mobile terminals), 4GLs have been primarily mainframe oriented.

- Having answered the general "Easy for doing what?" question first, it is now possible to address "Easy for whom?" INPUT has broken down the user set category into nine subsets (scientific, engineering, systems/procedures analyst, programmer, clerical/accounting, secretarial, administrative/managerial, executive, and casual).

- In their early form, 4GLs were used primarily by analysts (or more specifically analyst/programmers) and some end users in the clerical/accounting and administrative/managerial (for ad hoc reporting and query).

- Most "professional" programmers view 4GLs as being inadequate for any but the simplest reporting programs and are not impressed (Lesson 19). In addition, there is a general feeling among the professionals that only analysts who adopt a "quick and dirty" approach are attracted to 4GLs. This attitude continues to exist and is based primarily on early efforts to implement "systems" using 4GLs. (For example, personnel systems with wonderful reporting and query capability but no provisions for updating files or security.)

- This attitude on the part of the information systems department has tended to keep 4GLs "under control" and confined their use as described above.

3. DISTRIBUTED SYSTEMS DEVELOPMENT

- However, because of the productivity problems within the IS department and the advent of the personal computer, the environment has changed. This change was described in detail in INPUT's reports on productivity last year. Essentially, a new development environment has been created, and it has the following characteristics and ramifications:

  - The environment has been defined by INPUT as being one of distributed systems development (DSD). The DSD environment is manifested in personal computers, information centers, prototyping, and the desire for micro-mainframe links.

- 4GLs are one of the driving forces in the DSD environment, but it is important to recognize a number of conflicts which will arise as that environment evolves. INPUT identified the following which should be of concern:

  - Top-down design versus bottom-up design.
  - Security versus access.
  - Ease of use versus added function.
  - Data quality versus distributed data bases.
  - Micro demands on mainframes versus off-loading of mainframes.
  - Management reports versus management reports (conflicting information).
  - GST trends versus hardware/software planning.

- These conflicts translate into some serious potential problems in terms of systems quality and productivity.

  - Data base integrity and synchronization.
  - Security, protection, and privacy.
  - Conflicting reports to management.
  - Hardware performance problems.
  - Deterioration of data/information quality.

- . Unanticipated expense.

- . Unworkable systems solutions.

4. WHERE ARE FGLs GOING?

- As the driving force behind information centers and prototyping, 4GLs are evolving rapidly into FGLs (fourth, fifth, and/or future generation languages) which can be used for more complex applications than their predecessors. However, in breaking out of the restrictive environment in which they have been placed by the IS department, they run the risk of attempting to breach barriers which history tells us are quite formidable. Exhibit III-12 summarizes this challenge.

  - Having traditionally concentrated on the reporting requirements for decision support systems (output), FGLs now aim at becoming the primary development tool for major applications systems. This involves the other fundamental systems functions of input, transmission (communications), and processing. Lesson 15 pointed out that integration is accomplished at the cost of complexity. This complexity applies not only to the developed system, but also to the tools (languages) used for their development.

  - 4GLs have found their primary use where the data and major applications are, in other words, on mainframes. As they expand to encompass minicomputers and intelligent workstations, FGLs will run the risk of failing to satisfy any of the diverse user sets along the way (Lesson 14). In addition, the integration of mainframe, minicomputer, and intelligent workstation operating environments for development purposes (to say nothing of targeting appropriate levels of the network hierarchy for the resulting applications) represents a major technical challenge (Lesson 15).

- The evolution of 4GLs suitable for decision support systems toward transaction processing and batch systems, in the belief that anything written in 3GLs can best be done in 4GLs, defies Lessons 20 and 21. Ease of use and flexibility are not free, and reduced performance will restrict use of 4GLs (and FGLs) once this is proven again--and it will be.

- Lessons 20 and 21 also apply to the general trend from low to high in the systems requirements (performance) category. There is a tendency for both users and vendors to extend FGLs beyond their capabilities, and the results may have impact on even the intelligent use of FGLs because of overreaction as a result of the inevitable failures.

- As the types of systems being developed with FGLs are extended, so will the user set subsets being serviced. It is impossible to satisfy all categories of users with a single language, and "ease of use" has different meanings across user subsets, within a user subset (because of individual aptitudes) and over time as users become more skilled. Neither professional programmers or casual users will be satisfied with any language designed for both (Lessons 3, 14, and 19).

● The current enthusiasm for 4GLs is understandable; they have permitted the IS department to be more responsive to user requests and they have (when properly employed) gotten users involved at an early stage in the development process. Last year in Market Impact of New Opportunities Software Productivity Improvement, INPUT stated that "unless the questions of data/information quality and systems performance are addressed by FGLs, they will prove to be self-defeating." There is real danger is misusing FGLs as they are applied to more complex applications.

B. PROBLEMS OF MISUSE

1. FGLs AND THE DSD ENVIRONMENT

- As the driving force behind the DSD environment, FGLs obviously have some connection with any problems which arise in that environment. As INPUT stated in New Opportunities in Software Productivity Improvements, ". . .to the degree that the DSD environment creates IS problems, and fourth generation languages contribute to the implementation of that environment, fourth generation languages must be analyzed as part of the problem in order to ensure their continued acceptance.

- The problems which can be anticipated in the DSD environment were mentioned briefly earlier in this report. It is strongly recommended that these problems be understood as potential limiting factors in the FGL marketplace. Detailed evaluations of strengths and weaknesses in the DSD environment are contained in New Opportunities for Software Productivity Improvement.

- Some of the evaluations (by IS management and industry experts) run counter to the way FGLs are being sold in the marketplace. For example, the primary disadvantage of prototyping is considered to be "excess resource use and waste," whereas using FGLs for prototyping is being promoted as a primary means of productivity improvement. Such obvious conflict requires better understanding of what productivity really means.

2. PRODUCTIVITY AND PERFORMANCE

- In 1983 INPUT prepared a report on the Impact of Office Systems on Productivity which explored the general problems of white collar productivity. It was concluded that generalizations about productivity improvement were meaningless unless performance was measured in four ways:

- Hardware/software.

- Human/machine dyad.

- Work unit.

- Institutional.

- This resulted in the establishment of the performance systems category (Appendix A), and it was concluded that maximizing performance in one category does not necessarily ensure corresponding improvement in any other category; in fact, negative correlations can be established. The performance systems category can, and should, be used to evaluate not only office automation systems but also the systems development environment and any resulting applications systems.

- While all of this may be considered a "bunch of theory," it is INPUT's opinion that the conclusions which have been reached using this methodology are substantially closer to reality than any of the forecasts and/or competitive assessments which are normally made for applications development tools. As this report was being completed, a case of catastrophic misuse of a 4GL occurred, and it will serve as a useful case study.

3. REALITY VERSUS THEORY IN 4GL MISUSE

- The case study in 4GL misuse is the one involving the New Jersey Department of Motor Vehicles as reported in Computerworld (9/30/85), and INPUT brings it up with reluctance because none of us (users, vendors, or consultants) come out looking very good. However, there are some very important lessons to be learned from this experience and the fact that it is being aired publicly is healthy. All to frequently we try to bury our systems mistakes. Unlike the consulting firm involved, which felt that, "We do not want to get into all the details. We do not feel that it would serve any useful purpose", INPUT believes avoiding such misuse of FGLs in the future is necessary if markets are to develop and users are to achieve improved productivity.

- Simply stated, the facts are as follows:

  - A prominent and well respected 4GL product (and supporting relational DBMS) was recommended by a prominent and well respected consulting firm for developing a new and improved motor vehicle registration system for the state of New Jersey. The recommendation was justified based on "productivity gains anticipated during the coding and testing phases" of systems development (very much in line with the general trends in the industry).

  - The results, once the system was installed (and presumably tested), were as follows:

    - The system designed to support 1,000 terminals foundered at 200.

    - One problem was that specified response times of three to five seconds exceeded five minutes.

    - The new system, which cost $6.5 million, is not capable of keeping up with the daily workload even with overtime work. Presumably the workload was previously being handled by some less automated system.

    - The result has been that drivers have been unable to register their vehicles or are incorrectly listed as being unregistered. State police have been stopped from citing drivers for the offense.

    - That is not all. Renewal notices have been sent to the wrong drivers (those not due for registration), and many police cars and other public vehicles have been registered to the wrong municipality.

- Not surprisingly, this has led the state of New Jersey (to say nothing of the citizens) to raise some questions about the quality of the system. One official was even reported to have said, "It was a real barn-burner application, and I am not sure. . .(the 4GL). . .was the right technology to use." It would appear that either theory dies hard in the face of reality or that it is extremely difficult to be sure of anything when confronted with systems catastrophe.

- The consulting firm has agreed to "redo the system," but has made no statements as to whether the anticipated "productivity gains during coding and testing" were realized or what language and/or methodology will be used during the rework.

- The vendor of the 4GL has indicated that the consulting firm was warned not to use the product to develop the system ("We found ourselves in the unusual situation of our advice not being followed on our product.") for the following reasons:

    - Batch sequential processing takes as many lines of code in the 4GL as it does in COBOL, and therefore there is "no advantage" to using a 4GL.

    - Twenty percent of the subsystems required "heavy processing" and should have been written in COBOL.

- The whole situation has developed into a circular, finger-pointing contest and there is certainly enough blame to go around. In addition, it is not going to die down. The vendor has been reminded of previous public statements which claimed the product was "a functional replacement for COBOL" and that "there is virtually nothing you can develop in COBOL that you cannot also develop with. . ." These statements are now being qualified by saying that this remains true for "most applications."

- One thing is certain--everyone directly involved has suffered substantially already and the immediate impact will be unfavorable for roughly comparable products.

• It is appropriate at this point to review this continuing case study against INPUT's systems categories and determine what could have been known in advance of the New Jersey Motor Vehicle Registration System.

  - The productivity hierarchy systems category was established as a result of a major INPUT multiclient study in 1980. Essentially, the study concluded that any productivity improvement program must be composed of the following five elements (in decreasing order of importance): 1) commitment to quality, 2) end-user involvement, 3) broadbased management, 4) high quality personnel, and 5) tools, aids, and methodologies. From the facts above, it appears apparent that:

    . The commitment to quality on the part of the developer was secondary to ease of coding and "getting something running."

    . The user felt his involvement (and responsibility) could be minimized provided he was willing to buy a system.

    . The system development was managed solely by the consulting firm which contracted to "deliver the system" and reportedly ignored the advise of both the customer and the vendor in its use of the 4GL.

- The use of the 4GL was obviously dictated by analysts and designers who were at best naive and inexperienced. Perhaps they had prototyped a few decision support systems, but it is apparent that they were in over their heads at all stages of the development process.

- Given the weaknesses in the base of the productivity "pyramid," the choice of the right tools and aids was practically immaterial; the effort was doomed from the start.

- Adopting the development approach taken in the name of "anticipated productivity," however, requires an evaluation against the performance systems category. The following seem obvious and predictable:

  - Hardware/software performance suffered as a result of using the 4GL; for example, response time was off by between one and two orders of magnitude.

  - Perhaps performance at the human/machine dyad was enhanced to the degree usually promised by proponents of 4GLs in terms of relative lines of code--all the more reason to view productivity on a broader basis.

  - Indeed, it is possible that the implementation project team (work unit) got the system up and running faster than with a third generation language, and even on schedule. But this does not mean it was a highly productive effort--somebody has to redo it.

  - From the point of view of the institution, the impact on productivity has been catastrophic. The state cannot register cars even with overtime, and the police cannot tell whether cars are registered, overdue for registration, or stolen (evidently, even their own).

        . This is an excellent example of negative correlation(s) within the performance systems category and should illustrate the need for the category to assure a broad view of performance measurement.

    - Other systems categories mentioned earlier in this report can be applied with similar telling effect against this case study, as can the lessons we should have learned from the past. The one major lesson to be learned from the case study is that FGLs are not a solution to all applications and systems problems, and the penalties for believing (or pretending) they are can be substantial.

- It is also important to point out that the case study demonstrates the reality of the "theoretical" problems INPUT predicted for the DSD environment last year (Chapter III, Section 3 of this report). Specifically, it is a good example of performance problems, data and information deterioration, unworkable systems, and unanticipated costs, in other words, how 4GLs may be counter-productive if they are misused.

- Before leaving the case study, INPUT would like to return to Lessons 20 and 21 (Exhibit III-1) for just a moment and draw a few conclusions not directly supported by published information on the case study at this time.

    - The response time problem experienced in the case study (as opposed to any batch turnaround time they may have experienced) is probably related more to data base design than it is to language use. While the auto registration data base is large, it is not complex in terms of content or use. If the relational model was employed, it is probable that an unnecessary performance penalty was paid for flexibility on a data base with highly predictable usage patterns (Lesson 20).

- Therefore, if the consulting firm merely rewrites the batch processing modules in COBOL (as suggested by the vendor), it is probable that response time will continue to be unacceptable as the active terminals approach the specified 1000 level. Can all problems be solved with a relational DBMS? Perhaps, but not with performance acceptable to all users (for example, police officers in patrol cars), and poor performance will restrict use (Lesson 21).

C. IBM's STRATEGY

- IBM's general strategy was outlined in the predecessor to the report Data Base Management Systems. In that report, the four IBM strategic periods were briefly described, and they will not be redefined here except to state that there is a systems category which corresponds to them (Appendix A) and the importance for FGLs is as follows:

  - During the current SNA/DDP period, IBM will concentrate on the centralization of control under SNA, operating systems (VM/MVS and others at various levels in the network hierarchy), and DBMSs. This implies that the primary IBM competition for FGLs will evolve from its DBMSs during that period. During this period, IBM will be happy to let FGLs proliferate. They sell a lot of hardware and IBM cannot be held responsible for catastrophic misuse.

  - The electronic office period (1990-1995) will see IBM give more attention to language development supporting decision support systems. It is probable that such languages will address specific industries and user sets. In other words, IBM will facilitate integration of existing computer- and paper-based systems by providing familiar (differentiated) languages.

- This leads naturally to the expert systems period (1995-2000) where languages will become specialized to the degree that they will be mechanized within specific domains of knowledge-based systems.

- IBM may feel that current 4GLs are only suitable for the information center and cannot be used for the development of major systems in the electronic office period (much less be able to evolve or even be adapted to play any significant role in the expert systems period). It would certainly seem plausible that if IBM thought 4GLs were a proper base for future language development they would be doing something more than extending QBE and SQL and marketing Intellect.

- As always, it is wise to give some thought to what IBM may be thinking regardless of whether you agree with it. As described in <u>Data Base Management Systems</u>, IBM is emphasizing centralization during the SNA/DDP period, and this has the effect of turning the large central host into a data base machine, with the following ramifications:

  - Extracting data from large corporate or operating data bases (IMS) and passing these data to planning or personal data bases (DB2) is essentially a batch processing environment. INPUT believes that IBM recognizes this and realizes that MVS/XA is well suited for this environment (since this is what it evolved from).

  - Having had extensive experience with serial batch processing, IBM recognizes that the performance penalties associated with many current 4GLs makes them unsuitable for the development of such applications. This conclusion would seem to be supported by IBM's enhancement of its Query Management Facility to support the initiation of batch jobs (as opposed to enhancing the languages to permit development of batch applications).

- IBM must also be aware that the management and use of large data bases (whether data reduction, extraction, information retrieval, statistical analysis, or quality control) cannot be easily and/or effectively described in English-like nonprocedural languages.

- The large host data base machines of the SNA/DDP period do not present a user friendly environment to those who only want results without regard for how they are accomplished. Even if the 4GL-generated application does not result in a catastrophe such as the New Jersey Motor Vehicle Registration case study, the ongoing cost of the system will certainly make the user want to know what is going on.

• In the electronic office period, the term INPUT uses to signify reduced paper flow, this reduced paper flow implies significant changes not only in current paper procedures, but also in current office automation systems which expedite paper production.

  - The DSD environment and 4GLs have improved performance of the human/machine dyad, but this has generally been accompanied by a dramatic increase in paper documents of questionable value (quality). The impacts on the performance (productivity) of individual work units and at the institutional level have frequently been negative. (This is the basis for INPUT's continuing concern about information quality.) The focus of the electronic office period will be toward improved productivity at the work unit level.

  - Unlike the human/machine dyad where the concern is for generating information from data according to individual and flexible needs, most work units have stated goals and objectives based on established systems and procedures. Paper documents proceed in some semblance of order from workstation to workstation for human processing and/or analysis. Systems to support the electronic office will be automating data/information flow--a process.

- PERT charts and flow diagrams are more appropriate than English for describing such processes, and they are definitely procedural in nature. Certainly, the developers of such applications will have to not only understand how objectives are achieved, but will want to exercise direct control over the process. In fact, the justification for the substitution of electronic for paper media is usually based on consolidation and elimination of workstations.

- It is highly unlikely that such systems with all of their dependencies on outside data and information sources can be designed from the bottom up or evolved into being through prototyping. They are going to require a professional systems and programming effort. It is INPUT's belief that IBM is adopting a cautious approach to both LANs and the "office of the future" precisely because they recognize the magnitude of the task. If 4GLs were capable of implementing the applications required for the electronic office, the SNA/DDP period could be shortened considerably. INPUT does not believe this can occur.

• When considering the expert systems strategic period, it is important to recognize that there is great deal which can (and must) be done to improve the decisionmaking process which goes beyond providing pretty information (fancy reports and graphics) and yet falls short of the application of artificial intelligence. There is an inherent danger in the DSD environment that tools are dictating architecture of decision support systems, and this danger is currently even more pronounced as we approach the expert systems period (see <u>Artificial Intelligence and Expert Systems</u>, INPUT 1985). As far as languages are concerned, there are some very clear messages.

  - The implementation of expert systems do not lend itself to description in nonprocedural, English-like languages.

- The user interfaces to expert systems will have to be in more natural language than that provided by current 4GLs.

- Those skilled in English-like languages (including COBOL) are going to have a great deal of difficulty with LISP-like languages. In fact, "knowledge engineers" capable of developing and maintaining knowledge-based systems are already beginning to appear.

  . It is probable that knowledge engineers are nothing more than highly skilled programmers who would never be caught dead programming in either COBOL or a 4GL.

  . Nonetheless, the most important function of the knowledge engineer is to get the expert involved enough so the decision rules employed in his particular domain can be described (precisely the job of any good systems analyst).

- There is one important attribute of expert systems which has been recognized and should be carefully tracked as the systems develop. There is a requirement that the system be able to explain to the user how it has solved the problem (made the decision). This is directly opposed to the current "results oriented" trend which states that the user "need not be concerned" with where data is coming from or how a program "works." As the human/machine dyad becomes more symbiotic, there is a need for both to communicate clearly with each other, and there is little reason to believe the human will be able to dictate totally the language employed.

- IBM's work on artificial intelligence is concentrated in the Research Center at Yorktown Heights, and the recent announcement of Prolog should not be viewed as the official "blessing" of the market. IBM knows how long it takes for research projects to have any significant impact on the market, and the announcement should be viewed as nothing more as establishing a presence in the market. The real purpose of the IBM announcement is to continue to apply pressure on minicomputers in the network hierarchy.

- There is a tremendous gap between the capabilities of 4GLs and the requirements of all the IBM strategic periods. While it is probable that IBM knows this, it should not be concluded that IBM has the solution to the language problems. In fact, the FGL market is extremely promising precisely becasue IBM, based on past experience, seems content to let others lead the way.

IV OPPORTUNITIES AND CHALLENGES

A. 4GLs TO FGLs

- As pointed out earlier in this report, there are certain barriers to expanding the use of 4GLs. The case study identified the most critical and pressing of these barriers as performance. There is a point where improved performance at the human/machine dyad cannot be used to justify degraded performance at the hardware/software level; the resulting systems will not (or cannot) be used.

  - It is certainly a good indication that 4GLs have reached the point of diminishing returns on trade-offs between the two levels when COBOL is used as a standard for good performance. It is INPUT's opinion that as applications begin to spread across the processing hierarchy, the expense of running batch jobs (even those written in COBOL) will begin to receive renewed attention.

  - This will present those concerned with the development of FGLs with both challenge and opportunity. Given COBOL as target and recognizing that there are a variety of performance requirements along the road from a first prototype, through testing, and finally into production at various processing levels within the processing hierarchy, it should be possible to leapfrog COBOL in terms of performance.

- Switching from an interpreter to a compiler is not a magical solution to the variety of performance requirements briefly outlined above. The potentials for implementation of "quick and dirty" compilers and cascading up and down the language hierarchy with translaters in search of a good optimizer make both the choices and the results quite complicated. These options and their ramifications will be discussed in the next report of this set, Applications Development Techniques.

- During the course of the development of this report, the five-year effort to define a new, "standard" version of COBOL neared completion. Over 20 years ago, it was supposed to do all the wonderful things we are still talking about today, and now there are those who are making similar claims for current 4GLs (except for the standardization). One of the driving forces behind COBOL in its infancy was Jean Sammett, but she is now quoted in The Computer Science and Engineering Research Study (published as "What Can Be Automated?" MIT Press, 1980) as saying:

    - "Programming Languages, using any definition, are primary means by which a person communicates with a computer. . .The real truth seems to be that there is no single best way for people to communicate with a computer, hence, no single solution. Thus, there will be no single language useful to everyone. . ."

    - "Until we reach the situation described above (being able to instruct computers in natural language), the next best thing will be 'user-defined languages.' By this we mean (software) systems which permit users, first, to define languages that fit their own needs with respect to functional capability, jargon, and personal tastes in style, and then to easily implement them. The key part of the problem is to provide a system which permits easy implementation with an acceptable level of efficiency."

- INPUT agrees with Ms. Sammet's comments with one exception. It is our opinion that the quest for "instructing computers in natural language," while a worthwhile goal of those doing research in AI, is both impractical and undesirable for developing applications and/or systems (especially if one is ever to achieve an acceptable level of efficiency). The simple fact of the matter is that aptitudes for abstract reasoning and verbal ability vary tremendously (extension of Lesson 3), and there is no indication that outstanding programmers are facile with English or that a writer for Computerworld could ever figure out what went wrong with the New Jersey registration system.

- The important point is that computer languages, like natural languages, are going to continue to evolve, and arbitrary classification into generations or attempts to standardize only serves to obfuscate this fact. Attempts to apply 4GLs beyond their limits are doomed to failure.

B. LANGUAGE EXTENSIBILITY

- The concept of user defined languages is obviously well suited to natural evolution into FGLs, and it is not new. Technnically, the concept was born with macro assembly programs, now referred to as extensible languages. Such languages consist of three parts.

  - The base language, which INPUT believes should also be the systems implementation language (SIL), which can be easily extended into a limitless variety of programming languages. (The concept of using a SIL to generate a compiler for itself is sometimes referred to as "incestuous" use of a SIL and, while not new, is seldom kept pure in implementation. It is especially important in the development of extensible languages.)

- The metalanguage used to describe the necessary expansions, contractions, and other modifications to the base language and to create a new language.

- The derived language which then becomes the executable part of the program.

- It is obvious that the decision whether or not to open up the facilities of extensible languages within any given user organization is an important one, but the value for vendors in implementing new languages cannot be denied. In addition, the potential for "accommodating" diverse languages in an already chaotic network environment is of equal importance. The technical challenge of extensible languages is great, but the opportunities are commensurate with the challenge.

C. LANGUAGE OPPORTUNITIES AND STRATEGIC PERIODS

- In New Opportunities for Software Productivity Improvements, INPUT outlined a number of software tools and systems to control quality in the DSD environment. All of those systems have language ramifications. Briefly, those tools were as follows:

  - An Information Base Management System (IBMS) capable of managing dictionaries and directories for both encoded (computerized) and paper-based data/information bases. The need for languages suitable for programmers, librarians, lexicographers, etc. is apparent. Such a system is required right now during the SNA/DDP strategic period.

  - A Document Control System (DOCS) which requires tools to control an expanded storage hierarchy (on-line, paper, micrographic, optical disk, etc.) and languages (or diagrams) to permit the handling, control, and processing of these documents. It is important to recognize the distinction between information retrieval languages and those envisioned here, where the user is concerned about the location, movement, processing (or transformation), and control of a document. For example, an operator may wish to direct a certain class of retrieved documents in image form to a scanner/reader for selective updating of an encoded data base. The development of such tools must precede the electronic office strategic period.

- Current languages for the development of expert systems (LISP and PROLOG) will tend to slow development because there are not enough people skilled in their use. There is also every indication that highly skilled knowledge engineers will be required to maintain such systems even after they are developed. This raises some rather embarassing questions concerning the whole area of knowledge-based systems.

V CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

- Current 4GLs continue to be used primarily for secondary applications which deal with data which has already been processed, but success with "pseudo" production systems is leading to attempts to develop larger production systems which are transaction oriented. As 4GLs evolve into FGLs, this testing of the boundaries of traditional use is natural and even desirable. However, experience tells us that there will be substantial barriers to acceptance among some user sets.

- As demonstrated by the case study in this report, the potential problems of systems quality which were identified in Market Impact of New Software Productivity Techniques are very real. Since 4GLs are the driving force behind the DSD enviornment, they will be identified as part of the problem whenever they are used in an inappropriate manner.

- There is a natural tendency for solutions to the "productivity problem" to be first overrated and then undervalued. It is probable that 4GLs reached their peak of being overrated during the last year and may descend rapidly to being undervalued because of adverse publicity associated with any failures which will inevitably occur as the applications boundaries are tested.

- It is necessary to evaluate the impact of 4GLs against all performance levels (hardware/software, human/machine dyad, work unit, and institutional) and not merely against productivity in systems development. It is possible to be highly productive in systems development at a cost which is unacceptable at other performance levels.

- INPUT concludes that there is "residual expense" associated with the DSD environment (and 4GLs) which can more than offset the benefits to be gained from improved productivity in systems development. In addition to the normally identified problem of continuing operational costs, the tendency of evolving systems from the "bottom-up" encourages sloppy systems analysis which can have enormous costs over the system's life cycle.

- While there is currently no simple way to understand this complex technological environment, INPUT believes that a good start has been made through various reports we have published in the past and that these reports can give our clients an advantage in making effective use of the latest technology with minimum risk from unexpected impacts. During 1986, INPUT will emphasize the need to analyze "residual expense" in our report series. This decision was reached based on the conclusion that residual expense in the DSD environment is usually substantially more important than concerns for hardware residual values which receive so much attention early in the system's life cycle.

B. RECOMMENDATIONS

- Fourth generation languages must be used within the context of their capabilities. The New Jersey Motor Vehicles case shows the expense of crossing 4GL contextual boundaries.

- Establish boundaries (standards) of use for 4GLs and other tools of the DSD environment in terms of the following systems categories (Appendix A):

- Network hierarchy.

- Development/life cycle.

- Systems type.

- Systems requirements.

- Performance.

- Provide users with guidelines and tools to avoid premature attempts to penetrate the boundaries established. In other words, avoid the natural tendency on the part of both users and vendors to first overrate and then undervalue applications development tools, aids, and methodologies.

- Future languages will become more specialized, becoming both application and user dependent. The challenge for information systems (IS) professionals is to select the proper language for the correct circumstance. The growing array of languages present a palette of choices. The incorrect selection can cost more than the productivity savings that justified its use.

- Remember, productivity improvements must transcend the entire system's development/life cycle. A narrow view of productivity gains can lead to costly and embarassing occurances.

- In the near term, confine the use of 4GLs to decision support and report and query systems. Be sure you are aware of all the risks and costs of using non-traditional languages to do traditional programs.

- It is just as wrong to undervalue 4GLs as it is to overrate them. Productivity improvements can be realized in getting ad hoc information to end users. 4GLs do have a place in IS' portfolio of languages and tools. It is IS' responsibility to exploit a 4GL's strengths without using it as the universal solution to the systems development productivity problem. The next report in this series, Application Development Techniques, will focus on the tools available to improve software development productivity.

- INPUT believes that there is a tremendous need for the rapid development of 4GLs into FGLs. The statement contained in New Opportunities for Software Productivity Improvements still applies: ". . .it is INPUT's opinion that unless the questions of data/information quality and systems performance are addressed by FGLs, they will prove to be self-defeating. We are betting that will not happen." It is up to IS management to see that it does not.

APPENDIX A: INPUT SYSTEM CATEGORIES

A- GST DIRECTION

1- Centralization

2- Integration

3- Differentiation

4- Mechanization

B- QUALITY

1- Objectives

2- DIK

3- Auditability

4- Measurement

5- Feedback Loops

6- Validity/Reliability/Predictabiliy

7- Security/Privacy

C- NETWORK STRUCTURE

1- Input

2- Transmission

3- Processing

4- Storage

5- Output

D- NETWORK HIERARCHY

1- Large Mainframes
2- Minicomputers
3- Intelligent Workstations
4- Terminals
5- Mobile Terminals

E- SOFTWARE HIERARCHY

1- SNA
2- Operating Systems
3- DBMS
4- Languages/DSS
5- Industry Turnkey
6- Applications
7- DIK
8- Users

F- DEVELOPMENT STRUCTURE

1- Design
2- Program
3- Work Unit Organization
4- Operational
5- Rigidity/Flexibility

G- DEVELOPMENT/LIFE CYCLE

1- Requirements
2- Specifications
3- Analysis
4- Design
5- Programming
6- Testing/Debugging
7- Documentation
8- Installation
9- Maintenance

H- SYSTEMS TYPE

- 1- Batch
- 2- Transaction
- 3- Interactive
- 4- Realtime
- 5- Decision Support
- 6- Expert

I- SYSTEMS REQUIREMENTS

- 1- High/Low Transactions Rates
- 2- High/Low Processing Requirements
- 3- Large/Small Data Base
- 4- High/Low Functionality
- 5- Many/Few Decision Rules
- 6- High/Low Responsiveness

J- USER SET

- 1- Scientific
- 2- Engineering
- 3- Systems/Procedures Analyst
- 4- Programmer
- 5- Clerical/Accounting
- 6- Secretarial
- 7- Administrative/Managerial
- 8- Executive
- 9- Casual

K- PERFORMANCE

- 1- Hardware/Software
- 2- Human/Machine Dyad
- 3- Work Unit
- 4- Institutional

L- PRODUCTIVITY HIERARCHY

1- Tools/Aids/Methodologies

2- High Quality Personnel

3- Broadbased Management

4- End User Involvement

5- Commitment to Quality

M- IBM STRATEGIC PERIODS

1- SNA/DDP

2- Electronic Office

3- Expert Systems

4- Custom Systems