MANAGING THE SYSTEM DEVELOPMENT PROCESS

INPUT
LIBRARY

ABOUT INPUT

INPUT provides planning information, analysis, and recommendations to managers and executives in the information processing industries. Through market research, technology forecasting, and competitive analysis, INPUT supports client management in making informed decisions. Continuing services are provided to users and vendors of computers, communications, and office products and services.

The company carries out cor research. Working closely w tant issues, INPUT's staff n interpret the research data, mendations and innovative needs. Clients receive rep access to data on which an continuous consulting.

Many of INPUT's professional staff members have nearly 20 years experience in their areas of specialization. Most have held senior management positions in operations, marketing, or planning. This expertise enables INPUT to supply practical solutions to complex business problems.

Formed in 1974, INPUT has become a leading international consulting firm. Clients include over 100 of the world's largest and most techni- mpanies.

U-1980
SDP 0

AUTHOR
Heidenrich, George

TITLE

Headquarters
2471 East Bayshore Road
Suite 600
Palo Alto, California 94303
(415) 493-1600
Telex 171407

Los Angeles
4676 Admiralty Way
#401 C
Marina Del Rey, California 9
(213) 823-1230

ain Street
, Michigan 48170
-8730

on, D.C.
th Lynn Street
, Virginia 22209
-2118

UNITED KINGDOM
INPUT, Ltd.
Airwork House (4th Floor)
35 Piccadilly
London, W.1.
England
01-439-4442
Telex 269776

7-7-26 Nishi-Shinjuku
Tokyo
Japan 160
(03) 371-3082

TES
Infocom Australia
Highland Centre, 7-9 Merriwa St.,
P.O. Box 110,
Gordon N.S.W. 2072
(02) 498-8199
Telex AA 24434

Italy
PGP Sistema SRL
20127 Milano
Via Soperga 36
Italy
Milan 284-2850

INPUT Planning Services for Management

X-PRO
812

000006

MANAGING THE SYSTEM DEVELOPMENT PROCESS

DECEMBER 1980

MANAGING THE SYSTEM DEVELOPMENT PROCESS

TABLE OF CONTENTS

LIST OF EXHIBITS

I INTRODUCTION

I INTRODUCTION

A. PURPOSE AND SCOPE

- Much has been written about the larger issues confronting business enterprises trying to make the EDP function more responsive to their business needs. This study addresses the problem from a very specific viewpoint:

    - Its perspective is that of the EDP executive primarily concerned with the problem of furnishing services in a set environment with presently available resources.

- Questions to which an executive in this position might wish to have answers include:

    - How do I involve my clients, the end users, in the EDP system development process, in the sense that they feel responsible for the system and are willing to exert efforts to support it?

    - How do I handle the personality-fraught situation where a project is being designed for many different clients with contradicting needs and viewpoints?

    - How do I hold on to my "superstars"; i.e., the few best performers who turn in 80% of the useful output of my department?

- How do I make software maintenance more rewarding and enticing?

- Is there any way to check the "technical adventure" syndrome; i.e., the apparent desire of EDP professionals to launch state-of-the-art systems from scratch every time?

- How can the process of estimation and measurement be improved?

- How can "target drift" (i.e., the gradual divergence of a project from the originally chartered course) be controlled?

- This study addresses these questions and other related issues, and provides suggestions for possible solutions.

B. RESEARCH AND METHODOLOGY

- To support this study, INPUT interviewed EDP executives at 30 organizations, representing a variety of industries and geographical locations.

- Each respondent was asked the set of 34 questions listed in Appendix C.

- Organizations to be interviewed were selected first on their use of large-scale computing equipment, and then on their distribution by industry, to obtain a representative sample.

  - The decision to limit the survey to users of large computers was motivated by the assumption that such users are more likely than organizations with small-scale equipment to experience the type of information system development and management problems addressed in this study.

- Distribution of the interviewed organizations by industry, and the type of main computing equipment they employ, are summarized in Appendix B.

- In order to establish the perspective that each respondent adopted during the interview, INPUT asked all of them to classify their viewpoint as corporatewide, divisional or departmental. These data are also given in Appendix B.

- INPUT attempted to define the management position of the respondent by asking for a specification of the number of management levels supervised by the respondent. These data are also summarized in Appendix B.

- In addition to the data obtained from this specific interview program, certain "tool" and "technique" information for this study was obtained in conjunction with the recent multiclient study INPUT conducted on the topic of improving software productivity. Information concerning the availability of this study can be obtained from any INPUT office.

II EXECUTIVE SUMMARY

II EXECUTIVE SUMMARY

A. MANAGEMENT ISSUES IN EDP SYSTEM DEVELOPMENT

- Despite the widely held belief that management problems are primarily responsible for the large number of projects that experience cost and schedule overruns and fail to gain user acceptance, one-half of the organizations INPUT interviewed regarded the effectiveness of their EDP system development management as "above average."

  - Only 10% rated themselves "below average."

  - However, 25% declined to answer this question.

- The three most pressing things that management should do to improve the EDP system development process, according to the organizations INPUT interviewed, are shown in Exhibit II-1, and include:

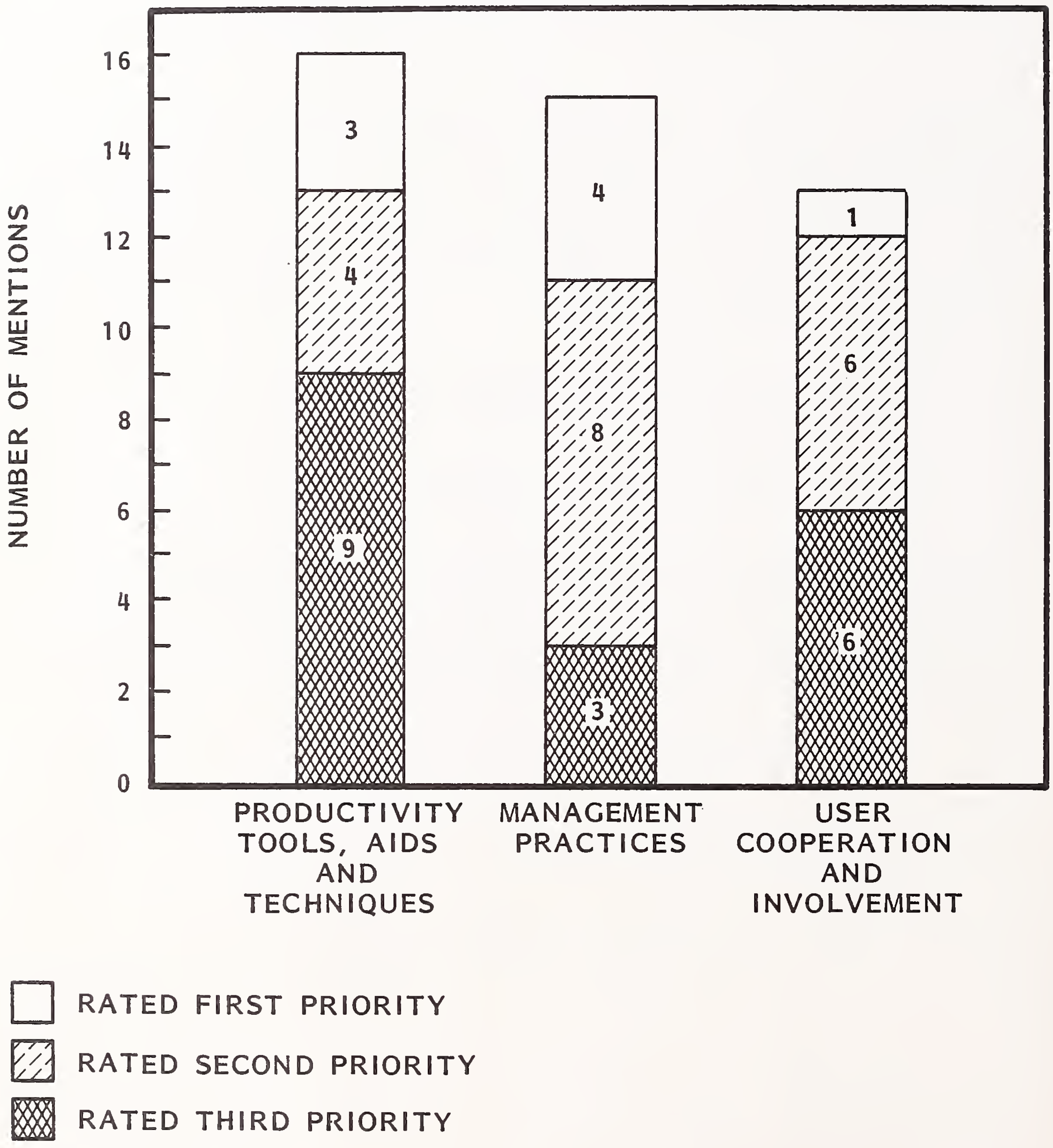
  - Increased reliance on tools, aids and techniques.

  - Improved management practices.

  - Increased user cooperation and involvement.

EXHIBIT II-1

THREE MOST FREQUENTLY MENTIONED
SYSTEM DEVELOPMENT IMPROVEMENT TECHNIQUES



TOTAL RESPONSES: 30

- Almost half of the 30 organizations interviewed reported that matrix organization variations are in effect; 73% of the respondents reported satisfaction with their current organizational arrangement.

- The effectiveness of people management within information systems organizations is mediocre, if not downright poor, according to the respondents.

  - Fifty percent rated their company's human resource management as just average, while 20% more thought it was below average.

- The three most important factors in motivating EDP professionals, according to the respondents, are:

  - Recognition by peers and management.

  - Congenial working environment.

  - Money.

- Innovative work schedules, such as flextime and work at home, are effective in retaining "superstar" performers and are in use in 70% of the organizations INPUT interviewed.

- Fifty-three percent of these organizations reported that the rate of turnover in their EDP personnel was below the average for the surrounding area.

B. GETTING THE USER INVOLVED

- There is little disagreement that obtaining the involvement of the end user in system development is important. The absence of such involvement is often responsible for unsuccessful and unsatisfactory conclusions to projects.

- In many instances, bureaucratic procedures such as reviews and sign-offs are regarded as a satisfactory manifestation of user involvement.

- In fact, effective user involvement means the cultivation of an emotional commitment on the part of the user to the promotion and support of the system, relative to both end user personnel and top corporate management.

- Such commitments require that the users be given real control over the content and direction of the project, including such steps as:

  - Giving the project leadership to a member of the end user department.

  - Including end user representatives as permanent members of the development team.

  - Looking to the user to perform the cost/benefit analysis for the proposed project.

- In situations where projects are being developed for several clients, there is danger that the EDP department might find itself embroiled in interdepartmental conflicts.

  - Rather than assume the role of mediator, for which the EDP department is not particularly well equipped, the suggested course in such situations is to assume the role of technical adviser and clarifier of technical alternatives, thereby capitalizing on the information systems department's strengths.

- A steering committee can be useful in resolving conflicts that cannot be worked out in other ways, but its main function is to set priorities and assure that proposed projects' cost/benefit analyses are compatible with the business goals of the organization.

C. OTHER NON-TECHNICAL ISSUES

- The successful information systems executive cannot and should not rely totally upon technical skills. Frequently, situations arise that can only be classified as "political" in that they threaten the status quo.

- More than any technical or technological factor, the totality of human interactions, motivations and behavior that comprise the term "politics" have been the primary influences on the course of system development.

- The EDP executive who is aware of the situations which contribute to, or cause, political problems is in a position to avoid or control them.

- Equally important is the EDP executive's ability to control and resist the "technical adventure" syndrome, which masquerades as creativity, but in fact is best described as a lack of discipline.

- "Technical adventures" need not be universally shunned, but if undertaken ought to be recognized in advance for the risk they represent.

D. MANAGING THE TECHNICAL SYSTEM DEVELOPMENT EFFORT

1. PROJECT LIFE CYCLE

- The life cycle of major systems averages 7.5 years, according to the 30 organizations INPUT interviewed.

- There is a wide divergence in the definitions of "small," "medium" and "large" projects.

- The results of the survey showed, however, that most organizations define "small" and "medium" projects in more or less the same terms, and that consequently the meaningful distinction is between "large" projects (generally longer than one year) and all other kinds.

- More than half of the 30 respondents agreed that small projects required a less extensive development phase relative to the total project life cycle.

- The examination of alternate system solutions can benefit from:

  - Greater use of existing systems and their components (reusable code), as well as application generator systems for those limited situations to which they are best suited.

  - A more searching cost/benefit analysis that takes into account non-computerized components of the solution.

2. PLANNING

- One major source of unsatisfactory projects is the failure to define, early in the planning cycle, the specific criteria that define:

  - When the project is technically complete.

  - When the project is successful from the user's point of view.

- "Target drift," a common problem in EDP projects, is the gradual deviation from the initial goals and specifications brought about by a variety of user-driven, as well as information system department-initiated, pressures.

  - Settling on a well-defined acceptance test early in the planning stage will be useful to minimize "target drift" tendencies.

- Objectively measurable criteria for technical completeness and business success of the project also provide some protection from this drift.

- The competence and experience level of the actual hands-on end users must be taken into account. They should not be expected to have the same level of systems expertise as the designers.

- Careful planning for the cutover to the new system from the old system is essential.

    - It is generally not a good idea to keep the old and new systems in operation side by side, since this creates an incentive not to use the new system.

3. SYSTEM DEVELOPMENT METHODOLOGY (SDM)

- An SDM, which is a comprehensive checklist of activities and procedures to be performed in each phase of the project's cycle, is desirable for many reasons:

    - To make sure that no tasks are ignored in the planning and estimation phases.

    - To provide "kill points" where wayward projects can be stopped and associated losses minimized.

    - To improve the speed with which new hires come up to productive levels.

    - To provide a mechanism for regulating outside contract work when required.

- Drawbacks associated with the use of an SDM include:

- The great amount of "front end" work required without "tangible" output. This characteristic can be unsettling to both user and top management, unless both are properly indoctrinated in advance.

- The customizing that a commercially available SDM generally requires before it becomes fully usable in a given environment.

4. MEASUREMENT AND ESTIMATION

- "Progress reporting" is often confused with "measurement," but many organizations are not employing even a basic progress reporting system.

  - In INPUT's survey, 53% of the organizations queried indicated that they did not employ any measurement techniques, and 57% reported the lack of a formal project control system.

- What is missing from current progress reporting systems is the firm linkage between effort expended and actual completion of tasks. This compounds the problem of inaccurate estimates.

- Estimation techniques in all forms are basically attempts to predict future behavior based on past experience.

  - Feedback from measurements of past activities can greatly improve the quality of future estimates.

- Two techniques for improving the estimation process are:

  - A "standardized module" approach, in which proposed systems are decomposed into standardized functions that have been previously measured or estimated.

  - A "fixed-size-module" approach, in which proposed systems are decomposed into modules of approximately equal size, regardless of function. Automatically measurable criteria are established for the state of completion of each module. By tracking the number of modules in various completion stages relative to the total number of modules in the system, a firm idea of actual progress is obtained.

5. DEVELOPMENT TOOLS AND AIDS

- A large number of tools, aids and techniques are available to improve the system development process, as shown in Exhibit II-2.

- Especially worth investigating are the following:

    - Inquiry languages associated with most DBMS systems, which can substantially reduce the EDP department's backlog, especially if the user can employ them directly.

    - Applications generators that can be used for "quick-and-dirty" short-lived systems, or for demonstrating the dynamic behavior and characteristics of proposed large systems.

6. MAINTENANCE

- About half of the system life cycle is dedicated to maintenance, according to the organizations INPUT interviewed.

    - Thus anything that can be done to improve the maintenance phase will have a significant impact on total life cycle costs.

- Unfortunately, there is a scarcity of support tools and aids in this area, and the current management practice of assigning new hires to maintenance activities reinforces the view that maintenance is "second-class" duty.

- Various techniques to alleviate this problem are available, including:

EXHIBIT II-2

CLASSES OF SYSTEM DEVELOPMENT TOOLS AND AIDS

- SYSTEM DEVELOPMENT METHODOLOGY
- STRUCTURED ANALYSIS AND/OR DESIGN METHODOLOGY
- DEVELOPMENT TEAM ORGANIZATION
- TESTING AND REVIEW PROCEDURES
- MAINTENANCE TECHNIQUES
- ON-LINE DEVELOPMENT TOOLS
- APPLICATION GENERATORS (INCLUDING "NON-PROCEDURAL" OR "MENU-DRIVEN" LANGUAGES)
- REUSABLE CODE
- REQUIREMENTS LANGUAGES
- CODE GENERATORS/PSEUDO-CODE SYSTEMS
- INQUIRY/RETRIEVAL LANGUAGES
- STRUCTURED HIGH-LEVEL LANGUAGES
- PROJECT CONTROL SYSTEMS

- Training new hires in maintenance, rather than in new development.

- Periodic rotation of development personnel through the maintenance activity.

- Assigning maintenance responsibility to the developers.

- Use of such incentives as work-at-home to make maintenance more attractive.

- Forming SWAT teams to deal with both routine and critical maintenance problems.

- Use of the "scheduled" maintenance concept.

E. RECOMMENDATIONS

- Recognize the importance of political and people-oriented factors in systems development.

- Make sure that the information systems executive and his or her key people are given the opportunity for training in people-oriented skills such as negotiating, persuading, listening and mediating.

- Secure user involvement by giving the user real control of project direction.

- Seek to gain the user's emotional commitment to the promotion and support of the project.

- Capitalize on the information systems department's reputation as the repository of technical expertise and skill. Be a provider of unbiased technical advice and a clarifier of technical alternatives, not an adversary.

- Introduce tools and aids carefully and at the right time. Be sure to understand where each tool applies.

- The first time a system development methodology is employed, be prepared, and prepare users and management, for a larger "front-end" effort than previously experienced.

- Be prepared, and prepare users and management, for learning-curve problems associated with new technologies.

- Establish "measurements" rather than "reports." If possible, use unobtrusive, automated means to collect them.

III MANAGEMENT ISSUES IN SYSTEM DEVELOPMENT

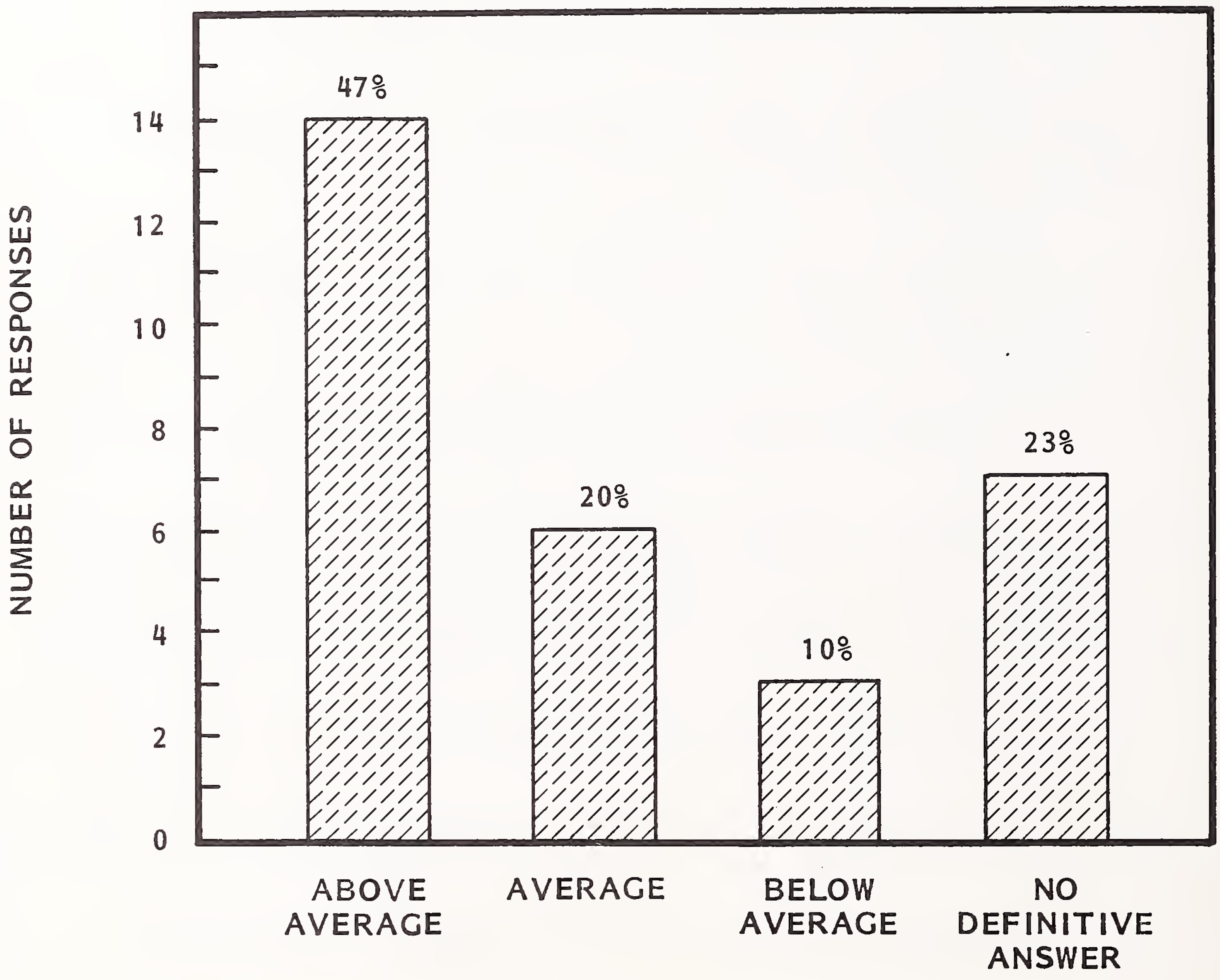
III MANAGEMENT ISSUES IN SYSTEM DEVELOPMENT

A. CURRENT CONDITIONS

- There is a widely held belief that management of system development in most organizations is in trouble.
  - Too many projects continue to experience significant schedule and budget overruns.
  - Too many projects fail to meet the users' expectations when delivered.
- In view of this, it is rather surprising that a plurality (14 of 30) of this study's respondents rated the effectiveness of their organization's management of information system development "above average," as shown in Exhibit III-1.
  - Only six respondents rated themselves "average."
  - Only three admitted that their EDP management was "below average."
  - A significant number (7) declined to respond to this question.
- Exhibit III-2 summarizes the respondents' opinions of the three most important things management should do to improve EDP systems development. Included in the response categories were the following specific items:

EXHIBIT III-1

RESPONDENTS' SELF-EVALUATION OF THE EFFECTIVENESS OF THEIR INFORMATION SYSTEM DEVELOPMENT MANAGEMENT



TOTAL RESPONSES: 30

EXHIBIT III-2

RESPONDENTS' OPINIONS OF THE THREE MOST IMPORTANT THINGS MANAGEMENT SHOULD DO TO IMPROVE EDP SYSTEM DEVELOPMENT

FACTOR	NUMBER OF TIMES THIS FACTOR WAS CITED AS:			TOTAL NUMBER OF TIMES MEN-TIONED
	PRIORITY ONE	PRIORITY TWO	PRIORITY THREE	
PRODUCTIVITY TOOLS, AIDS AND TECHNIQUES	9	4	3	16
MANAGEMENT RESPONSIBILITIES	3	8	4	15
END USER INVOLVEMENT	6	6	1	13
REQUIREMENTS AND PLANNING	4	4	4	12
PERSONNEL ISSUES	5	1	3	9
EDP EDUCATION	1	4	-	5
MORE RESOURCES	-	-	2	2

TOTAL RESPONSES: 30

- Productivity tools, aids and techniques:

  - System design methodologies.
  - Project control.
  - Reusable code.
  - High-level language.
  - DBMS.
  - Automated documentation.
  - Consistent style.
  - Structured techniques.
  - Reduction of rewrites.

- Management responsibilities:

  - Educate management in EDP.
  - Get management involvement.
  - Set clear goals.
  - Set realistic schedules and track progress.
  - Better allocation of costs.
  - Change organization.

- Use decentralized development.
- Reduce management turnover.
- Keep procedures and interfaces well defined.

- End user involvement:

  - More user involvement.
  - Train user in EDP systems.
  - Get user-friendly, simple languages.
  - Better communications with user and user management.
  - Define user responsibilities.

- Requirements and planning:

  - Better method of stating requirements.
  - Better status-reporting system.
  - Periodic review of requirements to establish continued need.
  - Watch cost-effectiveness.
  - Long-range planning.
  - Better planning.
  - Better review process.

- Personnel issues:

  - More challenging work.

  - Hire and retain better-qualified people.

  - Better selection of project leaders.

  - Better pay.

- Involvement of low-level people:

  - Provide state-of-the-art facilities.

- EDP education:

  - In general.

- More resources:

  - More dollars.

  - More computing equipment.

  - More people.

- It is highly interesting and significant that the use of tools and aids was not only the most cumulatively mentioned factor, but was also rated as the highest-priority item the largest number of times.

  - This concern with technological innovations is evidently due to the widely held, but mistaken, belief (or hope) that such technical solutions will cure all problems, including those essentially nontechnical in nature.

- Elsewhere in this report, some of the more significant tools and aids currently available are discussed. The information systems executive must recognize, however, that:

  - Such tools can be effective in improving certain aspects of the system development process - sometimes dramatically - but they are first and foremost ad hoc technical artifices.

  - This means that these tools and aids cannot be expected to solve organizational or people-oriented problems and conflicts.

  - Neither should they be expected to solve all technical problems unless current system development practices are changed.

  - Finally, the introduction of such tools into the organization must be handled carefully and with foresight. Belatedly trying to apply a new tool to a problem once the development process is already mired in serious difficulty generally results in worsening the situation, rather than improving it.

- Despite the fact that 83% of those who responded to INPUT's questions were managers with at least one level of management under their supervision, they overwhelmingly identified various management failures as the second most cumulatively cited area for improvement, and as the area which rated second priority the largest number of times.

- While the specific complaints cited against management may not be universally acknowledged as most pressing, the perception that EDP management (and corporate management) is largely responsible for many of the current problems in EDP systems development is shared by many independent observers as well as by the managers themselves.

- Information systems managers are often criticized - and rightly so - for ignoring the basic management principles of planning, organizing, staffing, executing, measuring and correcting.

  - Rather than apply management techniques, EDP managers frequently become overly involved in the supervision of technological processes and machinery.

- The information systems executive, like any other effective manager, must be much more than a skilled technician.

  - He must be concerned first and foremost with the management of people.

  - He must assign a high priority to obtaining end user cooperation and commitment.

  - These are essentially nontechnical issues to which technology offers few, if any, solutions.

- The third factor the respondents singled out for improvement was user involvement in the system development process. This factor received the third largest cumulative number of mentions (13).

  - This ranking matches a widely held view that the failure to obtain user participation is responsible for many, if not most, unsuccessful systems.

  - Probably the most common mistake EDP executives make is to regard user involvement as a technical or administrative problem that can be addressed by technological or administrative solutions.

- In fact, persuading users to be "involved" in the sense that they develop strong commitment to the support and promotion of a project is one of the primary people-oriented issues that the EDP executive faces. The next chapter is devoted entirely to this subject.

- The remaining areas that need improvement according to the organizations INPUT interviewed (requirements and planning, personnel issues, EDP education and resource issues) are also treated elsewhere in this report.

- It is interesting and instructive to note that, when asked if the areas singled out for improvement were actually being addressed by management, 25 respondents (83%) answered with an unqualified "yes," as illustrated in Exhibit III-3.

  - Three more (10%) replied "yes," but qualified their answer with various caveats.

  - Only two (7%) felt that the areas that needed improvement were not being effectively addressed by management.

- This high degree of self-satisfaction is in rather sharp contrast to the far less sanguine views of independent industry observers and does not correspond to the state of affairs described for specific problems even in this report.

  - In short, many managers appear to be kidding only themselves.

B. ORGANIZATION MANAGEMENT

- INPUT's questioning reveals in Exhibit III-4 that almost half the interviewed organizations (14 of 30) practice some form of matrix organization; i.e., a combination of the traditional, functional departments with project-oriented, ad hoc teams.

EXHIBIT III-3

EDP MANAGEMENT SELF-EVALUATION OF WHETHER OR NOT IMPROVEMENT STEPS ARE BEING TAKEN

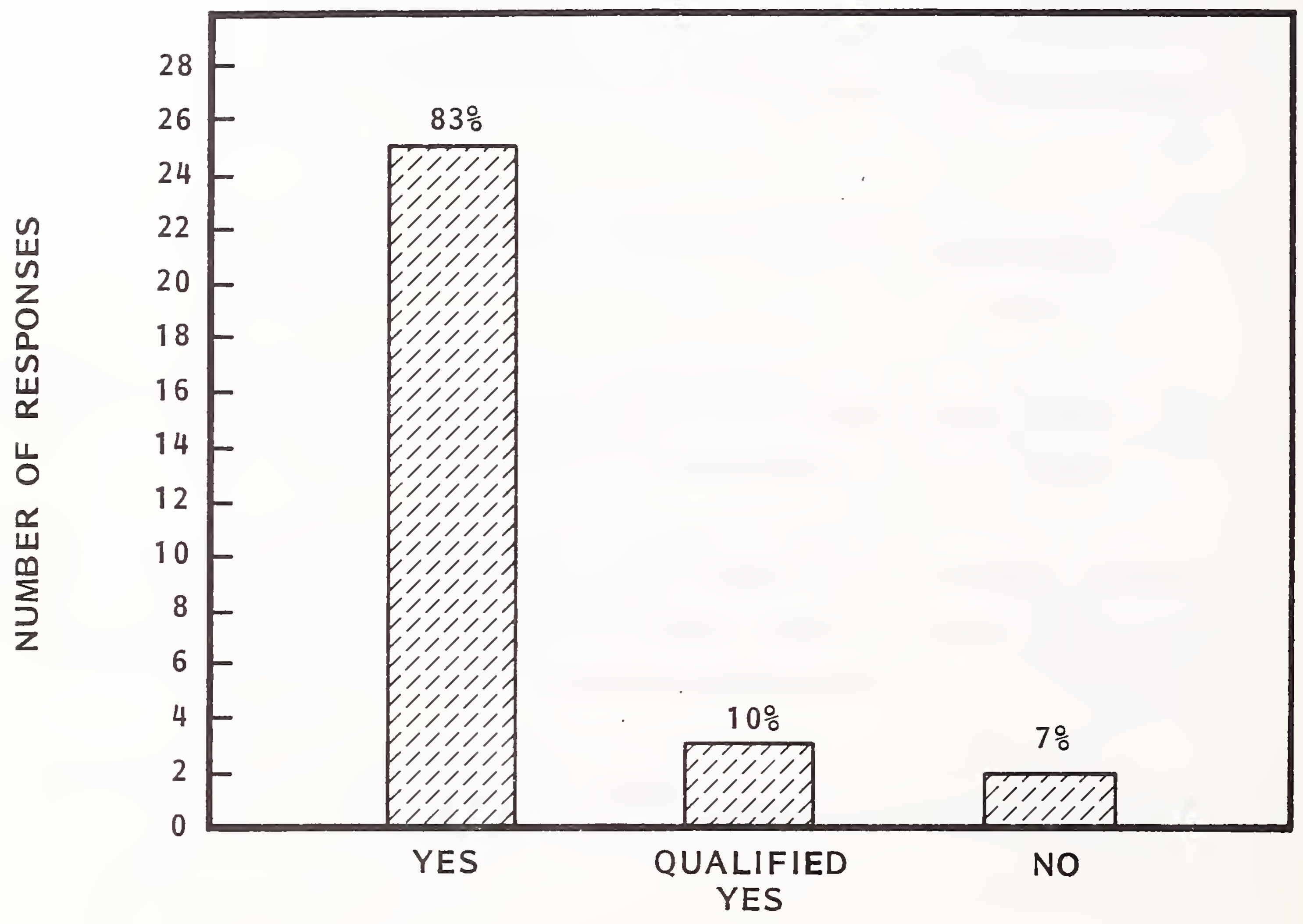
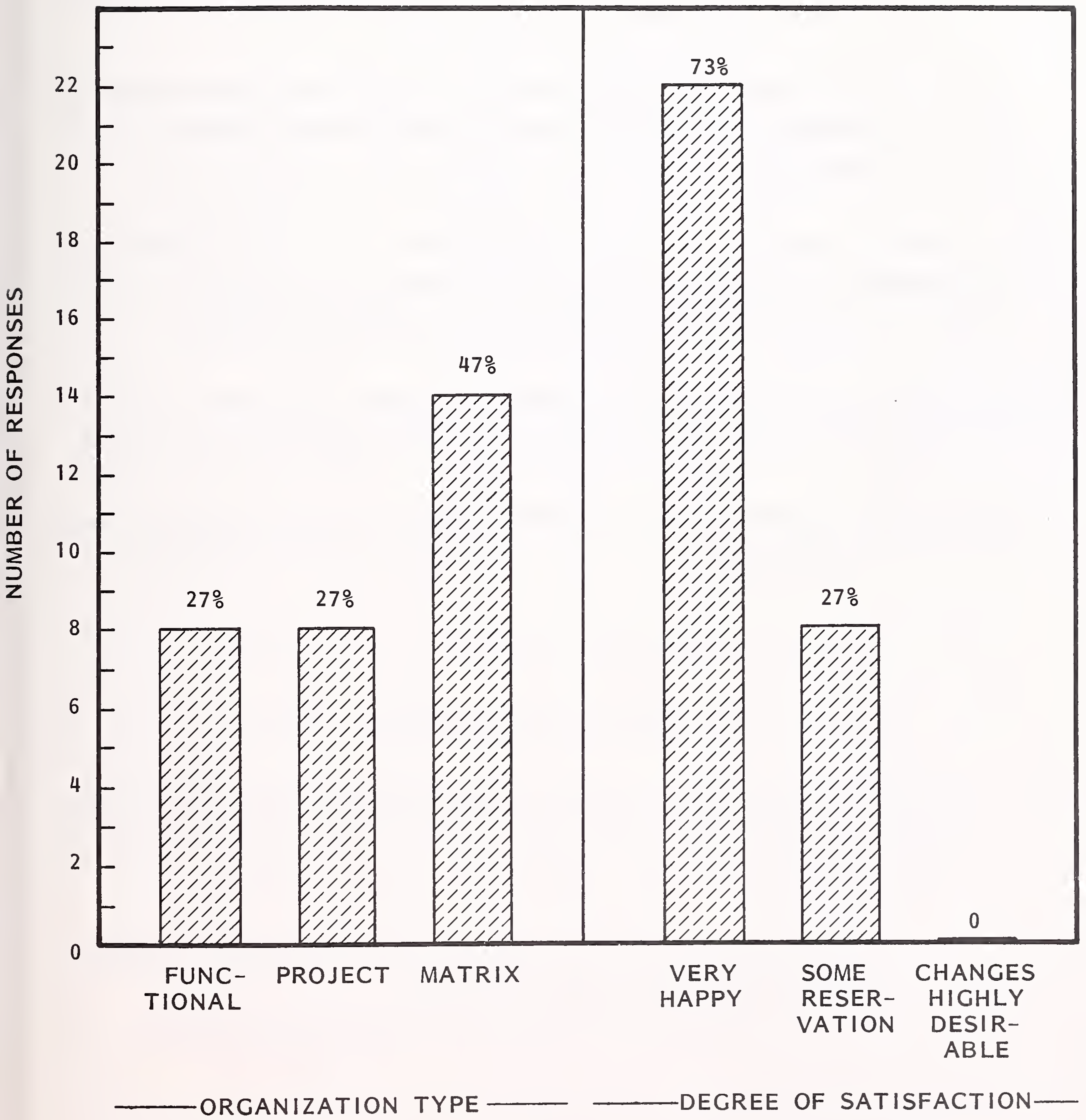


EXHIBIT III-4

CHARACTER OF THE EDP ORGANIZATION AND DEGREE OF SATISFACTION WITH IT



TOTAL RESPONSES: 30

- Eight respondents said they were purely functionally oriented.

- A surprisingly large number of respondents (eight) indicated that they were organized in a purely project-oriented fashion.

• A number of the respondents observed that a functionally oriented organization was more efficient when the environment was relatively stable; i.e., when the number and magnitude of new projects were limited.

    - But if large projects occur frequently, the resulting personnel borrowing and shuffling turns the functional organization into a de facto matrix organization.

• Another respondent in a matrix organization noted that personnel are borrowed to accommodate large projects, but that in periods of relative inactivity, project staff is turned over to the maintenance department.

• Again, as in other areas, the respondents indicated a high degree of satisfaction with the present situation as far as their EDP organization was concerned.

    - The majority (22) stated that they were very happy with their method of organization.

    - Only eight expressed some reservations.

    - None felt that a change was highly desirable.

• Yet several respondents noted that a reorganization had just taken place, while one reported that his organization underwent three such reorganizations within the span of four years.

    - INPUT's impression is that reorganizations are relatively frequent, judging from the sample used in this study.

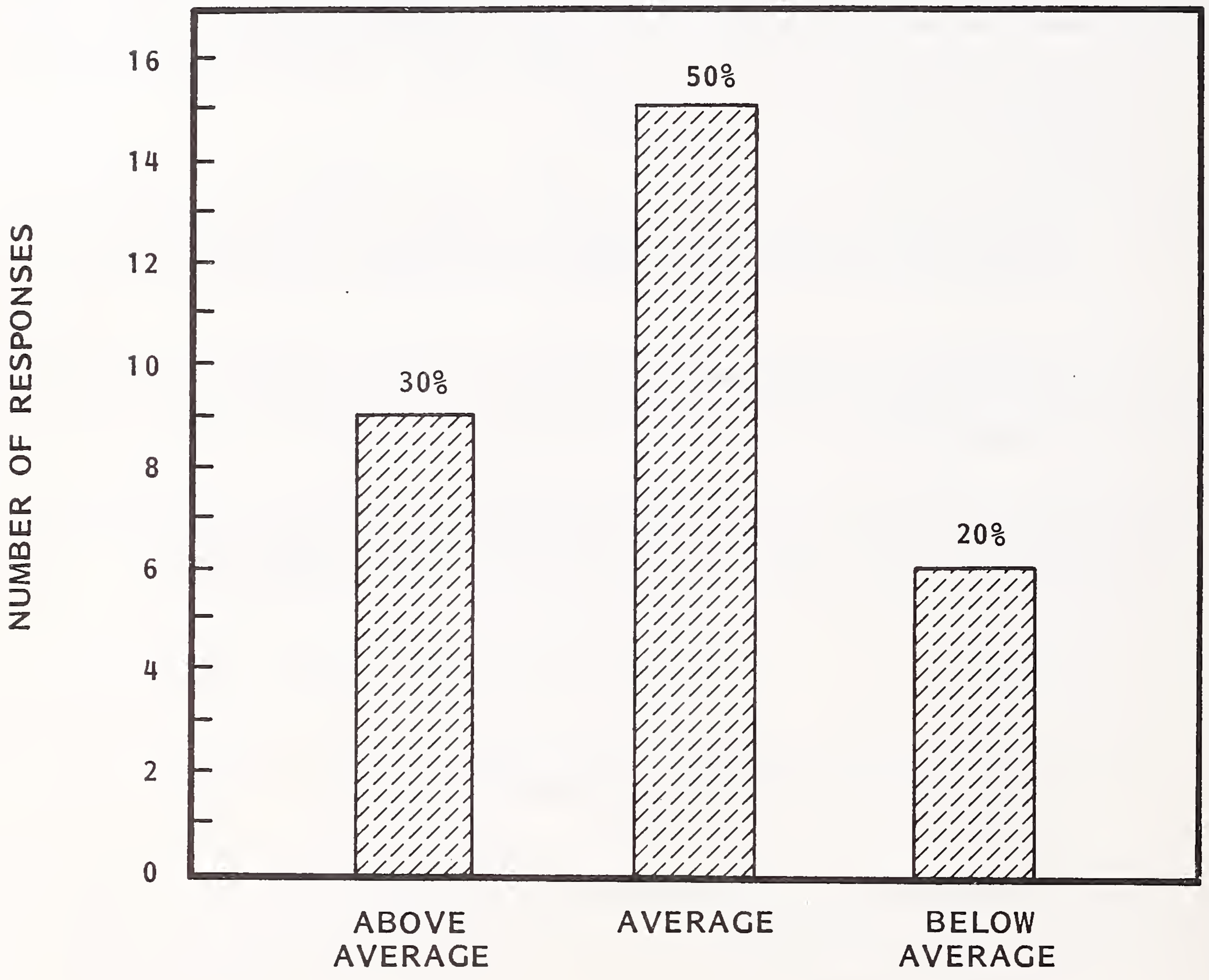
- This would contradict the respondents' claims that by and large, they are happy with their present arrangements. If such happiness was truly prevalent and shared by all management levels, the need for reorganization would be less pronounced.

C. PEOPLE MANAGEMENT

- That people management is largely mediocre, and perhaps downright ineffective, was brought out strongly in the responses INPUT received when respondents were asked to rate the effectiveness of their company's human resource management. Exhibit III-5 reveals that:
  - Fifty percent of the respondents rated their company's people management as only average, while 20% more thought it was below average.
  - Only 30% thought their company was above average in its people management.
- What motivates EDP people? INPUT asked the respondents to assign relative importance to four specific factors and an open-ended "other":
  - Money.
  - Congenial physical working environment.
  - Recognition by peers and management.
  - Company identification (loyalty).
  - Other.
- Exhibit III-6 summarizes the responses to this question.

EXHIBIT III-5

EFFECTIVENESS OF HUMAN RESOURCES MANAGEMENT



TOTAL RESPONSES: 30

EXHIBIT III-6

RELATIVE IMPORTANCE OF MOTIVATING FACTORS
FOR EDP PERSONNEL

FACTOR	NUMBER OF RESPONSES CITING THIS FACTOR AS:			TOTAL NUMBER OF TIMES CITED
	PRIORITY ONE	PRIORITY TWO	PRIORITY THREE	
RECOGNITION BY PEERS/ MANAGEMENT	15	12	3	30
CONGENIAL WORKING ENVIRONMENT	6	19	5	30
MONEY	4	2	24	30
COMPANY LOYALTY	1	9	20	30
CHALLENGING, INTERESTING WORK	(NOT PRIORITIZED)			9
CAREER PATH OPPORTUNITIES	(NOT PRIORITIZED)			8
AVAILABILITY OF STATE-OF-THE-ART TOOLS	(NOT PRIORITIZED)			6
SELF-ESTEEM, FEELING OF PROFESSIONAL ACCOMPLISHMENT	(NOT PRIORITIZED)			4
TRAINING	(NOT PRIORITIZED)			2

- Half of the respondents identified "recognition by peers and management" as the most important factor; 90% gave this factor either first or second priority.

- Over 60% classified "congenial working environment" as the second priority factor; 20% more thought it should be the first priority.

- Eighty percent of the respondents agreed that money was the third priority factor.

- "Company loyalty" was placed as the third most important factor by 67%, while 30% more thought it was second in priority.

• Based on these responses, the following order of significance appears justified:

  - First: Recognition by peers and management.

  - Second: Congenial working environment.

  - Third: Money.

  - Fourth: Company loyalty.

• The respondents also volunteered a number of other factors, listed as the fifth to ninth factors in the exhibit. These include such factors as:

  - Challenging, interesting work.

  - Career path opportunities.

  - Availability of state-of-the-art tools.

  - Self-esteem, feeling of professional accomplishment.

- Training.

- Numerous past inquiries have established that there is a very wide divergence in the productivity levels of DP analysts and programmers.

  - One "superstar" can produce 20 or 30 times the output of a run-of-the-mill individual.

  - The 80:20 principle seems to apply: 20% of the people deliver 80% of the results.

- The conclusions are fairly obvious:

  - Look for "superstars" when hiring.

  - Once secured, do all that can be done to keep them.

- Incentives that might appeal to "superstars" include:

  - Performance bonuses.

  - Income counseling program. It is surprising how many DP professionals are rather lacking in personal financial management know-how.

  - Customized fringe benefits program, if the corporation allows it. If not, it might be worth suggesting.

  - Permit the best performers to work partly or entirely at home and supply them with their own terminals.

  - Make the retention of the best performers (and minimizing of turnover in general) a part of your managers' measured goals on which their own bonuses or promotions depend.

- Anything that encourages employees to feel that to the maximum possible extent they are in control of their own destinies is of value for increasing job satisfaction.

  - While work-at-home and setting one's own hours should probably be reserved for best performers only, all personnel should be given a chance to exercise control, for instance, by having a voice in setting schedules for tasks assigned to them.

- "Bad apples" should be let go quickly, regardless of technical skills.

  - Not only do they expose the organization to unwarranted risks of poor performance, but their negative influence upon the morale of the rest of the staff is contagious.

- A large number of organizations queried by INPUT indicated that innovative work schedules such as flextime and work-at-home are in regular or occasional use, as illustrated in Exhibit III-7.

  - Forty-seven percent reported use of flextime.

  - Twenty-three percent reported work-at-home.

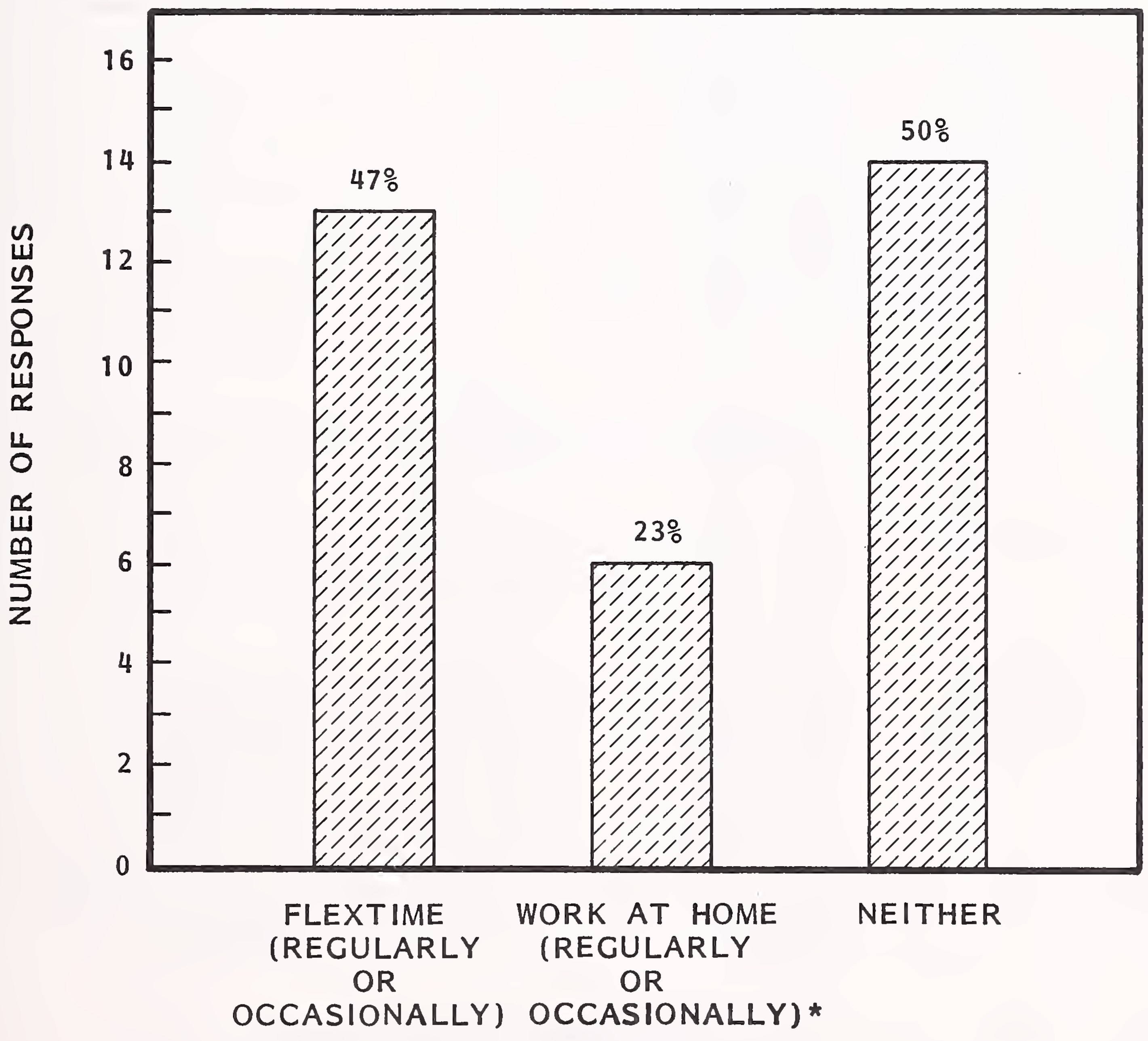
  - Fifty percent reported no use of either.

- Contrary to expectations, turnover may not be affecting the large organizations included in this study as severely as the published national average. The results summarized in Exhibit III-8 show that:

  - Fifty-three percent of the respondents felt that the rate of turnover of EDP personnel in their organization was below the average rate for the surrounding geographical area.

EXHIBIT III-7

INNOVATIVE WORK SCHEDULES

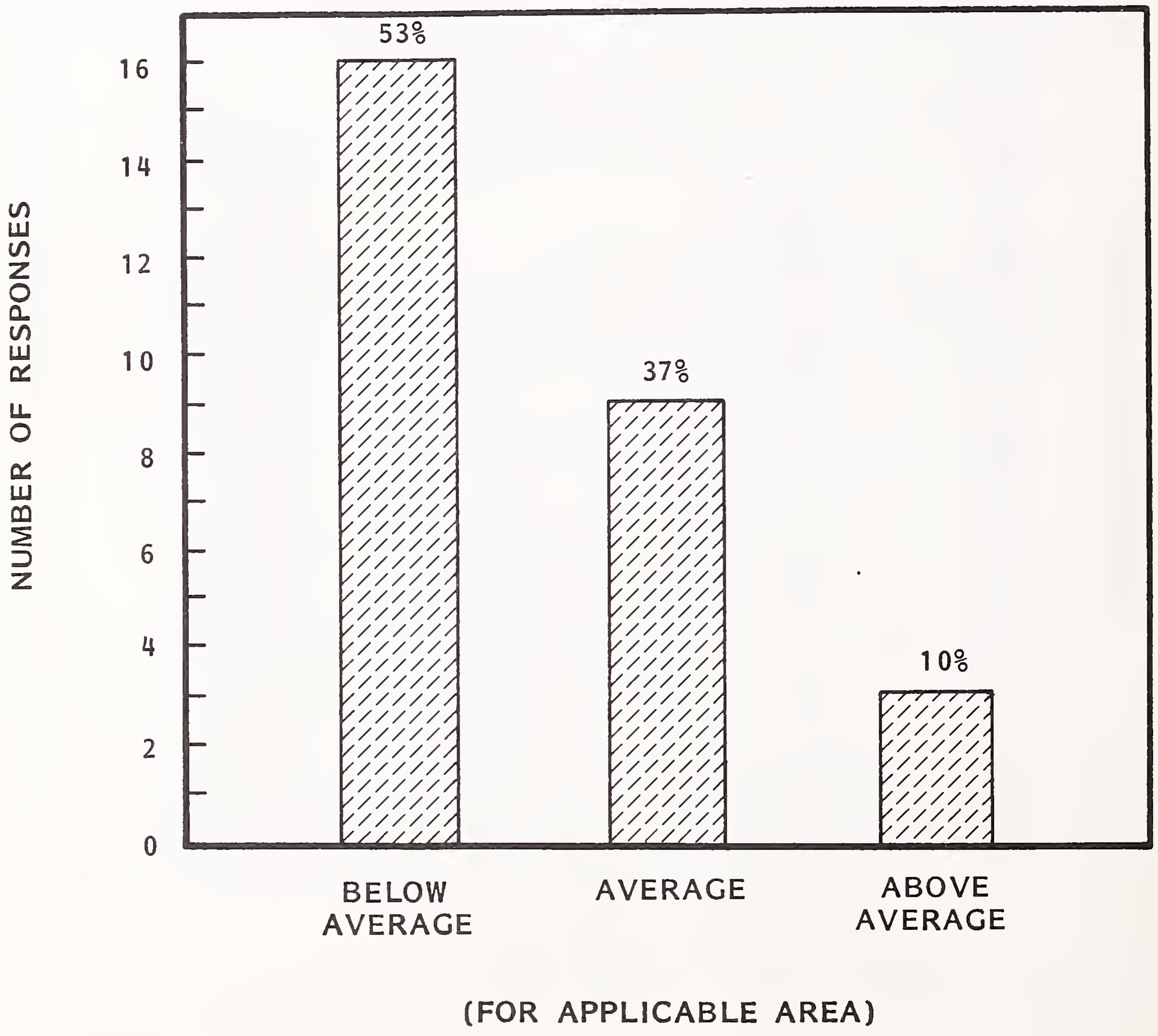


* ALL BUT ONE IN THIS CATEGORY ALSO EMPLOY FLEXTIME.

TOTAL RESPONSES: 30

EXHIBIT III-8

EDP PERSONNEL TURNOVER RELATIVE
TO RESPONDENTS' GEOGRAPHICAL AREA'S RATE



TOTAL RESPONSES: 30

- Thirty-seven percent thought their turnover rate was equal to the average prevailing in the area.

- Only 10% believed their turnover rate exceeded the local average.

• Nevertheless, the impact of turnover on the ability to bring projects to completion without overruns is rather sharp.

  - Exhibit III-9 shows how a 20% turnover rate leads to a 10% slippage in project completion date for a 240 person-month project.

  - Assumptions are a 50% efficiency level for the departing individual during the final month; a 50% efficiency level for the project leader in the following month; a 25% efficiency level for the new individual for the next two months; a 50% efficiency level for that individual during the second two months; and departures evenly spaced over the life of the project.

• Implications for the EDP executive are rather obvious.

  - Strive to reduce turnover by providing job satisfaction through proper motivation (as discussed elsewhere in this section).

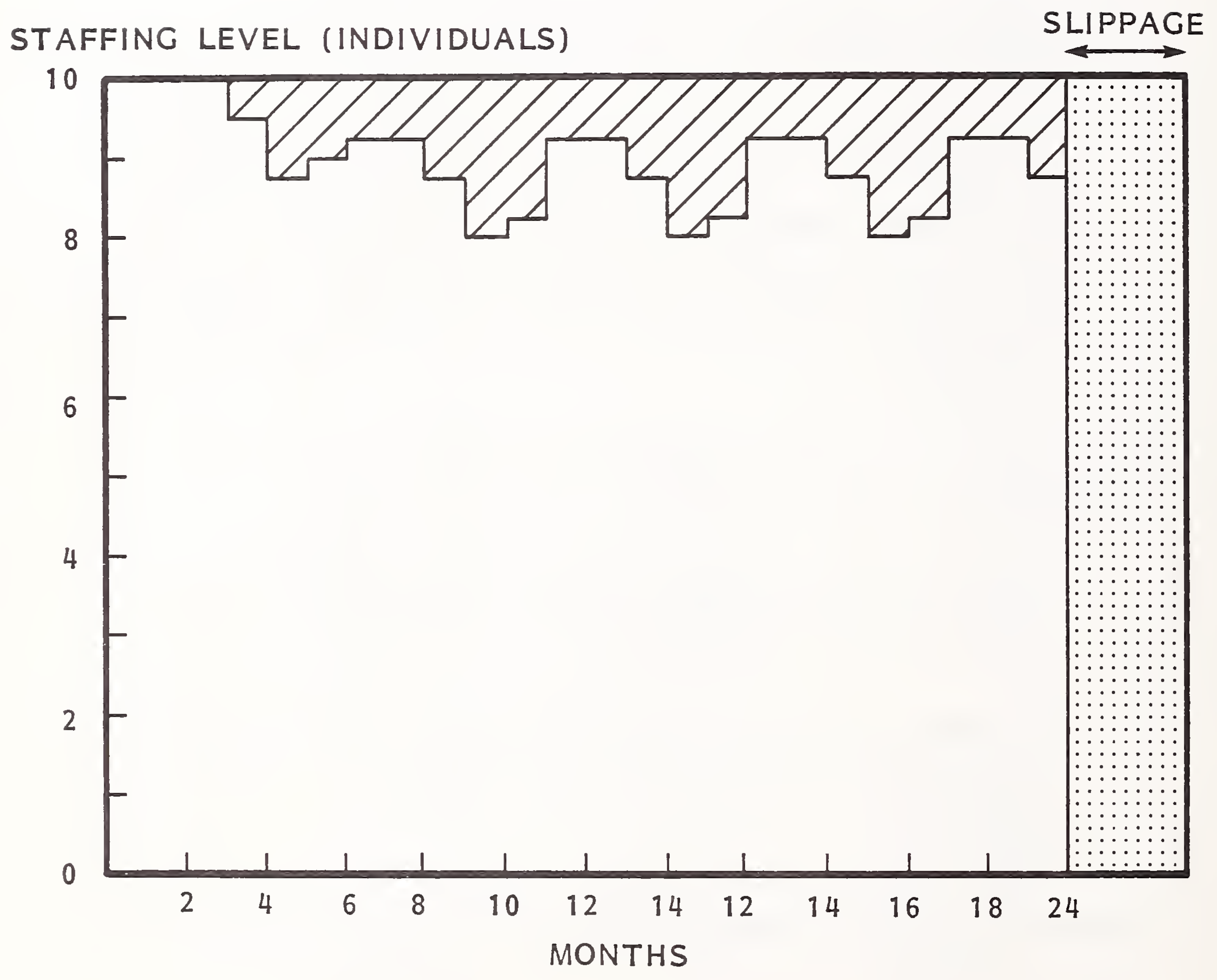
  - If nothing can be done to reduce turnover substantially, then the impact on project schedules must be calculated and included in the project plans.

• One way to minimize external turnover is to place fewer restrictions on internal transfers.

  - Some managers feel that employees who attempt to arrange internal transfers for themselves by conversing directly with other departments' managers should not be permitted to make such moves, as punishment for "bypassing channels."

EXHIBIT III-9

MINIMUM IMPACT OF 20% TURNOVER



TOTAL PROJECT = 240 PERSON-MONTHS
MINIMUM IMPACT = 25 PERSON-MONTHS
SLIPPAGE = 10.4%, OR 2.4 MONTHS

▨ EFFORT LOST
⬚ ADDITIONAL EFFORT REQUIRED

- This can only lead to the employee in question leaving the company altogether. It would be better to allow the internal transfer.

- In fact, some companies intentionally formalize such transfers by creating computerized "skill banks" in which the qualifications of all employees are kept.

    - When job openings occur, the internal "skills bank" is searched first.

    - Whether the job is filled by hiring, internal transfer or contracting out is determined jointly by the "human resources" department in conjunction with the other department managers.

    - Skills banks give the employees the feeling that their qualifications and skills are known and acknowledged within the organization. This minimizes one of the main reasons for job dissatisfaction - the feeling that "no one knows what I do, nor cares."

- Exit interviews may not be very effective, but they are the only practical way to get any kind of indication as to the reasons for the turnover.

- Contract programmers and analysts (usually calling themselves "consultants") present a special problem.

    - Their use is often preferable to hiring full-time staff who must remain idle or do "make-work" tasks when project activity subsides.

    - However, contract personnel usually command far higher straight salary than full-time employees doing equivalent work.

    - Despite explanations that such high pay is actually cheaper for the organization when fringe benefits, overhead and forced idle time are taken into account, full-time employees still resent the apparent disparity.

- One way to minimize the problem is to assign contract programmers the self-contained or standalone tasks that can be largely accomplished by working independently and off-site.

- If a satisfactory system development methodology (SDM) is in use, the necessity for spelling out in detail the organization's standards and procedures to the contract group(s) is greatly minimized. They can simply be given the relevant portions of the SDM documentation to guide their efforts.

- A few organizations have found that having their own captive professional services organization provides a cost effective alternative to retain those employees who would rather give up fringe benefits for a higher cash salary.

- The commonly held belief that DP personnel, especially analysts and programmers, require privacy and isolation in order to perform well does not seem to have any clear experimental substantiation.

    - To the contrary, there are indications that programmers can perform quite well even in a typical manufacturing company's "bullpen," when such other conditions as recognition of talent by peers and management offset the negative effects of the physical environment.

    - The feeling that the work performed is state of the art on advanced equipment can also largely offset poor office facilities.

IV GETTING THE USER INVOLVED

IV GETTING THE USER INVOLVED

- The preceding chapter identified the importance of strong, sustained user involvement in achieving a successful information system implementation.

- Measurement of the degree to which this objective is achieved is not really possible in that it cannot be statistically validated, and INPUT did not ask this specific question of the respondents. But almost all respondents (27 of 30) volunteered this information in response to a question dealing with the related subject of users' ability to solve EDP problems on their own. Exhibit IV-1 summarizes this information.

- While 18 of the 27 respondents indicated that users are involved in the specifications phase, or are asked to approve designs via a formal signoff procedure, or participate in final system test and approval, it is significant that in only two cases were users said to be active members of the development team.

  - There was only a single instance in which a member of the end user department was reported to lead the development team in large projects.

- These results generally confirm the widely held assessment that end user participation in the system development process is woefully inadequate.

EXHIBIT IV-1

USERS' INVOLVEMENT IN SYSTEM DEVELOPMENT

DEGREE OF INVOLVEMENT	NUMBER OF RESPONSES*	PERCENT*
SPECIFICATIONS PHASE ONLY, OR SIGNOFF PROCEDURES; FINAL ACCEPTANCE	18	67%
FREQUENT INTERACTION WITH DEVELOPMENT TEAM OR GROUP	4	15
USERS ARE ACTIVE MEMBERS OF THE DEVELOPMENT TEAM	2	7
USERS LEAD THE DEVELOPMENT TEAM ON LARGE PROJECTS	1	4
ON-LINE USER TOOLS PROVIDED	5	18

* THREE RESPONDENTS CITED TWO TYPES OF INVOLVEMENT EACH.

TOTAL RESPONSES: 27

- Lack of such end user participation has been identified as a major factor in the failure of EDP systems to achieve satisfactory acceptance.

- Most EDP executives vigorously subscribe to the concept that user involvement is highly desirable, even mandatory, for successful system development. Relatively few such executives recognize that persuading the user to become involved is, in fact, a political and educational problem, not technical or administrative.

- As a result, the techniques typically relied upon to secure user involvement are technical or administrative in nature:

  - Inviting or requiring the user's representatives to attend design review meetings.

  - Establishing signoff procedures that present end users with a stack of design documents and expect the user to "get involved" by signing them, often without understanding even how to interpret them.

- These techniques will not be effective unless the diplomatic aspect of obtaining user involvement is first recognized and addressed.

- True user involvement requires an emotional commitment on the part of the user to support and promote the development of the system.

- This emotional commitment cannot be fully achieved as long as the user views the system as a product or service which is purchased from a "vendor," namely the EDP department.

  - Neither can this commitment be achieved effectively by requiring the user to attend meetings and sign off on specification sheets and design documents.

- The most effective way to establish a powerful, emotional commitment is to create the conditions that make it possible for users to view the systems as their own.

- Giving the end user a feeling of "ownership" of the system being developed eliminates a deep-rooted fear:

  - People are fearful of situations in which they have little or no control over their own destinies, because such situations threaten the basic need for security.

  - If such fears are left unaddressed, the invariable reaction will be to resist the system development, or at least to refrain from active support of it.

  - "Being in the driver's seat" is the best way to counter such fears.

- Unfortunately, in an environment where information system resources are under the exclusive control of a centralized EDP department, which is chartered by the corporation to supply the personnel and technical expertise for the rest of the corporation, it is almost impossible for users to feel that they truly "own" an EDP system.

- It is, however, quite possible to achieve an environment in which the end user exercises real control over a substantial portion of the development process, thereby cultivating a real commitment on the user's part to the promotion and successful acceptance of the system.

  - The leadership of the project team charged with the task of developing the system can and should be entrusted to a user.

  - The necessary technical expertise can be supplied by an EDP professional, acting as "second in command."

- The end user department that has the leadership of the project in its hands has a real, tangible stake in the success of the project.

  - It is less likely to develop the "us against them" attitude in which problems arising during the project are blamed on the incompetence of the EDP department (not an uncommon situation when projects are led solely by the EDP department).

  - The end user department has a powerful incentive to promote the project actively when contention for resources needs to be brought before corporate management.

  - End user management also has a direct interest in selling its own staff (by training and otherwise) on the merits of the system being developed.

- Whether or not the project is actually led by an end user, it is important to have end user representatives assigned as full-time members of the project team.

  - They have important contributions to make during most phases of the life cycle.

  - Continuous interaction with the EDP members of the team can educate both types of personnel concerning each others' needs and concerns.

  - Full-time participation creates a continuing end user stake in the project and builds actual (as opposed to formal) commitment to the project.

- Even such a minor detail as where the project team holds its meetings may be significant. To the greatest extent possible, such meetings should be held alternately "on the end user's turf" and at the EDP department's facilities.

- The spirit should be one of cooperation and joint venture, not that of a buyer/seller relationship. The objective is to lead to stronger end user commitment, not to a "take it or leave it" attitude.

- However, one should be careful not to resort to artificial ploys as a substitute for meaningful user participation and control. Ploys of this type that have been used in the past include:

  - Intentionally presenting a deficient design at the initial stages, in order to allow the user to point out a substantial improvement. The effect is more likely to be the reinforcement of the opinion that the DP department is incompetent.

  - Telling the user the deficiencies he reported are "being taken care of," while attempting to convince him surreptitiously that the deficiency will not hurt him. This is likely to lead to loss of faith in the DP department's ethics.

- Routine reliance on ploys instead of substance does not serve the corporation or the information systems department well.

- Treating the end user with the same respect, courtesy and attentiveness as the corporation demands for handling external customers is always a good idea.

- User involvement in the technical and administrative process is essential to the project at several points of its life cycle.

  - The definition of business needs, the cost effectiveness analysis of the proposed system, and the prioritization of system goals and features are tasks that are best done by business analysts from the end user department supplying the business judgement.

- The information systems representatives should provide technical advice on EDP implementation alternatives and costs associated with each alternative.

- User management must, of course, be involved in major revisions to the plan or budget that result from this negotiated process.

- During the design phase, user project management, working with the ultimate end users, should pass judgement on such things as screen formats, reports and their formats, etc.

  - It is important at this stage to get inputs from the actual final end users, in order not only to account for their perspectives, but also to build up their emotional commitment.

  - Uncommitted end users at very low organizational levels (e.g., data entry clerks) can sabotage the success of systems by simply not being cooperative; for example, by not entering the input data correctly. Consistently re-entering transactions two or three times will destroy original performance specifications that were projected on much lower volumes.

- End users must clearly participate in the drafting of the implementation or cut-over plan, under which the steps taken to switch from the old system to the new one are worked out.

- Both during the design phase and final testing and acceptance phases, end users must be actively involved in the definition of details of testing, especially of the criteria for determining the system's success and its end-of-development point.

● Chronologically, the points at which specific end user activities are required are:

- In the planning stage for specifying business requirements and performing cost/benefit analysis.

- When the specifications are complete - a formal signoff at this point is desirable to minimize the tendency to request additional features during development.

- Whenever any major change is instituted during the development, to obtain agreement on the cost effective necessity for the change, and to agree to its impact on schedules and budgets.

- Just prior to implementation or cut-over from the old system, to certify that the system is ready for installation.

- At the completion of the implementation/cut-over phase, to declare the project formally complete. This marks the beginning of the maintenance phase.

- Ideally, if the management of the user department has developed the depth of commitment to the system that it would give to its own brainchild, the task of assuring user participation in the project and of successfully concluding the milestones listed above will be voluntarily assumed by the end user department.

  - End user management will then also assume the major responsibility for selling the system to its own people down the line to the actual hands-on end users.

- Information system executives and staff, particularly those with business backgrounds from other parts of the organization, must be careful not to overestimate their understanding of the user's business problems, concerns and needs.

- Even when the DP executive has past business experience in the organization, such experience may be outdated by years of growth and change. Of course, the same objection applies to the user liaison who is appointed to that function by default; i.e., the individual who will be missed the least in the user's day-to-day operation.

- Not having the day-to-day "flavor" of the business can detract from the ability to make proper design judgements.

- This is one clear area where responsibility should lie squarely within the end user's domain.

● Information systems executives should seize the opportunity to cultivate the users' commitment to the system by encouraging them to participate in the system's definition and implementation through their most informed and best personnel.

  - To the same extent, the best EDP analysts should be assigned to the specification function.

● Psychologically, giving end users an opportunity to make a contribution in an area in which they are especially competent will be both beneficial to the project and a great "ego builder."

  - Such "ego building" is essential to counteract the user's typical lack of expertise in EDP technology and terminology, which may lead to destructive cover-ups such as blaming all problems on the lack of skill of the EDP personnel.

  - Convincing the EDP staff that an attitude and approach of cooperation and joint venture is essential, without at the same time "caving in" to every user demand and whim, demands the highest application of managerial skill.

- It is INPUT's observation that EDP managers are rarely outstanding in their people management skills, with the result that systems flounder more often for non-technical rather than technical reasons, particularly in the requirements and specification phases.

  - The symptom may be technical inadequacy, but the cause is people-related.

- One of the more common phenomena during system development is the adoption of adversary postures by the user and the information system departments.

  - This breakdown in communication becomes especially acute when the project is not going well and the tendency to protect one's own turf by pointing an accusing finger at others is especially strong.

- The EDP executive should work to prevent this situation from happening by continually courting the user's support, and by capitalizing on the best assets of the information systems department, which are:

  - Technical knowledge and skill.

  - Ability to evaluate alternate technical approaches.

- The best posture for the EDP department is therefore that of a cooperative provider of technical advice and services.

  - Users should perceive the DP people as objective, technical advisors whose interest is to help the user evaluate technical approaches, rather than as an adversary camp whose interests are in conflict with the user's.

- Multi-user situations, in which several user departments are the beneficiaries of the eventual system, are especially prone to internal political pressures.

- In many cases, users' needs and requirements may not be compatible with each other, or may even be contradictory or mutually exclusive.

- Even in the absence of such conflicts, the multiplicity of diverse interests may make it difficult to reach consensus on system goals and features; users may simply not understand each others' problems, much less the EDP department's problems.

- Systems developed for multiple users often involve shared resources; for example, common data bases. Such questions as who owns what data and who is permitted each type of access may give rise to conflicts.

● A common mistake EDP executives make in such situations is to attempt to become mediators of the various conflicting interests.

  - Such attempts are generally motivated by the idea that smoothing over the difficulties will lead to faster development and less problems for the information systems department.

  - In reality, unless the EDP executive is especially adept at the political arts of persuasion and mediation, this attempt almost always ends up with the EDP department being blamed for the conflicts; or, worse still, the inadequately "papered over" differences among users manifest themselves later on in the life cycle, where their impact can be devastating or even fatal to the project.

● Generally, the EDP executive's strongest suit is technical knowledge and competence. Consequently, it is far better to capitalize on this quality by adopting the posture of provider of technical advice and clarifier of technical alternatives.

- A useful role model is that of the financial executive, whose objective is the well-being of the corporation, but whose sphere of advise and control centers on financial implications of alternate business decisions.

- Probably the best way to approach a politics-fraught, multi-user situation is to carry out discussions with each user separately.

    - In this way the EDP executive can gain an understanding of each user's perspective.

    - Even more important is the fact that in such "one-on-one" forums, the user is likely to be less guarded and more willing to discuss the real motivations, fears and wishes that drive the impersonal departmental "party line."

- Armed with such knowledge of each user's real needs and concerns, the EDP department can work out a proposal which represents a technical compromise between the various users' wishes.

- It is generally a mistake to try to present such a plan at a mass meeting of all concerned users.

    - Again, the best approach is to reveal the outline of the proposed solution to each user independently and privately.

    - Reactions to the plan can be discussed more completely and more candidly in such private meetings.

- The EDP representative should avoid taking sides, but should point out the technical implications of each conflicting demand.

- It may be necessary to go through several iterations of this process, until the outline of the solution emerges.

- By this time, the agreement of all concerned to the proposed solution will essentially have been secured already. A "mass meeting" at this time should be just a formality.

- If this approach still leaves unresolved major issues, the next line of attack that is likely to yield results is to try to break the large problems into their constituent elements, and attack each element separately if possible, again utilizing the one-on-one consultation approach.

    - This "divide and conquer" strategy regarding problem complexity is far more effective than attempting to gather a large committee to deal with the "fundamental issues."

    - "Dealing with fundamental issues" may be intellectually satisfying, but rarely results in reaching a consensus. On the contrary, attempting to attack large problems by brute force usually results in aggravating the conflict.

- Only as a last resort, when all attempts at persuasion have failed, should the issues in conflict be referred to a higher management level for resolution.

    - The steering committee is one handy mechanism for the resolution of such conflicts.

- Training is another important tool for obtaining meaningful user cooperation and support.

    - By explaining how the system works and how it is to be used, the fear of the unknown is greatly reduced.

    - This is a golden opportunity to do some subtle "selling" of the system by pointing out the benefits to the corporation, the user department and the end users personally.

- End user middle management should actively participate in the training by providing the necessary business background and by explaining how the new system fits with the corporate business goals.

  - The combined user-DP effort in the training program is an expression of the user-DP cooperation on the project itself.

- Ideally, training should begin long before the implementation phase.

  - Early training sessions can and should be used to obtain valuable "hands-on user's" perspective and suggestions.

  - Such sessions should therefore minimize the authoritarian teaching role and emphasize that the purpose is to listen and learn from the end user's comments.

  - These initial training sessions are also valuable for identifying early potential resistors and the concerns and fears that drive their objections.

- The EDP steering committee is a mechanism which is being rapidly adopted by an increasing number of organizations.

  - It is generally viewed as a mechanism for obtaining end user participation in the control of EDP projects and activities.

- There are generally two levels of such steering committees.

  - At the corporate vice presidential level, a steering committee is concerned with the cost/benefit analysis of projects, how well they support corporate business goals and strategies, and, in case of conflicts, which projects should receive priority.

- At the middle-management level, a steering committee establishes a forum for a cooperative working relationship between the users and the EDP department, and provides a mechanism for resolving problems and conflicts in the requirements, design and implementation phases.

- So much for the theory. In practice:

  - Steering committee meetings are often an ineffective substitute for meaningful user-DP cooperation.

  - Some managers substitute attendance at committee meetings for substantive management of their departments.

  - At the other end of the spectrum, some managers delegate attendance to subordinates who do not have sufficient authority to make decisions, thus wasting everyone's time.

- Steering committees are generally unnecessary when there are only one or two actual end users for the DP department's products and services, and when projects are infrequent or limited in scope.

  - In such situations, a far more effective approach would be to call ad hoc meetings of only the relevant people to deal with issues as they arise.

  - Formalizing their interaction in regular committee meetings leads to wasted time.

- Substantive issues that are properly the domain of a steering committee include:

  - Allocation of resources.

  - Establishments of priorities.

- "Kill points" - re-evaluation of ongoing projects to establish whether the need still exists and the cost/benefit analysis is still valid.

- Resolution of interdepartmental conflicts.

- Exhibit IV-2 summarizes some of the available techniques for fostering end user participation in the system development process. The ultimate goal of these techniques is to secure the users' emotional commitment to the project through the perception that the project is theirs to control.

EXHIBIT IV-2

SOME TECHNIQUES FOR FOSTERING END USER INVOLVEMENT

- HAVE AN END USER CHAIR THE PROJECT TEAM.
- HAVE END USER REPRESENTATIVES AS PERMANENT MEMBERS OF THE PROJECT TEAM.
- HAVE THE END USER DEFINE BUSINESS GOALS OF THE PROPOSED SYSTEM AND PERFORM COST/BENEFIT ANALYSIS.
- HOLD PROJECT TEAM MEETINGS ON END USERS' "TURF."
- HAVE EDP ANALYSTS WORK WITH THE USER IN THE USER'S ENVIRONMENT BEFORE AND DURING THE PROJECT.
- HAVE ONE OR MORE LEVELS OF EDP STEERING COMMITTEES.
- HAVE END USER REPRESENTATIVES PARTICIPATE IN DESIGN REVIEWS AND WALK-THROUGHS.
- HAVE A DEFINED SIGNOFF PROCEDURE FOR OBTAINING END USER FORMAL APPROVAL OF VARIOUS PROJECT DEVELOPMENT PHASES.

V OTHER NON-TECHNICAL ISSUES IN SYSTEM DEVELOPMENT

V OTHER NON-TECHNICAL ISSUES IN SYSTEM DEVELOPMENT

A. POLITICS

- There is a very strong tendency among EDP executives to view the system development effort as soley a technical and technological process.

  - Consequently, there is a marked tendency to seek out and rely exclusively on technical solutions to problems that surface during the development.

  - In fact, this report has recommended such an approach wherever feasible.

- In many cases, however, these problems are due directly to political factors. Attempts to cure essentially political problems with technological solutions, while very common, are rarely successful.

- The term "politics" in the context of this report refers to the entire spectrum of human interactions embedded in the fabric of the daily operation of any corporate organization.

  - These human interrelationships and interactions impact EDP system development efforts to a far greater extent than is usually allowed for or addressed.

- These "political" factors are singled out in this chapter and contrasted with technological and technical factors.

- Driven by the rapid increase in technological capabilities and capacities of basic data processing equipment and software, accompanied by a rapid decrease in unit costs, more and more organizations today are embarking on the development of ambitious new information processing systems.

  - These new systems are far more complex than those attempted only a decade ago.

  - They also have a far more direct, tangible and visible impact on the mainstream of the organization's business activities.

  - In many cases, these new systems are accompanied by major changes in the daily routines and organizational reporting structure of many employees.

- Most people perceive change as a destabilizing influence and a threat to their security.

  - The greater the unknown element in the perceived change, the more people instinctively fear such change and tend to view the status quo (however unattractive) as preferable to the proposed change.

  - Even when the benefits of the proposed change are explained fully and extensively, it is always more difficult to find enthusiastic promoters for the proposed change than it is to find resistors who prefer to see things go on as they have in the past.

- Failure to grasp this most elementary political principle, and to take active measures to lessen its impact, has resulted in the failure of more new information systems than all other factors combined.

- The ability to anticipate political problems is a skill of immense value to the information systems executive.

  - Knowing the common characteristics that usually identify these situations, even though the details vary greatly from one organization to another, can enable the appropriate adjustments to be made.

  - Exhibit V-1 summarizes these characteristics.

- In general, relatively politics-free situations involve projects conducted on behalf of a single end user department, and cover activities that are not in the mainstream of corporate business activities.

- On the other hand, situations where the probability of political conflicts is very high usually involve:

  - Multiple clients (i.e., several end user departments).

  - Geographical dispersal (i.e., several corporate locations at remote geographical locations).

  - Conflicting end user concerns, needs and requirements.

  - Highly complex systems.

  - Heavy use of, and reliance on, shared resources, such as common data bases.

  - Heavy use of, and reliance on, on-line access.

  - Sensitive activities in the mainstream of the corporation's business, where even short periods of down-time will be quickly reflected up to high management levels.

EXHIBIT V-1

CHARACTERISTICS OF POLITICAL VERSUS TECHNICAL SITUATIONS

SITUATION / FACTOR	TECHNICAL	POLITICAL
USER CHARACTERISTIC	Single user department; one corporate location involved; well-defined system requirements.	Multiple user departments; multiple geographical locations involved; conflicting user concerns, needs and requirements
APPLICATION CHARACTERISTIC	Simple application; well-defined inputs and outputs; limited inputs and outputs; easily defined measure of project completeness and/or success.	Complex application; heavy on-line access; range of possible inputs and outputs wide or not fully definable; heavy reliance on shared resources (e.g., common data bases); range of potential uses wide and unpredictable.
RELATIONSHIP OF APPLICATION TO CORPORATE GOALS	Not in the mainstream of corporate business activities.	Application is extremely visible to high management levels and is in the mainstream of corporate business activities.

- In order to manage these situations effectively, the EDP manager must develop the skills of diplomacy, mediation and negotiation.

  - Rigid adherence to technical solutions has rarely been sufficient, but is preferable to inept performance in the internal politics game.

  - Unfortunately, inept performers seldom recognize their own shortcomings.

B. CREATIVITY VERSUS TECHNICAL ADVENTURE

- A unique managerial function facing the EDP executive and, unhappily, also one of the most neglected, is the duty to control creativity and resist the "technical adventure" syndrome.

  - Creativity in the DP department is usually expressed in deviations from standards. Programmers are especially fond of expressing their creative abilities by designing "tricky" programs.

  - Such programming tricks are often responsible for the most subtle bugs that are extremely costly to discover and remove.

  - DP staffs in general are technology-driven and can rarely resist the temptation to launch "technical adventures" under the guise of satisfying users' requirements.

  - An example of the "technical adventure" syndrome might be the solution that involves heavy internal modifications to, for instance, an I/O access method (upon which numerous other applications rely), instead of attempting more mundane but far safer solutions that can be achieved without disturbing the systems software.

- Unfortunately, control of creativity is often confused with quashing creativity, and results in a highly regimented environment in which conformity of style is rigidly enforced.

  - Rigid enforcement of standards, often by lower-level clerical people, in itself does not achieve the desired end, and may contribute to the conditions that foster job dissatisfaction and high turnover.

  - The key to the control of the creative impulse (by channeling it into more useful avenues) is to indicate to the DP staff that readable, maintainable, simple programs and designs are far greater expressions of creativity than weird or "clever" tricks, no matter how many machine cycles they save.

  - Achieving this level of understanding is a purely educational, people-centered issue.

- "Technical adventures" are not necessarily inherently bad. On the contrary, such developments can quite often have significant payoffs. The EDP executive should not shrink from launching such daring enterprises, provided however that:

  - The risks involved have been carefully and thoroughly examined, including an analysis that compares the potential life cycle cost/benefits of alternate approaches.

  - The benefits of the proposed solution have been demonstrated in as quantitative a way as possible.

  - The benefits are so clear and overwhelming that the risks are worth taking, even if the effort does not succeed.

  - A backup plan is developed to assure recovery from as many of the foreseen risks as possible.

- The EDP executive has confidence in the staff's ability to undertake the risky project.

- The user and executive management are aware of the risks and prepared for them, including the likelihood that the system will be late and may deliver less than it promised.

• Ideally, the EDP executive, like any other executive, possesses leadership qualities.

  - Simply put, leadership is that quality that makes other people <u>want</u> to do what you ask them to do, and that inspires loyalty in other people.

  - True leadership is, of course, very rare; but most successful executives possess some components of this quality: natural authority, persuasiveness, character, personality, a sharp mind and, of course, a conferred authority by virtue of position.

  - Different people respond better to different aspects of the leadership quality; the successful executive should be alert to such differences.

  - By discovering which aspects of their personality work best in inspiring each of their subordinates, and by striving to improve and enlarge these aspects, executives increase their ability to lead.

• So much for theory. In the real world, executives have to function without the benefit of natural leadership abilities. That need not detract from their effectiveness.

  - Executives can achieve their goals by persuading, influencing, negotiating and being sensitive to human motivations and needs.

  - Staff only need to modify their behavior and performance, not their beliefs.

- These are all skills that can be learned, formally or informally, and improved through conscious efforts and practice.

- These abilities are the key to effective functioning in the highly political, people-oriented environment that characterizes most modern information systems operations within the organization.

VI MANAGEMENT OF THE PROJECT LIFE CYCLE

VI MANAGEMENT OF THE PROJECT LIFE CYCLE

A. PROJECT DURATION AND SIZE

- There was substantial agreement among the 28 organizations that responded to INPUT's question about the average life cycle of major systems, as shown in Exhibit VI-1.

  - The overwhelming majority of respondents defined such life cycles at figures varying from 4 to 10.5 years, with an average of 7.5 years.

- The significance of this point is that the average life of major EDP systems extends beyond the typical five-year useful life of EDP hardware.

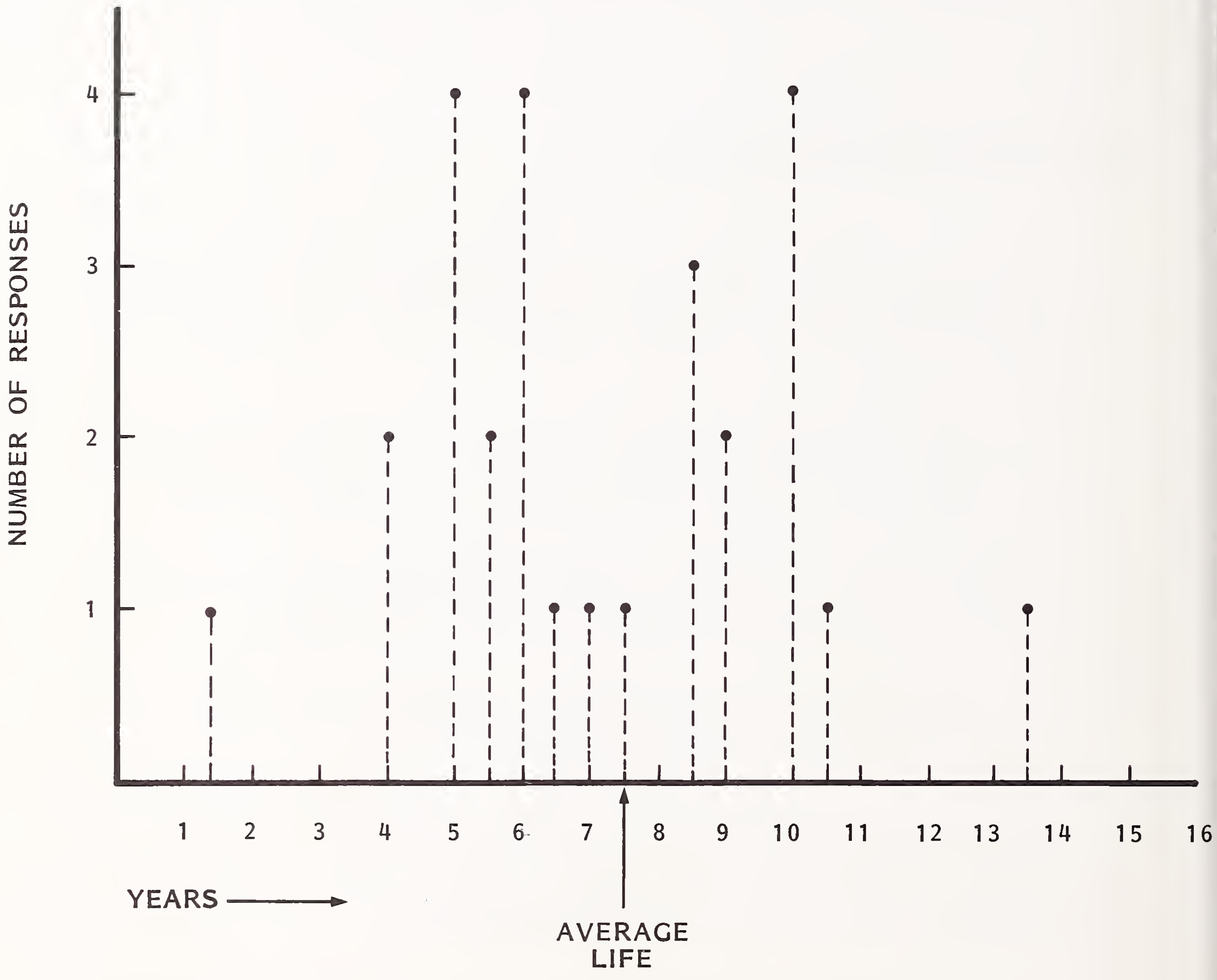
  - Consequently, most major systems will have to undergo at least one hardware conversion during their life cycle.

- There was less agreement on the relative extent of the elementary components of the development phase, as can be seen in Exhibit VI-2. However, it is possible to make at least the following generalizations:

  - Most repondents (18) reported that the specifications phase occupies less than 25% of the development cycle.

EXHIBIT VI-1

AVERAGE LIFE OF MAJOR SYSTEMS, INCLUDING DEVELOPMENT



TOTAL RESPONSES: 28

000006

EXHIBIT VI-2

RESPONDENTS' ESTIMATES OF
DEVELOPMENT PHASE'S DURATION

SUB-PHASE	NUMBER OF RESPONSES BY PERCENT OF DEVELOPMENT PHASE				
	15% AND UNDER	16-24%	25-35%	36-50%	OVER 50%
SPECIFICATIONS	9	9	7	5	0
DETAILED DESIGN	5	10	10	5	0
CODING	8	5	13	4	0
TESTING	7	8	10	3	2

TOTAL RESPONSES: 30

- Most respondents (20) pegged the detailed design phase as occupying between 16% and 35% of the development phase.

- Most respondents (18) agreed that <u>coding</u> and <u>testing</u> subphases also occupy between 16% and 35% of the development phase each.

- If the responses are weighted by the number of respondents, the average percentage of effort spent on each phase is as follows (total does not equal 100% due to weighting):

  - Specifications, 22.4%.

  - Detailed design, 25%.

  - Coding, 24%.

  - Testing, 25.4%.

- These percentages differ somewhat from the usual 40-20-40 split. It is tempting to think they reflect the reduced effort required for testing when structured analysis and design techniques are employed.

  - However, results cannot be considered definitive because no controls were employed by the respondents to certify these figures.

  - Neither was there any definite basis to indicate that the total elapsed time was shorter, as has sometimes been reported.

- There was a rather surprising disagreement on the definition of project size:

  - Although, as shown in Exhibit VI-3, most respondents (17) agreed that a major project takes one to five years, a full one-third (10) even classified projects of less than one year in this category. Only one answer regarded large projects as those lasting over five years.

EXHIBIT VI-3

DEFINITION OF PROJECT SIZE BY DURATION

PROJECT SIZE / DURATION	NUMBER OF RESPONSES		
	LARGE	MEDIUM	SMALL
UNDER 1 YEAR	10	20	28
1-5 YEARS	17	7	0
OVER 5 YEARS	1	0	0
NO ANSWER	2	3	2

TOTAL RESPONSES: 30

- There was general agreement that both small and medium projects last as little as one year or less.

  - Only seven respondents felt that one- to five-year projects should be classified as medium rather than large.

- The conclusion is that the organizations INPUT interviewed distinguish only between large projects on the one hand, generally lasting one year or more, and small projects, lasting one year or less.

- There was wider disagreement on the classification of project sizes by the number of people staffing the project, as revealed by Exhibit VI-4.

  - Over half of the respondents would call a project of five persons or more a large project. Twelve other respondents would also include staffs of under five people in this category.

- Medium and small projects, again, seem to be regarded as more or less the same by those organizations INPUT interviewed:

  - In both cases, the overwhelming majority of respondents felt that such projects are staffed by less than five people.

  - Only five respondents would define medium projects as having staffs of over five people, and only one respondent defined small projects this way.

- The differing perspectives on the question of project size are clearly illustrated in Exhibit VI-5, where the variance in the response to the questions of both project duration and staffing is summarized.

  - For instance, a large project varied in definition from as little as six months in duration to three years and up.

EXHIBIT VI-4

DEFINITION OF PROJECT SIZE BY TEAM SIZE

PROJECT SIZE / TEAM SIZE	NUMBER OF RESPONSES		
	LARGE	MEDIUM	SMALL
UNDER 5	12	22	27
5-9	9	3	1
10-24	5	1	0
25 AND OVER	2	1	0
NO ANSWER	2	3	2

TOTAL RESPONSES: 30

EXHIBIT VI-5

PROJECT SIZE DEFINITION:
VARIATION IN RESPONSES

PROJECT SIZE	LARGE	MEDIUM	SMALL
DURATION, LOW	6 MONTHS	2 WEEKS	1 DAY
DURATION, HIGH*	3 YEARS	1½ YEARS	LESS THAN 1 YEAR
TEAM SIZE, LOW*	2-3	1	1
TEAM SIZE, HIGH	OVER 100	25-30	8-10

TOTAL RESPONSES: 30

- Large projects in terms of staffs extend all the way from teams of two to three people to those of 100 people or more.

- These figures and others shown in the exhibit should be viewed in the context of the previously observed tendency to regard the development and maintenance phases as less than the total life cycle.

- It has often been taken for granted that small projects are entered into with substantially less front-end work (such as planning and requirements) than large projects. In an attempt to obtain data to support or refute this assumption, INPUT asked the respondents whether small projects required a shorter development phase, and if so, by how much.

  - Exhibit VI-6 summarizes the responses to this question.

  - More than half the respondents (17) did not think that small projects required any less development effort than large ones.

  - Of the 13 respondents who thought that small projects did require less development relative to their life cycle, nine furnished a percentage definition of the development phase.

  - Six felt that small projects need only a 20% development effort.

  - This is substantially less than the 52% average development component in the life cycle of all systems, reported elsewhere in this study.

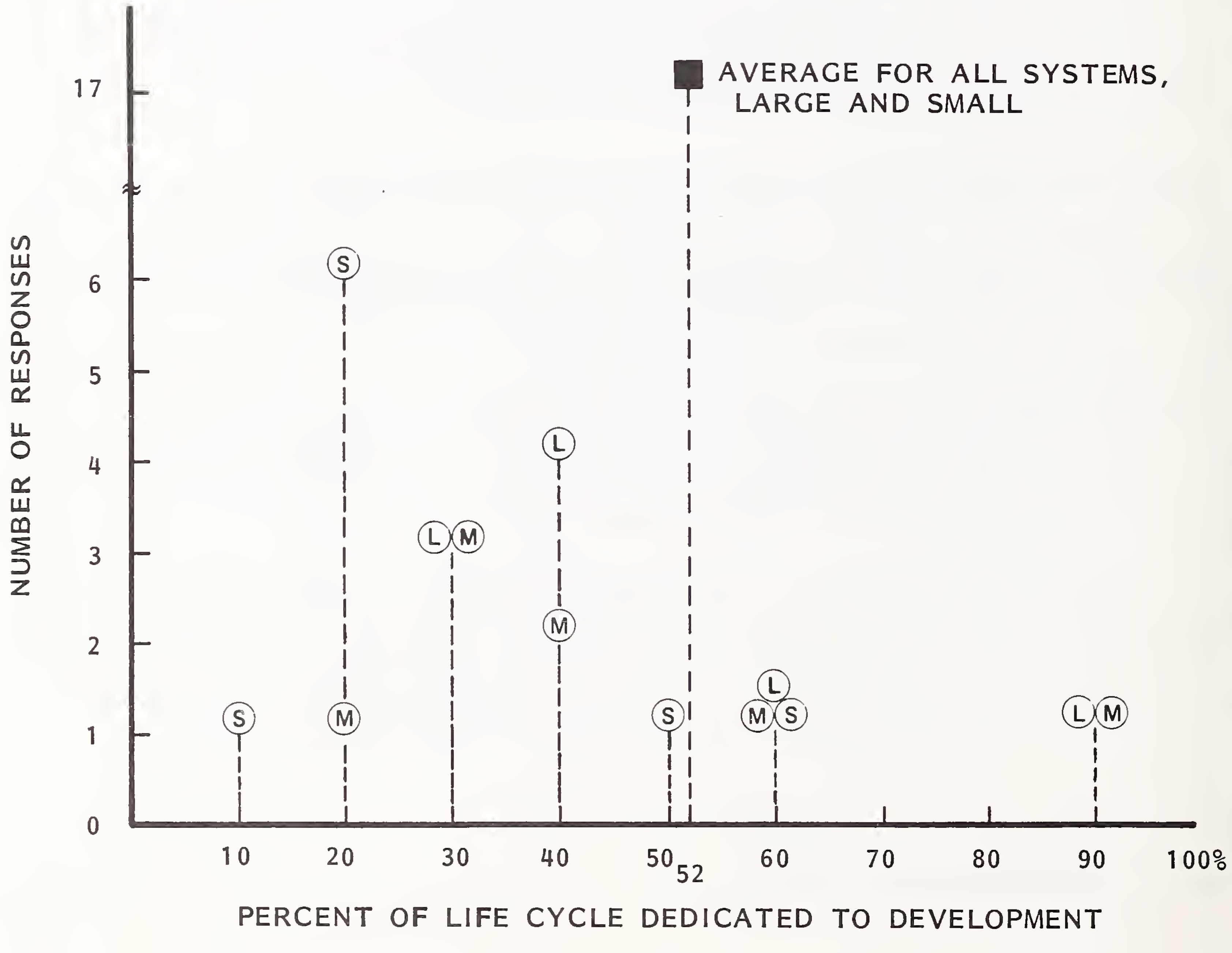
B. PLANNING THE PROJECT

1. INITIATION CRITERIA

- The main purposes of the planning stage are:

EXHIBIT VI-6

EFFECT OF PROJECT SIZE ON DURATION OF DEVELOPMENT EFFORT



Ⓛ LARGE PROJECTS
Ⓜ MEDIUM PROJECTS
Ⓢ SMALL PROJECTS

TOTAL RESPONSES: 30

RESPONSES IN GRAPH: 9

- To establish the desirability of the project on the basis of its cost/benefit potential and its conformity to the corporation's business goals.

- To examine alternate ways of achieving the desired end, and to select the way that is most appropriate according to agreed-upon criteria (e.g., lowest cost, shortest time, smallest staff, least impact on existing procedures, least required new resources, etc.).

• One of the textbook caveats regarding this phase is to "secure top management involvement."

  - The idea suggests that top management should evaluate the proposed system relative to the corporate goals, and review the cost/benefit analysis.

  - In some cases, the size of the project, its impact on the mainstream of corporate business, and the amount of resources required to implement it, automatically ensure such involvement.

  - In other cases, top management would not normally be involved. "Securing their involvement" requires some action.

• The realities of the situation, however, suggest that in the absence of the factors cited above that automatically force top-management involvement, it may not be necessary or even desirable to try to get such attention.

  - Waiting for top management (typically consisting of people with financial and marketing expertise) to grasp the significance of the proposed technical system may inhibit ever getting the system launched.

  - The practical reality is that these types of projects must be launched on middle management's initiative without the blessing or full understanding of top management.

- In such situations, it is doubly important to carry out a painstaking planning phase, especially the cost/benefit analysis. If the project develops unexpected problems that bring it to the attention of top management, the inquiry into the situation will inevitably involve questioning the wisdom of the project's original conception and launching.

- It is then important to be able to demonstrate the analysis work that convinced all concerned of the desirability of the project.

● Too often, the examination of available alternatives is shallow, consisting primarily of searching for arguments to justify an approach already selected.

  - This preselected approach then turns into a "technical adventure," requiring the creation from scratch of many system components.

● An effective EDP executive will assure that an honest, careful evaluation of alternatives does take place during the planning phase.

  - Parts or all of the proposed system may be achieved with off-the-shelf, commercially available or otherwise accessible existing systems. Some examples:

    . Packaged application systems.

    . Reusable code systems.

    . Applications generators (also sometimes called "non-procedural languages" or menu-driven programming).

    . DBMS with associated user-oriented inquiry/retrieval languages.

  - These are covered in more detail elsewhere in this report.

- Users may frequently desire to introduce computerization and automation into system activities that are best handled by manual operations and human judgement.

  - Such desires encourage the "technical adventure" syndrome, which the DP professionals, with their highly technical competence and inclination, tend to aggravate.

- It is the responsibility of the users to examine alternatives from the business point of view in an honest attempt to discover the alternative that best fits the department's and corporation's business goals.

  - Both the EDP executive and the user department's management share the responsibility for checking these "adventure" tendencies by encouraging and demanding honest, alternate evaluations and careful risk analyses.

2. CONCLUSION CRITERIA

- Perhaps no other aspect of the planning phase is so often ignored as the definitions of:

  - Project conclusion.

  - Project success.

- Incomplete, vague or obsolete definitions of project completion and project success are common causes for aggravated conflicts and disagreements between users and information systems departments.

- Theoretically, a project is complete when the implementation or cut-over phase is finished. It is a very common mistake to assume that this is a self-evident, clear milestone.

- In reality, implementation (introducing the new system into routine operation and discarding the old system or systems) is in itself a gradual process.

- During the implementation, as well as earlier in the process, users continually discover new capabilities they would like to have, or different ways to achieve already agreed-upon goals.

- Errors of judgement and lack of foresight on the part of systems personnel and end users are often discovered midstream in the development or implementation phases.

● All these factors combine to create powerful pressures to regard the system as incomplete until these additional capabilities and changes are incorporated.

● Accepting these "mid-course corrections" opens a Pandora's box of endless additional requirements and modifications.

  - The most common result of this attempt to "accommodate" is a never-ending project development phase.

● The way to minimize or eliminate this "target drift" effect is to obtain a user-DP consensus on the criteria that constitute the termination of the development/implementation phase and the beginning of the maintenance/enhancement phase.

  - A critical element in the consensus is a well-defined acceptance test that is written and signed by the managements of the end user and EDP departments.

  - Both the end user and EDP executives must resist the pressures to modify the system after the goals and acceptance test have been agreed upon.

- Before any changes are accepted, the EDP executive must assure that their impact on schedules, budgets, goals and acceptance test are understood and again agreed to, preferably in writing, by all parties.

- The EDP executive, however, must not rely on the acceptance test exclusively as the delineator of the project's concluding milestone.

  - It is rarely possible to design acceptance tests that answer all questions unambiguously.

  - As already indicated, both users and DP personnel may propose changes designed to correct errors and oversights. If those problems were caused by EDP personnel, the EDP executive will find it hard to insist that the initial acceptance test remain unchanged.

  - This is clearly a political arena where negotiation, mediation, persuasion and similar people-oriented skills are the key to the successful resolution of the problems.

- Independent of the technical completion of a project, its success (as measured by user satisfaction) may also be questioned.

  - End users often complain that the system does not do what they expected it to do, even when (technically) the items and features that were agreed to at the beginning of the project have been achieved.

  - These complaints are frequently related to human engineering factors, where the desired features are present but the hands-on users find it difficult or unpleasant to access them.

  - In other instances, the system may be technically complete but fail to achieve its business goals; for example, because of cost overruns that invalidate the initial cost/benefit analysis.

- For these reasons, it is essential that careful attention be given in the planning stage to the formal definition of the success criteria for the project.

  - As much as possible, success criteria should be objectively measurable.

  - If subjective criteria must be included, they should be carefully defined and agreed upon, including the methods by which they will be recorded.

- Of special concern to the EDP executive are those technical features of the system that figure prominently in the definition of "success."

  - As the result of an honest effort to define "success," users may realize that some requested features are actually secondary in nature, and could be omitted from the initial system with little or no impact on the project's success.

  - These omitted features can be reserved for the enhancement phase.

- In fact, a concerted effort should be made to define not only the requirements of a system as presently understood, but also all the ways it may be changed over the next five to ten years.

- One of the most common mistakes made during the planning phase is the implicit assumption that the hands-on end users who will eventually operate the system (or major parts of it) possess the same level of experience and competence as the designers.

  - The planning and design team must be acutely aware of the actual experience level and capabilities of the hands-on end users.

  - "HELP" screens to provide guidance for uninitiated operators should be part of the design goals.

- Reliance on menu-driven screens for data entry, rather than on handwritten forms, should be encouraged. Screens can be programmed to reject bad inputs at the source, minimizing the opportunity for bona fide errors and for intentional sabotage.

- Don't overlook the issues of privacy and security in the planning phase.

  - Restricted access to data may be required to satisfy federal, state, local and corporate privacy regulations.

  - Restricting access to maintain data integrity and to prevent unauthorized use of confidential data should be discussed early in the planning phase; this is a sensitive subject that can easily lead to aggravated political difficulties (e.g., who has the right to access and/or modify what data).

- Caveats for the planning phase that are still worth repeating are:

  - Provide for iterations on the detailed design and other phases to accommodate unforeseen design difficulties and changes. An SDM typically provides these mechanisms.

  - Provide "kill points" where the cost/benefit analysis is revaluated and the need for the project reexamined. Go/no-go points permit known gradual increases in the risks and exposures involved in the new system, rather than large surprises at the end of the development process.

  - Don't ignore such mundane things as tapes and diskpacks (and their mounting and storage charges) when estimating resources and costs.

  - Plan for the training of the maintenance team.

- Ideally, project plan approval should receive the same care and scrutiny as if it were an equivalent capital investment decision. A review of the steps involved in justifying the procurement of hardware of equal monetary value to the expected total project costs could be educational and could suggest some practical steps to be copied.

- Do not, however, make the planning phase so lengthy and cumbersome that decisions reached early in the process become obsolete and irrelevant by the time the next phase is ready for launching.

3. CUT-OVER PLAN

- One of the subjects that must be carefully addressed in the planning phase is the implementation or cut-over plan.

  - "Implementation" here is defined as the actual installation of the system on the production-oriented DP hardware and its integration with existing systems and/or business routines.

- If the new system takes over duties of one or more old systems, then a "cut-over" plan must be detailed as well.

- There is a great temptation to think that older systems can act as backups in case the new system encounters implementation difficulties. This is usually a mistake for several reasons:

  - Requiring the old system or systems to continue to operate alongside the new system provides a psychological "crutch" and a powerful incentive not to use the new system.

  - Because of differences in input format requirements, file organization and report format, the concurrent operation of both the old and new systems could cause substantial confusion and difficulties.

- It is probably better to plan for the cut-over on the assumption that "bridges will be burned" as the new system goes into operation.

- By the way of analogy, one might say that the new system requires driving on the right side of the road, whereas the old system requires driving on the left. In this case, a sharp cut-over is definitely preferable.

- Naturally, sharp cut-overs require a carefully thought-out plan.

C. SYSTEM DEVELOPMENT METHODOLOGY

- Nearly half of the organizations INPUT interviewed indicated that they employed a formal system development methodology (abbreviated "SDM"), as shown in Exhibit VI-7.

- An SDM (also called a program management system project life cycle, phased plan or similar terminology) is essentially an extensive checklist or guide for the execution of all life-cycle phases of a project.

  - SDMs are available commercially, varying greatly in extent; one commercially available SDM is packaged in a set of 25 volumes. Others vary from one to five or six volumes, containing the checklists and some tutorial material. These commercial SDMs carry hefty price tags, from a few thousand dollars to $60,000 and over.

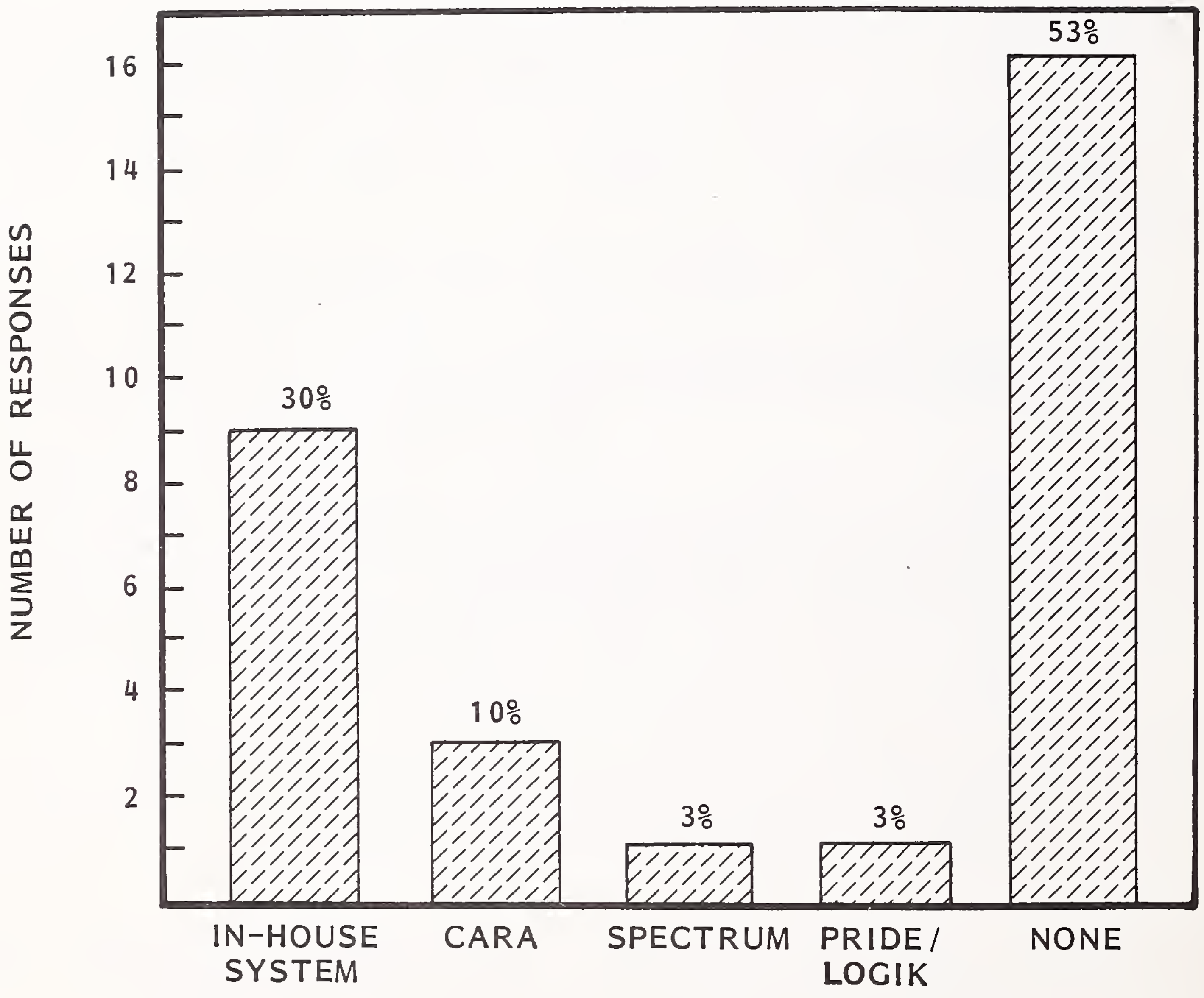
  - Other SDMs are simply published as textbooks, which can be purchased for as little as $25.

  - In many instances, companies have developed their own SDMs, either independently or starting from a commercial SDM as a base.

- Some SDMs, notably PRIDE-Logik from MBA Inc., are partially automated; i.e., assisted by computerized procedures.

EXHIBIT VI-7

USE OF SYSTEM DEVELOPMENT METHODOLOGY



TOTAL RESPONSES: 30

- Generally, a well-developed SDM will have some or all of the following elements:

  - Guidelines and forms for system planning, requirements specifications and cost effectiveness analysis.

  - Guidelines and forms for system analysis and design, generally based on structured concepts or a specific structured design methodology.

  - A well-defined, multiphased system life-cycle description, with clearly specified deliverables, review processes and signoff procedures appropriate to the conclusion of each life cycle subphase.

  - Well-defined standards for documentation, with forms.

  - Guides for estimating required resources, personnel and computing facilities.

  - Guidelines and forms for acceptance testing.

  - A project control mechanism.

  - Role definitions for the work groups and project teams that execute the SDM steps.

  - Tutorials - written or computerized explanations and/or demonstrations of the actual use of the SDM.

- An effective, well-designed SDM should have the following characteristics:

  - It must be easy to understand and use.

- It must be flexible, or modular, so that small projects do not require the cumbersome, detailed sequence of steps that are appropriate to major projects.

- It must include provisions for iterating specifications, designs, etc., where appropriate; i.e., where the review process results in changes to such specifications and designs.

- It must include provision for the maintenance phase of the life cycle, such as change control mechanisms that assure that the documentation reflects the maintenance activities.

- It must make provisions for extensive user participation in the development process.

- It must match the needs and characteristics of the organization which uses the SDM.

- It must apply to both batch and on-line systems, and to centrally developed as well as distributed systems.

• Needless to say, no single SDM can satisfy all these requirements. The above list is still useful as a measuring stick for any proposed SDM.

• The advantages and benefits of an SDM can be generalized as follows:

  - By providing a very detailed checklist of tasks and procedures throughout the project's life cycle, the SDM minimizes the chance that some step or aspect of the system's development will be overlooked.

• For example, an often ignored or vaguely executed phase is the planning stage, in which the business justification of the project (the cost/benefit analysis and the reconciliation of the project with the corporate business objectives) is established. By including a specific set of tasks and reviews to perform in this stage, the SDM assures that any omissions or abbreviations are the result of conscious decisions, rather than oversights or forgetfulness.

- By defining a multiphased, detailed life cycle with specific reviews, deliverables and signoffs at strategic milestones, the SDM makes possible project control and progress measurement.

- By providing several specific "kill points" at which a project can be terminated (for example, because costs escalate to the point that the original cost/benefit analysis is no longer valid), the SDM permits evaluation of gradually increasing risks and minimization of losses on early termination.

- The collection of guidelines, checklists and forms in the SDM in effect represent the collected wisdom of the organization, based on its past experiences. Having access to such a collection compensates for lack of experience on the part of new analysts, who can therefore achieve greater levels of efficiency in less time than would be possible without the SDM.

  - Having such a documented, standard way of doing things permits less painful personnel shuffles between projects, and simplifies the work specifications for consultants and contract personnel.

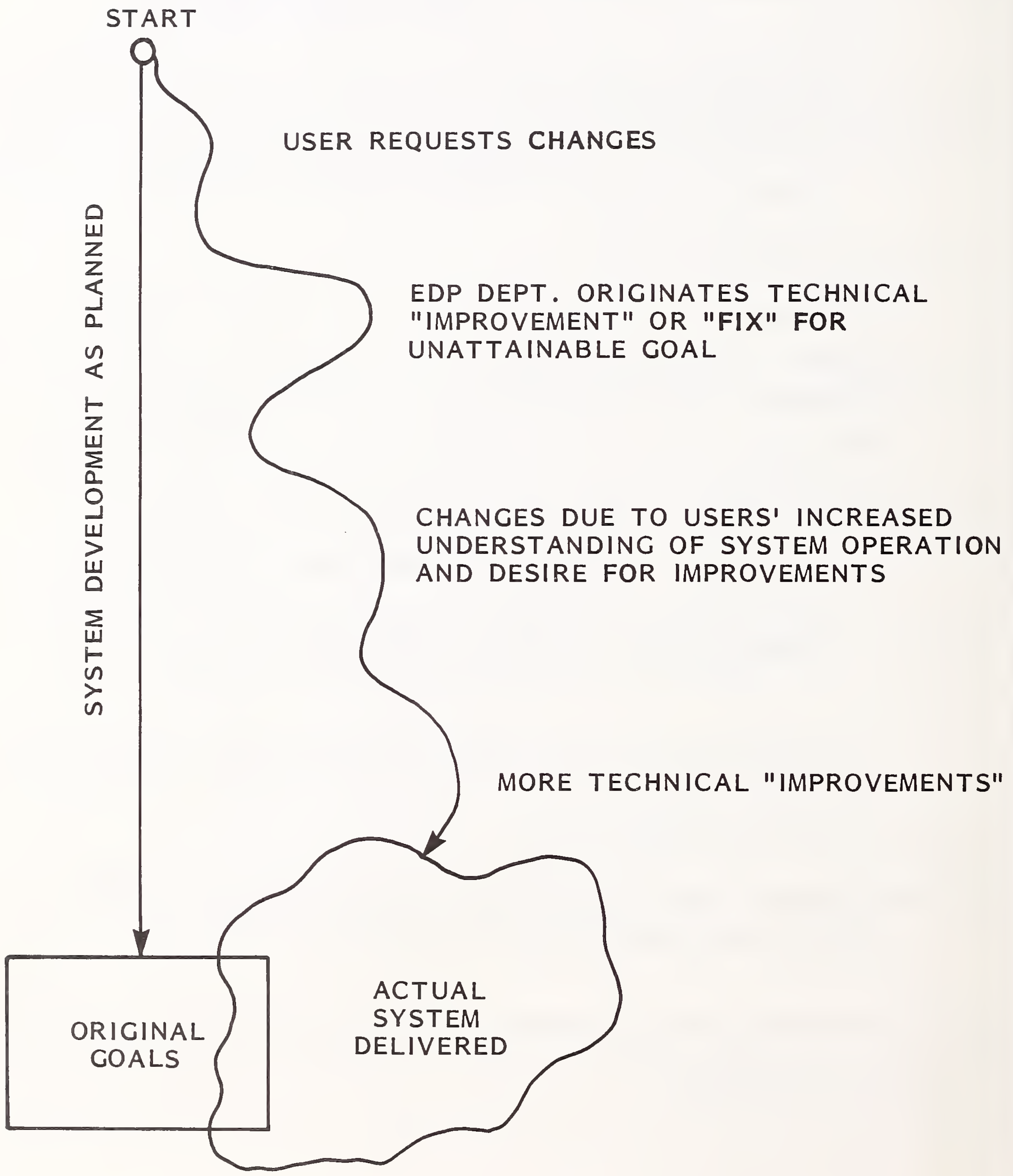
  - By enforcing standards of documentation and style, the SDM facilitates clearer communications between users, developers, maintenance personnel and management.

- Perhaps one of the most significant benefits of the use of an SDM is in its control over the process of system modification.

- Exhibit VI-8 is a graphical depiction of the way in which most systems are actually developed.

EXHIBIT VI-8

HOW SYSTEMS DEVELOP IN THE REAL WORLD:
"TARGET DRIFT"


START
SYSTEM DEVELOPMENT AS PLANNED
USER REQUESTS CHANGES
EDP DEPT. ORIGINATES TECHNICAL "IMPROVEMENT" OR "FIX" FOR UNATTAINABLE GOAL
CHANGES DUE TO USERS' INCREASED UNDERSTANDING OF SYSTEM OPERATION AND DESIRE FOR IMPROVEMENTS
MORE TECHNICAL "IMPROVEMENTS"
ORIGINAL GOALS
ACTUAL SYSTEM DELIVERED

- Both users and DP personnel are guilty of introducing modifications into the system's goals and design during the course of the development process.

- Users do this as they learn more about the system and discover undesirable characteristics or see the opportunity for improvements or additional features.

- DP people can rarely resist the temptation to improve the system's operation in terms of speed, memory and disk utilization, and so on.

- If this process of "improving" the system goes on unchecked, the final shape of the system diverges more and more from the originally envisioned one, with the probable consequence that the final system matches the user's original requirements only marginally.

- The SDM creates a built-in resistance to such a process of gradual drift by:

  - Specifying a definite requirements phase, at the end of which both users and EDP department in effect sign a contract to develop a well-defined system.

  - Requiring a definite review process before any midstream changes are introduced to the goals or design of the system. The review assures that the impact of the changes on the cost/benefit analysis, schedules and budgets is fully examined.

  - This review process can also be used to obtain a consensus on the impact of the desired changes on the definition of system completeness and system success.

- While the administrative safeguards introduced by an SDM can be effective in resisting the pressures to make modifications "on the fly," the EDP executive must not lose sight of the fact that the desire to include changes is very frequently driven by political, not technical, factors:

    - Users who discover too late that they made gross errors in specifying the requirements of the system will be highly motivated to propose changes that will cover up such mistakes.

    - Similarly, DP personnel who discover that the technical promises made earlier are not feasible, due to oversights, poor estimating and errors in judgements, will be motivated to request modifications that will similarly cover up such errors.

    - Negotiating the best way out of such situations requires political skills of the highest order. The SDM procedures above will not uncover the real motives for change, nor will they take cognizance of the face-saving and turf-protecting activities involved in such situations.

- The most important disadvantage of an SDM is that it requires a great deal of effort in "up-front" work on the planning and requirements phases.

    - These efforts typically result in little tangible progress toward actually developing the desired system.

    - This situation leads to a great deal of impatience among users and apprehension within management.

- It is essential that both users and top corporate management be properly prepared for this characteristic of an SDM.

- Other problems or limitations are generally associated with most SDMs:

- In order to fulfill its main mission, an SDM in general must be expressed in a voluminous set of books or binders. When these are provided by a commercial vendor, they are not likely to match exactly the needs and operational routines of any specific organization.

- A great deal of work must be done by the vendor or the purchaser in order to modify and customize the SDM's procedures and forms to fit local needs.

- The great volume of text involved in a fully detailed SDM tends to overwhelm users and developers, who may therefore resist its implementation.

- Most commercial SDMs were developed at the time when EDP operations were primarily batch-oriented; they do not take into account the fact that most new projects are highly interactive.

• Despite these shortcomings, an SDM can be an effective system development mechanism for many situations, especially those characterized by:

  - Large projects requiring a significant number of staff.

  - Staffs with significant differences in experience and capabilities.

  - High staff turnover rate.

  - Complex projects requiring or relying on a DBMS, wide geographical distribution and on-line access.

  - Projects that are highly visible and in the mainstream of the corporate business, where even short down-times are quickly reflected to the highest corporate levels.

  - Projects where more than one end user department is involved.

- Projects whose resulting systems are expected to be in operation for a long time (i.e., five years and over).

- An SDM is generally not needed when the projects involved are small and when just one user department is involved.

- Although in general an especially bad time to introduce new tools and techniques is after a project has already become mired in difficulties, an SDM is perhaps the only exception to this rule.

  - Its introduction can quickly point out the deficiencies in the system's design and/or justification. (However, this will neither speed up the process nor correct the problems automatically.)

  - A special "rescue" task force can be formed to study the SDM and attempt to apply it as best they can to the faltering project.

D. MEASUREMENT AND ESTIMATION

- There is a clear tendency to confuse reporting with measurement in EDP projects.

  - Most progress-reporting systems in use, whether verbal, written or formalized (based on fixed forms), tend to accept the reporter's opinions on the way the tasks are progressing as an indication of the actual progress.

  - Nearly all time and budget reporting systems are little more than a record of effort expended.

  - In neither case does the report measure in a tangible, objective way the progress made toward the completion of the task.

- Inherent difficulties exist in devising objective methods of measurement in the system development discipline, which is composed to a large extent of intellectual activities.

  - It can be plausibly argued that the work of clerks and secretaries, for example, is equally resistant to objective measurement, so why should one expect the work of analysts and programmers to be more amenable to such measurements?

- Nevertheless, the pervasive feeling in the industry is that the measurement of EDP system development is not as good as it could be.

  - But a surprising number of organizations queried in connection with this study indicated that they were not employing any kind of measurement in their EDP activities.

  - Only 46% claimed to have various reporting and measurement techniques in use, as shown in Exhibit VI-9.

  - Only 43% reported use of a project control mechanism, as shown in Exhibit VI-10.

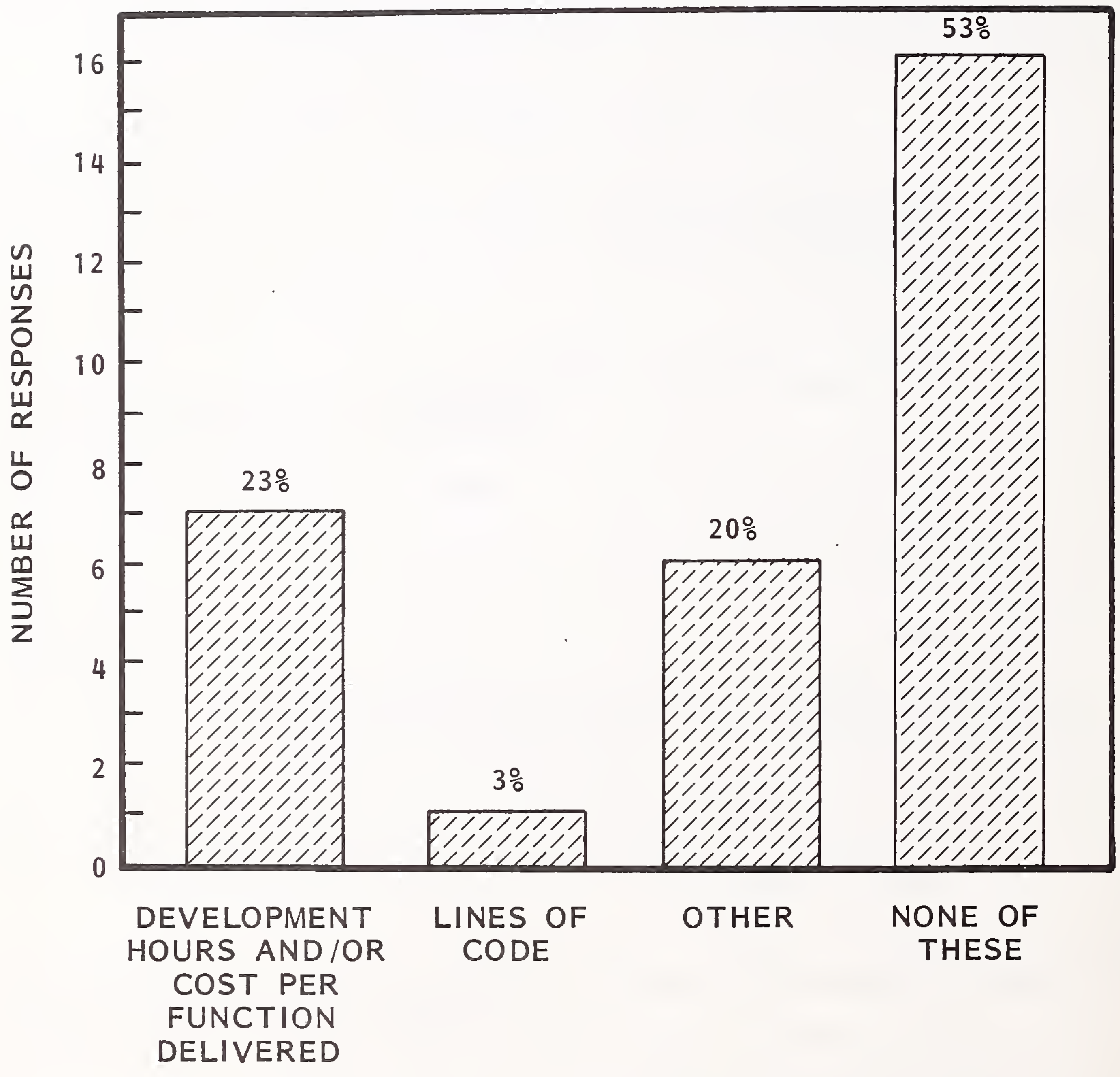
- The reliance on reporting of efforts expended, lines of code produced and reporters' opinions as measures of progress toward task completion are directly responsible for the "90% complete" syndrome.

  - Roughly stated, this syndrome is manifested by projects that are "90% complete" during 90% of their development cycle.

- What is clearly missing from the current so-called measurement systems is a firm linkage between efforts expended and/or lines of code produced on the one hand, and actual completion of tasks on the other.

EXHIBIT VI-9

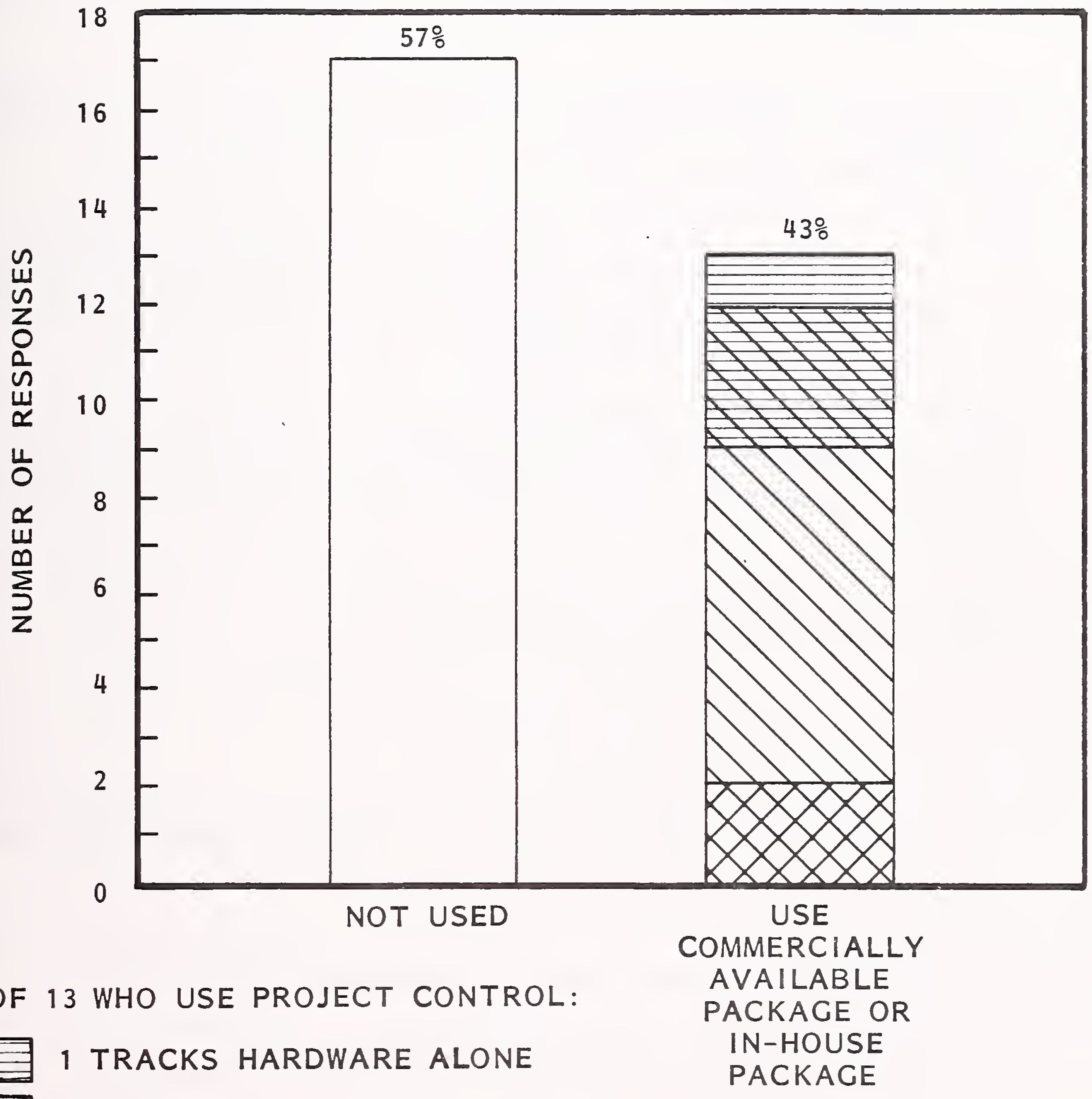
USE OF MEASUREMENTS



TOTAL RESPONSES: 30

EXHIBIT VI-10

USE OF PROJECT CONTROL



OF 13 WHO USE PROJECT CONTROL:

▤ 1 TRACKS HARDWARE ALONE

▨ 3 TRACK STAFF AND HARDWARE

▧ 7 TRACK STAFF ALONE

▩ 2 TRACK STAFF AND MANAGEMENT

TOTAL RESPONSES: 30

- The solution must begin by assigning clearly defined criteria for completion to each task and subtask in the proposed system.

  - Just as the project as a whole must have a clearly defined set of criteria which can be used to determine the end of the development phase, so should each subtask within the system.

  - These criteria can be developed during the detailed design phase, as the system's level of detail is gradually better exposed through top-down design.

- Estimation - a critical element in project planning - is closely related to measurement.

  - Even though there are a large variety of estimating techniques, they are all directly or indirectly related to past experience, sometimes supported by past measurements.

- Poor estimating, rather than change of scope or cost factors, is thought to be responsible for perhaps as much as 70% of all time and cost overruns in EDP projects.

- Some of the reasons for this dismal record are:

  - Overlooking one or more whole tasks.

  - Ignoring the effects of turnover, team size factors, fire-fighting distractions and non-productive time.

  - Failing to adjust the estimates at key project milestones.

  - Estimating by "gut feel" rather than past measurements.

- Overlooking tasks is a common problem when large projects are attacked without the benefit of checklists such as an SDM.

- Turnover will generally have a direct effect on project slippage. A 20% turnover rate can result in a 10% project slippage on a large project, as discussed elsewhere in this report.

- The larger the project team or teams, the more nonproductive time proliferates, due apparently to the complexity of interactions as the number of communicating individuals increases.

- The lost effort of personnel borrowed to "fight fires" (e.g., to perform corrective action on critical maintenance problems with existing systems) is often discounted or ignored when estimating elapsed time to complete a new system.

- Studies have shown that as much as 30% of each DP professional's time is routinely spent in non-productive and/or non-project related activities (training, vacation, sick leave, administrative details, meetings, etc.).

- In any system development, and especially in top-down designs, the level of detail exposed increases gradually during the design process. Estimates made during the planning phase must be readjusted at appropriate milestones to account for such detail.

- Basing estimates on "gut feel" is more subject to gross errors than basing estimates on some concrete measurements of past activities.

- Projects typically gather momentum as they progress; this momentum is lowest at the start of the project and reaches a peak somewhere towards the end of the development and the beginning of the implementation or cut-over phase.

- Momentum is closely related to the learning curve phenomenon, which explains that productivity is lowest at the beginning of the project and picks up gradually as people assimilate the general shape of the system and develop more confidence in their interactions with each other.

  - Breaking this momentum (for example, by relaxing the pace when for some reason more time becomes available) is generally not a good idea.

  - The non-linear expenditure of effort, and especially the non-linear completion of tasks, should be taken into account when estimating the duration of a project.

- Needless to say, allowances must be made for the level of competence and experience of the project's staff.

- Given the present state of the art, when projects always come in under budget and ahead of time, they are being estimated too generously.

  - An EDP department that routinely resorts to such inflated estimates for "protection" does not serve the organization well, and will eventually lose its most important asset - the reputation for technical know-how and skill.

- On the other hand, if estimates are worked out based on measured past experience, they are worth defending even when they seem excessive and there are pressures from both top management and end users to "correct" them in order to make the cost/benefit analysis "come out right."

- A form that facilitates estimation of future activities based on past experience is the "standardized module" approach. In this technique, EDP systems are classified into standard functions or modules, such as:

  - Report generation.

- File update and control.

- Data entry/edit.

- Screen format.

• These and similar functions are then allocated several levels of complexity; e.g., the file update or any other function would be further classifiable as simple, medium or complex.

  - Analyses of past activities in these modules or functions are then undertaken to establish the level of effort required to complete each such function.

  - The level of competence and experience of the DP professional that would be assigned to the task should also be reflected in the effort estimates. In those cases where a function can be performed by junior-level people and above, there should be separate effort estimates depending on whether the implementor is, in fact, junior, medium-grade or senior.

  - In lieu of an analysis of past activities (the data for which may be incomplete or nonexistent), a panel-of-experts technique could be used in which a number of DP professionals are asked to give their own effort estimates for each module, level of complexity and experience level. These estimates are then averaged to form the data base for future estimates.

• Each proposed new system is divided into components that match as closely as possible the standard components listed in the data base.

  - Estimates for the new project are then picked up from the data base by specifying the level of complexity and level of implementor's experience.

- The data base can be kept up to date and its accuracy increased by continually tracking performance on new projects.

- After some experience has been gained with the standardized-module technique, both productivity and the quality of the estimate produced by the EDP project leader can be improved by the "earned-hours" technique.

    - With this technique, tasks are divided into elementary modules.

    - Each module is assigned an "hours-to-complete" figure, based perhaps on a data base of the type previously described, but in any case determined in consultation with the analyst or programmer who is assigned the task.

    - For each module, completion criteria are established, again with the concurrence of the developer.

    - The developer is then credited with the number of hours assigned to any module on the demonstrated completion of that module, regardless of the actual number of hours expended.

    - Financial bonuses or other incentives are then distributed based perhaps on the degree of variance between actual and "earned" hours, or strictly on "earned" hours alone.

- Another estimation technique is also based on division into modules, not by function but by size.

    - In organizations emphasizing structured and modular programming, there are already ground rules restricting each module, usually to the size that fits on one print-out sheet; i.e., 30-50 lines of high-level code.

- The number of modules in the system is established initially by functional estimation techniques, somewhat like the standardized-module data base discussed earlier.

- Modules are assigned several states of completion. For example:

  . Identified modules are those that have been identified by the requirements/specifications analysts and have had their name and function entered in a data base.

  . Documented modules are those for which a narrative description has been created and entered in the data base.

  . Programmed modules are those for which the coding has been entered into the data base.

  . Off-line tested are those modules that have undergone a specific number of test runs in the development environment.

  . Production-tested are those modules that have undergone a specific number of tests in the final production environment.

- Within the appropriate software environment, these states of completion can be determined accurately, objectively and automatically by the computer.

- A measure of the project's progress toward completion can be defined, for example, as the number of modules in production-tested status relative to those that have been identified.

● One organization that employs this technique reports that the number of identified modules generally grows rapidly in the early stages of the project.

- However, by automatically measuring the weekly or daily change in the number of identified modules, some measure of the completion of the identification stage can be obtained. When the number of identified modules begins to stabilize, ratios comparing other completion states against this number become more meaningful as measures of total project completeness.

- This technique is not without flaws, of course. For example:

  - The number of times a module has undergone testing does not necessarily reflect progress toward completion, unless testing is exceptionally well-managed and -controlled.

- Nevertheless, this technique does seem to offer at least two major advantages relative to current estimating techniques:

  - It relies on unobtrusive measurements, which require no conscious effort on the part of the DP professionals involved.

  - It measures progress in objective, self-adjusting terms.

E. DEVELOPMENT TOOLS AND AIDS

- A large variety of manual, semi-automated and fully computerized tools have been developed, especially in the last 10 years or so, to improve the system development process in one sense or another.

  - Tools and techniques that assist in the system testing and maintenance process have also been developed, but to a far lesser degree.

- Exhibit VI-11 provides a basic summary of the various classes of such tools, aids and techniques; lists their applicability; and gives some examples of specific implementations (mostly available commercially).

- It is not the purpose of this section or this study to provide a detailed analysis of these tools and techniques.

    - Instead, a few observations will be made about some classes that are perhaps misunderstood or under-utilized at present.

- INPUT's queries revealed the rather startling fact that many organizations are not availing themselves of any of these tools, as shown in Exhibit VI-12.

    - Forty percent of the respondents reported that they made no use of any such tools.

    - Fifty-three percent reported no use of any type of structured analysis or design methodology, as shown in Exhibit VI-13.

- On the other hand, the majority of respondents indicated use of a DBMS system. Exhibit VI-14 indicates that 67% use some standard DBMS.

    - Fifty-three percent reported the use of an inquiry language, although an examination of the exhibit shows that some respondents have apparently mistaken communications monitors (like IMS/DC and CICS) for inquiry languages.

- Inquiry languages can be extremely effective tools for reducing the backlog of projects the EDP department is facing.

    - Some inquiry languages make it possible for users who are not versed in computer lore to create with relative ease, small to medium applications for their own specific needs.

EXHIBIT VI-11

OVERVIEW OF DEVELOPMENT TOOLS, AIDS AND TECHNIQUES

CLASS OF TOOL/AID	NATURE OF TOOL, APPLICABILITY OR INTENDED USAGE	SPECIFIC EXAMPLES*
System Development Methodology	Checklists for all system life cycle phases; sometimes sold with tools that address the substance of specific phase activities.	CARA PRIDE/Logik SDM/70 SPECTRUM SYSTEMACS
Structured Analysis (SA) and/or Design Methodology	A formalized intellectual analysis discipline, generally supported by a specific graphic diagramming method, and generally emphasizing top-down or functional decomposition techniques and system modularity. Most popular SAs are data-driven, in that they emphasize the design of data structures or data flow as a basis for procedural design.	Yourdon/Demarco Gane & Sarson Jackson Warnier-Orr SADT HIPO
Development Team Organization	Formalized or semiformalized definitions of development team members' roles.	Chief Programmer Team Egoless Programming* SADT Team Matrix Organization*
Testing and Review Procedures	Improve the efficiency of design and code inspections, resulting in more robust, bug-free designs.	Structured Walk-throughs* Test Data Generators Static Flow Analyzers Correctness Proofs* Mutation Analysis*
Maintenance Techniques	Improve the efficiency of maintenance activities.	Scheduled Maintenance* SWAT Team*

EXHIBIT VI-11 (CONT.)

OVERVIEW OF DEVELOPMENT TOOLS, AIDS AND TECHNIQUES

CLASS OF TOOL/AID	NATURE OF TOOL, APPLICABILITY OR INTENDED USAGE	SPECIFIC EXAMPLES*
On-Line Development Tools	Improve the efficiency and productivity of the coding, testing and maintenance phases. The first three examples contain a more comprehensive set of integrated tools than the remainder of the list.	Programmer's Workbench Programmer's Workstation Maestro CMS TSO ICCF/ETSS ROSCOE TONE WYLBUR
Applications Generators (or "non-procedural" or "menu-driven" programming)	Systems which facilitate quick development of relatively simple business applications by providing pre-coded and pre-integrated subsystems, with some parameters left open for local customizing. Generally closely related to an adjunct data base management system.	ADF DMS TAPS TRANS-IV
Reusable Code	Systems consisting of a collection of pre-coded modules or partially coded skeletons that can be parametrized, customized and integrated into large-scale systems.	Raytheon (UPP) Kapur Associates In-house Systems*
Requirements Languages	Languages specialized for expressing system descriptions and specifications (not simulation or modeling) of EDP or general business systems.	PSL/PSA BIAIT
Code Generators, Pseudo Code Systems	Semiformal or formal languages that permit programming on a very high level or in a natural English-like fashion.	GENASYS PDL

* ALL EXAMPLES ARE OF COMMERCIALLY AVAILABLE PRODUCTS EXCEPT THOSE MARKED WITH AN ASTERISK.

EXHIBIT VI-11 (CONT.)

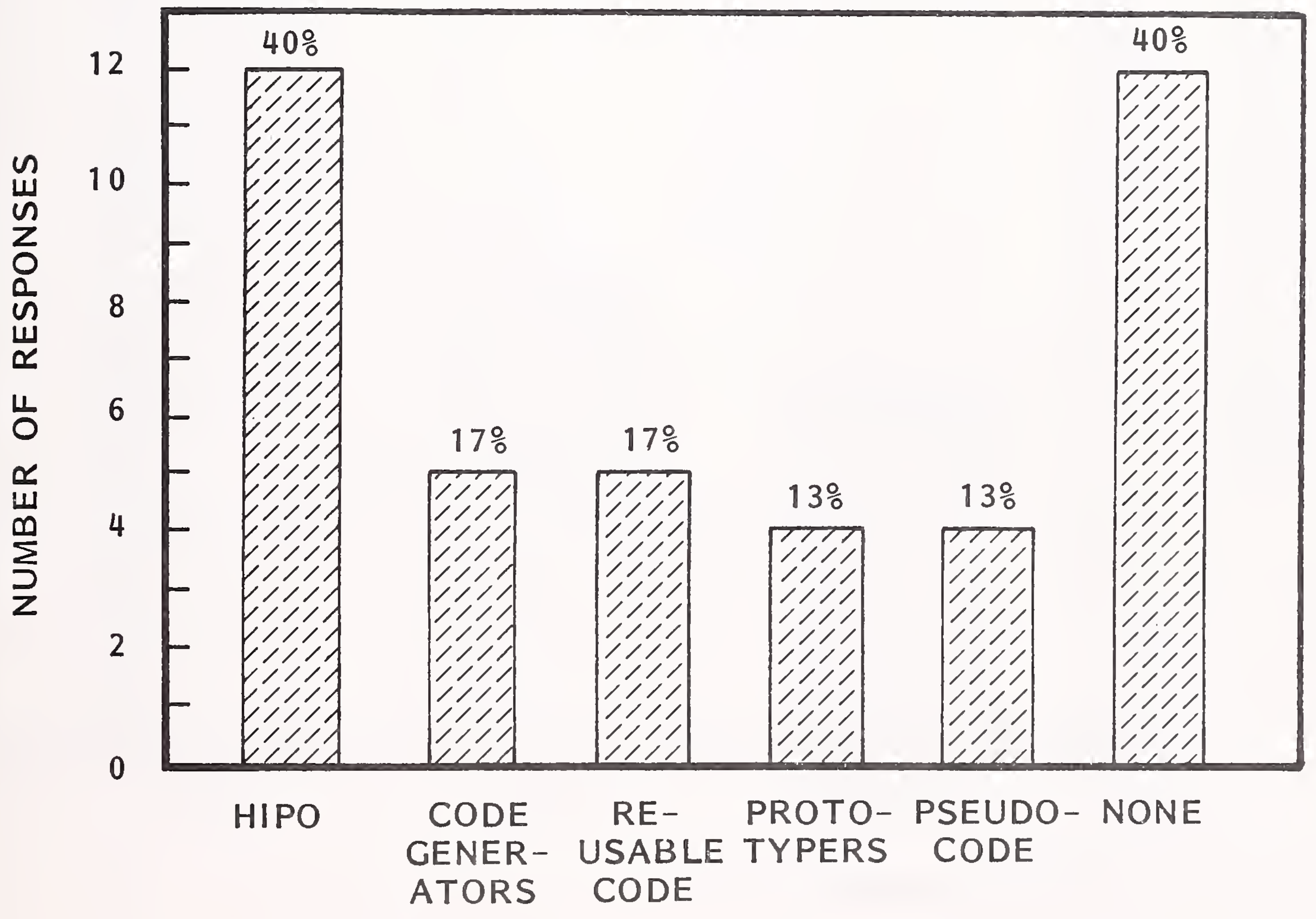
OVERVIEW OF DEVELOPMENT TOOLS, AIDS AND TECHNIQUES

CLASS OF TOOL/AID	NATURE OF TOOL, APPLICABILITY OR INTENDED USAGE	SPECIFIC EXAMPLES*
Inquiry/Retrieval Languages	User-oriented languages that permit easy definitions of systems of data base inquiries and report generation without involvement of the DP department.	EASYTRIEVE FOCUS NOMAD INQUIRY RAMIS
Structured High-Level Languages	Compilers that do not support "GO TO," but support essentially only the control structures (branching logic) permitted by the structured programming discipline.	PASCAL C ADA
Project Control System	Semi-automated techniques for tracking progress, resource expenditures and manpower utilization.	Nichols PAC-II PC/70

*ALL EXAMPLES ARE OF COMMERCIALLY AVAILABLE PRODUCTS EXCEPT THOSE MARKED WITH AN ASTERISK.

EXHIBIT VI-12

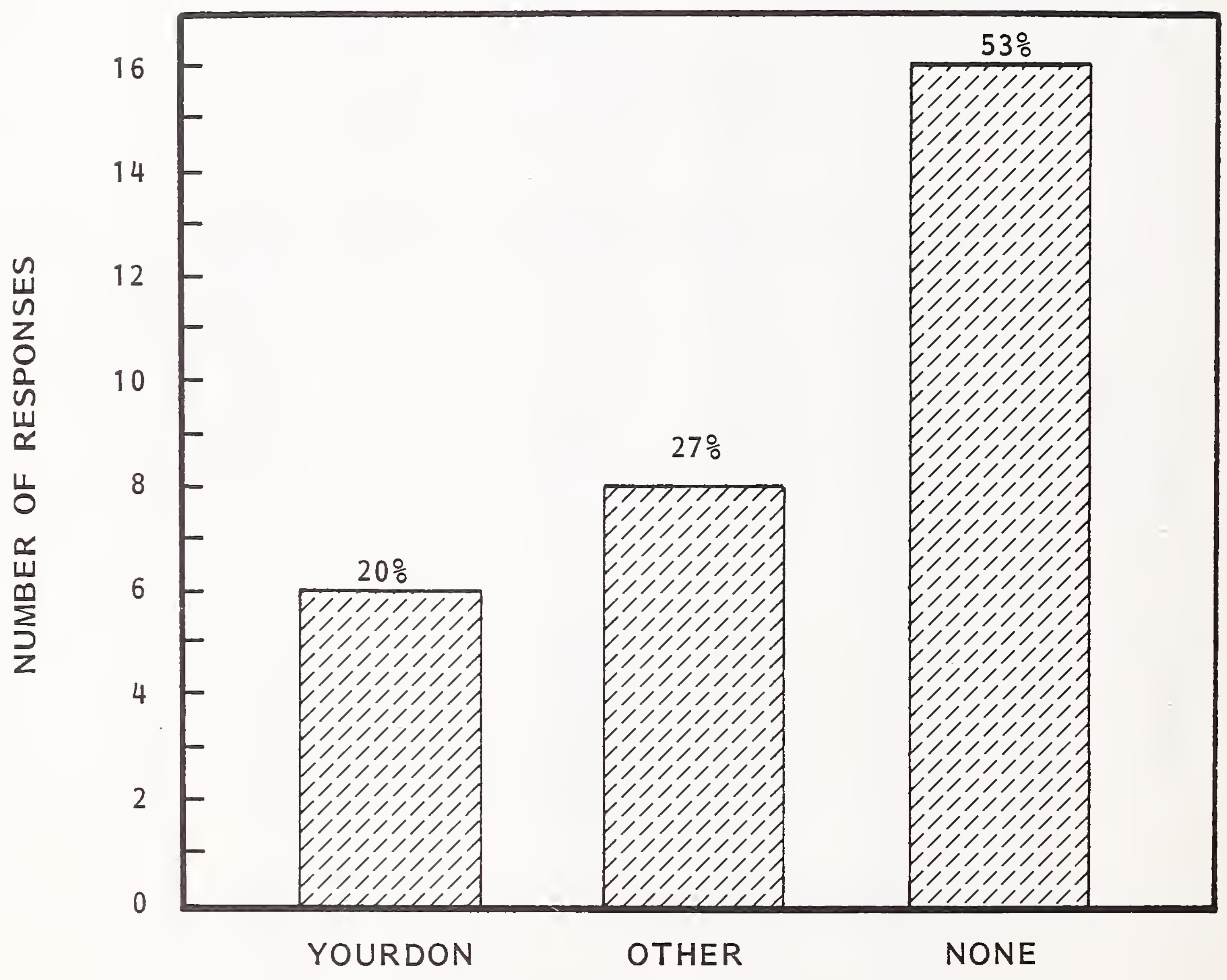
USE OF TOOLS AND AIDS



NOTE: MORE THAN ONE TYPE OF TOOL IN USE IN SOME INSTALLATIONS.
TOTAL RESPONSES: 30

EXHIBIT VI-13

USE OF STRUCTURED ANALYSIS,
DESIGN AND PROGRAMMING



TOTAL RESPONSES: 30

EXHIBIT VI-14

USE OF DBMS AND INQUIRY LANGUAGES

DATA BASE SYSTEM	NUMBER USING
IMS	6
IDMS	6
DATACOM	2
OTHERS	6
DO NOT USE	10
INQUIRY LANGUAGE	**NUMBER USING**
CULPRIT	3
IMS/DC	2
CICS	2
OTHERS	9
DO NOT USE	9
NO REPLY	6

TOTAL RESPONSES = 30

- The applications supported by these inquiry languages are generally limited to those that rely exclusively or heavily on the retrieval, manipulation, display and output of data already stored in a data base system.

- Most commercially available data base management systems today offer one or more inquiry languages as options.

• Inquiry languages are closely related to another tool that is beginning to command some attention in many organizations - the applications generator.

- These applications generators are also often called "non-procedural languages" because the user interface they present is so unlike the typical procedural languages, such as COBOL or PL/I. "Menu-driven programming" is perhaps a more accurate term.

- An applications generator is essentially a prepackaged system of typical business applications with some parameters and details left open to local customization.

- Applications generators sometimes support a unique, private data base system (as for instance in the case of TAPS), or they may have an interface to a commercially available DBMS such as IMS (e.g., the ADF applications generator from IBM).

- Unlike inquiry languages, which are generally limited to applications that interrogate but may not update data in the data base, applications generators generally permit the full range of retrieval and update activities; however, the relation between major subcomponents of the applications generator package is typically fixed and can only be parametrized to a limited extent.

- Still, such application generators are ideal vehicles for a number of important functions:

- They can be used as disposable code systems in which applications are worked out quickly to demonstrate to the user the type of screens and reports that will be available from a major proposed system.

- It is significant to note that, while 90% of the organizations INPUT interviewed reported that they encouraged end users to solve their own EDP problems, as shown in Exhibit VI-15, only 33% disclosed that they used such tools as ADF and DMS, as illustrated in Exhibit VI-16.

  - A possible conclusion is that the encouragement of user-implemented solutions to their own EDP problems is more imagined than real.

  - A stronger conclusion is that current tools do not adequately address user needs, so that potential use far exceeds actual use.

- Demonstration programs using powerful non-procedural tools can be far more effective than reams of specifications and diagrams because such documentation can rarely convey the dynamic behavior of the proposed system.

  - These tools can be used in much the same way as inquiry languages; i.e., to allow users to develop small- to medium-sized applications without involving DP department's personnel.

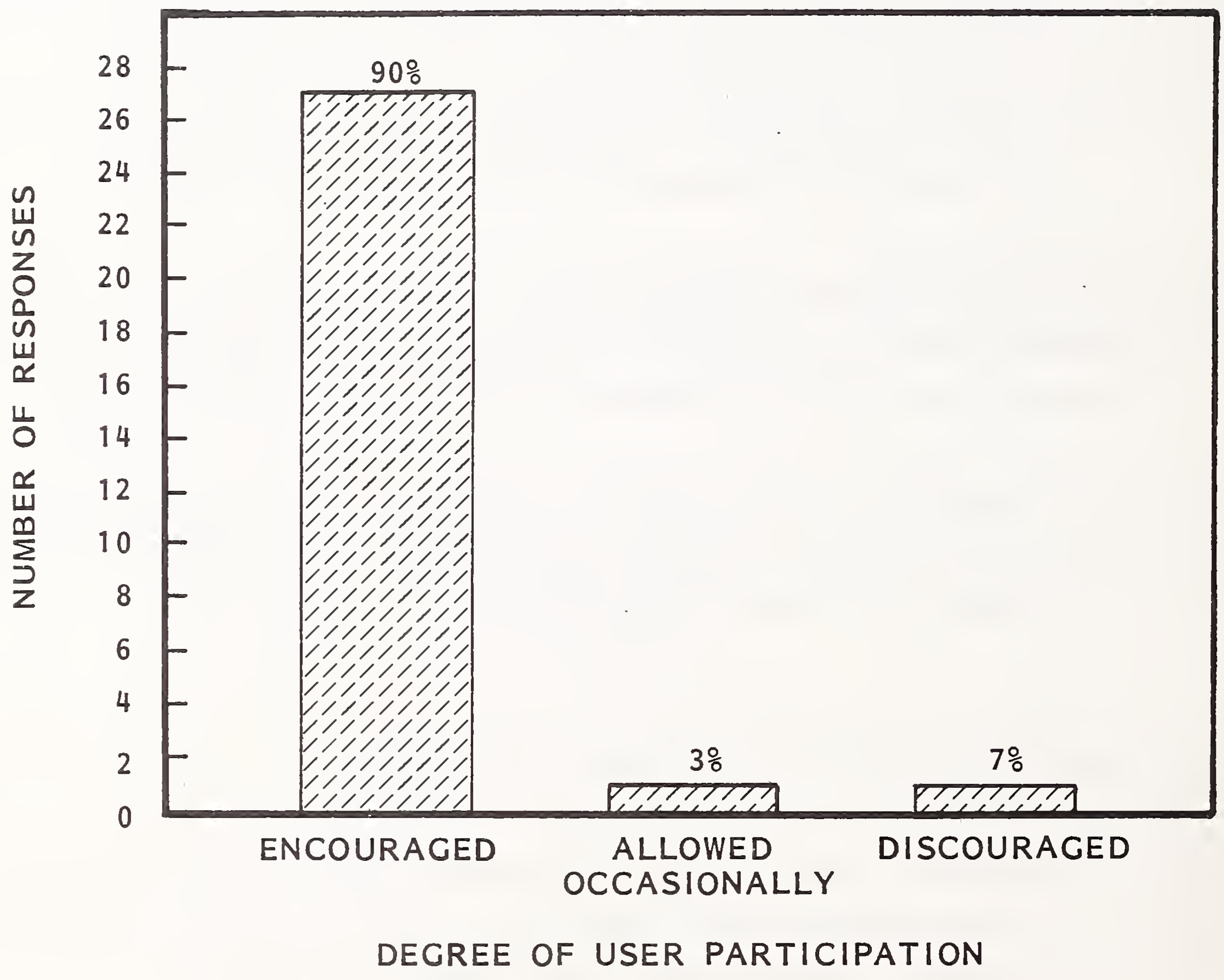
- The trend to on-line systems has become so overwhelming that the results presented in Exhibit VI-17 are perhaps superfluous:

  - Sixty-three percent of the respondents indicated that 75% or more of the process of program development in their organizations consisted of interactive operations rather than batch operations.

- Still, it is worthwhile to keep in mind that unrestricted on-line access can result in excessive resource utilization.

EXHIBIT VI-15

CORPORATE ATTITUDE TOWARD END USERS' SOLVING THEIR OWN EDP PROBLEMS



TOTAL RESPONSES: 30

EXHIBIT VI-16

USE OF ON-LINE DEVELOPMENT TOOLS

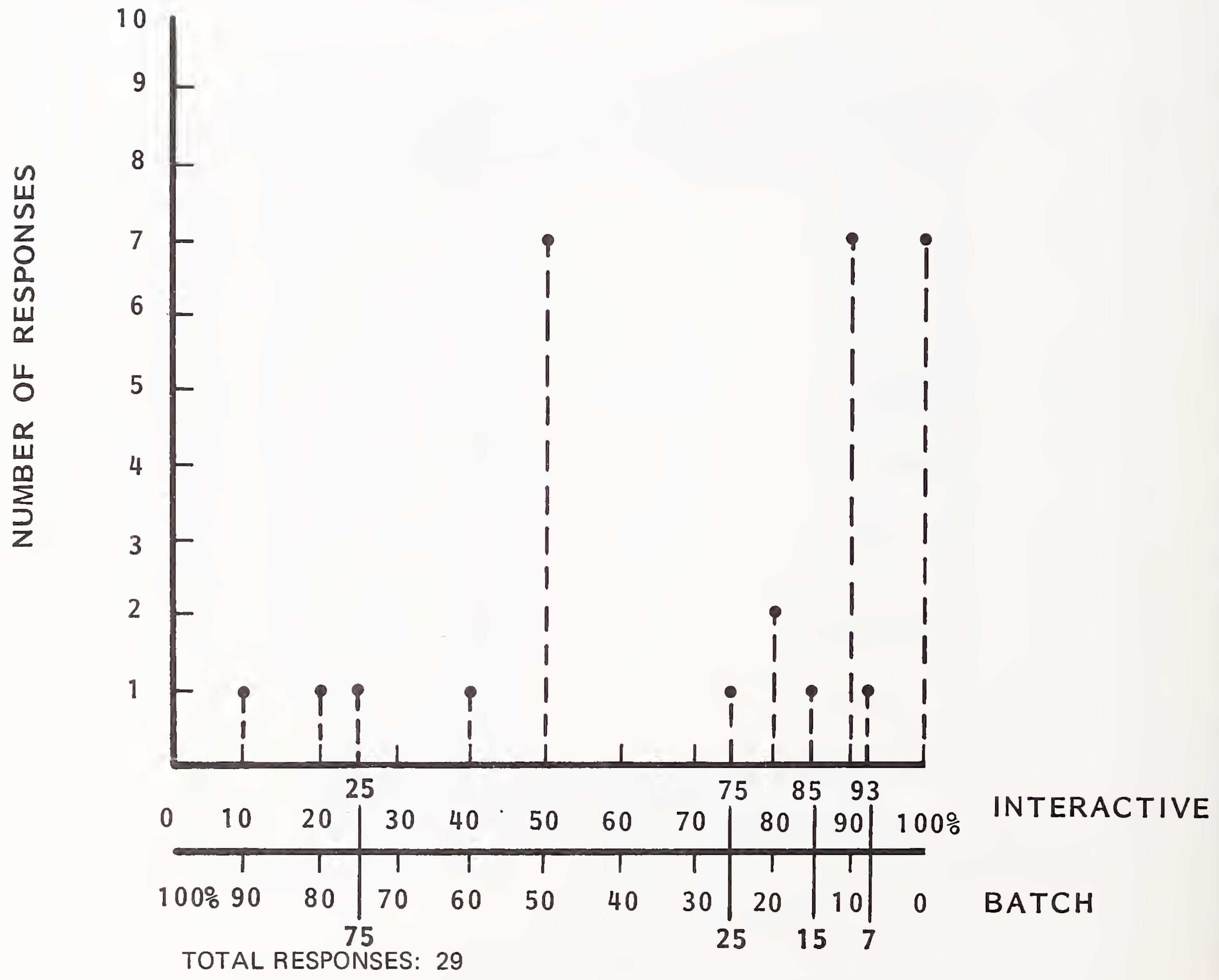
TOOL	RESPONSES
CMS	5
TSO	13
ADF	2
DMS	8
CICS	14
IMS/DC	4
OTHER	20

NOTE: SOME INSTALLATIONS REPORT MORE THAN ONE TOOL IN USE.

TOTAL RESPONSES: 30

EXHIBIT VI-17

PROGRAM DEVELOPMENT PROCESS:
INTERACTIVE VERSUS BATCH



TOTAL RESPONSES: 29

- There is also a nagging suspicion that excessive on-line access encourages sloppy analysis and design.

- Some techniques that counteract these trends are:

  - The mandatory use of structured walk-throughs and code inspections.

  - Reward systems for designers and programmers who achieve their tasks on time and use minimum amounts of computer resources.

  - Limiting access by various accounting mechanisms.

- INPUT's survey for this study revealed a relatively large proportion of organizations reporting the use of the Chief Programmer Team (CPT) concept, as shown in Exhibit VI-18.

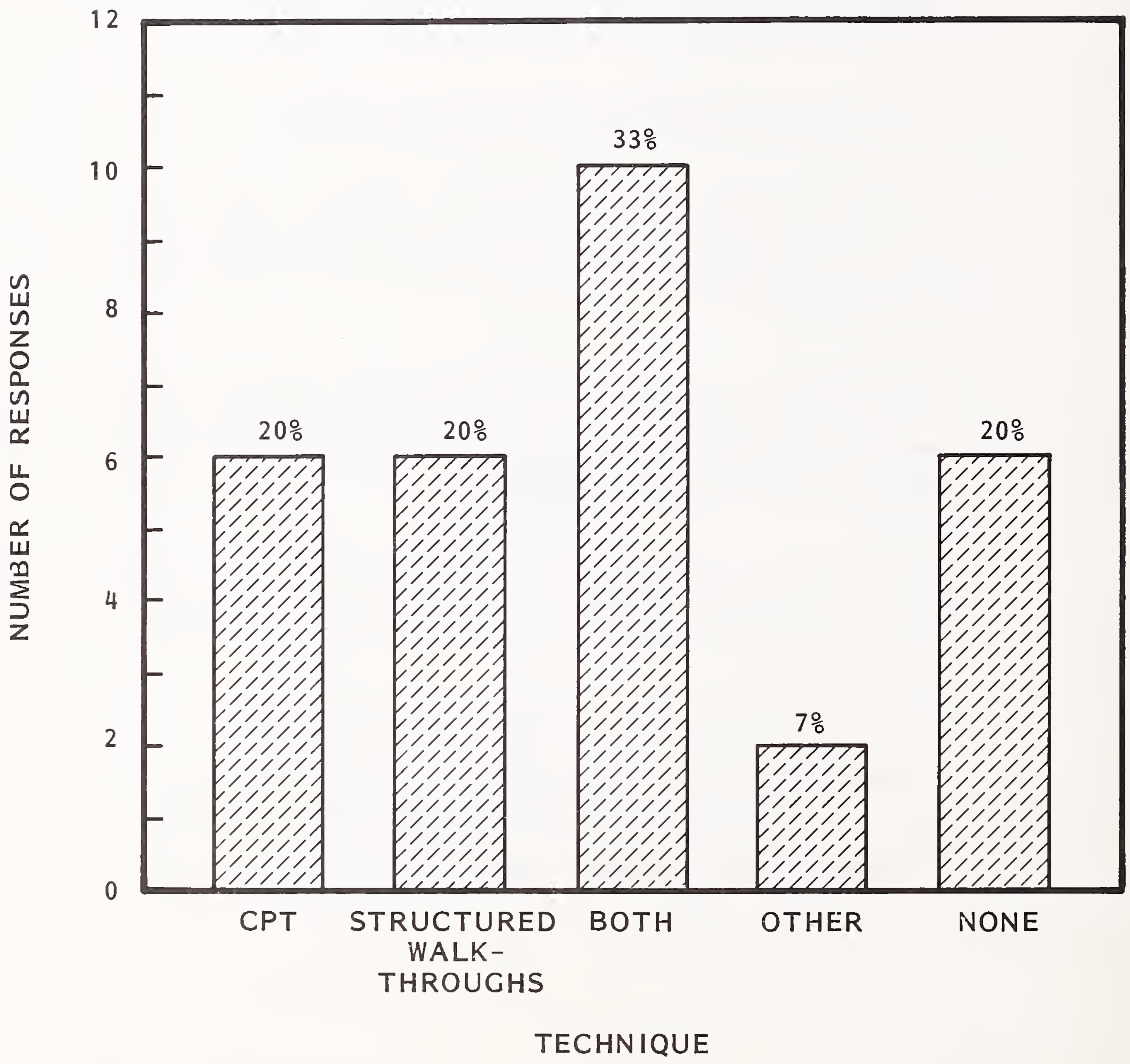
  - Fifty-three percent of the respondents reported that their organizations used the CPT concept, either alone or in conjunction with the structured walk-through review mechanism.

  - Other studies done to date have revealed little interest in CPT, often because of the lack of personnel qualified to lead such teams.

  - Where used, the concept has been frequently modified to eliminate the librarian and other paraprofessional support positions. This modification is strongly criticized by organizations that have implemented the full concept, but comparative productivity rates are inconclusive.

EXHIBIT VI-18

USE OF CHIEF PROGRAMMER TEAMS (CPT) AND STRUCTURED WALK-THROUGHS



TOTAL RESPONSES: 30

F. MAINTENANCE ISSUES

1. LIFE CYCLE BREAKDOWN

- Exhibit VI-19 summarizes the respondents' estimates of the life cycle percentage breakdown between development and maintenance in their organizations.

- While the actual figures for this breakdown varied substantially - from 10% development, 90% maintenance to 90% development, 10% maintentance - the average of 52% development versus 48% maintenance matches rather well the results from a recent study by Lientz and Swanson.

  - They queried 482 business data processing departments and concluded that the breakdown was 43.3% development versus 48.8% maintenance, with 7.9% classified as "other."

- The important point is that there is apparently a general agreement in the industry that maintenance forms a very substantial part of the life cycle.

  - Other studies and estimates by INPUT have placed the figure at 70% to 90% maintenance for the "mature" industries; i.e., banking/finance and insurance.

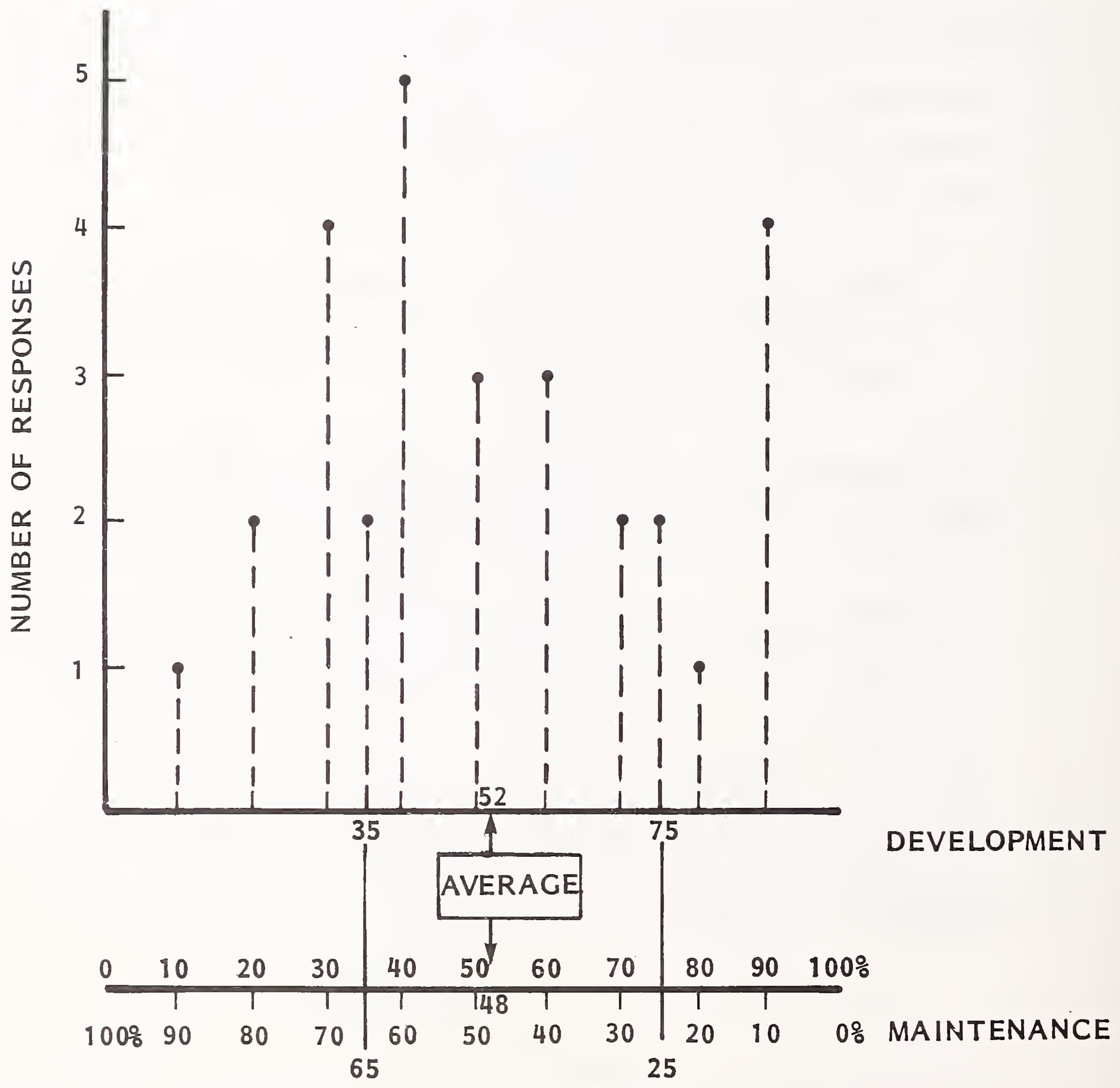
- One of the clear implications that ought to concern the EDP executive is that the management of maintenance activities is at least as important as development management.

  - Half the costs or more are in the maintenance.

- Unfortunately, most of the work that has been done up to now in improving EDP operations has concentrated almost exclusively on the development portion.

EXHIBIT VI-19

LIFE CYCLE BREAKDOWN:
DEVELOPMENT VERSUS MAINTENANCE



TOTAL RESPONSES: 29

- As a result, the tools, aids and techniques that have been developed over the past ten years or so have been all oriented toward the development process.

- The only exceptions are the SDMs (System Development Methodologies), which generally address both development and maintenance.

- However, SDMs are merely checklists that offer little tangible help in the substance of the maintenance problem.

- A surprising number of the organizations INPUT interviewed felt that they were satisfied with their current breakdown between maintenance and development, as can be seen in Exhibit VI-20.

  - Twenty-one respondents (70%) were satisfied, with only nine respondents (30%) registering dissatisfaction.

  - The dissatisfaction related almost entirely to the amount, rather than the quality, of maintenance.

2. MAINTENANCE STRATEGIES

- Maintenance is a problem because it is generally regarded as less desirable work by most DP professionals.

  - Working to correct or enhance someone else's creation is unattractive to most people.

- There is a tendency among EDP executives to assign less-qualified personnel and/or new hires to the maintenance activity.

  - This is perhaps the result of insufficient appreciation of the true relative importance of this activity in terms of costs to the organization.

EXHIBIT VI-20

LEVEL OF SATISFACTION WITH CURRENT DEVELOPMENT/MAINTENANCE BREAKDOWN

CLASSIFICATION	NUMBER OF RESPONSES
● SATISFIED WITH CURRENT BREAK-DOWN	21
- UNDER 50% MAINTENANCE	8
- 50-75% MAINTENANCE	11
- OVER 75% MAINTENANCE	1
- UNSPECIFIED MAINTENANCE	1
● DISSATISFIED	9
- UNDER 50% MAINTENANCE	4
- 50-75% MAINTENANCE	3
- OVER 75% MAINTENANCE	2
TOTAL RESPONSES	30

- This certainly is a response to the realities of the situation: assigning high-caliber people to maintenance runs the risk of losing them.

- But perhaps the primary reason is that EDP executives feel that by assigning their best people to the development phase, project development - which is by far the most visible part of the project to both end users and top management - will be done better and in less time.

- For whatever reasons, the assignment of new hires and less-qualified people to maintenance activities serves to underline the undesirability of the job in the eyes of those who are doing the maintenance as well as the rest of DP personnel and outside departments.

- There are several practical ways in which maintenance activities can be made more desirable to the people involved:

  - Rotate development personnel, including top performers and effective first-level managers, through the maintenance activity.

  - Make maintenance the responsibility of the developers.

  - Teach programming and analysis from the maintenance point of view; don't imply that new development is more challenging or more rewarding.

  - Provide special pay or fringe benefit incentives for maintenance personnel.

  - Attack maintenance with "SWAT" teams.

- For those individuals who respond to the problem-solving aspect of the work, maintenance is a more fertile field than new development.

- When frequent changes to a single system are required, the improvements achieved by implementing a table-driven approach, or creating a software tool to solve the problem once and for all, are valuable from both a psychological and a productivity point of view.

- Teaching new trainees to work more intelligently, not longer, is bound to pay off as this attitude eventually pervades the organization.

- Rotating development personnel through the maintenance department can be effective if accompanied by a demonstrated management commitment to the importance of the maintenance activity. The incentive programs described below are one such demonstration:

  - Rotating assignments have the added benefit of exposing the developers to some of their own mistakes and their consequences, and might give these individuals some insight into ways to avoid such mistakes in the future.

  - Furthermore, new hires then have a chance to work with the better performers and learn by example.

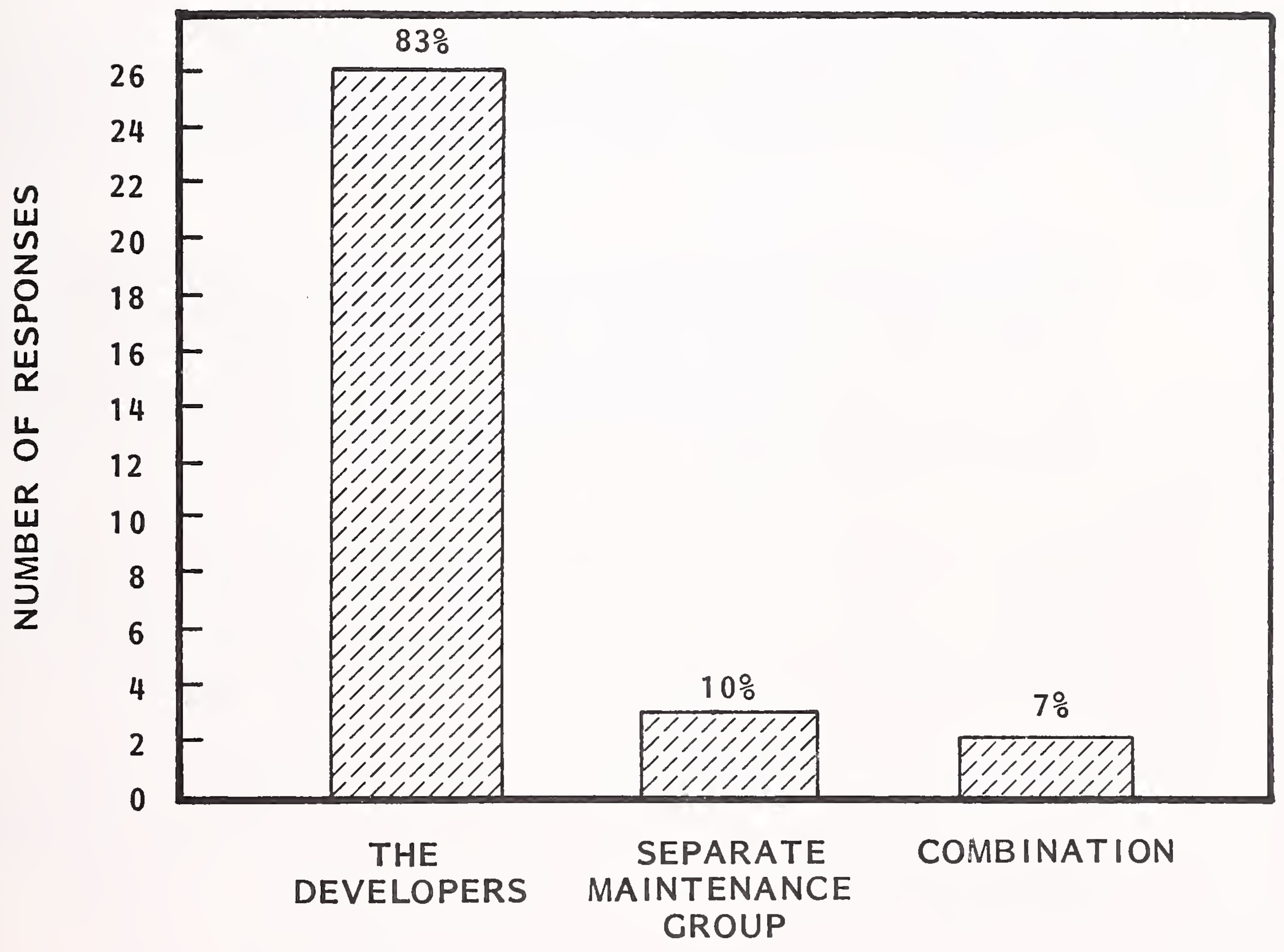
- It might be argued that, if all the above is true, then the best maintenance strategy is to make it the exclusive responsibility of the developers; i.e., to abolish, or never to establish, a separate maintenance group.

- In fact, as INPUT's survey reveals, the majority of organizations interviewed (83%) do operate in this mode.

  - Only 10% have a specific maintenance group, while 7% more practice a combination of both methods. Exhibit VI-21 illustrates the point.

EXHIBIT VI-21

RESPONSIBILITY FOR SOFTWARE MAINTENANCE



TOTAL RESPONSES: 30

- This is an example of a situation where the majority may not be correct, but the difficulty of achieving a successful separate maintenance operation is widely considered not worth the large management effort required to start it.

- Incentive programs for maintenance personnel might include:

  - Some financial bonus arrangement, based perhaps on meeting time and cost goals for enhancements, and on response time for fixing bugs.

  - Fringe benefits such as permission to set their own time schedules, work at home with free terminals, extra days off, etc.

- The "SWAT" team approach calls for handling maintenance chores by small teams of the best performers, assisted by less-qualified personnel and new hires.

  - Routine, noncritical maintenance is accumulated as "back orders" until a SWAT team can be formed to attack the entire batch. At the same time, the programs are brought up to current documentation and programming standards.

  - Ideally, this "scheduled" approach to maintenance should be reflected in the staffing requirements, so that SWAT teams can be formed without having to borrow personnel from urgent development efforts.

  - The SWAT team might attack the noncritical bugs and enhancement requests on a rotating schedule in which each major system in operation is addressed in turn.

  - Organizations that have tried this scheduled-maintenance approach generally report excellent results in terms of both user satisfaction and enthusiasm on the part of the DP personnel, as long as the effort does not extend over too long a period; i.e., more than two months.

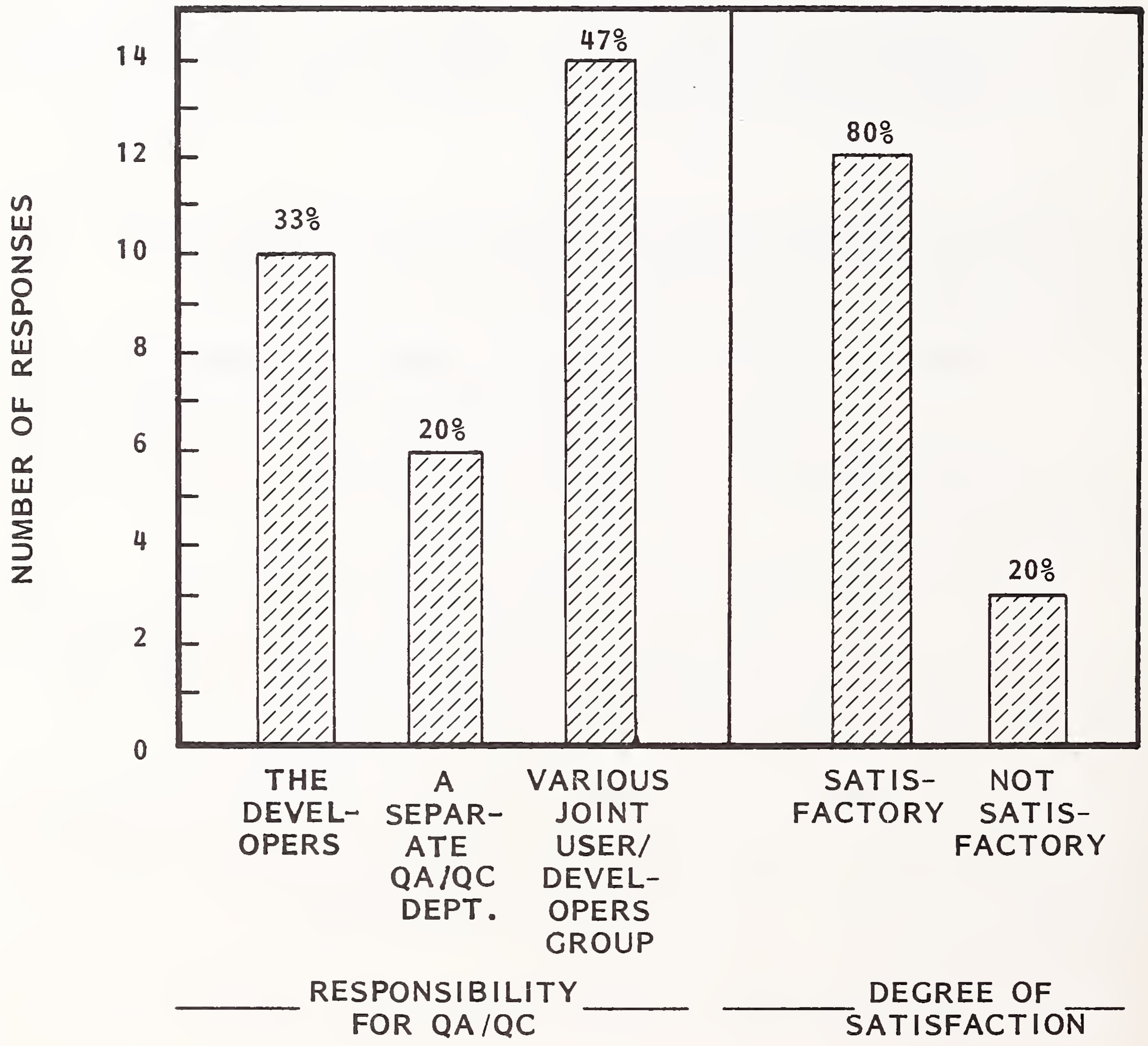
- Needless to say, critical maintenance tasks are best handled by senior personnel regardless of the method of handling routine maintenance.

3. QUALITY CONTROL

- Quality of systems produced has a direct impact upon maintenance for obvious reasons.
  - There is general agreement in the industry that the longer an error remains in the system undetected, the more costly it will be to detect and correct.
  - Passing all systems through rigorous QA/QC procedures before declaring them operational is of value if it leads to early detection of the maximum number of errors.
- Ideally, a QA/QC department with its own reporting structure and financing would appear to be the best mechanism for handling this task.
  - When the people doing QA/QC are subject to the same time, cost and business pressures as the developers' department, they are not likely to be as thorough or effective as they could be if isolated from such pressures.
  - On the other hand, a QA/QC organization operating as an ivory tower, completely divorced from the business goals of the organization as a whole, can create more problems than it solves.
- If the actual state of affairs in the industry as a whole is reflected by the responses shown in Exhibit VI-22, then it would appear that the QA/QC task is handled primarily by various joint user/developer groups.
  - Forty-seven percent of the organizations interviewed said that was the case.

EXHIBIT VI-22

RESPONSIBILITY FOR SOFTWARE QA/QC



TOTAL RESPONSES: 30

- Thirty-three percent reported that the QA/QC function was entirely in the hands of the developer. This probably means that no serious QA/QC function exists.

- Only 20% reported that they maintained a separate QA/QC department.

• That quality control is sadly lacking in many organizations is revealed by the breakdown shown in Exhibit VI-23 of the maintenance activities between repair and enhancement, as reported by the 30 organizations INPUT interviewed.

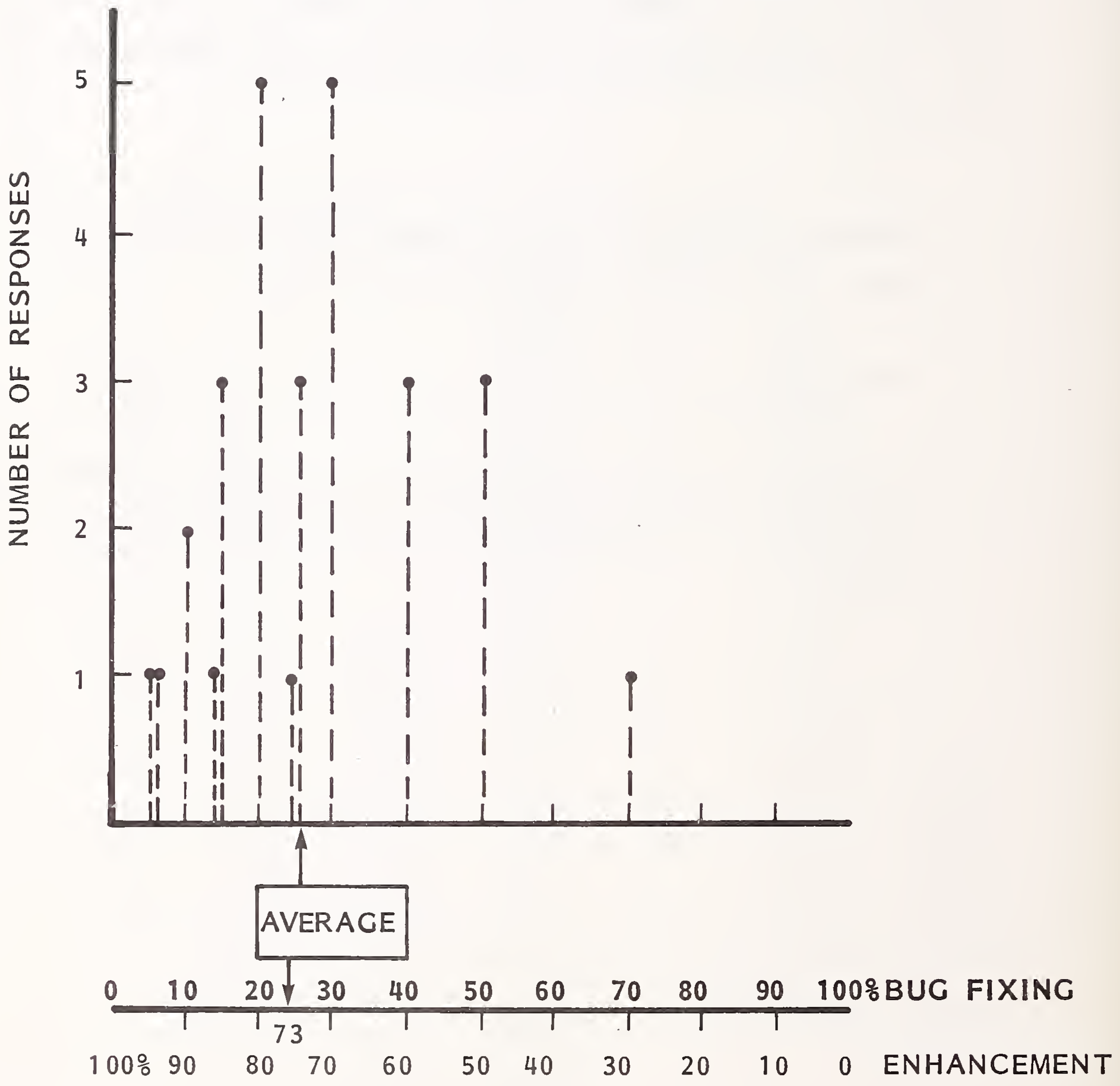
  - Twenty to thirty percent of the maintenance effort is reported to involve bug fixing, with the remainder being devoted to enhancements. Half of the respondents' replies fell in that range.

  - The average figure is 27% bug fixing and 73% enhancements.

  - This level of warranty work would put most production line managers out of a job. Software production managers ought to draw a timely lesson.

EXHIBIT VI-23

BREAKDOWN OF THE MAINTENANCE PHASE:
BUG FIXING VERSUS ENHANCEMENT



APPENDIX A: DEFINITIONS

APPENDIX A: DEFINITIONS

- **ADF - APPLICATIONS DEVELOPMENT FACILITY** ADF is an optional IBM IUP (Installed User Program) that works in conjunction with IMS and provides a menu-driven programming environment. Dramatic improvements in programmers' productivity (better than 10 to 1) have been reported for ADF, at the cost of about 50% more CPU time than equivalent IMS applications coded in COBOL. See Menu-Driven Programming.

- **BIAIT - BUSINESS INFORMATION ANALYSIS AND INTEGRATION TECHNOLOGY** BIAIT, developed by Donald C. Burnstine of BIAIT International, Petersburg, NY, is a formal discipline that provides structure and reproducibility in the problem analysis phase of business-oriented data processing work. The key concept underlying BIAIT is that the information-handling processes involved in many types of businesses can be described in terms that are descriptive of business <u>information handling</u> rather than EDP-related functions. A model of the business is expressed in a BICMX (Business Information Control Matrix), which arrays business functions against the "inventories" that each function keeps track of. Every "inventory" can be broken down into steps of its generic information life cycle. Within the matrix, users, suppliers, authority and responsibility are interrelated by information interfaces. The system has had some use within and outside IBM.

- **CARA** CARA is a commercial, manual system design methodology contained in two volumes, a general reference card and preprinted forms. The system defines what tasks must be done and when, the deliverables at the conclusion of each life cycle phase and the documentation standards. The system is priced at $22,500 and is marketed by CARA Corp., 1010 Jorie Blvd., Suite 124, Oak Brook, IL 60521, (312) 654-2405.

- **CHIEF PROGRAMMER TEAM** The Chief Programmer team is an organizational structure for the development of software systems, developed by the Federal Systems Division of IBM in 1969-1971 and later released as part of the Improved Programming Technologies package. In this scheme, software is developed by small teams, each consisting of a minimum of three and a maximum of five persons. The team is headed by the Chief Programmer, who does the design, codes the highest level of implementation and supervises the work of additional programmers who code "stubs" (subroutines or lower levels). The Backup Programmer aids the Chief Programmer and is ready to assume the role of Chief Programmer, should it become necessary. The Programming Secretary performs the clerical work according to a highly disciplined routine that makes all source listings, computer run results and data "publicly available" in a Programming Production Library (PPL) (also called DSL - Development Support Library) and in archive records. The CPT system relies on structured programming and top-down design concepts, and is claimed to have resulted in doubling programmers' productivity and in enhanced reliability and maintainability of the resulting code. IBM reported that a five-person CPT is expected to produce up to 25,000 lines of source code in 12-18 months, or three times the "normal" rate.

- **CMS - CONVERSATIONAL MONITOR SYSTEM** CMS is a "native" component of the VM/370 operating system for IBM mainframes. It is oriented toward the support of a single terminal engaged in program development and debugging activities. By running multiple "CMS machines," a system under VM/370 can provide the same type of interactive, multi-user program development environment that is the object of TSO under MVS.

- **CORRECTNESS PROOFS** Correctness proofs are attempts to demonstrate the "correctness" of programs by rigorous mathematical methods. Correctness usually means the ability of a program to carry out a specified task and only that task; i.e., without errors. The most common approach to this problem has been to define the program specifications and the program steps in terms of function theory, a mathematical discipline. Structured programs are especially well suited to being stated in such terms. The method has so far proven impractical for large programs, because the effort involved in creating the proof is, as a rule, far greater than the effort required to debug the program by empirical techniques.

- **DBMS - DATA BASE MANAGEMENT SYSTEM** A software system intended to centralize the creation, control and maintenance of data files, so that multiple application programs can access the data without having to create duplicate private file systems. DBMS generally involves a Data Definition Language - DDL - to create the data base, which is typically under the control of one person or department, called the data base administrator. The accessing of the data base is done through a Data Manipulation Language (DML) which can be embedded in a programming language like COBOL, or can be a standalone language, as in many data base query/retrieval languages. A great deal of standardization work in the area of data base management has been done by the CODASYL Data Base Task Group. A well-designed data base with an associated on-line query/retrieval system generally results in significant reductions in the number of special coding projects that a DP department is required to do to satisfy users' needs for specialized reports. The term DBMS is used by DEC as the name of a specific data base system for the PDP-10 and its successor 36-bit systems.

- **DMS - DEVELOPMENT MANAGEMENT SYSTEM** DMS is a menu-driven program development system from IBM that permits nonprogrammers to create useful systems by supplying fill-in-the-blank specifications either interactively, via 3270 terminals, or off-line, via specially printed forms. The system relies on CICS for terminal monitoring and control, and is available for IBM mainframes under DOS or OS environments. A version of DMS is also available on the IBM 8100 and 3790 systems. See Menu-Driven Programming.

- **DSL - DEVELOPMENT SUPPORT LIBRARY** DSL is a formalized, computerized archiving system developed by IBM's Federal Systems Division, which takes the view that all computer-run results, source listings, test data, etc., are "public records" that should be highly visible to both project personnel and management. This is in contrast with the more typical practice where individual programmers keep private records and the results of "bad" runs - which may contain useful data for the project - are often discarded. DSL was made formally a part of the Chief Programmer Team concept (see entry) under the name PPL - Programming Production Library.

- **FOCUS** FOCUS is an integrated data base management system and query language. It requires TSO or CMS for terminal control. It maintains its own data files, although IMS files can be operated on as an option for the TSO version. In addition to the data management facility and report generator, FOCUS also incorporates a transaction processor. The query language is close to natural English. Optional features include a host language interface (to allow access to FOCUS files from COBOL, Fortran, PL/I or BAL); a graphics subsystem; statistical analysis and financial modeling packages; and the IMS interface. FOCUS is marketed by Information Builders Inc., 254 West 31st St., New York, NY 10001, (212) 736-4433. Prices begin at $46,000 for the basic package, including data management, report generator and transaction processor.

- **FUNCTIONAL DECOMPOSITION** When used in conjunction with design or programming methodologies, this term implies a top-down design technique (see entry). The term actually originated in the theory of functions - a mathematical discipline - where it has a more limited and precise meaning.

- **GANE & SARSON** A design methodology, one of several offshoots of the ideas first proposed by Larry Constantine and augmented by Edward Yourdon. This particular branch was enhanced by Chris Gane, one-time Vice President at Yourdon, and Trish Sarson, also a past member of the Yourdon team, who are now the principals of Improved Systems Technologies, Inc., where they offer a system development methodology known as STRADIS, plus training seminars and workshops in its use.

- **HIPO - HIERARCHY PLUS INPUT PROCESS OUTPUT** HIPO is an IBM-originated program-design aid and documentation technique, part of a package of tools dubbed IPT - Improved Programming Technologies - released by IBM in the early 1970s. HIPO is intended primarily as the documentation system in a top-down design of systems. It consists of a chart showing the hierarchy of processes in the system (the "H" chart); for each process, there is also an Input-Process-Output ("IPO") chart. Unlike other data-oriented structured design methodologies (e.g., Jackson, Warnier-Orr), HIPO is primarily process-oriented and does not indicate sequence relationships.

- **IMS - INFORMATION MANAGEMENT SYSTEM** IMS is the primary IBM data base management system for the 360/370, 303X and 4300 hardware series. It has an associated telecommunications monitor, generally called DC or IMS/DC, which interfaces the on-line terminals with the applications programs using IMS in much the same fashion as CICS does for its application programs.

- **JACKSON DESIGN METHODOLOGY** This program design methodology, developed by Michael Jackson in England, is based on the structured programming constructs (sequence, iteration and selection) and takes as its departure point the design of data structures. Program structures are designed to match the data structures, with special provisions to resolve cases where such a match is impossible ("structure clashes"). A simple diagram system, consisting of boxes and lines, is used to represent both data and process structures. Based on Jackson's method, augmented by top-down implementation and structured walk-throughs, Exxon Corporation developed the PST (Program Structure Technology), supported by an interactive graphics package called PSTAIDS.

- **MAESTRO** Maestro is a standalone "programmer's workbench" system, developed by Softlab GmbH of Munich, Germany, to run on Four-Phase computers. Among the tools offered by the Maestro system are: "Structograms" for structured COBOL programming; HIPO charts; interactive construction of decision tables, with automatic checking of completeness, inconsistencies and redundancies; syntax guidance for COBOL programs; and the standard text-editing features, including searching, thumbing and marking. Programs developed under Maestro are generally intended to run on IBM 370/303X mainframes. The system was, until recently, marketed in the U.S. by ITEL; it is now marketed by Maestro Systems, 3 Embarcadero Center, San Francisco, CA 94111, (415) 986-1188.

- **MATRIX ORGANIZATION** An organizational scheme in which specific projects are staffed by personnel from functional departments. Project people continue to report to the functional department heads but perform project-oriented tasks for the project leader. Matrix organization is an attempt to combine the longer-term concerns that are addressed by a functional organization with the short-term, project-oriented goals of a pure project organization. There are many variations of matrix organization. Professional project leaders may have business responsibility for a number of unrelated projects, while functional group managers, relieved of day-to-day project concerns, are free to concentrate on recruiting, training, standards and career development.

- **MENU-DRIVEN PROGRAMMING** A generic term describing systems in which some or all of the traditional procedure-oriented programming tasks are replaced by "fill-in-the-blank," menu-driven, interactive sessions with the user. Examples include the IBM DMS/VS and IMS/ADF systems (see entries). The recently announced NOCODE system by General Automation is another example of a menu-driven coding system. Wang computers employ menu-driven programming to replace traditional JCL sequences. The term "nonprocedural programming" has been applied to menu-driven systems in some literature; INPUT prefers to use "menu-driven programming" to avoid confusion with such nonprocedural (mathematical) languages as SIMSCRIPT.

- **PDL - PROGRAM DESIGN LANGUAGE** PDL is a high-level pseudo-code system designed to aid the production of structured, top-down designs. It combines plain English vocabulary with syntactical constructs representing the basic structured programming elements of iteration (DO) and selection (IF). A computer program is available to record system descriptions expressed in PDL and to produce reports, giving cross-references between program segments and using the standard indentation technique to show levels of nesting. PDL is close enough to actual code that such code can be produced from the PDL description quickly. PDL has no special facilities for describing data structures.

- **PRIDE-Logik/ASDM** PRIDE-Logik/ASDM is a system design methodology marketed by Milt Bryce Associates of Cincinnati, OH. The degree of computerization varies from none in PRIDE itself (Profitable Information by Design) to some when Logik is added (Logical Organization and Gathering of Information Knowledge); the combination is then called ASDM - Automated System Design Methodology. Additional computerization takes place with the addition of ADF - Automated Design Facility (a data dictionary manager) - and Genasys (a code generator). Key concepts of the system are: structured system and data component design, recognition of five levels of system components and six types of data components, system-data relationships, and a nine-phase design/development cycle. When fully configured, the system, which is written in COBOL, has its own data base, which is created from input forms furnished by the designer. It can produce a variety of reports, charts and documentation and perform checks for completeness and accuracy. The system runs on IBM 370, Burroughs, Honeywell, CDC, DEC, ICL and Univac mainframes and on Hewlett-Packard 3000 minicomputers.

- **PROGRAMMER'S WORKBENCH** The term Programmer's Workbench is used both as a generic name for a class of on-line program development systems, and specifically for a particular implementation of the concept under the UNIX operating system for the DEC PDP-11 line of computers. PWB/UNIX was developed by Bell Laboratories to give programmers a variety of powerful on-line tools for program development, documentation and testing. The product is marketed "as is" (i.e., without ongoing support) by Western Electric, and is also licensed to resellers who add features and offer additional documentation and support. Other products that fall in this category include Maestro (see entry) and Programmer's Workstation, both of which run on Four-Phase equipment, but only the latter is marketed by Four-Phase. PWB/UNIX can be used to develop programs for UNIX, as well as for IBM and other "host" systems under an RJE arrangement. Maestro and PWS are intended specifically as standalone program development machines for IBM host systems.

- **PSEUDO-CODE** Pseudo-code is a generic term, describing a language that generally combines structured control constructs (IF-THEN-ELSE, DO WHILE) with free-form, natural-language English. It is also the name of a specific implementation, released as part of IBM's IPT (Improved Programming Technologies). The term Metacode has also been used for this type of language, of which a prime example is PDL (see entry). The idea is to express the functions required at each step of a proposed program in natural English, while retaining the control (branching) mechanisms of the eventual target programming language. Pseudo-code is more readable than a COBOL or FORTRAN program, and at the same time can be translated into the target language very quickly.

- **PSL/PSA - PROBLEM STATEMENT LANGUAGE/PROBLEM STATEMENT ANALYZER** PSL is a language for describing information processing systems; PSA is a computer program that analyzes PSL descriptions and produces reports and messages. Both have been developed under the direction of Daniel Teichroew and Ernest A. Hershey, III of the ISDOS project at the University of Michigan. PSL is based on the model of an abstract system, consisting of objects, which may have properties, which may in turn have property values. The objects may be interconnected in various ways, which are called relationships. The PSA builds a data base from the system description expressed in PSL, and can produce a number of reports, including modification history, a list of all objects with their type and date of last change, a list of all properties and relationships for a particular object, and analyses of gaps in the information flow or unused data objects. A graphic (flow-chart) description of the dynamic behavior of the system can also be produced automatically by PSA. PSA, which is coded primarily in FORTRAN, is operational on IBM 370, Univac 1100, Honeywell 600/6000 and PDP-10. The PSA software is available to participants in the University of Michigan-sponsored PSL/PSA research; participation costs $15,000/year.

- **REUSABLE CODE** Reusable Code is both a generic term, describing systems in which standardized functions are coded into universally available code modules, which are catalogued in a dictionary and filed in a data base; and a specific product, to be offered by Raytheon. The Raytheon Reusable Code System consists of four main elements: a logic shell generator, based on structured design; a reusable code library containing some 1,500 modules; a reusable code dictionary, describing the modules so that the desired ones can be located by keyword search; and a productivity measurement system. Kapur Associates of Danville, CA, is also marketing a reusable code methodology.

- **SADT - STRUCTURED ANALYSIS DESIGN TOOL** SADT is a proprietary, structured design methodology based on the structured analysis concepts of Douglas T. Ross. The system is marketed by SofTech, 460 Totten Pond Rd., Waltham, MA 02154, (617) 890-6900. SofTech does not market any computerized assist, but a tool is available on Cybernet. The SADT methodology employs a simple but highly disciplined graphic "language" to describe successive levels of action and of data. A key concept, forcing terseness and clarity, is that a "bound context" limits each "box" in the charts to dealing with a completely separate activity, related to the rest of the system by well-defined input(s), output(s) and control(s). Terseness and hierarchical clarity are also promoted by the rule that limits each page to no more than six boxes and each box or arrow label to no more than six words. A team with a clearly defined mission for each member ("authors," "readers," etc.) is also part of SADT; it is somewhat similar to the Chief Programmer Team from IBM. SADT is being promoted as an aid in system requirements and specifications, a system design discipline and a documentation system. The tool is especially popular in Europe.

- **SDM/70 - SYSTEMS DEVELOPMENT METHODOLOGY** SDM/70 is a commercial, manual system development methodology, consisting of six volumes covering the following topics: system life cycle and methods; documentation standards; estimation guidelines; administration, management and executive involvement guidelines; quality assurance; user involvement guidelines; and tutorials on such subjects as structured design, HIPO, data base management, etc. The system has been used for projects other than DP as well. Usage can start at any phase, for large, small and maintenance projects. The system has an interface to PC/70, an automated project control system from the same vendor. An enhanced "Version 2" has been recently released. The complete system sells for $30,000; an abbreviated version lists for about $10,000 less. The vendor is Atlantic Software Inc., Lafayette Building No. 910, 5th & Chestnut St., Philadelphia, PA 19106, (215) 922-7500.

- **SPECTRUM** SPECTRUM is one of several commercially available, completely manual system development methodologies. The SPECTRUM methodology is described in a set of from 18 to 25 "books," some of which are specialized to specific tasks, so that people performing these tasks (programmer, analyst, supervisor and project manager) need not consult the entire set of books. Three separate life cycles are defined for (a) major projects, (b) small projects and (c) maintenance. Procedures are also defined to control purchased software and to assure that maintenance activities are reflected in the documentation. Two versions are currently offered: the full package, SPECTRUM-1, costs $42,000; a reduced version, SPECTRUM-2, costs about $10,000 less. The supplier is Spectrum International Inc., 32107 West Lindero Rd., Westlake Village, CA 91361, (213) 991-5350.

- **STRUCTURED ANALYSIS** A generic name for a group of top-down, hierarchical design methodologies, including those developed by Larry Constantine, Edward Yourdon, Tom DeMarco, Chris Gane and Trish Sarson, Victor Weinberg, Douglas Ross, Jean Dominique Warnier and Kenneth Orr.

- **STRUCTURED PROGRAMMING** A program coding discipline that, in its purest form, admits only three types of program constructs: sequence, iteration and selection (also called choice or decision). A mathematical proof that any program that can be represented by a conventional flow-chart can also be constructed exclusively from these three elements, was published in 1966 by Corrado Bohm and Giuseppe Jacopini (so-called Jacopini-Bohm or Structure Theorem). Edsger W. Dijkstra of Holland and Harlan Mills and R.W. Floyd of the U.S. have also made major, early contributions to the subject. Structured programs require no GO-TO statements; the technique was therefore initially known as "GO-TO-less programming". Limiting the branching logic to just two standard forms (iteration and selection) results in programs that can be read and understood strictly from the top down, without the "jumping around" that characterizes conventional, unrestricted-branching programs. Structured programs are easier to maintain and modify. Because of its top-down property, structured programming is especially suitable for implementing top-down designs. Structured programming is now often combined with other techniques, such as hierarchical design, chief programmer team and structured walk-throughs, to form various program design methodologies.

- **STRUCTURED TABLEAU** A structured system design method developed by Donald R. Franz of Southwestern Bell (St. Louis). The method is based on the use of classical decision tables to describe decisions and the corresponding actions in a succinct, unambiguous way. Franz augmented the classical decision table by adding the required structured design constructs for iteration and sequence, and called the resulting artifice "Structured Tableau" to distinguish it from "plain" decision tables. The method offers a simple, quick way to assure design completeness and to estimate resulting program complexity.

- **STRUCTURED WALK-THROUGHS** Structured Walk-Throughs are a programming design review technique introduced by IBM as part of the Improved Programming Technologies package. Assumed to apply to Chief Programmer Team organizations, it can also be used with conventional programming organizations. The purpose of the technique is to eliminate the negative aspects normally associated with reviews. The developer (coder or designer) is responsible for calling the review, selecting the reviewers, setting the objectives and distributing copies of the work to be reviewed before the meeting. Four to six people and one to two hours are typical of good walk-throughs. The developer "walks" the reviewers through the work by explaining the work in a step-by-step manner, which simulates the actual execution of the program or function. This process brings to light errors in logic and coding. There is no relation between structured walk-throughs and structured programming, although the use of the latter eases the task of making sure that the walk-through has been complete.

- **SYSTEM DEVELOPMENT METHODOLOGY (SDM)** SDM is a set of ground rules for the development of systems, generally containing at least several, if not all, of the following elements: project control mechanism; guideline for system analysis and design (usually based on structured techniques); a well-defined, multiphase project life cycle definition; definition of deliverables at the conclusion of each phase and sign-off procedure, and procedures for iterating (implementing changes); documentation standards; estimation techniques; test plan; and tutorials (written or computerized explanations of how to use the system). Some SDMs are entirely manual; i.e., they consist of written descriptions of procedures; for example, SDM/70, CARA and SPECTRUM. Other SDMs may be computerized to varying degrees; for example, PRIDE-Logik/ASDM with ADF, SYSTEMACS.

- **TAPS** Terminal Applications Processing System. See TTP.

- **TOP-DOWN DESIGN** Top-down describes a design discipline in which the design process is carried from the highest level of generality (abstraction) through successively more detailed levels, until the desired degree of detail (e.g., functions directly implementable in a given programming language) is reached. Each general function or task identified at the highest level is successively expanded into progressively more detailed or elementary functions. Functional decomposition is another name for top-down design methodology. In programming (coding), top-down implementation calls for coding the most general control routines first, leaving the implementation of "stubs" (calls to as-yet-uncoded subroutines) for later levels. In this way all calling sequences are defined before any of the called subroutines are coded, eliminating a major source of programming errors. The term "top-down" is also used to loosely describe a property of structured code: it is sequentially readable, from the top down, without the need to "jump around" as in more conventional coding.

- **TSO - TIME SHARING OPTION** TSO is a system for supporting multiple interactive terminals engaged in program development and testing activities. Originally available under IBM's MVT operating system, it is now found mostly in MVS systems. To the operating system, TSO appears to be just another user; but TSO acts as a telecommunications monitor and supervisor for its terminals, allocating system resources among these terminals. TSO includes text-editing facilities for the creation and maintenance of source code, and an interface to the language compilers to permit dynamic debugging.

- **TTP - TAPS TRANSACTION PROCESSOR** A software package intended to ease the task of developing transaction-oriented application programs. The package includes a communications monitor similar to CICS or IMS/DC (the latter two can be substituted for the native CM). Like CICS and IMS/DC, TTP also has data base management facilities, transaction processing, and message handling/CRT screen support. TTP also incorporates menu-driven programming features, like those of DMS. Unlike CICS or IMS, which run only on IBM mainframes, TTP runs on both IBM machines (under OS and DOS) and on a number of minicomputers, including Interdata, DEC, HP, Prime and Harris. An interface to Intel/MRI System 2000 DBMS was announced in July, 1980. It is marketed by Decision Strategy Corp., 708 Third Ave., New York, NY 10017, (212) 687-6548 (recently acquired by Informatics, Inc.). Pricing ranges from $25,000 for a typical minicomputer, to $30,000 under IBM DOS and $40,000 under IBM OS.

- **WARNIER-ORR** A system and program design method based on concepts developed by Jean-Dominique Warnier in France and Kenneth Orr in the U.S. The system emphasizes structured <u>data</u> design as the starting point. Data structures are expressed in the Warnier Data Diagram. From the data structure, using time sequence analysis and change analysis, W/O derives the procedural program design almost as a one-to-one translation. The control structures for the program are first translated into pseudo-code, and then into actual code. A computerized system for creating Warnier-Orr diagrams, called STRUCTURES(S), as well as seminars on the W/O methodology, are offered by Langston, Kitch and Associates, 715 East 8th St., Topeka, Kansas 66607.

- **YOURDON/DEMARCO/CONSTANTINE METHOD** This structured analysis methodology is based on ideas first proposed by Larry A. Constantine, W.P. Stevens and Glenford J. Myers (all of IBM), and later refined and developed cooperatively by Tom DeMarco and Edward Yourdon. A Data Flow Diagram or Bubble Chart is used to discover and describe the changes that each input must undergo in order to produce a desired output. A Structure Chart is then developed to show the hierarchy of the changes or actions. The method can therefore be classified as data-driven. The Structure Chart is translated into pseudo-code, which can then be turned into executable code. Key concepts in the method include cohesion, coupling and span of control and effect; they are used to judge the modularity of the resulting design, its correctness and completeness.

APPENDIX B: PROFILE OF INTERVIEWED ORGANIZATIONS

APPENDIX B: PROFILE OF INTERVIEWED ORGANIZATIONS

- Exhibits B-1, B-2 and B-3 in this appendix summarize:
  - The distribution of interviewed organizations by industry.
  - The viewpoint (companywide, division or department) adopted by the respondents and their positions in their organizations as reflected by the number of management levels they supervise.
  - The type of the main processor(s) used by the reporting organizations.

EXHIBIT B-1

INTERVIEW DISTRIBUTION BY INDUSTRY

INDUSTRY SECTOR	NUMBER OF RESPONDENTS
DISCRETE MANUFACTURING	6
PROCESS MANUFACTURING	4
TRANSPORTATION	1
MEDICAL	1
SERVICES	4
UTILITIES	4
RETAIL	2
BANKING	2
INSURANCE	2
STATE GOVERNMENT	2
EDUCATION	2
TOTAL	30

EXHIBIT B-2

RESPONDENTS' VIEWPOINT AND POSITION

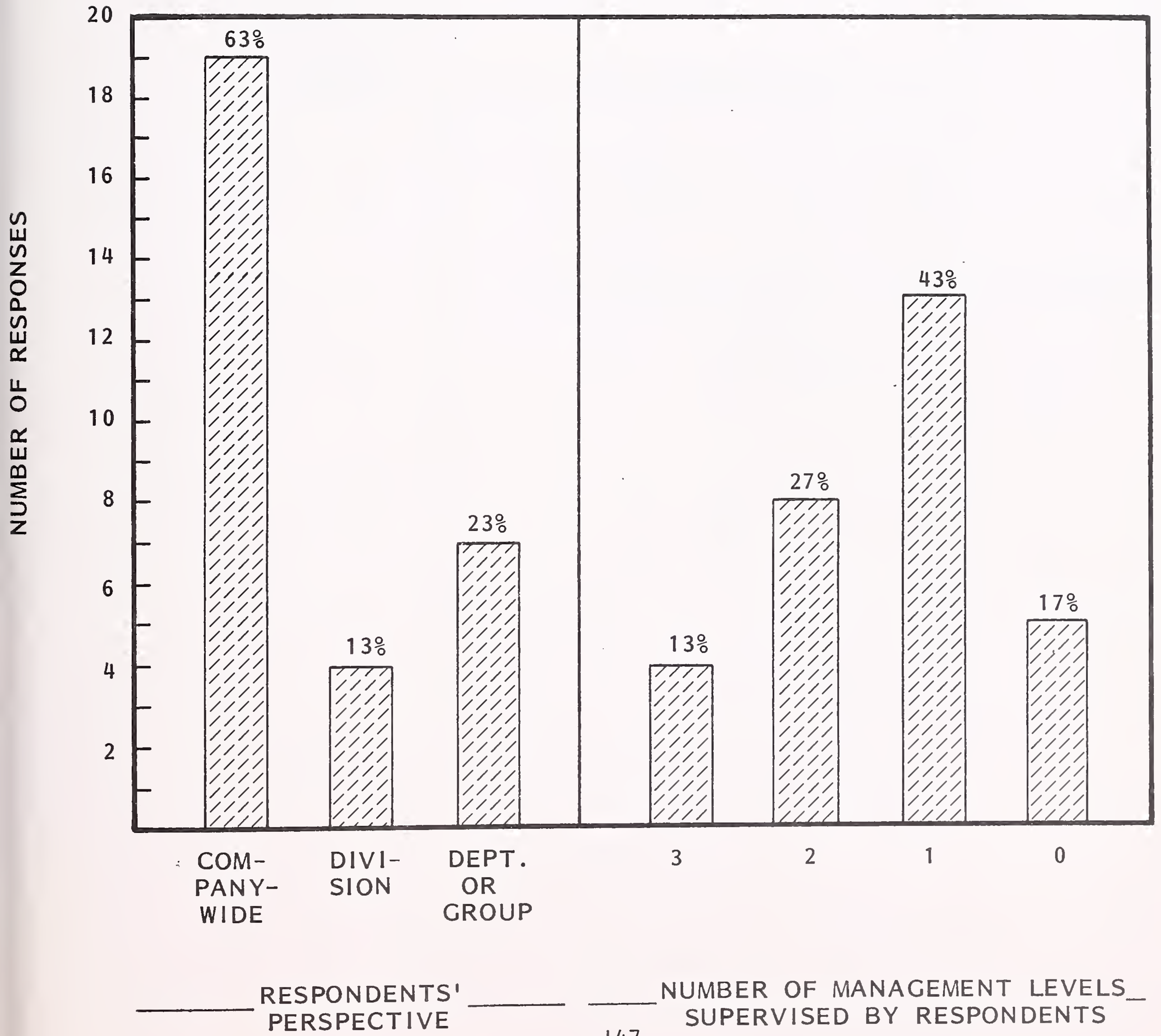


EXHIBIT B-3

MAIN PROCESSOR TYPE(S) USED BY INTERVIEWED ORGANIZATIONS

PROCESSOR TYPE	NUMBER OF TIMES THIS TYPE MENTIONED BY RESPONDENTS
IBM SYSTEM/370 MODEL 155	1
IBM SYSTEM/370 MODEL 158	12
IBM SYSTEM/370 MODEL 168	4
IBM SYSTEM/3031	6
IBM SYSTEM/3032	2
IBM SYSTEM/3033	8
IBM SYSTEM/4331	1
IBM SYSTEM/4341	3
AMDAHL 470V/6	1
AMDAHL 470V/8	1
HONEYWELL 6250	1

TOTAL RESPONSES: 30

APPENDIX C: QUESTIONNAIRE

CATALOG NO. USDP□□□

MANAGING THE SYSTEMS DEVELOPMENT PROCESS

I. GENERAL

1. Please define the perspective you will use in answering the questions in this interview.

   _____ Company-wide

   _____ Division (name) ________________________.

   _____ Department or Group ________________________.

   _____ Other (describe) ________________________.

2. How many levels of management are under your supervision?

   _____

3. What type of mainframe(s) do you use? (list quantity, manufacturer and model number(s))

   ____

II. EDP MANAGEMENT ISSUES

4. How effective overall is the management of the EDP systems development process?

   _____ Above average

   _____ Average

   _____ Below Average

   a. If above or below average, explain:

   ____

   ____

5. Would you characterize your EDP organization as:

_____ Functionally-divided

_____ Project-oriented

_____ Matrix (both functional and project)

_____ Other (describe) ________________________________

6. How happy are you with this organizational structure?

_____ Very happy

_____ Some reservations

_____ Change is highly desirable, because:

____

____

____

7. Is your company using:

_____ Chief Programmer Teams

_____ Structured Walk Throughs

_____ Other specialized development organizational structures.

Explain: ____________________________________

____

8. Are end users permitted to solve their own EDP problems?

_____ Encouraged

_____ Occasionally allowed

_____ Discouraged

9. What are the three most important things management should do to improve EDP systems development?

____

____

____

____

____

____

____

10. Are these things being done?

_____ Yes

_____ No., Why? _______________________________

____

____

11. Are the following practices used in your EDP organization?

_____ Flextime

_____ Working at home

_____ Other: _______________________________

____

____

12. Do you feel the turnover rate of EDP personnel is (for your geographical area):

_____ Above average

_____ Average

_____ Below average

a. If above or below average, explain:

____

____

____

13. Please rate the following factors as to their success in motivating EDP personnel to higher productivity.
(on a 1-5 scale 1=not successful, 5=highly successful)

_____ Money

_____ Congenial physical working environment

_____ Recognition by peers/management

_____ Company identification (loyalty)

_____ Other (describe) ___________________________________

____

14. Is the effectiveness of your company's human resources management:

_____ Above average

_____ Average

_____ Below average

a. If above or below average, explain: _____________

____

III. LIFE CYCLE, SYSTEMS, PROJECTS

15. On the average, how long will a major system be in routine operation? (including development and maintenance)

    _____ years

16. Please estimate the relative level of effort for development versus maintenance in major projects:

    _____% Development

    _____% Maintenance

    100 %

    a. Is this ratio satisfactory?

       _____ Yes

       _____ No. Why? ______________________________
       ____
       ____

17. What percentage of the development phase is devoted to the following areas? (including documentation in each area.)

    _____ % Specifications and requirements

    _____ % Detailed Design

    _____ % Coding

    _____ % Testing

    _____ % Other (describe) ______________________________

    100 %

18. What percentage of the maintenance phase is devoted to:

_____ % Fixing bugs

_____ % Enhancements

_____ % Other (describe) ____________________

100 % ____________________

19. Who is responsible for software Quality Assurance and Control (verifying that new systems meet specs and are bug free?)

_____ The developers

_____ A separate QA/QC department

_____ Other, explain: ____________________

____

____

20. Is this QA/QC method satisfactory?

_____ Yes

_____ No. Explain: ____________________

____

____

21. Who is responsible for software systems maintenance?

_____ The developers

_____ A special maintenance group

_____ Other. Describe: ____________________

22. Please define project size:

	Total Duration	# People Involved
Large	____________	______________
Medium	____________	______________
Small	____________	______________

23. Do small projects require less development effort in relation to the total life cycle?

_____ Yes

_____ No

a. If yes, please estimate the percentage of development effort relative to total life cycle for the following project size categories.

Large _____ %

Medium _____ %

Small _____ %

IV. PROJECT CONTROL AND MEASUREMENT

24. Are you using an automated project control system?

_____ Nichols

_____ PAC II, III

_____ PC/70

_____ Other. Name ______________________________

_____ Do not use. (Go to Question 27)

25. Which of the following are included in the project control system:

_____ All staff (if not all, who? ______________________)

_____ All management (if not all, who? ______________________)

_____ All hardware resources (if not all, which? ____________)

_____ Other. Describe ______________________________

26.a. What are the three things you like best about your current project control system?

____

____

____

b. What are the three things you like least?

____

____

____

27. Are any of the following being measured?

_____ Lines of code per day/month, etc. (which?)

_____ Number of errors discovered per l.o.c.,/per month/etc. (which?)

_____ Hours of development time (or cost) per function delivered.

_____ Other output measurements. Describe: ______________

____

V. TOOLS AND AIDS

28. Which structured analysis methodologies are used in your organization?

_____ Constantine/Yourdon/DeMarco

_____ Gane & Sarson

_____ Jackson

_____ SADT

_____ Warnier-Orr

_____ Other. Name ______________________________

_____ None

a. Benefits associated with the use of structured techniques: ______________________________

____

b. Problems associated with the use of structured techniques: ______________________________

____

29. Is your company using a formalized System Design Methodology?

_____ Pride/Logik

_____ SPECTRUM

_____ CARA

_____ SYSTEMACS

_____ SDM/70

_____ Other. Name and manufacturer: ______________________________

_____ Home brewed, based on ______________________________

_____ None

a. Benefits associated with the use of this SDM: ________

____

b. Problems associated with the use of this SDM: ________

____

30. Are you using any of the following tools or aids?

_____ Requirements Language (PSL/PSA)

_____ Pseudo Code System (PDL)

_____ Reusable Code System (Raytheon, Kapur)

_____ HIPO Charts

_____ Code Generators. Describe: ______________________

_____ Prototypers or discardable code. Describe: __________

____

31. a. Do you have a data base management system?

_____ Yes. Name: ______________________________

_____ No.

b. Do you have an interactive inquiry/retrieval language or system for the data base?

_____ Yes. Name: ______________________________

_____ No.

c. List benefits/problems associated with DBMS/inquiry language.

____

____

____

____

CATALOG NO. U S D P

VI. ON-LINE DEVELOPMENT MANAGEMENT

32. Is there a CPU dedicated entirely or partially to program development activities?

_____ Yes (CPU model)

_____ No

_____ Partially. Explain: ______________________________

____

____

33. Is the process of program development in your company

_____ Interactive

_____ Batch

_____ Combination (estimate percentages)

34. Which on-line development tools are used?

_____ CMS

_____ TSO

_____ Programmer's Work Bench/UNIX

_____ ADF

_____ DMS

_____ CICS

_____ IMS/DC

_____ Other. Describe: ______________________________

____

THANK YOU FOR YOUR COOPERATION.