A survey of structured programming practice

by I. ST. J. HUGO

Infotech International
Berkshire, England

ABSTRACT

This paper refers to the results of our survey of over 300 companies worldwide to determine the extent and type of their involvement in structured programming.

INTRODUCTION

Reason for survey

In November 1974 and March 1975, Infotech staged a total of two conferences and one tutorial on structured programming. Many of the speakers were well-known, e.g., E. W. Dijkstra, C. A. R. Hoare, D. Parnas, F. T. Baker, and M. Jackson, but nevertheless the total attendance of some 800 people from 19 different countries exceeded expectations. Because of this high degree of interest in the subject, a program of training courses was launched in mid-1975.

At the beginning of 1976, it was decided to survey the use of structured programming techniques with the following goals in mind.

1. To assess how well the program of training courses fitted the current needs and plans of software development groups.
2. To extend Infotech's own information on the subject.
3. To develop a useful, saleable product.

A note on terminology

The term "structured programming" has acquired many meanings that often lead to confusion. Unless otherwise stated, the term is used in this paper in its most general, but not abstract, sense. That is, it denotes a range of design, coding, documentation and management techniques that use structure in some way to impose greater clarity and control on the software development process. These techniques have become popularly known collectively as structured programming and many are often wrongly attributed to Professor E. W. Dijkstra. Where appropriate, specific techniques are mentioned by name.

The published survey

This paper is concerned with the analysis of the completed questionnaires returned by computer users, on the follow-up to these questionnaires (see Survey Method below) and on the case studies obtained for the published Survey and Report. The complete published work is based additionally on an analysis of 690 documents, papers and reports in the existing literature. The full Survey and Report consists of the following five volumes:

Survey
Volume 1: International Overview
Volume 2: Survey and Analysis of User Experience
Volume 3: Methodologies and Techniques

Report
Volume 1: Guide to Techniques and Implementation
Volume 2: User Experience in Structured Programming

SURVEY METHOD

The survey project was publicised in advertisements and mailing shots, computer users being invited to provide brief data on the extent of their organization's involvement in structured programming and to request a detailed questionnaire. The motivation for participation in the survey was that each organization completing and returning a detailed
questionnaire would receive a copy of the analysis of the questionnaire returns. A total of 1080 organizations provided brief data and requested detailed questionnaires. The request form and detailed questionnaire are shown in Appendix 1.

The total of 1080 detailed questionnaires requested was reduced to 309 usefully completed and returned, which form the basis of the questionnaire survey, by the following occurrences:

1. Some of the questionnaires requested were not returned and no explanation was given.
2. Many of the organizations who requested questionnaires were in fact planning to use structured programming techniques rather than actually using them and so were unable to complete the questionnaire.
3. Miscellaneous reasons that rendered returns irrelevant to the purposes of the survey; e.g., inconsistencies in the data, jokers, etc.

On the basis of the adequately completed questionnaires, 54 organizations were selected and contacted by telephone for clarification and expansion of the reported data. A further 22 organizations were visited by Infotech staff in person, again to achieve greater understanding of the use being made of structured programming and of the results obtained. Finally, 13 specific case studies were commissioned and rights to a further existing 19 case studies acquired.

SURVEY RESULTS

Profile of responders

The questions in section A of the questionnaire (see Appendix 1) were designed to provide a profile of the responders. Figure 1 gives the breakdown of responding organizations by size of software department.

The geographical spread of responders was very wide, although few responses were obtained from most countries. Approximately one third of the responding organizations were in the USA, a quarter in the UK and a further quarter from 15 countries in continental Europe. The remaining responses were from Australia, Brazil, Canada, Japan, the Middle East and South Africa.

Twenty-seven percent of the organizations placed themselves in the software house category. However, slightly over half of these were software departments within user companies that had been established as a service division independent of other functional divisions within their organization. Allowing for this, 87 percent of responders were from organizations whose primary business was not data processing.

This figure is surprising in that organizations whose primary business is not data processing account for only about half of the usual participants in Infotech activities. True, responses to advertisement could have produced a slight bias outside Infotech's normal profile of participants but hardly a bias of that size. However, it is apparent, in the UK at least, that the primary interest in the use of structured programming lies with the computer users outside the DP industry. The large majority of software suppliers show more interest in selling aids, courses, etc., than in using the techniques themselves.

Given this, the percentages of effort devoted to scientific/technical, commercial and system software are very much what might be expected (see Figure 2). So too were the languages used; over 50 percent of responders used COBOL. Approximately 25 percent used Assembler or PL/1 and rather less, about one sixth, used FORTRAN.

Interestingly, about one sixth also reported using some other language, which is rather higher than expected, given
the commercial orientation of the responders. True, ALGOL, BASIC and RPG accounted for a number of these but comments on the unsuitability of the major current languages for structured programming were frequently made and there was significant evidence of a search being made for more suitable languages. No estimate of the amount of use of each language was asked for; had it been, it is likely that these other languages would have paled into insignificance. However, it would appear that the use of structured programming techniques is creating a greater awareness among commercial programmers of how a language shapes and constrains one’s approach to problems.

Regarding the mix of maintenance and new development work, two thirds of responders reported maintenance work involving between 20 and 60 percent of their effort and new development work between 40 and 80 percent. The median was 71-80 percent new work, 20-29 percent maintenance, with the curve heavily skewed towards new work. These figures bode well for the adoption of new techniques and suggest a limited demand for structuring cosmetics for existing software.

The processors in use at the installations surveyed are of no particular relevance. The purpose of the question was simply to check on the sample obtained. As expected, IBM machines predominate and a wide spectrum of other suppliers’ equipment was also represented in the sample.

INTRODUCTION TO STRUCTURED PROGRAMMING

Section B of the questionnaire sought to establish the means by which structured programming came to the attention of users and how implementation of the techniques was started. As might be expected, nearly all responders reported first learning of structured programming through the general technical press or at conferences. Most then sent one or two people on courses before using one or more of the techniques on a project.

Within the survey sample, there was not a single organization that had adopted a set of techniques as standard, to be used by existing staff and to form part of the training program for new staff. A few, but only a few, organizations were planning toward this goal and had made measurable progress. Given the general familiarity with structured programming, the practice of it is perhaps surprisingly spasmodic. The general impression gained is of apostles spreading the gospel where they can but generally with little co-ordinated support from software managers. Many reasons can be advanced for this but they will not be discussed here.

Regarding specific techniques used first, the largest group of responders used structured design and structured coding in combination. The next largest group described themselves as using structured coding and top-down development in combination. Smaller groups had used structured design or structured coding alone. Overall, across all combinations of techniques, structured design, structured coding and top-down implementation were most frequently cited as techniques used in the first step and HIPO and structured analysis were the least frequently cited.

EFFECTS OF STRUCTURED PROGRAMMING

Sections C and D sought to establish the noted effects of using structured programming techniques and the length of experience on which the observations were based. The large majority of responders had two years or less experience of structured programming, most of those having between one and two years experience. Three organizations claimed 13 or 14 years experience.

The principal effects reported from the use of the individual techniques listed in section D of the questionnaire are given below.

STRUCTURED ANALYSIS

The most noted effects of the use of structured analysis were a decrease in the number of errors made and a decrease in the number of man-hours spent debugging. The next most significant gains reported were, in order of decreasing importance, greater control of project progress, greater motivation/job satisfaction of DP staff and greater user satisfaction. It was perhaps also relevant that very nearly equal numbers of responders reported (a) a slight increase (b) no change and (c) a slight decrease in both the elapsed time of projects and the total number of man-hours of effort expended.

STRUCTURED DESIGN

The results for structured design were almost exactly the same as for structured analysis, except that an increase in technical coordination replaced increased user satisfaction as a major gain.

It will be seen from the questionnaire that responders were asked to state the type of structured analysis or design. It was obvious from the responses that there was great difficulty in differentiating between analysis and design and that, when unable to differentiate adequately between the two, most responders opted to describe the overall process as design. This being the case, it may be that the noted effects of structured analysis and structured design should be considered together rather than independently.

STRUCTURED CODING

The results for structured coding were remarkably consistent. It was claimed that benefits were noted right across the board and there were virtually no dissenters from this glowing picture. The greatest gains were observed in the reduction of debugging man-hours and of computer time for testing. Only slightly lower gains were claimed in relation to
project elapsed time, project man-months, errors made and maintenance man-hours.

TOP-DOWN IMPLEMENTATION AND TESTING

One benefit stood out above all others for top-down implementation and testing and that was the reduction noted in the number of man-hours spent on debugging. Significant claims of benefits were also made in relation to the number of errors made and the productivity of DP staff. Once again there was no significant contention in the results noted.

TEAM OPERATIONS

The noted effects of team operations were somewhat difficult to interpret. For instance, gains were consistently reported in the productivity of DP staff whilst, at the same time, there was no consistency in the observed effects on project man-months and project elapsed time. There was, however, unanimity in reporting greater motivation/job satisfaction of DP staff, so it is possible that the productivity gains were more imagined than tangible.

In general, the noted effects of team operations were much less certain than was the case with most of the other techniques. The apparent misgivings regarding team operations showed up strongly in the follow-up to the questionnaire, particularly in relation to the chief programmer team structure.

PROJECT LIBRARY OPERATIONS

Project library operations was one of the least used of the techniques listed and the benefits noted were generally rather muted. However, there was little inconsistency in the results, the greatest gains being claimed, as might be expected, in the areas of technical co-ordination, control of project progress and the number of errors made.

STRUCTURED WALKTHROUGHS

The most consistently noted effect of structured walkthroughs was, the expected reduction in the number of errors made and of the man-hours spent de-bugging. Rather surprisingly, however, the next most consistently reported gain was an increase in the motivation/job satisfaction of DP staff. The fourth most consistent gain was in the area of technical coordination.

Follow-up to the survey, and section G of the survey itself, revealed a considerable amount of disagreement in user's experience of structured walkthroughs. Particularly noticeable was a tendency for opinion to polarize either very strongly in favor of structured walkthroughs or very strongly against them.

Contention reveals itself in this section of the survey in that, despite the consistently reported gains in relation to errors made and debugging man-hours, there was no agreement on the effect of structured walk-throughs on project elapsed time and project man-months. As many responders reported these times increased as reported then decreased.

HIPO

The sample of responders with experience of HIPO was relatively small (44 in total). The most consistently reported benefit was a reduction in the number of errors made, followed by gains in user satisfaction and technical co-ordination. Once again, there was considerable disagreement over the effect on project elapsed time and project man-hours.

PROJECT MANAGEMENT SYSTEM

This title was devised to allow experience on any of the several formalized project management systems around to be recorded. Two results were noteworthy. Despite the fact that gains were consistently reported under the control and management headings, there was no consistency in the noted effects on project elapsed time and project man-months. Also, very few responders claimed any benefits other than under the control and management headings.

FACTS AND FIGURES

Section E of the questionnaire was an attempt to gather some hard data. Such data is notoriously difficult to find and, even when available, often cannot easily be interpreted because of the complex of factors affecting it. For this last reason, no attempt was made to try to obtain data with a view to producing direct, quantified comparisons between the results obtained in different organizations.

Figures quoted for the productivity of programmers writing COBOL programs illustrate this point very well. A number of responders quoted the productivity achieved, using various combinations of structured programming techniques, in terms of COBOL statements per man-day over the duration of a project from design through to acceptance testing. The figures quoted are as follows: 26, 28, 30, 36, 40, 48, 56, 60, 65, 75, 100 and 150! Perhaps the only significance of these figures is that each represented, for a different organization, a considerable jump in productivity. Only five responders gave the productivity achieved prior to using structured programming and all five quoted figures in the range of 10-12 COBOL statements per man-day.

Most responders had no data to offer. A number had collected data but had introduced structured programming on projects involving a new type of work and often new hardware as well; the effects due to the new software production methods were therefore uncertain.

Fifty-six responders provided facts and figures of one sort or another. The aspects of the software development
process for which figures were most frequently given were elapsed time, design time, testing time, maintenance effort, productivity (lines of code per day) and documentation. Before these are discussed, it should perhaps be pointed out that the significance of improvements depends to some extent on the state of affairs before the introduction of the new methods. In many cases, the introduction of structured programming would appear to be the first time that what could properly be termed a method was used at all.

PROJECT ELAPSED TIME

Few responders reported gains in elapsed time. Those that did had achieved gains of between 10 and 20 percent. The most striking result was the number of reports of project time estimates being consistently met. One is so used to hearing of projects being behind schedule that one does not stop to think of the general state of depression in which most software project managers live. A single quote sums up the normal situation: "The certain knowledge of failure is the only thing that keeps you sane." The relief with which it was reported that time estimates were consistently being met was most noticeable and appeared to provide project managers with a general confidence in their work that they had never had before.

DESIGN TIME

Design time was consistently reported to have been increased by the introduction of structured programming. Increases in design time reported were between 50 and 200 percent with most responders reporting an increase of 100 percent or more. There was evidence of a more thorough approach to design, which may have accounted for some of the higher figures reported: "We felt free to examine two or three alternatives instead of grasping at the first working solution."

DEBUGGING, TESTING AND MAINTENANCE

Large gains in debugging man-hours, computer time for testing and the time and number of people devoted to maintenance were generally reported. Overall, the reductions were in the range of 50 to 80 percent and comments such as "at no time was it necessary to alter any processing sequence," were common. The example below is fairly typical.

**Techniques used:**

Structured design and coding, top-down implementation and testing, team operations, project library, walkthroughs, HIPO

**Type of work:**

Commercial programming in PL/1

<table>
<thead>
<tr>
<th>Type</th>
<th>Pre-Results</th>
<th>4/75</th>
<th>4/76</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average lines of code per man-day</td>
<td>13</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>% staff on maintenance</td>
<td>60</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Total m/c usage on testing</td>
<td>-15%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Errors per program not identified until acceptance testing or final systems testing</td>
<td>-65%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DOCUMENTATION**

Time for documentation was generally reported to be increased, often with the corollary that it hadn't been done either properly or at all before. "Our gain in code/test throughput has somewhat been used up by doing tasks which were not done before, such as user documentation." Where percentage increases were quoted, 50 percent was a typical figure.

**MAJOR BENEFITS AND PROBLEMS**

Section F of the questionnaire sought an overall assessment of the major benefits and problems experienced in relation to structured programming. Broadly, the major benefits most consistently reported were the quality and maintainability of the product, the productivity and job satisfaction of the DP staff and the greater visibility and control in the software development process. The most consistently reported problems were in gaining acceptance and backing for structured programming, both by higher management and by senior programmers, the time and cost involved in retraining staff, the difficulty in producing programming standards, especially in relation to current languages, and in enforcing them and the cost of documentation. Some interesting points emerged.

The most evident benefit can be described as greater visibility and control; this was shown in many ways. Several responders said that their program specifications were much clearer than before and that staff now had the confidence to reject specifications that were inadequate. Many also said that modifications to large programs were no longer dreaded and that programs could be passed between staff easily. Many claimed great improvements in documentation, although many also reported the time and cost involved in producing the documentation as a problem, which suggests that documentation is still often regarded as an adjunct to rather than an integral part of software. Overall, the comments displayed a new confidence that understanding and control of the software development process had been achieved. This had an important side-effect. Increased user satisfaction was widely reported and several responders said that user departments were now coming forward and asking for more jobs to be pro-
grammed, rather than seeking to avoid involvement with DP.

An interesting contrast were the claims made for greater motivation and job satisfaction and the many comments that gaining acceptance of structured programming was a problem. Gaining acceptance was, in fact, the major problem reported. Lack of acceptance and support by senior management appears to be an unresolved problem and is one that could seriously impede the progress of structured programming. Lack of acceptance by senior programming staff appears in many cases to have been a temporary problem, a number of different methods being used to win these staff over. A few responders issued a caution against over-optimism, pointing out that the benefits reported had been hard won and involved a great deal of discipline and monitoring.

A number of responders reported confusion caused by the number of different versions of structured programming and had difficulty in assessing whether the differences between them were real or imaginary; this resulted in difficulty in deciding in detail which course to follow.

An interesting difficulty is that programmer team structures apparently are rendered impossible by some of the rigidly enforced personnel hierarchies that are encountered in some public bodies. There was apparently no way round this problem.

After lack of acceptance, the major problem reported was the cost of retraining DP staff. Several responders pointed out that having just a few staff trained in structured programming could give you the worst of both worlds but that the cost of retraining everybody in a short space of time was unacceptable; moreover, the time involved made such a course impractical. Some large organizations have got round this problem by introducing structured programming to relatively self-contained subsets of the DP staff one at a time. Another problem reported in this area was the lack of training courses in structured programming for junior trainees.

RECOMMENDED IMPLEMENTATION SEQUENCE

There was no agreement on the sequence in which the various structured programming techniques should be introduced except that most responders recommended that structured coding and/or structured analysis and design should be introduced first. This, as it happens, was what most of the responders had done so this result is perhaps to be expected.

HINDSIGHT

Section G of the questionnaire sought to gain the benefit of hindsight. Most responders had approached structured programming in a rather ad hoc manner and were very conscious of the fact. Therefore, overall, the principal recommendation was to adopt a planned approach and to stick to it. This general recommendation broke down into:

the following list of most consistently mentioned points:

1. Get advice from an outside consultant or an experienced user
2. Appoint an internal consultant
3. Do not introduce structured programming in conjunction with other major changes
4. Select a pilot project that is not too difficult
5. Use a team on the pilot project and then use that team to organize and control the training of others
6. Use your most competent personnel for the pilot project
7. Monitor the pilot project closely and collect data on it
8. Get management commitment
9. Structured programming must be sold to staff
10. Don’t oversell
11. Anticipate acceptance problems
12. Apply more time and money to training
13. Reduce training elapsed time
14. Have follow-up courses
15. Apply more effort on standards

One or two conflicts emerged. There were differences of opinion on the amount of selling of structured programming that should be done within the organization and also on the degree to which adherence to the new techniques should be enforced. There was strong disagreement on the number of techniques that should be introduced initially, responders polarizing into the all-at-once camp or the little-at-a-time camp. There was also strong disagreement on the usefulness or otherwise of programmer teams and walkthroughs.

Some of the recommendations were of the more-in-hope-than-anger variety. Getting management commitment, for instance, was recommended by very many responders but had been achieved by very few. Similarly, many of the responders who recommended more training and more concentrated periods of it recognized the impracticality of their recommendations in many situations.

SUMMARY AND CONCLUSIONS

The results of the questionnaire survey are subject to a number of qualifications that should be made explicit. In the first place, no major failures were encountered in the survey sample. This is hardly surprising, in that organizations asked to volunteer information are rather more apt to do so about their successes. Clearly, it is likely that a number of failed attempts to implement structured programming do exist and, if the experience gained could be made public, it might be even more revealing than the experience of success.

Secondly, the organizations using structured programming are probably among the most experienced in producing software and hence more likely to succeed. Also, the very attempt to introduce new techniques suggests a commitment to improving their methods of software production (despite the lack of management backing).
Thirdly, it can only be guessed how much of the improvement reported is due to the techniques themselves and how much to the training and extra effort put into their introduction. Despite the generally reported need for more training and greater planning, some extra effort must have been involved in introducing structured programming.

Fourthly, many of the figures quoted were taken from pilot projects, which often show a higher degree of success than subsequent general implementation.

Fifthly, the "specific" techniques listed might more aptly be described as somewhat broad categories of techniques. It is clear that a certain amount of confusion exists about the nature of differences between different versions of the true gospel. No attempt was made to be precise in the definition of terms because it was felt that any such attempt would be bound to fail or else to reduce the surveyed population, probably to 1. However, the purpose of this survey was not to provide insights that might further the theoretical basis of programming but to seek the opinions of those practising programming under normal commercial constraints.

In spite of these qualifications, it is contended that the survey was a useful exercise and did produce helpful information. It is certain that many organizations are making significant gains, on a continuing basis, both in the quality of their software and in the cost of producing it, through the use of structured programming techniques. Above all, there emerges the clear impression that, at least for the large majority of software projects, the software development process has become visible and controllable. That is a significant step for the management of software production in the commercial field.

ACKNOWLEDGMENT

The survey described in this paper is the work of several people within Infotech, notably C. Watkins and D. Bates. The raw data was obtained with the kind co-operation of organizations throughout the world that are too numerous to mention individually but to whom thanks are due.
Structured Programming Survey

Free Copy of Results Analysis

Infotech is currently compiling a comprehensive practical Survey of experience in the use of structured programming and the related dp techniques.

The published Survey will contain an authoritative guide to the new techniques, checklists of do's and don'ts for implementation, case studies of implementation experience, survey of analysis, design and programming aids, a full literature survey and a detailed analysis of user experience in all the new techniques.

If you have been using any of these new techniques and would like to participate in the survey please complete and return the form below.

By filling in a simple questionnaire on your experience you qualify for a free copy of the results analysis.

Structured Programming Survey
Free Results Analysis Request

I would like to receive a free copy of the Results Analysis from the Infotech Structured Programming Survey
( I am willing to complete a questionnaire)

Name ........................................................................................................................................................................
Position ........................................................................................................................................................................
Organisation ..............................................................................................................................................................
Full Postal Address ......................................................................................................................................................
Telephone No ...........................................................................................................................................................

My organisation has experience in

☐ Structured/Top Down Implementation and Testing
☐ Structured Program Design
☐ Project/Program Library
☐ Structured Coding/Programming

(Language......................)

☐ Team Operations
☐ Structured Walkthroughs
☐ HIPO
☐ Automated Analysis/Design Aids

(Type....................................................)

Return this now to: Clive Wilkins, New Structured DP Techniques Division, Infotech International Limited,

APPENDIX I—Brief questionnaire
International Survey of Structured Programming Practice

A Your organisation and equipment

Is your organisation a software house or similar supplier of software and programming services?

- Yes
- No

Total number of systems and programming staff (including managers)

<table>
<thead>
<tr>
<th>Type of computer applications</th>
<th>Commercial (%)</th>
<th>Scientific/Technical (%)</th>
<th>System software (%)</th>
<th>Total 100%</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Nature of work</th>
<th>New system development (%)</th>
<th>Maintenance (%)</th>
<th>Total 100%</th>
</tr>
</thead>
</table>

Processor(s) in use: (manufacturer & type number)

- (core size)
- (operating system)

Major programming language used for development

APPENDIX I—Detailed questionnaire
C Length of experience
This section is designed to show the sequence in which you are using, or going to use, the techniques and the extent of your experience (see instructions below table)

D Noted effects of using the new techniques
This is the most difficult but most fruitful section for you to complete. We would like to know: technique by technique, the effects you have noted resulting from their use. Our analysis of this information will show whether your use of particular techniques has produced similar benefits and penalties to others.

Use this scale to show in the table below the effects of each technique you are using (see instructions below table)

| Much more/ much greater | 5 | More/ greater | 4 | Same as before | 3 | Less | 2 | Much less | 1 | Not noted | 0 |

<table>
<thead>
<tr>
<th>AREAS AFFECTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development</td>
</tr>
<tr>
<td>Production &amp; maintenance</td>
</tr>
<tr>
<td>DP staff</td>
</tr>
<tr>
<td>Users</td>
</tr>
<tr>
<td>Control &amp; management</td>
</tr>
<tr>
<td>Other effects noted</td>
</tr>
</tbody>
</table>

THE TECHNIQUES

1. Structured analysis (type )
2. Structured design (type )
3. Structured coding
4. Top down implementation & testing
5. Team operations
6. Project library operations
7. Structured walkthroughs
8. HIPO
9. Project management system

TECHNIQUES USED IN COMBINATION (circled techniques)

1 2 3 4 5 6 7 8 9

Example:
For a particular technique, e.g. Structured Design, a 5 entered in the first column indicates that one noted effect of using structured design is that total project elapsed time is much greater. Similarly a 3 entered in the second column indicates that the total man months needed to complete a project using structured design is the same as before.

APPENDIX I—Detailed questionnaire (continued)
E Facts and figures
In section D, we asked you to note the effects of using the new techniques. We are asking you here to provide us with any facts and figures you have that will help us to quantify the experience summary we produce.

What facts and figures do you have on your experience with the new techniques?

F Results of your experience
This section is provided to help you compare the problems you have met and the benefits you have gained with those of other organizations.

What are the major problems you have encountered so far and how have they been overcome?
E.g. lack of training, management commitment and staff acceptance, over-optimism, time, cost and control of change.

What would you say were the most important benefits to your organization of using the new techniques?
See 'effects' headings to columns in section D for possible benefit areas.

As a result of your experience so far, in what order would you recommend that the new techniques be implemented?
Use the technique numbers 1-9 as in sections C, D above to indicate recommended implementation sequence. You may group techniques at any stage.

APPENDIX I—Detailed questionnaire (continued)
### Implementation guidance

Four answers in this section will be summared in lists that we are compiling of recommended do's and don'ts for implementing and using the new techniques.

If you were able to start again knowing what you know now, what, if anything, would you do differently?

<table>
<thead>
<tr>
<th>What would you say were the most important do's and don't in implementing and using the new techniques?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look back through your answers to previous questions to compile your list.</td>
</tr>
</tbody>
</table>

### Over-all summary

How would you sum up your experience to date with the new techniques?

### What do you see as the most important developments to your organisation's computer activities in the next 5 years?

Include: system development methodologies and techniques, management & control methodologies, computer personnel productivity, user involvement & satisfaction, hardware developments.

---

Thank you for your help – please return this questionnaire NOW.

APPENDIX I—Detailed questionnaire (continued)