Peter Naur

What happens during program development – an experiment.

K.3.2

Contribution to Festskrift til Professor Börje Langefors,
'systemering 75"

WHAT HAPPENS DURING PROGRAM DEVELOPMENT

AN EXPERIMENT

Peter Naur
Copenhagen University

August 1974

Abstract

The paper describes an experiment in which 12 students solved 3 program development problems. During the process of development they recorded every step of their work in writing, as far as practically possible. In addition they described the goals they had pursued. The reports of the students are summarized and compared, and related to a set of 33 development activities. The main conclusion is that the experimental approach used is capable of yielding usable information. The influence on program development of the goals actually pursued and of programmers' personality differences appear as important issues that need further study. The dependence of the program development activity on the nature of the problem being solved emerges clearly. The relation to structured programming is obscured by uncertainties of concepts and terminology.

1. INTRODUCTION

As a contribution to the methodology of computer programming, this paper describes an attempt to find out what actually happens when programs are developed by people. It reports on an experiment involving 12 students at Copenhagen University. Briefly, the students were asked to solve three different programming problems, noting continuously what they were doing in the course of the program development. In addition they were asked to comment in various ways on their work. In the present paper these student reports are analyzed.

The emphasis in the experiment and in the present analysis of it is on the experimental method. Using this method an attempt is made to find out what activities were undertaken by the students

and how these differed among the students and from one problem to another. In addition the question of what goals were actually pursued by the students will be touched. The analysis of the characteristics and quality of the programs written by the students is beyond the present enquiry.

The study relates directly to a series of recent works centered around what is called structured programming. This has been described by Dahl, Dijkstra, and Hoare (1972), and a similar approach, programming by stepwise refinement, has been advocated by Wirth (1971). Subsequent works by Henderson and Snowdon (1972) and by Ledgard (1974) report on practical experience of applying structured programming. A common characteristic of these works is that they reflect the long personal experience of some of the leading thinkers in the field. What is lacking is evidence on the value of structured programming if applied by programmers of average background and ability. A partial explanation of this lack is the difficulty of obtaining such evidence. Generally what would seem to be needed is carefully designed and controlled experiments, involving selected groups of programmers working under suitably supervised conditions. Experiments along such lines, although not aiming at clarifying the issues considered here, have been described by Schatzoff, Tsao, and Wiig (1967), by Sackman, Erikson, and Grant (1968), and by Gold (1969). These studies demonstrate very clearly the difficulty of the approach. In fact, the effect that was the primary subject of the studies, viz. the influence on the programming efficiency from the difference between off-line and on-line access to the computer, could hardly be detected from the experiments. The dominating factor was found to be the individual differences of programmers.

In order to clarify the problems of programming methodology raised by the adherents of structured programming, experiments ai-

ming at detecting quantitative differences of performance depending on the methods used would certainly be highly desirable. However, the difficulty of obtaining convincing conclusions from this approach may be expected to be even greater than in the performance studies quoted above. Indeed, while either off-line or on-line access to the computer may be imposed on a programmer through purely administrative measures, the use of either structured programming or otherwise by a programmer lies beyond what can be controlled, or even detected, by an external agency. It would appear even to lie beyond what can be controlled by the programmer himself.

In view of these difficulties it appears desirable to attack the problem from a different angle. One such angle is the one taken here. In finding out what actually happens during program development we may hope to gain insight into how much of the programming activity might likely be influenced by, and gain from, a structured programming approach.

The experiment is based on written recordings of the programming activity, done by each student in the course of the work. An alternative technique would have been to make sound recordings of verbal accounts given by the students. The two recording techniques, by writing and by sound recording, each have obvious advantages and limitations. Sound recordings tend to produce very large quantities of material that are difficult to handle in the analysis. Recording by writing will be felt as a heavy burden by many programmers, and may be expected to exert a strong influence on the programming process itself.

The present study continues an earlier experiment by the author (Naur 1972) by involving a group of students rather than just one individual, and three separate programming tasks, rather than just one. It remains severely limited in several directions. The most

serious limitation would appear to be the lack of observations concerning large, and even just medium size, programs. As another serious limitation the question of the goals actually pursued by the students is treated inadequately.

2. UNDERLYING ATTITUDE

The choice of form of an experiment such as the one described here is the result of an attitude. This underlying attitude has at least the following components. First, the experiment is based on a respect for, and a belief in the value of, the effort of the participants, even if it is recognized that their talent, education, and other such background factors differ a great deal. Second, the experiment assumes that persons differ to such an extent that it is inappropriate to speak of one correct, or proper, or best, method of program development, that can be recommended to any programmer. Rather, it is admitted that a method that is effective for one programmer may not be so for another, even when they face the same problem. Third, it is assumed that the teaching of highly complex and personal matters such as programming is best done by exposing the students to methods and to examples of their use, while any compulsion to use specific methods should be avoided.

3. THE PARTICIPATING STUDENTS

The students participating in the experiment were those who chose to take the course. Since the course was only one out of a number of elective courses, it is clear that the one common characteristic of the participants is their interest in the subject as described in the course announcement (see section 4 below). Otherwise the participants form a somewhat mixed assembly. The number of students actively starting the course was 12, a comparatively

high number as our elective courses go. Of these only 8 worked through the course to the end. The ages of the students range from 21 to 26 years. The educational background ranged from 2 to 6 years of participation in our regular course program. Several of the students have part-time jobs as practising programmers.

4. THE PHASES OF THE EXPERIMENT

The course ran over one semester, with three hours a week. Our course plan admits a considerable degree of freedom for each course, which was utilized as described below. The course was given a somewhat pretentious title, that caused puzzled amusement when first announced: "Data system development at a high level of consciousness". This was amplified as follows:

> The purpose of the work is to throw light on what happens in the designer's mind during the development of data systems. The primary activity is work with specific projects, using a form of work that, as far as possible, retains the details of the development process. Among the aspects that are expected to be illuminated are the considerations of alternative solution possibilities, the exploitation of the designer's experience, and the influence of personal, temperamental differences among people. The primary form of work will be "system development at the typewriter", i.e. a form that requires the complete development process to be recorded in writing. This form will be supplemented by discussions and mutual comments. In addition some of the literature on system work will be referred to, see Naur (1972) and Dahl, Dijkstra, and Hoare (1972).

The work on the course was begun by 12 students on 1974 Febru-

ary 1st. The final concluding report on the work during the course was submitted at the end of May 1974 by 8 students.

The development of the course was not planned in detail beforehand, but was the result of discussions in which the students were invited to contribute freely. The activity may be described as 8 phases, admitting some overlap. (one per student ?)

Phase 1. Development of course goals

Starting from the announced course title and description, the students were invited to take part in a process of course goal formulation. This process was carried out during the first few hours of the course work, in four steps. In step 1 suggestions for part goals were solicited and collected on the blackboard. In step 2 the interest of each proposed goal was agreed upon, expressed along a five-level scale, with 1 representing "very high interest" and 5 representing "slight interest". In step 3 the difficulty and cost of contributing to each goal was estimated, the result being again expressed along a five-level scale, 1 representing "very easy" and 5 representing "very difficult". In step 4 the goals were ordered according to the product of the measures of interest and difficulty. This product varies from 1, corresponding to "very high interest" and "very easy", to 25, corresponding to "slight interest" and "very difficult". The resulting list of course goals and measures of interest and difficulty is given below.

Phase 2. First program development

Immediately upon completing the goal formulation during phase 1 the students were given the formulation of the problem to be solved during the first program development. The problem was to partition the nodes of a directed graph into maximal strong components. Details of this problem and of the students' solutions are given in section 5 below.

Course goals

I = measure of interest, 1 is high, 5 is low

D = measure of difficulty, 1 is easy, 5 is difficult

I	D	I×D	Goal, to contribute to the knowledge about:
1	-	-	What happens during program development
1	1	1	The importance of communication with others
2	1	2	The influence from other, unrelated activities
2	2	4	- - - previous education
2	2	4	- - - computer access
2.5	2.5	6	- - - personality, temper
2	4	8	The importance of the goal pursued by the programmer
4	2	8	What the literature says about program development
2	5	10	The importance of the programming language
2	5	10	In case of team work: the importance of the subdivision of the programming task
3	4	12	The influence of the level of documentation on the quality of the resulting program
4	4	16	The need for description tools during the initial program development phases
4	5	20	The importance of the technical level of the environment of the programmer
5	5	25	The influence from the planning
5	5	25	The importance of easy access to the literature

The students were asked to produce a detailed, written account of their thoughts as they developed during the solution of the problem, roughly along the lines of the description given in Naur (1972). A period of 2 weeks was assigned to the work, the expected effective work load being of the order of 20 hours. The students were asked to solve the problem by individual efforts, although they were free to communicate among themselves and with outsiders and to use the literature, as long as this was properly described in their work accounts. They were left a free choice of computer and programming language, to suit their individual background and situation.

Phase 3. Discussion of the first program development

Upon completion of the first program development the reports produced by the students were circulated among all participants, as a preparation of the discussion of the experience obtained. During a subsequent free discussion among all participants an attempt was made to identify the various kinds of activity that had gone into, or might have gone into, the program development. The result of this discussion was a list of activities, ordered according to the major stage of the program development. This list was later revised in various ways and in its final version identifies 33 activities. It is given in section 8 below. The list may be regarded partly as an output of the experiment as a whole, partly as a guide for the participants in their further work during the experiment.

Phase 4. First program development: goal analysis

As a result of the discussion of phase 3 it became clear that the development had been strongly influenced by the priorities of goals that had been adopted, more or less tacitly, by each student. In order to clarify this influence, each student

was asked to add to his or her program development report a summary report containing any such observations on the program development process that were deemed relevant. Specifically the summary report should indicate the stress put on particular goals. The priorities thus stated by the students to have been actually pursued are discussed in section 9 below.

Phase 5. Second program development

The choice of the problem to be solved during the second program development was made during a discussion among all participants. It was considered important that this problem would differ from that of the first program development, in such a way that as far as possible new aspects of the program development process would be brought to light. The problem selected was to develop a program that converts a representation of the hole patterns of a punched card into an equivalent representation of a string of characters of the ISO 7-bit character set. The problem is described in more detail in section 6 below.

For solving this problem the students were given a work period of 2 weeks, corresponding to an effective work load of the order of 20 hours. As in the first program development, the students were asked to record their progress in detail in writing and to end their development reports with summaries, including priorities of goals actually pursued. The terms of work differed from those of the first problem in that the students were given the freedom to work individually or in teams, according to their own preference.

Phase 6. Discussion of second program development

As in phase 3, the discussion of the second program development started with a circulation of all development reports among all participants. The discussion itself was directed toward completing the list of activities established during phase 3, to cover the new experience.

Phase 7. Third program development

The choice of the third problem again was the result of a free discussion among the participants. It was decided to focus on the problems of interactive computer use, but otherwise to leave the problem to be solved open. More details of the problem are given in section 7 below. The choice of the actual problem to be solved was left open because it was felt that the specification of any particular problem would limit the scope of possibilities to such an extent that the primary aim of the problem, viz. to invite the students to consider the varieties of interactive computer use, might not be realized. The formulation did create irrelevant problems for some of the students, who found it almost impossible to find a suitable subject for the program.

The students were given 4 weeks, corresponding to about 50 hours, for the third program development.

Phase 8. Students' concluding summaries

As the final phase the students were asked to produce individual, concluding summaries of their experience of the course work. In addition to comments in free format, the summary should include a more systematic account of the stress put on each of the program activities identified during phases 3 and 6. The outcome of this final enquiry is described in section 8 below.

5. SOLVING THE FIRST PROBLEM

The experiment produced a substantial material of reports from the students. As a whole the students have done the required, detailed recording of their work as they went along and the reports give a lively picture of their trials and errors, of their failures, disappointments, and successes. In this section and the four that follow their reports will be summarized.

The first problem, solved during phase 2 of the experiment, was taken from an unpublished paper by E. W. Dijkstra, EWD376, communicated privately to the present author in January 1974. Dijkstra's paper contains a detailed description of his solution of a problem formulated as follows:

> First problem: Maximal strong components of directed graph.
>
> Given a set of nodes and a set of directed arcs leading from a node to a node, it is requested to partition the set of nodes into maximal strong components. A strong component is a set of nodes such that the arcs between them provide a path from any node of the set to any node of the set; a single node is a special case of a strong component: then the path can be empty. A maximal strong component is a strong component to which no nodes can be added.

This problem was solved by 12 students working individually in between 5 and 15 hours. During their work they produced in the order of 15 pages of written documentation of the progress of their work. The method used by most students was to form the incidence matrix of the given graph and to use this for forming the transitive closure of the relations defined by the arcs. The maximal strong components may then be found fairly simply in a third step. In the descriptions below a solution thus based on forming the transitive closure will be denoted TC. Where, in addition, the algorithm is based on Warshall's theorem on Boolean matrices (Warshall 1962) the method will be denoted TCW. The individual solutions are described briefly below.

Student 1. Method: Traversal of graph using recursive procedure. Main effort: Inventing a solution; hand experiments with methods; debugging. Program: 149 lines of Algol 60, including brief comments.

Student 2. Method: TC. No special difficulty. Program: 137 lines of Algol 60, including brief comments.

Student 3. Method: TCW. No special difficulty. Program: 36 lines of Fortran, without comments.

Student 4. Method: Traversal of graph using recursive procedure. Main effort: Invention of a method. Program: 70 lines of PLC, a subset of PL/I, including brief comments.

Student 5. Method: TCW. Some effort spent on a comparison of alternative approaches. Program: 41 lines of Algol 60, without comments. The program only forms a matrix representation of the transitive closure of the arc relations, not the strong components directly.

Student 6. Method: TCW. Main effort: Proving that the method works; extracting the strong components. Program: 52 lines of Fortran, including brief comments.

Student 7. Method: TCW. Main effort: Justification of the use of the transitive closure; selecting an efficient representation. Program: 48 lines of Algol 60, including brief comments.

Student 8. Method: TCW. Main effort: Trying to get through with traversal of graph using a recursive procedure, before switching to TCW. Program: 66 lines of Algol 60, including brief comments.

Student 9. Method: TCW. Main effort: Proving that the method works correctly. Program: 62 lines of Algol 60, including brief comments.

Student 10. Method: TCW. Main effort: Design of input and output; analysis of the execution time of the solution. Program: 37 lines of Algol 60, including brief comments.

Student 11. Method: TC. Main effort: Finding a suitable criterion of convergence of the process. Program: 142 lines of Algol 60, including brief comments.

Student 12. Method: Traversal of graph. Main effort: Trying to

understand the problem (without success). Program: 140 lines of Fortran, including lengthy comments. The solution does not solve the given problem.

As a whole, the effort in solving problem 1 was spent on play with methods of solution, often alternating between algorithmic descriptions, expressed more or less formally, and hand experiments to see whether the method would work. In this work descriptions at a high level are certainly used, as in structured programming. This does not always prove to be a help. For example, student 8 arrives on page 4 of his description at a beautiful strategy, expressed as 7 high-level actions, tied in with a clear data representation. He admits that he has not gone into complete detail, but has a strong feeling that the scheme can be realized fairly simply by means of recursive procedures. The report of the next day concludes that the scheme cannot be realized without too much administration and trouble. He then switches to Warshall's algorithm.

A comparison of the students' solutions and Dijkstra's unpublished solution, mentioned above, shows Dijkstra's approach to differ radically from that of any of the students'. Most strikingly, Dijkstra makes no use whatever of examples, and even ends by insisting that examples should not be used, as a matter of principle. Such a recommendation contrasts strongly, not only with the approach used successfully by the students, but also with the recommendations of those who have studied creative mathematical reasoning, see Polya (1954).

6. SOLVING THE SECOND PROBLEM

The second problem, solved during phase 5 of the experiment, requires the development of a program that converts a representation of the holes of a punched card that takes the card row-wise, into

a representation of the characters of the card columns in the form of a string of ISO 7-bit characters packed into words. As the basis for the conversion the standard ISO representation of characters on cards is given.

This problem was solved by 8 students or pairs of students. They used between 10 and 35 hours of work on it, producing between 6 and 40 pages of documentation. The solutions were briefly as follows.

Students 1 and 2. Main effort: Finding a space-economic representation of the conversion table; generation of test cases. Program: Approx. 230 lines of Algol 60, including tables and comments.

Students 3 and 6. Main effort: Finding a space-economic representation of the conversion table. Program: 105 lines of Fortran, including tables and comments.

Student 4. Main effort: Simulation of the card reader; production of the internal conversion table. Program: 26 lines of PL/I, 87 lines of assembly language for the IBM S/370, both including comments.

Student 5. Main effort: Comparison of 7 alternative methods of conversion; programming of a highly ~~effective~~ efficient conversion process. Program: 90 lines of Algol 60, including brief comments.

Student 7. Main effort: Finding a space-economic representation of the conversion table; simulation of the card reader. Program: 120 lines of Algol 60, including tables and brief comments.

Students 8 and 9. Main effort: Achieving fast conversion, paying attention to the high frequency of code positions with no holes. Program: 204 lines of Algol 60, including brief comments.

Students 10 and 11. Main effort: Getting familiar with a new computer at the level of machine language. Program: Approx. 440 lines of assembly language for the PDP 11/45.

Student 12. Main effort: Finding out about the operation of

a card reader at the machine language level. Program: Approx. 210 lines of Fortran, including tables and comments.

In brief summary, finding a solution to the second problem did not cause any difficulty. Rather the effort was spent on selecting a reasonably ~~effective~~ efficient solution. Much of the effort went into finding usable, systematic patterns in the card column representations of the ISO 7-bit characters. In addition there were problems of finding out how the equipment operates at the appropriate level of detail.

7. SOLVING THE THIRD PROBLEM

The third problem, solved during phase 7 of the experiment, was to develop a self-documenting program. By self-documentation is meant that the user for the most part gets to know the features of the program and to work out input data through a dialog with the program. The subject of the program was left to the free choice of each student. The final results of the development were to be made available for the other participants in the course. The reactions thus provoked were to enter into the discussion of the testing of the programs.

Problem 3 was attacked or solved by 7 students or pairs of students. It caused considerable interest, the time spent by each student being typically of the order of 40 hours, but ranging all way up to 130 hours. The documentation produced ran between 10 and 75 pages. The solutions were briefly as follows.

Student 1. Problem selected: Postage determined from destination, weight, etc. Main effort: Finding a structure in the postal rules; organization and contents of dialog. The solution was not carried through to a program.

Student 2. Problem selected: Development of programs that

control a graphical plotter, i.e. interactive teaching of a special-purpose programming language. Main effort: Development of the dialog and the associated communication procedures. Program: Approx. 1150 lines of Algol 60, including extensive comments.

Students 5 and 7. Problem selected: Editing of Algol 60 programs, with extensive checking. Main effort: Organization of the dialog; comparison of ways to store the program text being handled. The solution was not carried through to a program.

Student 6. Problem selected: Filling in football coupons on the basis of winning chances of teams. Main effort: Development of the dialog. Program: Approx. 370 lines of Fortran, including brief comments.

Students 8 and 9. Problem selected: Multipurpose conversational system, with stress on games and jokes. Main effort: Development of conversational basis to achieve variations in the dialog; formulation of adequate machine answers; design of particular games. Program: Approx. 1120 lines of Algol 60, including texts and comments.

Student 10. Problem selected: Game equivalent to noughts and crosses on a 3 by 3 board. Main effort: Analysis of the game, to arrive at an adequate strategy. Program: 480 lines of Algol 60, including brief comments.

Student 11. Problem selected: Postage determined from destination, weight, etc. Main effort: Design of communication procedures; representing the postal rules. Program: Approx. 450 lines of Algol 60, including brief comments.

It may be added that the somewhat artificial formulation of the third problem, a formulation that could hardly be encountered in a real-life problem, turned out to affect only a short, introductory phase of each program development. As soon as the students had

decided on what problem to solve the development proceeded quite normally, except that the interest in the problem was rather higher than normal.

8. THE STUDENTS' CONCLUDING SUMMARIES

In the last phase of the experiment, phase 8, the students wrote summaries of their experience. In order to obtain information that would lend itself to a more systematic comparison of the students among one another and of the three problems, the students were asked to relate their experience to the activities identified on the list produced during phases 3 and 6. This list has 33 items and is repro-

LIST OF PROGRAM DEVELOPMENT ACTIVITIES HERE

duced below. Each student was asked to fill in a form having 33 lines corresponding to the activities and a total of 6 columns, two for each of the three program developments. Each column should contain a set of weights adding up to 100, describing the relative effort spent on the various activities. For each of the three program developments the first column should give the effort actually spent during the course work, while the second should give the effort that should have been spent, according to the student's retrospective view of the most appropriate way to do program development. The material thus consists of $33 \times 6 = 198$ weights for each of 8 students, or in total 1584 weights. Before describing the contents of this material it should be noted that 3 of the 8 students remark on the difficulty of completing the table, saying that they found it "exceedingly difficult" or "awfully difficult".

For the purpose of analysis these weights were plotted in $3 \times 33 = 99$ small activity maps, one for each activity and problem. In each activity map the weights given by each student to the activity of that particular problem were represented by a point, the weight

Program development activities

A. Early stage of program development

1 Look at examples

2 Get ideas by talking to others

3 Use the theory you know, theorems, methods

4 Use the literature, particularly that which you have read before

5 Describe the problem in mathematical terms

6 Internal data representation, think of alternatives

7 Method, strategy, think of alternatives

8 Delimitation of the problems that will be covered

9 Selection of computer and language, think of alternatives

10 Make explicit the dependence of the solution on computer and programming language

11 Input data, think of alternative forms

12 Check of input data

13 Output data, alternatives

14 Describe the main sections of the program, perhaps as a flow chart

15 Central algorithms

16 Invariant data defining the flow of control, in particular control tables, require the utmost reliability; consider using check digits

17 Estimate the storage and execution time requirements of the program

18 Put priorities on the goals of the program development

19 Test input data, consider automatic generation

20 Test output, where in the program should it be produced, and which

B. Intermediate stage of program development

21 Plan for the work, what should be done

22 Check of the solution chosen, proofs

23 Invariant descriptions of the data representation

24 Action clusters

25 General snapshots

26 Alternative realizations, better solutions

27 Testing, collection of test cases

28 Documentation, collection of notes intended for various consumers

29 Means for measuring the execution time

30 Automatic generation of test input data

C. The program is written, before debugging on computer

31 List of variables with columns recording declaration, initialization, use, and change of value

32 Step-by-step desk testing with trivially simple input data

D. Retrospective analysis

33 The resource efficiency, comparison with earlier estimates

actually given being the x-coordinate and the weight that should have been spent being the y-coordinate.

A point in an activity map shows at a glance a good deal of the student's view of the particular activity as applied to the particular problem, as shown in figure 1. When the points corresponding to all

FIGURE 1 HERE

students are plotted in the same activity map the scatter of the points shows at a glance whether there is agreement among the students on the weight that has been given, or should have been given, to the corresponding activity. Finally, in a comparison of the activity maps corresponding to the three problems the influence of the nature of

the problem on the students' view of the activity shows itself. Sample activity maps are given in figures 2 and 3.

A scrutiny and comparison of the activity maps yields the following:

1. There are large variations among the students in the sense that in no activity is there a uniformly high weight for all students. Even in the case of activities that receive the highest average weight there occur very low weights for some students. As a typical illustration, figure 2 shows the maps for activity 15, central algorithms. This activity is among the three or four that receive the

FIGURE 2 HERE

highest average weights. Even so, some students have given no weight whatever to this activity. This general feature of the activity maps should be kept in mind during the following descriptions.

2. A group of activities receives high average weights of the order of 10 in all three problems, with no clear tendency that this weight should have been different. Figure 2 shows the activity maps of a member of this group. Ordered with decreasing average weight the activities of this group are:

7 Method, strategy, think of alternatives

6 Internal data representations, think of alternatives

15 Central algorithms

27 Testing, collection of test cases

11 Input data, think of alternative forms

3. A group of activities receives moderate average weights of the order of 5 in all three problems, with no clear tendency that this weight should have been different. The group includes:

26 Alternative realizations, better solutions

12 Check of input data

18 Put priorities on the goals of the program development

19 Test input data, consider automatic generation

20 Test output, where in the program should it be produced, and which

32 Step-by-step desk testing with trivially simple input data

8 Delimitation of the problems that will be covered

21 Plan for the work, what should be done

10 Make explicit the dependence of the solution on computer and programming language

4. A group of activities receives moderate average weights of the order of 5 in all three problems, with a fairly clear tendency that the weight should have been greater than the one actually applied. A sample of this group is shown in figure 3. The group

FIGURE 3 HERE

includes:

28 Documentation, collection of notes intended for various consumers

2 Get ideas by talking to others

13 Output data, alternatives

33 The resource effectiveness, comparisons with earlier estimates

29 Means for measuring the execution time

17 Estimates of the storage and execution time requirements of the program

5. One activity receives moderate average weight of the order of 5 in all three problems, with a fairly clear tendency that the weight actually applied should have been smaller:

14 Describe the main sections of the program, perhaps as a flow chart

6. A group of activities receives weights that differ markedly from one problem to the other. With one exception (activity 1, problem 1) there is no clear tendency that the weights should have been different. The activities of this group and the average weights given to them in the three problems are as follows:

Activity	Average weight: 1 Maximal strong component in graph	2 Punched card code conversion	3 Interactive program
1 Look at examples	Too high	Low	Moderate
3 Use the theory you know, theorems, methods	Moderate	Low	Low
4 Use the literature	High	Low	Moderate
5 Describe the problem in mathematical terms	Moderate	Low	Low
16 Invariant data defining the flow of control	Low	High	Moderate
22 Check of the solution chosen, proofs	High	Low	Low
30 Automatic generation of test input data	Low	High	Low

7. The remaining activities receive low average weights of the order of 2 in all three problems, with no clear indication that the weights should have been different. These activities are:

9 Selection of computer and language, think of alternatives

23 Invariant descriptions of the data representation

24 Action clusters

25 General snapshots

31 List of variables with columns ...

The points made in the remaining, unstructured part of the students' concluding summaries may be summarized as follows. Of the 8 students 7 remarked on the form of work, the recording of the program development process through typing while the work is in

progress. One student notes that this form is "a bit tiring", another that "it is a strain to have to make specific one's more or less crazy ideas and to keep an eye on the time used". However, these sides seem to be compensated amply by several positive aspects: "it forces you to work more consciously", "the form of work was essential to the result; it made me consider many ideas more thoroughly and increased the joy of the work by the clarification of the ideas", "specially in case of team work the form is a great advantage; you don't have to discuss many ideas several times when somebody has forgotten them again". One participant notes that "perhaps it is not too surprising that such techniques are fruitful; they are similar to the care with which experimentors work with their note books". Another student concludes that "the basis for analysis of program development must be reports of the form used during the course".

Some of the students comment on the course as such. One found it "an interesting and instructive course to participate in", another has found it "a bit varying and generally with a feeling that nobody knew what should come out of it; even so I found many of our discussions fruitful", and yet another has "gained experience with more effective modes of work that undoubtedly will prove useful later". A criticism voiced by several students is that too little effort was put into the analysis of the material collected.

The list of program development activities was commented on by two students. One of them writes: "The check list, perhaps suitably extended, is an excellent cookbook. It should be used regularly during one's work ... Since the check list takes a broad view there will always be a number of points that apply well to the task at hand. The only thing that I cannot follow is the division into the four stages. It cuts across my own habits." Another student writes that "a list of concepts and methods tends to become unwieldy and uninter-

esting, since many items on the list often are meaningless in relation to the specific task at hand. On the other hand a set of hints and guidance may be very useful, in particular in connection with the teaching at a more elementary level, but to be of value such a guide must be strongly supported by arguments and examples. It can hardly be less than perhaps 100 pages long".

One student comments on the analysis of goals (see section 4, phase 4, above). He writes: "The idea to define the goal explicitly seems to me very successful, because it forces one to keep a rather tight rein on oneself and not to neglect certain parts of the problem being solved. Further one is forced to justify changes of the goals as they come along, which is a good thing."

9. GOAL PRIORITIES AND STRUCTURED PROGRAMMING

As a result of phase 4 of the course, it was decided that each participant should try to characterize the goals actually pursued during the program development. In order to obtain comparable data it was decided that the students should state the stress that they had put on each of 7 goals, in terms of a scale ranging from 1, "very slight stress", to 5, "the utmost stress". Unfortunately the question was not pursued sufficiently vigorously at the time and the data actually obtained are quite incomplete. For this reason the results shall only be summarized briefly, taking the goal priorities for all three problems and all students together.

Goal 1, to produce full documentation, had a fairly low priority, in the range from 1 to 3. Goal 2, to get through to a workable program, was put very high, with a few exceptions. Goal 3, to produce a neat program, was put either very low or very high, with few in between. Goal 4, to follow one's inclination, is put roughly equally along the range of priorities. Goal 5, to produce a fast program, also is put

nearly equally along the range, with only a slight tendency toward high priorities. Goal 6, to achieve good economy of storage, was placed about equally as goal 5. Goal 7, to achieve freedom of programming errors, was put very high, above any other goal.

Even on this limited evidence it may be concluded that the actual goals pursued differ considerably from one student to the other. In view of the results reported by Weinberg (1972) a difference in explicit goal may have considerable influence on programming performance. It thus seems likely that the difference in goals adopted by the students will have caused a significant difference in their solutions. This effect may explain some of the large differences in the stress put on various programming activities reported in section 8 above. Thus it might seem that the present experiment suffers from inadequate formulation of goals, and that further experiments should pay more attention to this point.

The issue of goal selection raises some difficult questions in relation to the ideas of structured programming. For one thing, it is not clear whether to regard structured programming as a goal in itself, or whether to regard it as a means to some other end, and if so, which. Further questions arise that possibly could be settled merely by using clearer concepts and terms. Thus we find in section 8 point 2 as the two activities that receive the highest average weight in the present experiment: "7 Method, strategy, think of alternatives" and "6 Internal data representation, think of alternatives". If these activities are not structured programming, at least they are closely related to it. Perhaps, unwittingly, the students in the experiment have been doing structured programming?

10. CONCLUSIONS

On the basis of the limited material and the incomplete analysis of the present study only a few firm conclusions are warranted. The most useful conclusion seems to be that the experimental method used, viz. individual, continuous recording in writing of the program development, is capable of yielding usable material, at least with some, not too exclusive, groups of people.

Second, it seems clear, what common sense would have, but what often seems to be forgotten in discussions of programming methods, that there are important techniques that should or should not be used, depending on the particular problem being solved.

Third, as the experiment was actually conducted the stress put on various program development activities varied greatly from one student to another. It appears likely that some of these differences might have been smaller if the goals to be pursued by the students had been more clearly defined. However, whether clear goal definitions would remove all personal differences can only be determined by further experiments.

Fourth, the relevance and importance of the ideas of structured programming to the kind of program development considered in the experiment remains unclear. In particular the relation of structured programming to the goals of programming needs clarification.

REFERENCES

Dahl, O.-J.; Dijkstra, E. W.; and Hoare, C. A. R. "Structured Programming". Academic Press, London, 1972.

Gold, M. M. "Time-sharing and batch-processing: an experimental comparison of their values in a problem-solving situation". Comm. ACM 12, 5 (May 1969), 249-259.

Henderson, P.; and Snowdon, R. "An experiment in structured programming". BIT 12 (1972), 38-53.

Ledgard, H. F. "The case for structured programming". BIT 13 (1973), 45-57.

Naur, P. "An experiment on program development". BIT 12 (1972), 347-365.

Polya, G. "Induction and analogy in mathematics. Volume 1 of Mathematics and plausible reasoning". Princeton University Press, 1954.

Sackman, H.; Erikson, W.J.; and Grant, E. E. "Exploratory experimental studies comparing online and offline programming performance". Comm. ACM 11, 1 (Jan. 1968), 3-11.

Schatzoff, M.; Tsao, R.; and Wiig, R. "An experimental comparison of time sharing and batch processing". Comm. ACM 10, 5 (May 1967), 261-265.

Warshall, S. "A theorem on Boolean matrices". J. ACM 9 (1962), 11-12.

Weinberg, G. M. "The psychology of improved programming performance". Datamation 18, 11 (Nov. 1972), 82-85.

Wirth, N. "Program development by stepwise refinement". Comm. ACM 14 (April 1971), 221-227.

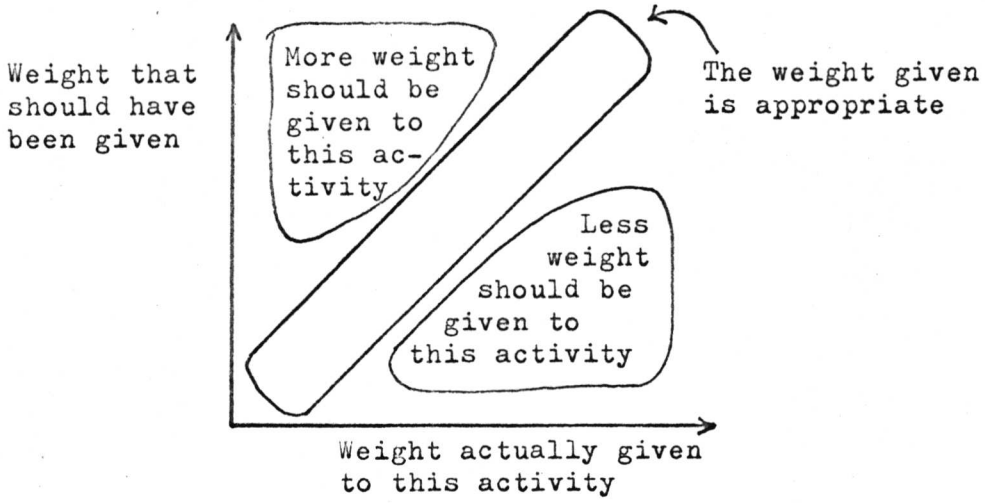

Fig. 1. Interpretation of activity map

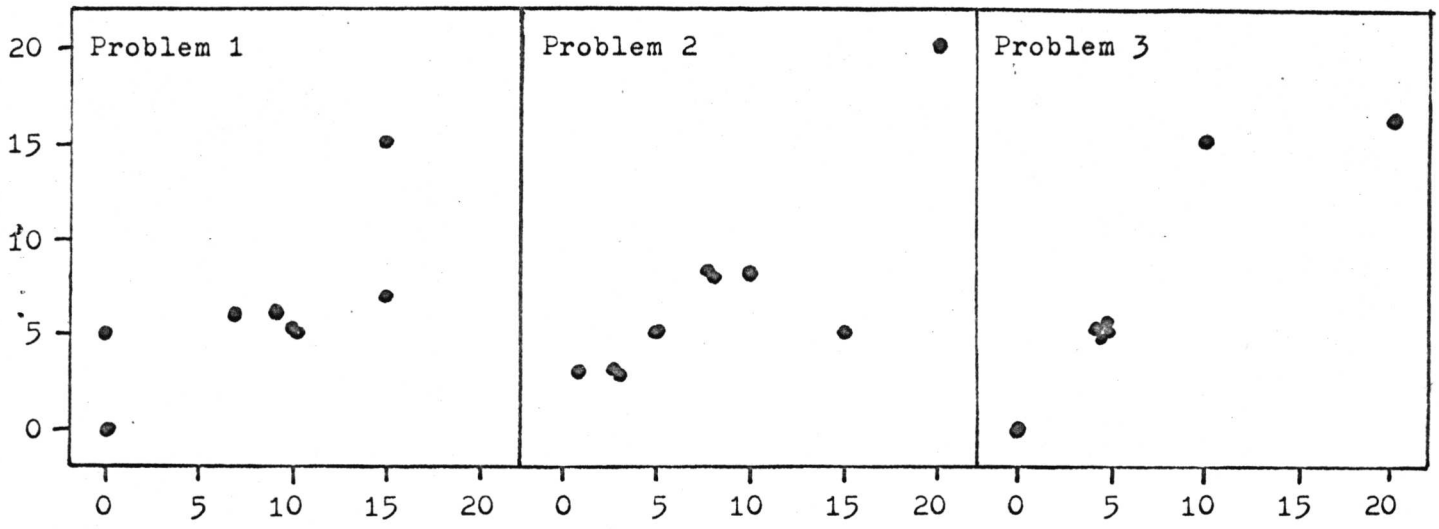

Fig. 2. Activity maps for activity 15: Central algorithms

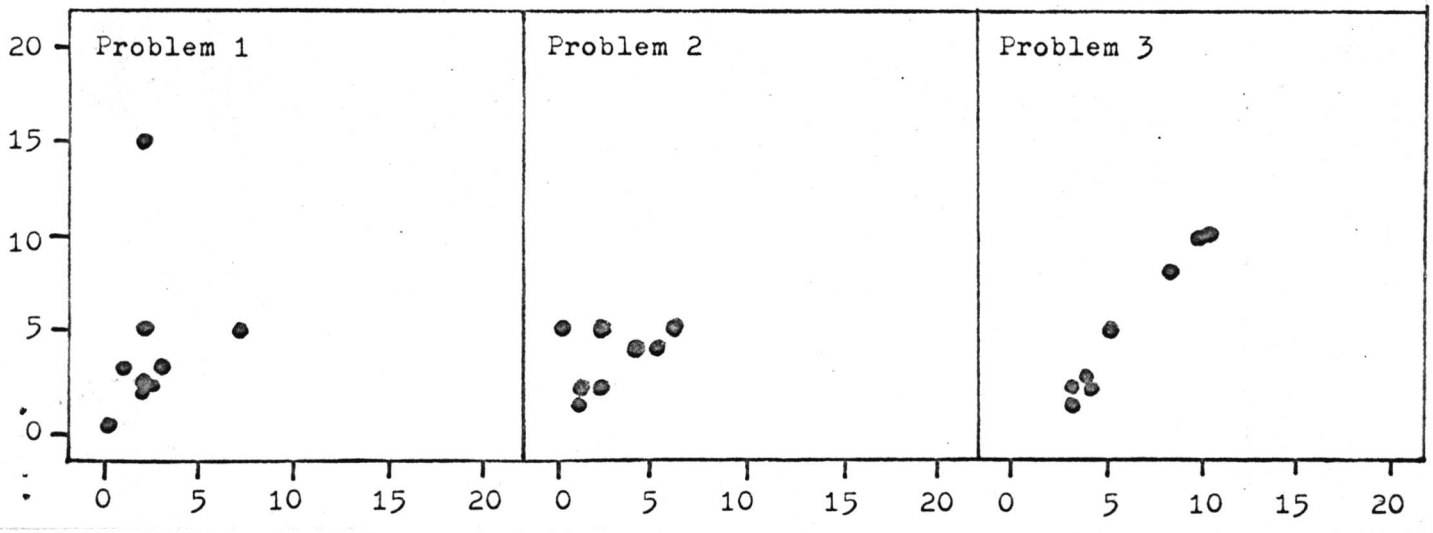

Fig. 3. Activity maps for activity 13: Output data, alternatives