SELECTION OF SYSTEMS ANALYSTS AND PROGRAMMER TRAINEES

RAYMOND M. BERGER
University of Southern California

The subject of this talk is primarily the selection of Systems Analysts. Time permitting, I shall also discuss work done on the selection of programmer trainees. The research program, as many of you have heard described in previous talks, was divided into three phases. The first was the job analysis phase in which 17 job dimensions were revealed to cover systems analyst and programmer positions. The second phase dealt with the development of criterion tests for evaluating programmer and systems analyst performance. The third phase was concerned with the selection of people for training in programming and systems analysis. I have reported on the results of the job analysis phase, and on the development and use of a test dealing with programmer proficiency, the Basic Programming Knowledge Test. Today I will report on the other criterion test, the Systems Analysis Test. A detailed report on the development of the Systems Analysis Test and an analysis of the results of its use is planned. I will discuss some of those results here.

Why do we want to develop a systems analyst test? Figure 1 shows a prediction diagram which illustrates the effect of certain errors in the prediction of job performance. This prediction diagram relates not only to systems analysis jobs, but to other types of jobs as well. In the diagram either success or failure is predicted on the basis of the test or the selection procedure that is being used. Job performance for the individual predicted is categorized simply either success or failure on the job according to a given criterion. If you predict that the person will be successful on the basis of your selection procedure, and he is successful on the job, you have a "hit" . . . a high hit. However, if you predict that he will be successful and he is unsuccessful, you have a "false positive." If you predict he is going to be a failure and he fails on the job, then you have another hit . . . it is called a low hit. However, if you predict he will be a failure and he actually succeeds on the job, then you have a "false negative." You try to reduce both the false positivies and the false negatives in any selection procedure.

In the situation where there is a small- to moderate- size company, where every technical person really counts, and where you place a great deal of reliance upon the systems analyst, you would like to reduce the false positive error as much as possible. In this case, reducing the false negatives is incidental; we are more concerned with avoiding the person who is a false positive. He may be selected for a crucial job and it is very important for a small organization to be as successful in its prediction as possible. When a large computer organization, like IBM, is involved, then perhaps you can afford a certain proportion of false positives. If you have many positions to fill, you certainly do not want to lose people that are good, so there the objective may be to reduce the false negatives.

Briefly then, in the situation where you need to select the right person for an important job, you want to reduce the probability of a false positive. In the

		JOB PERFORMANCE	
		Failure	Success
PREDICTION	Success	False Positives	High Hits
	Failure	Low Hits	False Negatives

FIGURE 1
Prediction or Selection Errors Diagram

situation where you need many people, where you can afford to let the job shake people out, and you do not want to risk losing good people to the competition, you try to reduce the false negatives. To do this you may actually lower your cutting scores or your acceptance standards to be sure you get as many of the good people as possible, letting those who are not good (false positives) fall out along the way after selection.

The Systems Analysis Test was devised primarily to reduce the false positives as much as possible. It increases the accuracy in predicting high hits. It may also be effective in distinguishing between the high hits and the false positives when it is used after a preliminary prediction is made by other methods.

Now let me turn to the kind of test and the kinds of problems that are appropriate for the systems analyst. The problems should call upon the individual to display analysis and synthesis abilities. Multiple-choice questions do not work well here, because when we test recognition or recall only, we miss important parts of the systems analyst job. Missing are the types of problems in which he has to create and synthesize as well as analyze. We therefore used the short essay, listing, or fill-in type of format predominantly. The content of the problems was related to the systems analysis dimensions found in our job analysis.

Figure 2 shows a Computer Position Profile of a systems analyst based upon the job analysis. This is for a typical systems analyst at a government agency in Washington. This is not necessarily typical of all systems analysts. In a job analysis of the positions we found that the activity dimensions were high on Program System Analysis and Design, Program System Analysis in Business and Logistics, Program Production Planning and Scheduling, and Lead Programming Responsibility. Surprisingly, the profile dimensions were also high on Utility Program Development-General Purpose and Library, Program Systems Testing, and Program Modification and Installation. As expected, the profile is low on the General Programming Operations and Debugging dimensions. Using the "high profile" dimensions and the activities associated with them, we tried to construct problems, or rather, gave instructions to our experts to construct problems for our Systems Analysis Test.

I have gone around to a great number of computer installations, and convened panels of experts to find a definition for systems analysis. I found that systems analysis is a many-splendored thing, but like love it is also a very illusive thing. So I decided to let the set of problems that we developed out of the job activities, and the problems that were reported to be good, define the area of systems analysis. This would constitute an operational definition of systems analysis given by the operations required to do the problems.

NAME Systems Analyst

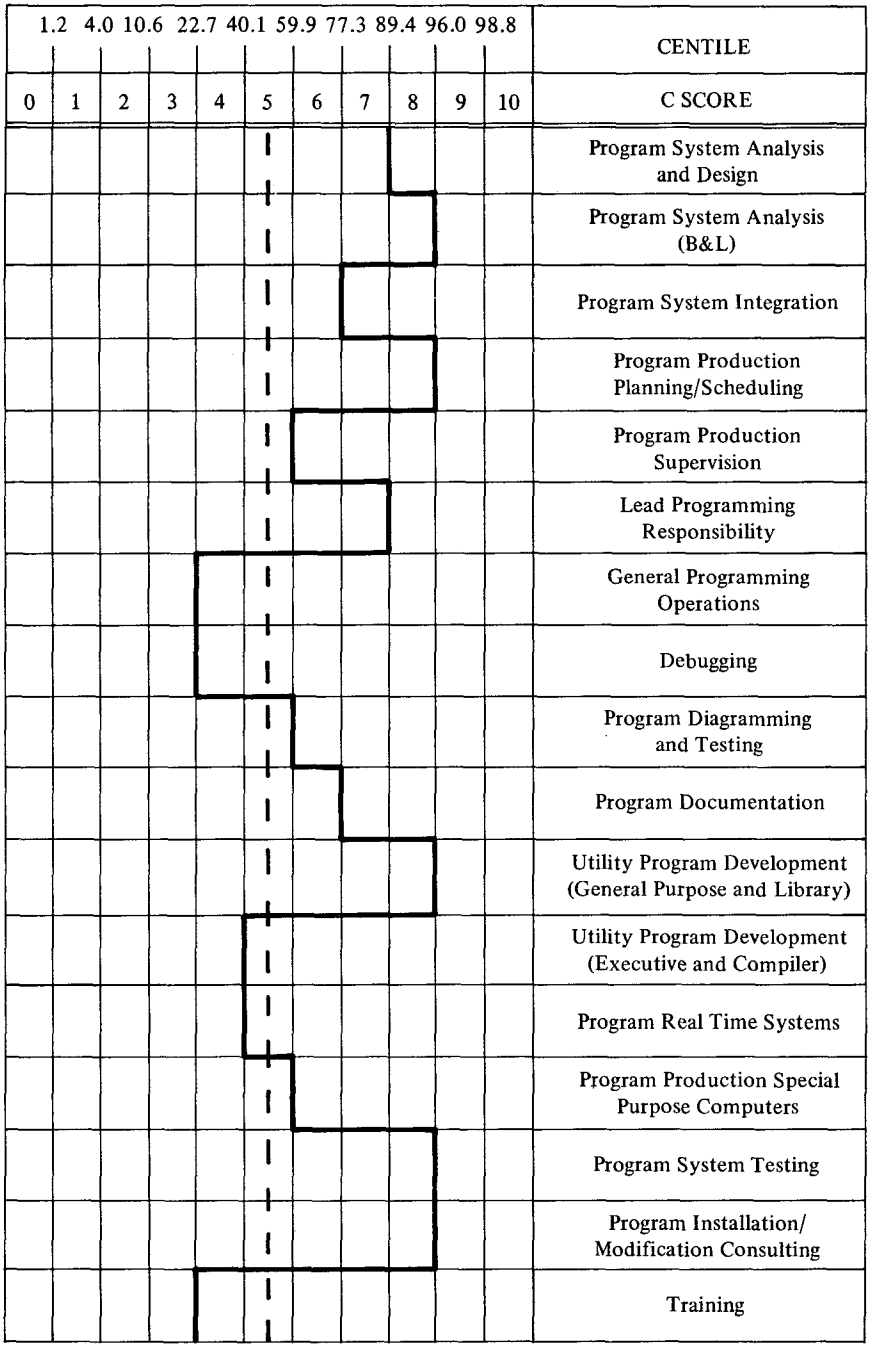


FIGURE 2
Computer Personnel Position Profile

About 50 problems were constructed initially pertaining in some way to the systems analysis field. They were done by systems analysts, systems designers, and systems experts of other designations. Tryouts of the problems were made at several conferences and organizations including the ACM Conference and the Fall Joint Computer Conference. The problems were revised or eliminated on a basis of feedback from these tryouts. Expert opinion was used all along, individually and collectively, for evaluation of the problems at any one particular stage. The final set included 18 problems divided into three sections.

Figure 3 shows the subject evaluation sheet for one subject in an aerospace company that participated in the preliminary tryout of the final set of problems. First, we might look at the kinds of problems included. Shown are problems relating to inventory control, data transmission cost determination, computer conversion considerations, a problem on multi-file system-inherent problems, multi-file system-cost benefit analysis, a billing procedure logic, questionnaire form codification, and airline reservation flowchart changes.

Section two problems become a bit more abstract: computer configuration selection in a business area, computer configuration selection, in a scientific area, a punch-card layout, an evaluation of EDP installation costs, an expansion of the management information system, a traffic pattern analysis, a computer assisted instruction situation in which the subject has to list the system functions and the lesson functions, and to show the CAI configuration design. The subject receives separate scores for sections one and two, and a combined score for both sections. The third section, which consists of one rather complex problem, is also separately scored and is used optionally for special area testing.

Each problem was scored using four evaluation categories: zero was given for no response or an unacceptable response; one as the weight for a poor response; two as the weight for an acceptable response, and three as the weight for an excellent response.

In the first tryout-test scoring, we tried to adjust the evaluation scale for each problem to fit both the range of responses and the expectations of the experts. The development of a reliable scoring system was the most difficult part of the total test development. Since we have short essays and listings, the problem was to make the scoring as objective as possible. I would like to go into this in some detail lest the charge be leveled that the Systems Analysis Test is as subjective and unreliable as other tests that are open-ended. The validity of the Systems Analysis Test rests primarily upon its content validity, and the primary objective in the scoring was to achieve score reliability. The final set of problems was first tried out on 20 systems personnel at a Los Angeles aerospace company. The test administration took about three hours, two and one-half in actual testing time. Using the scoring guide developed prior to the testing by my group of experts, two judges, working independently, scored the 20 papers. These two then got together and reconciled their differences in evaluating each subject on each problem. The scoring guide was revised at that time. Using the revised scoring guide, a third judge who was kept unaware of the combined scores given by the previous two judges, evaluated the 20 papers. His scoring was compared to the combined scoring by the first two judges, and the two sets of scores were found to be very close.

Subject Evaluation Sheet No. 002

Section I Score 11
Section II Score 13
Total Score 24

SECTION I

Problem	Unacceptable 0	Poor 1	Acceptable 2	Excellent 3
Inventory Control	x			
Data Transmission Cost Determination				x
Computer Conversion Considerations		x		
Multifile System – Inherent Problems			x	
Multifile System – Cost Benefit Analysis			x	
Billing – Procedure Logic		x		
Questionnaire Form Codification	x			
Airline Reservation Flowchart Change			x	
Score Category Number	2	2	3	1
Weighted Score	0	2	6	3

SECTION II	Unacceptable 0	Poor 1	Acceptable 2	Excellent 3
Computer Configuration Selection (Business)		x		
Computer Configuration Selection (Scientific)			x	
Punched Card Layout	x			
Evaluation of EDP Installation Costs				x
Expansion of a Management Information System			x	
Traffic Pattern Analysis		x		
CAI – System Functions			x	
CAI – Lesson Functions		x		
CAI – Configuration Design		x		
Score Category Number	1	4	3	1
Weighted Score	0	4	6	3

FIGURE 3
Systems Analysis Test

We decided to see if a graphic evaluation form (as in Figure 3) would be useful in evaluating an individual. The total score for this individual was 24, which was a little above average for this group. The grades on the individual problems are also shown, I had a systems expert study the problems and the grades made, and write an evaluation of each individual. He drew heavily on Bloom's Taxonomy of Educational Objectives, which discusses analysis, synthesis, seeing relationships and implications, etc., abilities I thought would be important to the systems analysis job. The following is from his written report.

> Subject #2 -- Subject Evaluation. The subject scores at the acceptable level or better on those problems that require high level system analysis experience and skills. His reponses indicate he has the knowledge required in computer-oriented problems that are associated with the systems analysis level of work. These include identifying probable requirements, and determining feasibility and desirability. The scores also indicate familiarity with systems and data base concepts, and to some extent aspects of computer hardware. Of the problems on which he performed poorly, most were designed to measure one's ability to perform low-level, design tasks. These tasks include detail program design and input card formating.

Such an evaluation would be useful, I think, to the person who is doing selection. He can match it against the job requirements to see whether he wants to hire the individual, to train further in certain areas, or to reject him.

A larger sampling with the Systems Analysis Test was done in the Washington area with 60 systems personnel. In that testing I developed norms for the group that could be used in future selection. I found that scoring development had to proceed further because the range of responses for the larger group was much broader than that for the previous group of 20 analysts. The scoring was done, again, by several judges. The scoring guide, however, was expanded because of the broader range of responses. Every individual response for the first 45 individuals was classified as acceptable or non-acceptable. Supplementary lists to the scoring guide were then made up consisting of these acceptable and non-acceptable responses as agreed upon by these judges.

The scoring guide and the supplementary lists were then used to score the last 15 papers by two scorers working independently; the results showed high inter-judge agreement. The judges found that if they could not score the person using the general scoring principles and examples, they could go to the list of acceptable and the nonacceptable responses and find a response that was very similar and use that particular scoring weight. When the scoring was completed the part and total scores were used to establish norms, and the results in the form of percentiles were fed back only to the people tested.

Background data were collected from the people tested and related to the Systems Analysis Test scores. The total group mean for sections one and two was 22.1 and the standard deviation was 6.3. The correlation between sections one and two, which can be considered one kind of reliability check, was .60 corrected by Spearman-Brown prophecy formula. The first table in Figure 4 shows the score comparison for the 47 males and 12 females. As you can see, the females have the slight edge, contrary to the view that has been expressed by some people regarding women doing systems analysis work. The score ranges are 10 to 35 for the male and 13 to 35 for the females. The score difference is actually too close to be significant. I would speculate that when a women does get into system analysis work, she is really good; certainly any organization that

BACKGROUND VARIABLES AND TEST SCORES

Total N = 59
Section I and II Mean Score = 22.1
Standard Deviation = 6.3

SEX

	N	M	RANGE
Male	47	21.9	10 – 35
Female	12	22.7	13 – 35

AGE

	N	M	RANGE
45 – 49	7	22.9	16 – 30
35 – 44	17	22.7	11 – 32
30 – 34	15	22.1	10 – 35
25 – 29	18	22.1	13 – 35

FIGURE 4
Systems Analysis Test

is considering hiring a female would not go wrong if she fulfills the proper training and experience requirements.

With regard to age, there was a very comforting result. As we go from 25 to 49 years, we find that the mean scores do not decrease as we had found with the Basic Programming Knowledge Test. A decrease in scores with age is true with most aptitude tests. The Systems Analysis Test results, if anything, show a slight increase in score with age. The trend, however, is not significant. Although the sample is not large, these results suggest that age should not be a factor in the selection of experienced systems analysts.

Figure 5 shows the relationship of the scores with education level. As you can see, there were only four people in the sample whose highest level was high school. For 1, 2, or 3 years of college completed, there were only nine people. An interesting trend appears here with system analysts as it has in other studies that I have done with programmers and programmer trainees. I found in these studies that there is no linear increase in scores as the education level increases from high school to one, two and three years of college. But we did get a significant score increase in the three studies between three years and four years of college completed (or the bachelor's degree). Moreover, we had an even greater jump in scores between the 4 year level only and those who have done postgraduate work. My guess is that the person who drops out of college because of grades or has only gone to a two-year college usually has less computer-related ability than the fellow who completes 4 years of college. This is something that can be considered in selection. The person with less than a bachelor's degree should be looked at pretty hard.

With regard to college major, there were some surprising results. In the Basic Programming Knowledge Test study, and the aptitude tests study, those people who indicated math as their major in college usually get better scores. But in this particular study, the mean score for 12 math majors was slightly below the group average, and the mean for 9 engineering majors was right at the group average. From the results here, one should not simply select somebody who indicates a math major; he should look pretty close at the math grades and at the college where the applicant majored in math. The big surprise, however, was the high mean, 24.0, for the business and accounting people. These people may be getting better training in systems concepts in college. They perhaps have had more exposure to the types of problems given in billing, data cost transmission, and so forth, and it may be for that reason, also, they have done better.

Figure 6 presents the score relationships for one of the questions asked concerning the training the examinee had received. The examinee indicated one of the following: no training in systems analysis; on-the-job training; a formal course at school or at the plant; or both formal and on-the-job training. As you know, there is no standardized curriculum for systems analysis training, and I think perhaps the results reflects this. The means were not too different between those who had received training and those indicating none. There was no particular advantage to either formal training or on-the-job training. There was a slightly higher score for those people who indicated they had a formal course

EDUCATION LEVEL

	N	M	RANGE
High School	4	19.5	11 – 27
1, 2, & 3 yrs. College	9	19.2	12 – 30
4 years or B.A.	29	21.4	10 – 35
Post Grad.	16	25.6	19 – 35

COLLEGE MAJOR

	N	M	RANGE
Math	12	21.1	11 – 35
Engineering	9	22.3	12 – 27
Physical & Biol. Sci.	2	28.0	26 – 30
Soc. Sci., Education, Humanities	13	20.1	10 – 30
Acct. & Bus.	14	24.9	17 – 35

FIGURE 5

TRAINING – SYSTEMS ANALYSIS

	N	M	RANGE
None Indicated	11	22.1	11 – 35
On-the-Job Training	29	21.8	11 – 35
Formal Course	10	22.8	13 – 27
Both Formal & On-the-Job	9	22.6	10 – 30

JOB TITLE

	N	M	RANGE
Sr. Programmers	9	20.0	11 – 30
Systems Analysts – Previous Programming Title	16	21.0	10 – 35
Systems Analysts – No Previous Programming Title	13	25.5	17 – 35
Others – Mathematicians, Computer Specialists, Operations Research Analysts	17	21.5	11 – 35

FIGURE 6

in systems analysis, but it is not significantly higher. The implication of these results is that at the present time it really does not make much of a difference if an applicant for a systems analysis position had a formal course in systems analysis or was trained on the job.

Looking at the table relating mean score to job title we note that the lowest mean score was for those who said their title was really senior programmer. They were sent to be tested by their supervisors as systems analysts, or as being involved in systems analysis work. Perhaps their mean score of 20.0 reflected the fact that most of them *were* essentially senior programmers. For the other job titles, the surprise was the difference in performance between systems analysts with previous programming titles and systems analysts with no previous programming titles. In the Basic Programming Knowledge Test study, I found just the reverse situation. Those systems analysts who had previous programming experience did better than those who were supposedly pure systems analysts. Here we find that the systems analyst who did not come up the programming route did considerably better on the average (25.5) than the systems analyst who had had a programming title (21.0). If these results are reliable, I would speculate that when a person comes in strictly as systems analyst and not via the programmer route, he has to have an excellent educational background and the necessary abilities to acquire systems analysis proficiency. Many of those who have had previous programming titles have risen through the ranks and been put in systems analysts positions even though they may really not be qualified to handle systems analysis work.

Let me summarize the relationships between the background characteristics of the systems analysts and their scores on the Systems Analysis Test. It should be stressed here that these are inferences from an incomplete analysis on a sample of 60 people in one organization. Further research with other groups will show whether the results here are more general or are specific to this group.

There is no difference in systems analysis proficiency between male and female, and age makes no difference. Education level seems to make a difference when the individual has four or more years of college. The personnel man seeking a systems analyst should give extra weight to a Bachelor's degree, or even better, graduate work. College major does not seem to be an important factor, except that a person from the social sciences, education, or humanities should be looked at for other training received. A math major is probably still desirable, but it is suggested that one take a good look at the math grades and the college attended. A college or formal training course in systems analysis is desirable, but there is no clear indication that the person will do better as a result. Previous programming experience should be looked upon as an advantage only when there are other indications of systems analysis potential or proficiency. The person who has not had previous programming experience can do all right if he has picked up enough programming and computer knowledge for most systems analysis work.

* * *

The second part of this talk concerns the research with a battery of aptitude tests for predicting programmer trainee grades. Some of the same principles that applied to the systems analyst selection also apply to programmer

trainees selection. Let us refer again to the prediction diagram on Figure 1 . . . but this time we substitute training performance for job performance and we try to predict for training performance. If you apply the prediction scheme to small organizations or wherever the selection ratio is low (selection ratio is the ratio of the number of applicants accepted to the total number available), you want to reduce the number of false positives. For large organizations which can afford to train a lot of people and where the selection ratio is high, the objective may be to reduce the false negatives. That is, try to find a better way of selecting those people who would succeed in training but who our present selection procedure predicts would fail. What is needed for better selection in these two situations, I think, is a good prediction battery of aptitude tests and some good indicators related to the background characteristics of the individual. A strategy for applying tests and background indicators to both selection situations will be presented after some details are given on the development of the test battery and the investigation of background characteristics.

The first experimental predictor-test battery included 17 aptitude tests and the Strong Vocational Interest Blank (SVIB). The test battery was tried with a preliminary sample of 112 programming trainees, and as a result of checking the correlations with training grades the 17 aptitude tests were reduced to five. I kept the Strong Interest Inventory Blank, even though the results of this test with the preliminary sample were poor. I could always return interest profile sheets to the people taking the test battery as a kind of reward.

As shown in Figure 7, the final test battery consisted of 7 tests. Included were five from the preliminary battery: Figure Matrix, Logical Reasoning (a syllogisms test), Operations Sequences, Ship Destination (a general reasoning test) and the SVIB. Added to the battery were a mathematics test and a simulated programming aptitude test called the Idiot Helper Test. The first is the ETS Advanced Mathematics Test, which is ordinarily used to place entering college students at the proper math level.

The Idiot Helper Test was specially constructed for this study. It simulates the programming task. The individual never exposed to programming before is shown how to write simple routines to solve arithmetic and simple math problems. The test then requires him to write these little programs for a variety of problems. The results with this particular test in predicting midterm grades was quite phenomenal.

The midterm grades were based on examinations and homework assignments during the first four weeks of training on basic principles. The final grade was based largely (80%) on a special complex problem that was subjectively scored. All the results obtained indicate that the final grade was not too reliable a criterion. Since the midterm grade was based on a grading method similar to those used in other organizations, I have put more reliance on it as a criterion. The correlation of the Idiot Helper Test and midterm grade was .70, a quite high validity coefficient. This .70 correlation stood up on a cross-validation. In doing a multiple correlation of the entire final battery against the midterm criterion using a stepwise regression program (tests are added to the correlation process one at a time) -- the Idiot Helper was first with .70, and the next test, Logical Reasoning, increased the correlation to .73. Adding a third test, Operations Sequence, raised the multiple correlation only slightly, .736. All seven tests yielded a multiple correlation of .75, which is very good,

Correlations* of Final Test Battery Scores with
the Criterion Measures (N = 138)

VARIABLE	MIDTERM	FINAL	MEAN	SD
1. Figure Matrix	.23	.17	9.04	2.58
2. Logical Reasoning	.59	.34	12.24	4.51
3. Operations Sequence	.52	.39	19.60	5.83
4. Ship Destination	.45	.32	37.83	6.00
5. SVIB	.24	.10	20.30	12.62
6. Advanced Mathematics	.55	.35	560.64	108.16
7. Idiot Helper Test	.70	.45	17.55	7.05

Multiple Correlations Computed By Stepwise Regression Program

TESTS IN ORDER OF ENTRY	MIDTERM	TESTS IN ORDER OF ENTRY	FINAL
7	.703	7	.449
2	.730	3	.475
3	.736	4	.487
4	.741	6	.490
1	.746	5	.491
5	.749	1	.491
6	.750	2	.491

**r significant at .05 level = .17; r significant at .01 level = .22.*

FIGURE 7

but it is only an increment of .05 over using the Idiot Helper Test alone.

My recommendation for any selection program in which these tests are considered is that a combination of the Idiot Helper Test and the Logical Reasoning test can be used. The Idiot Helper Test takes an hour and a quarter to administer and the Logical Reasoning test only 15-20 minutes. I think they make a very potent battery for selecting programmer trainees.

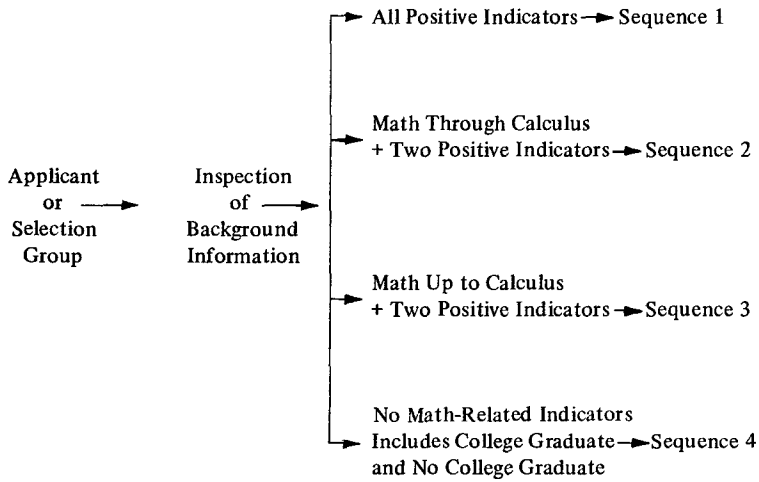
Now, let us go to the background characteristics that I found to be related to training grades. I will not give the full data analysis; that is given in a technical report and will probably be the subject of a future paper. But I will summarize the principal outcomes of the analysis. Look at Figure 8. Shown are the positive indicators, the significant background indicators, of programming success. As I noted previously in the study of systems analysts, one significant indicator was the completion of four years of college (or a Bachelor's degree), or postgraduate work. The preferred college major for this particular study in training success was mathematics or engineering, and the college minor, eingineering and physical or biological science. "Time since last course" was another indicator: if the person had had a formal course within the last 12 months, there was an increased probability that he would do well in training. This did not necessarily apply on the job; in training at least, he retained the study habits that seem to be necessary. Mathematics was a positive indicator as "the best liked" high school subject, and Mathematics or Engineering as the best liked college subject. Under college math difficulty, we asked whether they found math easy, fairly easy, moderately difficult, or quite difficult. We found there was a significant difference for those people who said math came easy or fairly easy. Math level revealed a critical point. If a person had had differential and integral calculus, *i.e.*, one year of calculus or beyond, he seemed to do much better than a person who had only mathematics up to calculus.

The next question is how to best combine the test battery results with the background indicators to select programming trainees. Figure 9 shows the multi-stage sequential strategy that could be used for selection. It is one of several possible strategies that could be adopted, and at this point it has not been validated. There are four selection sequences indicated here which only differ slightly as you go from one sequence to another. But the selection sequence strategy is arranged to save time and money on both the part of the person who is hiring and the person who is being selected. We start by having applicant or selection group fill out an application form that has the positive indicators as key questions. An inspection of the background information then determines which of the four sequences might be followed. If the applicant has all the positive indicators we would put him through Sequence 1. If he has math through calculus plus two positive indicators, we would put him through Sequence 2. Math up to calculus, plus two positive indicators would indicate Sequence 3. A person with no math-related indicators, regardless of whether he is a college graduate or not, would go through Sequence 4.

Figure 10 shows a breakdown of the first two sequences. In these two sequences the applicant has had math through calculus or beyond. In the first sequence, all positive indicators are there: college graduate, math beyond calculus, etc. You give this applicant test battery A, which consists of the Idiot

VARIABLE	SIGNIFICANT INDICATORS
Education Level	4 yrs. college or Bachelor's degree Post-Graduate, Master's degree
College Major	Mathematics Engineering
College Minor	Engineering Physical or Biological Sciences
Time Since Last Course	0 to 12 months
Best-Liked H.S. Subject	Mathematics
Best-Liked College Subject	Mathematics Engineering
College Math Difficulty	Easy and Fairly Easy
Math Level	Differential, Integral Calculus Beyond first year Calculus

FIGURE 8
Background Variables That Have Positive Indicators
of Programmer Training Success



Sequences are Determined by Background Indicators

FIGURE 9
Strategy for Multistage Sequential Selection of Programmer Trainees

Helper Test and Logical Reasoning Test. A combined or weighted score for the two tests is used. If the person is above the cut-off that you have set, you accept him for training. If he is below the cut-off, then you may want to consider another question before you make a decision. Do you have a low or a high selection ratio operating? If you have a low selection ratio, that is, you want to select a very few, then you could reject the applicant outright and not bother further. If you have a high selection ratio, then I would suggest that a probing interview be done to see whether the test results can be explained away or ignored. If the interview outcome is favorable, select the person for training; if it is unfavorable, then reject him.

The second sequence also requires math through calculus, but only two of the positive indicators. This could include college graduate and recent formal training. Test battery A, the same test combination as Sequence 1, is employed. This time, also, if the applicant is above the cut-off he is accepted; however, if he is below the cut-off he is rejected without further interviewing. Modifications of these two sequences may be desirable if this general selection strategy is adopted.

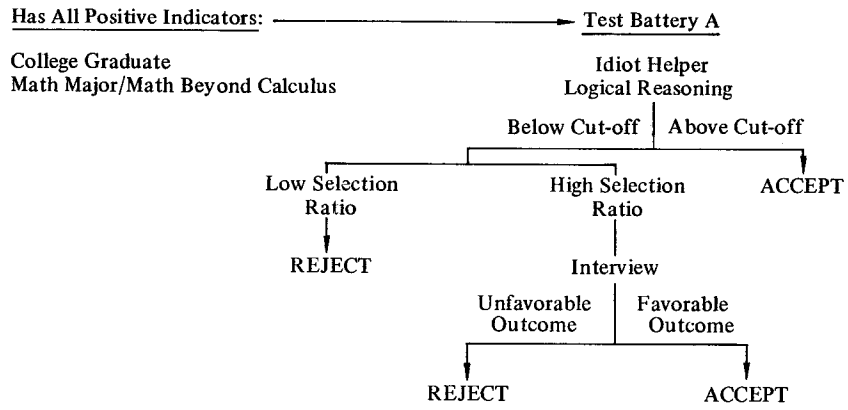
The third and fourth sequences are shown in Figure 11. The applicant has not had math through calculus. However, if he had two positive indicators, i.e., two of: college graduate, engineering major or math major, liked math, math was easy for him, and recent formal training, then you give him test battery B. Test battery B has the Idiot Helper Test, the Logical Reasoning test, and perhaps another aptitude test, Operations Sequence or Ship Destination. In addition, a mathematics test should be used where the applicant has to reach a certain level to show that even though he has not had the background in math he does have the necessary math proficiency required for programmer training. You accept or reject him on the basis of his test results.

Sequence 4 is the lowest level. No math related indicator is found in the person's background. If you have a low selection ratio, that is, you only want to select a very few, then you would reject at this point without wasting time testing. If you have a high selection ratio, i.e., you want to get as many in training as possible (reduce the false negatives) then give him test battery B which is the same as that for Sequence 3.

A very important consideration in the use of the test scores is the cut-off score. Figure 12 shows an expectancy chart which can be used for this purpose. If you look at the chart for midterm grade, the bars show the percentage of people in a certain score category that are in the upper half of the class. For example, of the people who made a score on the Idiot Helper in the score category 25-30 (there are 30 problems in all), over 90% were in the upper half of the class on midterm grade. The odds then for someone making a score between 25 and 30 and being in the top half of his class on midterm grade are nine in ten. The odds for a person scoring between 20 and 24 are seven in ten that he will be in the upper half. It drops to even odds, five in ten, when a person makes a score of 17. For the individual making a score of nine or lower on the test, from our results, the odds were practically zero in ten chances that he would be in the upper half of this training class.

The expectancy chart gives you an indication of the correlation of the Idiot Helper and its utility if we wanted to use that alone. Figure 13 shows the expectancy chart for the Logical Reasoning test. For those who score between

SEQUENCE 1



SEQUENCE 2

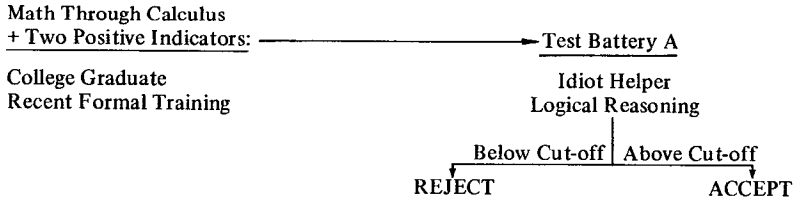
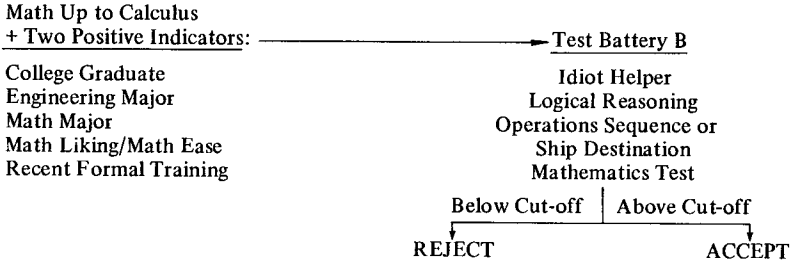


FIGURE 10

SEQUENCE 3



SEQUENCE 4

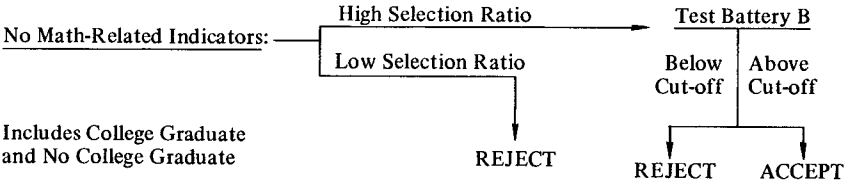
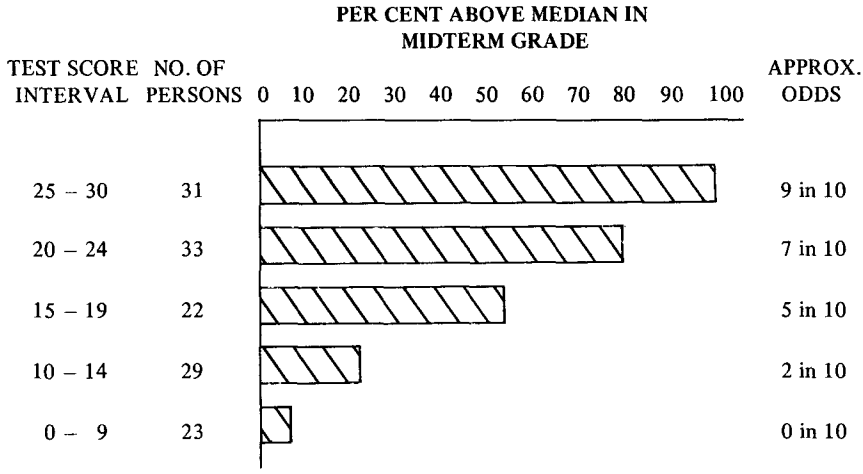
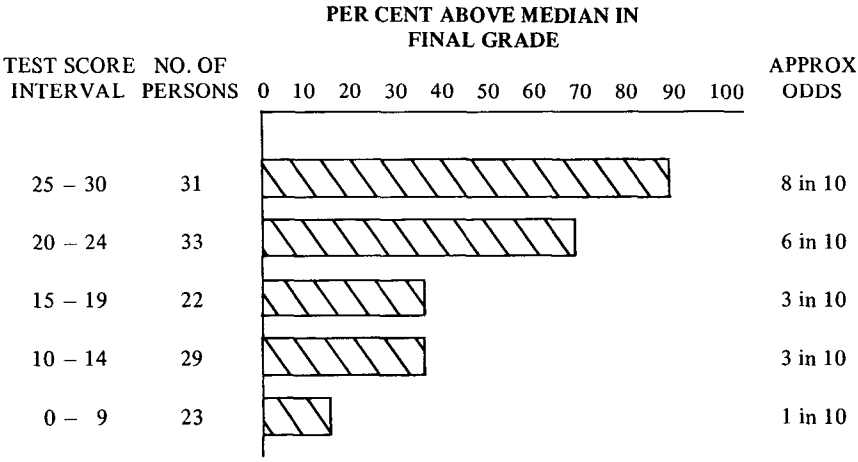


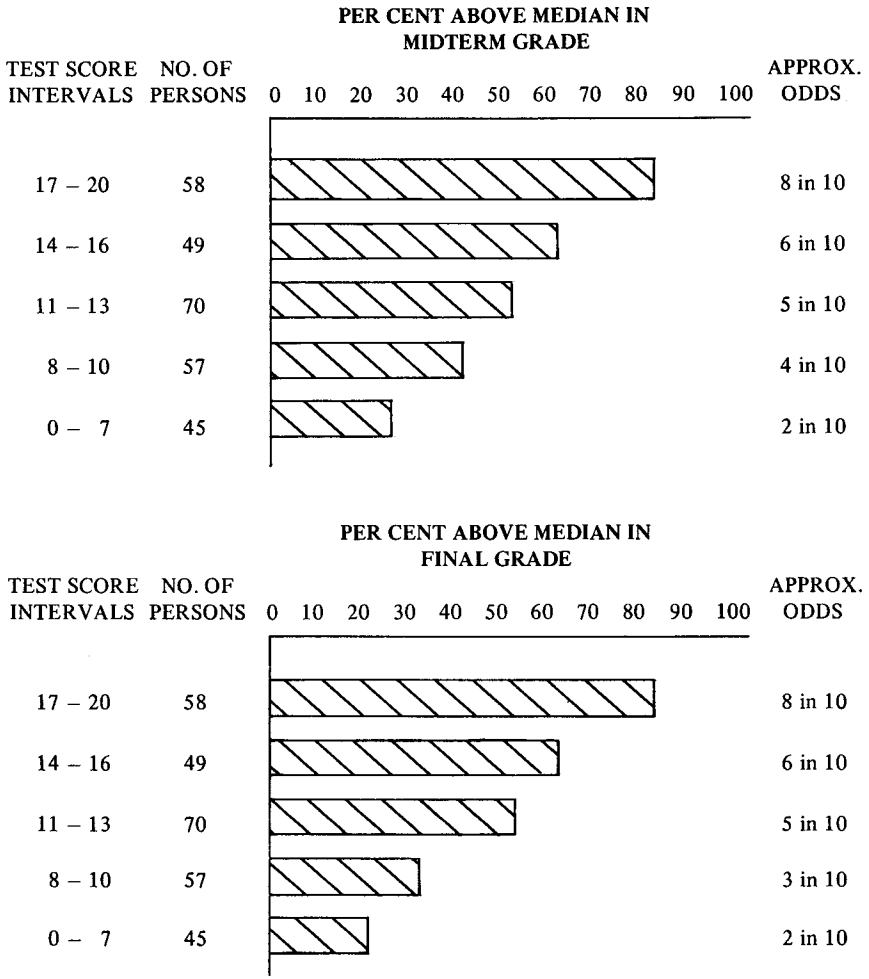
FIGURE 11





Expectancy chart showing chances of being in the top half of programming training class for trainees with different test scores in the Idiot Helper Test.

FIGURE 12

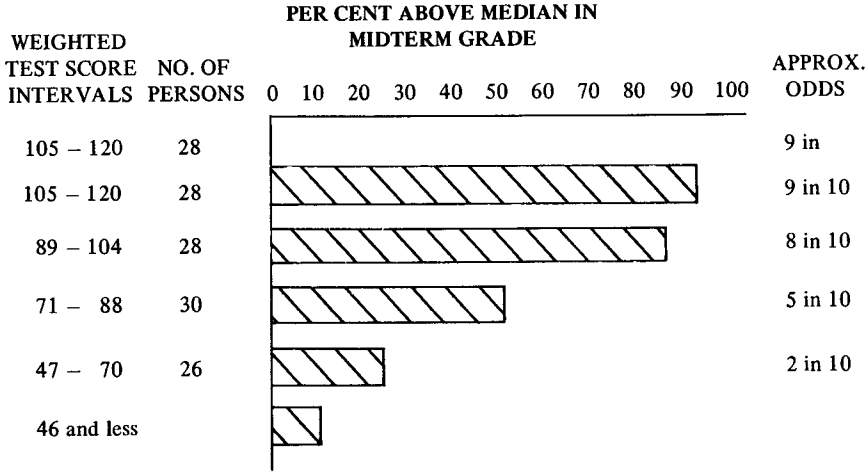


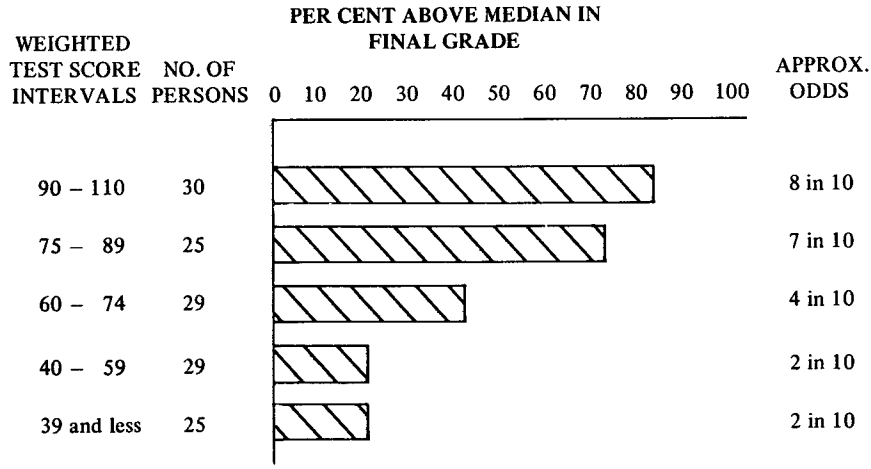
Expectancy chart showing chances of being in the top half of programming training class for trainees with different test scores in the Logical Reasoning Test.

FIGURE 13

17 and 20 on the Logical Reasoning test (there were 20 problems involved), the odds were eight in ten that they would be in the upper half of the class; the next score category drops the odds to six in ten. If we use a weighted combination of the two tests, we can construct an expectancy chart as shown in Figure 14. The Idiot Helper Test score is weighted three times to two times for the Logical Reasoning score. These weights were determined by the regression weights in our multiple regression analysis. For those who score between 105 and 120 in this weighted combination, the odds are nine in ten that they would be above the median midterm grade of the class. The odds drop rather dramatically on anybody with a combined score of 88 or below; the odds are even or less, that he will be in the upper half of the class. Thus we have a very convenient method for establishing a cut-off, in this case a score of 89 or above, to select applicants for programming training.

In summary, I have presented the results of the research with the Systems Analysis Test and a strategy for selecting programmer trainees that involves certain background indicators in conjunction with tests, such as the specially developed Idiot Helper Test. It remains for future applications to corroborate these results.





Formulas for weighting and combining test scores
Midterm: 3 Idiot Helper score + 2 Logical Reasoning score
Final: 3 Idiot Helper score + 1 Logical Reasoning score

Expectancy chart showing chances of being in top half of programming training class using weighted combination of Idiot Helper and Logical Reasoning test scores.

FIGURE 14