CERTIFICATION: A SUGGESTED APPROACH TO ACCEPTANCE

Robert N. Reinstedt
Ray Berger

May, 1973
CERTIFICATION: A SUGGESTED
APPROACH TO ACCEPTANCE *

Certification of programmers is undoubtedly the most pressing personnel problem facing the computing industry today. Unfortunately, there is virtually no evidence at this time to indicate the industry's readiness to come up with a valid certification test. On the other hand, there are unambiguous signs that the day is close at hand when such an instrument must be produced. And all the professional data processing societies seem to firmly agree on one point: if certification is to become a reality, it is far more desirable for the development and "maintenance" to be self-generated than it is to have it imposed and regulated by some outside (i.e. governmental) agency.

If the societies are to prevail in that desire it is not enough to merely initiate a program of certification; it will have to be comprehensive in the sense that both those within and without the industry know precisely what the certification certifies. It must be valid in the sense that one who holds the certificate has been evaluated (either by education, test, experience, knowledge, or a combination thereof) and found to possess the qualifications to satisfactorily perform those tasks which compose the certification requirements. Put simply, certification boils down to: exactly what is the certificate holder expected to do, and can he or she do it?

There are some additional components which demand consideration. One is the temporal aspect. Would certification endure the test of time, or would there have to be periodic

*To be published in Datamation.
updating? Are ethics and standards a necessary part of continued "membership"? For example, if a certificate holder uses his knowledge to embezzle or to rig an election is there to be a process by which he is no longer certified and if so, who does the policing and how?

There are numerous other considerations; however, for purposes of brevity, it seems more productive for this article to limit its scope, and address the two primary problems mentioned earlier--how can an effort achieve the ends of: (1) determining what to certify and (2) given the answer to (1) how to certify in a way which assures a satisfactory degree of validity?

Certification actually must serve two populations: (1) those who have been granted a certificate, and (2) those who seek the services of a certified programmer. For the former, the certificate attests to the attainment of a particular level of expertise. For the latter, certification establishes some minimal and premeasured degree of confidence in the programmer's knowledge and ability to perform. Analogous situations are the certification of doctors and lawyers. Their licenses say to those outside of their field that they have passed through a prescribed course of training and have been adjudged capable of executing their respective tasks with competence. Doctors and lawyers are subject as well to losing their credentials.

Like the medical profession, there are diverse jobs, both in type and degree in "programming". Perhaps some of them will be ultimately subsumed by other disciplines (e.g., a medical diagnostic programmer under the aegis of the AMA) but even if this should be the case, it would seem appropriate that the programming field should want to remain the sponsor of certifying members of its own discipline.
This type of sponsorship will not be possible, however, if there is a single measuring stick used in some binary fashion to divide the world simply into "certified programmers" and "all others". The problem is not even as straightforward as determining degrees of programming competency (e.g., junior programmers, programmer, senior programmer, etc., if this type of a breakdown should be decided). There may have to be columns as well as rows so that a further delineation can be made as to applications of skills (such as medicine, accounting, scientific, general purpose and whatever is decided to be the appropriate breakdown for that part of the matrix).

No matter what is decided as to gradations or applications, the essence of the problem is to find out: "Can a given individual be expected to do a predetermined task with a necessary degree of competence"? The hooker here (and it is not a "happy" one) is in defining that predetermined task. Programmers often say that their job would be made much easier if the customer could articulate clearly what he wanted. Test constructors face the same problem, and the computing world is no more guilty (or innocent) than most other fields in being able to define in sufficient terms what it is that should be tested.

Ideally, the interview, the selection test, salary reviews, evaluating an employee, deciding on continued employment, and certification would all be based on identical criteria. Admittedly, they are different approaches using different methods, but they all ought to be devices designed to extract data to be measured against the description of performance outlined as necessary elements for a given job. This may appear to be blatantly obvious, and in
conversations with various managers there seems to be universal agreement, but how many managers go into (say) an interview situation with any kind of crystallized thoughts about what they specifically want to correlate with expected job performance? This is not to say that in all cases nothing is learned or accomplished, even if it is as simple as getting an indication of compatibility, etc., but if more thought were given to the job requirements, the interview could become significantly more valuable.

Selection tests are similarly misused since they are assumed to correlate with a given job for a given individual in a given company, but are based on group findings, and usually not within the company. (E. L. Thorndike once pointed out that "Some of the tests which parade behind the banner of educational science measure the fact in question about as well as the noise of the thunder measures the voltage of the lightning").

But selection devices are not the villain any more than past attempts at certification instruments are. What has been lacking and must be corrected if certification is ever to be possible, is to establish requirements. If done right, this is a fairly long, exacting and laborious task, but it is both possible and essential if the program is to succeed.

Other difficult selection, training and evaluation problems have been solved. An example can be seen as far back as World War II when the (then) U.S. Air Corps at the outbreak of war was training approximately 300 pilots per year; by war's end over 180,000 pilots had been selected, trained and received their wings. The problem was certainly difficult—that of selecting in very large numbers from a (previously) peacetime population to perform a function which demanded a combination of a high degree of intellectual, physical, and psychological requirements.
A concerted, systematic effort was conducted, based on first establishing the detailed job functions of pilots. Once done, the rest was almost routine. Even at this point, one must still go about it in a scientific manner ("testing" the tests, "evaluating" the evaluation, etc.) but the main design work is done when the "job" is defined.

Certifying programmers must be approached in the same way. There should be a research design which meets the needs of the problem, and that design already exists in part as a result of the recent work that one of the co-authors (Dr. Ray Berger) has been doing for AFIPS. The approach and a proposed road map to certification will be discussed shortly, but a couple of general points still remain to be mentioned before getting to the specifics.

First of all, it would seem to be sheer folly to expect some group (committee) comprised of data processing personnel to devise a certification program to meet the requirements acceptable both internal and external to the computing field. External acceptance covers acceptance in the usual sense of the word, but also in the legal sense.

Second, although one would reasonably enough be interested in a cost-effective solution, arriving at a certification program is probably best served with less emphasis on the developmental costs. Already, various societies are talking in terms of when the program becomes self-supporting. This type of requirement could be counter-productive. No one suggests that thought should not be given to the long- or even short-run economics of the problem, but don't lose the program for "want of a nail".

Finally, there is a very real possibility that the only way to accomplish the task is through universal internal acceptance, which in turn can only result from the combined
effort of the entire computing industry. In this sense, the decision to have all groups represented regarding further certification is an excellent one.

Given that the data processing field is seeking to professionalize itself, it must develop standards that will identify those individuals who have reached a certain level of competence. How these standards are developed, and who sets them is of prime importance. It was indicated earlier that standards, whether technical, professional, or academic, should come from people who are active in the field. It would be a mistake to accept the argument that since management must be served they therefore must set the standards. It is perhaps a large error to let the federal (or state) government be the agency to set standards.

Managers may be able to evaluate the end-products, but are not necessarily schooled in the most effective means of arriving at them. The federal government, for the same reasons that they have abstained from setting standards in the medical, legal, accounting, and other professions, should not have to set the professional standards for programmers and, in turn, the programming profession should not want the government to do so. The computer professional should derive his identity and certification from a society composed of his peers.

The computer profession, among whose basic functions is analysis, has done very little systematic analysis of its own job. Such an analysis would employ the paradigm shown below that starts with a job analysis and ends with an effective set of standards.
PARADIGM FOR ESTABLISHING JOB OR PROFESSIONAL STANDARDS

<table>
<thead>
<tr>
<th>Job Analysis</th>
<th>Job/Professional Standards and Minimums (per job/professional classification)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Job Description</strong></td>
<td><strong>Personal Requirements</strong></td>
</tr>
<tr>
<td>Tasks--Skills</td>
<td>Experience</td>
</tr>
<tr>
<td></td>
<td>Training</td>
</tr>
<tr>
<td></td>
<td>Education</td>
</tr>
<tr>
<td></td>
<td>Abilities</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Job Analysis

Describing the job in terms that can be translated into professional standards is the touchstone of a properly done job analysis. The job descriptors must be detailed, relevant, generalizable beyond a specific organizational setting, and at the same time capable of being reduced to skill requirements. The accumulation of job descriptions put out by many organizations, including governmental agencies are, at best, a useful literature bank.

A universe of task statements should be collected as the basic raw data from which to extract the job descriptions. Statements of tasks, including all activities and responsibilities, can originate in several ways: from observation, from interpreting the user's job requirements, and from employee and manager reports. The recommended procedure is a combination of all these, with emphasis on the employee self-report. The format of the task statements to be
responded to, is preferably a simple declarative sentence starting with a verb to denote some activity of the programmer. Examples can be used but must be carefully checked for their applicability to the entire programming field.

In the AFIPS job analysis project, it became painfully evident that an example of an activity drawn from scientific programming did not clarify the general sense of the statement for a business programmer. Indeed, the differences in tasks according to programming area made it expedient to create subdivisions of task statements for the several programming areas: (e.g., business, scientific, engineering, etc.). However, one should be wary of maintaining the subdivisions as separate job universes since to do so would eliminate the possibility of later establishing a job communality, so that one could speak of a computer programmer with a speciality in (say) business data processing, etc.

When the universe of tasks has been assembled, the process of evaluation, modification, and further evaluation begins. The first evaluations are usually informal. It is amazing how many ways a task statement can be interpreted. As a result, a statement often must be modified several times before it acquires a sufficiently common meaning. This is a costly, time-consuming procedure, but prematurely locking-in on a particular set of statements can reduce the reliability of the final evaluation, ultimately resulting in an even more costly venture.

The informal evaluations should be done by highly rated individuals, actively engaged in the job being analyzed. The well-intentioned (but non-programming) manager with a wealth of experience, not all of which is current, may only introduce ambiguity and delay into the evaluation.
The final evaluation, preferably made on some quantified scale, should be performed by groups that are representative to some degree of the "idealized" professional population. When the final goal of the job analysis is the determination of professional standards, then the criterion group making the evaluation of task and skill requirements must itself be in the professional mold. A job analysis based on the evaluations of a thoroughly representative sample of the industry as a whole is too difficult to achieve at the present time. For one thing, there is no accurate census of the programmer population on which to determine a representative sample. The best procedure is to carefully select your criterion group first, and then to sample, as well as possible, within defined strata of the general programmer population.

The second part of the job description is the building of a skills-and-techniques bridge between the job tasks and the people who perform the tasks. What specific skills do the tasks require? How are the skills identified, and their importance determined? These questions are answered by following the same procedure used for defining the set of tasks: assemble a universe of skill statements from a variety of sources and evaluate the extent of their involvement in the programmer's repertoire.

When the job has been described in terms of the task requirements and the subject-matter skill requirements, one can then move with scientific (and legal) force to the personal requirements. The individual can only perform competently when he or she has the training, experience, and abilities that have been shown to be directly related to the task and skill requirements.
One way to determine just how much training and experience are required is to survey the kinds and amounts of education and training computer people have. Another way is to ask training directors what the training requirements are for the described tasks and skills.

Surveys have revealed a great variety of training and education patterns for programmers and systems analysts. Self-study and on-the-job training are the most frequent ways in which knowledge and skills are acquired for performing programming tasks. Training programs themselves vary widely in objectives and methods, and there are even wide differences among training programs that share the same goals. Fair standards for training and educational requirements will obviously be difficult to determine, but the arbitrariness may be reduced by careful reference to information yielded by biographical survey and task/skill descriptors.

The experience variable is a little less problematic; although the relationship between experience and proficiency may be non-linear, it is, nonetheless, definitely positive. Job analysis data correlated against experience would expect to uncover the minimal experience required for different levels of programming jobs. These requirements can be translated into standards for minimum experience for professional standing. For example, it might be ascertained that a minimum of one year of full-time programming is one requisite to become a candidate for certification, and that three years of experience may be the minimum to qualify for advanced standing. In any event, the minimum experience requirements, like the job descriptors, should be based on a job analysis rather than on some committee's arbitrary decision.
It is essential that the determination of all personal requirements, training, experience, and abilities, be started with a database of important tasks and skills. A survey of the background characteristics of individuals in the field, the judgments of panels of experts, and the psychological insights of a professional experienced in job analysis can indicate the parameters of the personal correlates of on-the-job proficiency; recommendations for professional standards follow naturally.

Determining Professional Standards

When a job analysis is complete, then the establishment of the standards for professional membership becomes appropriate. The education, training, and experience requirements should emerge in large measure from the job analysis. A final requisite, perhaps the ultimate objective of conducting a scientific job analysis, has now to be satisfied: the construction of a qualifying examination.

Such an examination (there would more than likely be several in order to cover different levels and areas) must be based almost entirely on the job analysis. Any exam chosen for interim use until a job analysis is completed may do more harm to a certification program than no exam at all. A test developed by a special interest group may tend to reflect only concerns of that group irrespective of whether that is the motive of the group or not. A case in point: an exam presently in use was originally constructed to certify competence in several areas for which study was required. The exam, because of its high correlation with recency of education, has had an inverse relationship with amount of experience and thus gives an advantage to those who have just completed training (or schooling).
Reports on the examination results indicate that on the average, those with greater experience are disadvantaged in their scores. Moreover, the exam covers areas not directly related to the technology of programming. Such an exam, however suitable for its original purpose, would not seem to be an appropriate instrument for certification even as an interim measure. Any future achievement or professional qualification exams, to be useful, must be constructed according to acceptable psychometric standards.

In summary, the case has been made here for a scientific approach to a certification program, acceptable internally and externally to the profession. The design is accomplished using careful job analysis for defined subdivisions of the programming field. Starting with the job description: the job is structured in terms of important task elements and skill elements, and these elements will indicate the personal requirements.

This information then provides the means for establishing professional standards and qualifying examinations. Fortunately, the first (and perhaps most difficult) stage of the above paradigm is nearing completion, specifically, the job analysis project sponsored by AFIPS. The subsequent stages of the paradigm remain to be carried out, but if done as described, the end result will be a better and more lasting foundation for professional standards and the establishment of a certification program acceptable both to those in the field and those the field serves.