TITLE: "HIRING THE RIGHT PROGRAMMER"

by Tim Bryce
Managing Director
M. Bryce & Associates (MBA)
P.O. Box 1637
Palm Harbor, FL 34682-1637
United States
Tel: 727/786-4567
E-Mail: timb001@phmainstreet.com
Yahoo! IM: littleleaguerng
WWW: http://www.phmainstreet.com/mba/
Since 1971: *"Software for the finest computer - the Mind"*

"A resume is either an accurate description of a person's capabilities or demonstrates how well someone can write fiction."
- Bryce's Law

INTRODUCTION

Finding a good programmer can be a difficult task. Often times you will come across a candidate who interviews well and appears to have impressive credentials, yet you discover too late that he is simply not as proficient as you thought he was. Now you have someone you will either have to eventually eliminate or invest considerable money in to bring him up to speed (or both). What to do? True, you should probably improve your interviewing skills and learn to read between the lines of a resume, but there are a few other things you can do.

Basically, there are three things you, as a manager, want to know about a new employee; his background (job history), his knowledge, and how well he will adapt to your corporate culture. His background should be revealed by the interview, his resume, and any references he might have, but determining his knowledge and adaptability to the corporate culture is a little trickier.

CORPORATE CULTURE

I have discussed the importance of corporate culture many times in the past; in particular, see:

No. 28-*"Understanding Corporate Culture"* - June 13, 2005
http://www.phmainstreet.com/mba/ss050613.pdf

Basically, in order for any employee to properly function and succeed, it is imperative that he is able to adapt to the corporate culture. If not, the culture will reject him and the employee will become an outcast. Before we can evaluate the employee's adaptability though, we should understand our own culture first. For example:

- What are the corporate ethics? Do you value honesty and integrity or are you a politically charged environment with considerable backbiting, finger pointing, piracy, and other questionable office tactics?
- Do you commonly seek "quick and dirty" solutions or do you operate more as skilled craftsmen?
- How rigid are your operating policies, e.g., dress codes, hours of operations, conduct, etc.?
- What are interpersonal relations/communications like in your office; e.g., speech, form of address, decorum, cooperation, etc.?
- What form of management do you practice; dictatorial with considerable supervision of do you empower your employees to make decisions?

Ascertaining a candidate's adaptability will be primarily based on your observations of the candidate during the interview.

SKILLS & PROFICIENCIES

A candidate's resume will say one thing, but you may be looking for something else. As part of the interview, you may want to ask the candidate to complete a Skills Assessment which lists the skills pertaining to your area and his level of competency (proficiency). The following is a sample Skills Assessment we have used for gathering background information on programmer candidates. It is certainly not universal and should be tailored to your organization's needs. Regardless, after the candidate has completed the Skills Assessment, it should be compared against his resume in order to look for discrepancies.

SKILLS ASSESSMENT

Please indicate your level of expertise in the following areas:

1. Please circle your proficiency with the following PROGRAMMING LANGUAGES:

ASSEMBLER: Expert - Competent - Novice - Don't Know

BASIC: Expert - Competent - Novice - Don't Know

C++: Expert - Competent - Novice - Don't Know

COBOL: Expert - Competent - Novice - Don't Know

(continued on page 2)

(continued from page 1)

JAVA: Expert - Competent - Novice - Don't Know

SQL: Expert - Competent - Novice - Don't Know

(allow the candidate to add other languages not listed)

2. Please circle your proficiency with the following COMPUTER CONTROL & TAG LANGUAGES:

DOS BAT: Expert - Competent - Novice - Don't Know

HLP: Expert - Competent - Novice - Don't Know

HTML: Expert - Competent - Novice - Don't Know

IBM JCL: Expert - Competent - Novice - Don't Know

REXX: Expert - Competent - Novice - Don't Know

PostScript: Expert - Competent - Novice - Don't Know

XML: Expert - Competent - Novice - Don't Know

(allow the candidate to add other languages not listed)

3. Please circle your proficiency with the following COMPUTER OPERATING SYSTEMS:

DOS: Expert - Competent - Novice - Don't Know

Linux: Expert - Competent - Novice - Don't Know

MVS: Expert - Competent - Novice - Don't Know

OS/2: Expert - Competent - Novice - Don't Know

VMS: Expert - Competent - Novice - Don't Know

Windows 2000: Expert - Competent - Novice - Don't Know

Windows XP: Expert - Competent - Novice - Don't Know

UNIX: Expert - Competent - Novice - Don't Know

(allow the candidate to add other operating systems not listed)

4. Please circle your proficiency with the following types of DBMS ARCHITECTURES:

HIERARCHICAL: Expert-Competent-Novice-Don't Know

CODASYL NETWORK: Expert - Competent - Novice - Don't Know

RELATIONAL: Expert - Competent - Novice - Don't Know

OBJECT ORIENTED: Expert - Competent - Novice - Don't Know

5. Please circle your proficiency with Double Byte Character Set (DBCS) technology:

DBCS: Expert - Competent - Novice - Don't Know

6. Please circle your proficiency with the following TOOLS:

Establish a list of in-house tools used; some suggestions:

4th Generation Languages
CASE Tools
Data Dictionaries/Repositories
DBMS Packages
GUI/Screen Design tools
Program Generators
Programmer Workbenches
Project Management Tools
Prototyping Aids
Report Writers
Test/Debugging Aids
TP Monitors (ISPF, ROSCOE, TSO)
Visual Programming Tools
Web Design Tools
(allow the candidate to add other tools not listed)

KNOWLEDGE

Now, more pointedly, you need to know if the candidate truly knows how to program or not. College degrees, certificates, and participation in trade groups are important, but you need to convince yourself the person has substance as opposed to facade. Samples of work are useful, but then again, are you sure the person actually produced it? We have always found it useful to provide a simple test for the person to verify he knows what he is talking about. He can either substantiate his knowledge through a test or he cannot.

The following is a sample questionnaire we used over the years to evaluate a programming candidate's credibility. It was designed to evaluate both general and specific areas of concern for us. Again, it is by no means a universally applicable test but, instead, gives you an idea of how to construct your own questionnaire. Most of it

(continued on page 3)

(continued from page 2)

requires freeform answers and takes about thirty minutes to complete, but we have found it to be time well spent.

SOFTWARE ENGINEERING QUESTIONNAIRE

NAME: _____

DATE: / /

TO APPLICANT:

The purpose of this questionnaire is to review your Software Engineering knowledge. Please be brief and concise in your answers.

1. What is the normal sequence of tasks you follow to develop a PROGRAM?

This freeform question is aimed at reviewing the respondent's approach to problem solving and design.

TYPICAL RESPONSE:
Analyze, Design, Program, Test/Debug, Review

2. What program design techniques are you familiar with or use?

This freeform question is used to determine the types of techniques the programmer is familiar with.

TYPICAL RESPONSE:
Structured Programming
Object Oriented Programming
(Graphics techniques)

3. What are the differences between Structured Programming and Object Oriented Programming? What are the benefits of each?

TYPICAL RESPONSE:
Structured Programming represents a "top-down" modular approach for dividing software into smaller, more manageable, building blocks (to divide and conquer). This is useful for splitting up complex programs, and for maintenance.

Object Oriented Programming breaks programs into self-sufficient software that can be easily combined with other objects. An object can represent just about anything in real life. Both the data structure and the processing is bound together within the object.

4. What are the normal steps for testing a program?

This freeform question is intended to review the person's testing techniques.

TYPICAL RESPONSE:

A. The major modules within the program should be tested individually using only valid data, e.g., no error conditions should be tested.

B. Independent error conditions should be tested one at a time (single errors).

C. Error conditions should be tested together to see whether one error condition has an effect upon another (contingencies).

D. Conditions that depend on volume can be tested.

5. On a separate piece of paper, draw a structure chart for a program performing the following functions. (The purpose is to review your structured design knowledge). To avoid details, limit the structure chart to only 3 or 4 levels.

Basically, the program performs simplified TIME REPORTING functions for a company employee interactively. It accesses two files: EMPLOYEE FILE and TIME RECORD FILE.

The EMPLOYEE FILE contains records which include EMPLOYEE NO., EMPLOYEE NAME, a list of CURRENT ASSIGNMENT CODES and NAMES, and NUMBER OF THE TOTAL CURRENT ASSIGNMENTS.

The TIME RECORD FILE contains records which include EMPLOYEE NO., DATE, ASSIGNMENT CODE, ACTUAL HOURS FOR THE ASSIGNMENT, ESTIMATED HOURS REMAINING TO COMPLETE THE ASSIGNMENT.

The program first requests the user to enter his EMPLOYEE NUMBER from the keyboard. Then it displays EMPLOYEE NUMBER, EMPLOYEE NAME, CURRENT DATE, and a list of CURRENT ASSIGNMENTS. For each current assignment, it should display the ASSIGNMENT CODE, ASSIGNMENT NAME, a blank field for ACTUAL HOURS, and ESTIMATE REMAINING for the assignment (if there is no ESTIMATE REMAINING reported yet, a blank field is displayed).

(continued on page 4)

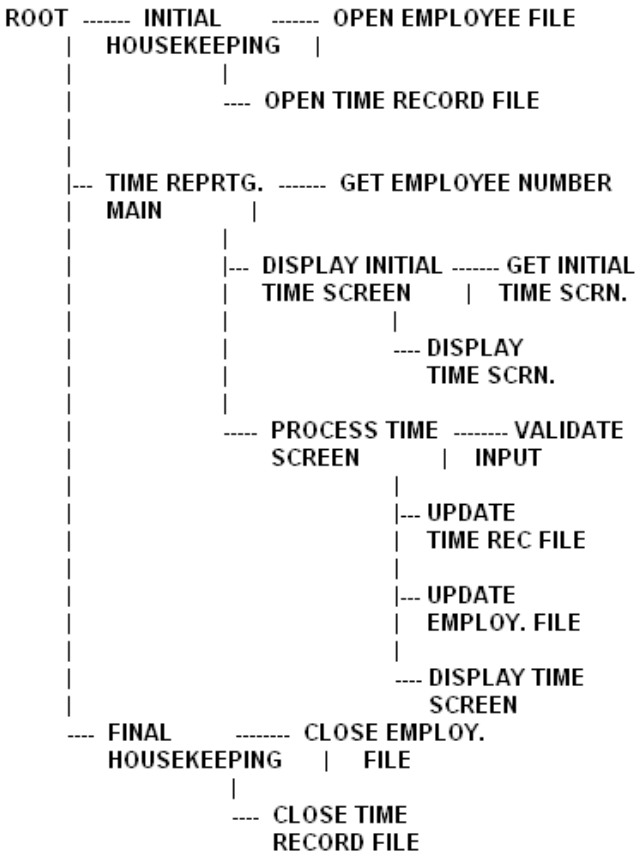
(continued from page 3)

The user can only report for one assignment at a time (either for ACTUAL HOURS or ESTIMATE REMAINING but not for both). The program redisplays the screen and updates the hours on the screen and updates the hours on the screen every time the user reports the hours and hits ENTER.

A time record is generated when the user reports hours against an assignment. If user reports 0 hours for ESTIMATE REMAINING for an assignment, the EMPLOYEE RECORD must be updated in EMPLOYEE FILE.

The program terminates when the user types EXIT and hits ENTER.

SAMPLE ANSWER:



DATA STRUCTURES

In the following questions, use the 'C' language to produce an example. If you do not know 'C', use COBOL or another language you are more familiar with:

6. Code an example describing the data structure of a multidimensional TABLE (or ARRAY) and describe how to access its elements.

C EXAMPLE:

```
int.  Table [10] [10]; /* 2 dimensional array */

      for (i=0; ; <10; ++i)
      for (j=0; ; <10; ++i)
          printf("%d/n",table[i][j]);
```

The above example will print out all data on a separate line. The array can be 3, 4, 5 dimensions. Access to individual elements can also be done by
printf("%d\n",table[5][5];

COBOL EXAMPLE:

DATA:

```
01  THREE-DIMEN-TABLE.
   05  TWO-DIMEN-TABLE       OCCURS 100 TIMES
                             INDEX BY TBL-INDX-1.
      10  ONE-DIMEN-TABLE     OCCURS 100 TIMES
                              INDEX BY TBL-INDX-2.
         15  TABLE-ELEMT      OCCURS 100 TIMES
                              INDEX BY TBL-INDX-3
                                PIC X(8).
```

PROCEDURE:

```
SET TBL-INDX-1 TO 1.
SET TBL-INDX-2 TO 1.
SET TBL-INDX-3 TO 1.

MOVE THREE-DIMEN-TABLE (TBL-INDX-1 TBL-INDX-
2 TBL-INDX-3)
    TO FIRST-TBL-ELEMT.
```

7. STACKS:

A. Define a STACK:

TYPICAL RESPONSE:
An ordered collection of items into which new items may be inserted and from which items may be deleted at one end, called the "top" of the stack.

(continued on page 5)

(continued from page 4)

B. Code an example of its data structure:

C EXAMPLE:

```
struct stack (
        int num;
        struct stack *next  );
top=push (num,top) /* returns address of last record */
```

COBOL EXAMPLE:

```
01  STACK.
    05  STACK-TOP          PIC 9(3).
    05  STACK-ITEM         OCCURS 100 TIMES  PIC X(8).
```

8. What is a RECURSIVE function?

TYPICAL RESPONSE:
A function (or procedure) called or performed directly or indirectly by itself.

A. Can you perform a recursive call in COBOL?

(NO)

9. QUEUE:

A. Define a QUEUE:

An ordered collection of items from which many items may be deleted at one end (called the "front" of the queue) and into which items may be inserted at the other end (called the "rear" of the queue).

B. Code an example of its data structure:

C EXAMPLE:

```
struct queue (
          int num; struct queue * next;)
top = getfirst (top)
last = getlast (num,last)
```

COBOL EXAMPLE:

```
01  QUEUE.
    05  QUEUE-FRONT       PIC 9(3).
    05  QUEUE-END         PIC 9(3).
    05  QUEUE-ITEM        OCCURS 100 TIMES  PIC X(8).
```

10. POINTERS & LINKED LISTS:

A. What is a POINTER?

TYPICAL RESPONSE:
A pointer is a data item whose content is the address of another data item.

B. Define a LINKED LIST:

TYPICAL RESPONSE:
A collection of items where each item contained within itself the address of the next item.

C. Code an example of its data structure and describe the logic to create a LINKED LIST:

C EXAMPLE:

```
struct |node (
       int main;
       struct |mode *next;
             )
```

Linked lists are sequentially accessed but they may not be sequential in memory. In above example, main has integer data while next would be set to point to the next link in the list.

COBOL EXAMPLE:

DATA:

```
01  POINTER                PIC 9(3).
*
01  LINK-LIST.
    05  LINK-LIST-ITEM       OCCURS 100 TIMES.
        10  ITEM-INFO        PIC X(8).
        10  NEXT-ITEM        PIC 9(3).
```

PROCEDURE:

```
    MOVE ZERO TO POINTER
      PERFORM 100-CREAT-LINKED-LIST   UNTIL
(POINTER = 100).
*
100-CREAT-LINKED-LIST.
*
    ADD 1 TO POINTER.
    COMPUTE NEXT-ITEM (POINTER) = POINTER + 1.
*
105-DONE-LINKED-LIST.
```

(continued on page 6)

(continued from page 5)

11. Define a BINARY TREE:

TYPICAL RESPONSE:
A finite set of elements that is either empty or contains a single element called the "root" of the tree and whose remaining elements are partitioned into two disjointed subsets, each of which is a binary tree.

12. What is a GRAPH?

TYPICAL RESPONSE:
Consists of a set of nodes (or vertices) and a set or arcs. Each arc in a graph is specified by a pair of nodes.

13. HASHING ALGORITHMS:

A. Define the function of a HASHING ALGORITHM:

TYPICAL RESPONSE:
To transform a key into a table index.

B. What would a good HASHING ALGORITHM achieve?

TYPICAL RESPONSE:
Produce as few hashing clashes (collisions) as possible, e.g., it should spread the key uniformly over the possible table indices.

OPERATING SYSTEM CONCEPTS

14. Explain the following concepts:

A. VIRTUAL MEMORY

TYPICAL RESPONSE:
A computer operating system which allows users to have larger address space than its actual memory space. Usually disk storage is used for the extension.

B. STATIC LINKING and DYNAMIC LINKING

TYPICAL RESPONSE:
All external references in a program are resolved at link time.

All external references in a program are resolved at execution time.

C. DEADLOCK

TYPICAL RESPONSE:

When several processes are executed concurrently, two or more processes are permanently blocked waiting for resources.

CONCLUSION

The examples used in this bulletin are applicable for programmers only. However, you may want to devise comparable Skills Assessments and tests for Systems Analysts, Project Managers, and Data Base personnel.

Testing is an invaluable means for determining if candidate qualifications as stated in resumes are legitimate. Basically, it helps differentiate between facade and substance. Some Human Resource departments frown on such testing, others welcome it. For programmers, I consider it vital. Frankly, you have better things to do than waste time on someone who is not truly qualified for the position. Remember, *"An ounce of prevention is worth a pound of cure."*

END

About the Author

Tim Bryce is the Managing Director of M. Bryce & Associates (MBA) of Palm Harbor, Florida and has 30 years of experience in the field of Information Resource Management (IRM). He is available for training and consulting on an international basis.

"PRIDE" Special Subject Bulletins can be found at:
http://www.phmainstreet.com/mba/mbass.htm

They are also available through the "PRIDE Methodologies for IRM Discussion Group" at:
http://groups.yahoo.com/group/mbapride/

You are welcome to join this group if you are so inclined.

The "Management Visions" Internet audio broadcast is available at:
http://www.phmainstreet.com/mba/mv.htm

Also, be sure to read Tim's Blog at:
http://blogs.ittoolbox.com/pm/irm/

"PRIDE" is the registered trademark of M. Bryce & Associates (MBA) and can be found on the Internet at:
http://www.phmainstreet.com/mba/pride/