TITLE: "THE RATIO OF ANALYSTS TO PROGRAMMERS"

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
WWW: http://www.phmainstreet.com/mba/
Since 1971: *"Software for the finest computer - the Mind"*

"Good specifications will improve programmer productivity far better than any programming tool or technique."
- Bryce's Law

INTRODUCTION

In terms of systems development, during the 1960's and early 1970's you were either a Systems Analyst or a Programmer. Period. At the time, there were substantially more analysts than programmers (at least a 2:1 ratio). This was due, in part, to the fact that computing was just coming into its own in the corporate world and there were still people around who could look at systems in its entirety. However, there was a screaming need for people to program computers and, as such, this became the boom years of programming. If you knew COBOL, Fortran, or PL/1 you could just about right your own ticket. Salaries were good, and you could intimidate your employer simply by what you knew (you had to commit something like murder to get fired). The emphasis on programming became so great that authors rushed out voluminous books to increase programmer productivity, hence the birth of the Structured Programming movement of the late 1970's, which was followed shortly thereafter by the CASE movement (Computer Aided Software Engineering).

While programming was growing in stature, Systems Analysis was in sharp decline. Trade groups such as the Association for Systems Management (ASM) saw their membership dwindle to nothing and were forced to close their doors. The last of the old Systems Analysts either retired or were put out to pasture by corporations in the 1980's. New job titles emerged, such as Software Engineer and Analyst/Programmer. This latter title is a bit of a misnomer as the emphasis was on programming and not systems analysis.

Although programming excelled, a noticeable void began to appear in terms of people who could see systems in its totality. Writing a good program is one thing, getting it to interface with other programs to form a whole system is something entirely different. By the turn of the century, the industry started to talk about such things as "Enterprise Architecture," "Business Processes," "Business Rules," "Business Analysis," etc. Further, new conferences, trade groups, and job titles began to emerge. Today, programmers are considered a dime a dozen and the stock of a true analyst is on the rise.

All of this is indicative of the industry trying to reinvent systems theory. In reality there is nothing new here as systems analysis is systems analysis. But as companies implement these concepts and job titles again, they are a bit uncertain as to where they fit in and their relationship to other Information Technology functions.

CHARACTERISTICS

A Systems Analyst goes by many names these days; e.g., Business Analyst, Enterprise Architect, Systems Engineer (my personal preference), etc. Nonetheless, we are talking about a person whose mission is to study the information requirements of a business and design a total system solution to satisfy them. Further, the analyst is responsible for specifying the software requirements and, as such, is considered the intermediary with the programming staff. The personal characteristics of the analyst are considerably different than the programmer. Whereas the programmer tends to be more introverted and focused on technology, the analyst tends to be more business oriented and extroverted. Analysts possess good communications skills (verbal and written) to effectively work with both the end-users and the programming staff. They know how to conduct an interview and make a presentation (salesmanship). In addition, they tend to look at the bigger picture as opposed to just a portion of it, and possess an entrepreneurial spirit.

The analyst understands the business problems of the end-user and is intimate with the operation of the user's department. In other words, the analyst can comfortably walk in the shoes of the end-user. If they are doing their job properly, analysts make excellent candidates to assume responsibility in the management hierarchy. But because analysts were in decline for so many years, this hasn't happened for quite some time. The last time I heard of a systems analyst graduating to a major management position was Dan Boone who was made President and COO of Armco Steel in the late 1970's.

If systems analysis is performed correctly, programmer productivity should improve as analysts should be providing good specifications for application assignments.

(continued on page 2)

(continued from page 1)

In the absence of systems analysts, considerable time is lost by the programmer who has to second-guess what the end-user wants. Inevitably, this leads to rewriting software over and over again. Good data and processing specs, as provided by a systems analyst, will improve programmer productivity far better than any programming tool or technique. This means programmers are the beneficiaries of good systems analysis.

This brings up an interesting point, what should be the ratio of Systems Analysts to Programmers in a development organization? Frankly, I believe there should be twice as many analysts than programmers. By concentrating on the upfront work, programming is simplified. Let me illustrate the point by using the following triangles (below) representing the total amount of effort in a project (as an aside, I picked this up from my customers in Japan who share my opinion):

The triangle on the left represents the traditional approach whereby there is twice the number of programmers to systems analysts. Under this approach, considerably more time is spent producing software to satisfy poorly defined requirements. The Japanese point out the bottom of the triangle is actually bottomless as it means more time is needed to complete a project. Compare it to the triangle on the right where there are twice as many analysts to programmers. Under this scenario, more time is spent analyzing the problem, designing the system, and producing better programming specs. Consequently, the programmers do not have to second-guess what has to be performed and can go about their work more productively.

The problem with the diagram on the right though is that Systems Analysis is considered to be somewhat of a nebulous concept to management. Programming, on the other hand, is more tangible and easier for people to grasp; you are either writing code and producing a program or you are not. Therefore, the mindset in management is that you are not being productive unless you are coding, hence the inclination to shortcut systems analysis. This is a key reason why Systems Analysis collapsed in the 1980's. And this is why it is necessary to provide training so management appreciates the need for systems analysis. Frankly, I have found management can be very supportive if it is presented to them properly.

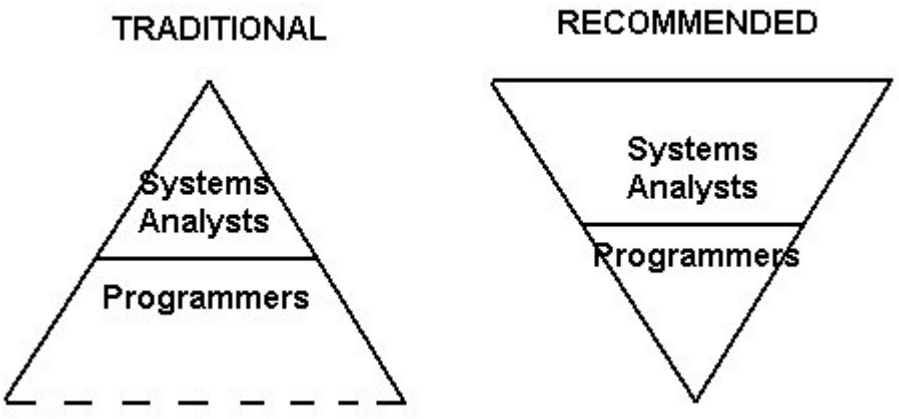
CONCLUSION

Whether you call them Systems Analysts, Business Analysts, Systems Engineers, or Enterprise Architects, it is very encouraging to see this vital function being reintroduced to companies. As far as I am concerned, it was inevitable. I guess companies finally figured out you cannot satisfy your systems problems simply by using better programming tools and techniques.

We are also beginning to see the resurgence of related trade groups to replace such groups as the Association for Systems Management (ASM), for example:

The International Institute of Business Analysis
http://iiba.com/

(continued on page 3)



(continued from page 2)

The IIBA appears to be picking up where ASM left off, including certification. Whereas ASM developed and offered the Certified Systems Professional (CSP) certification years ago, IIBA wants to create something similar.

All of this is indicative of how the industry is trying to reinvent systems theory. Whereas such systems work was well known up until the 1980's it was forgotten over the last twenty years due to the emphasis on programming. Fortunately, companies have finally realized the importance of systems work and are trying to get their houses in order. I guess what goes around, comes around.

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:

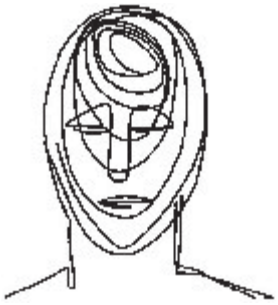
http://www.phmainstreet.com/mba/mv.htm

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/



Since 1971: *"Software for the finest computer - the Mind"*