TITLE: “SYSTEMS: WHAT'S IN A WORD?”

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

"Do not try to apply a band-aid when a tourniquet is required to stop the bleeding."
- Bryce's Law

INTRODUCTION

What's in a word? Plenty. One of the most troublesome words used in our field is the word "System." What's so troublesome? Doesn't everyone know what an Information System is? Maybe the obvious isn't as obvious as you might think.

A DEFINITION

An Information System is a prescribed set of processes operating routinely to produce information for users to fulfill business actions and decisions. Information Systems are used to pay employees, manage finances, manufacture products, monitor and control production, forecast trends, process customer orders, etc. These are all excellent examples of how systems support information requirements.

But what is a Computer System? What does it produce? Information about computers? Or, are we merely trying to describe how a system was implemented? I beg the issue as to what is more important; the system or how it was implemented? To many in the field, implementation is more important than what is being implemented. Many also believe the word "application" is synonymous with the word "system." Think about it. What does the word "application" really mean? It has come to mean the application of the computer to some task within an overall Information System. It could be just a small portion of the system, perhaps just a single program.

CONFUSION IN THE FIELD

Recently we were involved with a large aerospace and electronics conglomerate who was trying to develop a complete business systems architecture. Obviously, this requires a well thought out systems oriented methodology and IRM Repository to define and maintain the information resources to be built; see "Managing Design Complexity" at:

"PRIDE" Special Subject Bulletin No. 27 - Feb 07, 2005
http://www.phmainstreet.com/mba/ss050207.pdf

Unfortunately, the company's I.T. organization had been engrossed in the use of CASE tools and had come to the erroneous conclusion that software engineering was no different than systems engineering. Because of this, they convinced management (perhaps "browbeat" is a better adjective) that this programming approach, coupled with a consultant conversant with the technology, will be able to achieve their business systems goal. This is a classic example of a tool oriented approach as opposed to a management oriented approach to solving systems problems. What is more disturbing is this has become more prevalent in today's corporate world.

Software is obviously the opposite of "hardware". Software is machine processable instructions permitting the hardware to perform specific functions. Software has the same relationship to systems as robots have to a manufacturing process. Even if the robot performs properly or a program executes correctly, it does not necessarily mean you are producing anything worthwhile. What is more important in this situation is whether the logical system design is correct or not. We have always subscribed to the following formulas:

Good Systems Design + Good Programming = Great Systems
Good Systems Design + Bad Programming = Good Systems
Bad Systems Design + Good Programming = Bad Systems
Bad Systems Design + Bad Programming = Chaos

The physical implementation can be less than perfect and you can still produce good results; it may not be the most elegant technologically, but it does solve the business problem adequately. Vise versa is not true. As I have described before, being "effective" (doing the right things) is clearly more important than being efficient (doing things right). Also remember that a system can be implemented manually and, in many cases, manual implementation of certain business processes is still a cost effective alternative.

The intent of CASE tools is to build "software," not "systems." They are particularly well suited for organizing programming specifications and producing the software. CASE tools include diagramming aids, screen prototyping aids, program generators (whether 3GL source code, a

(continued on page 2)

(continued from page 1)

4GL, or a report writer), and test/debugging aids. In "PRIDE"-ISEM terms, these tools are used throughout Phases 4-II, 5, and 6, the software engineering phases of development. But make no mistake about it, they do not alleviate the need for the important up-front systems work. In fact, most CASE tools acknowledge and support the need for the proper front-end systems design. However, when you're obsessed with a particular technology, as the aerospace and electronics company was, you tend to lose objectivity and begin to believe the tool can work miracles.

"Systems" and "software" are as dissimilar as "information" and "data" (which represents another point of mass confusion). It is like comparing apples with oranges. As such, CASE tools are an "ineffective" way to perform true systems work and a misapplication of the tool. We have no problem with their use (in fact we applaud the developments in this area) but we balk at their use where they are not suited.

TELLTALE SIGNS

So how do you know if your I.T. department is overtly software oriented? Look for these simple signs:

• THERE IS A 2:1 RATIO OF SOFTWARE ENGINEERS TO SYSTEMS ENGINEERS

This is a classic sign. In fact, my ratio may be too modest. It is not uncommon to find shops where there is a 3:1, 4:1, or 5:1 ratio; or, even worse, no systems people at all.

What this means is that systems analysis is being performed at the wrong time. Instead of receiving meaningful design specs from a systems person, programmers must deduce requirements from cryptic notes (usually from e-mail messages, the back of envelopes or cocktail napkins). In other words, they are forced to do the systems design work backwards. Unfortunately, this approach leads to disjointed software that doesn't interface well with other programs.

This trend needs to be reversed. There should be at least a 2:1 ratio for systems engineers to programmers. The systems engineers should be viewed as the architects of the systems, laying out the blueprints for the carpenters (programmers) to implement. Better system designs (logical design) leads to better software specifications and physical design. Unfortunately, this type of person has historically been sacrificed by companies over the years in favor of hiring additional programmers. This leads us to the next sign...

• 85% OF THE TIME IN A DEVELOPMENT PROJECT IS SPENT IN PROGRAMMING

This is an environment where an inordinate emphasis is placed on software development (particularly coding and testing). I.T. management, under pressure to produce, forces a system through to programming before it is ready. Further, both I.T. management AND corporate management naively believe the development staff is not being productive unless they are programming. Consequently, the systems design work is sacrificed.

Naturally, the system, if it is ever finished, is fraught with errors and inconsistences requiring extensive maintenance. "Firefighting" thereby becomes the standard mode of operation.

• STILL USING THE CLASSICAL "5 STEP" APPROACH FOR SYSTEMS DEVELOPMENT

The same rudimentary approach taught to every student in college when they are learning programming (not systems) is still in vogue today:

I. Feasibility Study/Requirements Definition
II. Analysis/Design
III. Programming
IV. Testing/Implementation
V. Maintenance

This basic approach, which originated in the 1960's, is still prevalent today. It is often erroneously referred to as the "System Life Cycle." As we should all know by now, systems do not have a life cycle, only projects do. This is another example of the sloppy thinking persevering in our industry.

Admittedly, some form of methodology is better than none. Organization of any sort is better than sheer chaos. But when your methodology emphasizes software (as this one does) then true systems design work will be impaired.

Obviously, these "signs" are related to each other. Whenever you encounter them, you will probably find a company who has invested heavily in programming tools and training in the hopes a better mousetrap will somehow solve their problems. If a company has gotten too engrossed in their technology, they will inevitably lose sight of some very basic management concepts and systems principles.

(continued on page 3)

(continued on page 2)

IS THERE A WAY OUT?

Of course there is. But it requires a change of perspective and culture. First, you have to recognize that a problem exists. There is an old adage in psychology stating, *"You cannot treat a patient if he does not know he is sick."* If the corporate culture is such that no one perceives a problem, particularly management, it is highly unlikely anything will ever change. Only after a few significant snafus does management typically perk up and pay attention, such as:

- Excessive overhead due to an overstocked inventory.
- Production slowdowns due to insufficient supplies.
- Loss of customer data (particularly orders).
- Customer service is impaired and complaints consequently arise.
- Billings or statements to customers fall behind in delivery thus affecting cash-flow.
- Financial data is improperly calculated (such as payroll, accounts receivables or accounts payables).
- Data bases are corrupted with erroneous or outdated data making its validity highly suspect.
- Strategic systems are never brought in on time or schedule.
- Automation is viewed as unreliable and the company reverts back to tried and proven manual procedures.

To pacify angered executives, I.T. management has historically applied superficial remedies, such as the acquisition of additional tools which offer the promise of improved results. As any surgeon will tell you, do not try to apply a band-aid when a tourniquet is required to stop the bleeding. If you are content with treating symptoms, the more severe problems will probably be overlooked.

What is necessary is to step back and take a new perspective on the problem. This can be extremely difficult for some people to do. It is not unusual for people to lose their objectivity when they have become too intimate with a problem, particularly technicians who tend to recommend a solution that is technically elegant, but not practical to implement. You need people who can get up on the mountain and see the whole picture. This is one reason why many Japanese companies use end-users to assume the role of I.T. Director. With no computer biases, they are able to bring pragmatic solutions to their systems management problems.

WHAT'S IN A NAME?

Let's take it a step further; there is a lot of discussion about the job titles used by today's development personnel. In the old days, we simply had "Systems Analysts" and "Programmers." This eventually gave way to "Software Engineer" which hints at the change of focus mentioned above. Today, there is a realization there should be someone concerned with the overall architecture of the corporate systems, hence the use of the title "Systems Architect" representing the successor to "Systems Analyst." I don't have a problem with this off-hand but, instead, opt to use the title "Systems Engineer" based on our observations of the engineering/manufacturing principles needed to design a system. Sorry, "Analyst/Programmers" do not qualify as systems engineers; they are still programmers in sheep's clothing.

The difference between the use of the title "Engineer" and "Architect" is minuscule in comparison to the use of "Systems" and "Software." Whereas the former is an argument of semantics, there are significant differences with the latter.

The skills required to perform "Systems Engineering" are distinctly different than "Software Engineering." True, both agree there are principles involved derived from engineering, but each has a separate focus and orientation. Whereas the "Systems Engineer" requires a more extroverted personality with people and business oriented skills, the "Software Engineer" tends to be more inclined towards an introverted personality with technical skills. Interestingly, there are few people who can adequately perform both job functions adequately. One person may be good in one and lousy in the other. Frankly, sharpening the skills in one tends to lessen the skills of the other.

Most people entering the I.T. field usually start out on the software side, then graduate to systems. Inevitably, they still see things through the eyes of the programmer. Then, as they are promoted to manager, they still have a programming perspective. Universities are essentially no different; their curriculum begins with software and ends in systems. What kind of focus does the student have at the end of their college career? That the only legitimate problems worth solving are those that can be conquered by technology, not by systems.

CONCLUSION

Over the last thirty years I have observed the subtle shift of the industry from systems to software. For example:

- We no longer talk about "M.I.S." or "I.S." but, rather "I.T."

(continued on page 4)

(continued from page 3)

- Publications such as *"Infosystems"* and the *"EDP Analyzer"* have been superseded by computer hardware/software rags.

- Industry associations have changed their focus, the classic example being the *"Data Processing Management Association"* (DPMA) renaming itself the *"Association of Information Technology Professionals."* Other groups, such as the Association of Systems Management (ASM) have gone the way of the dodo bird.

- The change in job titles as mentioned above.

Most of this can be attributed to sloppy terminology in the industry and marketing glitz. Regardless, we now have an industry with a focus on technology where we try to implement the latest technological innovation to our businesses (this is what I call the "solution looking for a problem" phenomenon or "cart before the horse"). Instead, a systems perspective defines the problem, then seeks a suitable and cost effective technical implementation. One drives the other. Which do you embrace?

Just remember, it is all a matter of perspective.

END

"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.