**TITLE: "DIAGNOSING SYSTEM PROBLEMS"**

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"*

*"Without a road map, you might be driving in circles."*
*- Bryce's Law*

**INTRODUCTION**

Okay, you've run your program debugger repetitively and everything checks out fine. But for some unknown reason, the whole system is inoperable. Both the software and data base design looks fine, but you are going stark-raving mad trying to locate the problem. Have you considered that it might not be a flaw in the design of the software or data base at all? That perhaps the problem resides in the overall system architecture, or possibly its just you?

In many cases, diagnosing a problem is more painful than correcting it. Whereas I have reviewed basic testing principles in the past, see;

No. 41 - *"Testing 1, 2, 3..."* - Sept 12, 2005
http://www.phmainstreet.com/mba/ss050912.pdf

Here, I want to discuss some tips for diagnosing problems.

**THREE TIPS**

**1. Walk through the system and check the man/machine interfaces.**

Years ago, we were contracted by a large manufacturing company in the northeast who was having trouble implementing their new shop-floor control system. The system was state-of-the-art in terms of programming and DBMS technology. But they simply couldn't get it to work no matter what they tried. Frustrated, the company hired us to see if we could find the problem. Instead of studying source code, as the development staff had done, we began by mapping the overall system architecture.

I've described the "PRIDE" Standard System Structure Concept on more than one occasion, but in a nutshell, a system can be drawn as a four-tiered hierarchy representing a product structure. Whereas a product structure consists of four levels representing products, assemblies, subassemblies, and operations, "PRIDE" likewise decomposes the system into:

LEVEL 1 - SYSTEM
LEVEL 2 - SUB-SYSTEM (Business Processes)
LEVEL 3 - PROCEDURES (Administrative & Computer)
LEVEL 4 - OPERATIONAL STEPS (for Administrative Procedures) and PROGRAMS (for Computer Procedures)

This universally applicable approach for defining the system architecture makes a convenient road map for walking through all aspects of the system and validating its integrity. Such hierarchy diagrams can either be produced from IRM Repositories or from some simple graphic tools. In our consulting assignment though, we simply sketched it out using paper and pencil. Basically, we walked through the system, sampled work and looked for man/machine interfaces. Inevitably, we came upon a sub-system whereby the computer displayed errors in the shop-floor requiring attention by the foreman. The foreman was to take the corrective action and respond to the computer. There was only one problem with this: nobody had told the foreman about any of this. We then wrote a simple Administrative Procedure for the foreman who took the necessary actions and the system operated correctly thereafter ("miraculously" as our client said).

This brings up an important point: systems will fail more for the lack of administrative procedures than for well programmed computer procedures. Although the manufacturing company had produced some rather elegant software, they had completely overlooked the man/machine interface. Again, the "PRIDE" Standard System Structure Concept had provided the necessary road map, but because the client didn't appreciate the need for such a top-down blueprinting technique, they had no idea where everything was.

**2. Work backwards.**

When diagnosing business processes, procedures and programs, there is a natural inclination to go from start to end in diagnosing a problem. Sometimes you can find a hiccup using this approach, other times you cannot. Instead, try working backwards from end to start, from output to input. Again, map the design using a flowchart or some other graphical technique. If processing involves considerable decisions, draw a decision tree or table. Such graphics are invaluable for validating design logic.

*(continued on page 2)*

*(continued from page 1)*

**3. Have a second pair of eyes look over your work.**

As we become imbued in the mechanics of a design, too often the obvious becomes less obvious to us. Here, another set of eyes can readily see a problem we have overlooked. This is particularly beneficial in shops operating in accordance with certain design standards. Uniform design practices makes it easier to spot anomalies than without such standards.

Where the second person comes from is also important. If the person comes from your work group and is familiar with your style of design, he/she may very well be able to spot a problem. Then again, maybe not. Perhaps the problem will be invisible to them as well. In this case, you might want to consult a neutral third person with a fresh perspective on the problem. This can either be a person from within the company or possibly an outside consultant.

**CONCLUSION**

Graphic aids, such as flowcharts and diagrams, are helpful for diagnosing a problem but also remember to challenge the graphic. Its not uncommon for graphics not to match what is happening in fact. A good IRM Repository is also invaluable for substantiating designs. The design is either properly recorded in the IRM Repository or it is not. Further, such a tool provides the means to study the relationship of information resources (aka "impact analysis") which may reveal unknown components affecting a design.

More importantly, the idea of maintaining a system architecture (as implemented by the "PRIDE" Standard System Structure Concept) provides the needed road map to find your way through a system regardless of its complexity. Many programmers view such charts as frivolous primarily because they are only concerned with their small piece of the puzzle and are unconcerned about the total picture. But for those of you who need to see the total picture, the system architecture is the logical first step for diagnosing problems.

For more information on the "PRIDE" Standard System Structure Concept, see:
http://www.phmainstreet.com/mba/pride/is.htm

**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

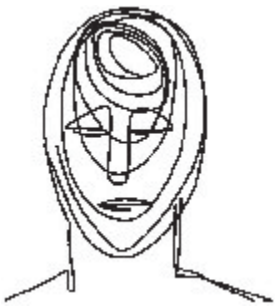
*Also, be sure to read Tim's Blog at:*

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"*