THE Review

August/September 1991 Volume VII Issue 4 A Publication of the Association of Human Resource Systems Professionals

Q and A: Everything You Wanted To Know, But . .

By Naomi L. Bloom

The following is excerpted from the session "Q & A: Everything You Wanted to Know, But . . ." led by Naomi L. Bloom on Tuesday, April 30, 1991, at the HRSP Annual Conference in Chicago.

Bloom, a well-known HRMS consultant and a frequent speaker before HR systems groups, responded to questions from the audience. The following three questions and answers are illustrative of the lively discussion that took place. An audio tape of the entire session is available.

[redacted] How will HR software vendors truly integrate the various application modules, e.g., payroll, HR, benefits, applicants, FLEX, etc., in a meaningful way?

First, what do we mean by integration? Every vendor gets zillions of RFI's, not to mention RFP's and other questionnaires that say: "We want an integrated system." Integration is good. We don't know what it is, but we all want it.

Here's my proposed definition for what we mean by integration. A system is not integrated unless it has the following three characteristics. First, an integrated system must have a common user interface. What does that mean? It means every interaction that the human being, whoever that person is, has with the system, has a common look and feel. If I learn the dialogue once, it works every time. The same commands mean the same thing. Everything works exactly the same way. Furthermore, there is a common point of entry. The cosmic main menu. I don't care if it's a GUI or command format, there's only one cosmic main menu.

Every vendor gets zillions of RFI's, not to mention RFP's and other questionnaires that say: "We want an integrated system." We don't know what it is, but we all want it.

The second aspect of integration is a shared, non-redundant database. Shared

means that all applications work off exactly the same database. We don't have separate files; we don't extract for this or that or the other thing. Non-redundant comes about because you have built a logical data model and - except for the tuning required to achieve necessary performance, i.e., the deliberate denormalization to achieve performance - there is no redundant data. None. And there are fully normalized attributes if you've really worked from a logical data model – a bonus!

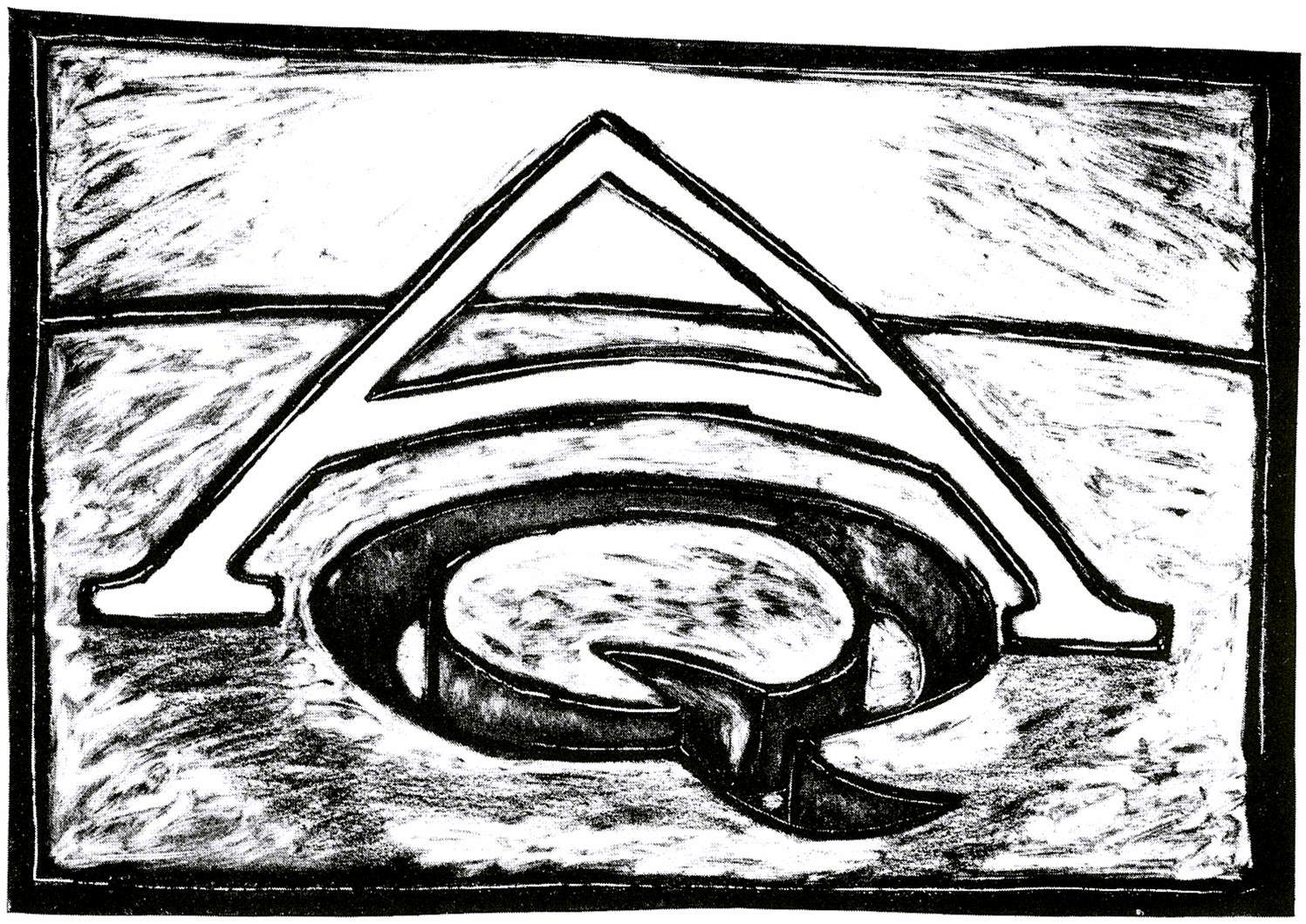
The third aspect of integration is the most subtle point. It cannot be an integrated system unless all components are based on the same design principles. We are sharing common modules. We have standard coding procedures. To a programmer, it looks the same.

Now, let me ask the $64,000 question. Does anybody know of an HRM software package that meets these criteria? None of them do? Why? Because it's a goal. And I just made it up. And the vendors haven't had time to catch up. They're in this room, and they're going to write it down, then go off and make one.

If you say that you want an integrated system, the vendors say they've got one. If you don't know what you mean by integration, then how can you hold them accountable for reading your mind and answering the question without knowing what you're asking?

How will the vendors truly integrate, given our definition of integration? It's not going to be easy. Some of the vendors will have to start over from scratch. And some of them are. Some of them have. Others are going to take some aspect of their system which they believe has the right characteristics and redo the other parts to conform. This does not come cheap. The vendors will not do this unless there is a market for it. If they think nobody knows what integration means, they are not going to spend this year and next year and the following year's profits achieving this.

[redacted] When is it appropriate to use a navigational database management system (DBMS)? And when is it appropriate to use a relational DBMS? And why do we care? Or do we care? Can we use our logical data model with either?



In the world of DBMS, there are only two kinds - navigational and relational. Navigational come in flavors like IMS, which is hierarchical; IDMS (I believe) is networked. In every navigational DBMS, your path through the data must be navigated via pointers that are predefined. That means that you have to know upfront, as part of the design, the expected paths through the data.

Maybe we should spend twice as much on computers. They are getting cheaper. Are people getting cheaper?

Navigational databases are usually the most efficient. All other things being equal, such as hardware, etc., they are the most efficient when you are processing along those predetermined paths. They are absolutely a nightmare if you leave the path.

In a relational DBMS, paths are not predefined. The data carries with it the ability to navigate. A relational DBMS is absolutely the best thing going if you often follow unpredicted paths through the data.

Payroll is inherently navigational. That's the way it is. That's the way it works. There is no reason whatsoever, if you're just doing a payroll system, not to use one of those great navigational databases.

But what about HRM? It's inherently relational. That is, the business is inherently relational. Tell me all of the employees who are likely to retire in the next five years? Tell me all of the employees likely to retire in the next five years that we would like to get rid of sooner because they're deadwood? Those are the kinds of questions we are dealing with in HRM. You certainly can't predict all of

them today and build software around these predictions that will stay stable for more than three minutes.

It is my personal opinion that you cannot have an effective human resource management system (HRMS) unless it is based on a relational DBMS. Even though relational databases can be very inefficient - especially for high volume batch processing - their advantages far outweigh this one disadvantage for most HRM applications.

By the way, every one of the major, non-relational DBMS products has worked hard at overcoming this inherent difficulty. They have good inquiry tools. They have all kinds of techniques for making HRM-type (often called browsing) inquiries more efficient. Nonetheless, the mapping from our business into a navigational DBMS requires that we decide upfront the most likely-to-be traversed path. And we have to live with it.

When confronted with the DBMS challenge, I say it should be relational. It is going to cost more in computing resources, but look at where we are going to save. Maybe we should spend twice as much on computers. They are getting cheaper. Are people getting cheaper? Has a computer ever filed a sexual harassment suit?

If you have an application that is inherently navigational, then use a navigational DBMS. Good examples are general ledger, inventory management, and payroll. The HRM business is inherently relational. When you look at the data model of HRM and you look at the many-to-many relationships that we have, it is clear that only a relational DBMS can really support HRM.

How do you know when your system has outlived its usefulness? Once you make that decision, do you kill it or do you try to revitalize it?

There are some things that seem to signal that it is time for the thing to die. I will tell you my personal list. Kill it if it is still using cards. (Don't laugh. You'd be amazed.) If it is still using tape master files. If the thing was written in spaghetti code twenty years ago and Betty, the original programmer, is now 84 years old. There are visible signals when it's time for the thing to die. All the bright, young people that we are recruiting into our IS organizations look at it and say, "I'm not touching that." That's the easy part. It's easy to recognize one that should be killed off.

The harder thing is to recognize when we have spent a lot of money and did not get the value added to the business that we should have had.

The harder thing is to recognize when we have spent a lot of money and did not get the value added to the business that we should have had. We are not sure whether we should kill it off, start over, or keep throwing money down the rabbit hole.

Let me give you a classic example. You did a user needs assessment by interviewing at least 200 people, keeping meticulous notes on every conversation. Then you produced this giant volume that eliminated duplicate points, but otherwise was an unsynthesized jumble. The user wish list.

Then you bound this and shipped it to every software vendor on the planet, most of whom were smart enough not to respond. Some of them wrote back a three page response, you know, with 400 pages of marketing literature. And they said, "We can do it all. We will meet your needs."

Then we went out and did due diligence. We took two people from the HR department who were known to be incompetent and therefore available for our project. We took Betty, remember Betty the payroll lady who's 84? We took the most junior person from the IS department, the most junior programmer/analyst. He'd never even seen an HR software package. Off they went.

They visited each vendor. They listened to marketing presentations. They checked references. They didn't know what to ask, but they called everybody. They made a recommendation. We bought the software. Half a million dollars.

Now the software arrives, and the vendor person shows up to install it. The data center says, "It's CICS release what?" So we have a little problem, but it will pass because the vendor is going to upgrade the software any minute.

Then the HR executive, who was only vaguely aware of this project or vaguely aware of anything, gets fired. We have decided we are serious now about Human Resources, and we bring in a dynamo. She's going to take the place apart. No more wimpy HR executives here! "And at my last company, we used the XYZ package. What do you mean you didn't buy that one?"

The bottom line is, we're holding a leaking bag. Everybody agreed the old system should be put out of its misery. Right? No, nobody ever really agreed. The payroll people love the old one. They have this super zapper transaction so that, when all else fails, they put it in and just overlay the offending data. They're happy as clams. They never wanted a new system.

It all comes out now. The payroll people never wanted the new system. The compensation manager never understood that the new system would mean we would have edits on compensation. So those special deals we have cooked-up that nobody knows about are now going to be in a database. When one thing goes wrong, when something changes the mix.

what looked like a done deal starts unraveling.

At that point, what do you do? I say, start over. Now, I don't mean you throw out every piece of paper that has ever been collected on the project and forget everything you know. I say, you go back to square one, lay out a responsible approach to reaching a consensus and making a decision. And then use everything that's already been done to contribute to that.

Then you have the third case. We've got a system. It may not be integrated, but we've got a lot of stuff automated. We send extracts to all the divisions and they've built up subsystems. The core software, whether we built it ourselves or we bought it, has only been installed for eight or ten years. Of course, it was designed ten years before that, but *we've* only had it for eight or ten years. It cost a lot of money to implement and we've been building on it for years. Then the new HRMS manager says, "I know there's a better way."

But nobody wants to listen. Because you're really okay, not great, but okay. We don't have an order entry system, and we don't have a financial management system, so there's really bigger fish to fry.

That, to me, is the most challenging situation. That calls for greater shrewdness than either of the other two. Because it calls for someone to really understand the business. Not the basic stuff, like the fact that we have to keep track of employee records and a little applicant tracking, and a little 401K stuff. But the serious stuff. Like why are employees leaving us? And why are we constantly hiring and laying off? Why don't we have a good fit between the people we have and the work that is going to be done?

The real question becomes, "How can I add value to the stuff I've already got without spending tons of money?" Most firms use a process called systems audit ing to revitalize existing systems. We do this in our personal lives. I mean, don't we periodically look in the mirror and say, "My hair's turning gray. I don't think I can cope with this. Maybe a little rinse would do it." I don't chop off my head.

It's the same analytical process. It's a lot easier to start over and make a big bang. It's a lot easier to yell, put the thing out of its misery and make a big bang. It is really quite difficult and takes a good bit of sophistication to know the business you're serving well enough to say, "for just a little tweek, for just a small investment, for just a Saturday of my time, I could really add value." If you do that for three or four years, somebody's going to notice. And then when you announce that it is time for replacement, you have a lot of credibility with which to start over.

ABOUT THE AUTHOR

Naomi Bloom has nearly 25 years experience in strategic planning and design, development, implementation, audit, and support of financial, data analysis, administrative and, especially, human resource management systems. Her experience includes resolving the organizational, functional, technical, and project management issues related to these systems. She is also trained in the use of state-of-the-art life cycle productivity tools and techniques, including CASE concepts and principles. Using her formal systems planning methodology, Bloom leads corporate clients from strategic systems planning for HRM through the life cycles of recommended projects. Bloom, who holds an MBA in finance and accounting systems from Boston University, can be seen nationally on the PBS television course, "The New Literacy: An Introduction to Computers." ■