Information resource planning and management methodologies

by KEITH GREYSTOKE

Database Consultants Europe B.V.
Amsterdam, The Netherlands

ABSTRACT

During the last ten years it has become fashionable to create structured methodologies which, in theory, are totally removed from current state-of-the-art technology. This however, is a contradiction in terms since analytical methods are only created in the computer world to solve state-of-the-art shortcomings. Thus before any analytical methodology is selected first question is knowing where your company lies with regard to the state-of-the-art technology. This paper attempts to show the evolution through the previous stages. It would be erroneous to think that the methods are driving the technology, as opposed to the other way around. This would be the case if the premise of producing methodologies in blind ignorance of the state-of-the-art were true.
This paper discusses the evaluation of analytical methods and the reasons for their development—dating from the early concept of database technology, where this technology is seen as the recognition for a shared data environment.

To understand the entire analytical process that is necessary in order to make information systems work, we must consider together three factors, which in theory we would like to separate, but which in practice are so closely linked that it is of no real use to consider them in total isolation. These three factors are

1. The problem of defining the exact requirements of the information-handling systems to be installed and reconciling those requirements among the various users of data
2. The state-of-the-art software and hardware that is available
3. The analytical tools and methods that can be used as tools to achieve specific solutions to problems within any given company

To understand our current thinking one must examine the historical evolution that has led us to where we now stand. In the 1960s we built computer systems, application-by-application, by including both the logic and the handling of the data into the programs themselves and as a consequence, converted the raw elements of data into information. Since the words “data” and “information” have distinct meanings, let us examine the difference. Data are raw elements. For example, a number—12709—in itself not very meaningful, particularly since it would be necessary to determine what that number actually meant. Let us suppose that this number were an order number. We may still consider it to be raw data, and on its own, not particularly useful. When combined with other data and presented to the user it becomes information. Thus, a combination of the order number, plus a product number, plus a quantity asked for, plus a price, would provide information possibly to a salesman. All users within a company combine different elements of data to create information as they see it.

If we consider the original systems of the sixties, we can see that programs were written based on users’ requirements, piecing together various elements of data to provide information to the user. Each program would have the responsibility of manipulating, changing, updating, and generally caring for the requirements that the user had. The data were turned into records and kept inside the machine in the same form that one would keep information in a filing cabinet. As a consequence, “filing cabinets” sprung up all over the machine. These were kept on disks or tapes, since the core of the machine was not large enough to store all of this information.

Originally, it was necessary for each program to code its own access methods to get hold of the information that it had put on the disks. However, it was quickly recognized that it would be much easier if the hardware manufacturers were to provide the access methods required themselves, as a service to programmers, since this was a complex procedure requiring intimate knowledge of the hardware, channel commands, and the like. Nevertheless, the control over the information itself lay entirely within the program. As situations arose within the company where the structuring of the data needed to be changed, e.g., an element such as a delivery number might be added, each program using information made up of data from the order, which now contained a delivery number, had to be changed. Needless to say, this was a long and laborious process requiring a great deal of work. The tools available for analytical purposes as a consequence of our above-stated working methods were limited to the processing requirement—tools such as flow charts.

In the sixties certain common problems began to appear throughout companies using these conventional methods of working. These problems were clearly identifiable and the reasons behind their occurrences were also identifiable. The first problem was that duplication of data was paramount. This was obviously occurring since many users’ requirements for information used common elements of the data and some users even created, or had created for them, systems that used exactly the same elements of data but in different sequences using different keys.

For example, a salesman might keep a file of information concerning clients using the client name as the key for entry into the data, and an entirely separate file by the name of the contact he had as the key within the company he was dealing with, and yet a third file with exactly the same data elements using the address as the key. As a result of the duplication, which in itself cost both time and manpower resources to keep updated, incompatibility started to creep in. For example, a file may change because a certain client changed his address and had informed the salesman, but the salesman had not informed the accounts department. Thus the files of the accounts department continued to contain the old address, although the salesman’s data were correct. The next problem was that control became dissipated and virtually impossible since each user of the data was a law unto himself. Likewise, the problem of security. It is clear that it is far more difficult to protect secrets within the company when there are 50 or 100 copies of the same data, rather than the existence of only one copy.

The next problem was that the effort required to maintain systems using changing data, where those data were being used by many and being handled by the program itself, meant excessive maintenance when changes occurred. This problem
was exacerbated by the fact that programmers and analysts were all left to work in their own manner with no common standards being applied from one to the other. Thus, not only was the maintenance excessive but more complex once the analyst or the author of the program had left the company.

The next problem concerns the data dictionary facilities—at that time hand-driven—representing an encyclopedia of information about the programs and the data. Each separate company clearly had such a system, even if it were only in the form of listings (which it usually was) spread around the company. Thus, there was no easy way for people to derive common terminology, find the basis of the work produced by others, or find where the data elements were being used to derive the required information.

Another problem lay in the fact that the data were structured solely for the applications that used them; in general this was unsatisfactory for other users. Even if an attempt were made to structure data over a greater variety of applications, the software facilities for achieving such desires were not available. The result of these problems was a general mixture of unprofessional implementations that proved difficult to use.

In the late sixties all the factors mentioned above caused industry to turn its attention to viewing data in a totally different manner—to removing control and structuring capabilities away from the programmer and to creating a records department on the machine itself. With respect to the analytical tools, great changes took place. Because the desire to achieve the results required a clearer view of the data—separate from the processes—in order to solve not so much the technical problems but the inherent business problems of attempting to share this data among a wider variety of users, questions needed to be answered: Who owns the data? Who can see the data? Whose data are correct when inconsistencies are found? Whose coding systems are acceptable where several exist for the same items? How would controls be implemented in terms of availability, of security, of privacy, of ease of access, of reliability, of back-up, of recovery?

Thus the industry turned its attention to analyzing the data in their own right, pulling out the elements that became known as entities, which were chunks of data representing real-world items that existed within the company, such as employee, building, order, car, invoice, etc., and attempting to view the inherent business relationships that existed between these various pieces of data. In theory, we need have looked no further, since the overall picture of the data was that which we wished to incorporate on the machine. Unfortunately the manufacturers imposed structuring rules upon us from the software, which was provided to handle the data. Not all of these relationships could be made optimal.

It was necessary to question which relationships were more important than others, which were used more frequently. The only way to gain these answers was to question the actual usage of the data in terms of the information that was required by the applications that used the data. How frequently were activities carried out? Within those activities, which data were required? Which activities took priority over others? Thus the analytical methods broke into two distinct parts—the first attempting to view the entities and their inherent relationships that existed through the eyes of the business, and the second the processes that acted on those data.

The clear view was to take a picture of the entire company’s data so that the database, or records department, would provide information for the entire company—a logical argument based on the assumption that the sharing of data and information was a good thing and could be argued strenuously. It is clear that this argument was good for the following reasons. First, the data were the only resources available within a corporation that clearly could not be replaced. There were banks to meet shortfalls of cash. There were other buildings if the current ones proved unsatisfactory. There were production lines to add stock if that were necessary. There was new equipment available if the old equipment wore out. There were job-seekers to replace those who left.

But what would a company do if all its information was suddenly lost? Clearly it would have to close the doors, it could no longer function. And yet if one were to ask any question about the other resources; about such things as how much money is available; what buildings are ours; or how much stock is available, one could get answers to the very last detail. In the case of data and information, however, if one were to ask the average company how much it cost to collect the information they now have; how much they spend on maintaining the information they have obtained; or where one would find replacements; no satisfactory answer would emerge. This is an interesting factor considering that information is the only resource that is irreplaceable. Nevertheless it turned out that both the analytical methods and the software provided were incapable of coping with the entirety of a major corporation’s data resources. This rendered incorrect the first assumption that we should share the data throughout entire companies.

The second assumption that was made, and for which traditional databases were produced, was that data would be centrally handled on very large mainframe machines. This was a logical assumption in the late sixties and early seventies because that was precisely the direction of the hardware manufacturers. However, as in the first case, this assumption proved to be incorrect. From the manufacturers’ viewpoint, machines became smaller and smaller and more and more powerful, while the manpower investments in programming became greater and greater. This happened so much that in today’s world every major department within a given company can afford to acquire its own very powerful microcomputer costing no more than $7,000 or $10,000. Thus, these departments declined to use the central service and started to build their own systems using their own data.

The third assumption made by the software manufacturers was that the relationships that existed between data were in practice, static, whereas the data themselves were volatile. For example, if there was a relationship existing between an employee and an address, when the employee moved, the data—the address—changed. If we consider an exact example, supposing Mr. Vemer currently lives in Amsterdam and then moves to Utrecht, we cross out Amsterdam and change the data to read Utrecht. This is a subtle misconception. Just because Mr. Vemer leaves Amsterdam, does not mean that Amsterdam disappears. What really happened was that the
relationship that existed between Mr. Vemer and Amsterdam changed and the relationship now became between Mr. Vemer and Utrecht. Because the assumption that data, as opposed to the relationships, were volatile, it was concluded by the majority of software manufacturers that it was not a problem to incorporate physical pointers within the data; a subtle error that rendered most packages very inflexible. Of course, they were flexible in their ability to change the data by adding records, adding fields, and changing fields, because the assumption was that the data were volatile; but they proved very inflexible when it was required to change the relationships. So much so that several such changes would often imply a complete redesign.

The next assumption was that the encyclopedia-type requirement of meta-data provided by the data dictionary would require only passive assistance. This means that it would be used as a “look-up” mechanism in much the same way as a telephone directory would be used for people wishing to find information about telephone numbers. This assumption also proved to be incorrect because the majority of users of the data had difficulty in predetermining the relationships that they would use and the pieces of data that would be required to view their information. Thus, in our telephone book example, we may not know the name of the company but only its address—an impossible situation if you wish to look up a telephone number. In reality, it is necessary to be able to go to the dictionary, ask the question, and receive the answer—at the time you wish to pose the question. Would it not be easier if we had a telephone book or a service we could call and only decide at the time of the call which pieces and relationships we wished to use—and still be able to acquire our view of the information as we wished it to be determined?

The next assumption was that each user of the data must have his own language in order to communicate with the database management system. Thus different languages were invented—a data manipulation language for the programmers, a data definition language for the designers, and a query language for the users. Once again, if we were all talking about the same data, this is a totally incorrect assumption because if we are to share things, we must all communicate in the same language.

We can therefore conclude that almost all predetermined assumptions made by the industry for its requirements in the use of database management systems were wrong. This has an impressive consistency—five assumptions and a 100% record of failure. As a result of these assumptions being incorrect, certain problems began to appear generally throughout all companies. Rather than go into them in detail, I shall simply list them.

1. Duplication of data
2. Inconsistent data
3. Loss of central control
4. Excessive maintenance
5. Lack of standards
6. Unprofessional implementation
7. Duplication of effort

Ironically enough, these are exactly the same problems that we had set out to solve initially. Clearly, the analytical tools would not on their own solve these problems, although a recognition began to emerge. It was obvious that the data must be viewed quite separately from the processes that acted on them. In theory, data-modeling techniques were enough to build a database, but the state of the art did not permit us to optimize every relationship that existed between the elements of data. Thus the input from the analytical tool on examining the activities separately was required in order for us to produce a physical design. This single factor backs up the argument that the analytical tools cannot, in practice, be separated from the state-of-the-art technology that exists. Further, tools were required to bring together the modeling techniques with the activities that acted on and used the data to create information. It also became clear that there was another requirement—a clear statement as to why the analysis was being performed in the first instance, since analysis on its own will solve no problems if it turns out that the problem is not one of computer inefficiency or misuse of data. For example, suppose a company were to decide that because its bills were not being paid quickly enough, a new computer system was necessary. Upon further examination it was determined that the bills must first be presented to the accounts department, then sent to the heads of departments incurring the bills, then sent for approval to the persons who incurred them, then sent back to the heads of departments, then returned to the accounts department, and then sent to the payables department. If this cycle took 40 days, no computer system in the world would improve the situation. Consequently, it was recognized that a clear statement must be made as to the intention and purpose of the new system and for the data and analytical tools to be used.